TABLE OF CONTENTS

Chapter 1
Wiring Considerations

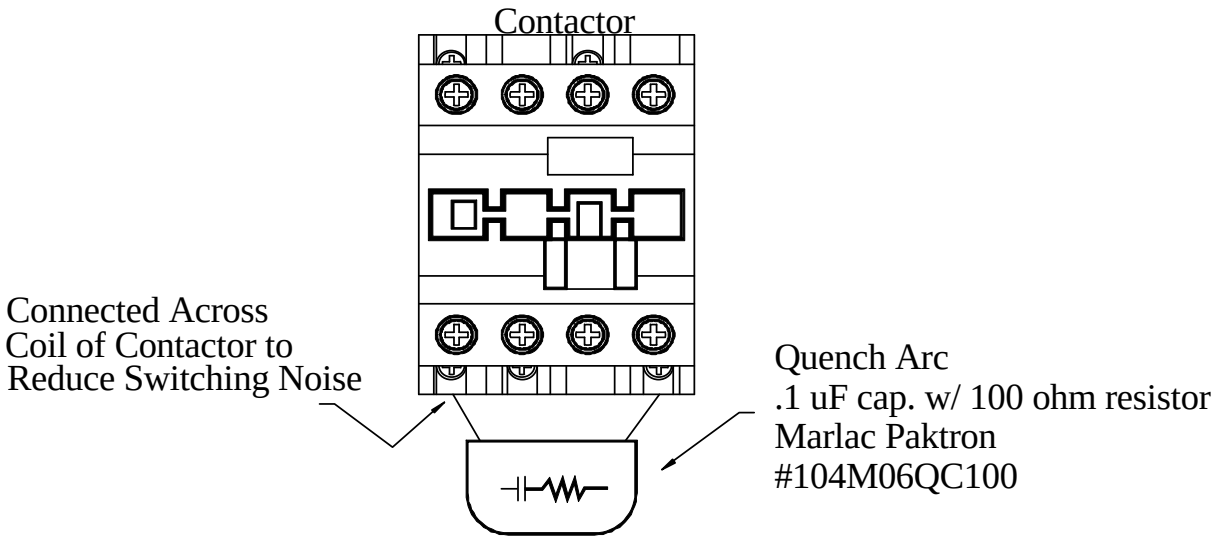
Routing and terminating wires and cables properly is very important to the reliable operation of your equipment. Taking the time to do the jobs correctly, with the proper tools, will save time and aggravation in the future.

Separation of High Voltage AC and Low Voltage DC Signal Wires

Whenever possible wires and cables that carry High voltage AC (110 Vac, 230 Vac single and 3 phase) should NOT run beside low voltage DC (5 Vdc, 12 Vdc, 24 Vdc), signal wires and cables. Electromagnetic noise can be induced in a DC signal wire if an AC line is running parallel to it. AC and DC wires can cross at right angles with no noise problems. Using wire tracking to hold the cables and wires in place makes the layout and wiring look better, and can help prevent noise. When used properly, the AC and DC signal wires should not be run in the same wire track. Also, the wiring inside of the track must be kept neat. Excess wire and cable length should be trimmed NOT crumbled up and stuffed into the wire track then hidden by the track cover. Locate the wire track to leave sufficient room to connect wires to the components.

Quench arcs/Snubbers for all contactors

All contactor MUST have a Quench arc/Snubber installed across the contactor coil. A Quench arc/Snubber is a capacitor and a resister in series. (100 µf, 600 V and100 Ω) This absorbs the noise spike that results when a contactor is activated.



Proper Grounding Procedures

All AC power cables should have a ground wire. This can be identified by either green insulation, or yellow insulation with a green stripe. These wires must be connected at both ends, the frame ground in the magnetics, or control cabinet, and the correct ground location for the device that they control, motor, solenoid, etc. Because of paint, oil, etc., just bolting a motor or switch body to the machine frame may not actually ground it. A good example is a mill table. The table actually rides on a thin film of oil and does not make physical contact with the mill frame.

All signal cables going from the magnetics, or control cabinet to the machine must be shielded cables. Shielded cables have either a braided wire jacket or a foil wrap with a drain wire under the outer insulating jacket. This helps protect the wires inside of the cable from outside electromagnetic noise. The shields on the cable MUST be grounded at both ends. If the shields on the cables are not connected, they can cause noise on the signal wires contained within the bundle by acting as an antenna. There are exceptions to this rule however, specifically when dealing with motors.

Wire Sizes

All wires and cables are defined by the size of the wire and the dielectric strength of the insulating material. The size of the wire determines the amount of current the wire can handle before overheating. Overheating usually results in the wire or cable being destroyed. The dielectric strength of the insulating material is equally important. If a wire or cable has a dielectric rating below the voltage applied, it is possible that the applied voltage could arc through the insulation and destroy the wire or cable, or worse, damage the machine and the controller. Below is a quick reference chart to help determine the proper size wire or cable to use.

Wire sizes

WIRE SIZE	MAX CURRENT	NORMAL USE
22 AWG	7A	DC SIGNALS
20 AWG	10A	DC SIGNALS
18 AWG	13A	AC AND DC POWER SUPPLYS SOLENOIDS COILS
16 AWG	18A	CONTACTOR COILS
14 AWG	25A	AUX. MOTORS
12 AWG	30A	INCOMING AC, SPINDLE MOTORS
10 AWG	40A	INCOMING AC, SPINDLE MOTORS
8 AWG	60A	INCOMING AC, SPINDLE MOTORS
6 AWG	80A	INCOMING AC, SPINDLE MOTORS

- All wires should be sized for the maximum amount of current they will carry.
- The insulation on the wires and cables must be rated for the amount of voltage they will carry.

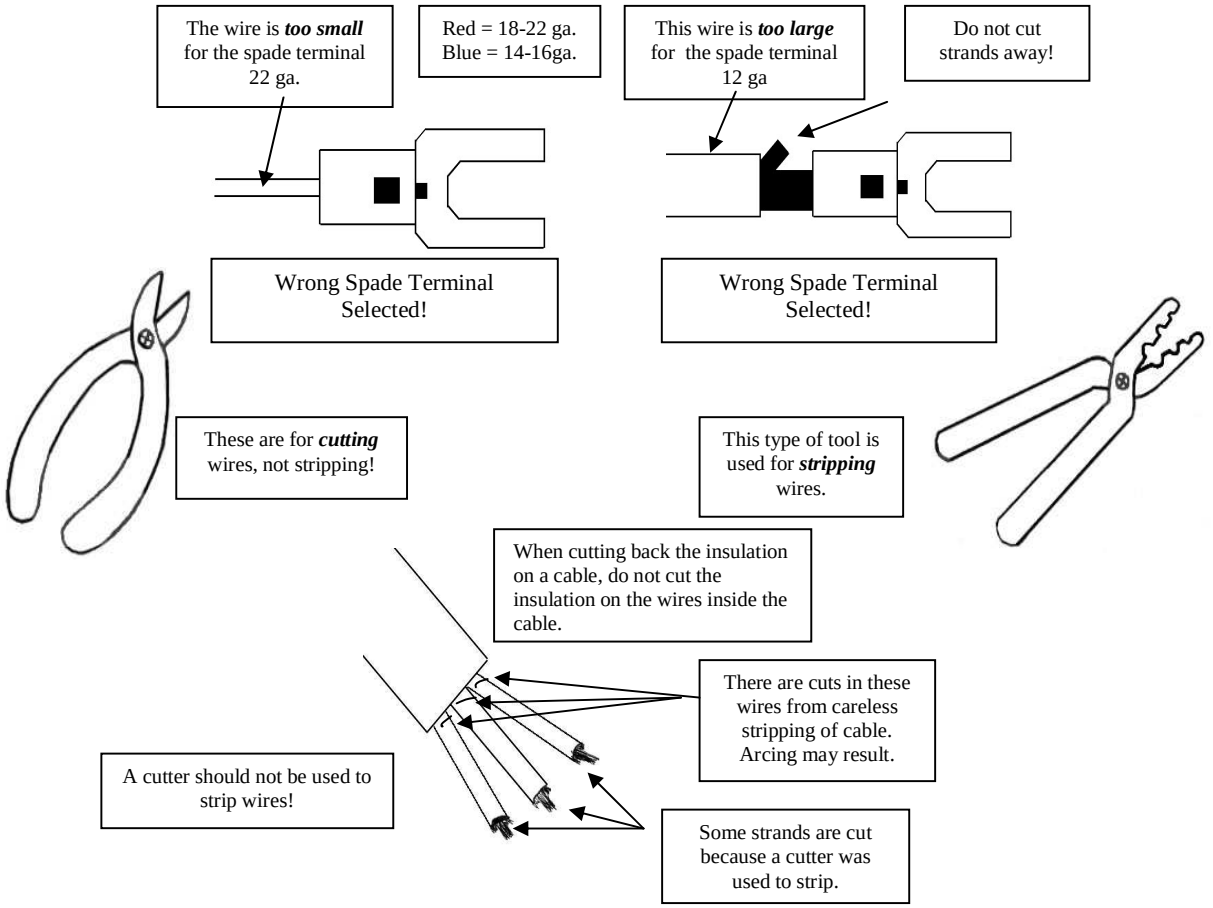
Use of crimp on terminals

The use of crimp terminals is very convenient since they don't require soldering. However, if crimp terminals are not used properly a great amount of time can be spent troubleshooting a problem that can be easily avoided. Crimp terminals are divided into sizes and color codes determined by the size, or sizes, of wire the terminal can accommodate. Proper crimp tools are also color coded to show the user which crimp station to utilize when performing a crimp. The preferred crimp tool is the ratcheting crimp tool. This tool will not allow the operator to under or over crimp the terminal. An under crimped terminal can cause an intermittent problem that can be very difficult to find. An over crimped terminal will shorten the life of the connection and lead to a possible reduction in current handling, and a possible break in the connection that can also be very difficult to find.

Use of crimp on terminals

1. The proper crimping tool MUST be used on crimp terminals
2. Use of the incorrect crimping tool or even the proper tool in poor condition will cause problems.
3. The wire to be inserted in the terminal must be stripped to the correct length.
4. The correct size wire strippers MUST be used so that no wire stands are cut off.
5. Care must be taken that the terminal is crimped onto the wire and not the insulation.
6. The proper size terminal MUST also be used.

The diagrams below illustrate several examples of what **NOT** to do:



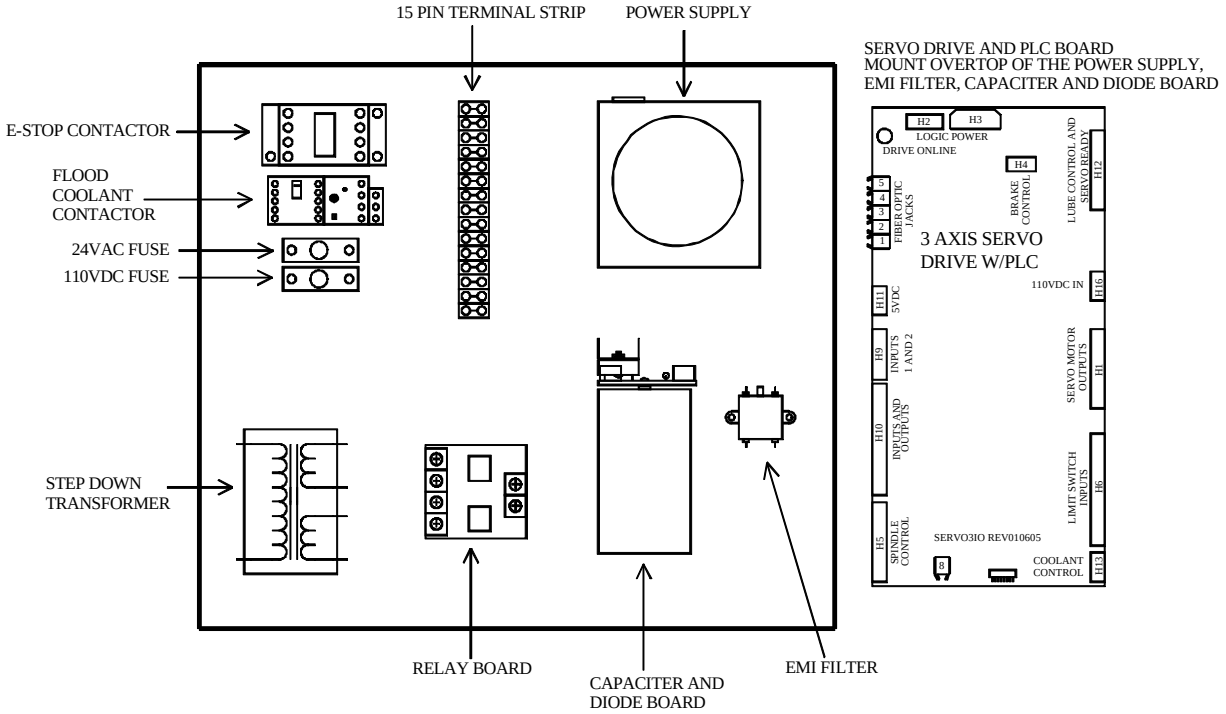
Hardware Installation

Panel/Cabinet Guidelines

When laying out the electrical components on a panel or in a cabinet, the following guidelines should be used.

1. Locate components so that high power AC wires and low power DC signal wires are not run next to each other. See section above on Wiring Considerations for additional information.
2. Leave enough room around components to easily run wires.
3. Locate components to ease troubleshooting.
   - Make sure Contactors can be reached easily.
   - Make sure LEDs can be seen clearly.
   - Make sure fuses are easily accessible.
4. Keep High Power terminals away from components that will be accessed the most.
5. Allow room for the cooling of components.

See the diagram below for an example of a well laid out electrical panel. Notice how all of the guidelines listed above have been adhered to.



Cabinet Size

We recommend an enclosure measuring at least 24” x 36” x 12” in order to properly contain all of the electrical components and the PC. If you are planning to mount the inverter inside the cabinet as well, a larger cabinet will be required.

Selection of Electrical Components (Board Level)

The following list of electrical components is necessary for the proper installation of the control system. The components shown in bold at the bottom of the list are optional.

1. ALL systems
   - Rotary interlocking door disconnect.
   - Fuses or breakers for incoming power.
   - E-stop contactor with quench arc snubber.
   - Servo PC with CPU10
   - PLC (see 2 - 6 below)
   - Servo drive. (see 6-11 below)
   - Servo motors with encoders and cables.
   - **Console/Jog pendant.**
   - **Terminal blocks.**
   - **Contactors with quench arcs snubbers for 3 phase motor control.**
2. RTK3
   - 110 Vac in.
   - 24 Vac in.
   - 220 Vac Three-Phase / 110 Vac Single Phase
   - 3 fiber optic cables.
3. PLCIO2
   - +5, +12 and –12 Vdc power supply.
   - +5, +12 or +24 Vdc power supply for input power.
   - 3 fiber optic cables.
4. PLC15-15
   - 24 Vac in.
   - +5 Vdc power supply for input power.
   - 3 fiber optic cables.
   - **Fuses for outputs.**
   - **Spinover board for spindle speed control.**
   - **+12 and –12 Vdc power supply for Spinover board.**
   - **PWM cable from PLC15-15 to Spinover board.**
5. SPIN232
   - DB9TXF board.
   - 1 fiber optic cable.
   - +5, +12 and –12 Vdc power supply.
6. DC3IO
   - +5 and +12 Vdc power supply.
   - 110 Vdc power supply for servo drive motor power.
   - 5 fiber optic cables.
   - **Relay boards for outputs.**
   - **Fuses for outputs.**
7. DCSINGLE
   - +5 and +12 Vdc power supply.
   - 110 Vdc power supply for servo drive motor power.
   - 5/2 fiber optic cables.
8. SD3
   - 310 Vdc power supply for servo drive bias and motor power.
   - 4 fiber optic cables.

9. SD1
   - 310 Vdc power supply for servo drives bias ad motor power.
   - 4 fiber optic cables.
10. SERVO1
    - 110 Vac input.
    - 110 Vdc power supply for servo drive motor power.
    - 2 fiber optic cables.
    - **Cables for drive fault and limit connection to the PLC.**
11. QUADDRIVE
    - 110 Vac input.
    - 110 Vdc power supply for servo drive motor power.
    - 2 fiber optic cables.
    - **Cables for drive fault and limit connection to the PLC.**
12. OPTIC1
    - +5, +12 and –12 Vdc power supply.
    - 2 fiber optic cables.
13. 60 CYCLE LOW POWER DC SERVO DRIVE POWER SUPPLY
    - Transformer with 83 Vac secondary.
    - Full wave bridge rectifier.
    - 15 amp inrush limiter.
    - 12000 µf 160 V capacitor.
    - 6.8 kΩ 5 W bleed down resister.
14. 60 CYCLE HIGH POWER DC SERVO DRIVE POWER SUPPLY
    - Transformer, 200 Vac with center tap secondary.
    - Dual stud diode assembly.
    - 15 amp inrush limiter.
    - 12000 µf 160 V capacitor.
    - 6.8 kΩ 5 W bleed down resister.
15. 50 CYCLE HIGH POWER DC SERVO DRIVE POWER SUPPLY
    - 3 phase transformer, 83 Vac secondary.
    - 3 phase bridge diode 50 amp 800 V.
    - 15 amp inrush limiter.
    - 12000 µf 160 V capacitor.
    - 6.8 kΩ 5 W bleed down resister.
16. 230VAC 50/60 CYCLE AC SERVO DRIVE POWER SUPPLY
    - 3 phase bridge diode 50 amp 800 V.
    - 25 amp inrush limiter.
    - 1000 µf 450 V capacitor.
    - 50 kΩ 5 W bleed down resister.

17. 380/400/415/460VAC 50/60 CYCLE AC SERVO DRIVE POWER SUPPLY
    - 3 phase transformer, 230 Vac secondary.
    - 3 phase bridge diode 50 amp 800 V.
    - 25 amp inrush limiter.
    - 1000 µf 450 V capacitor.
    - 50 kΩ 5 W bleed down resister

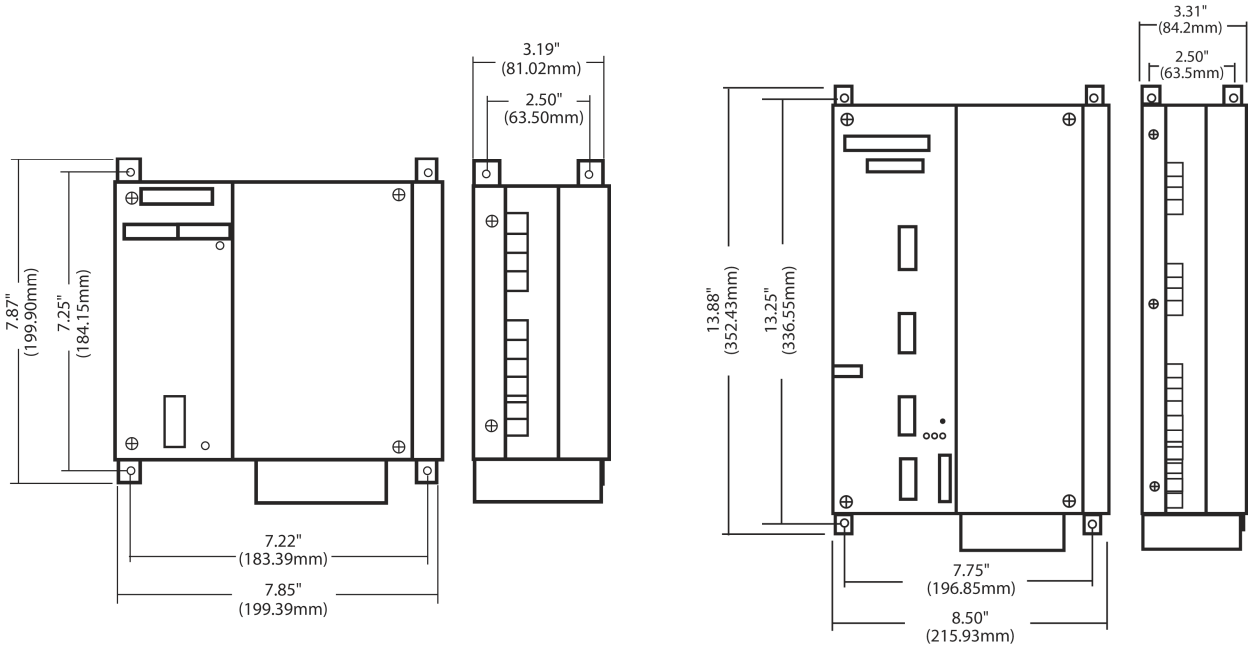
Servo Drive Installation

SD1/SD3 Description

The SD Drives are an integral part of the Centroid CNC control solution. Packaged as a complete motor drive for AC brushless as well as DC brush motor control, the SD3 and SD1 can be configured through software for a wide variety of applications. The SD3 will drive three motors, any combination of brushed or brushless, and accept an encoder input for the control of an external spindle inverter. All motors can run off a single power input or can have separate input voltages as required. Logic power supply voltage is developed from a DC input of 100 volts to 320 volts, which can be the same source as the motor power. Motor feedback is wired directly to the drive where the incremental encoder is read by an individual processor for each axis. Fiber optic communications with the drive are bi-directional so that status reporting can be viewed at the PC. An open collector transistor is available for a brake on each of the axes. This can be wired to an external relay to drive any type of motor brake.

An internal fan provides cooling to the main switching IGBT's as well as the entire circuit. Thermal sensing will warn and then shut down the drive in case of an overload. All motors are protected against over-currents through a fuse on each axis as well as current sensing circuitry that will shutdown and automatically reset on command. Fuses are mounted internally. All safety precautions must be followed when replacing fuses.

Specifications:



Outside Dimensions:

SD3	SD1
13.88 X 8.50 X 3.31 (in)	7.87 X 7.85 X 3.19 (in)
352 X 215 X 84 (mm)	200 X 199 X 81 (mm)

Weight:

SD3 5 lbs / 2.3 Kg
SD1 2.5 lbs / 1.13 Kg

Power Input:

Logic Power	100-340 Vdc	
Motor Power	5 – 340 Vdc	2000 Watts Continuous
	240Vac Max	5700 Watts Peak per axis

Encoder Input: Quadrature differential pair incremental encoder plus commutation. 5 V Max voltage. 5 inputs. Brushless motors are limited to 1024, 2048, 4096 line encoders. Spindle encoder input is for control of an external spindle drive through PLCIO module or equivalent. Cable Length 15m (50ft) maximum.

Control Interface: Synchronous Serial Communication Full Duplex to CPU9SD or CPU10 only – 4 optical fibers.
Max length 9m (30 ft). SD1 systems
Max length 12m (40 ft) SD3 systems

Motor Output Voltage:
5- 220 Vac AC Brushless
5- 320 Vdc DC Brush (100 -160 Vdc typical)
Motor output peak voltage will be the input level.

Motor Output Power:
2000 Watts continuous @ 220 Vac
5700 Watts Peak @ 220 Vac, 22 A 3 Phase
3 independent outputs.
Current mode, position commanded

SD1 High Power Model

4000 Watts continuous @ 220 Vac
11400 Watts Peak @ 220 Vac, 45 A 3 Phase

Axis Brake Driver: 50 Volts DC Max. (5 and 24 Volt typical)
0.5 Amps DC Max.

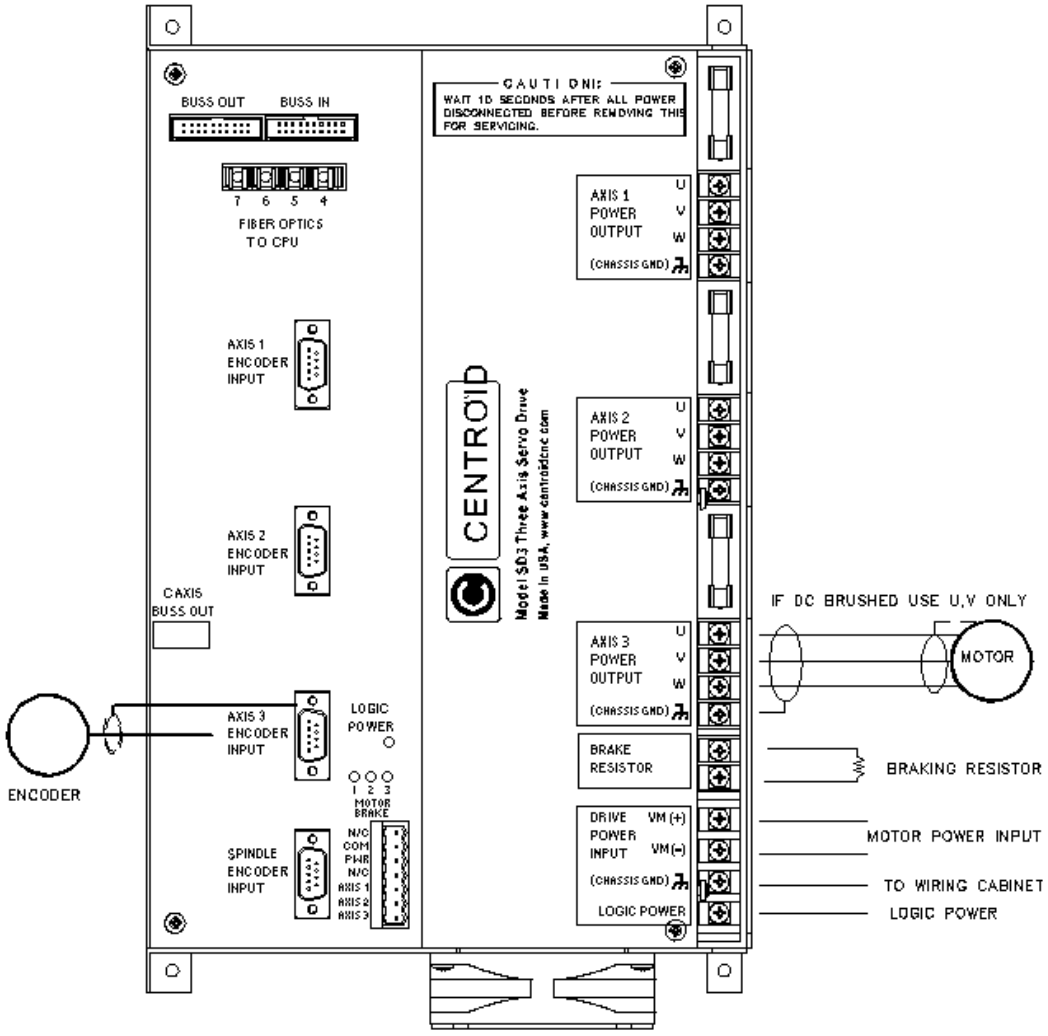
Braking Resistor: 40 ohms min., 200 watt minimum.

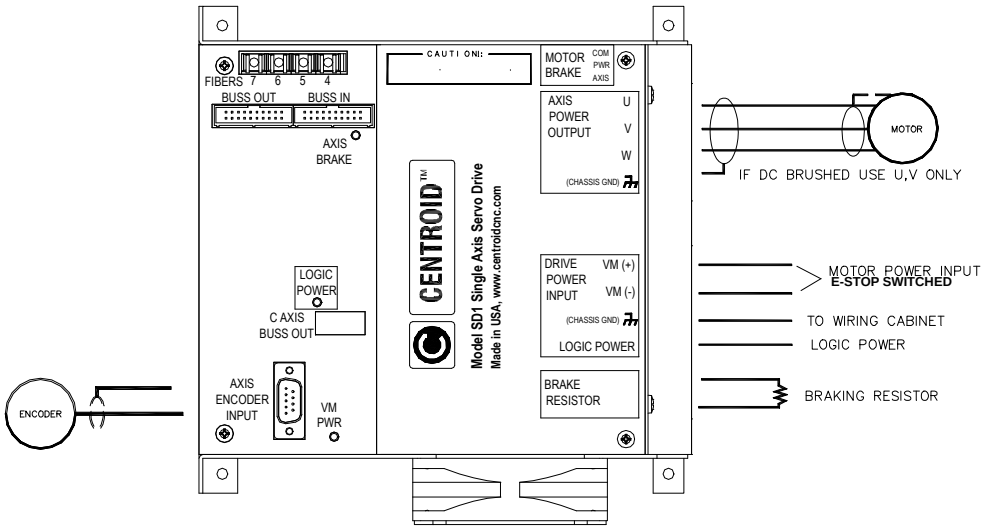
Motor cable length: 15 meters (50 feet) maximum

Installation:

Optimal mounting for the SD drive is vertical onto a metal backing. Note the direction of the airflow through the drive must go up. The drive chassis must be electrically connected to an earth grounded structure. Indicators on the drive cover should be visible without removing the unit for trouble-shooting purposes.

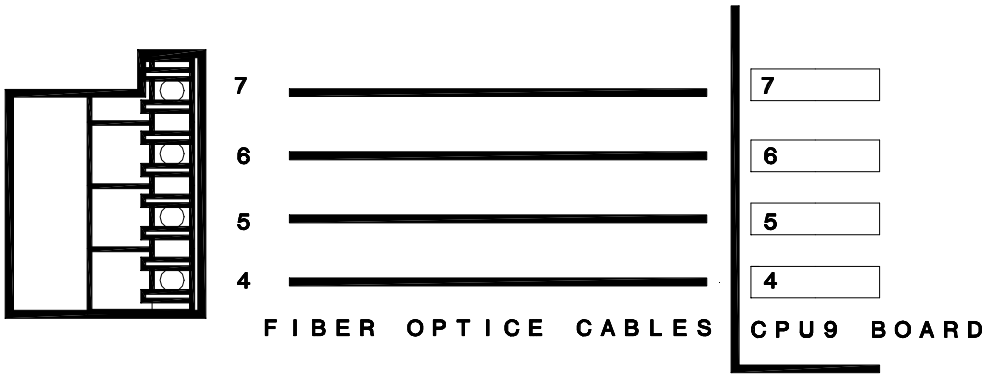
Wiring:





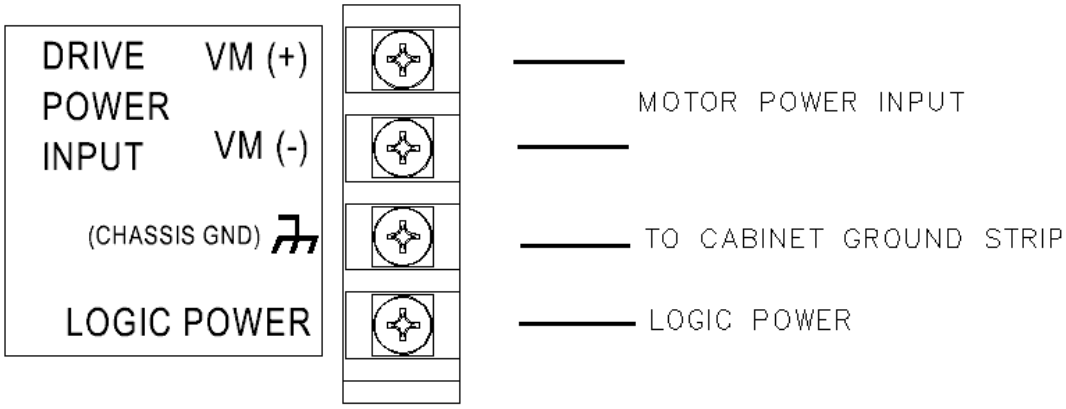
Motor power and encoder cables should not be run together. Only specified connectors are to be used.

Fiber optic cable connections are as shown. Note that the Data Transmit from the CPU9SD or CPU10 connects to the Data Receive of the SD drive and the Data Receive connects to the Data Transmit



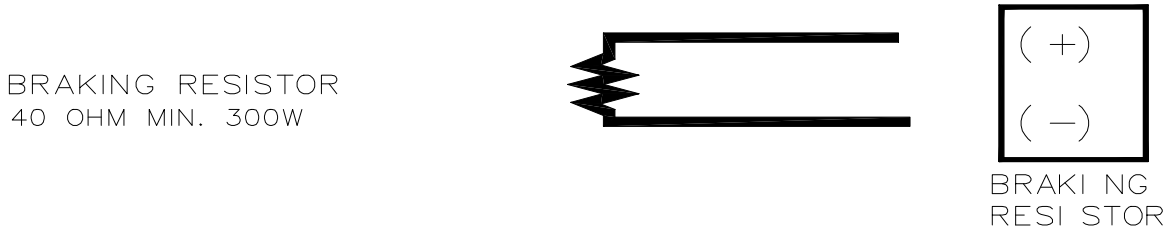
Input Power.

Power to the drive is DC only. Motor power and Bias power can be from the same source. The Motor power must be run through the E-STOP contactor before connecting to the drive. This voltage should match all the motors being driven. Motors of different voltage inputs may be used but a special input must be wired at the factory. Logic power voltage should be wired to come on with the main power disconnect. **If Logic Power is removed the machine power will need to be cycled to reboot the drive processors.**



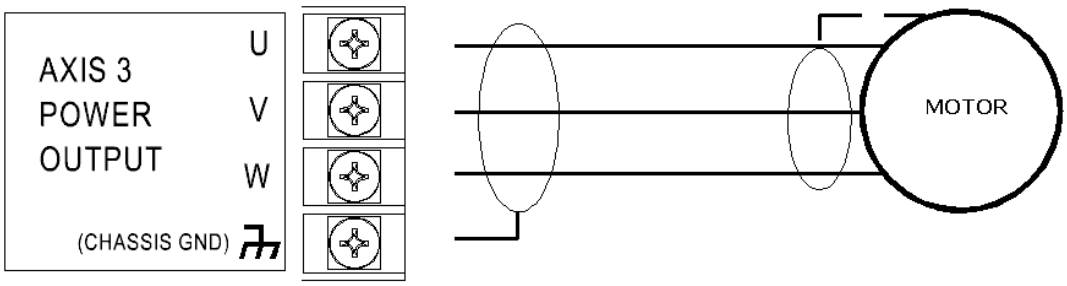
Brake resistor

A brake resistor must be installed on every system. High inertia systems require braking, as in the case where a motor generates back EMF into a drive and may cause damage. This resistor will help save the drive and slow the system under extreme braking. Braking current is limited to 10 A for 5 seconds.



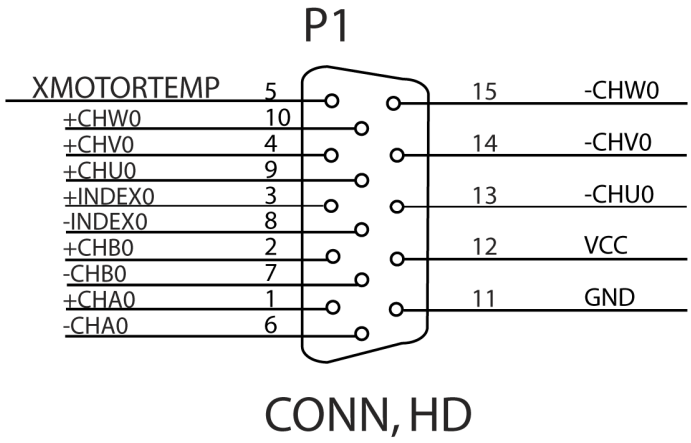
Motor power

Wiring to any motor must be with the approved connector and cable. The cable must include a shield that attaches at both ends. Brushless motors are wired to spin clockwise as viewed from the front of the motor while performing the sync-up operation in the drive menu. Brushed DC motors should wire the RED lead to the U connection and the Black to the V connection. W will remain unconnected for brushed motors.



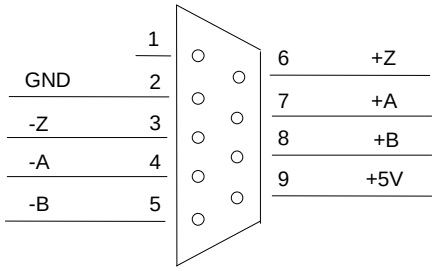
Encoder Input

Incremental encoders are used for either motor type but must be binary count for brushless motors (1024, 2048, 4096 lines). Encoders are wired to the drive only. Cables must be shielded with the shield drain attached to the drive side only. **DO NOT ATTACH SHIELD AT THE MOTOR SIDE!** To check encoder wiring, power up the system and go to the PID menu. Turn the shaft clockwise as viewed from the front of the motors; the absolute position should count up in positive numbers.



Spindle Encoder Input

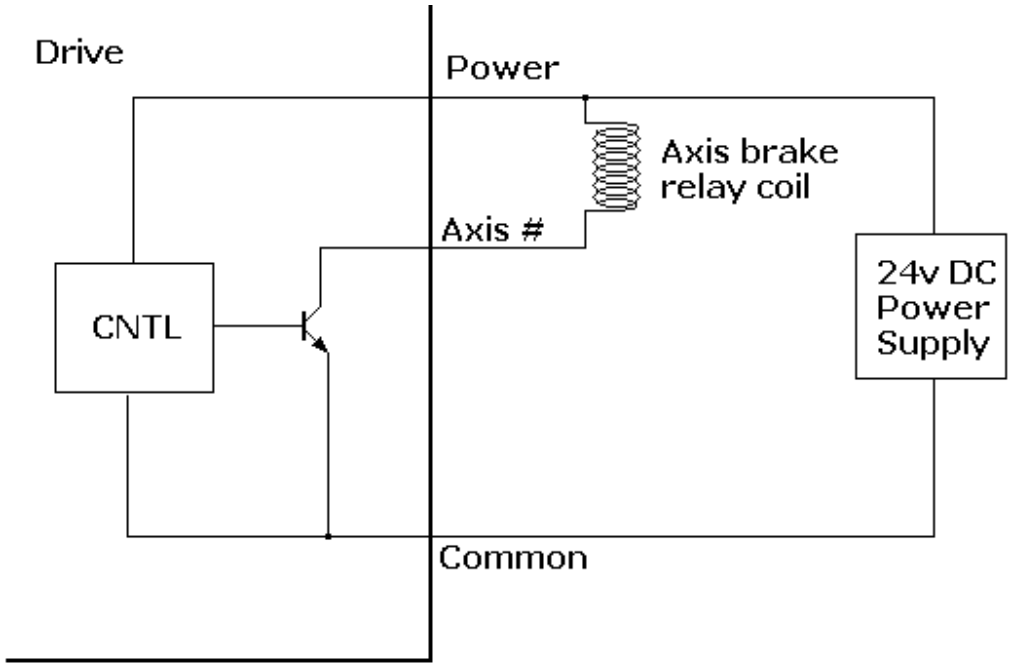
This input does not require commutation channel and is compatible with previous wiring. This spindle input is always axis 6 in the setup. Parameter 35 should be set to 5.



Axis Brake

A brake output is provided for each motor axis. It will engage whenever the motor drive is active. The circuit is an open-collector transistor that can be wired to a relay with a DC coil to operate the brake circuit. This circuit is isolated from the drive electronics and will need a separate power source ranging from 5 to a max. of 24 volts DC.

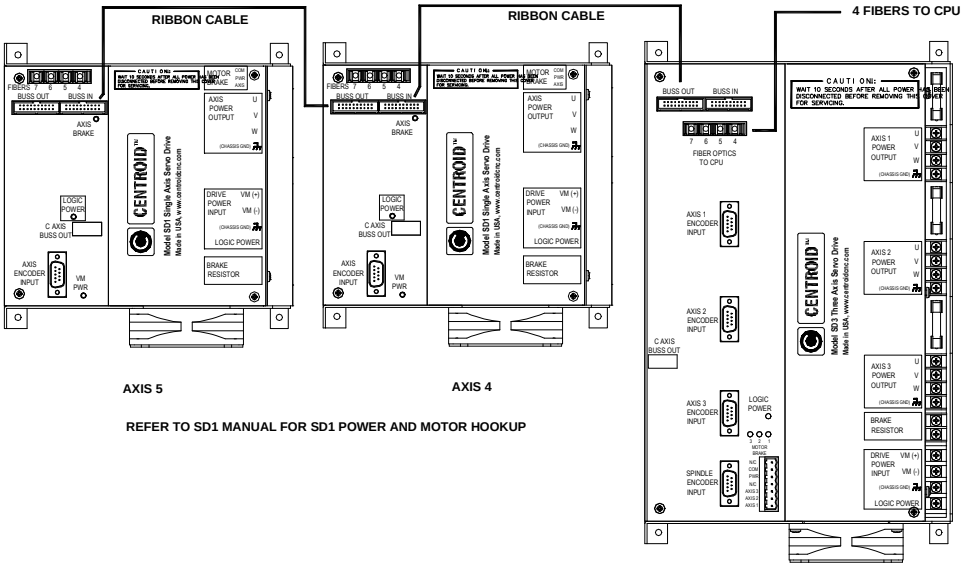
Brake Circuit:



Typical axis brake hookup

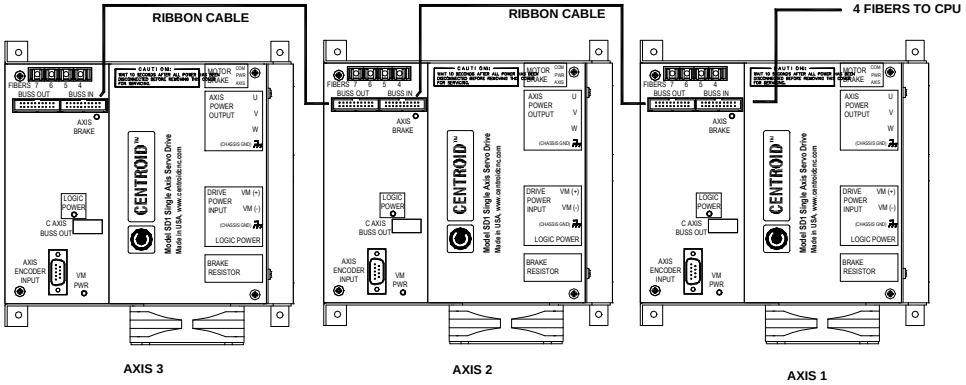
Adding Fourth and Fifth Axis Drives to an SD3 drive

Additional axes may be added using the BUSS OUT connector, on the front panel of the SD3, and a 20 pin ribbon cable. This connects to a SD1 single axis servo drive for communications. An addition axis may then be connected to the SD1 for up to 5 axes of motion. The SD1 is powered separately but communicates through the BUSS OUT connection of the SD3 back to the CPU9SD. The SD1 drives must be configured as the fourth or fifth axis with internal jumpers. The top cover must be removed to access these jumpers.



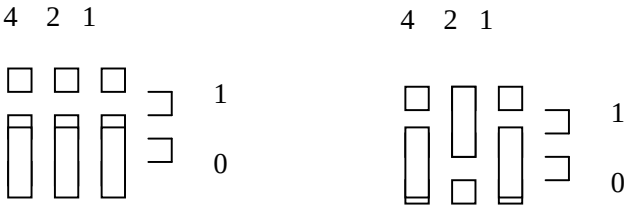
Systems of SD1 drives

The SD1 drive may be used without the SD3 for systems requiring 3 and 4 KW motors. The fiber optic communications cables connect into one of the SD1's and the rest connect via ribbon cables for up to 5 axes of motion. A CPU10 card must be used with these systems. The CPU10 provides the Spindle Encoder connection. All SD1's are powered separately and communicate through the BUSS OUT connection of the first SD1, which in turn communicates to the CPU10. The SD1 drives must be configured for the particular axis with internal jumpers. The top cover must be removed to access these jumpers.



Jumper setting:

To set drive for Axis 1, all jumpers should be in the 0 position. The axis number is a count from the binary addition of the jumpers. Axis 2 would have an axis select jumper 1 set and so on.



AXIS 1 AXIS 3

The J16 jumper, when installed, will disable the fiber optic components if you have an SD1 that has the fiber optic components installed but are using the ribbon cable into COMM IN.

Operation:

There are no manual adjustments on the SD drives or the CPU10. The entire configuration setup is done with software via the machine configuration, PID setups, and Drive Configuration menu. Drive current may be limited for smaller motors through the menu setting. The maximum current setting is 16 amps per axis. Brushless motors are designed with multiples of magnetic poles requiring numbers of cycles of the sine wave inputs to rotate the shaft a full revolution. Brushless motors may have 4, 6, and 8 poles and must be configured in the DRIVE MENU. A setting of 0 (zero) poles changes that axis to a Brushed DC drive. PID settings in the drive menu are factory set and must not be changed.

Driving brushless motors requires the knowledge of the motor's rotational position at all times. The encoder commutation channel gives the angular information to the drive. The drive generates 3 sine wave voltages phased relative to the motor position and the direction the motor is commanded to turn.

There are two configuration jumpers intended for future features on the PCB: J3 to enable the spindle DSP and J15 to disable the fiber input. A jumper should be in place on J3 for the standard setup.

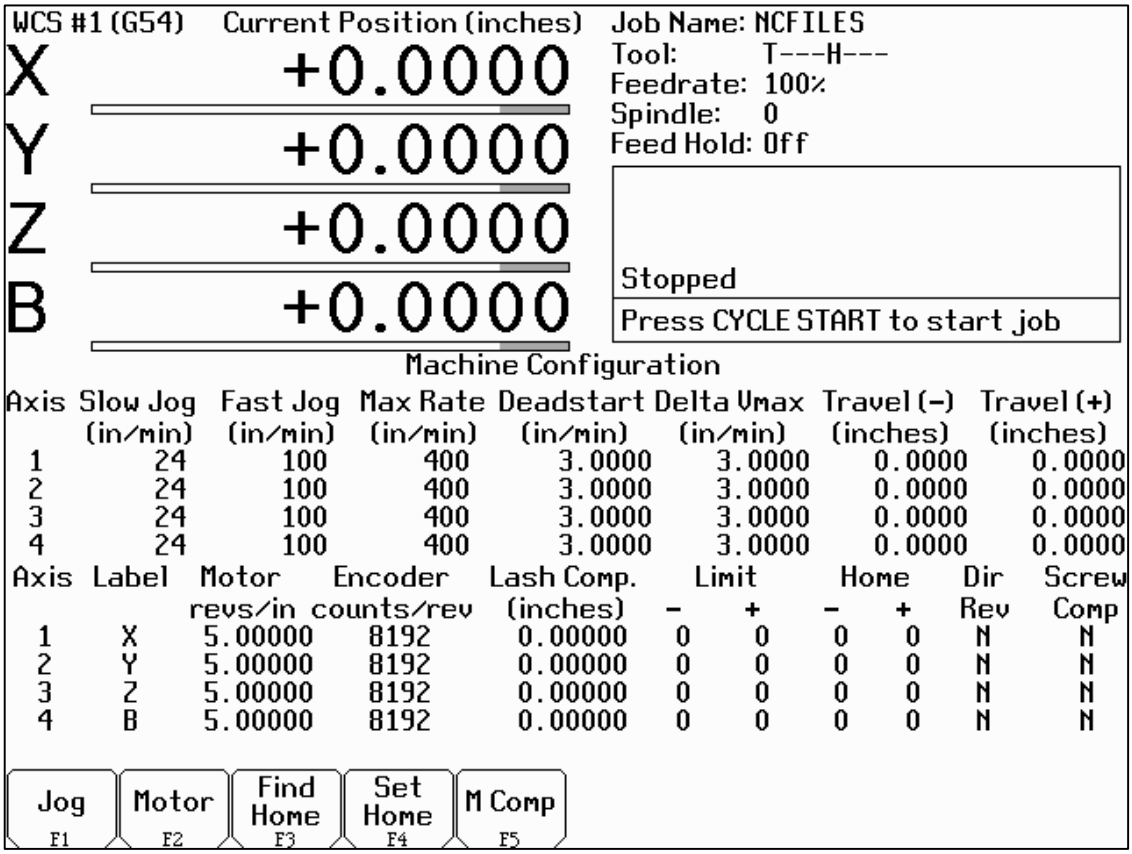
Motor Sync Procedure:

This procedure may be performed at any time to verify the operation of the motor and the encoder feedback.

a) Go to the Drive menu by pressing, <F1> Setup, <F3> Config, <F4> PID, and <F8> Drive.
b) Press <F2> (Move Sync)
c) Press <F1> (Change Axis).
d) Press <F10> to move the motor, verify that the motor turns clockwise. Verify also that the encoder reading counts up and rolls over at the max line count. The Commutation reading should count sequentially up to 6.
e) Repeat steps b) through d) 4 times for each motor.

After the procedure is complete back out to the first menu by pressing escape four times. **If the encoder has been repositioned then power must be cycled.**

SD MACHINE CONFIGURATION

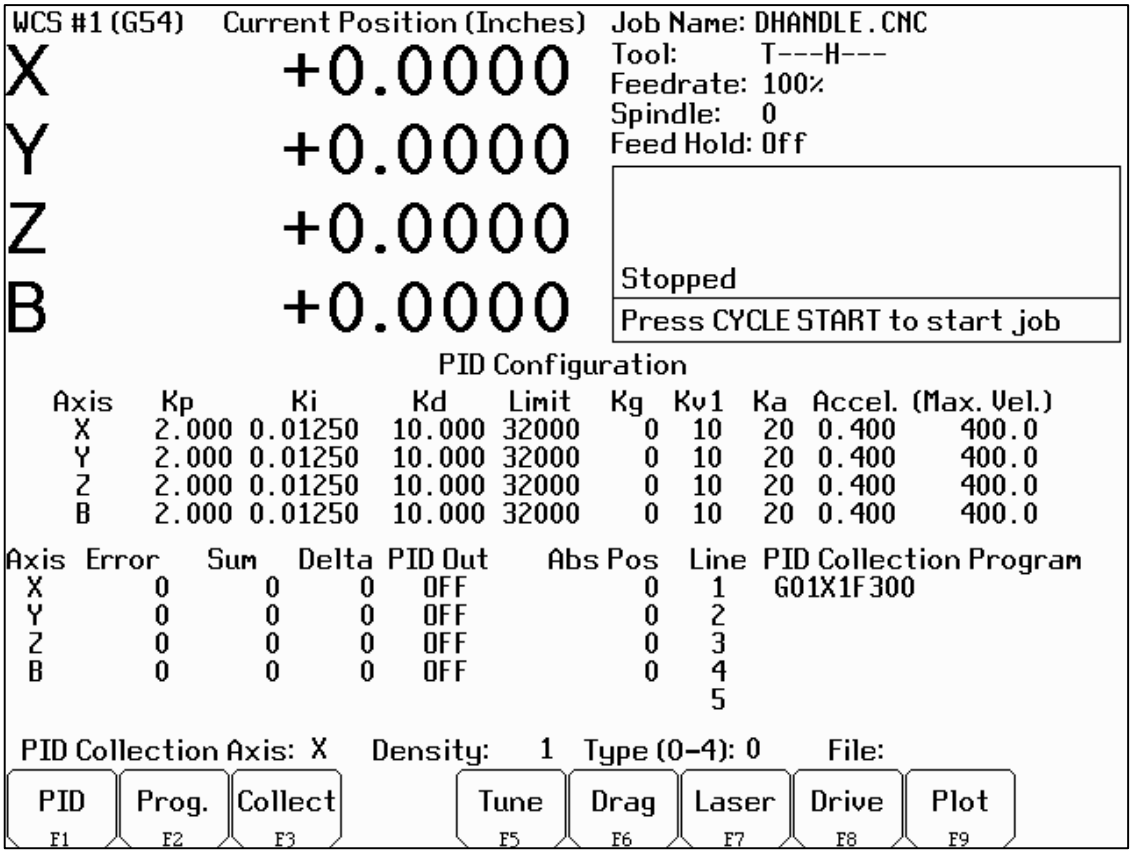


Setup notes:

Encoder counts / rev must be a binary count – 2048, 4096, 8192, 16384…
This is four times the line count of the encoder and is usually printed on the encoder device. SD drives accept only binary count encoders with matching commutation tracks. Brushed motors accept all line count encoders and do not require commutation tracks.

NOTE: If any axis is disabled then enabled using the label, the systems power must be turned off then on again.

SD PID MENU SETUP



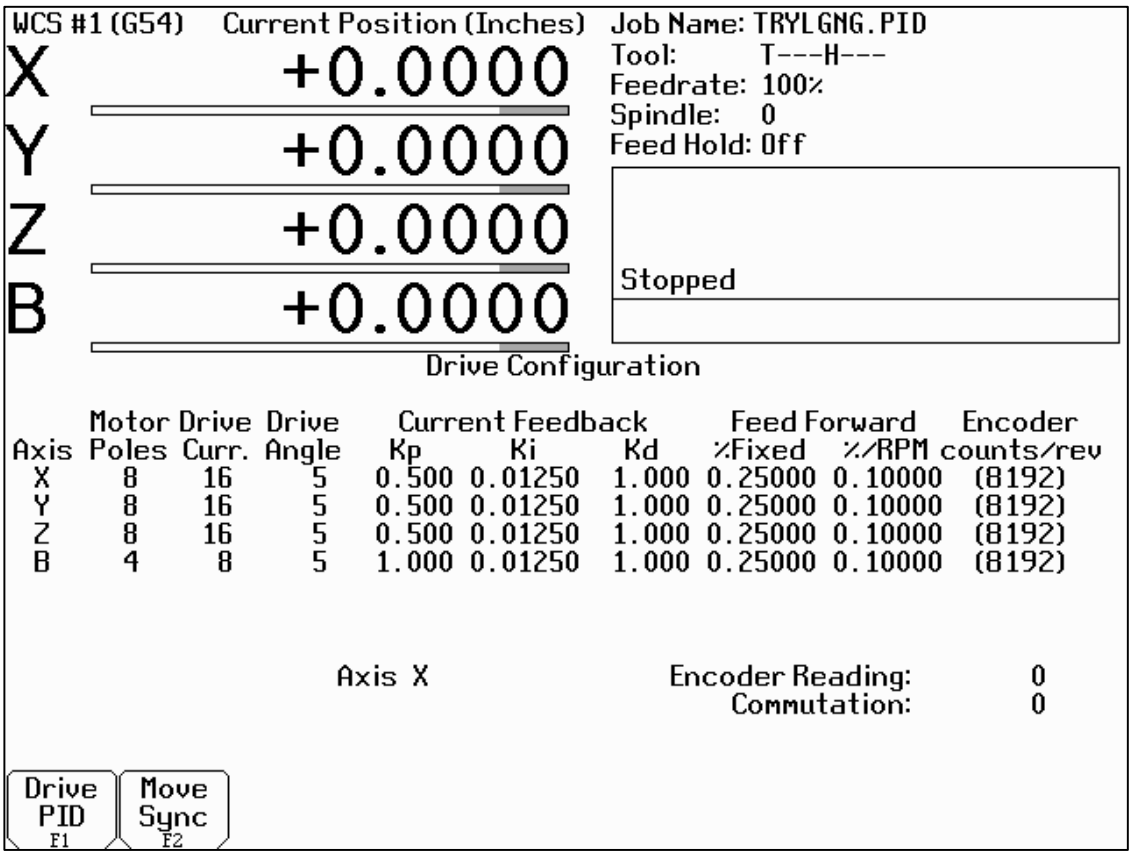
Pressing <F1> PID Key will enable you to adjust the above parameters. Ka should be left at zero, while the Kg and the Kv1 may be set by observing the PID Out while the axis is at rest and during a slow jog respectively.
F3 Collect Key
The Data Density can range from 1-32. (It is passed to CPU9 as 0-31).
The Data Type can range from 0-4.

0=Collect Error (n), Sum Error (n), Delta Error (n)
1=Collect ADC Input A, ADC Input B, ADC Input C
2=Collect PID Request A, ADC Input A, PID Adjusted A
3=Collect PID Request B, ADC Input B, PID Adjusted B
4=Collect PID Request C, ADC Input C, PID Adjusted C

The File Extension should be unique to your collection.

The Drive Configuration Screen (F8) is not modification by any unauthorized individual.

SD DRIVE MENU SETUP



This menu will only appear if the control has been configured for an SD drive, with a CNC9.hex present in the CNC10 directory.

F1 DRIVE PID

Motor Poles are 0 for Brush Motors or 4, 6, 8 for Brushless Motors.
(For 0, Brush Motors, the Drive Angle and the Feed Forward %'s are all passed to CPU9 as 0's). **750 W, 1 KW, 2 KW, 3 KW, 4 KW, is 8 poles. 400 W is a 4 pole motor**.

Drive current is scaleable current output from the drive 1 to 16 amps. A zero setting will give 16 amps. Drive current directly effects the acceleration of a motor and its heating. **Smaller motors such as the 750 watt and 1 KW motor should use a reduced current setting of 8 for 750, 12 for 1 KW.**

Current Feedback PID and Feed forward – current loop PID parameters use the default parameters for the application. Consult the factory if you feel it necessary to make a change.
-The Encoder Counts/Rev. parameter is for information only, in this screen. To modify it you must go to the Machine Configuration Setup Menu.

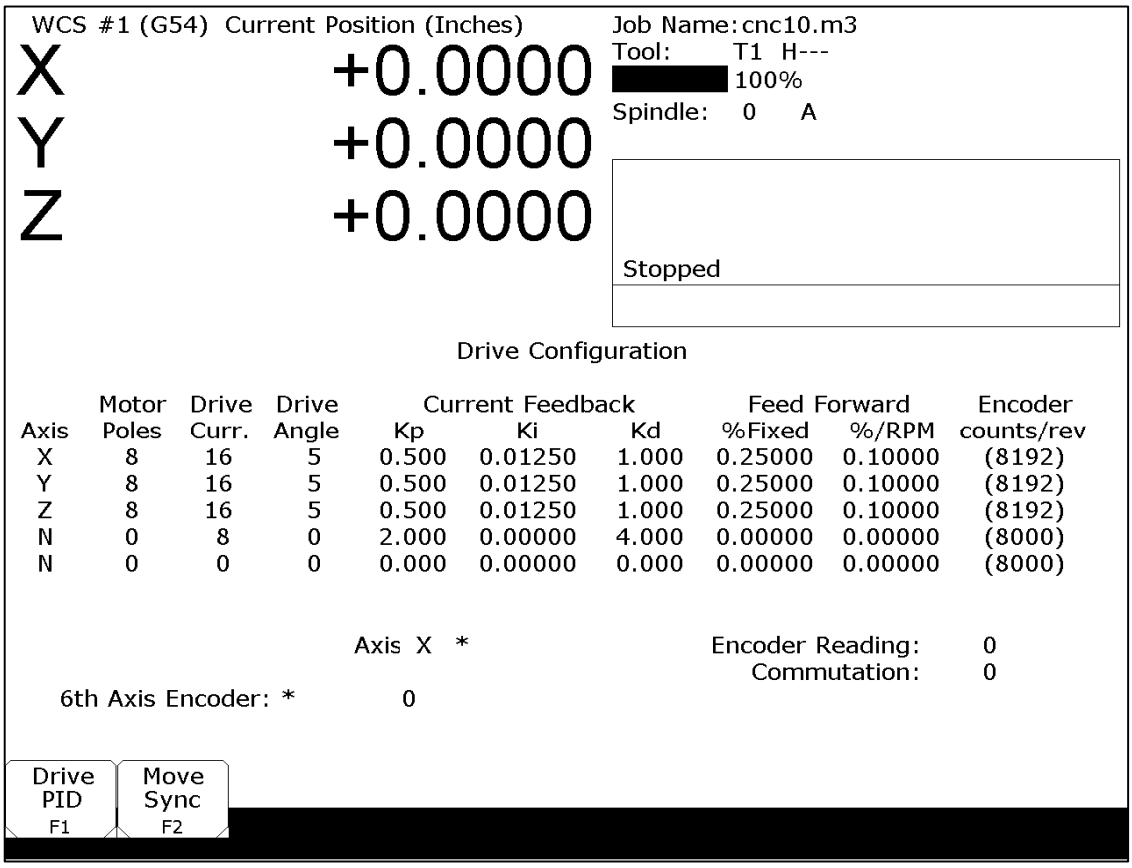
F2 MOVE SYNC

Move Synch causes the Axis Motor to move to a Synch Position and Hold Power, ignoring the Encoder Input. If the rotor is at the correct Synch Position the Index Pulse will be at Encoder Position 0 or within 20 counts +/-.

Move Synch may be executed several times to Move the Motor to the proper quadrant. It takes four cycles on an 8 pole motor. This is a good tool for trouble shooting motor – encoder problems.

This menu shows a standard setup for a three axes mill with a 4th axis brushed DC motor rotary table.

DRIVE MENU WITH DC MOTOR 4TH AXIS ROTARY TABLE



Drive current is 16 amps for large DC motors, 19 in-lbs or greater. Drive current directly effects motor heat and should be check during normal operation.

TROUBLE SHOOTING GUIDE

Symptom	Cause	Corrective Action
Logic Power Indicator OFF	This indicates the presence of power the logic circuits internal to the DRIVE.	If this lamp is out and there is power to the Logic Power input the factory must service the unit.
Motor turn backwards during the alignment procedure. (message)	The motor power wiring must be backwards in the cable of the motor itself.	Examine the cable and reverse two of the motor wires.
Encoder Counts runs backward.	The A and B signals from the encoder may be swapped. Encoder may operate in the reverse.	Check encoder cable. Check encoder spec- A must lead B when rotated clockwise as viewed from the front of the motor.
Motor jerks when first moved.	The motor may be wired backward or the encoder counts backwards. Motor may have lost its encoder alignment or never aligned.	Go through the motor sync procedure in the DRIVE menu with the **motor unloaded**.
Servo Drive Output error. (439)	This is a loss of signal coming back from the drive to the CPU9SD.	Power the system down and then back up. Check the Logic Power indicator. If illuminated then start the system. Run through the motor alignment procedure.
Servo Drive Over voltage (441)	Indicates the Input power has gone higher than 340 volts and will shut down the drive and remove power. The Motor brake will engage for 5 seconds in this condition.	Check input voltage. It must be below 340 volts DC.
Servo Drive Under voltage (442)	Indicates that the input voltage to the supply is less than 80 volts.	Check supply voltage.
Encoder connection bad (412)	The DRIVE checks the encoder line for differential signals. –A must have the opposite voltage level as +A.	Check the wiring of the encoder cable, then the encoder itself.
Servo Drive Over Temperature (444)	There is a temperature sensor mounted on the heat sink internal to the DRIVE that detects over temperature.	The drive is being run at over capacity. The cooling fan is not functioning or the inlet or outlet is blocked.
Over Current. (445)	Individual axis current is monitored and will shut down the drive when a level is measured above the preset limit.	Try to jog the axis. The DRIVE will reset the current limit and try to move the motor. If you continue to get an over current check for a short in the motor output.
Axis runaway	If the motor moves when not commanded it is possible that noise is getting on the encoder	Check that the motor cable shields are attached. Check that the encoder shield is

	line and creating a constant position error	attached at the drive end only. Check that the DRIVE is electrically grounded to the machine to which the motors are mounted. Separate the motor power cables from the encoder cables and other power cables.
Encoder Commutation Error (443)	Control detected invalid commutation zone value	Perform a motor Move Sync in the Drive menu. A Zero (0) or Seven (7) is an invalid zone. Check for a wiring problem in the cable or motor end cap.

CENTROID AC MOTOR DATA

October 1, 2004

Motor	Working RPM	Kv V/KRPM	KT IN/LBS/ AMP	Ip (Peak Current) AMPS	Peak Torque with Centroid drive (in.lbs)	Constant stall Torque (in.lbs)	Power Rating
6 in.lbs AC(SEM)	6000 @ 220 Vac	44	4.5	7	31	6 @ 1.4 A	400 W
35 in.lbs AC(LD85)	3600 @ 220 Vac	48	10	7	60 (11 Amps)	35 @ 3.5 A	750 W
50 in.lbs AC (CM1K-0-1)	3500 @ 220 Vac	48	8.6	16	137 (16 Amp)	50 @ 6 A	1 KW
50 in.lbs AC (CM1K-0-2)	2200 @ 220 Vac	79	12	22	264 (16 Amps)	50 @ 4 A	1 KW
50 in.lbs AC(SEM)	3200 @ 220 Vac	66	6.6	34	148 (22.5 A)	50 @ 8 A	1 KW
100 in.lbs (CM2K-0-2)	2000 @ 220 Vac	84	12.4	23	280 (22.5 A)	100 @ 8 A	2 KW
100 in.lbs (SEM)	2500 @ 220 Vac	88	9	33	202 (22.5 A)	100 @ 11 A	2 KW
150 in.lbs (CM3K-0-2)	1700 @ 220 Vac	100	12.8	25	288 (22.5 A)	150 @ 12 A	3 KW
150 in.lbs (SEM)	1350 @220 Vac	130	13.5	42	455 (34 Amps)	150 @ 11 A	3 KW
200 in.lbs (SEM)	1350 @ 220 Vac	130	13.5	70	607 (45 Amps)	200 @ 14.8 A	4 KW

Centroid SD DRIVE Brushless Motor Configuration

SD DRIVE Brushless Motor Configuration

		PID MENU								**Drive Menu**							
MOTOR	Motor	Kp	Ki	Kd	Kg*	Kv1*	Ka*	Accel*	MaxVel**	Motor	Drive	Drive	Current Feedback			Feed Forward	
	Power									Poles	Current	Angle	Kp	Ki	Kd	% Fixed	% RPM
HJ130C8 (S)	1 KW	2	0.0125	10	0	10	0	0.4	3400 / (pitch)	8	16	5	0.5	0.0125	1	0.25	0.1
HJ130E8 (S)	1.6 KW	2	0.0125	10	0	10	0	0.4	3400 / (pitch)	8	16	5	0.5	0.0125	1	0.25	0.1
HJ130G8 (S)	2 KW	2	0.0125	10	0	10	0	0.4	2500 / (pitch)	8	16	5	0.5	0.0125	1	0.25	0.1
HJT155B8 (S)	3 KW	2	0.0125	10	0	10	0	0.4	1350 / (pitch)	8	12***	5	0.5	0.0125	1	0.25	0.1
HJT155D8 (S)	3 KW	2	0.0125	10	0	10	0	0.4	1350 / (pitch)	8	16***	5	0.5	0.0125	1	0.25	0.1
HR70A4 (S)	400 W	2	0.0125	10	0	10	0	0.2	3400 / (pitch)	**4**	4	5	1	0.0125	1	0.25	0.1
LD85 750W	750 W	2	0.0125	10	0	10	0	0.4	3600 /(pitch)	8	**8**	5	0.5	0.0125	1	0.25	0.1
CM1K-0-1	1 KW	2	0.0125	10	0	10	0	0.4	3400 /(pitch)	8	**12**	5	0.5	0.0125	1	0.25	0.1
CM1K-0-2	1 KW	2	0.0125	10	0	10	0	0.4	2200 /(pitch)	8	**16**	5	0.5	0.0125	1	0.25	0.1
CM2K-0-2	2 KW	2	0.0125	10	0	10	0	0.4	2000 /(pitch)	8	16	5	0.5	0.0125	1	0.25	0.1
CM3K-0-2	3 KW	2	0.0125	10	0	10	0	0.4	1700 /(pitch)	8	16	5	0.5	0.0125	1	0.25	0.1

* Kg, Kv1, Ka, accel are values to be set before autotune is performed, autotune will then calculate values suited to the application.
** This is the value expected after autotune, it is dependant on the pitch of the ball screw and the belting.
*** 45 AMP SD1 ONLY

Servo 1/Quad Drive

Features

- Wide input motor voltage range.
- Optic isolation for operation in high noise environments.
- Current matching to application.
- Three (Servo 1) or four (Quad) Brushed DC axis drives on one card.
- Optically isolated limit switch inputs.
- Drive Status LED's.
- Drive Fault output, Optically Isolated.
- Drive Fault relay contact closure output for motor voltage contactor control.
- Under voltage protection.
- Third axis brake motor contact closure output, the quad drive also includes a forth axis brake motor contact closure output..

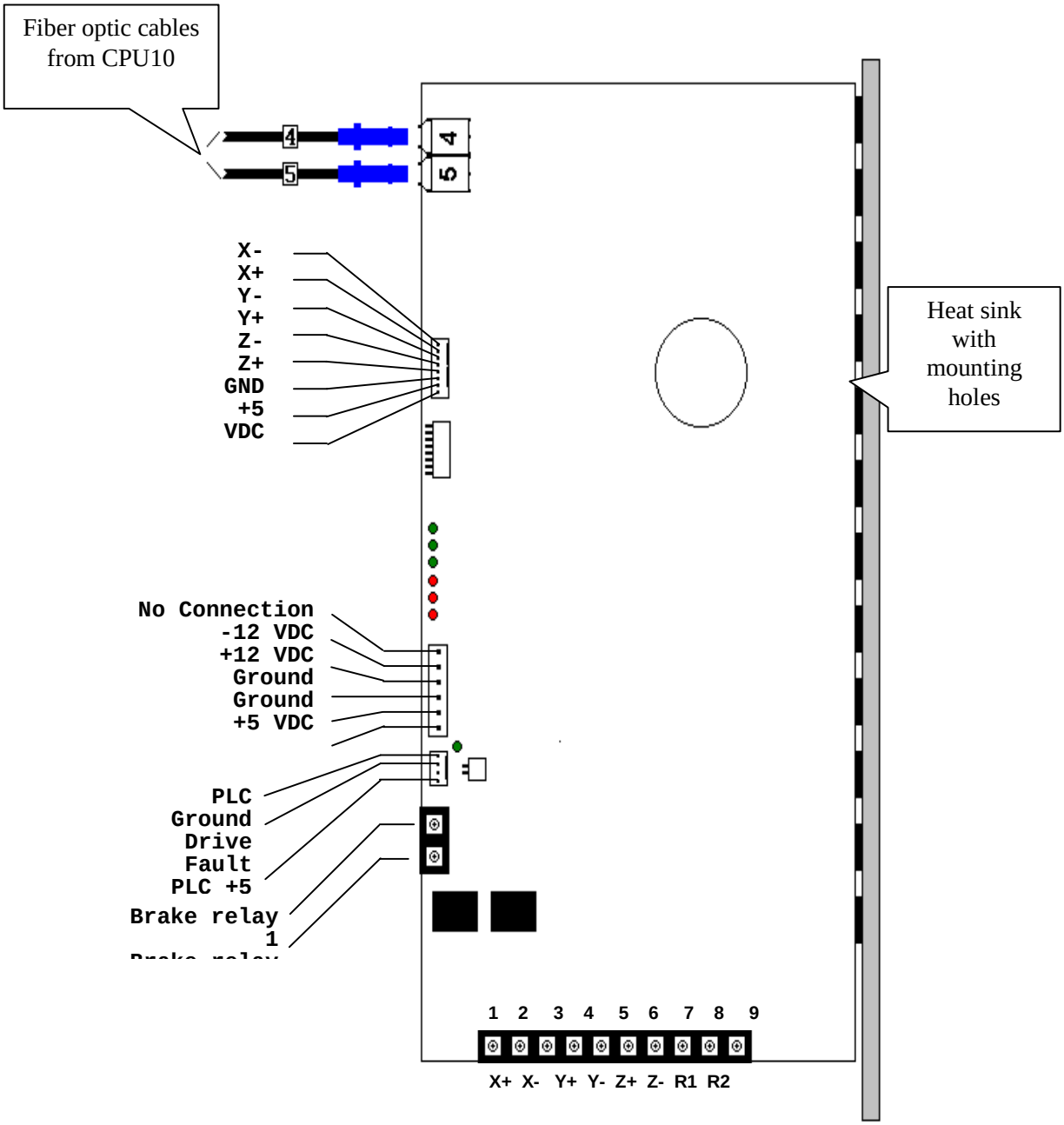
Board Power and Motor Specifications

Parameter	Symbol	Max	Typ	Min	Unit
Motor Voltage Supply Input	Vm	140	110	80	Volts
Motor Voltage Supply Current	Isupply	3Im(x)		0.25	Amps
Motor Current (Large Mills) *See Note 1	Im(1)	26	25	23	Amps
Motor Current (Medium Mills) *See Note 1	Im(2)	16	14.5	14	Amps
Motor Current (Regular Mills) *See Note 1	Im(3)	13	12	11	Amps
Motor Current (Tables)#	Im(4)	10.5	9.5	9	Amps
Motor Current (Small Tables) *See Note 1	Im(5)	6.5	5	4.5	Amps
+5V Logic supply input	I_{+5V}			1	Amp
+12V Logic supply input	I_{+12V}			0.5	Amps
-12V Logic supply input	I_{-12V}			0.2	Amps

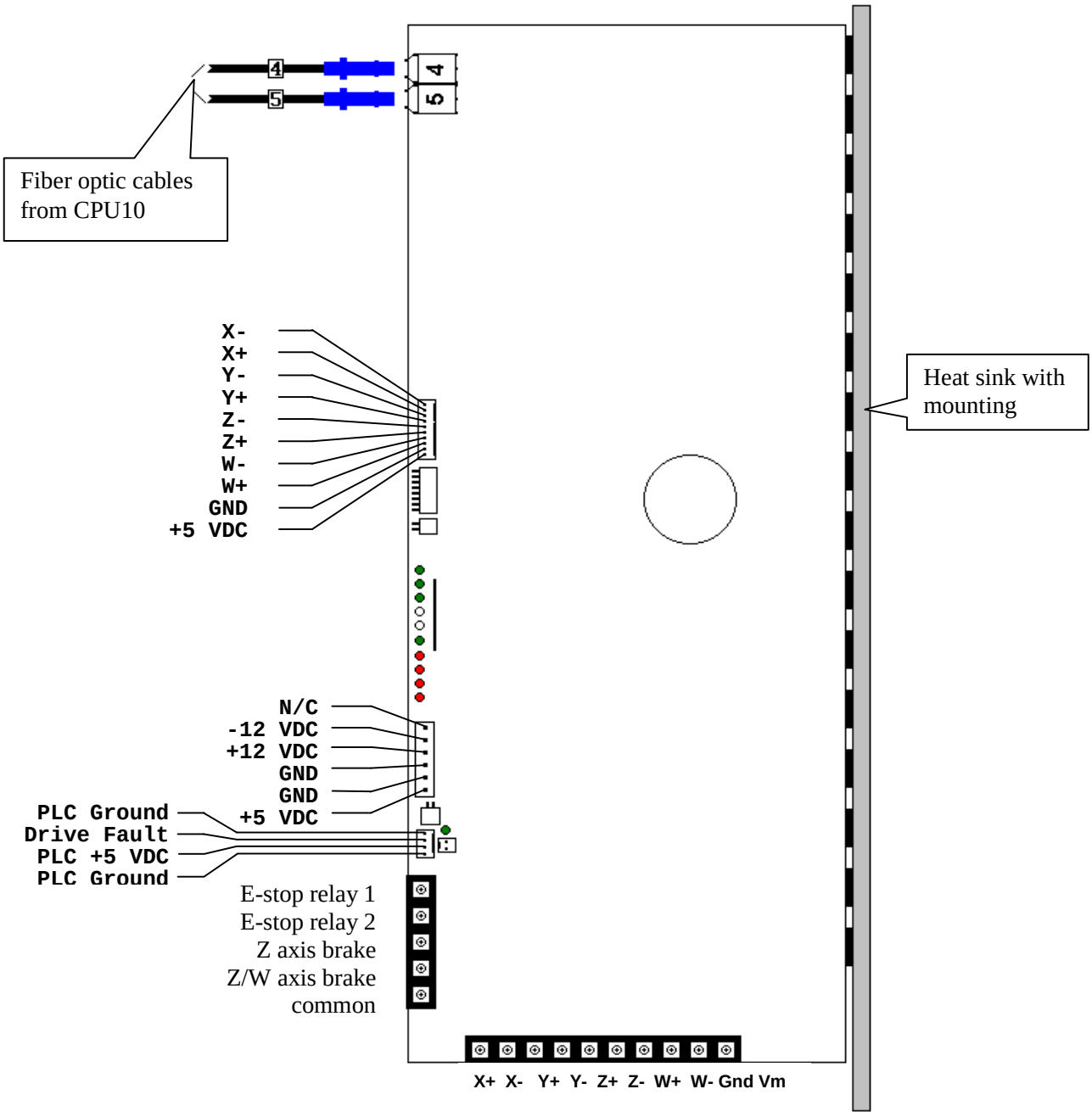
Note 1: Motor Current Set during drive production for particular application.

Ensure that this is motor <u>continuous</u> stall current and not peak stall current or peak pulse current.

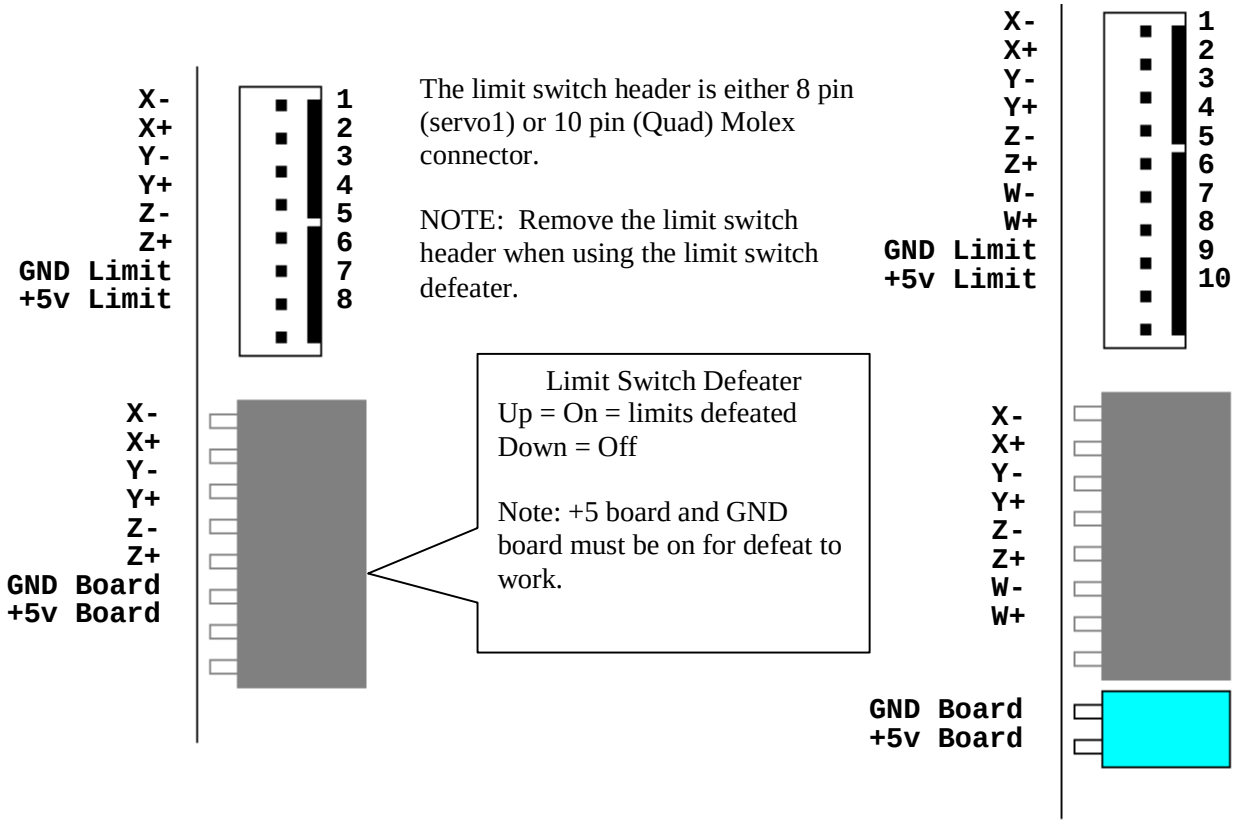
Servo 1 Hookup Diagram



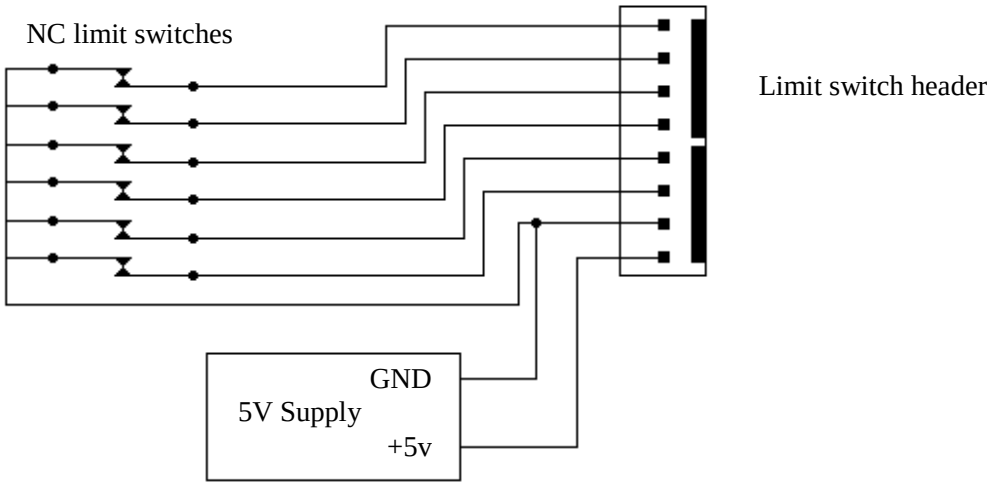
Quad Hookup Diagram



Limit Switch Input and Defeater Switches

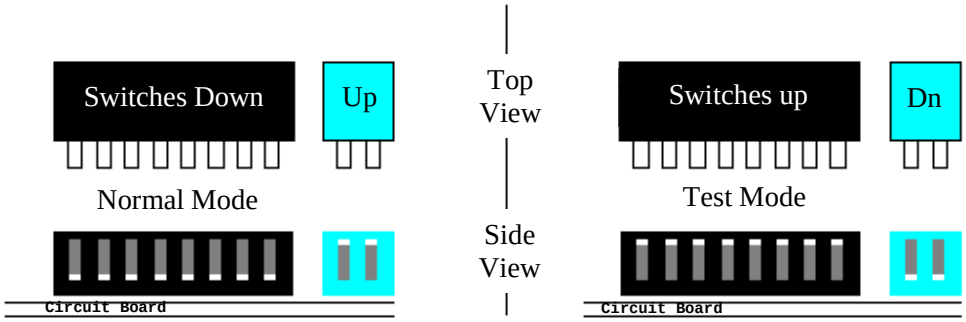


The limit switch header is a 0.1 inch center Molex connector. All inputs should be connected to a normally closed limit switch connected to the GND LIMIT input pin. An isolated +5 Volt supply is needed for limit switch operation. The 5 Volt supply should be connected to the +5 LIMIT pin and the GND LIMIT pin.



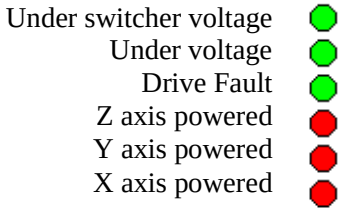
When a problem with the limit switch wiring is suspected the limit switch defeater can be used to confirm that the drive board is operating properly and that the problem is with the limit switch wiring, by bypassing the limit switch inputs to the drive.

*Note: The limit switch connector must be disconnected from the drive board when the limit switches are disabled. For normal operation, the DIP switches should all be in the off position. Failure to do so might lead to improper functioning of Drive or PLC.

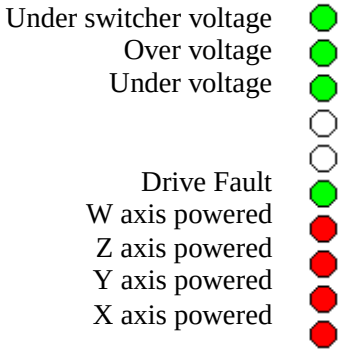


Switches are shown for the Quad drive, the bank of 2 switches are opposite the bank of 8 switches. When the bank of 8 switches are in the down position (OFF) the bank of 2 are up. The Servo1 does not use the bank of 2 switches

Status LED configuration:
The status LED's can be used to debug any problem you might have with the servo drive. The figure shows the status LED configuration. Following is a description of each of the LED's.



Servo1 drive status LEDs



Quad drive status LEDs

Under Switcher Voltage:
ON denotes good gate drive IC supply voltage.
OFF denotes Low Switcher Voltage; self-resetting.

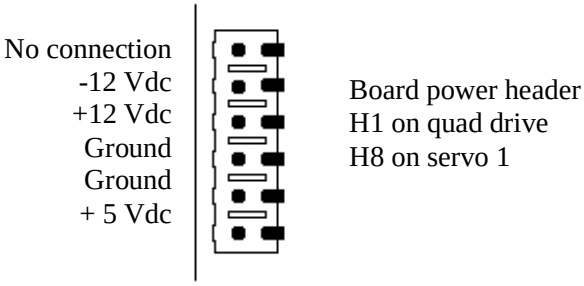
Over Voltage: (Quad drive only)
ON denotes good motor supply voltage.
OFF denotes High motor voltage; self-resetting.

Under Voltage:
ON denotes good motor supply voltage.
OFF denotes Low or No motor voltage; self-resetting.

Drive Fault:
ON denotes normal operation.
OFF denotes drive fault or Drive-Stop condition. Also may indicate (DSP, CNC10, or CPU10) drive shutdown commands or that no commands are reaching the drive (bad fiber optic connection).

Axis Powered: Z, Y, X, (W for quad drive). (Red LED)
ON indicates motors are free or not yet commanded; no power applied to motor.
OFF indicates motors are holding or moving as commanded; power applied to motor.

SERVO1/Quad Drive Analog and Digital Power Header:
The power header is a 0.156-inch center Molex connector. The servo drive is supplied with a COSEL brand power supply and cables for connection into this header.



SERVO1/Quad Drive Analog and Digital Power Requirements:

Parameter	Symbol	Min	Unit
+5 Vdc Logic supply input	I_{+5V}	1	Amp
+12 Vdc Logic supply input	I_{+12V}	0.5	Amp
-12 Vdc Logic supply input	I_{-12V}	0.2	Amp

Drive Stop Signal and Brake Power Control
The SERVO1 and Quad drive card have a Drive Fault / CPU Not Ready output to the PLC and contact closure for motor power supply contactor control. The DRV FAULT to the PLC output header also requires an isolated power supply. The power supply used for the limit switches and other PLC inputs can also be used for this header.
The contactor relay closure for the Drive Fault / CPU Not Ready is accessible at the 10 pin terminal block H4 pins 7 and 8 on the Servo1, and on the 5 pin terminal block H? pins 1 and 2 are on the Quad drive. This closure can handle up to 10 Amps at 250 Vac or 30 Vdc.

In addition to the fault contact there is also a brake relay closure for the Z, or 3rd, axis for Servo1 and the Quad drive. The Quad drive also has a W, or 4th, axis brake relay closure. The Z brake output is intended for Mills with heavy heads or knees that fall when not powered. The output will close an external power supply across the brake winding of a brake motor. The Quad drive's W, or 4th, axis brake output can be used for a rotary table or powered knee. The power supply needed to power the brake windings must be connected external to the SERVO1/Quad board. Diagram 2 shows the schematic hookup for the brake motor.

E-Stop to PLC output header and Brake relay output terminal strip

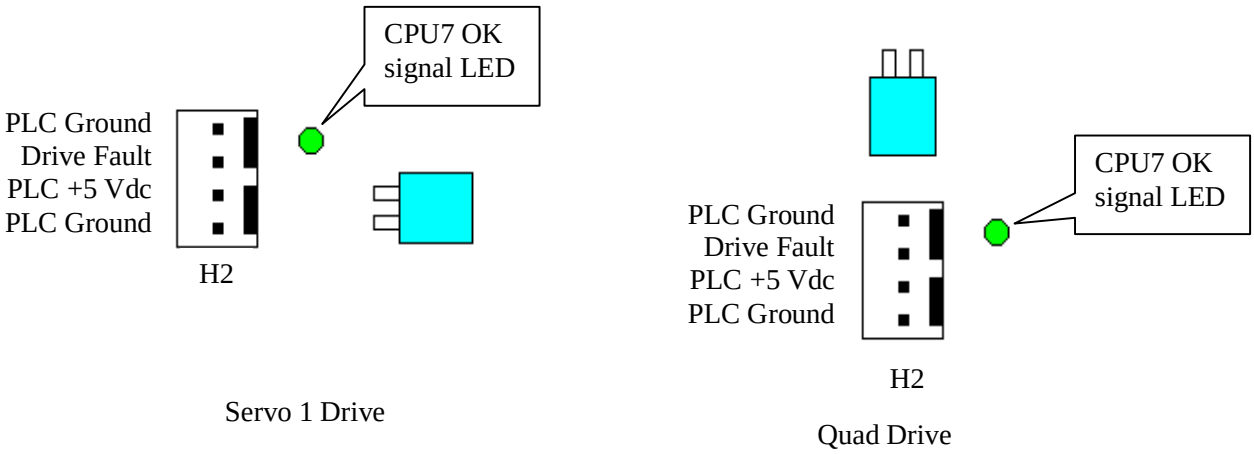
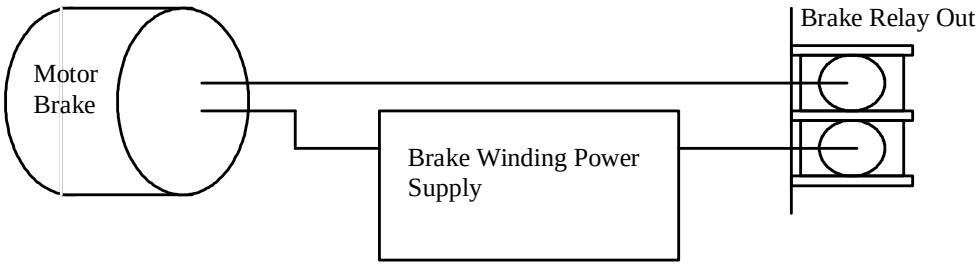
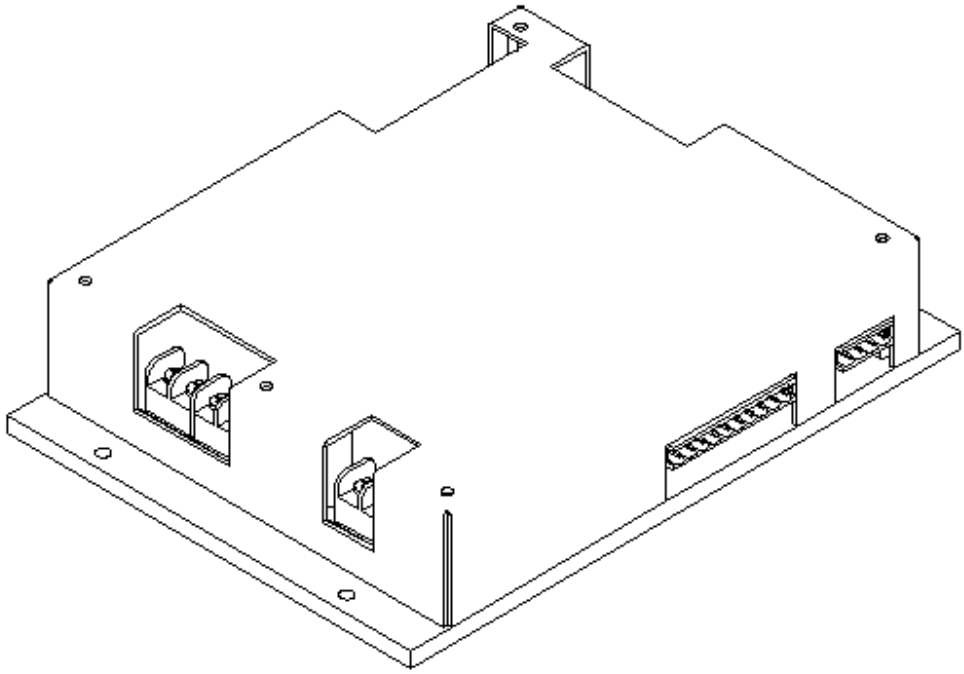


Diagram 2 Brake Motor Hookup



DCSINGLE Installation Guide (Revision 040803)

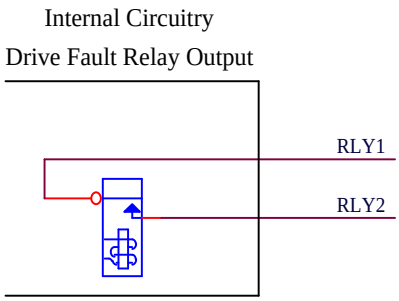


The DCSINGLE is a single axis DC brush motor drive with an optional PLC section. This drive is aimed at 4th axis upgrades to existing three axis systems. The drive can be ordered as axis 1, 2, 3, or 4, but does not allow for 5th axis operation. An optional PLC section allows for single axis applications if necessary. The DCSINGLE uses technology from the SERVO3IO drive including the PLC and drive section fiber communications and available current ratings of 9, 12, and 15 amps. A unique feature of the drive is the fiber repeaters. The drive fibers from the motion control card (CPU10) run to the DCSINGLE and a short set of fibers run from the repeaters to the original drive.

Fourth Axis Connection

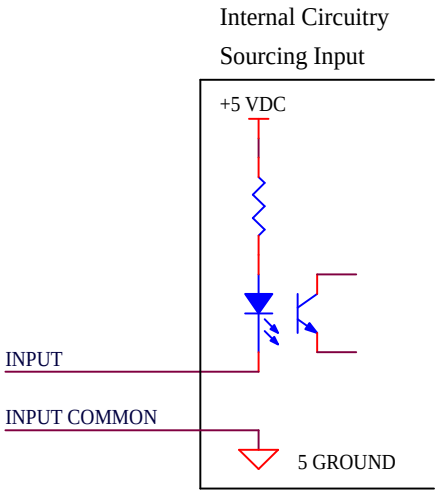
Typical wiring of the DCSINGLE requires only logic power, motor power, motor, drive fault, and fiber optic connections. Fiber optics 4 and 5 receive data from the motion control card. Fiber repeaters 4RPT and 5RPT chain drive information to the original DC servo drive. Drive fault RLY1 and RLY2 connections are wired in series with the emergency stop power loop to cut motor power in the case of a fault. The drive fault signal is passed to the control from the original drive, since it runs off the DCSINGLE fiber repeaters. The limit defeaters are normally on for rotary axis use.

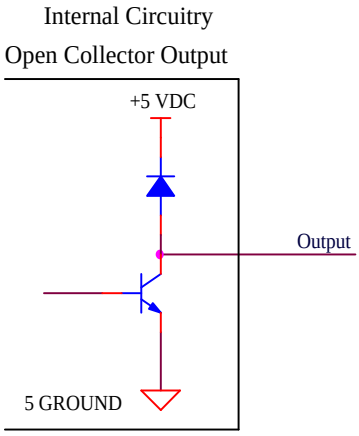
Open collector brake and drive fault outputs are available for special applications. The brake output can drive a 5 volt relay to release a motor brake when the axis is enabled. The drive fault open collector output may be wired to an external PLC to signal a fault condition.



Optional PLC

The DCSINGLE can be equipped with a PLC section, allowing for single axis applications. Six inputs and six outputs are available for use on a PLC equipped DCSINGLE. All outputs are open collector types suitable for driving 5 Volt relay coils. The PLC section requires a motion control card equipped with a CPU711 compatible (old style) PIC chip.





DCSINGLE I/O MAP

PLC Input	Use	PLC Output	Use
1	General	1	General
2	General	2	General
3	General	3	General
4	General	4	General
5	General	5	General
6	General	6	General
7	X+ Limit	7	N/C
8	X- Limit	8	N/C
9	Y+ Limit	9	N/C
10	Y- Limit	10	N/C
11	Z+ Limit	11	N/C
12	Z- Limit	12	N/C
13	Drive Fault (internal)	13	N/C
14	W+ Limit	14	N/C
15	W- Limit	15	N/C
16	!OUT16 (internal)	16	OUT16 (internal)

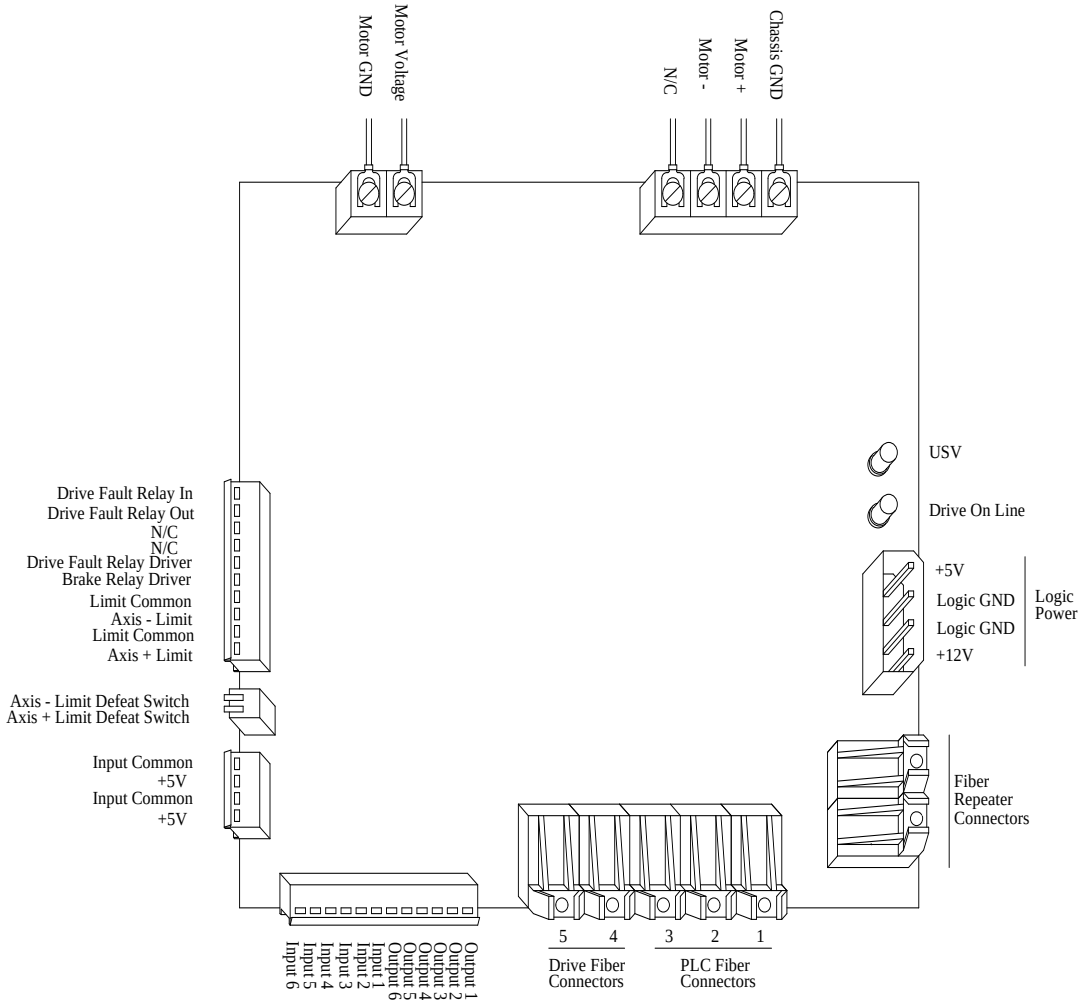
DCSINGLE Specifications

Characteristic	Min.	Typ.	Max.	Unit
5 Volt Supply Current	1.0	-	-	A
12 Volt Supply Current	0.5	-	-	A
Input Pull up Voltage	-	5	-	V
Relay Output Current	0.01	-	10	A @ 125 Vac
Open Collector Output Current	-	-	500	mA
Open Collector Output Voltage	-	5	-	V
Input Operating current	9	11	15	mA
Motor Output Current	6	12	15	A
Motor Supply Voltage	30	115	130	V
Size: 9 * 6.625 * 1.875 (W*D*H)				Inches

DCSINGLE Troubleshooting

Symptom	Possible Cause	Corrective Action
USV LED out	No motor voltage	Measure voltage at input terminals, check contactors, wiring, and fuses accordingly
	Insufficient motor voltage	Voltage should be over 30 Vdc
Drive Online LED out	Motion control card hasn't booted up	Start software, wait for the main screen to load
	Logic power not applied	Measure +5 Vdc and +12 Vdc at the connector, correct wiring or supply problems
	"Servo Power Removed" due to fault	Restart system to reset runaway or other serious fault condition
	Fibers 4 and 5 connected incorrectly or faulty	Check connections one at a time, swap with a known good set of fibers
Secondary drive not working	Fiber repeaters 4 and 5 connected incorrectly or faulty	Check connections one at a time, swap with a known good set of fibers
LEDs on, but motor doesn't run	Fuse F1 blown	Check fuse with a meter, replace as necessary
	Limits tripped	Push down the limit defeat switches

DCSINGLE Connections



DC3IO Revision 040914 User Guide

The DC3IO is a three axis DC brush motor drive with an integrated PLC. A range of motor drive currents may be ordered, although 12 and 15 amps per axis configurations are the most popular. The PLC section includes 30 inputs, 31 digital outputs, and one analog output (see the PLC section for details). The DC3IO is operated by a CPU10 or compatible motion control card equipped with an IO2PIC. The DC3IO is an upgrade from the SERVO3IO, optimized for Centroid "S" series control systems.

DC3IO Features

Drive Application:	DC Brush Motors
Number of Axes:	3
Current rating per axis:	3 to 15 Amps
Motor Voltage:	30 to 120 Volts
PLC Inputs:	30
PLC Outputs:	31
Spindle Analog resolution:	12 bits
Control Interface:	5 fiber optics to CPU10 compatible motion control card with IO2PIC
Dimensions (W*D*H):	16 * 8 * 5.25 inches

Drive Section

The DC3IO drive section is based on Centroid's proven DC brush motor drive technology. Several built in features allow for easy integration with a variety of hardware.

Each axis can be built with a range of current ratings determined by the windings on the current sensor. Current ratings of 3, 6, 9, 12, and 15 amps can be provided on the DC3IO.

Open collector output drivers are provided for a brake on each axis (see "DC3IO Connections" and "PLC Section" for wiring details). The brake output drivers can be wired to a 5 volt relay to release motor brakes when each axis is enabled.

A drive fault relay output is provided for connection of the E-stop power loop. The relay contacts stay closed as long as valid data is received on drive fibers 4 and 5 and no serious faults exist.

An analog current request output is provided on the 3rd (Z on a mill) axis for running third party drives. This feature is particularly useful for C axis lathe applications. The current request signal swings from –10 volts to +10 volts and is centered at 0 volts. This signal is used for spindle control in positioning mode. See the "DC3IO Connections" page to locate the C axis analog and C axis common pins.

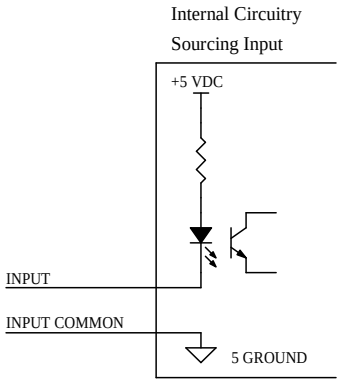
PLC Section

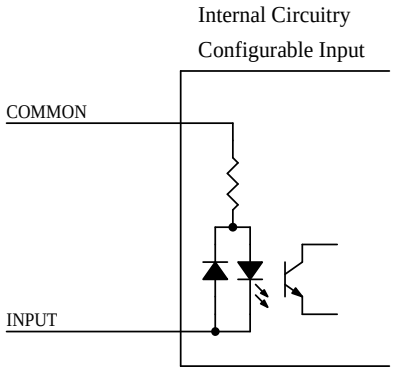
The DC3IO has 30 inputs, 31 digital outputs, and one analog output. Some I/O is dedicated to a particular function, but 21 inputs and 29 digital outputs can be used for any purpose. See the "DC3IO I/O Map" for an overview.

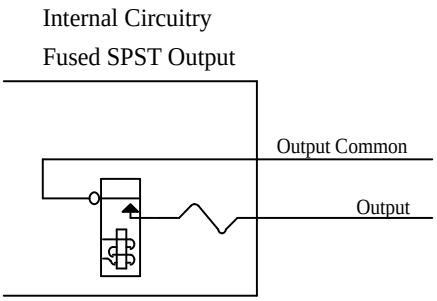
Twelve inputs are configurable types, 16 are sourcing, and 2 are internally wired. The internal drive fault and error check inputs are dedicated and not user definable. X-, X+, Y-, Y+, Z-, and Z+ limit inputs are not configurable for other uses since they are hard wired to drive circuitry that inhibits axis motion. The emergency stop input is also dedicated and has increased pull up current. The 21 remaining inputs can be configured for special purposes if necessary.

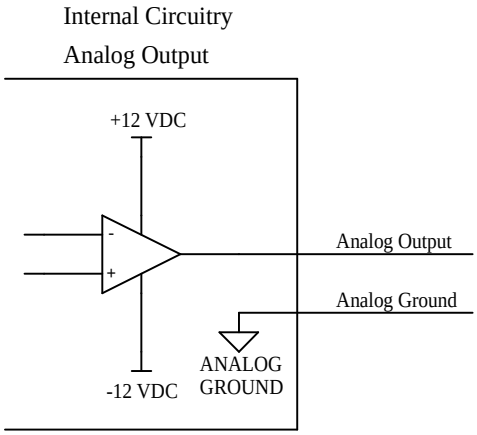
Several output types are used on the DC3IO. Relay outputs are provided for common functions. Signal relays are used on spindle outputs to provide a reliable connection on low level outputs when connecting an inverter. Fused power relays are provided for the rotary clamp and other higher level outputs. Outputs that are not used on many systems are open collector type. These outputs will usually need to drive an external 5 volt relay to interface with higher power devices. Check the "DC3IO I/O Map" and "DC3IO Specifications" sections to determine an output's type and capability. The spindle direction output is not available for other uses. The spindle analog section uses this output to determine polarity when configured as a bipolar output (-5 to +5 or –10 to +10). Internal error checking and spindle speed bits are also dedicated, leaving 29 outputs definable for custom uses.

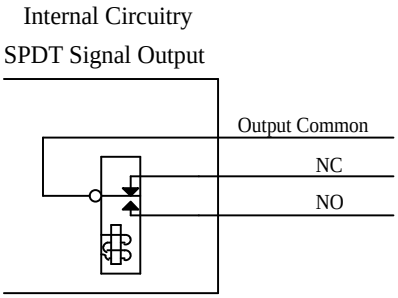
The DC3IO analog output for spindle control has a 12bit resolution. This should not be confused with the C axis analog output described in the "Drive Section". Four analog output ranges can be selected. See the "Spindle Analog Output Adjustment" section for jumper settings.

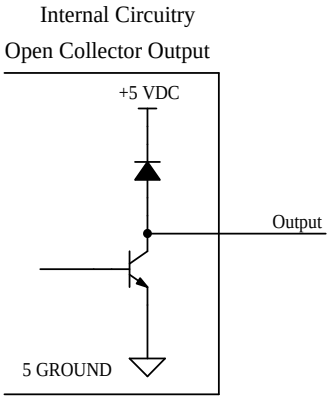


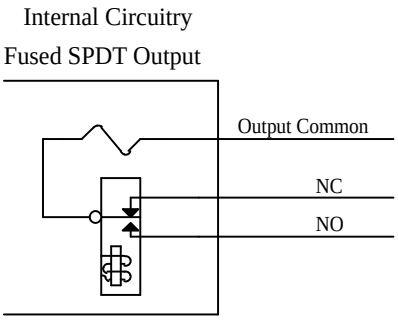










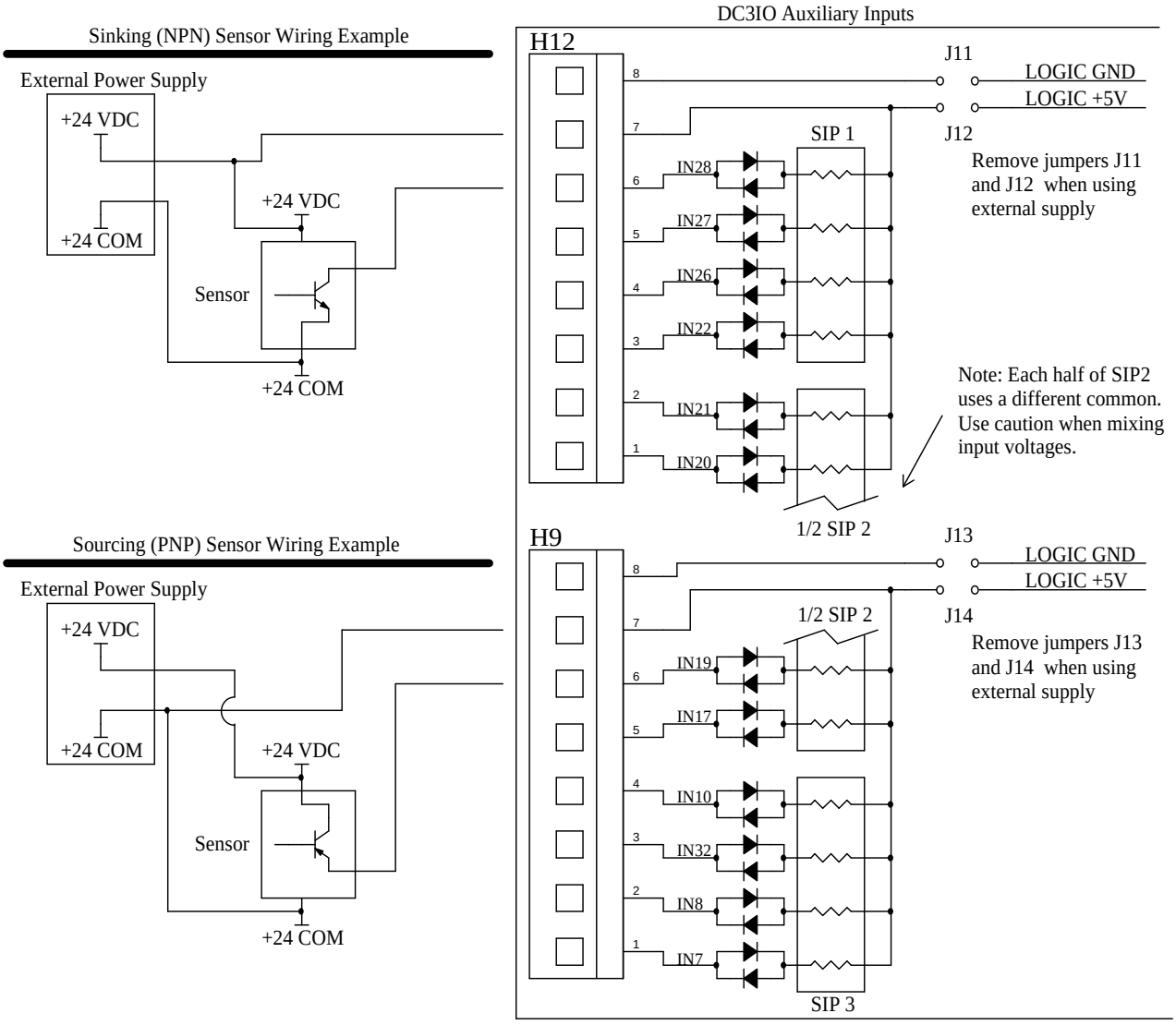


Auxiliary Configurable Inputs

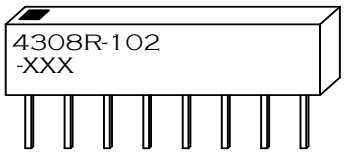
Configurable inputs are available through the auxiliary input connectors for custom applications. These inputs can be used with 5, 12, or 24 Vdc sensors or switches. Resistor packs SIP1, SIP2, and SIP3 must be changed to match the input voltage for auxiliary inputs. Sinking or sourcing operation is determined by the wiring configuration.

Jumpers J11 through J14 may be installed to power the inputs from the DC3IO's logic power supply. External power may be wired through pins 7 and 8 of H12 and H9. Make sure there are no jumper blocks on J11, J12, J13, or J14 before applying external power or the DC3IO will be damaged.

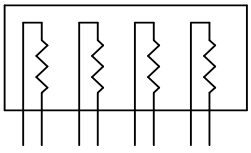
Auxiliary Inputs Schematic



SIP Identification - XXX Indicates Value



SIP Internal Wiring / Pinout



SIP Input Reference

SIP Designator	Related Inputs
SIP1	11,26,27,28
SIP2	17,19,20,21
SIP3	7,8,32,10

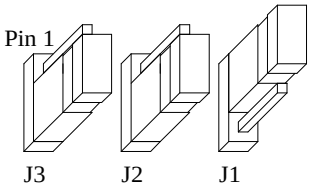
SIP Input Voltage Selection

SIP Value Marking	Resistor Value (Ohms)	Input Voltage
471	470	5
122	1.2k	12
222	2.2k	24

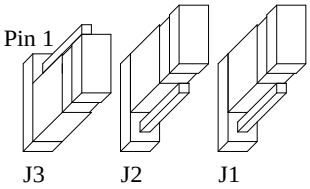
Spindle Analog Output Adjustment

Output voltage range can be set to 0 to +5 Vdc, 0 to +10 Vdc, -5 to +5 Vdc, or -10 to +10 Vdc by setting jumpers J1, J2, and J3 according to the diagrams below. Trimming the output can be accomplished with VR6 and VR7 potentiometers. See the "DC3IO Connections" diagram for the location of adjustment hardware. The analog levels are adjusted at the factory for the 0 to +10 Vdc range, so only slight adjustments should be needed for each installation. Only adjust the "OFFSET" potentiometer, VR6, at the minimum possible spindle speed. This adjustment is intended only to null the voltage level when 0 RPM is commanded. The "GAIN" pot, VR7, should be used at maximum speed to match actual RPM with commanded RPM. Adjustments to the analog output should be very minor and cannot be used to compensate for incorrect inverter or control settings.

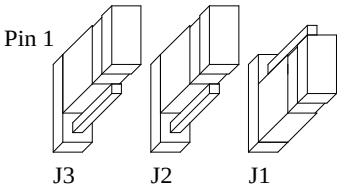
0 to 10 Vdc Jumper Settings



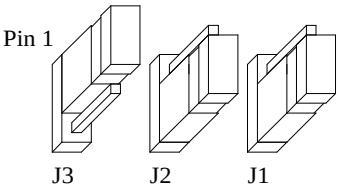
0 to 5 Vdc Jumper Settings



-5 to +5 Vdc Jumper Settings



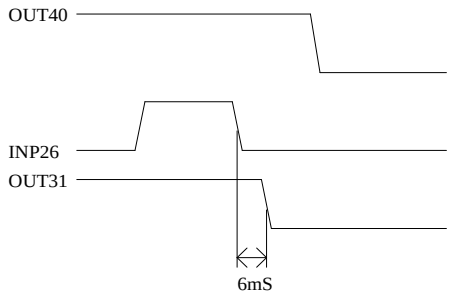
-10 to +10 Vdc Jumper Settings



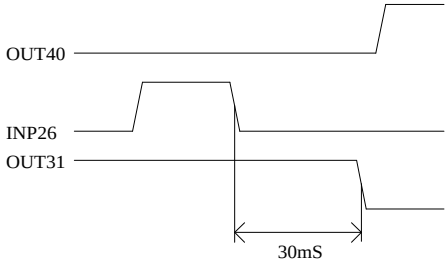
Fast I/O Operation

The Fast I/O is a hard-coded function that is enabled when output 40 is turned on in the PLC program. The Fast I/O immediately turns off output 31 when a falling edge is detected on input 26. This is done immediately before sending any data back to the control. The function is self-resetting - after output 31 is turned off, output 40 must be turned on again in order to reactivate the Fast I/O. The Fast I/O feature was developed to prevent a tool carousel from moving too far due to communication delays. When output 40 is off, output 31 and input 26 work normally. Output 40 is not a physical output, since using this output for purposes other than Fast I/O enable could cause confusion.

Timing Diagram - Fast I/O Enabled



Timing Diagram - Normal PLC Operation



DC3IO I/O MAP

Input Specification				Input Location	
Number	Function	Type	Use	Connector	Pin
1	X- Limit	Sourcing	Dedicated	H14	1
2	X+ Limit	Sourcing	Dedicated	H14	2
3	Y- Limit	Sourcing	Dedicated	H14	4
4	Y+ Limit	Sourcing	Dedicated	H14	5
5	Z- Limit	Sourcing	Dedicated	H14	7
6	Z+ Limit	Sourcing	Dedicated	H14	8
7	W- Limit	Configurable	General	H9	1
8	W+ Limit	Configurable	General	H9	2
9	Range	Sourcing	General	H13	11
10	5th+ Limit	Configurable	General	H9	4
11	Emergency Stop	Sourcing	Dedicated	H14	10
12	Servo Drive Fault	Internal	Dedicated	N/A	N/A
13	TT1	Sourcing	General	H14	11
14	Probe	Sourcing	General	H13	1
15	Probe Detect	Sourcing	General	H13	2
16	Error Check	Internal	Dedicated	N/A	N/A
17	Door Interlock	Configurable	General	H9	5
18	Low Lube	Sourcing	General	H13	4
19	Spindle Zero Speed	Configurable	General	H9	6
20	Spindle At Speed	Configurable	General	H12	1
21	Spindle Orient Complete	Configurable	General	H12	2
22	Tool Clamped	Configurable	General	H12	3
23	NOT USED	N/A	N/A	N/A	N/A
24	Tool Release Switch	Sourcing	General	H13	5
25	Spindle Drive Fault	Sourcing	General	H13	7
26	Tool Counter *	Configurable	General	H12	4
27	Carousel Out / TP Up	Configurable	General	H12	5
28	Carousel In / TP Dwn	Configurable	General	H12	6
29	NOT USED	N/A	N/A	N/A	N/A
30	Rotary Home	Sourcing	General	H13	8
31	Rotary Clamped	Sourcing	General	H13	10
32	Air Pressure Low	Configurable	General	H9	3
33	NOT USED	N/A	N/A	N/A	N/A
34	NOT USED	N/A	N/A	N/A	N/A
35	NOT USED	N/A	N/A	N/A	N/A
36	NOT USED	N/A	N/A	N/A	N/A
37	NOT USED	N/A	N/A	N/A	N/A
38	NOT USED	N/A	N/A	N/A	N/A
39	NOT USED	N/A	N/A	N/A	N/A
40	NOT USED	N/A	N/A	N/A	N/A
41	NOT USED	N/A	N/A	N/A	N/A
42	NOT USED	N/A	N/A	N/A	N/A
43	NOT USED	N/A	N/A	N/A	N/A
44	NOT USED	N/A	N/A	N/A	N/A
45	NOT USED	N/A	N/A	N/A	N/A
46	NOT USED	N/A	N/A	N/A	N/A
47	NOT USED	N/A	N/A	N/A	N/A

Output Specification			Output Location	
Number	Function	Type	Connector	Pin
1	Emergency Stop	Open Collector	H7	1
2	Lube Pump	Fused Relay SPST	H9	7,8
3	Flood Pump	Fused Relay SPST	H10	9,10
4	Mist Solenoid	Fused Relay SPST	H11	7,8
5	Carousel Direction	Open Collector	H7	2
6	Carousel Out Solenoid	Open Collector	H7	3
7	Tool Clamp Solenoid	Open Collector	H7	4
8	Air Blow Through	Open Collector	H7	5
9	Carousel In Solenoid	Open Collector	H7	6
10	Orient	Open Collector	H7	7
11	Spindle Chiller	Open Collector	H7	8
12	Spindle Cooling Fan	Open Collector	H6	1
13	Spindle Direction	Signal Relay SPDT	H10	1,2,3
14	Spindle Enable	Signal Relay SPDT	H10	4,5,6
15	Inverter Reset	Signal Relay SPDT	H11	1,2,3
16	Error Check	Internal	N/A	N/A
17	Spin. Speed Bit 0	Internal	N/A	N/A
18	Spin. Speed Bit 1	Internal	N/A	N/A
19	Spin. Speed Bit 2	Internal	N/A	N/A
20	Spin. Speed Bit 3	Internal	N/A	N/A
21	Spin. Speed Bit 4	Internal	N/A	N/A
22	Spin. Speed Bit 5	Internal	N/A	N/A
23	Spin. Speed Bit 6	Internal	N/A	N/A
24	Spin. Speed Bit 7	Internal	N/A	N/A
25	Spin. Speed Bit 8	Internal	N/A	N/A
26	Spin. Speed Bit 9	Internal	N/A	N/A
27	Spin. Speed Bit 10	Internal	N/A	N/A
28	Spin. Speed Bit 11	Internal	N/A	N/A
29	Gear Change	Open Collector	H6	2
30	Rotary Clamp Solenoid	Fused Relay SPDT	H11	4,5,6
31	Carousel Enable *	Open Collector	H6	3
32	Red Light	Open Collector	H6	4
33	Green Light	Open Collector	H6	5
34	Yellow Light	Open Collector	H6	6
35	Work light	Open Collector	H6	7
36	Auxiliary 1	Open Collector	H6	8
37	Auxiliary 2	Open Collector	H4	1
38	Auxiliary 3	Open Collector	H4	2
39	Auxiliary 4	Open Collector	H4	3
40	Fast I/O Enable *	N/A	N/A	N/A
41	NOT USED	N/A	N/A	N/A
42	NOT USED	N/A	N/A	N/A
43	NOT USED	N/A	N/A	N/A
44	NOT USED	N/A	N/A	N/A
45	NOT USED	N/A	N/A	N/A
46	NOT USED	N/A	N/A	N/A
47	NOT USED	N/A	N/A	N/A

48	NOT USED	N/A	N/A	N/A	N/A
49	NOT USED	N/A	N/A	N/A	N/A
50	NOT USED	N/A	N/A	N/A	N/A
51	NOT USED	N/A	N/A	N/A	N/A
52	NOT USED	N/A	N/A	N/A	N/A
53	NOT USED	N/A	N/A	N/A	N/A
54	NOT USED	N/A	N/A	N/A	N/A
55	NOT USED	N/A	N/A	N/A	N/A
56	NOT USED	N/A	N/A	N/A	N/A
57	NOT USED	N/A	N/A	N/A	N/A
58	NOT USED	N/A	N/A	N/A	N/A
59	NOT USED	N/A	N/A	N/A	N/A
60	NOT USED	N/A	N/A	N/A	N/A
61	NOT USED	N/A	N/A	N/A	N/A
62	NOT USED	N/A	N/A	N/A	N/A

* Fast I/O Related

48	NOT USED	N/A	N/A	N/A
49	NOT USED	N/A	N/A	N/A
50	NOT USED	N/A	N/A	N/A
51	NOT USED	N/A	N/A	N/A
52	NOT USED	N/A	N/A	N/A
53	NOT USED	N/A	N/A	N/A
54	NOT USED	N/A	N/A	N/A
55	NOT USED	N/A	N/A	N/A
56	NOT USED	N/A	N/A	N/A
57	NOT USED	N/A	N/A	N/A
58	NOT USED	N/A	N/A	N/A
59	Auxiliary 13	Open Collector	H4	4
60	Auxiliary 14	Open Collector	H4	5
61	Auxiliary 15	Open Collector	H4	6
62	Auxiliary 16	Open Collector	H4	7
	Drive Fault	Power Relay SPST	H11	9,10
	X Brake	Open Collector	H3	1
	Y Brake	Open Collector	H3	2
	Z Brake	Open Collector	H3	3

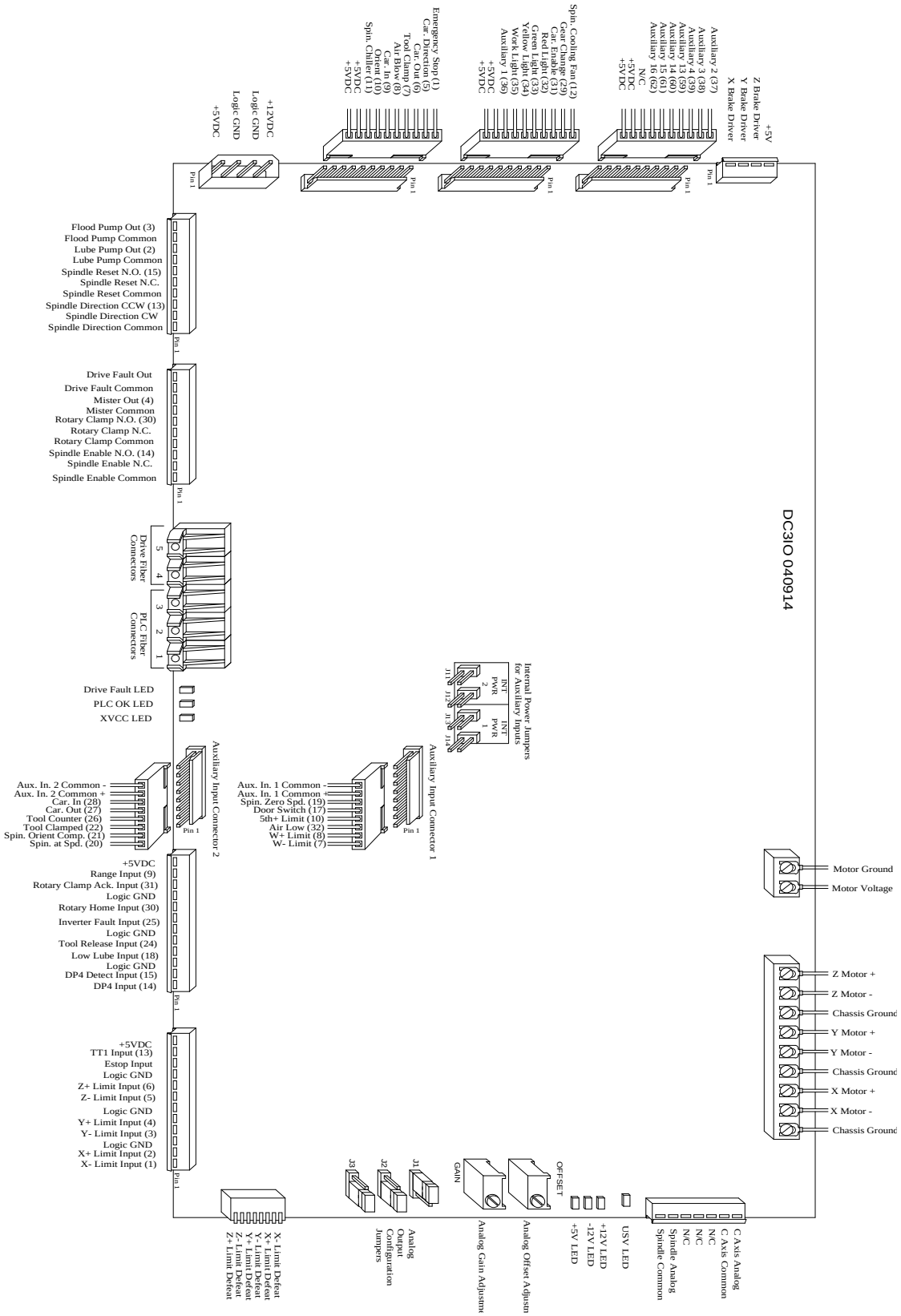
DC3IO Specifications

Characteristic	Min.	Typ.	Max.	Unit
5 Volt Supply Current	2	-	-	A
12 Volt Supply Current	0.5	-	-	A
Input Pullup Voltage	-	5	-	V
Power Relay Output Current	0.01	-	10	A @ 125 Vac
Power Relay Output Current	0.01	-	5	A @ 30 Vdc
Signal Relay Output Current	0.001	-	0.5	A @ 125 Vac
Signal Relay Output Current	0.001	-	1	A @ 24 Vdc
Open Collector Output Current	-	-	500	mA
Open Collector Output Voltage	-	5	-	V
Input Operating current	9	11	15	mA
Analog Output Resolution	-	12	-	bits
Analog Output Voltage	0	-	10	V
Analog Output Current	0	1	20	mA
Motor Output Current	6	12	15	A
Motor Supply Voltage	30	115	130	V
Dimensions: 16 * 8 * 5.25 (W*D*H)				Inches

DC3IO Troubleshooting

Symptom	Possible Cause	Corrective Action
All status LEDs out	Logic power not applied	Measure +5 Volt and +12 Volt at the connector, correct wiring or supply problems
USV LED out	No motor voltage	Measure voltage at input terminals, check contactors, wiring, and fuses accordingly
	Insufficient motor voltage	Voltage should be over 30 Vdc
DF LED out	Motion control card hasn't booted up	Start software, wait for the main screen to load
	Fibers 4 and 5 connected incorrectly or faulty	Check connections one at a time, swap with a known good set of fibers
	"Servo Power Removed" due to fault	Restart system to reset runaway or other serious fault condition
	Incorrect .HEX file	Make sure CNC8.HEX is loading
PLC OK LED out	Motion control card hasn't booted up	Start software, wait for the main screen to load
	Fibers 1, 2, or 3 connected incorrectly or faulty	Check connections one at a time, swap with a known good set of fibers
	Incorrect PIC on CPU10	Install IO2PIC
LEDs on, but motor doesn't run	Axis Fuse blown	Check fuses with a meter, replace as necessary
	Limits tripped	Push down the limit defeat switches
No analog output or non-linear output	Incorrect Parameter 31 setting	Set P31 to -1
XVCC LED out	Overload has damaged PLC section	Return for Repair
+12, -12, or +5 LED out	Overload has damaged analog section	Return for Repair
Input doesn't work with sensor	Incorrect wiring	Correct wiring for sensor type (sinking or sourcing), check that SIP values are appropriate for the input voltage
	Voltage drop across sensor is too high	Use 3-wire sensors with lower voltage drop spec.

DC3IO Connections



PLC Installation

PLC 15 / 15

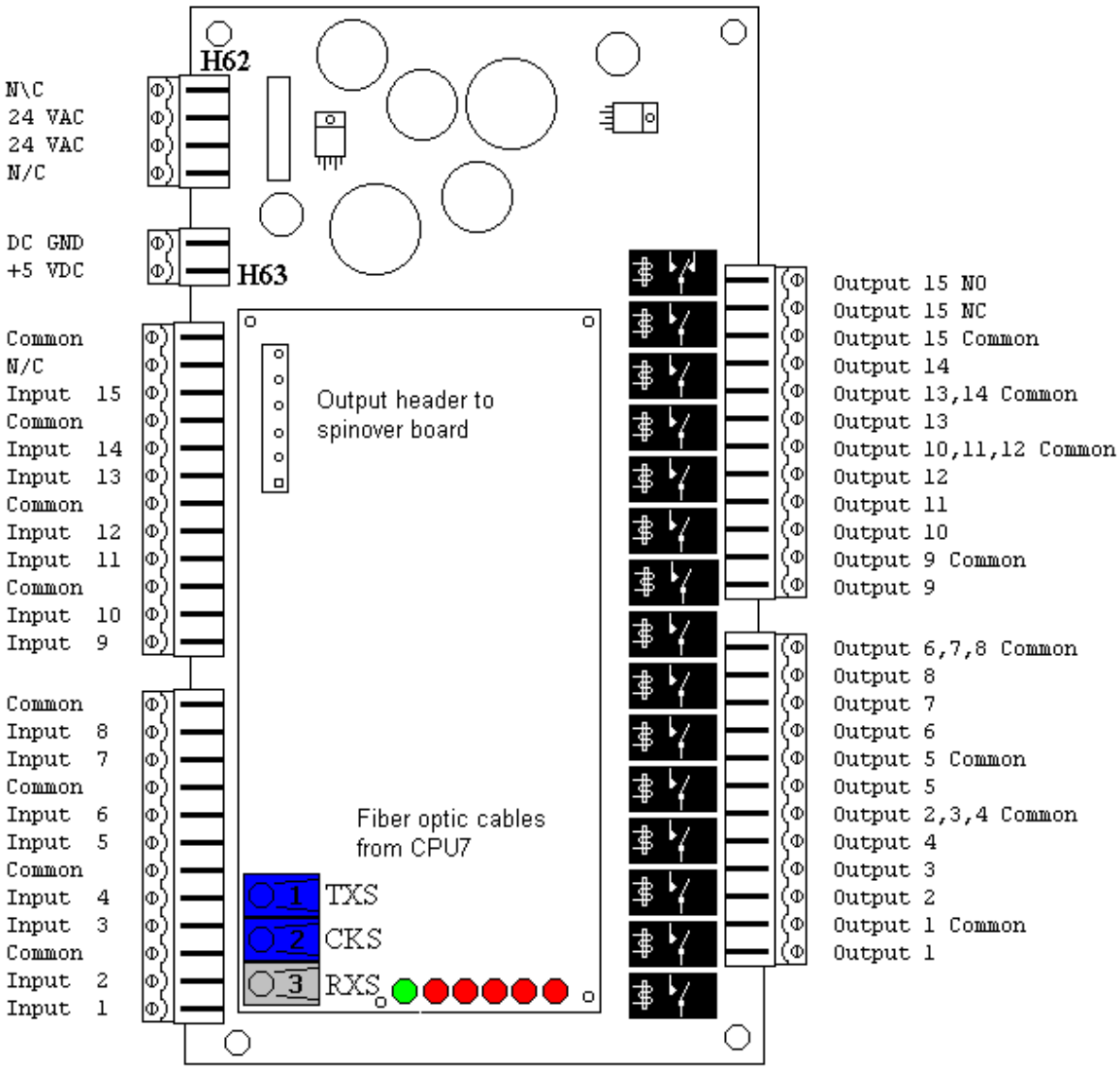
PLC15/15 is a PLCI/O relay board that provides the following I/O:

- 15 5 Vdc inputs
- 14 SPST relay outputs
- 1 DPST relay output
- Optional 0-10 Vdc spindle output

Power Connections

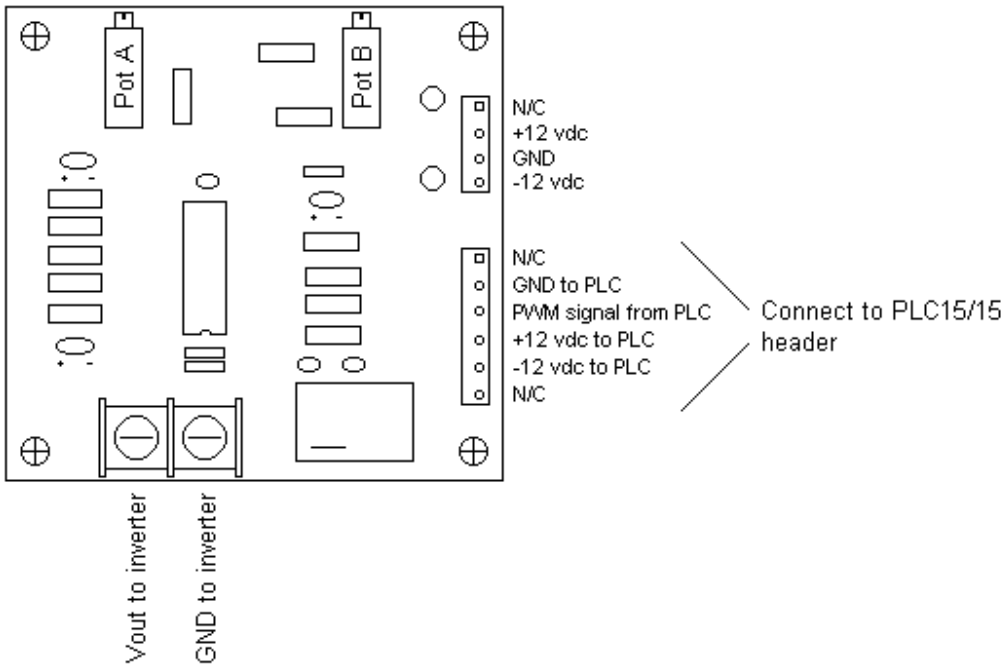
The PLC15/15 requires a 5 Vdc and a 24 Vac power source.

PLC 15/15 Hookup Diagram



Spinover Board

The Spinover board is an optional add-on for the PLC 15/15 to provide 0-10 volt output for inverter controlled spindles.



Spinover Board Hookup Diagram

PLCIO2

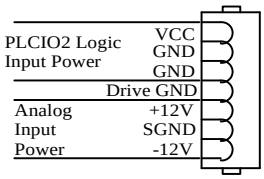
PLCIO2 is a programmable logic controller that provides:

- 35 Inputs (bipolar, with a choice of 5 Volts or 24 Volts)
- 39 Outputs (20SPST, 2 SPDT, 17 open collector)
- 1 Analog output (12 bits for spindle control)

PLCIO2 is compatible with our standard fiber optic (3 fibers) interface, but requires a special PIC chip (PLCIO2 PIC) on the CPU10 CNC motion control card.

Power Connections

The PLCIO2 requires a 5 Volt logic power source along with a +/- 12 Volt analog power source. The grounds from each power supply are NOT to be connected together. Doing so would negate the isolation and possibly cause noise from the spindle speed inverter to be transmitted to other parts of the PLCIO2 board.



H1 Power Input

Wiring Inputs to the PLCIO2

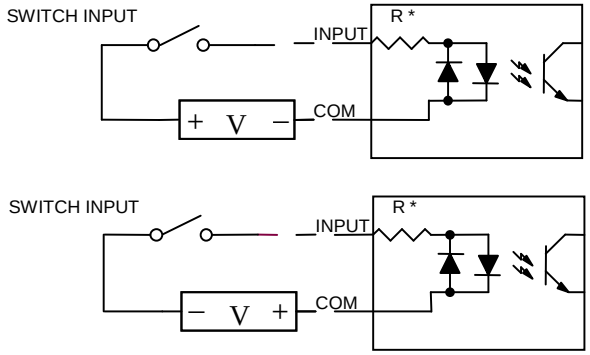
The PLCIO2 can accept contact closures, PNP, and NPN type inputs. Inputs are grouped into banks of four with the common for that particular bank on the fifth pin. NPN and PNP sensors must use separate banks for proper operation. Switches can be wired so they resemble either type of sensor so they can share banks with sensors of similar operation.

Wiring to the Relay Outputs

The PLCIO2 has a common and a normally open contact for each output. Outputs 15 and 32 also have available a normally closed contact in addition to the common and normally open contacts. The relays on the PLCIO2 are rated for:

- 5 A @ 30 Vdc
- 0.3 A @ 110 Vdc
- 10 A @ 125 Vac
- 8 A @ 277 Vac

Typical PLCIO2 Output Relay



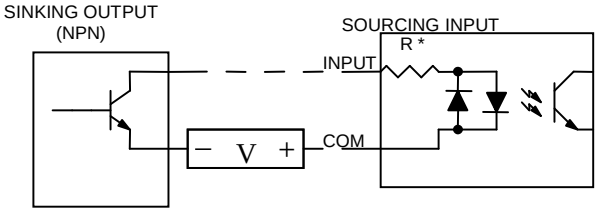
*Note: V is user supplied 5 or 24 volts. (See next section)

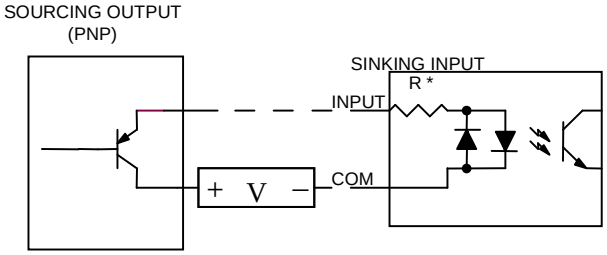
Determining Input Voltage

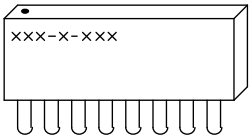
The input voltage accepted by each bank, +V, is determined by the value of the SIP that is associated with each bank These sips are normally in sockets and are located directly above the input headers. To determine which voltage a specific bank is set up for look at the last 3 digits on the SIP. These numbers are either located on the top or the front of the SIP.

- 471 = 470 ohms –> 5 Volt bank
- 222 = 2.2K ohms –> 24 Volt Bank

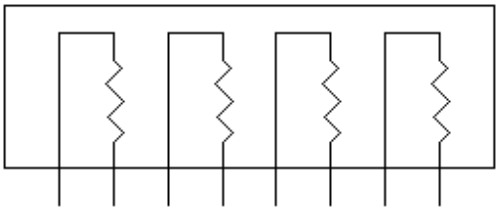
These resistor values produce about 10 mA across the optocoupler inputs. Replacing a SIP with a different value SIP can change the input voltage range that the PLCIO2 takes. If a SIP is not available, four individual resistors can be used in its place.



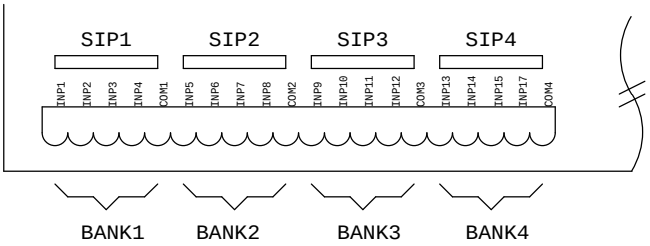




Typical SIP



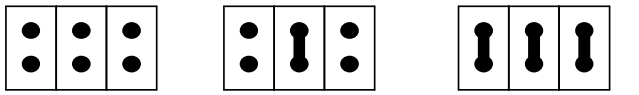
Internal SIP diagram



Input SIP Locations

Limit Switch Power Source

There is a jumper block located above the limit switch header on the PLCIO2. This is for the limit switch inputs. If the limit switch cable is not used then no jumpers should be in place. If the limit switch cable is used then see the diagram to the right. Note: it is not recommended that the PLCIO2 supply the power for the limit switches, as it will negate the isolation of the inputs.



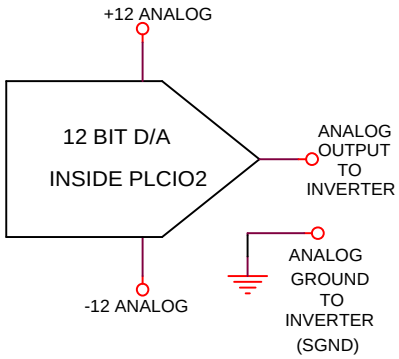
Limit Switch Cable Not Used | External Supply Needed for Limit Switches | PLCIO2 Supplies 5 Volts for Limit Switches

Jumper settings

Analog output for Spindle

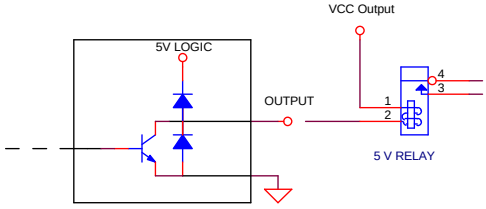
The PLCIO2 has a 0-10 Volt analog output for inverters. This output is located on the input side of the PLCIO2 next to the power header. These pins get connected to the analog input of the inverter. Do NOT connect the ground from this output to any ground other than the one for which it was intended.

Typical DAC output



Wiring Transistor Outputs

The PLCIO2 has 17 available transistor outputs. These outputs are to be used with 5 Volt relays to drive any additional signals required. To hook up the relay simply attach one side of the coil to the 5 volt Vcc output and then attach the other side to the output pin. It is not recommended that these outputs be used for any other purpose because they are not noise immune, and improper use of these outputs may cause the PLCIO2 to function improperly.



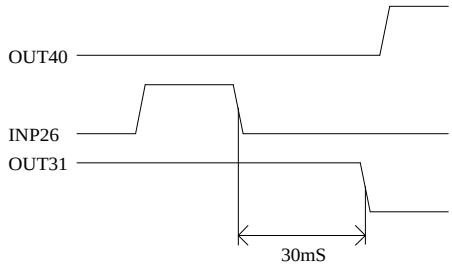
Typical Transistor Output

Fast I/O

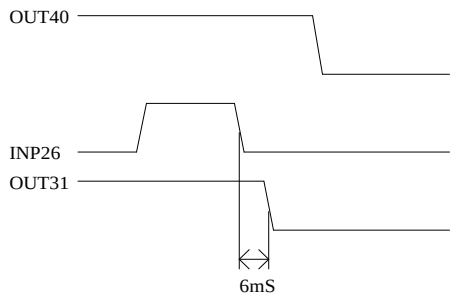
The Fast I/O is a hard-coded function that is enabled when Out40 is turned on in the PLC program. The Fast I/O immediately turns off Out31when a falling edge is detected on Inp26. This is done immediately before sending any data back to the CPU10. This function is self-resetting. This means that after Out31 is turned off the PLCIO2 needs to resend Out40 signal in order to reactivate the Fast I/O. This function was developed to prevent the possibility of a carousel moving too far due to communication delays.

Note: when Out40 is low the Out31 and Inp26 function normally. If you don't want to use Fast I/O simply never turn on Out40.

Timing Diagrams of Fast I/O



Fast I/O Enabled



Normal PLC Operation

Power Supply and Input Specs

Characteristic	Min.	Typ.	Max.	Unit
Logic Supply Voltage (Vcc)	4.5	5	5.5	V
Logic Supply Current	—	.75	2.6	A
DAC Supply Voltages	±11.5	±12	±15	V
DAC Supply Current	—	150	300	mA
Inputs (limit switches)*	—	5	9	V
Inputs (others)	—	24	26	V

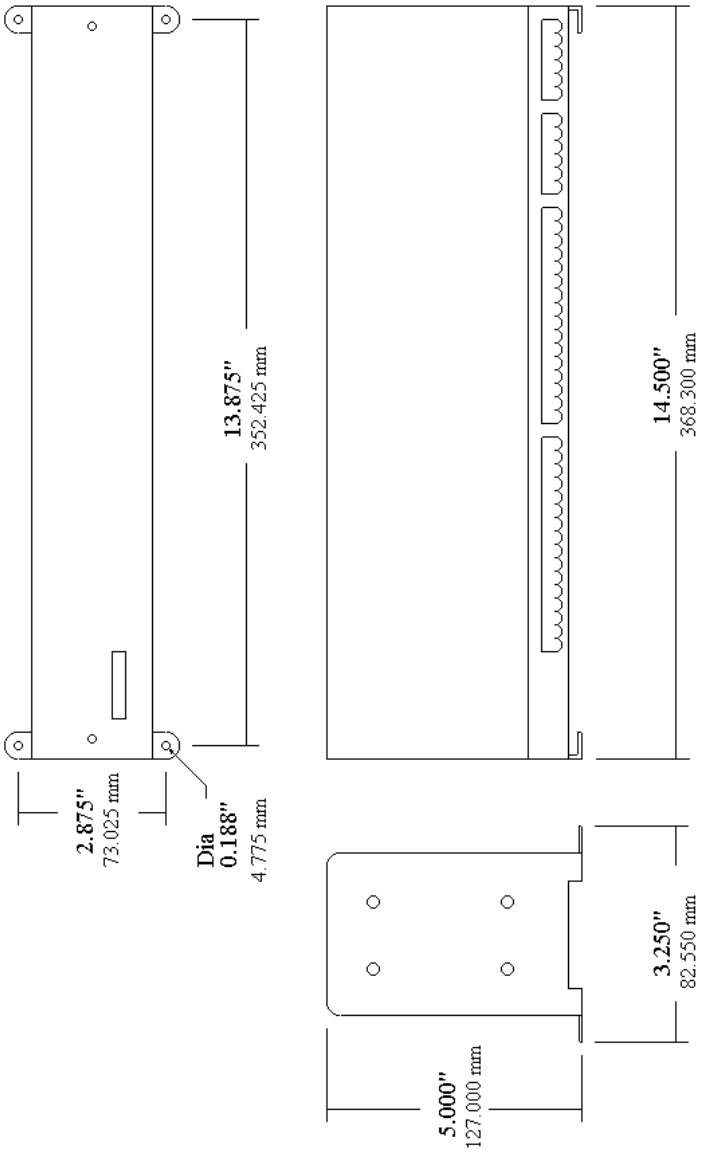
*The normal configuration is the first 8 inputs on the PLCIO2 are set up for 5 Volt limit switches with all other inputs being set up for 24 Volt inputs.

Output Specs

Fiber Optic connections to CPU10

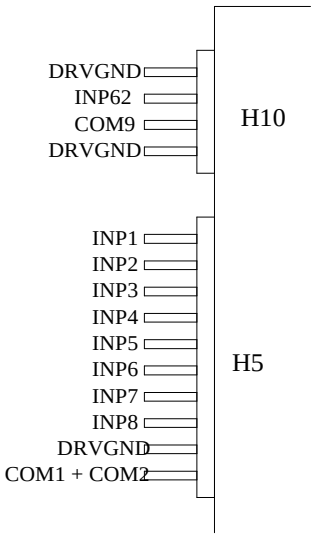
Characteristic	Min.	Typ.	Max.	Unit
Relay Output Voltage	—	24	277	V
Relay Output Current @ 277 Vac	—	—	8	A
Relay Output Current @ 30 Vdc	—	—	5	A
Relay Output Current @ 110 Vdc	—	—	300	mA
Relay Output Current @ 125 Vac	—	—	10	A
Transistor Output current sinking	—	—	400	mA
Transistor Output Voltage	—	5	5.5V	V
Analog output Voltage	0	—	10	V
Analog output Current	—	1	20	mA

PLCIO2 Physical Dimensions

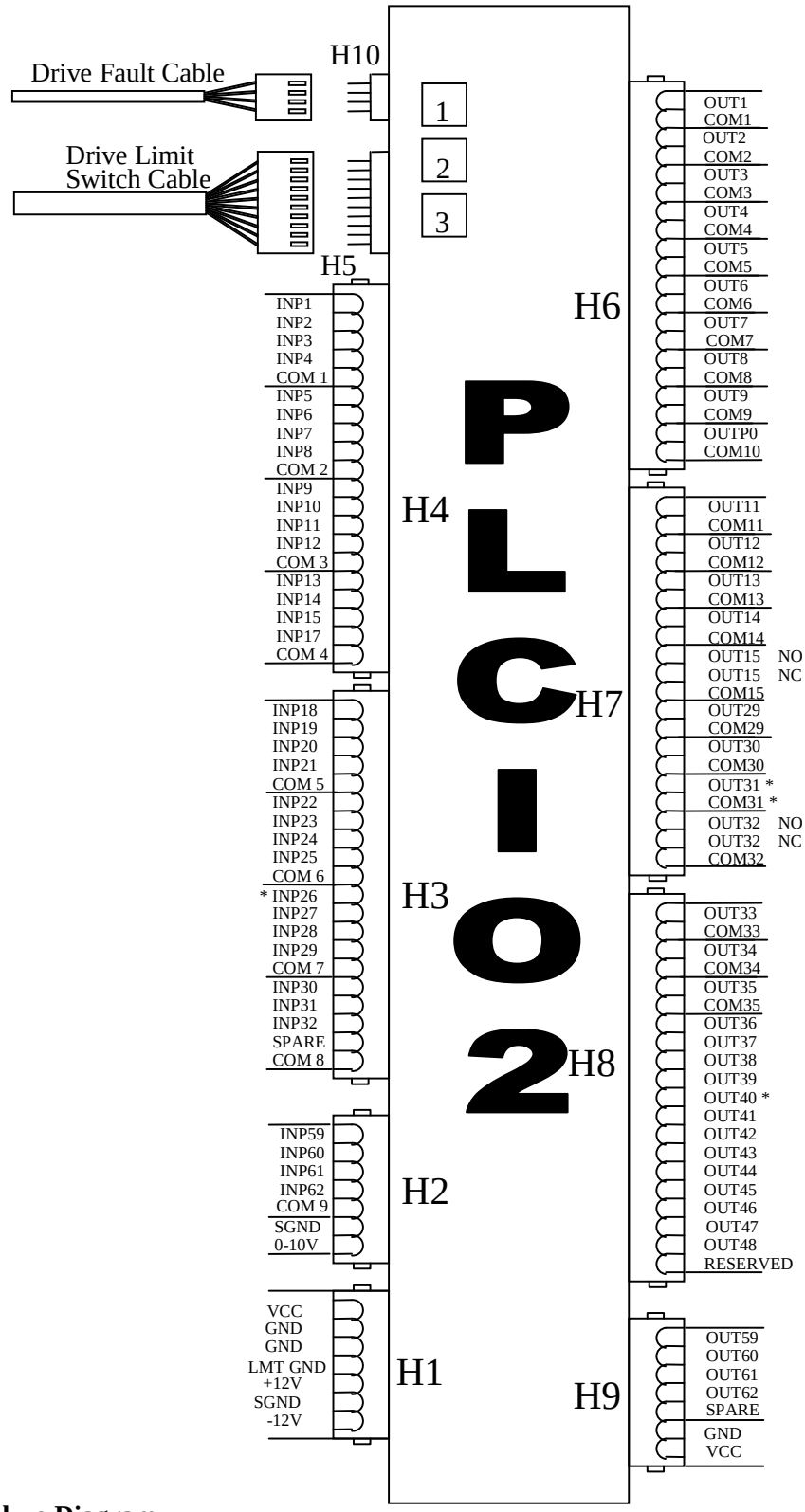


Limit switch header H5 carries duplicates to INP1 thru INP8 and drive fault header H10 carries a duplicate of INP62.

(See figure below)

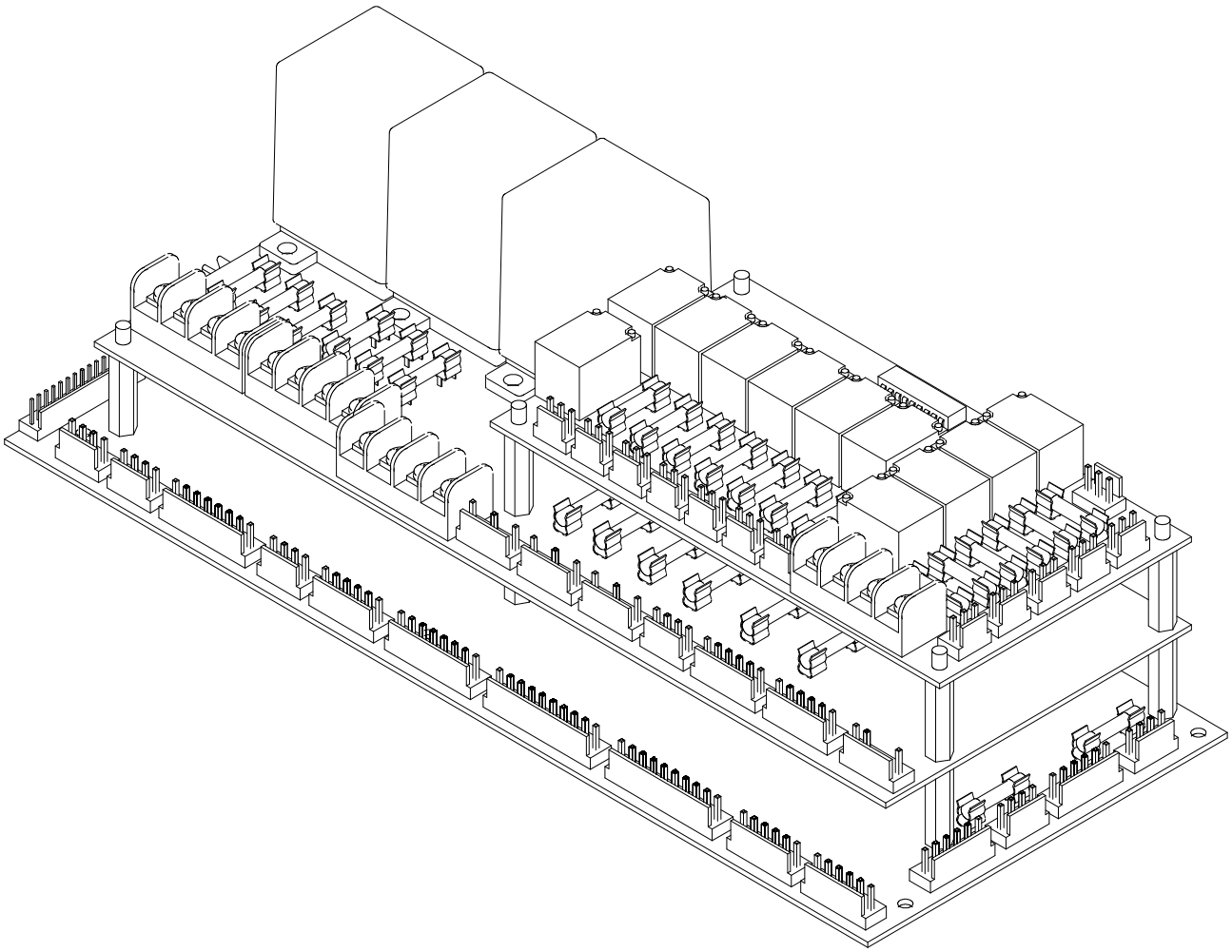


* NOTE: inp26, out31, and out40 have special Fast I/O functions. See the section dealing with Fast I/O for more details



PLCIO2 Hookup Diagram

RTK3 Logic Controller (Revised 11-3-04)



The RTK3 is intended to simplify and expedite control wiring. Centroid's PLCIO2 was the starting point for the RTK3, and modifications were aimed at increasing efficiency on standard system configurations. The goals were to reduce the number of connections in the control cabinet. Inputs and most output voltages have been standardized. Logic and input power supplies are built in to the RTK3 to further simplify installation. Positive locking connectors on pre-assembled cables eliminate the need to individually connect each wire to the RTK3.

There are technical similarities between the PLCIO2 and RTK3. The motion control card, CPU10, connected to the RTK3 must be equipped with an IO2PIC chip, see TB133 for the upgrade procedure. Both PLC compilers, PLCCOMP or XPLCCOMP, can be used with the RTK3. The RTK3 also retains the Fast I/O feature, which is now dedicated to a sourcing sensor input and three-pole relay output.

There are three layers in the RTK3 to minimize footprint and maximize cabinet efficiency. Only rarely used connectors are located on the backside of the unit to allow mounting against the cabinet wall if necessary. The bottom board has six mounting holes for direct attachment to standoffs. See the "RTK3 Mounting Dimensions" section for mounting standoff configuration. The bottom, or logic, board is home to the logic controller's processor, input isolators, analog output section, output drivers, and power supply section. This is where the fiber optic and almost all input connections are made. The middle board has 13 relays, one logic input, and three power inputs. Three large relays control the flood, carousel direction, and carousel enable. Remaining relays control varying voltage and load levels. The top board is equipped with 110 Vac outputs and the 110 Vac power input connector.

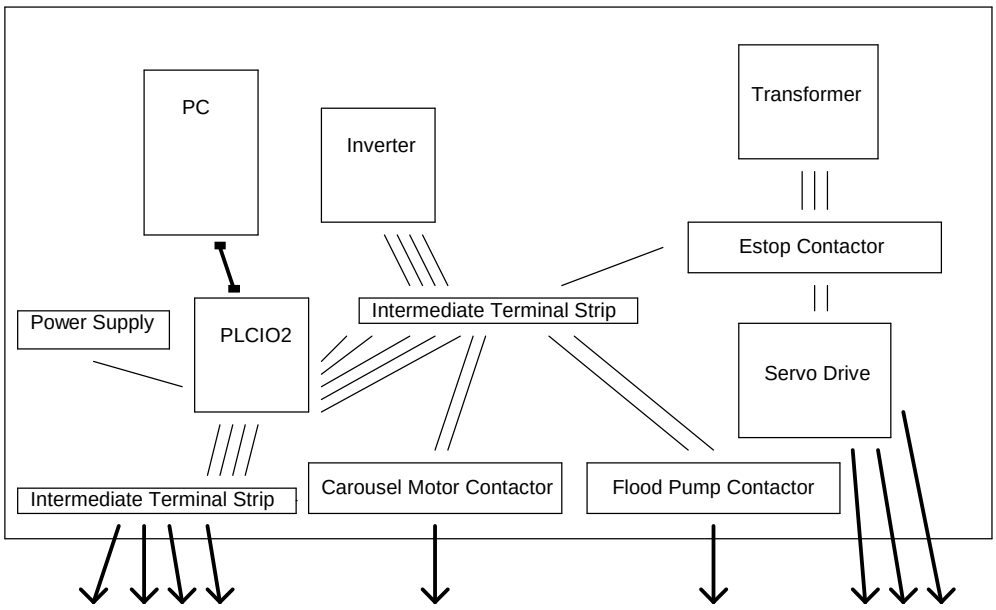
RTK3 System Constraints

Carousel motor and flood pump motor outputs are limited to 250 Vac by the on board fuse ratings. Both of these outputs use the same voltage. To run higher voltages, contact Centroid for alternate wiring options. Single phase carousel and flood pump motors can also be used, please contact Centroid for more information.

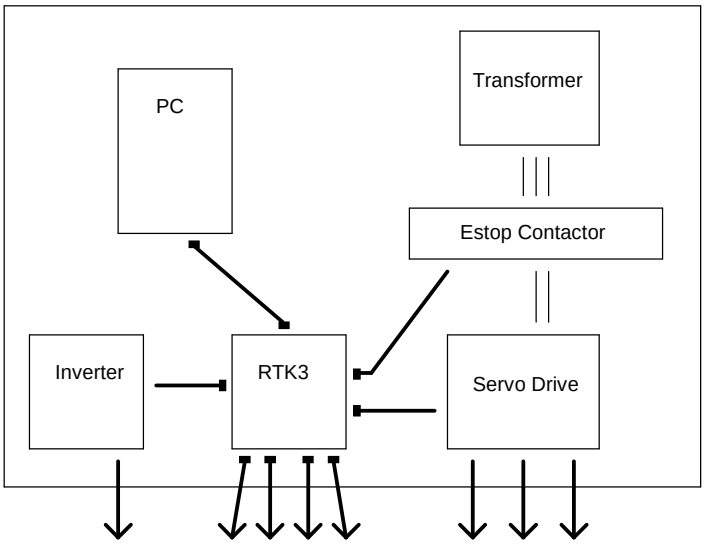
The lube pump output must be either 220 Vac or 110 Vac single phase. In addition, the E-stop power loop must be 24 Vac, solenoids and lights must be 110 Vac, and sensors must be compatible with 24 Vdc input when using an RTK3.

Wiring Concept Visual Comparison

Previous ATC Wiring Sample



RTK3 Wiring Sample



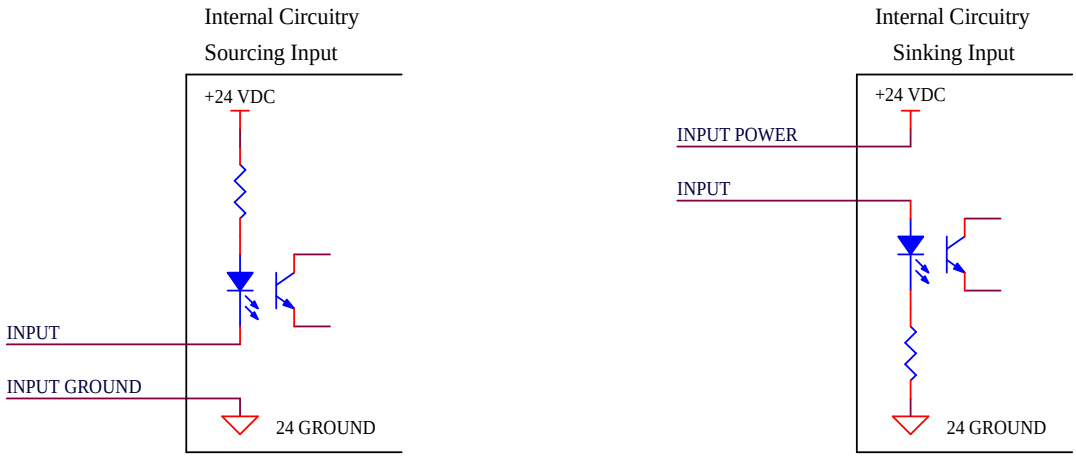
Note the simplified layout and wiring enabled by the RTK3. Fewer individual wires, represented by thin lines, are used and the intermediate terminal block connections are reduced. Cables, represented by thick lines, with quick connect ends replace many individual wires. Integrated components and specialized connection points on the RTK3 reduce the quantity of external hardware required. These advantages allow for quick, organized, compact and reliable panel wiring.

Power Connection

Four power connections are necessary on a full RTK3 installation. Three phase power is connected through a four position screw terminal strip labeled "3 Phase Input." Three phase power is used for the flood pump and the carousel motor outputs; this requires these motors to be the same voltage rating. All other power inputs are plug-in connections using a 5 position plug. Three wires are positioned uniquely in the housing to allow easy identification of different voltages. The 220 Vac input supplies the 220 Vac lube and spindle cooling fan outputs. The 110 Vac input powers the on boards supply as well as 110 Vac outputs. Three pole relay coils and the E-stop output use the 24 Vac input.

Input Wiring

All inputs on the RTK3s are optically isolated and 24 Vdc powered. Some inputs are sourcing and others are sinking, see the "Input Map" section for details. The two basic input configurations are shown below.



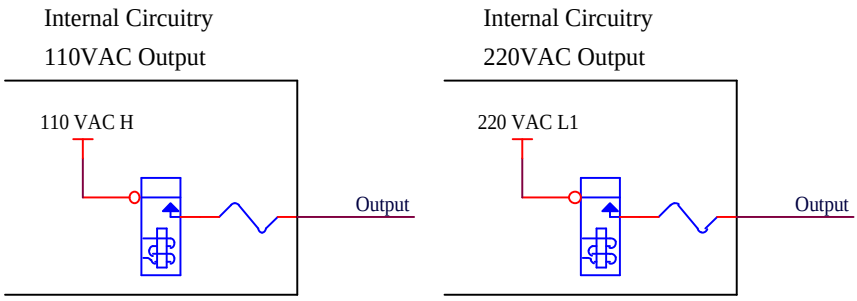
Either type of input may be used with a switch. Sourcing inputs may be used with sinking sensors that are capable of sinking 15 mA at 24 Vdc. Sinking inputs may be used with sourcing sensors that provide 15 mA at 24 Vdc. Care must be taken to wire each input correctly, as the cable for each input has signals that are not used in all cases. These include 24 Vdc and Input Ground for powering sensors and a shield connection that should normally be connected only to the RTK3. The 24 Vdc supply fuse is located on the bottom, or logic, board. The "DIGITIZE PROBE" connector has an additional fuse for the 24 Vdc supply pin, which can also be found on the logic board. These fuses are in series such that the digitizer fuse could fail without affecting other inputs. However, a short in any other 24 Vdc circuit will blow the main 24 Vdc fuse, causing a loss of all input voltages.

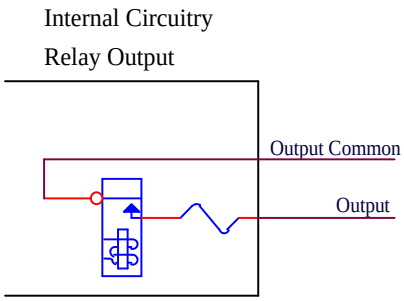
Connectors are available on the RTK3 to plug in limit and fault inputs from a Centroid DC brushed servo drive. The "DRV FAULT" and "LIMITS TO DRIVE" headers are intended only for allowing straight through cable connections to a drive. Limit switch wiring should be done from the "X- / X+ / Y- / Y+ LIMITS" and "W+ / W- / 5+ / 5- LIMITS" headers. When using a Centroid DC brushed servo drive with an RTK3, the "G" and "5" power DIP switches on the drive should be turned on.

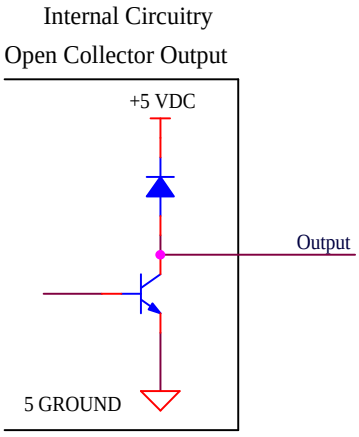
Output Wiring

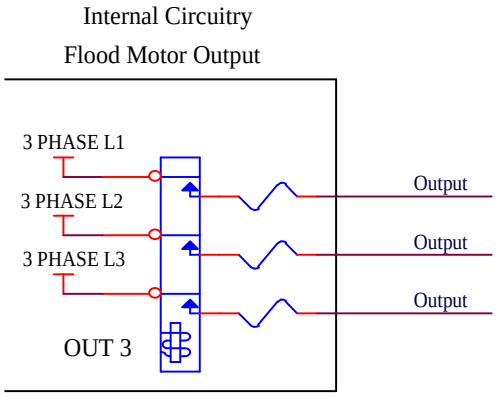
Several output types are present on the RTK3 to reduce wiring time when interfacing with specific components. These include 110 Vac, 220 Vac, 3 phase outputs, 24 Vac, and an analog output. Open collector and relay contact outputs are also available for more general application. See the "Output Map" section for more information.

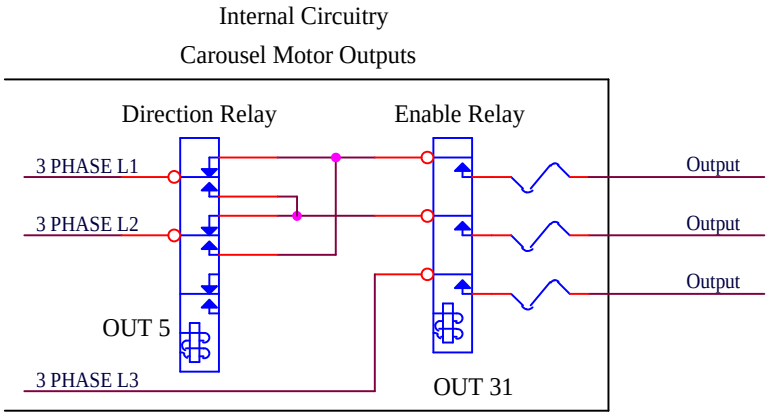
Each power relay output is protected by a fuse. The "DOOR FAN" and "CONSOLE" 110 Vac outputs are also fuse protected. Fuses are 5x20mm and should be rated equal to or less than the ratings printed on the RTK3 for safety. Signal relays, such as those on the inverter outputs, are not fuse protected.





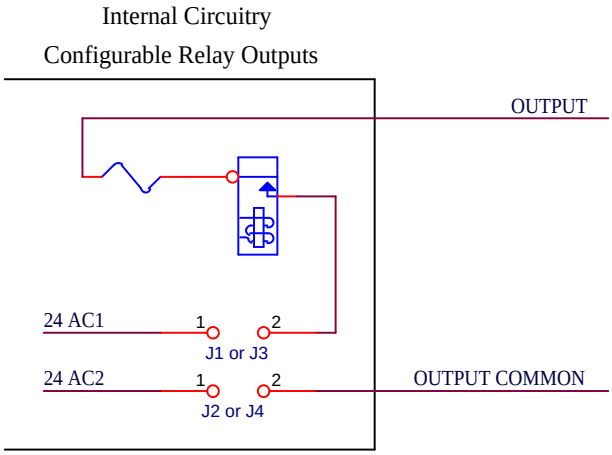




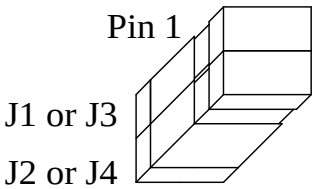


Configurable Outputs

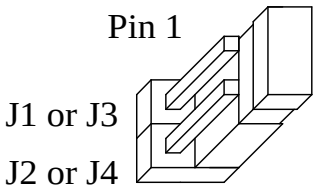
Outputs 11 and 37 may be configured as 24 Vac or relay contact outputs. These outputs are routed to the "AUX 2 / CHILLER" connector. The configurable output jumper settings must be verified before connecting these outputs to prevent damage to the RTK3 or associated hardware.



24 Vac Output Setting

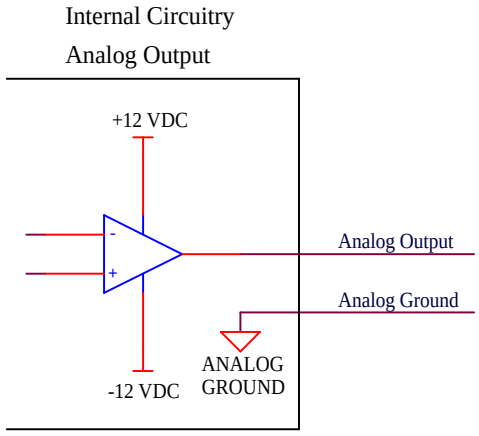


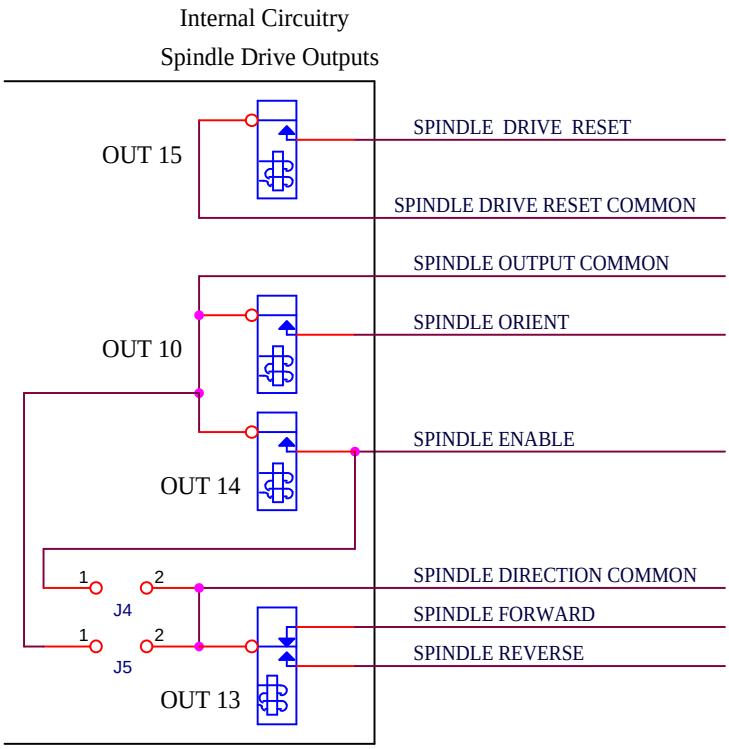
Relay Contact Output Setting



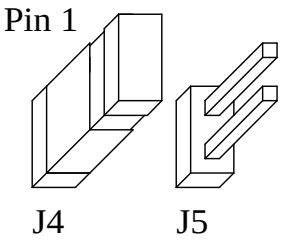
Spindle Outputs

The RTK3 is equipped with an analog output and specialized relay outputs to simplify spindle drive wiring. The analog output has a 12-bit resolution and can be configured for 0 to +5 Vdc or 0 to +10 Vdc output. The analog ground should not be connected to other grounds on the RTK3. All input and output functions required for a standard inverter connection are located on the 15 pin "Inverter" connector.

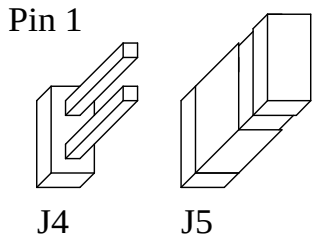




Jumper Setting for Direction Run through Enable



Jumper Setting for Direction Pulled to Common

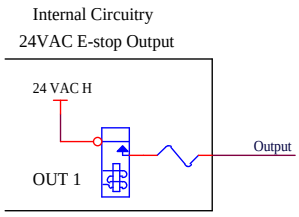


Analog Output Adjustment

Output voltage range can be set to 0 to +5 Vdc or 0 to +10 Vdc by setting jumpers J1, J2, and J3 according to the diagrams below. Trimming the output can be accomplished with VR1 and VR2 potentiometers. See the "RTK3L (Bottom) Board Connections" diagram for the location of adjustment hardware. The analog levels are adjusted at the factory for the 0 to +10 Vdc ranges, so only slight adjustments should be needed for each installation. Only adjust the "OFFSET" potentiometer, VR2, at the minimum possible spindle speed. This adjustment is intended only to null the voltage level when 0 RPM is commanded. The "GAIN" pot, VR1, should be used at maximum speed to match actual RPM with the commanded RPM. Adjustments to the analog output should be very minor and cannot be used to compensate for incorrect inverter or control settings.

Emergency Stop Connector

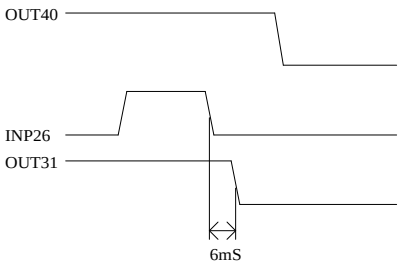
The E-stop connector has an input and output, unlike most RTK3 connectors. The input notifies the control of an external E-stop press. The output allows an E-stop to be triggered by the PLC program. Because the E-stop connector sources 24 Vac, a 24 Vac contactor coil must be used. This reduces the number of connections at the cost of some versatility.



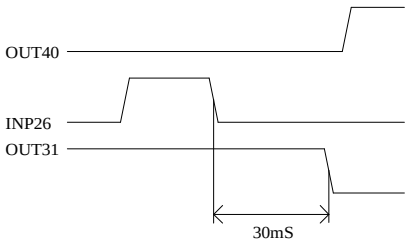
Fast I/O Operation

The Fast I/O is a hard-coded function that is enabled when output 40 is turned on in the PLC program. The Fast I/O immediately turns off output 31 when a falling edge is detected on input 26. This is done immediately before sending any data back to the control. The function is self-resetting - after output 31 is turned off, output 40 must be turned on again in order to reactivate the Fast I/O. Using the Fast I/O prevents the carousel from moving too far due to communication delays. When output 40 is off, output 31 and input 26 work normally. In the RTK3 implementation of the Fast I/O, input 26 is dedicated to a sourcing tool counter sensor and output 31 controls the 3 pole carousel enable relay directly. Output 40 is not a physical output, since using this output for purposes other than Fast I/O enable could cause confusion.

Timing Diagram - Fast I/O Enabled



Timing Diagram - Normal PLC Operation



Status Indicator LEDs

Six green indicator LEDs (lights) located on the bottom board display RTK3 status. All six LEDs will be lit under normal operating conditions. Five of the LEDs indicate the presence of required operating voltages. The 110 Vac input feeds the power supply to generate these voltages. The “PLC OK” LED lights when the RTK3 and motion control card, CPU10, have a working communication link. This LED will not be lit as soon as power is applied. After the control software, CNC10, has initialized the motion control card, communication can be established with the RTK3, which lights the “PLC OK” LED.

Input Map

Input Specification			Connection Location		
Number	Function	Type	Board	Connector	Pin
1	X- Limit	Sourcing	Bottom	X- / X+ / Y- / Y+ LIMITS	1
2	X+ Limit	Sourcing	Bottom	X- / X+ / Y- / Y+ LIMITS	3
3	Y- Limit	Sourcing	Bottom	X- / X+ / Y- / Y+ LIMITS	5
4	Y+ Limit	Sourcing	Bottom	X- / X+ / Y- / Y+ LIMITS	7
5	Z- Limit	Sourcing	Bottom	Z+ / Z- LIMITS	1
6	Z+ Limit	Sourcing	Bottom	Z+ / Z- LIMITS	3
7	W- Limit	Sourcing	Bottom	W+ / W- / 5+ / 5- LIMITS	1
8	W+ Limit	Sourcing	Bottom	W+ / W- / 5+ / 5- LIMITS	3
9	5th- Limit	Sourcing	Bottom	W+ / W- / 5+ / 5- LIMITS	5
10	5th+ Limit	Sourcing	Bottom	W+ / W- / 5+ / 5- LIMITS	7
11	Emergency Stop	Sourcing	Bottom	ESTOP	1
12	Servo Drive Fault	Sourcing	Bottom	DRV FAULT	2
13	TT1	Sourcing	Bottom	TT1 / AUX	1
14	Probe	Sourcing	Bottom	DIGITIZE PROBE	1
15	Probe Detect	Sourcing	Bottom	DIGITIZE PROBE	3
16	Error Check	Internal	N/A	N/A	N/A
17	Door Interlock	Sourcing	Bottom	DOOR SWITCH	1
18	Low Lube	Sourcing	Middle	LOWLUBE	1
19	Spindle Zero Speed	Sourcing	Bottom	INVERTER	10
20	Spindle At Speed	Sourcing	Bottom	INVERTER	13
21	Spindle Orient Complete	Sourcing	Bottom	INVERTER	11
22	Tool Clamped	Sourcing	Bottom	TOOL CLAMP/UNCLAMP	3
23	Tool Unclamped	Sourcing	Bottom	TOOL CLAMP/UNCLAMP	1
24	Tool Release Switch	Sourcing	Bottom	TOOL REL SW	1
25	Spindle Drive Fault	Sourcing	Bottom	INVERTER	12
26	Tool Counter *	Sinking	Bottom	CAROUSEL INPUTS	4
27	Carousel Out / TP Up	Sinking	Bottom	CAROUSEL INPUTS	2
28	Carousel In / TP Dwn	Sinking	Bottom	CAROUSEL INPUTS	1
29	Auxiliary 1	Sinking	Bottom	CAROUSEL INPUTS	3
30	Rotary Home	Sourcing	Bottom	ROTARY INPUTS	1
31	Rotary Clamped	Sourcing	Bottom	ROTARY INPUTS	3
32	Air Pressure Low	Sourcing	Bottom	AIR LOW	1
33	NOT USED	N/A	N/A	N/A	N/A
34	NOT USED	N/A	N/A	N/A	N/A
35	NOT USED	N/A	N/A	N/A	N/A
36	NOT USED	N/A	N/A	N/A	N/A
37	NOT USED	N/A	N/A	N/A	N/A
38	NOT USED	N/A	N/A	N/A	N/A
39	NOT USED	N/A	N/A	N/A	N/A
40	NOT USED	N/A	N/A	N/A	N/A
41	NOT USED	N/A	N/A	N/A	N/A
42	NOT USED	N/A	N/A	N/A	N/A
43	NOT USED	N/A	N/A	N/A	N/A
44	NOT USED	N/A	N/A	N/A	N/A
45	NOT USED	N/A	N/A	N/A	N/A
46	NOT USED	N/A	N/A	N/A	N/A
47	NOT USED	N/A	N/A	N/A	N/A
48	NOT USED	N/A	N/A	N/A	N/A
49	NOT USED	N/A	N/A	N/A	N/A
50	NOT USED	N/A	N/A	N/A	N/A
51	NOT USED	N/A	N/A	N/A	N/A
52	NOT USED	N/A	N/A	N/A	N/A
53	NOT USED	N/A	N/A	N/A	N/A
54	NOT USED	N/A	N/A	N/A	N/A
55	NOT USED	N/A	N/A	N/A	N/A
56	NOT USED	N/A	N/A	N/A	N/A
57	NOT USED	N/A	N/A	N/A	N/A
58	NOT USED	N/A	N/A	N/A	N/A
59	Range 3 / Arm Clamp	Sinking	Bottom	AUX SPINDLE INPUTS	1
60	Range 2 / Arm Stop	Sinking	Bottom	AUX SPINDLE INPUTS	2
61	Range 1 / Arm Home	Sinking	Bottom	AUX SPINDLE INPUTS	3
62	Spindle Chiller OK	Sinking	Bottom	AUX SPINDLE INPUTS	4

* Fast I/O Related

Output Map

Output Specification			Connection Location		
Number	Function	Type	Board	Connector	Pin
1	Emergency Stop	24VAC	Bottom	ESTOP	4
2	Lube Pump	220VAC	Middle	220 LUBE	1
2	Lube Pump	110VAC	Top	110 LUBE	1
3	Flood Pump	3 PHASE	Middle	FLOOD MOTOR	1,2,3
4	Mist Solenoid	110VAC	Top	MISTER	1
5	Carousel Direction	Internal	Middle	N/A	N/A
6	Carousel Out Solenoid	110VAC	Top	CAR OUT	1
7	Tool Clamp Solenoid	110VAC	Top	TOOL CLAMP	1
8	Air Blow Through	110VAC	Top	AIR BLOW	1
9	Carousel In Solenoid	110VAC	Top	CAR IN	1
10	Orient	Relay Contact	Bottom	INVERTER	3
11	Spindle Chiller	Configurable	Middle	AUX 2 / CHILLER	1,2
12	Spindle Cooling Fan	220VAC	Middle	SPIN FAN	1
13	Spindle Direction	Relay Contact	Bottom	INVERTER	5,6,7
14	Spindle Enable	Relay Contact	Bottom	INVERTER	4
15	Inverter Reset	Relay Contact	Bottom	INVERTER	1
16	Error Check	Internal	N/A	N/A	N/A
17	Spin. Speed Bit 0	Internal	N/A	N/A	N/A
18	Spin. Speed Bit 1	Internal	N/A	N/A	N/A
19	Spin. Speed Bit 2	Internal	N/A	N/A	N/A
20	Spin. Speed Bit 3	Internal	N/A	N/A	N/A
21	Spin. Speed Bit 4	Internal	N/A	N/A	N/A
22	Spin. Speed Bit 5	Internal	N/A	N/A	N/A
23	Spin. Speed Bit 6	Internal	N/A	N/A	N/A
24	Spin. Speed Bit 7	Internal	N/A	N/A	N/A
25	Spin. Speed Bit 8	Internal	N/A	N/A	N/A
26	Spin. Speed Bit 9	Internal	N/A	N/A	N/A
27	Spin. Speed Bit 10	Internal	N/A	N/A	N/A
28	Spin. Speed Bit 11	Internal	N/A	N/A	N/A
29	Gear Change	Relay Contact	Middle	AUX1 / RANGE	1,2
30	Rotary Clamp Solenoid	110VAC	Top	ROT CLAMP	1
31	Carousel Enable *	3 PHASE	Middle	CAROUSEL MOTOR	1,2,3
32	Red Light	110VAC	Top	RED LIGHT	1
33	Green Light	110VAC	Top	GRN LIGHT	1
34	Yellow Light	110VAC	Top	YEL LIGHT	1
35	Worklight	110VAC	Top	WORKLIGHT	1
36	Auxiliary 1	Relay Contact	Middle	AUX 1 / RANGE	3,4
37	Auxiliary 2	Configurable	Middle	AUX 2 / CHILLER	3.4
38	Auxiliary 3	Open Collector	Bottom	AUX DRIVERS 1	10
39	Auxiliary 4	Open Collector	Bottom	AUX DRIVERS 1	9
40	Fast I/O Enable *	N/A	N/A	N/A	N/A
41	Auxiliary 5^	Open Collector	Bottom	AUX DRIVERS 1	7
42	Auxiliary 6^	Open Collector	Bottom	AUX DRIVERS 1	6
43	Auxiliary 7^	Open Collector	Bottom	AUX DRIVERS 1	5
44	Auxiliary 8^	Open Collector	Bottom	AUX DRIVERS 1	4
45	Auxiliary 9^	Open Collector	Bottom	AUX DRIVERS 1	3
46	Auxiliary 10^	Open Collector	Bottom	AUX DRIVERS 2	10
47	Auxiliary 11^	Open Collector	Bottom	AUX DRIVERS 2	9
48	Auxiliary 12^	Open Collector	Bottom	AUX DRIVERS 2	8
49	NOT USED	N/A	N/A	N/A	N/A
50	NOT USED	N/A	N/A	N/A	N/A
51	NOT USED	N/A	N/A	N/A	N/A
52	NOT USED	N/A	N/A	N/A	N/A
53	NOT USED	N/A	N/A	N/A	N/A
54	NOT USED	N/A	N/A	N/A	N/A
55	NOT USED	N/A	N/A	N/A	N/A
56	NOT USED	N/A	N/A	N/A	N/A
57	NOT USED	N/A	N/A	N/A	N/A
58	NOT USED	N/A	N/A	N/A	N/A
59	Auxiliary 13	Open Collector	Bottom	AUX DRIVERS 2	6
60	Auxiliary 14	Open Collector	Bottom	AUX DRIVERS 2	5
61	Auxiliary 15	Open Collector	Bottom	AUX DRIVERS 2	4
62	Auxiliary 16	Open Collector	Bottom	AUX DRIVERS 2	3

Fast I/O Related

^ Often used to store the tool number

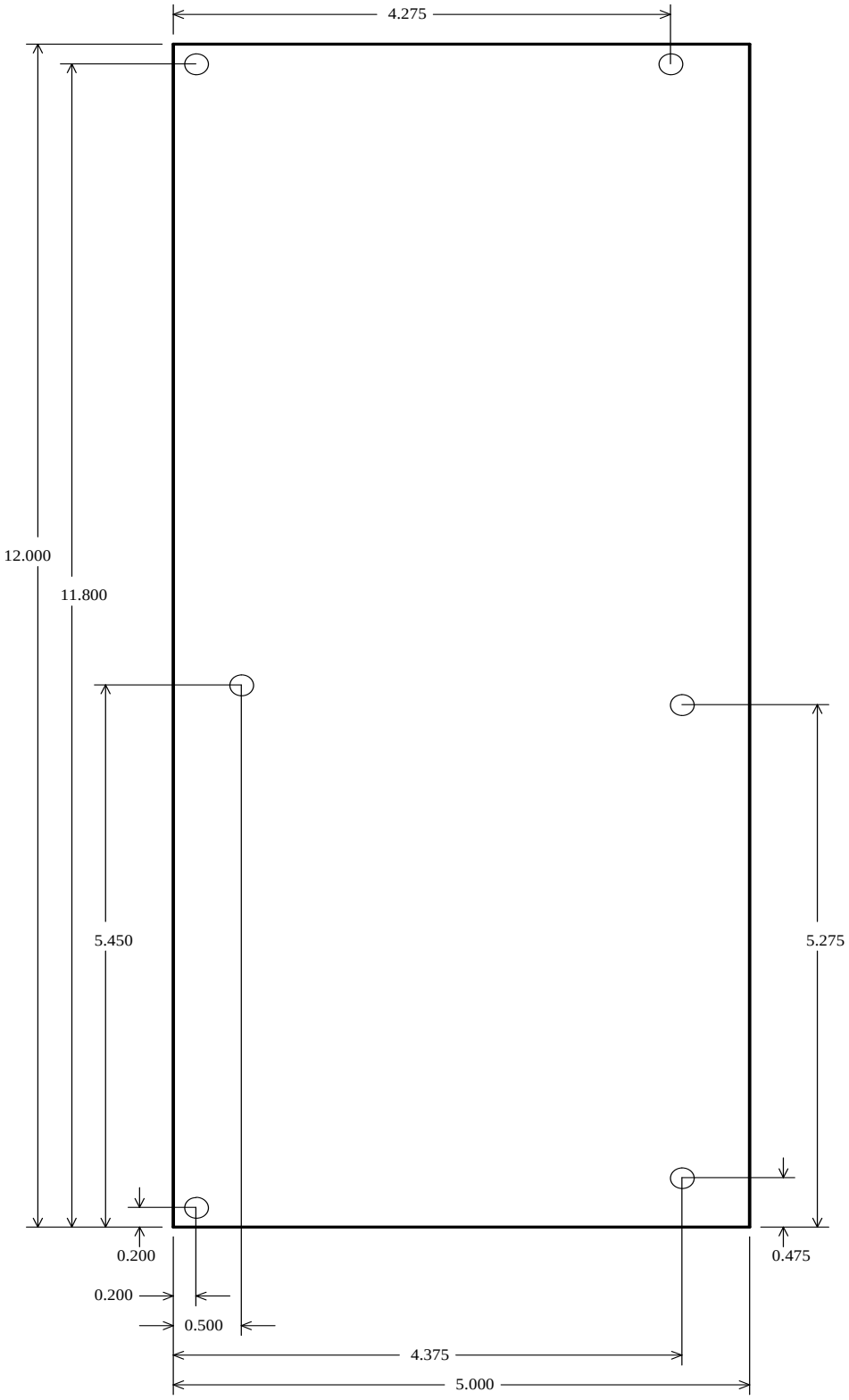
RTK3 Specifications

Characteristic	Min.	Typ.	Max.	Unit
5 Volt Supply Current (Ext. Use)	0	0	0.3	A
24 Volt Supply Current (Ext. Use)	0.0	0.1	0.4	A
Input Pullup Voltage	23	24	25	V
Power Relay Output Current	0.01	-	10	A @ 250 Vac
Power Relay Output Current	0.01	-	5	A @ 30 Vdc
Power Relay Output Current	10	-	400	mA @ 100 Vdc
Signal Relay Output Current	0.001	-	0.5	A @ 125 Vac
Signal Relay Output Current	0.001	-	1	A @ 24 Vdc
Open Collector Output Current	-	40	500	mA
Open Collector Output Voltage	-	5	-	V
3 Phase Output Load	-	-	0.5	HP @ 250 Vac
110 VAC Input Current	-	9	25	A
3 Phase Input Current	-	-	8	A / Phase
Input Operating current	9	11	15	mA
Analog Output Resolution	-	12	-	bits
Analog Output Voltage	0	-	10	V
Analog Output Current	0	1	20	mA
Size: 12*5*4 (W*D*H)				inch

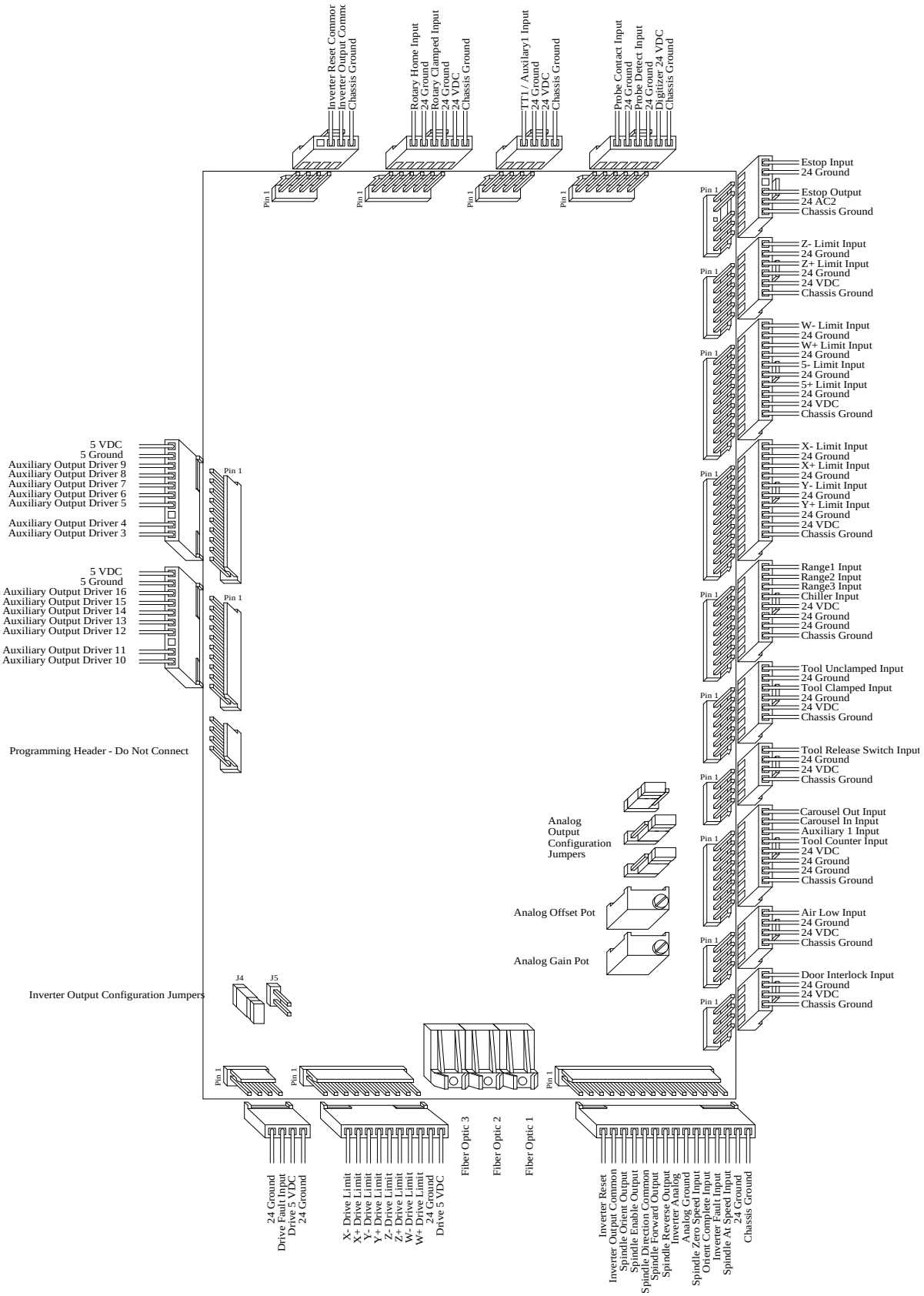
RTK3 Troubleshooting

Symptom	Possible Cause	Corrective Action
All status LEDs out	110 Vac not connected	connect 110 Vac supply to H7 on top board, check voltage between 110 ACN and 110 ACH terminals
	F1 Fuse on logic board blown	Return for repair
PLC OK LED out	Motion control card not initialized	Start CNC10 software
	Incorrect PIC on motion control card	Install IO2PIC
	Fibers faulty or incorrectly connected	Check connections, inpect or swap fibers
+24 LED out	Fuse blown or poor connection	Check +24 Vdc fuse and fuseclips on bottom board
No digitizer +24 Vdc	If +24 Vdc LED is lit, digitizer fuse blown or poor connection	Check digitizer fuse and fuseclips on bottom board
No analog output or non-linear output	Incorrect Parameter 31 setting	Set P31 to -1
3PDT Relays and Estop contactor not working	24 Vac not connected	plug 24 Vac cable into 24 Vac input on middle board, check connections and voltage
3 phase fuses blowing	Incorrect PLC program for certain 3PDT relays	Check for the most current PLC program
AUX2 or CHILLER outputs not working	Jumpers J1-J4 on Output (middle) board not set correctly	Refer to page 6 in the manual

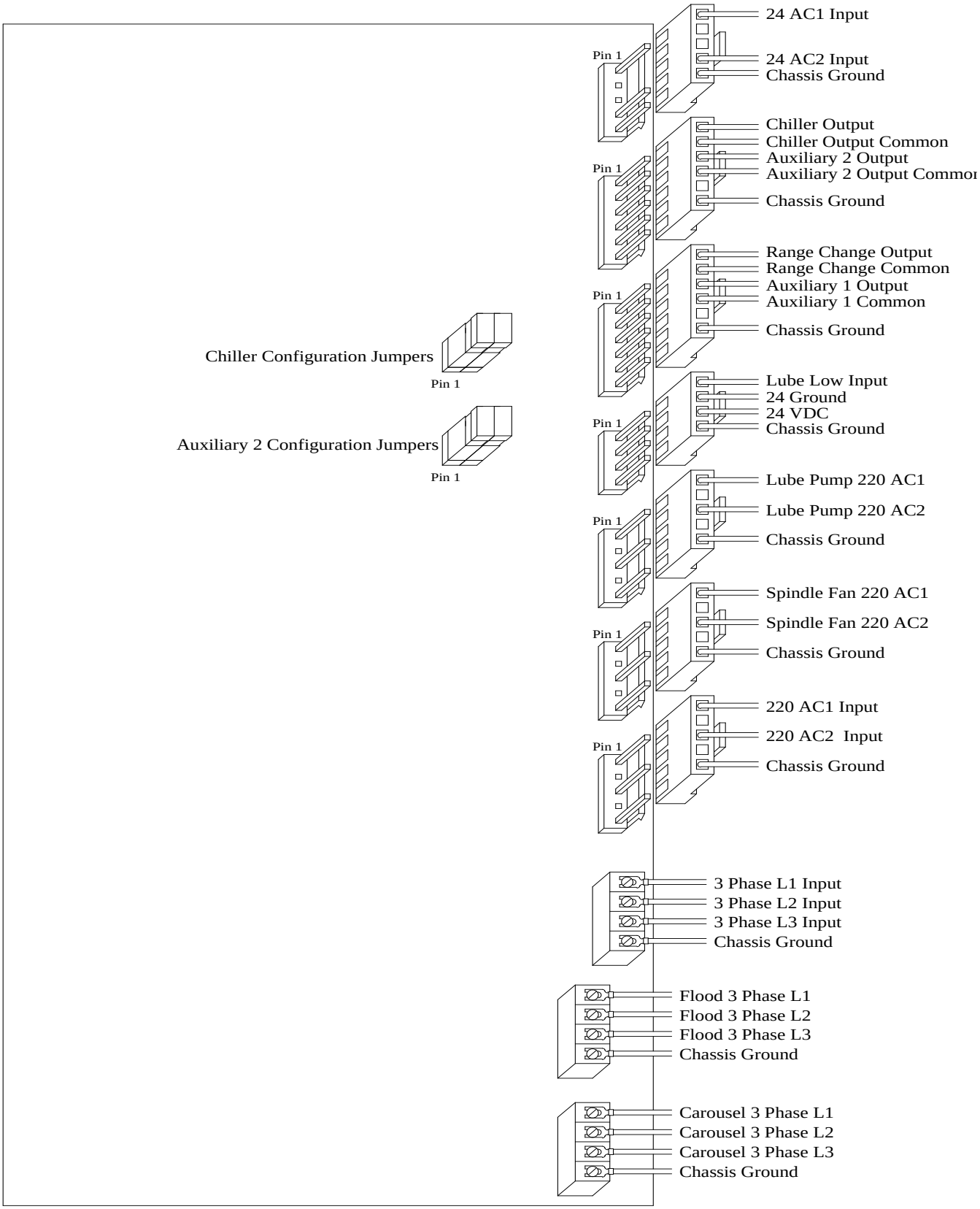
RTK3 Mounting Dimensions



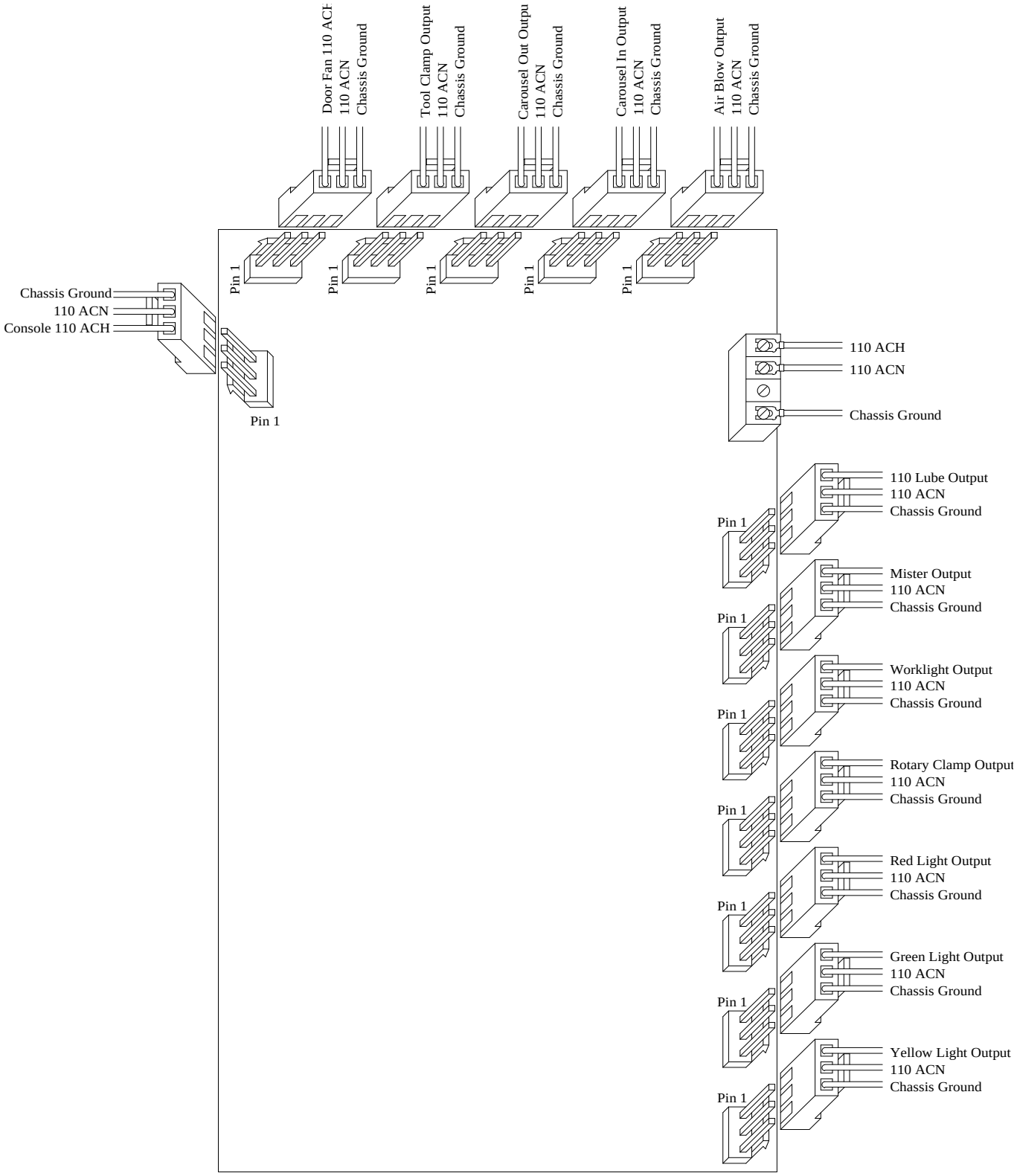
RTK3L (Bottom) Board Connections



RTK3O (Middle) Board Connections

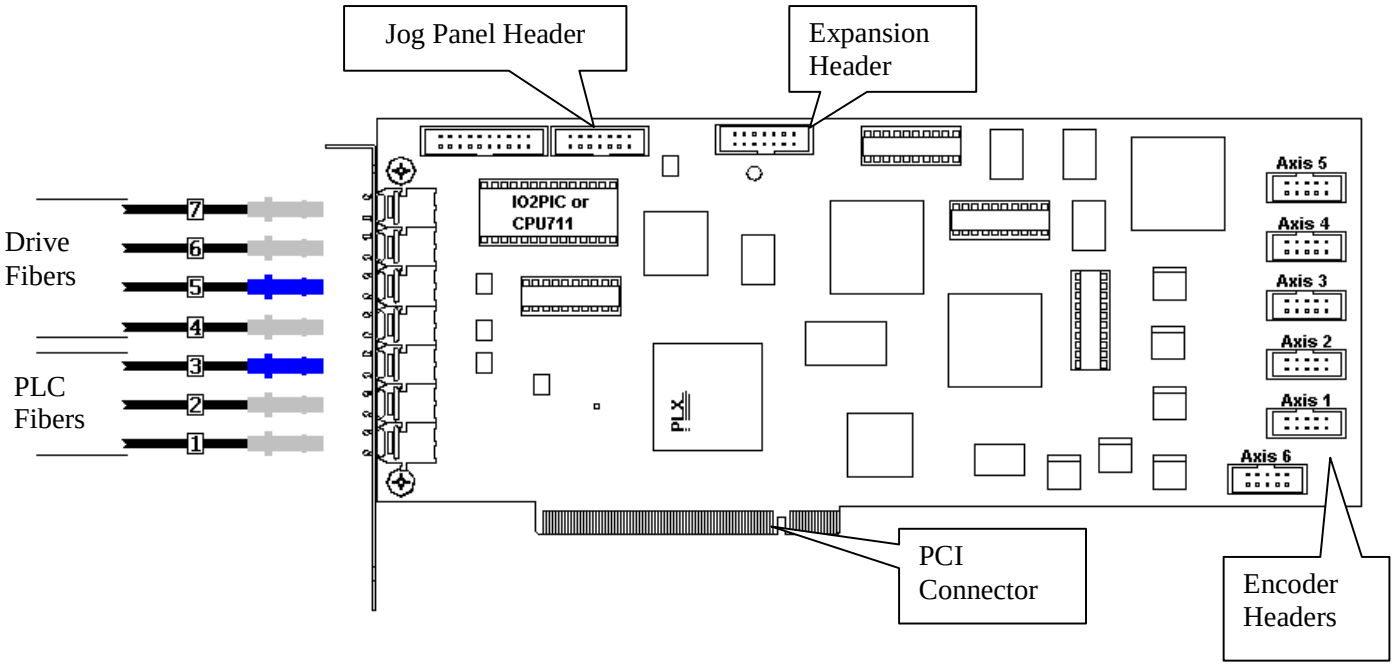

24 AC1 Input
24 AC2 Input
Chassis Ground
Chiller Output
Chiller Output Common
Auxiliary 2 Output
Auxiliary 2 Output Commor
Chassis Ground
Range Change Output
Range Change Common
Auxiliary 1 Output
Auxiliary 1 Common
Chassis Ground
Lube Low Input
24 Ground
24 VDC
Chassis Ground
Lube Pump 220 AC1
Lube Pump 220 AC2
Chassis Ground
Spindle Fan 220 AC1
Spindle Fan 220 AC2
Chassis Ground
220 AC1 Input
220 AC2 Input
Chassis Ground
Pin 1
Chiller Configuration Jumpers
Pin 1
Auxiliary 2 Configuration Jumpers
Pin 1
3 Phase L1 Input
3 Phase L2 Input
3 Phase L3 Input
Chassis Ground
Flood 3 Phase L1
Flood 3 Phase L2
Flood 3 Phase L3
Chassis Ground
Carousel 3 Phase L1
Carousel 3 Phase L2
Carousel 3 Phase L3
Chassis Ground

RTK3T (Top) Board Connections



Motion Control Card

CPU10



The CPU10 is designed to replace both CPU7 and CPU9 cards with a compatible PCI slot version as a result of elimination of ISA slots from PC motherboards.

1. PCI bus Card will work in PCI slot without manually setting address of card. Cards will work with other PCI cards in use without conflict.

2. Inputs: inputs will include Jog panel, expansion header, PLC header, six encoder inputs. All six encoder inputs will have polarity check.

3. Fiber optic communication: All fiber optic components will be available for both the SD drive system (4 fiber) and the brushed system (2 fiber). Software will make the adjustment for the system. PLC fiber optic communications remain the same as CPU7, CPU9.

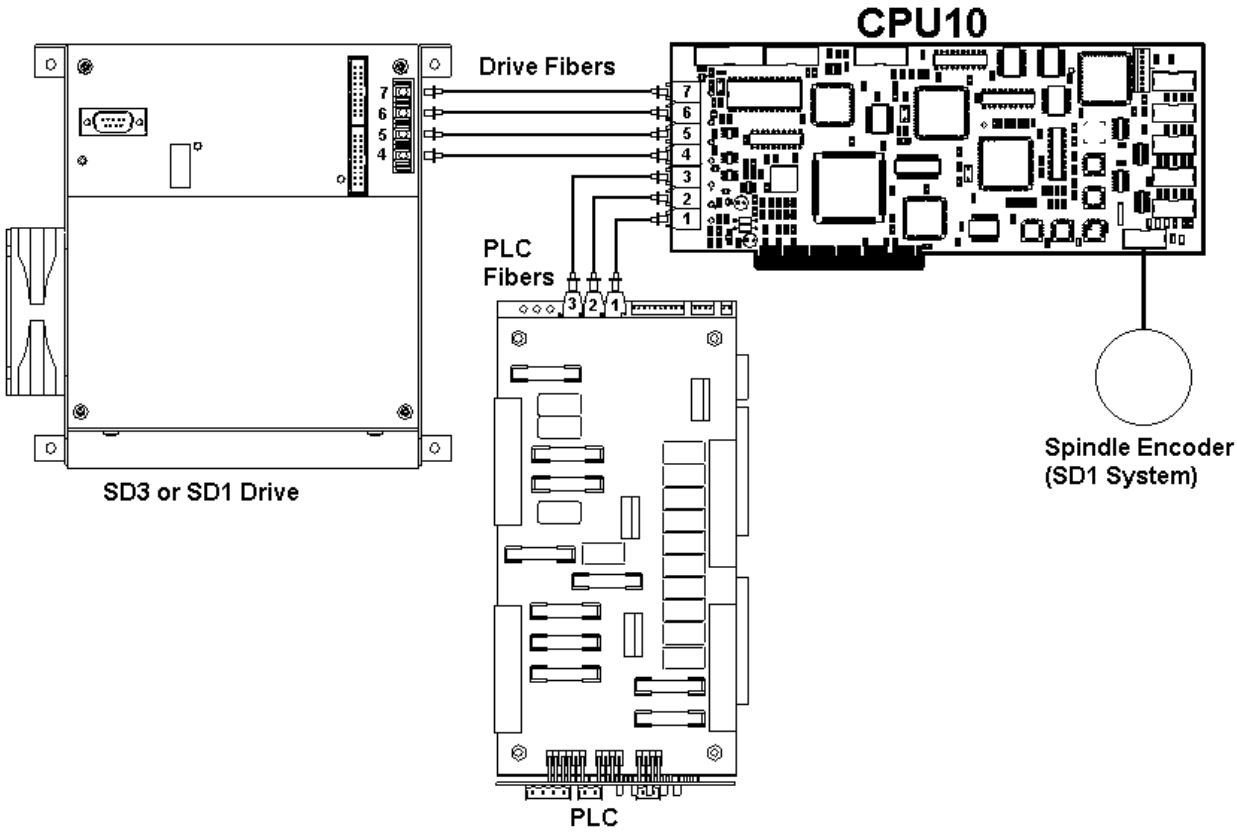
4. Card will fit in standard PC case and in the Centroid Console updated for the card.

5. CPU10 will accept spindle input for multiple SD1 drive system (3 KW and 4KW motors). SD3 systems must use the spindle encoder input on the drive.

6. The type of PLC being used will determine whether the CPU10 will have an IO2PIC or CPU711 PIC on board. DC3IO, PLCIO2 and RTK3 PLCs use the IO2PIC. Servo3IO,RTK2 and PLC 15/15s use the CPU711 PIC.

Applications:

SD drive system (Brushless or Brushed motors)

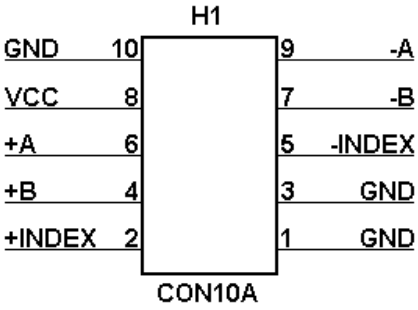


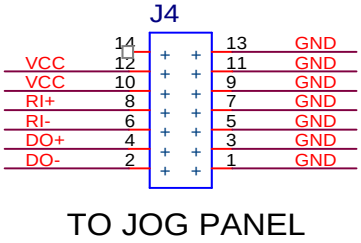
SD drive system (Brushless or Brushed motors)

The DC3IO system leaves 2 fiber transmitters open and uses the encoder inputs for all motors. This can also be used for the Servo1 and Quaddrv and with any Centroid PLC.

The CPU10 card performs at the same speeds and functionality as the CPU7 and CPU9, block through put and system response time are not affected. The CPU10 **requires** the Linux or Windows operating system and different mounting hardware in the Centroid Console.

Encoder Connections:





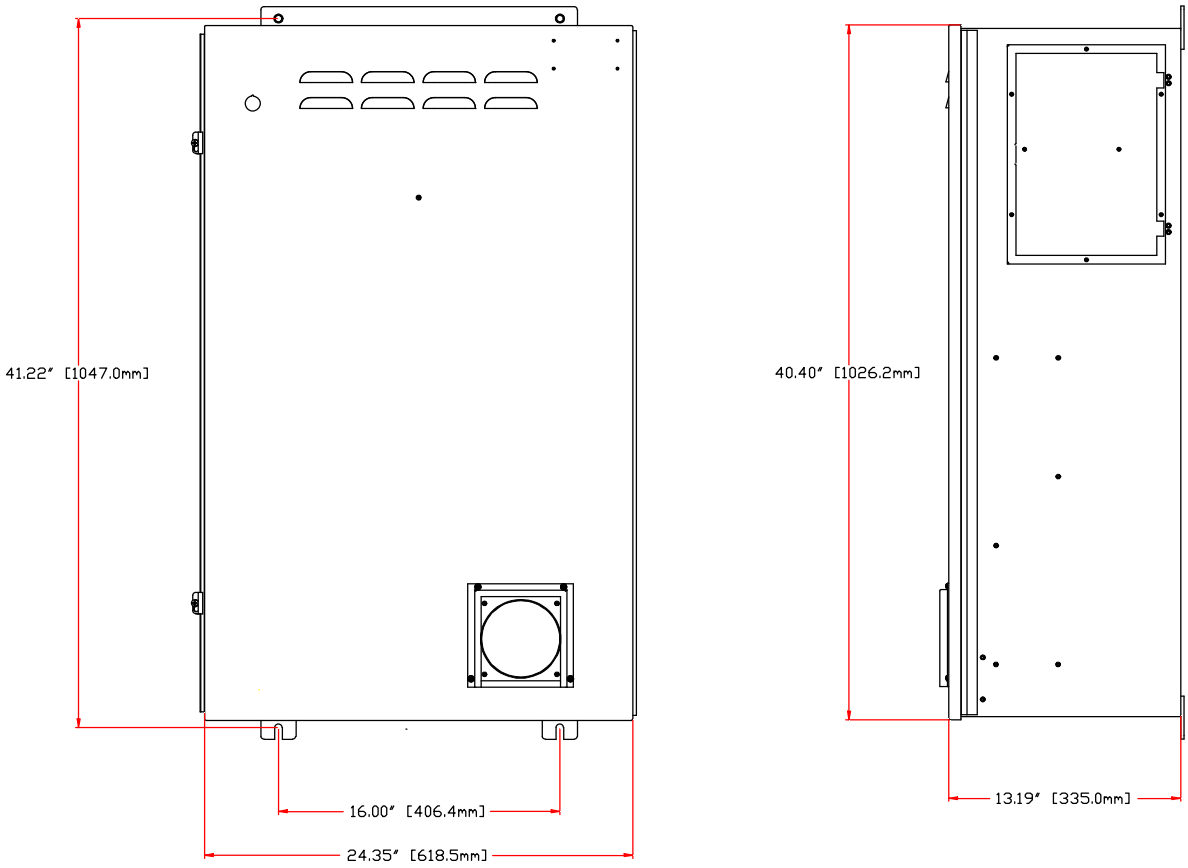
Jog Panel Connections:

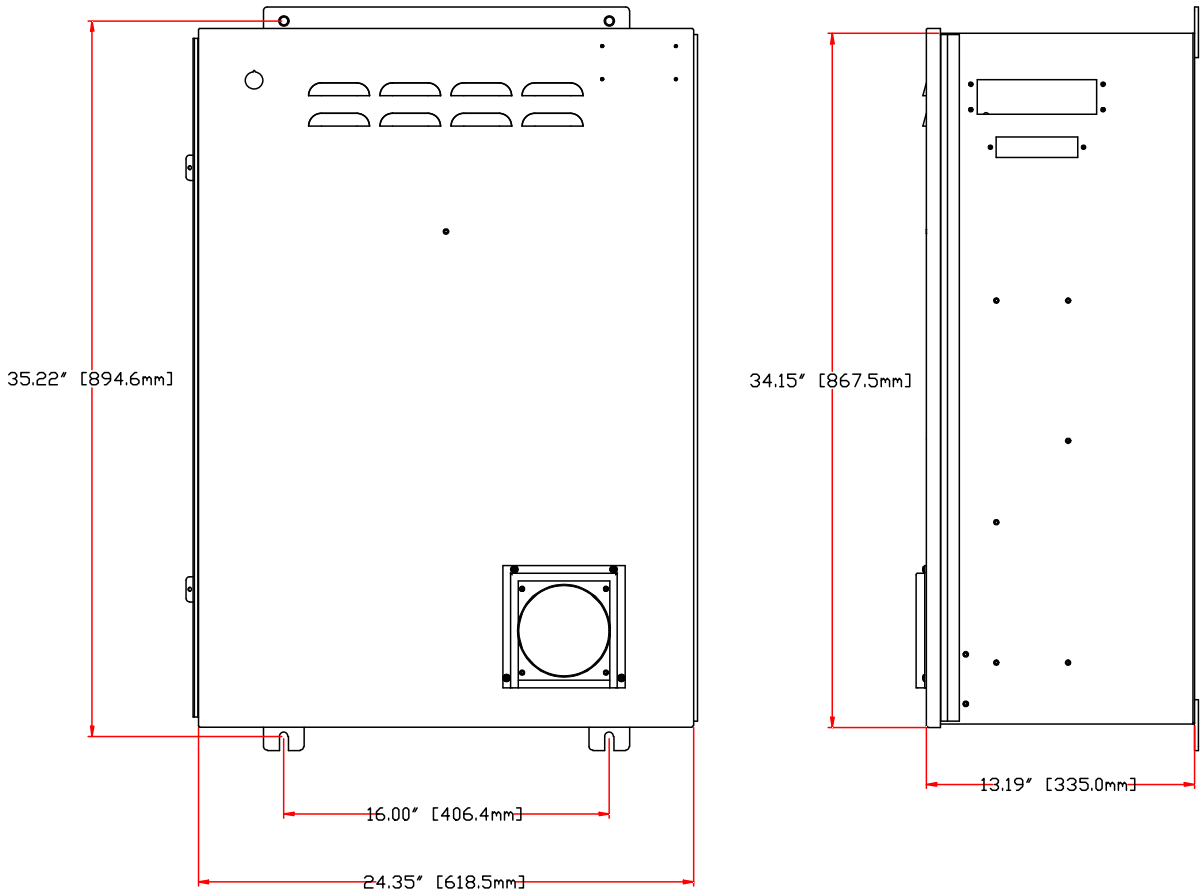
Complete Kit Hardware Installation

Mounting the Magnetics cabinet

The magnetics cabinet should be mounted on the back or the right-hand side of the mill. If the machine will not allow the cabinet to be directly mounted, it must be mounted to a stationary object such as a wall very near the machine. If it is to be mounted on the right-hand side of the mill, the cabinet must be far enough to the rear that the Way covers, the Table, the Head (on a bed mill), Splash guards and any cables or hoses will not hit the cabinet or interfere with the ability to open the cabinet door. Also, in many cases the top of the cabinet will require spacers to level the cabinet due to the shape of most machines.

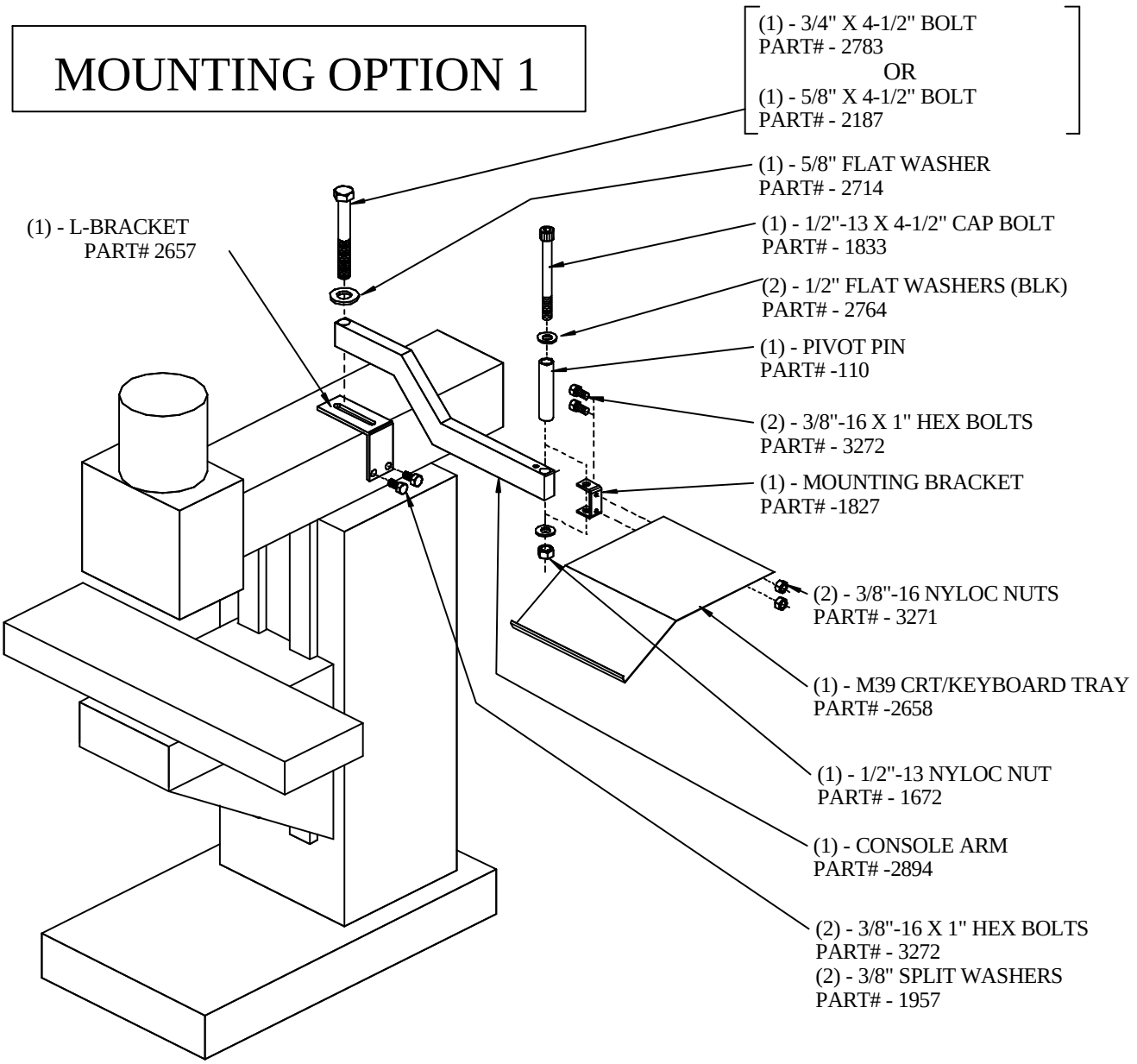
The magnetics cabinet should be mounted high enough to allow room for the cabling coming out of the bottom of the cabinet. If a Rotary table is going to be added then room must be left for connection of the rotary table connector. For most mills mounting brackets will have to be made. Aluminum bar stock ¼" thick, 1" wide and 18" to 24" long will normally work quite well. Dimensional drawings of the two cabinet sizes are shown below.



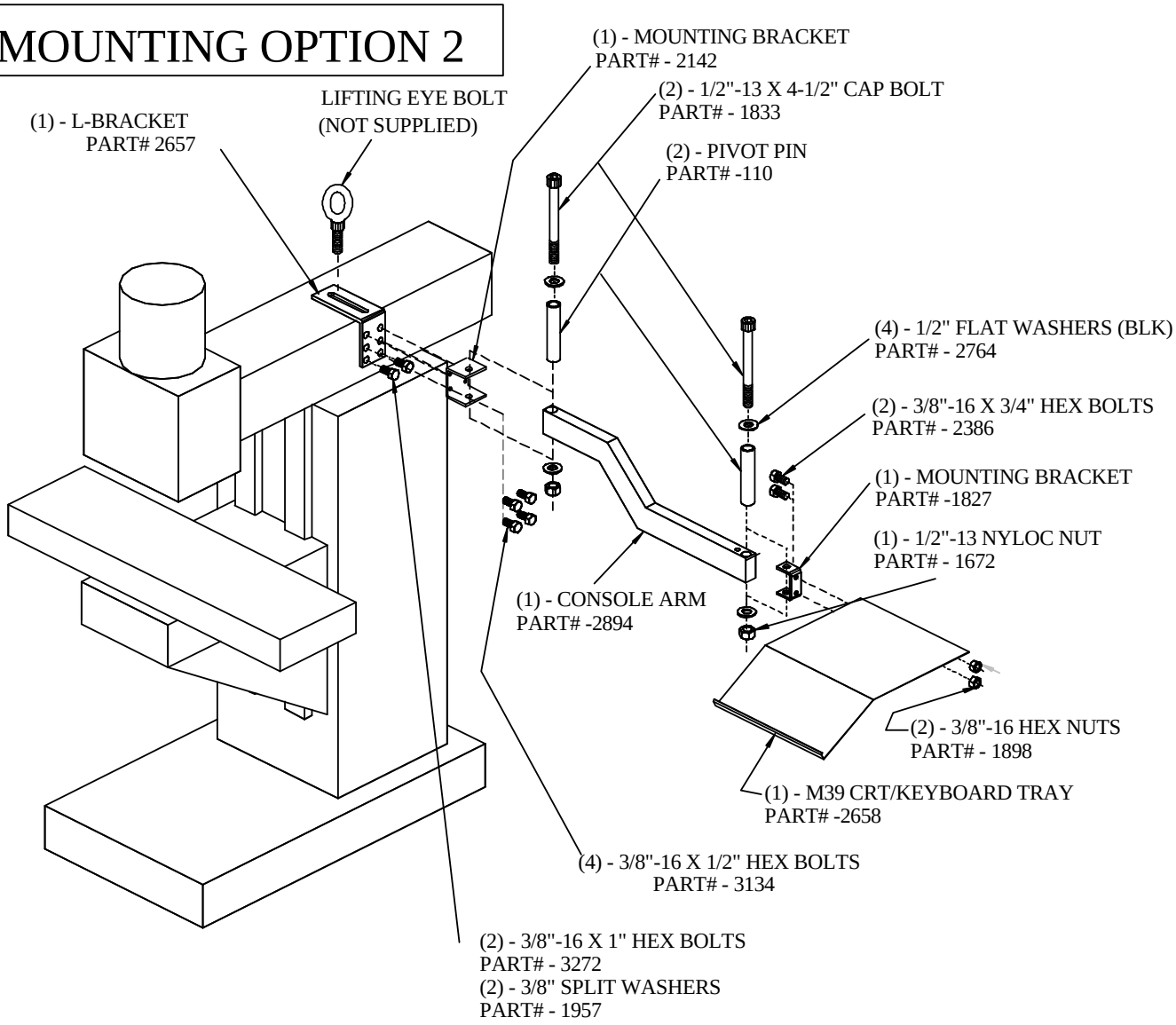

35.22" [894.6mm]
16.00" [406.4mm]
24.35" [618.5mm]
34.15" [867.5mm]
13.19" [335.0mm]

Mounting the console

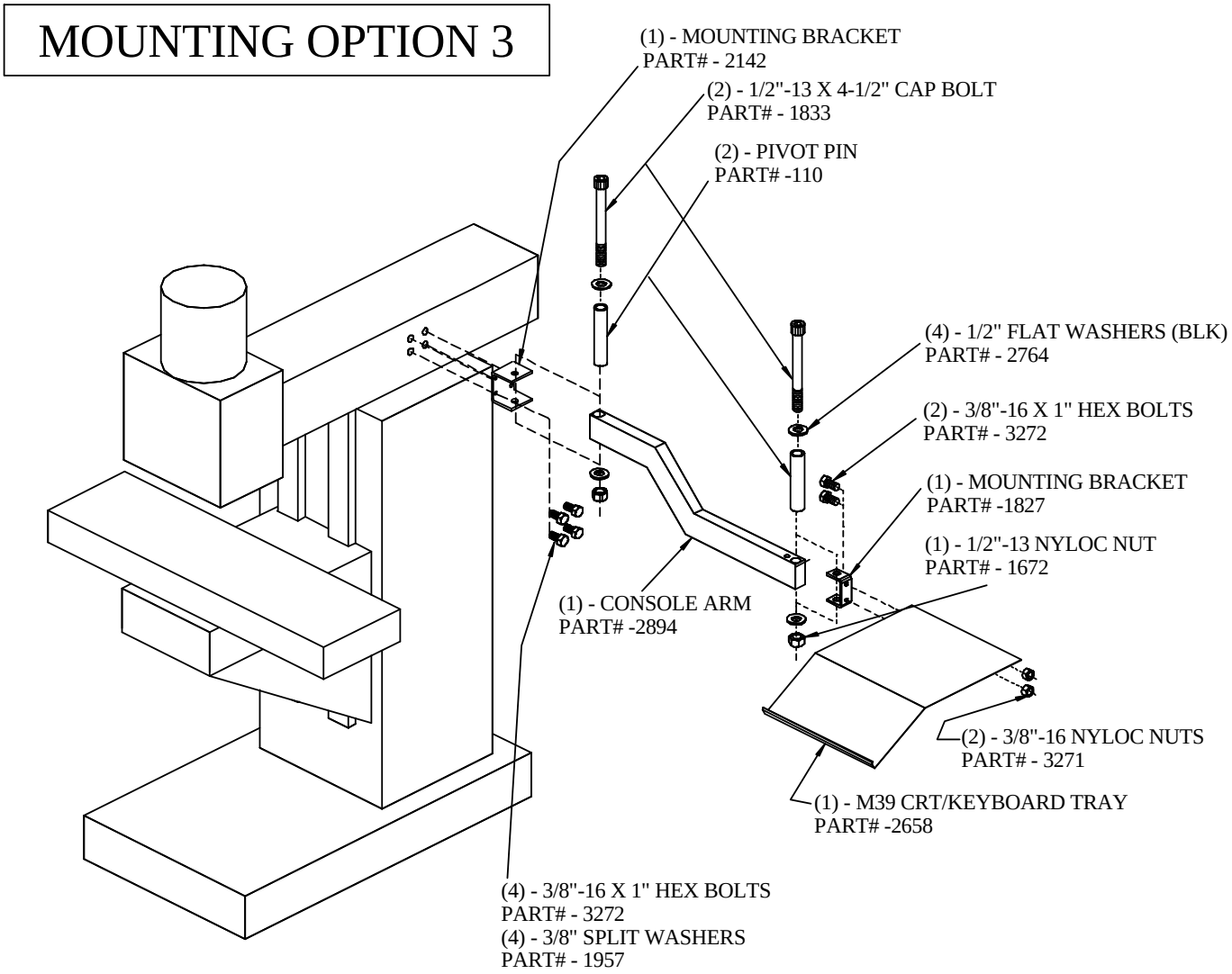
There are currently three different mounting arm options available. Each is shown graphically below. For additional information, reference Tech Bulletin TB132.



MOUNTING OPTION 2



MOUNTING OPTION 3



Console Installation

To ensure the console is installed in such a way as to make it comfortable and functional for the user, the following guidelines should be used.

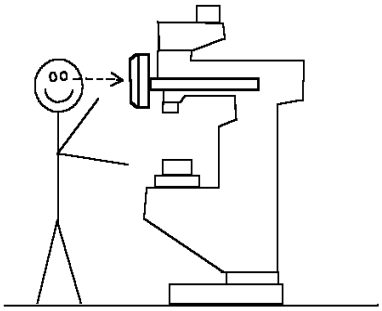
The correct height of the console is determined by putting the middle of the console at eye level. The average height user should be able to stand at the console and look directly into the middle of the console without having to bend their neck. It is important to take into account the shipping pallet if present. To adjust for the pallet, stand on an equal height pallet when checking the console height. For a knee mill, the bottom of the console should be no lower than the bottom of the spindle when it is fully retracted and a tool is not present.

The arm must always put the console at a comfortable height and within easy reach. The arm should not place the console too far back or too far forward. An easy test to determine that the arm length is correct is whether or not the operator has to lean over the table to operate the console. The operator should not have to do this. If a table guard is to be mounted on the table later, this should be taken into consideration when testing the arm length.

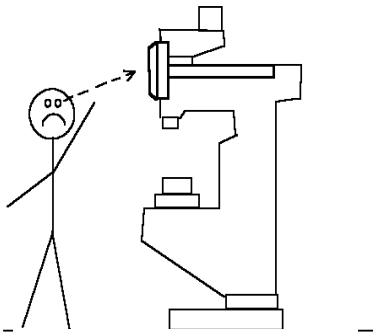
The arm should not place the console too far in front of the machine. The arm length of the console should be such that the operator can stand close to the machine without hitting the table and not have to lean back or step away from the machine to operate the control. If the operator needs to lean back or step away from the machine, the arm is too long.

Also, use a stop bolt to prevent the arm from swinging into the head or shop light. Mount the shop light on the left if an automatic tool changer (ATC) is not present.

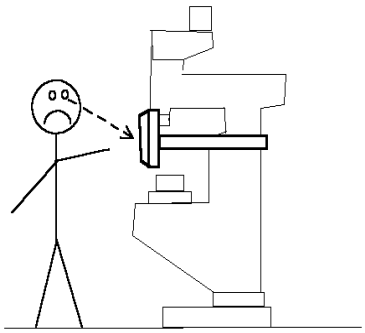
The following pictures illustrate the guidelines given previously.



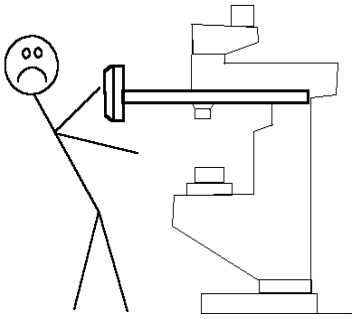
Eye Level: Not Too High, Not Too Low



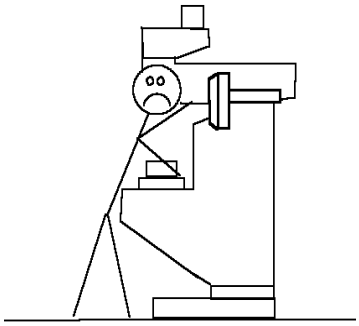
Control Too High



Control Too Low



Control Arm Too Long

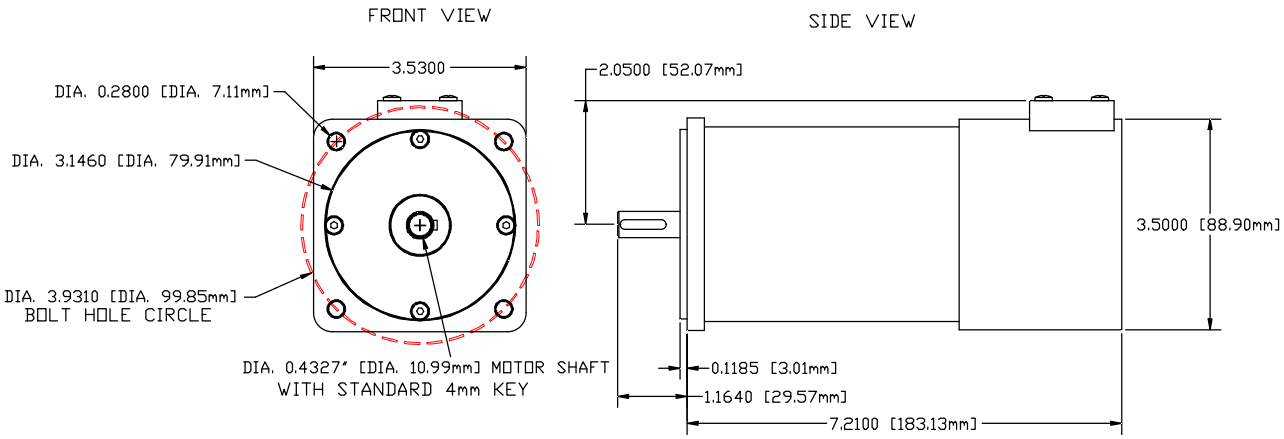


Control Arm Too Short

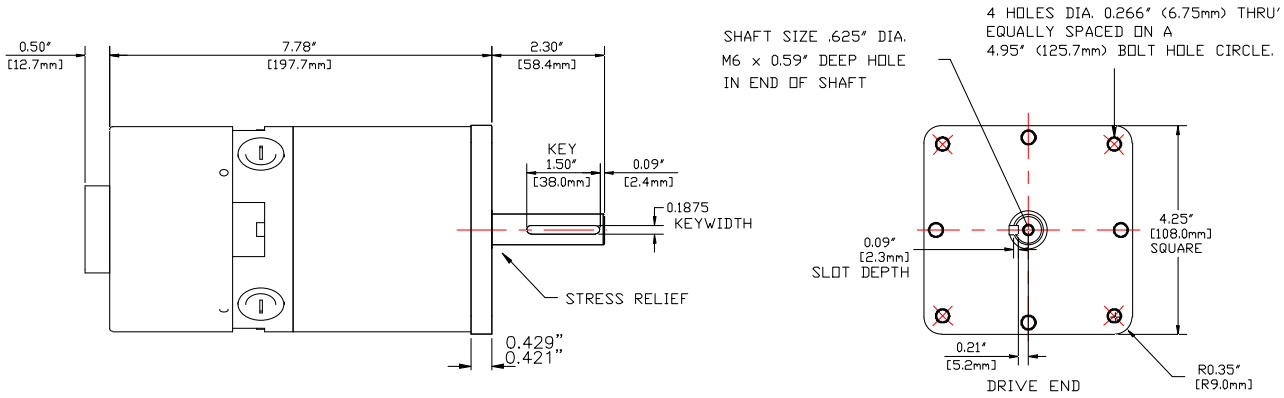
Servo Motors Installation

Motor Space Requirements and Mounting Bolt Hole Spacing:

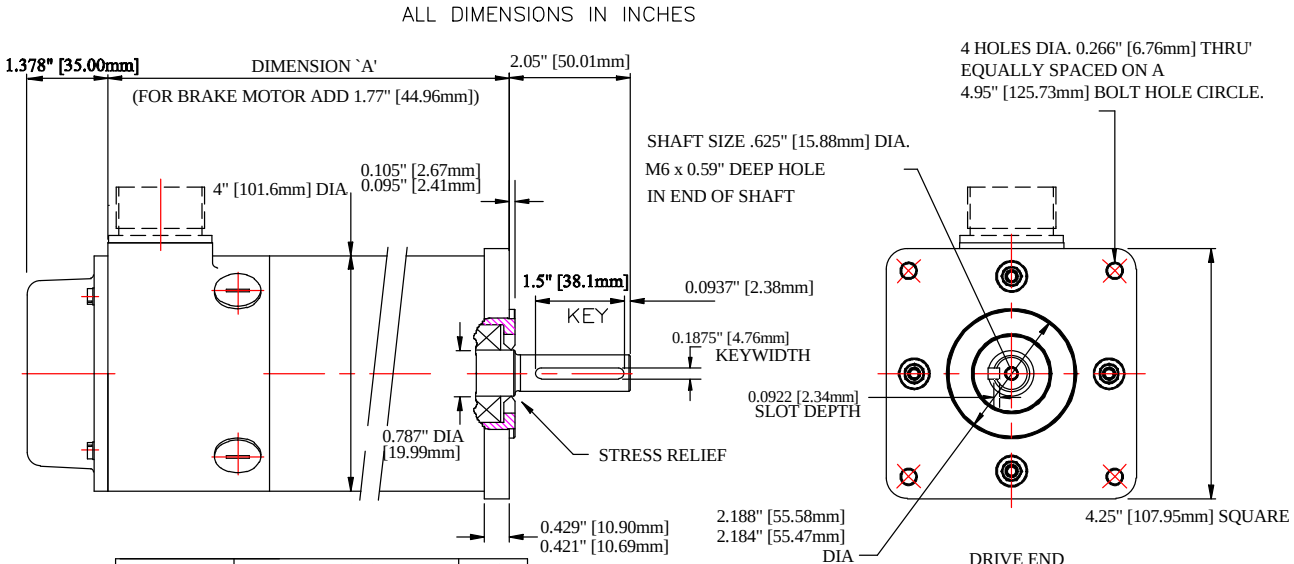
Redcom 17 in-lb DC Servo Motor



Redcom 29 in-lb DC Servo Motor

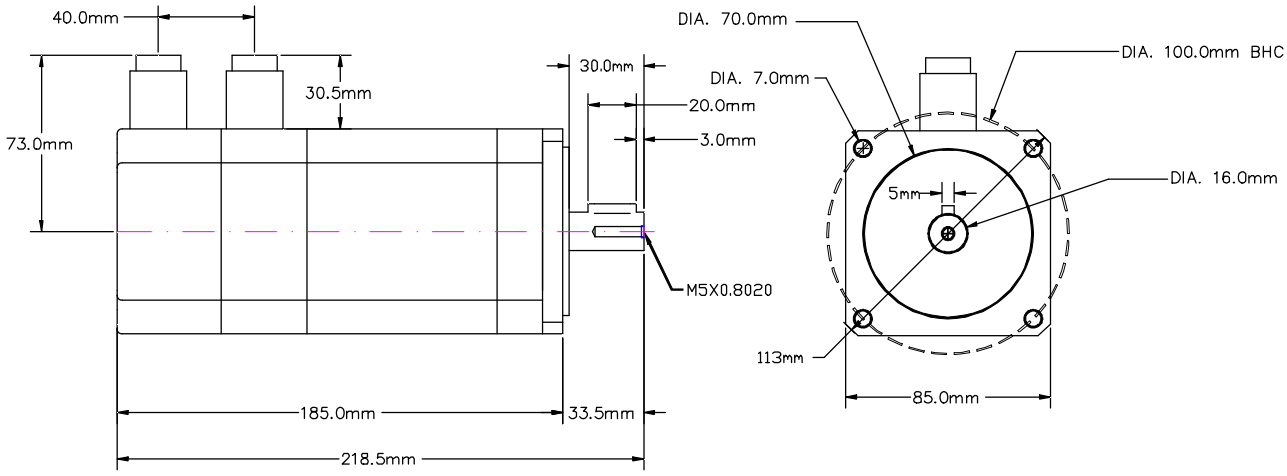


SEM 29 in-lb / 40 in-lb DC Servo Motors

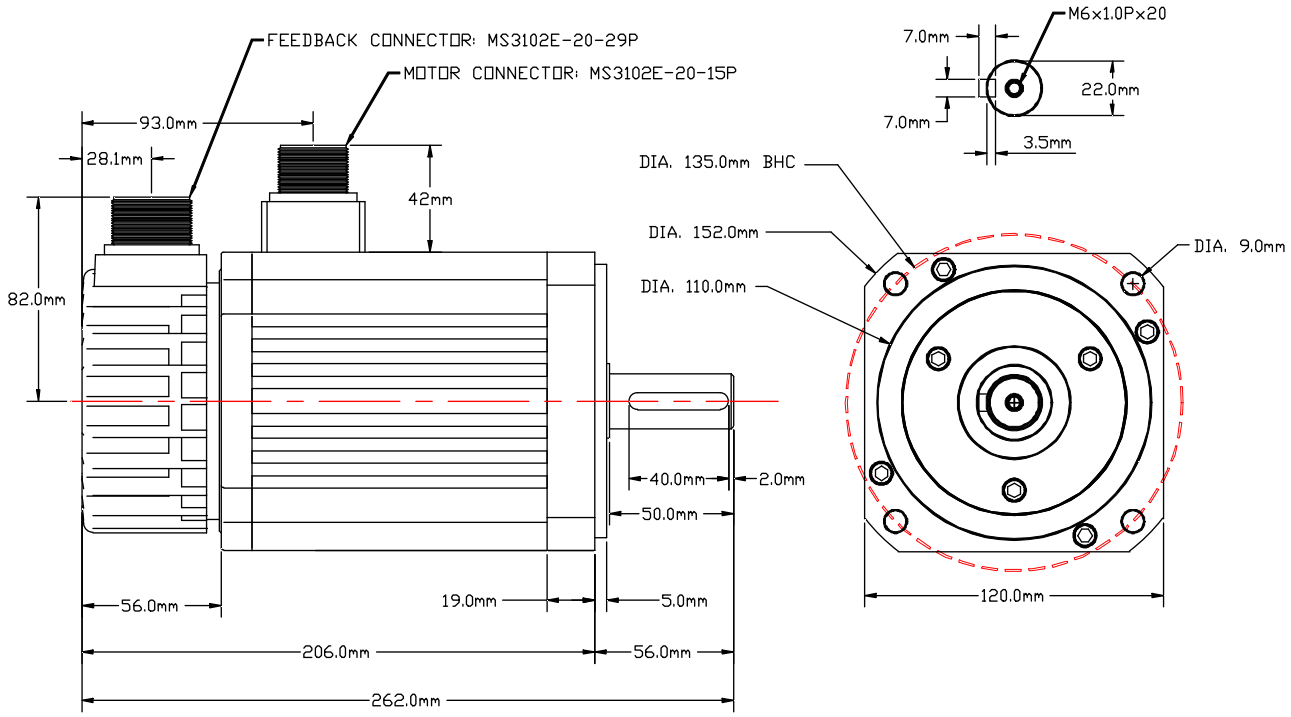


FRAME	DIM 'A'	in/lb
MTS30M4	8.36" [212.34mm]	29
MTS30U4	10.12" [257.05mm]	40

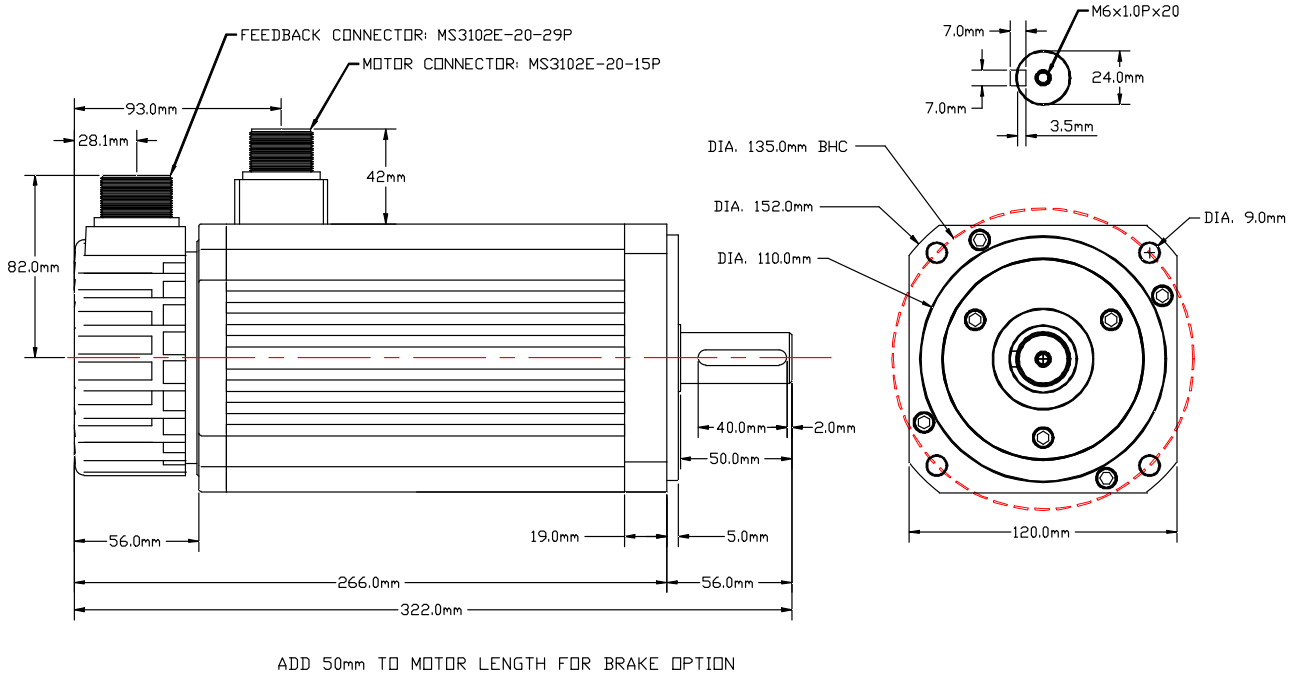
LeeDan 0.75kW AC Servo Motor



LeeDan 1kW AC Servo Motor



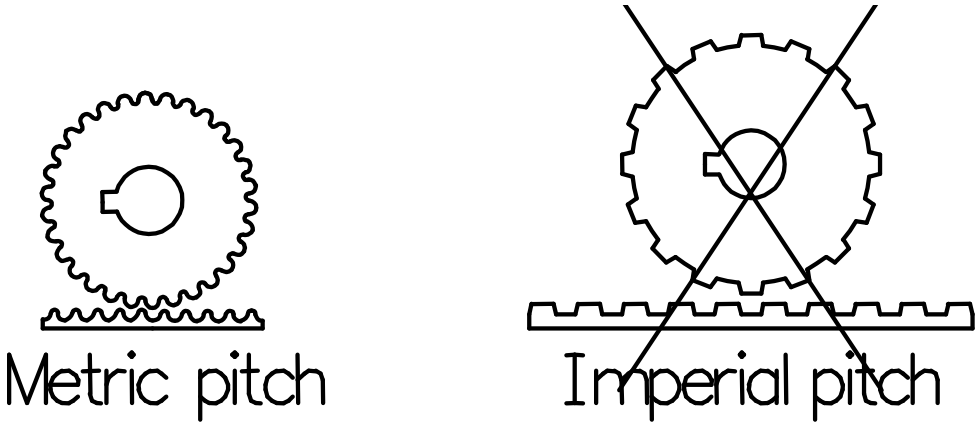
LeeDan 2.2kW AC Servo Motor



Pulley and Belt Selection

A large number of machine tools employ timing belts and sprockets to transfer motion of the servo motor shaft to the ball screw. This allows the machine tool builder to easily change the overall turns ratio of the axis drive system, and allows the servo motor to be placed in a space efficient position.

Although belts and sprockets are available in both Imperial and Metric pitch, for best overall performance and wear resistance, use 5mm pitch belts and sprockets on your machine tool. Use 8mm pitch on a very large machine tool. Two sources for these belts and sprockets are: Gates Rubber Company HTD PowerGrip system (distributed by bearing houses world-wide), H. Neuman & Company, Schiller Park, Il.. Voice #847-671-5885, Fax #671-3603. For additional information on pitch styles, reference Tech Bulletin TB003.



Pulley Installation

WARNING: Do not pound pulley onto shaft or use excessive force during installation.

Installation Procedure

1. Polish the motor shaft with a fine abrasive (Scotch Bright Pad) to remove all burs.
2. Remove any dirt from the motor keyway.
3. Remove any burs from the keyways using a flat file (see Fig. 1a, 1b).

Note: One or two passes with the file should work.

4. Remove any burs on the key with a flat file as well (see Fig. 2).
5. Test fit the key in the motor keyway.

Note: Key should fit snugly into keyway (no play).

6. Ream pulley collar to remove burs.
7. Clean motor shaft, pulley, and key with a solvent (carburetor cleaner).
8. Without the key in the keyway, slide the pulley onto the shaft.

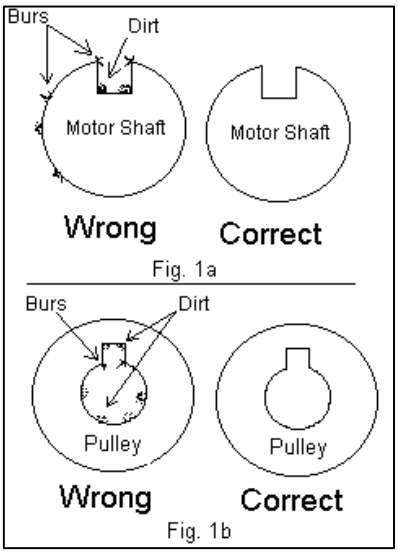
Note 1: Pulley should slide on easily by applying only hand pressure.

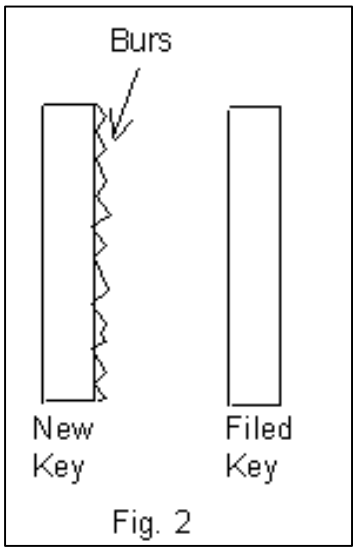
Note 2: Do not use any lubricant on shaft.

9. Repeat steps 1-8 if pulley does not fit.
10. Remove the pulley from the shaft and insert the key into the keyway.
11. Gently slide the pulley onto the motor shaft.

Note: The pulley should fit snugly on the shaft without any play.

12. Align the pulley to the proper position and tighten the pulley set screws.

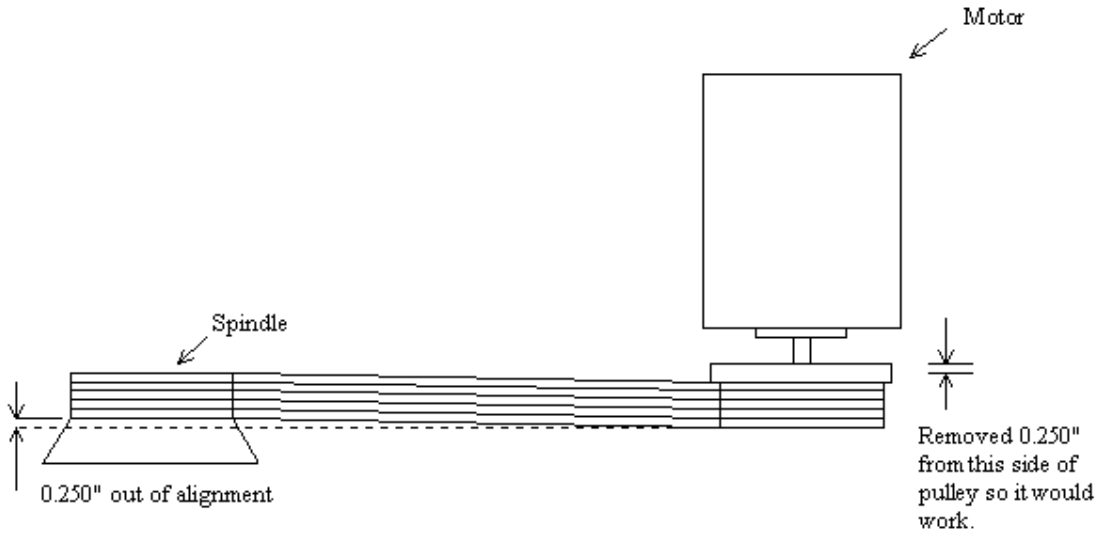




Pulley Alignment

If the pulleys and belts are not properly aligned, damage can occur to the machine. See the following example of a spindle motor and pulley setup. Although a spindle is used as an example, the same problem can occur with servo / ball screw combinations as well.

This diagram shows the motor being mounted 0.250" lower than the spindle pulley. This will cause excessive noise and create heat build up on the spindle pulley and belt. The belt would surely become frayed and in time would eventually be destroyed.



To fix this problem, you must bring the motor pulley into alignment. One recommendation is removing the pulley from the motor and machining 0.250" from the backside of the pulley. You will also have to make a new retainer to hold the pulley on.

Pulleys on the motor and spindle that are of a poor quality cast and have numerous voids will also decrease belt life. Be sure the belt is the correct multi-groove type.

Direct Drive Transmission Coupling

If using a direct drive system, it is very important to make sure the coupler is properly aligned. A misaligned coupler can come loose, causing a very noisy racket, which sounds like a machine gun firing, giving the false impression of a bad motor.

Realigning the Noisy Coupler

Remove the old coupler from the machine, remove the screws on the coupler, place red loctite on the four screws, and loosely reconnect the couplers. Slide a 5/8" ground rod through both sections, thereby re-aligning the couplers. Tighten up the screws, and remove the 5/8" ground rod. Then place the coupler back on the machine.

Replacement Couplers

Should your coupler need replacing; KTR and Zero-Max can supply you with excellent couplers.
KTR Rotex GS model, coupler size 28/38. It is silver in appearance and uses a red poly coupling. KTR's phone # is 219-872-9100 and their Email: KTRcorp@ktrcorp.com (www.ktrcorp.com)

Zero-Max,Inc. Servo Class # SC 050. Their phone # 1-800-533-1731

Splash Shields

The SEM servomotors carry an IP65 rating from the manufacturer that means that the motors are liquid resistant - NOT liquid proof. The position of the servomotors on a machine, especially the Y-axis motor on knee mills, can allow flood coolant or machine way lube to pour directly onto the motor. Over a period of time, coolant or way lube seeps into the motor, rendering it totally useless. The only fix is to buy a new motor. Since this kind of damage is not covered under any warranty, the end user is forced to purchase a brand new motor.

A far easier solution is to install a simple sheet metal shield over the motor in order to prevent liquids and oils from pouring directly onto the motor. This shield doesn't have to be anything fancy. It just has to divert any liquids or oils away from the motor, must stay in place, and must not interfere with the motion of the axes, or catch any motor or limit switch cables. This simple shield will save you or your customer the cost of a brand new servomotor as well as the associated down time.

Limit / Home Switches

Routing the Control Cables

Depending on the use of the mill protection for the cables may need to be provided for. The best protection is Sealtite conduit. Spiral Wrap provides good protection from abrasion, plus and minus of both.

Cables to the head

Knee mill with a solid ram

If possible the cables to the head should be routed through the mill casting, picture of cabling through a BPBoss, Rhino.

Knee mill with a movable ram

Enough slack needs to be left in the cables to allow for complete forward and reverse movement of the ram, picture of Gain mill in forward and reverse position.

Bed mill

A flexible wire track should be used, picture of wire track, manufacturer, where to order from, average price.

If no wire track is used enough slack needs to be left in the cables to allow complete movement of the head. Care must be taken that the cables do not get caught on stuff, limit switches, Work light, Etc, and that they do not interfere with the correct operation of the limit switches.

Cables to the table/bed

Knee mill

Slack must be left in all of the cables to allow for the up and down position of the Knee, picture of Gain mill, knee up and knee down.

Slack must be left in the cables for the X axis motor, picture of Gain mill, table completely forward and left and table completely back and right.

Moving limit switches

Bed mill

The Y axis motor usually does not move and is located at the front of the mill. If possible route the motor cable through the casting from the back.

The limit switches on a bed mill normally only move forward and back with the bed. Slack must be left in the cables for this movement.

The X axis motor and Ball screw are sometimes mounted in the Saddle with the Ball nut connected to the Table. In this case the X axis motor only moves forward and back. On other mills the X axis motor and Ball screw are mounted on the Table and the Ball nut is connected to the Saddle. In this case the X axis motor will move both forward and back and left/right. Slack must be left in the cables to allow for this movement.

Cables to the lube pump

The lube pump cable can usually be routed along with the cables to the head or the table depending on the physical location of the lube pump.

Cables to the flood coolant pump

On a Knee mill and a small bed mill the flood coolant pump is located in a sump in the bottom rear of the mill casting. There is usually an access panel to this space. A hole can be drilled through this panel for a conduit connector or a strain relief connector. A U shaped slot can be cut at the edge of this panel to allow cables to pass through while still being able to completely remove the panel for access to the flood pump area.

Cables for an ATC (Umbrella type)

If the cables for the ATC Carousel are run through conduit the 3-phase power cable and the control cable should be in separate conduits to avoid electrical noise.

Connecting the control cables

Refer to the Schematic Diagram that came with the control for all wiring of components. Connections to components on the machine are shown inside of Thick Dotted Lines.

Inverter Installation

Inverter and Braking Resister Mounted Inside Magnetics Cabinet

The inverter should only be mounted inside of the magnetics cabinet if it is small enough to leave the required space for cooling as recommended in the inverter manual. The braking resister should also be mounted where there is sufficient room for cooling. It should be mounted vertically. (Heat rises) Braking resisters can get HOT. All cables should be kept well away from the braking resister.

Inverter and Breaking Resister Mounted Outside Magnetics Cabinet

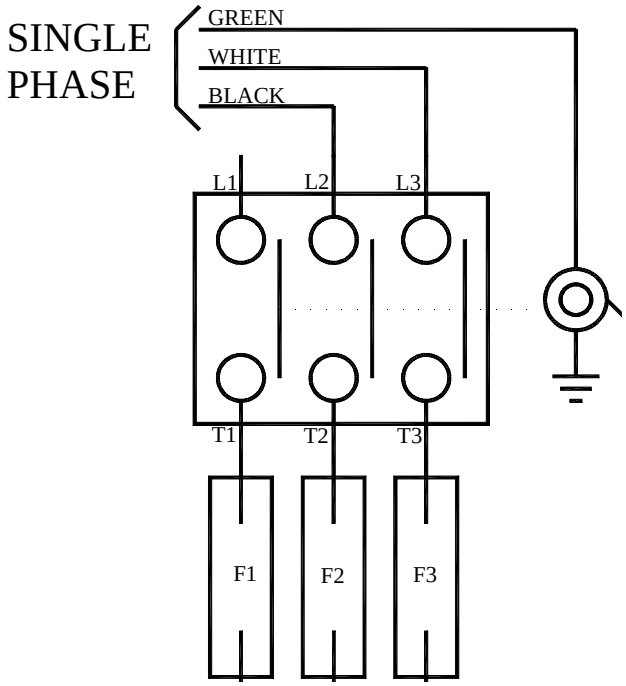
When the inverter is mounted externally, a separate cabinet should be used. A cabinet is much cheaper then replacing an inverter that has been damaged due to chips and/or fluid. The size of this cabinet should be based on the inverter manual recommendations. If the breaking resistor is to be mounted externally, some form of guard should be used to keep someone from accidentally coming in contact with it. Braking resisters can get HOT.

Single Phase operation of Centroid Controls

A phase converter is not recommended or needed for running most controls using single phase. Most 3-phase motors can be operated using a Inverter for less cost than a noisy phase converter.

A. Main control

1. DC servo systems made after January 1 1995 are designed to use single phase to power the main servo control. The L1 line is not used by the control. It is only used to power 3 phase motors. For single phase operation make sure that power is connected to L2 and L3.
2. AC servo systems can also be powered from L2 and L3. Single phase operation is not recommended for systems using servo motors over 1KW. Systems with only 2 axis can use up to 2KW motors although most systems that use larger servo motors also use larger spindle motors and the use of single phase for spindle motors over 7.5hp is not practical.



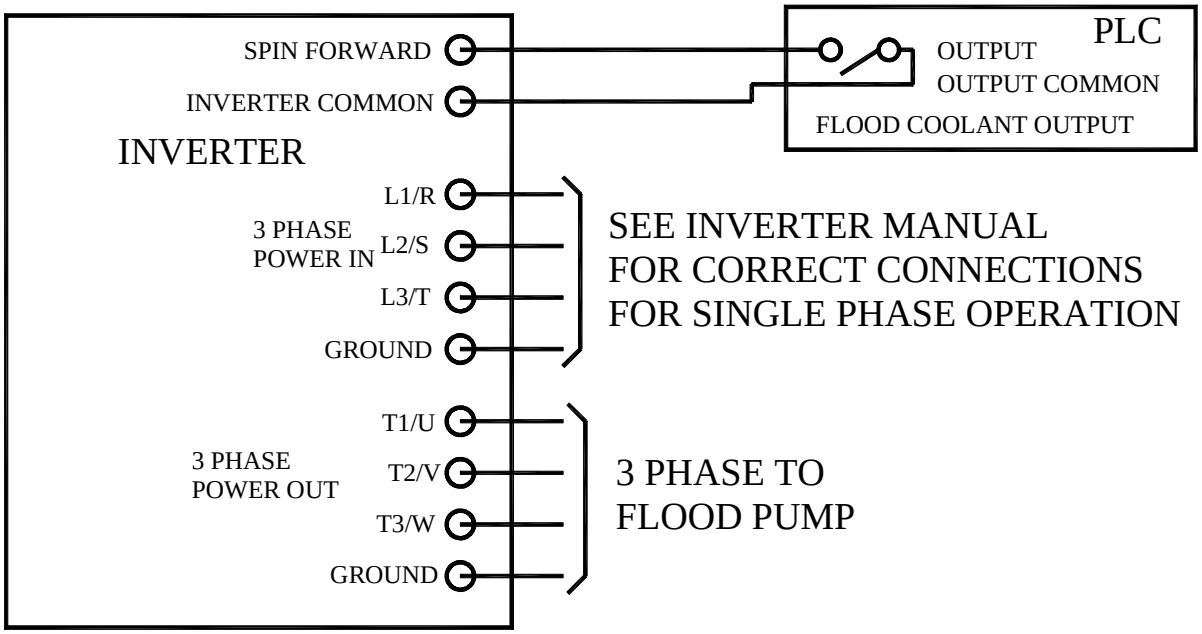
B. Spindle motor operation with single phase.

1. Spindle motor of 7.5hp or less can be run using single phase power. Spindle motors higher than 7.5hp require to much current (Over 100amps) to be practically run with single phase.
2. A inverter is recommended for spindle operation. A inverter to replace the reversing contactors is much less expensive than a phase converter. (A 5hp inverter is around $450) Most controls can be set up for spindle speed control but the inverter can be used to just turn the spindle on and off also.
3. A inverter capable of single phase operation must be used. The following inverters will work.
   a. Mitsubishi FR-A500 series for ATC units.
   b. Hitachi SJ100 series
   c. Automation Direct GS1 and GS2
4. For other models consult the manufacturer to find out if they are capable of single phase operation.
5. Oversize the inverter accordingly.

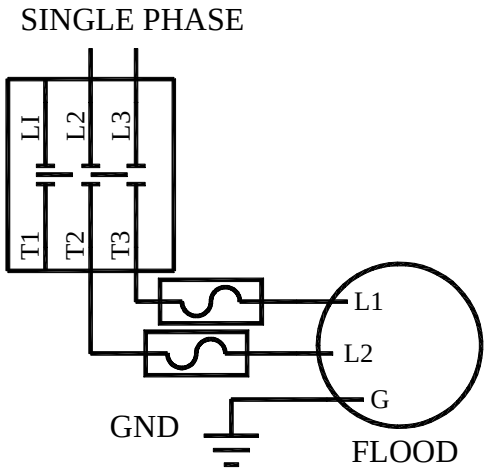
3 phase Spindle motor size	1.5 Hp	2 Hp	3 Hp	5 Hp	7.5 Hp
Inverter size	2 HP	3 Hp	5 Hp	7.5 Hp	10 Hp

6. For correct connection of inverter for single phase operation refer to the inverter manual. Different inverters use different connections for single phase operation.

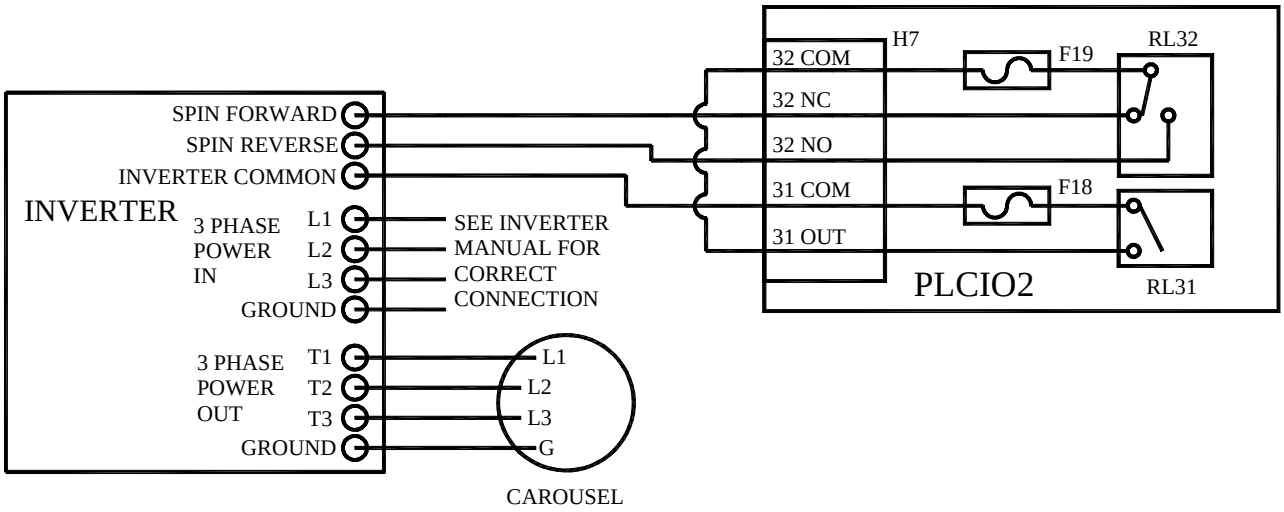
C. To run a 3 phase flood pump using single phase an inexpensive inverter ($100) can be used to run the flood pump in place of a contactor. The output of the inverter is set to 60Hz and the output on the PLC is used to turn the inverter on and off. Disconnect the original contactor control wires and connect the inverter control wires as shown.



D. To run a single phase flood pump the overload protector on the flood contactor must be removed and replaced with the correct size fuses.

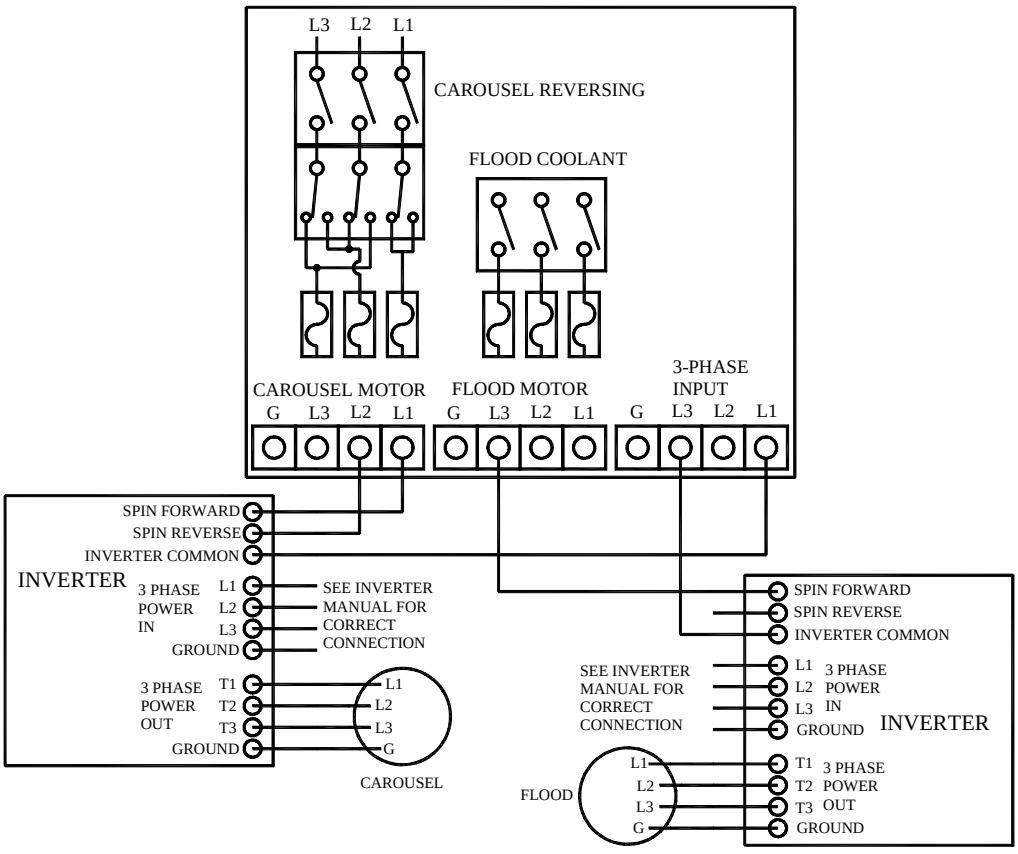


E. To run a 3phase ATC Carousel motor with a PLCIO2 using single phase an inexpensive inverter, around $100, can be used to run the carousel motor. The output of the inverter is set to 60Hz and the outputs on the PLC are used to turn the inverter on and off. Disconnect the original contactor control wires and connect the inverter control wires as shown.

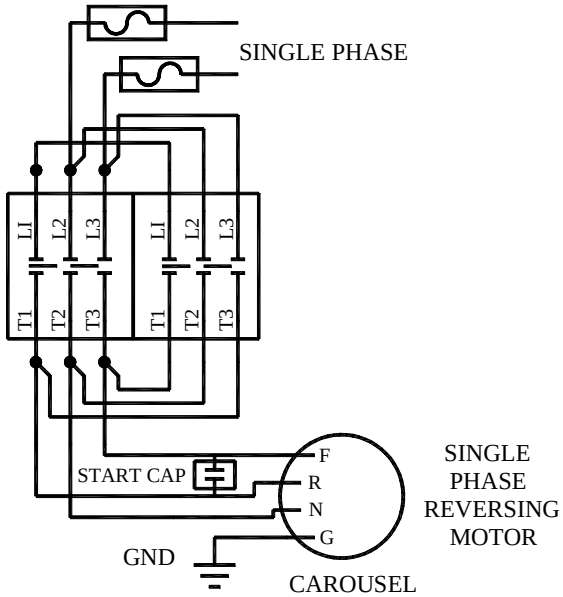


F. To run a 3phase ATC Carousel motor and flood pump with a RTK3 PLC using single phase an inexpensive inverter, around $100, can be used to run the carousel motor and another inverter to run the flood pump motor. The output of the inverters is set to 60Hz and the outputs on the PLC are used to turn the inverters on and off. Disconnect the original power wires and connect the inverter control wires as shown.

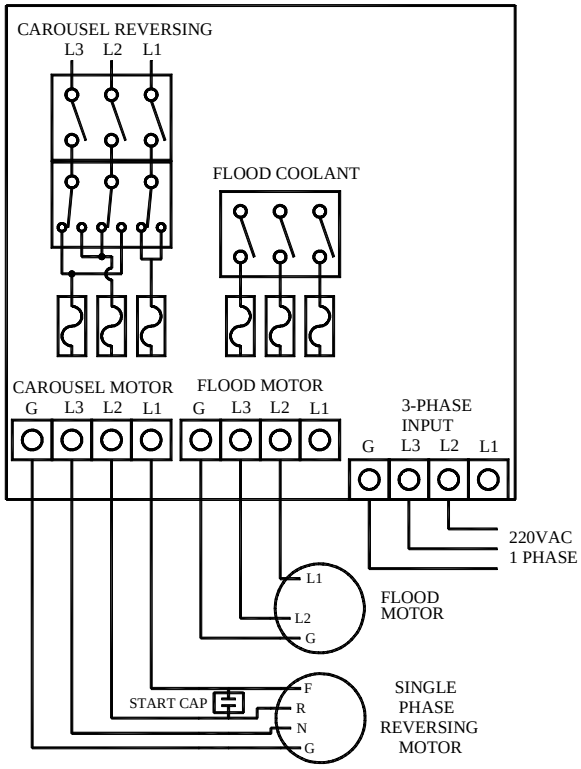
G.



H. To use a single phase reversing carousel motor with a PLCIO2 plc. Connect the incoming single phase power and motor wires as shown without changing the contactor control wires.



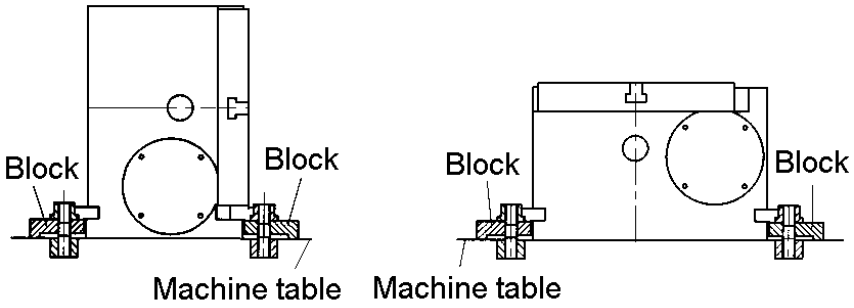
I. To use a single phase reversing carousel motor and single phase flood pump motor with a RTK3 plc. Connect the incoming single phase power and motors as shown.



Rotary Tables

Installation steps

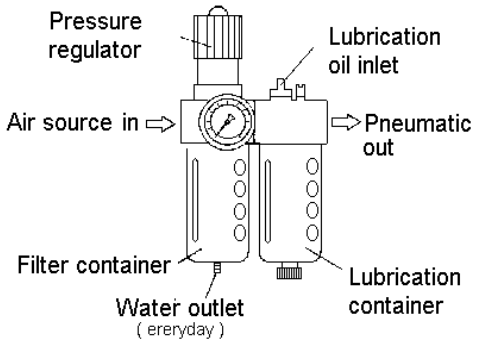
7. Remove rust-preventing grease, and then put rotary table on the machine table.
8. Connect MS connector and conduit and air line.
9. Align the table to the machine.
10. Clamp the rotary table on the machine table.
11. Make sure the 4th axis is turned on.



(3 - 1)

Notice

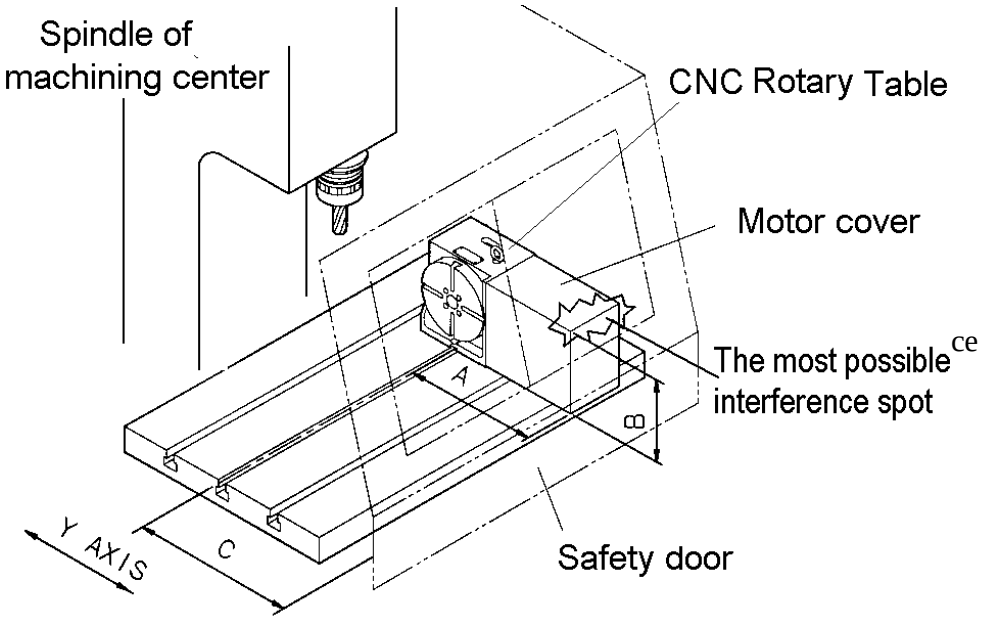
Clamping force is driven by air, it is very important to install filter, regulator, and lubrication as close as possible to the pneumatic equipment for which they are used.



Filter & Regulator & Lubricator for Pneumatic

A. To protect all components from rust it is necessary to install filter, regulator & lubricator for pneumatic supply.
B. The pneumatic pressure should be set between 57-85 psi. When pressure is under 57 psi, the mechanism's clamping accuracy will be reduced.
C. Drain the water inside filter container everyday before operation.

How to avoid interference between Rotary table and M/C



A → Width from center of guide-block on the CNC rotary table to the end of motor cover.
B → Height of motor cover.
C → M/C moving to end of the Y axis, the distance between locking center of CNC rotary table to the safety door.

NOTE: Before you purchase or install the rotary table, check the width of 'A'. It should be smaller than the dimension of 'C'. Also consider the height of the table 'B', so it doesn't interfere with the machine surround.

MAINTENANCE & LUBRICATION

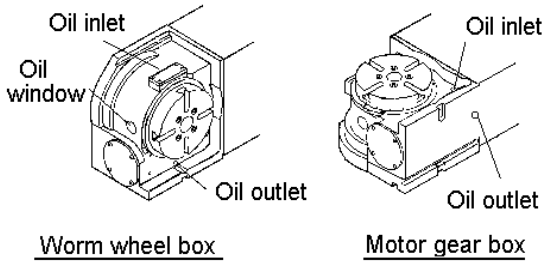
Lubrication replacement time and condition

1. Normal operation: Change lubricant every six months.
2. Continuous operation: Change lubricant every three months.
3. If machine is idle over six months, change the lubricant oil before use.

Steps to change lubricant

A. Drain oil by unscrewing oil outlet screw.
B. Make sure the oil reservoir is clean.
C. Open the oil inlet cover.
D. Refill with recommended lubricant into oil reservoir.
E. Replace the oil inlet cover.

Be sure the lubricant is over center of the oil window.



Worm wheel box Motor gear box

Recommended lubricants:

CPC-HD	HD150
Texaco-Meropa	150
Shell-Omela	100
Mobil-Mobilgear	629

Characteristics of recommended lubricants:

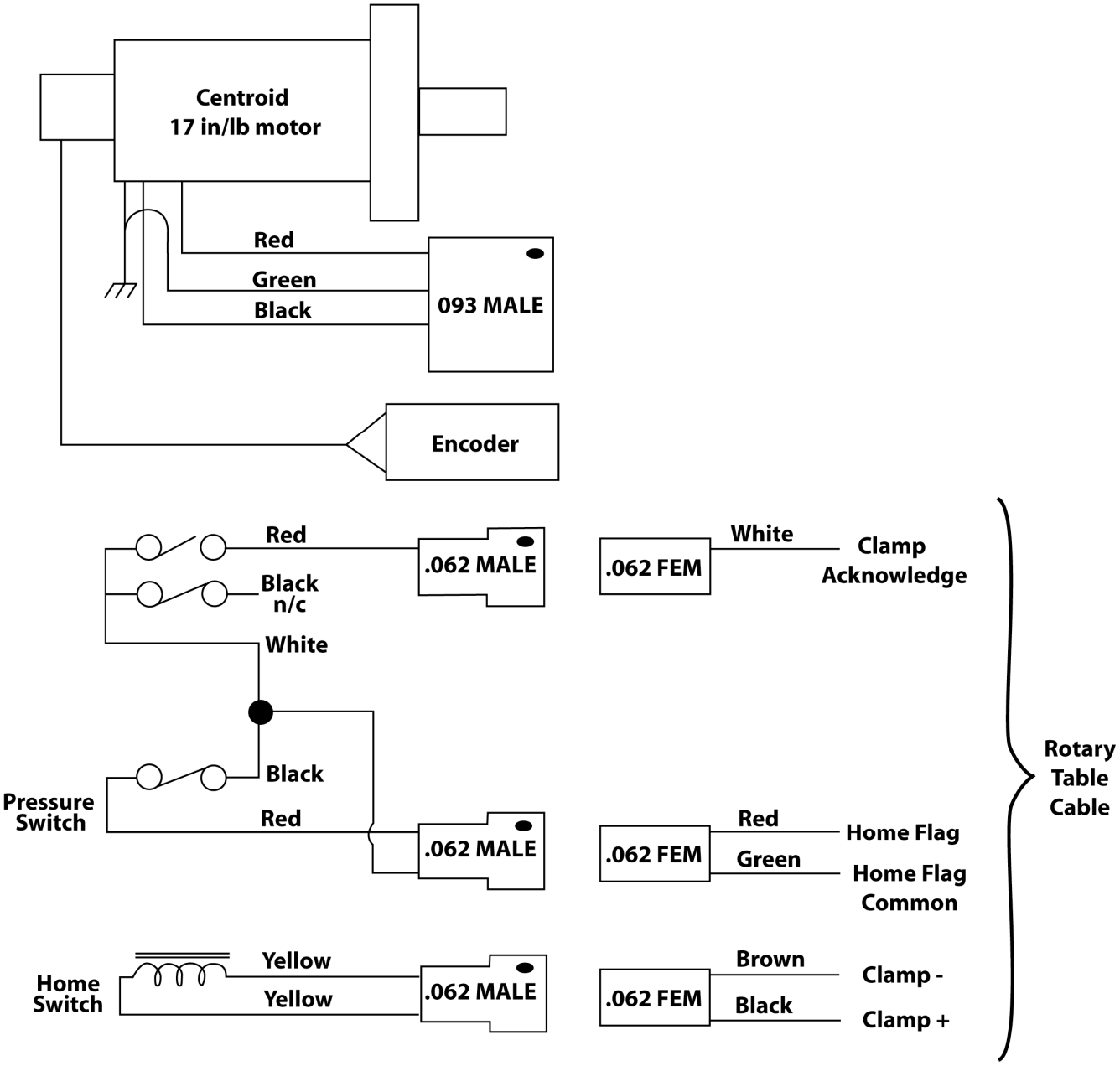
Agma grade	4-EP
Flash point	400° F
ISO Grade	100-150
SUS Viscosity	At 100° F, 72.5
	At 210° F, 75. 5

Notes:

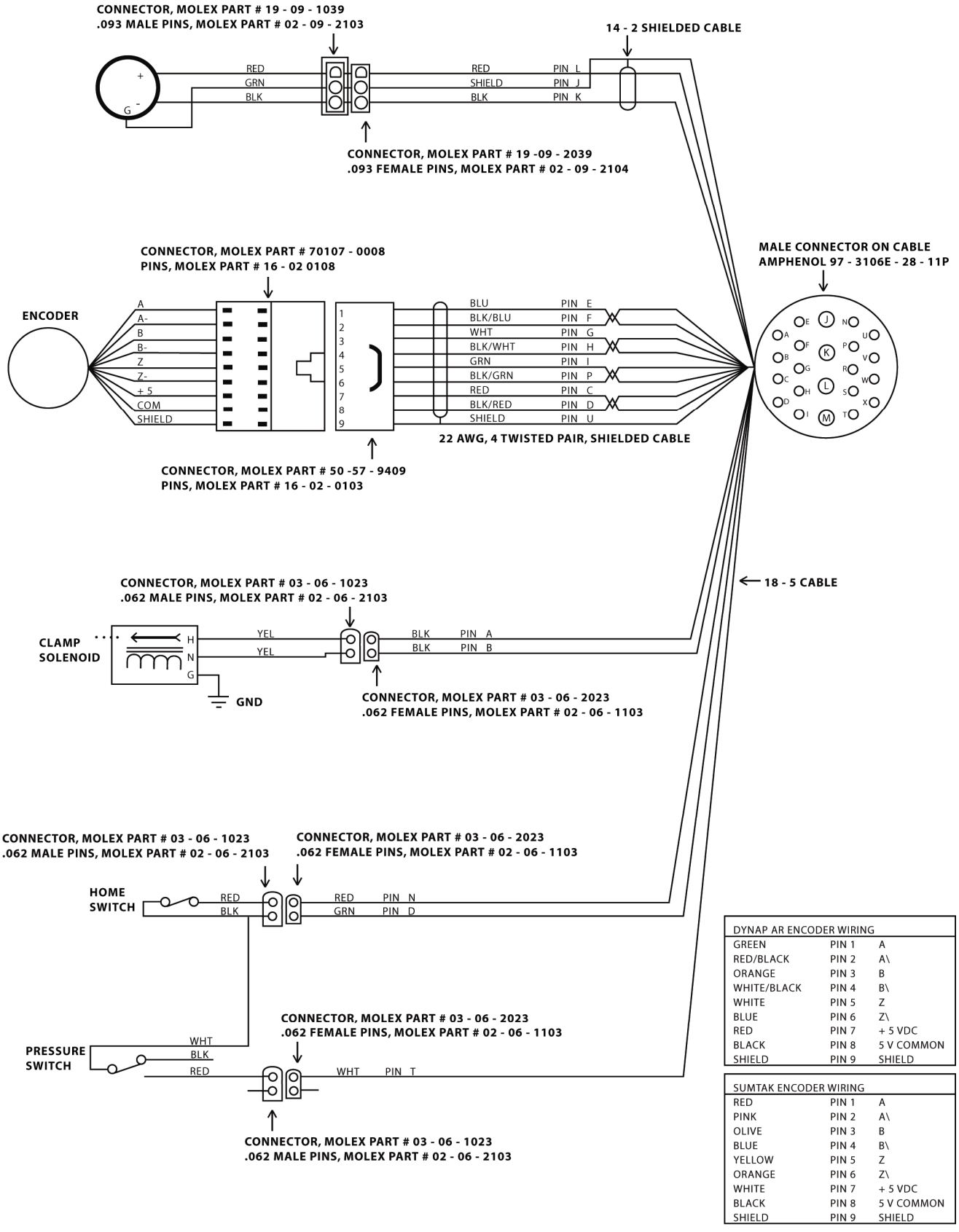
(1) Be sure no chips fall into table body during refilling oil.
(2) Fill oil up to middle of the scale window.
(3) Clean chips and coolant off of Rotary Table every day.

Rotary Table Parameters

Rotary Table	Mtr/Tbl Ratio	Slow Jog	Fast Jog	Max Rate	Rev/ degree	Encode counts	Limit + -	Home + -	Dir
Troyke	180:1	360	720	1,200	0.5	8,000	0 0	* *	N
Yuasa SUDX 220 w/5.27	90:1	2,000	3,937	3,937	0.25	8,000	0 0	* *	N
Yuasa SUDX 220 w/6.00	90:1	2,000	12,000	12,000	0.25	8,000	0 0	* *	N
Yuasa SUDX 140 w/5.27	70:1	2,000	3,937	3,937	0.2	8,000	0 0	* *	N
Yuasa SUDX 140 w/6.00	70:1	2,000	12,000	12,000	0.2	8,000	0 0	* *	N
Centroid RT-200	DC 90:1	2,000	9,000	9,000	.25	8,000	0 0	* *	N
	AC 90:1	5,000	11,500	11,500	.25	8192	0 0	* *	N
Centroid RT-150	DC 90:1	2,000	9,000	9,000	.25	8,000	0 0	* *	N
	AC 90:1	5,000	11,500	11,500	.25	8192	0 0	* *	N
Centroid RT-100	DC 90:1	2,000	9,000	9,000	.25	8,000	0 0	* *	N
	AC 90:1	5,000	11,500	11,500	.25	8192	0 0	* *	N


Centroid
17 in/lb motor
Red
Green
Black
093 MALE
Encoder
Red
Black
n/c
.062 MALE
.062 FEM
White
Clamp
Acknowledge
White
Pressure
Switch
Black
Red
.062 MALE
.062 FEM
Red
Home Flag
Green
Home Flag
Common
Home
Switch
Yellow
Yellow
.062 MALE
.062 FEM
Brown
Clamp -
Black
Clamp +
Rotary
Table
Cable

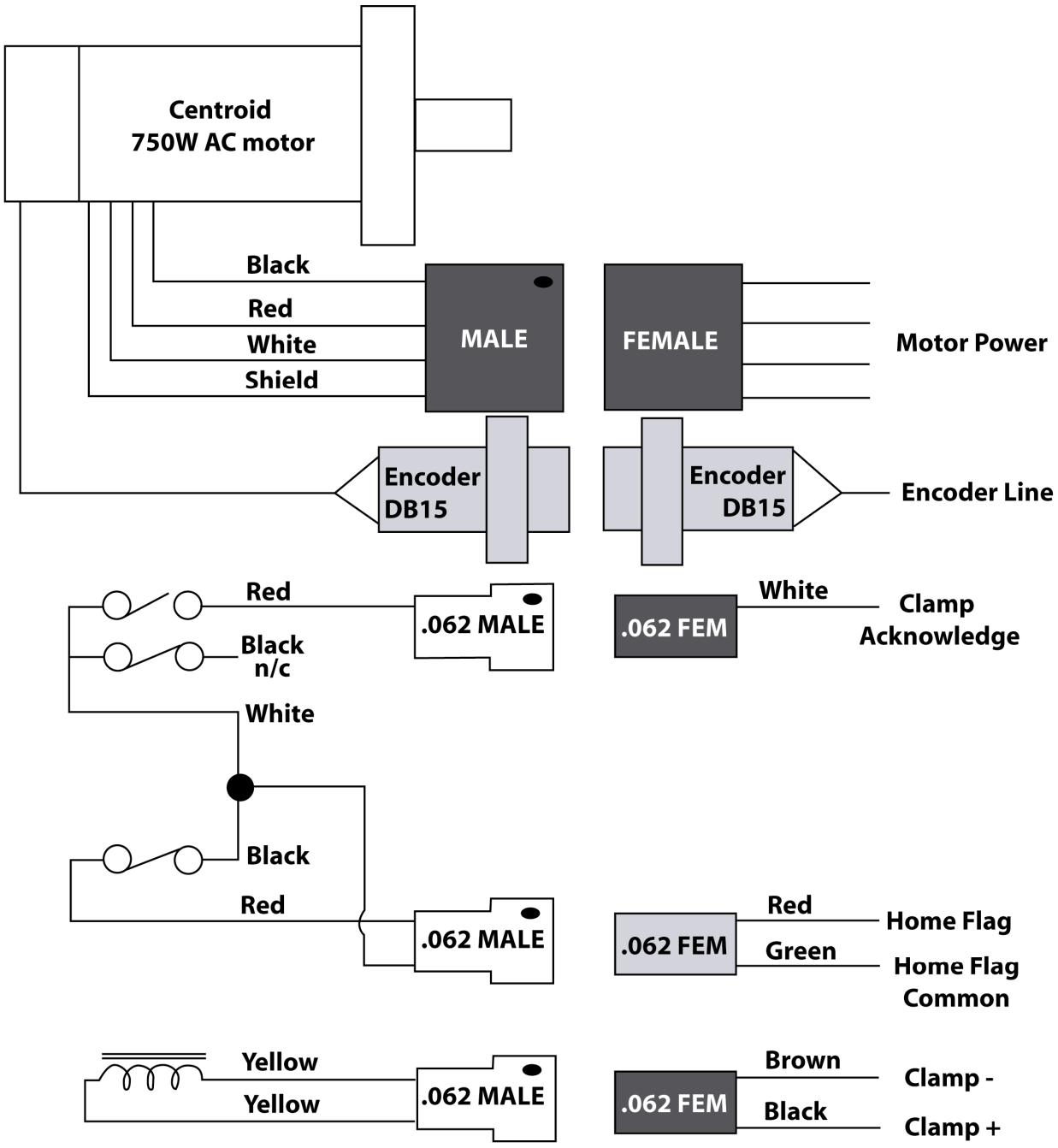
DC Wiring



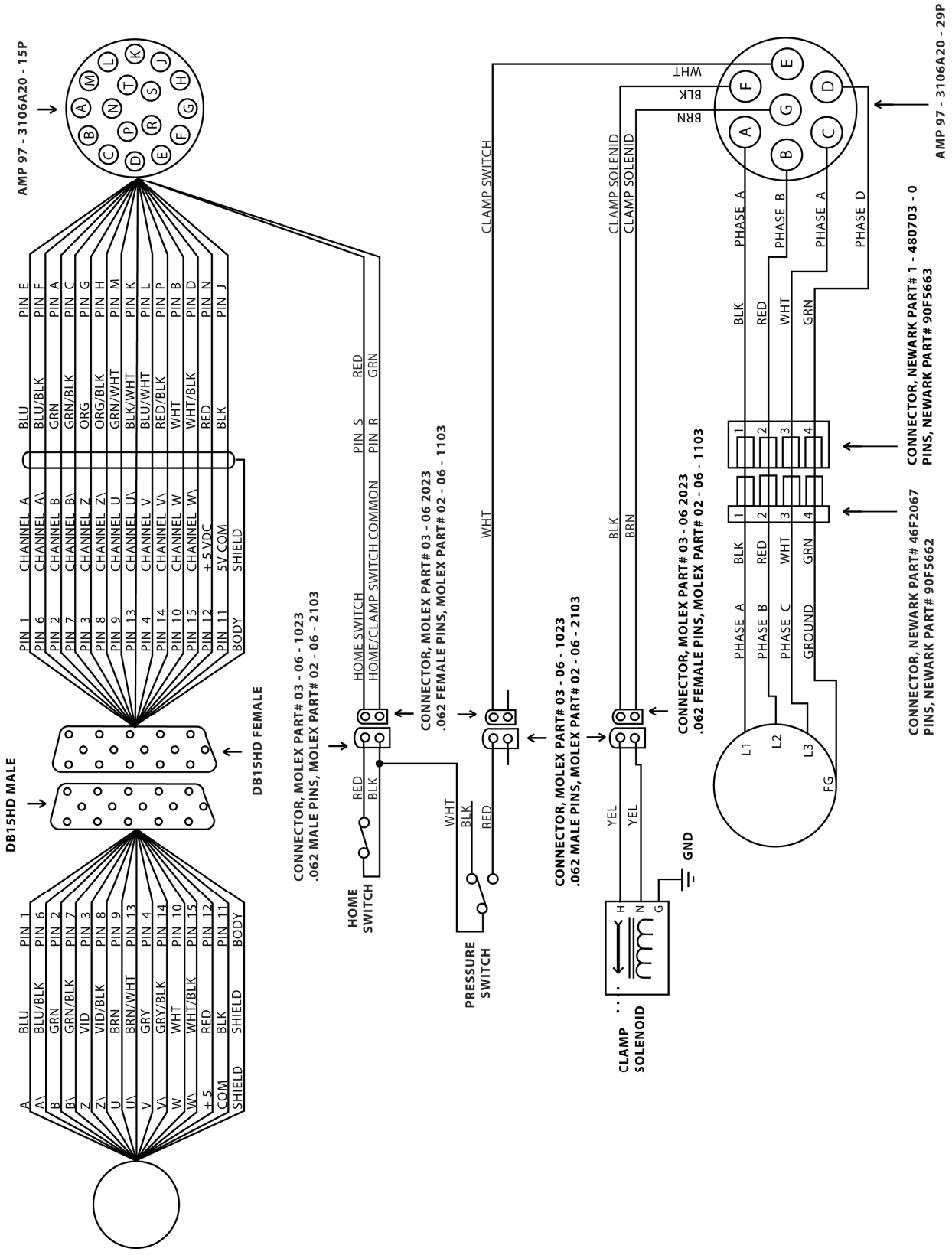
DYNAP AR ENCODER WIRING		
GREEN	PIN 1	A
RED/BLACK	PIN 2	A\
ORANGE	PIN 3	B
WHITE/BLACK	PIN 4	B\
WHITE	PIN 5	Z
BLUE	PIN 6	Z\
RED	PIN 7	+ 5 VDC
BLACK	PIN 8	5 V COMMON
SHIELD	PIN 9	SHIELD

SUMTAK ENCODER WIRING		
RED	PIN 1	A
PINK	PIN 2	A\
OLIVE	PIN 3	B
BLUE	PIN 4	B\
YELLOW	PIN 5	Z
ORANGE	PIN 6	Z\
WHITE	PIN 7	+ 5 VDC
BLACK	PIN 8	5 V COMMON
SHIELD	PIN 9	SHIELD

DC Rotary Table Connections



AC Rotary Table cable uses two 1/2" conduit lines
All wires and connectors indicated in light gray go through encoder conduit
All wires and connectors indicated in dark gray go through power conduit



AC Rotary Table Connections

Chapter 2
Software Setup

CNC Linux Hardware Requirements
Basic Requirements

- 586 400 MHz Processor (1 GHz recommended)
- 128 MB RAM (1 GB RAM recommended)
- 512 MB Flash Disk (and IDE flash card reader)

Supported Motherboards
The following is a current list of motherboards that have been tested with CNC Linux. These motherboards have network devices that are supported by CNC Linux. CNC Linux should work with other motherboards provided they satisfy the basic requirements.

- Gigabyte GAM61S ME-S2
- Gigabyte GAM61P ME-S2
- MSI K9VGM-V
- MSI MS-6368, MS-6787
- VIA Epia M10000 Mini-ITX
- AAeon PCM-6890-B10

Supported Network Cards

- VIA Rhine series
- RealTek RTL-8139, RTL-8129, RTL-8130
- Intel EtherExpress Pro, EtherExpress Pro 100, EtherExpress 10
- 3com 3c509, 3c529, 3c579 "Etherlink III"
- 3com 3c515 ISA "FAST Etherlink"
- SIS 900, SIS 7016

CNC Linux & CNC10 Board Level Installer CD
What can I do with this CD?

1. Install CNC Linux and CNC10 on a hard drive or flash disk.
2. Make CNC10 update disks.

How does it work?
This is a bootable CDROM that will boot a diskless version of CNC Linux. It may be used on the target system itself or on a different computer. Once booted, the installer program runs.

Is it safe to use the installer on my desktop computer?
If you make only update disks, it is safe.

If you want to use your desktop computer to install CNC Linux on a hard drive or flash disk, you need to be careful. It is possible to wipe out any operating system and files on the computer. Back up your system. Make sure that you know how to identify the hard drives by how they are plugged into the IDE ports and the jumper settings on the drives. The safest way to proceed is to completely disconnect the power cable of any hard drive that contains valuable data. Once you can positively identify the target drive or flash disk and you are comfortable using the installer program, you can reconnect it. This is more convenient if you want a "dual-boot installer system" so that you can

boot either the hard drive or the installer without changing cables; however, you still need to be careful when you use the installer.

You cannot install CNC Linux on a different partition on the same hard drive as other operating system(s). There is currently no need to support this option.

How do I identify and configure my drives?

Never disconnect or connect any cable until your system is safely shut down and the power is off!

Use the following table to cross-reference IDE ports and drive jumpers to names used by BIOS Setup and Linux:

IDE Port	Drive Jumper	BIOS Setup Name	Linux Name
IDE 1	Master	Primary Master	hda
IDE 1	Slave	Primary Slave	hdb
IDE 2	Master	Secondary Master	hdc
IDE 2	Slave	Secondary Slave	hdd

Typical installation configurations:

System Type	Device	Drive
Target (Control)	Target Flash Disk or Target Hard Drive	hda
	CDROM	hdb
System Type	**Device**	**Drive**
Dual-Boot Installer (Desktop)	CDROM	hda or hdb
	Desktop Hard Drive	hda or hdb
	Target Flash Disk or Target Hard Drive	hdc

What are the requirements for the installer system?

Processor:	Pentium (586) 800 MHz
RAM:	128 MB minimum
CDROM:	ATAPI IDE CDROM (Required for installation only)
Video/Display:	VESA Compatible XVGA
Disk:	IDE Hard Drive or Flash, 128 MB Minimum
Mouse:	PS/@ MS Compatible (Required for installation only.)

How do I Install CNC Linux?

You may use the installer on the "target system" (the control system) itself or the "installer system" (a different computer.)

To install CNC Linux and CNC10 directly to a flash disk:

1. Insert this CD ROM into the CD ROM drive of the installer computer.
2. Shut down and power off the installation computer.
3. Configure, install and identify your hard drives according to the tables, above.
4. Turn on your system.

5. If necessary, press Del during boot to change the BIOS settings so that your system will boot from the CD ROM.
6. At the "boot:" prompt, press Enter. Or simply wait and CNC Linux will start.
7. When the Installer loads, click on the "Install CNC linux" button.
8. Under "Flash Card Device" select the target drive, the drive on which you want to install CNC Linux. *Caution! The installer will format the drive the instant that you click the "Install" button. You will not get a warning.* See above: "How do I configure and identify my drives?" Get it right. Now.
9. Select Install CNC10 Mill, Lathe, or None.
10. Choose a PLC program from the list on the right.
11. Choose "DC Brushed" or "AC Brushless", under "Drive Type".
12. Review your choices, and correct any mistakes now. This is your last chance.
13. Click the "Install" button. The installer begins formatting the selected, drive the instant that you click the button.
14. If you chose to install CNC10, the software will be installed according to the selected options immediately after the drive has been formatted.
15. When the "Installation Complete" message appears, click "Close".
16. Click "Exit", and "Exit" on the main screen. The installer automatically begins the reboot process.
17. When the system nears complete shutdown, it will pause and eject the CD. Remove the CD and wait for the restart.
18. If this is the target system, it should start up on reboot.
19. If this is a dual-boot installer system, turn the power off when the system reboots, remove the flash disk or hard drive. Install the drive in the target system as Linux drive hda.

From Power Up to CNC10

When CNC10 is installed on a CNC Linux system, CNC10 is automatically started when the system is computer turned on. The following is a step by step description of what happens when the system is powered up and how CNC10 is started.

1. **LILO loads Linux kernel** - LILO stands for Linux loader. This program is stored in the Master Boot Record (MBR) of the disk containing CNC Linux. Its job is to load the kernel. The kernel is currently /boot/vmlinuz-2.4.7-10.
2. **Execute /sbin/init** - This program executes the /ete/inittab file.
3. **Process /etc/inittab file** - This file executes startup scripts for the current runlevel and logs in root.
4. **Execute scripts in /etc/rc3.d directory** - Theses scripts mount file systems, bring up the network, start the PLC executor, load the driver for the PCI CPU10 card.
5. **Automatically login root** - After executing rc3.d scripts, inittab calls /etc/autologin which automatically logs in the root user.
6. **Execute /cncroot/c/cnc10/.bash_login** - This file is executed whenever root logs in. In .bash_login is a command to start X windows.
7. **Execute /usr/X11R6/bin/startx** - This script starts the sawfish windows manager.
8. **Execute /cncroot/c/cnc10/.sawfishrc** - This file contains commands to move the mouse pointer, bind Alt-F6 to the action to launch an xterm and call "/cncroot/c/cnc10/bin/runcnc.sh --mill" (mill) or "/cncroot/c/cnc10t/bin/runcnc.sh --lathe" (lathe).
9. **Execute /cncroot/c/cnc10/bin/runcnc.sh** - This script replaced CNC7M4.BAT and CNC7T4.BAT. It starts CNC10.

CNC Linux file system

Although CNC10 and CNCEDIT maintain the illusion of a DOS-like file system with drive labels, the real CNC Linux file system is completely different. Linux does not use drive labels. Disks and partitions may be mapped to any directory in the Linux file system. The following list provides descriptions of the main CNC Linux directories contained in the root '/' directory.

- **/bin** - Contains many of the basic unix programs such as *bash*, *cat*, *cp*, *less* and *ps*
- **/boot** - Contains the Linux kernel *vmlinuz-2.4.7-10*
- **/cncroot** - Directory containing all CNC10 programs and files
- **/dev** - Contains devices files. These files are used for communicating with hardware.
- **/etc** - Contains system configuration files and start up scripts. Startup scripts are located in **/etc/init.d**
- **/lib** - Contains basic shared libraries used by most CNC Linux programs. Kernel drivers are found in **/lib/modules**
- **/mnt** - The floppy disk and windows shares are automatically mounted in subdirectories of this directory.
- **/proc** - Contains files created by the kernel for drivers and running processes. These files provide useful system information. The file */proc/driver/plc* exists when the PC PLC executor is loaded.
- **/root** - Home directory for user *root*. When CNC10 is installed, roots home directory is changed to **/cncroot/c/cnc10** (mill) or **/cncroot/c/cnc10t** (lathe).
- **/sbin** - Contains system programs such as *lilo*, *fdisk*, *fsck*, *ifup* and *ifdown*.
- **/tmp** - Temporary directory. Files may be periodically deleted.
- **/usr** - Contains directories for non-essential CNC Linux programs and libraries. These programs add to the functionality of CNC Linux and are required for CNC10.
- **/var** - Contains variable data files. These files are used by some programs such as X and samba.

For more information regarding the standard Linux filesystem, go to **http://www.pathname.com/fhs/.**

/cncroot directories

Since Linux does not use drive letters, CNC10 and CNCEDIT emulate drive letters to provide a familiar look to current DOS/Windows users. Paths such as *c:\cnc10\ncfiles\cirpock.cnc* are really */cncroot/c/cnc10/ncfiles/cirpock.cnc*. The user does not need to be aware of the real paths of their files unless they use the Linux command window. The following is a description of directories used by CNC10.

- **/cncroot** - All directories used by CNC10 are subdirectories of this directory. This directory contains the one letter subdirectories *a* and *c*. More one letter directories are created when Windows shares are mounted. These one letter directories are used to emulate drive labels in CNC10 and Cncedit.
- **/cncroot/c** - Contains everything that CNC10 sees as residing on the c drive.
- **/cncroot/a** - Directory where the floppy drive is mounted in CNC10 and Cncedit. Appears as *a:* in CNC10 and Cncedit.
- **/cncroot/x** - Directory where the Windows share associated with drive x: is mounted. Windows share drive letter associations are assigned in *cnc10.net* in the cnc10 directory.
- **/cncroot/c/cnc10** (and **/cncroot/c/cnc10t**) - CNC10 main directory. Similar to C:\CNC7 and C:\CNC7T directories with Cnc7. Contains ALL CNC10 configuration files. PLC programs and source files are also found here. CNC10 does NOT have a separate PLC directory.
- **/cncroot/c/cnc10/ncfiles** - CNC10 job files directory. Same as C:\CNC7\NCFILES.
- **/cncroot/c/cnc10/bin** - Directory containing executables and scripts used by CNC10.
- **/cncroot/c/cnc10/doc** - Directory containing help files used by the *help* command.
- **/cncroot/c/intercon** (and **/cncroot/c/icn_lath**) - Intercon file directories.
- **/cncroot/c/plx** - Directory containing PCI Cpu7 card driver.

Important CNC Linux commands:

- ls - List directory contents
- cd - Change current working directory
- cp - Copy file(s)
- mv - Move file(s)
- rm - Remove file(s)
- mkdir - Create directory
- rmdir - Remove empty directory
- less - Display a text file

- pwd - Print current working directory
- cnc10m4 - Start CNC10 mill
- cnc10t4 - Start CNC10 lathe
- cncedit - Edit a text file
- statplc - Display status of plc executor
- version - Display CNC Linux version number
- cnc10update - Install update from floppy disk outside of CNC10
- install-cnc10 - Install CNC10, Utilities and Config files
- install-config - Only install CNC7 or CNC10 config files
- 7to10 - Convert CNC7 config files to CNC10 config files
- exit - Close the command window (xterm)
- help - Displays CNC Linux help information

CNC Linux hot keys

- Alt-F6 - Launch a command window (xterm)
- Alt-Tab - Switch between applications
- Tab - Autocompletes file names at the command line
- Up/Down Arrow keys - Browses command history

Networking CNC Linux

Easier to Use Networking (Zero Configuration)

- Zero Configuration (Zeroconf) is a new, automatic configuration protocol
- Any machine honoring the protocol can immediately join the network.
- Currently used in wireless LAN's, for example, airports and Internet cafes.
- Randomly picks an IP address, then checks to see if it is in use
- Uses local address range 169.254.x.x -- over 65,000 possible addresses
- Windows uses modified Zero Configuration addresses by default

Default Control Automatic Network Configuration Sequence

1. Try to find a DHCP (Dynamic Host Configuration Procotol) Server:
    - Windows machine with Internet Connection Sharing turned on, or
    - Specially configured LAN server, router, or proxy server.
    - The DHCP server determines complete network configuration of the control
2. If DHCP search fails, fall back on Zero Configuration Protocol.

Network Considerations -- What Kind of Network do You Have?

- Simple Network -- a few Windows XP machines connected by hub or switch
    - Run Networking Wizard
    - Turn on File Sharing
- Simple Network with Internet Connection
    - Run Networking Wizard
    - Turn on Internet Connection Sharing
    - Turn on File Sharing
- Complex Network with Dedicated Servers
    - Consult the Network Administrator
    - May need to supply Administrator with control's HW address
    - Administrator may require use of a specific DHCP hostname
    - In many cases, no special configuration is required.

Accessing Windows Shares

- Edit file cnc10.net
- Add lines for each share, each containing:
    - //<computer-name>/<share-name> <CNC10-drive-letter> "<options>", for example:
    - // winserver/shareddocs e "username=joe,password="
- Shares will be mounted at boot.
- Access a share by its "drive" letter from the CNC10 load menu.

Sharing Control Files

- CNC10 will not share control files directly
- Set up a shared folder on a Windows machine (preferably one always turned on)
- Access the shared folder from the control using cnc10.net
- Load and save files to that shared directory
- Other computers will be able to access the files.

Manual Configuration

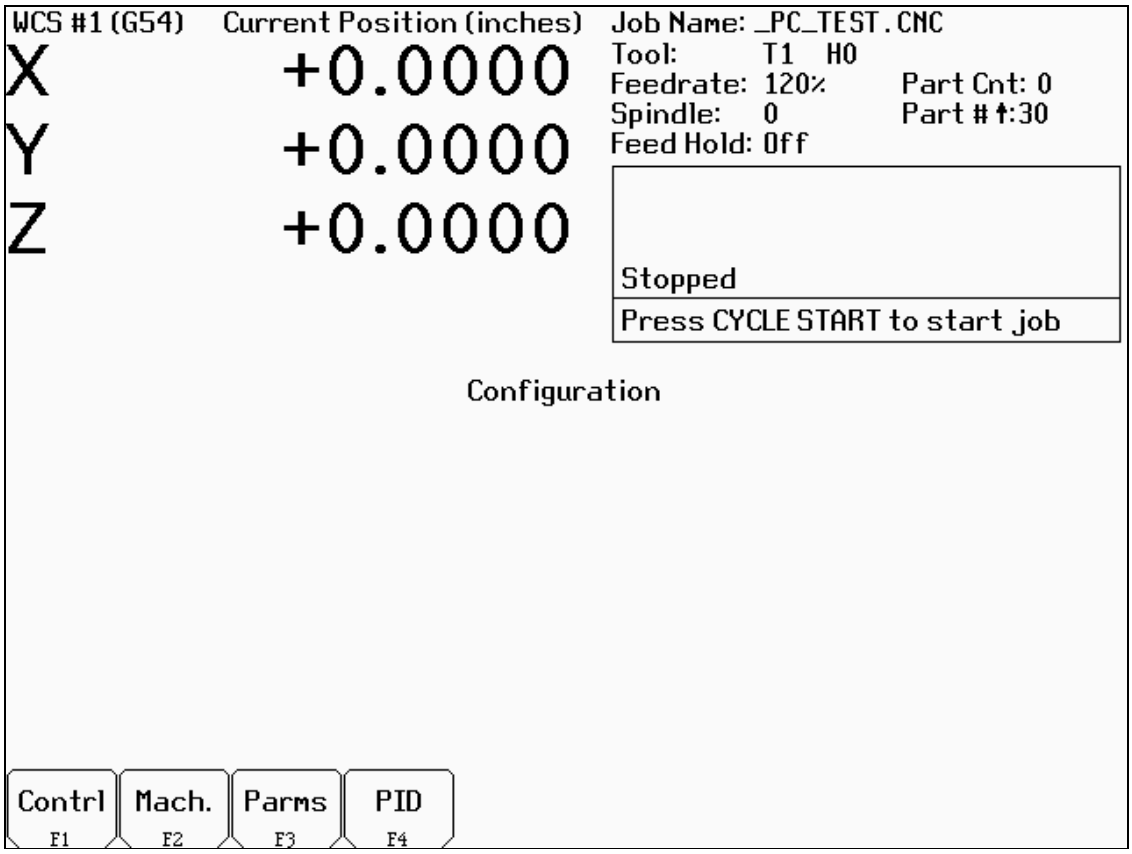
- Edit file **/etc/sysconfig/network-scripts/ifcfg-eth0** to manually configure system.
- Should only be necessary for complex networks with a System Administrator
- System Administrator will supply necessary data
- UI Configuration for Linux will be available in a future release
- Need to specify such things as IP address, mask, gateway, etc.

Networking-Related Console Commands

- **ifconfig eth0** - display network configuration
- **ifdown eth0** - disable network interface
- **ifup eth0** - initialize network interface
- **mount** - display mounted file systems including network shares
- **smb-mnt-entry //<computer>/<share> <drive> "username=<username>,password=<password>"**

Configuration

(F3 from Setup)



General

The first four options, <F1> through <F4>, will display a set of parameters. Each option is explained in detail below. The <ESC> key will return you to the previous screen (Setup).
The configuration option provides you with a means for modifying the machine and controller configuration. The majority of information in this section should not be changed without contacting your dealer. Some of the data, if corrupt or incorrect, could cause personal injury or machine damage.

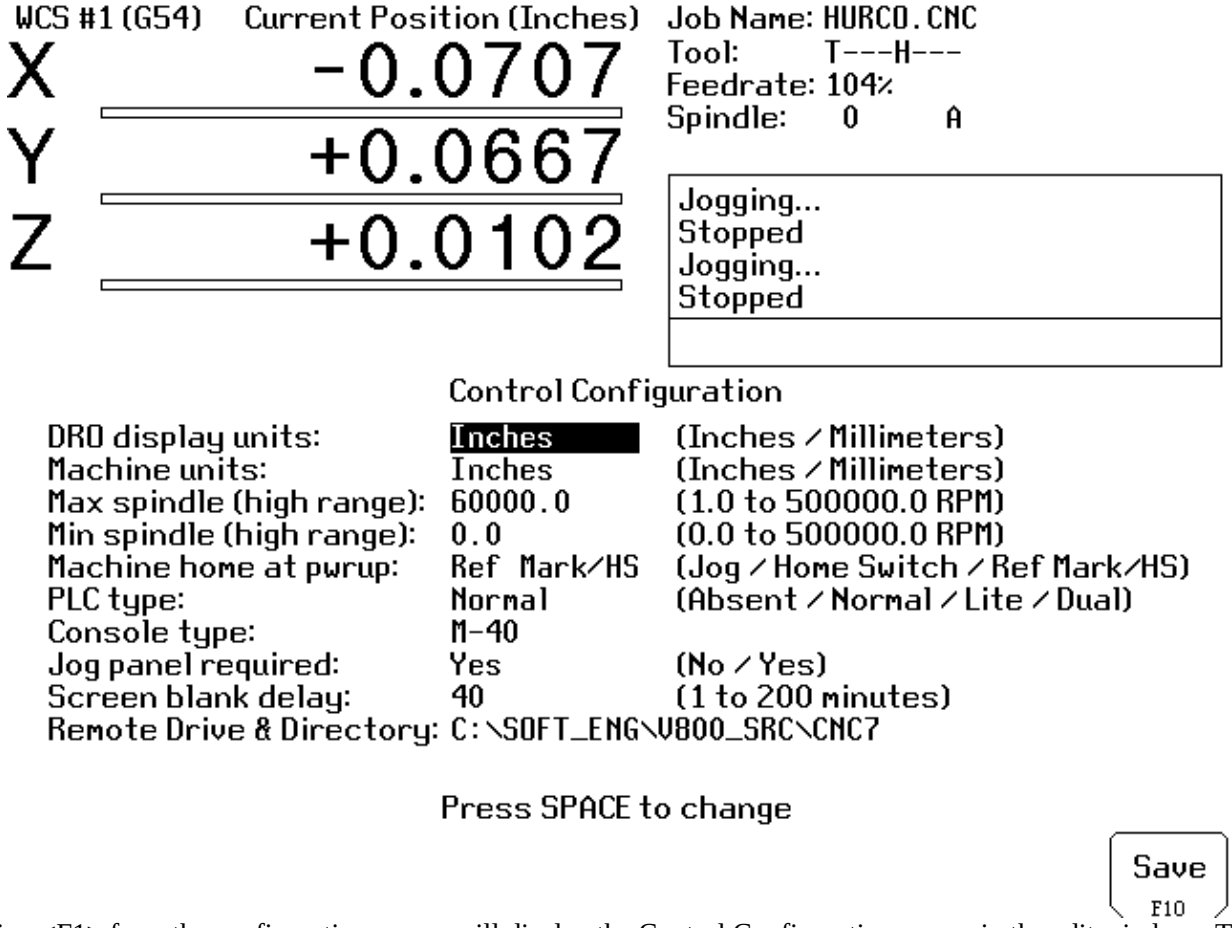
Password

When you press <F3> from the Setup Screen, you may be prompted to enter a password. This level of security is necessary so that users do not accidentally change vital parameters. The original default password is distributed in the documentation provided to the owner of the machine when the control is installed. The default password can be changed via Parameter 42. Instructions are given in the parameters section below.
If you know the password, type it and press <ENTER>. If the password you enter is incorrect, a message will appear telling you the password was incorrect and the password prompt will reappear. Pressing <ESC> will remove the prompt.

If you don't know the password, simply press <ENTER>. You will be given access to the configuration options so that you can view the information. However, you will not be able to change any of the data.

Control Configuration

(F1 from Configuration)



Pressing <F1> from the configuration screen will display the Control Configuration screen in the edit window. The Control Configuration screen provides you with a method of changing controller dependent data. Each of the fields is discussed in detail below.

If you wish to change a field, use the up and down arrow keys to move the cursor to the desired field. Type the new value and press <ENTER>, or press <SPACE> to toggle. When you are done editing, press <F10> to save any changes you have made. If you wish to discard your changes and restore the previous values, press <ESC>.

DRO Display Units

This field controls the units of measure the DRO displays. The two options are 'Millimeters' and 'Inches.' When this field is highlighted by the cursor, "Press SPACE to change" appears at the bottom of the screen. This message is explaining that pressing the <SPACE> key will toggle the value of this field between the two options.

The DRO display units do not have to be the same as the machine units of measure (explained below). This field is provided for users of the G20 & G21 codes so that they may view the tool position in terms of job units.

Machine Units of Measure

This field controls which units of measure the machine uses for each job. The two options are 'Millimeters' and 'Inches'. Press <SPACE> to toggle the field between the two options.

This field determines the default interpretation of job dimensions and feedrates. If 'Inches' is selected, all feedrates and dimensions will be interpreted as inches as well as any unit dependent parameters.

* NOTE: This field should rarely, if ever, be changed. If you wish to run a job in units other than the default machine units, use the G20 & G21 codes.

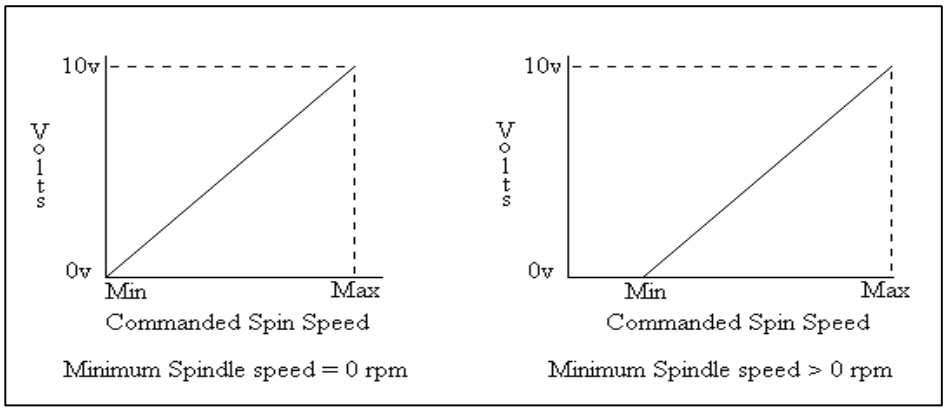
Maximum Spindle Speed (High Range)

This field sets the high range maximum spindle speed for those machines that have a variable frequency spindle drive controller (VFD). All spindle speeds entered in a CNC program are output to the PLC as percentages of this maximum value.

If your machine is equipped with a dual range drive and VFD, the controller will not exceed the spindle speed set by this field while in high gear. See the Machine Parameters section below for information on setting the gear ratios for medium and low gear ranges. If your machine has a VFD but is not equipped with a dual range drive, this field determines the maximum spindle speed.

Minimum Spindle Speed (High Range)

This parameter is used to adjust the minimum spindle speed for the high range. If minimum spindle speed in set to a value greater than zero, no spindle voltage will be output until the commanded spindle speed is greater than the minimum spindle speed. The values stored can range from 0 to 500000.0 RPM.



Machine Home At Powerup

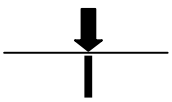
This field controls how the machine will home at powerup. There are 3 types of machine homing.

Home Switch

This option is to be used when home switches and/or limit switches will set the machine home position on ALL axes. See the Machine Configuration section below for detailed instructions on how to setup the switch to be used on each axis.

Ref Mark-HS

This option is to be selected when a physical switch will not set the home position of ALL axes. On each axis that does not have a physical switch (which can be all, none, or any combination of axes), some type of permanent marks (see below) must be used by the operator to line up that axis. On a knee mill, for example, where you do not want the home position at the edge of the travel (so the table is not hanging far off the machine), reference marks can be used to center the table to a repeatable position while a physical switch can still be used to set the vertical axis. See the Machine Configuration section below for detailed instructions on how to setup the switch or ref. mark to be used on each axis.



Jog

This option is to be used ONLY if the following conditions are met:

- There are NO physical switches of any kind on the machine
- There is NO need for the operator to consistently have the same machine home position

Note: If this option is chosen, the operator will be required to reset each Part Zero upon powerup of the machine. Also, when using the **Jog** option, the software limits must be closely monitored to avoid damage to the machine or operator.

For Home Switch and Ref Mark homing, the control will run the file "CNC10.HOM" to set home. This file maybe customized for special homing needs. This file is located in the "CNC10" directory on the control, and is created the first time the control is powered up after the "Machine Home at Powerup" is set to either of the two homing methods.

Default contents of the three axes "CNC10.HOM" file for a machine with home switches:

M92/Z : Move Z axis to the plus home switch
M26/Z : Set Z home
M91/X : Move X axis to the minus home switch
M26/X : Set X home
M92/Y : Move Y axis to the + home switch
M26/Y : Set Y home

If a 4th axis, such as a rotary table, is added, the CNC10.HOM file must be altered in order to home that axis as well. Assuming the 4th axis will be labeled "W", simply insert the following text at the end of the file and save your changes.

M92/W : Move W axis to the plus home switch
M26/W : Set W home

If you do not have a home switch on the 4th axis, just add the M26/W line.

PLC Type

This field tells the controller which PLC type is installed. The possible values are 'Absent,' 'Lite,' 'Normal,' and 'Dual.'

PLC Type setting	**PLC Hardware Installed**
Normal	PLC 3/3, PLC 15/15, RTK 2, PLCIO2, Servo3 I/O, RTK3, DC3IO
Lite	RTK1, PLC Lite
Dual	Koyo, Koyo + PLC 3/3, Koyo + PLC 15/15

This value should not be changed unless a different PLC type is installed.

Console Type

Determines what type of jog panel is installed.

Current Types	Notes
Uniconsole-2 ****Current Shipping Type****	Integrated console control panel, Uniconsole's and Uniconsole style pendent control panel
T-39	Lathe pendent control panel – Old T shaped pendent
M-39	Mill pendent control panel – Old T shaped pendent
Offline	Set for offline conversational and demo software
Keyboard	Setting for no jog panel, <ALT J> displays keyboard jog panel overlay
Previous Types	
M-10 M-60	Models M10 – M60 and T40 were originally CRT based consoles 9.4 and 10.4 in the type specify the size of the LCD used on that model An "M" in the type specifies a mill control panel, "T" specifies lathe.
T-40	
M-15 9.4, M-400 9.4	
M-15 10.4, M-400 10.4	
T-400 9.4, T-400 10.4	
Uniconsole	

Jog Panel Required

This field tells the controller whether a Jog Panel must be installed in order to run jobs. If set to Yes, Cycle Start must be pressed twice to start a job, if set to No, Cycle Start only needs to be pressed once to start a job.

Screen Blank Delay

This field determines the delay used for the screen blanker function. When a value other than zero is set, the screen will blank after the specified number of minutes. The blanking function only works if no jobs are running.
The value you enter is measured in minutes. Therefore, a value of 5 would blank the screen in 5 minutes if no actions were taken. When the screen is blank, pressing any key will restore the screen.
If you do not wish to use this feature, enter a value of zero to disable it. However, if the display is kept on for long periods of time without the blanker enabled, the image of a screen may become 'burned' into the monitor. That is, you will be able to see this image of the screen on the monitor whether the monitor is on, off, or in some other screen.

Remote Drive & Directory

This field sets up the remapped default drive and directory for the <F3> key in the Load Job screen. This allows you to conveniently load files from an attached computer via LAN network (via RJ-45 Ethernet connection). The Control will usually remap the attached computer's C hard drive as drive E, depending on the way it was set up

User Specified Paths

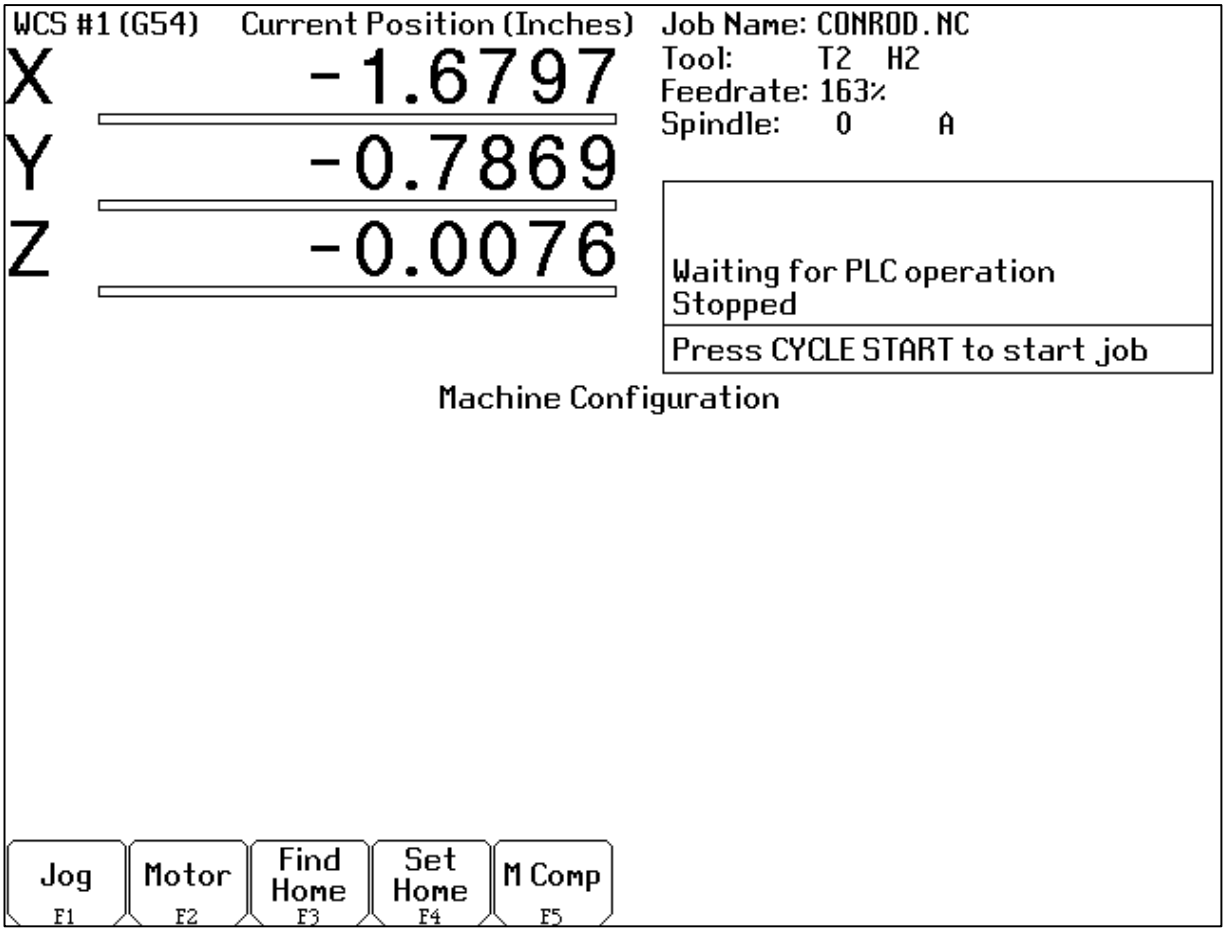
Operators can now specify paths for INTERCON files, posted INTERCON files, Digitize files and CAD files. These paths are specified in PATHM.INI. This file is automatically generated by CNC10 if it does not exist. This is a text file, found in the CNC10 directory, that can be edited with any available text editor. The default PATHM.INI file is:

```
INTERCON_PATH=C:\INTERCON\
ICN_POST_PATH=C:\CNC10\NCFILES\
DIGITIZE_PATH=C:\CNC10\NCFILES\
CAD_PATH=C:\CNC10\NCFILES\
```

Path tag	Purpose of path
INTERCON_PATH	Main directory containing *.ICN files
ICN_POST_PATH	Directory INTERCON places *.CNC files created when posting *.ICN files.
DIGITIZE_PATH	Directory digitize files are saved to. Directory used by F4 key in Load Job menu when parameter 4 is set to 2.
CAD_PATH	Directory for CAD files generated with the DIG->CAD option in the Utility menu.

Machine Configuration

(F2 from Configuration)



Pressing <F2> from the configuration screen will display the machine configuration screen in the edit window (see figure below). The machine configuration screen provides you with a method of changing machine dependent data.

F1 - Jog Parameters (Values should be recorded on the Control Parameters page at beginning of manual.)

Jog Parameters

Axis	Slow Jog (in/min)	Fast Jog (in/min)	Max Rate (in/min)	Deadstart (in/min)	Delta Vmax (in/min)	Travel (-) (Inches)	Travel (+) (Inches)
1	60	200	360	5.0000	5.0000	0.0000	0.0000
2	24	200	380	5.0000	5.0000	0.0000	0.0000
3	24	100	290	5.0000	5.0000	0.0000	0.0000
4	200	330	385	5.0000	5.0000	0.0000	0.0000
5	0	0	0	0.0000	0.0000	0.0000	0.0000

*** NOTE**: Some of these values are set automatically by the Autotune option (See PID Configuration below).

Slow Jog
Determines the speed of motion on an axis when slow jog is selected and a jog button is pressed. The slow jog rate cannot be set to a value greater than the maximum rate.

Fast Jog
Determines the speed of motion on an axis when fast jog is selected and a jog button is pressed. The fast jog rate cannot be set to a value greater than the maximum rate.

Max Rate
Determines the maximum feedrate of each individual axis. The feedrate on each axis can never exceed Max Rate, even if the feedrate override knob on the front panel is turned up above 100%. (See also the Machine Parameters section for the "Multi-Axis Max Feedrate" parameter that limits the feedrate along move vectors, not just each individual axis.). For information about setting this parameter automatically, see PID Configuration section below.
*** NOTE**: The maximum rate may be set to a smaller value if you wish to run your machine at a slower rate.

Deadstart
Determines the speed to which an axis decelerates before stopping or reversing direction. A low setting will cause a large slowdown before reversals of direction, causing your machine to be more accurate. A high setting will cause less slowdown before reversals, but this may cause your machine to "bang" and you may lose accuracy.

Delta Vmax
The maximum instantaneous velocity change that will be commanded on a vector transition. This parameter should not be changed.

Software Travel Limits
The following two parameters will create software limits for the machine. This is particularly useful if there are no limit switches installed on the machine. The software limits are what the control uses to limit the physical size of a machined part. If the software limits are set beyond the physical limit switch or hard stop, then, the control will attempt to run the job until it hits a physical stop resulting in an improperly machined part.

Travel (-)
This is the maximum distance the axis can travel in the minus direction from the home position. A value of 0 entered in this field will disable the software limit in this direction. This will not have any effect on the operation of the machine.

Travel (+)
This is the maximum distance the axis can travel in the plus direction from the home position. A value of 0 entered in this field will disable the software limit in this direction. This will not have any effect on the operation of the machine.

F2 - Motor Parameters (Values should be recorded on the Control Parameters page at beginning of manual.)

Motor Parameters

Axis	Label	Motor revs/in	Encoder counts/rev	Lash Comp. (Inches)	Limit -	Limit +	Home -	Home +	Dir Rev	Screw Comp
1	X	5.00000	8000	0.00000	2	1	0	0	N	N
2	Y	5.00000	8000	0.00000	4	3	0	0	N	N
3	Z	5.00000	8000	0.00000	6	5	0	0	N	N
4	N	5.00000	8000	0.00000	0	0	0	0	N	N
5	N	5.00000	8192	0.00000	0	0	0	0	N	N

- **WARNING**: The Motor Parameters **should not** be changed without contacting your dealer. Corrupt or incorrect values could cause damage to the machine, personal injury, or both.

Special function indicators:
These appear, if present, between the axis number and the label. 's' – axis is the spindle, 'p$' – axis is paired with axis '$', 'h$' – axis is a handwheel paired with axis '$', '*' – pairing conflict. See Machine Parameters for more information on setting up special functions.

Label

The letter you want to use to identify the axis. The first three axes should normally be X, Y, and Z. If a fourth axis is installed, it is usually named W or B. If you change a label, for example from X to A, the controller will then accept G-codes for axis A instead of X.

If fewer than four axes are present, the unused entries should be labeled N. If an axis is manually operated (it has an encoder but no motor), it should be labeled M.
For a manual Z axis, the 3rd axis label should be set to @. This setting allows for two axes posting in Intercon.
* WARNING: Intercon does NOT post two axis programs if the 3rd axis is labeled M.

* NOTE: Tool length compensation (G43-G44) and canned drilling cycles (G73-G89) always affect the third axis, regardless of its axis label. Tool diameter compensation (G41-G42) always affects the first and second axes, regardless of their axis labels.

Motor revs/unit

This field is the number of revolutions of the motor that results in one unit of measurement of movement. That is, if the machine units of measurement are inches, then Motor revs/inch is the number of revolutions of the motor that results in one inch of axis movement. If the machine units of measurement are in millimeters, then Motor revs/mm is the number of revolutions of the motor that result in one millimeter of axis movement Two methods of calculating the motor revs/unit are described below.

Current Ratio	New Ratio - after
Motor Rev/In	calc. or measure
X	X
Y	Y
Z	Z
W	W

How To - Measuring, and setting Motor Revs/Unit:

Method 1 - Calculated, assuming ball screw pitch and pulley ratios are known:

Using the following formula, calculate and enter the correct motor revs/inch value in the machine parameter screen as shown above.

FORMULAS:

tpi =	1 / pitch
ratio =	# Teeth on ball screw pulley / # Teeth motor
revs/inch =	tpi * ratio

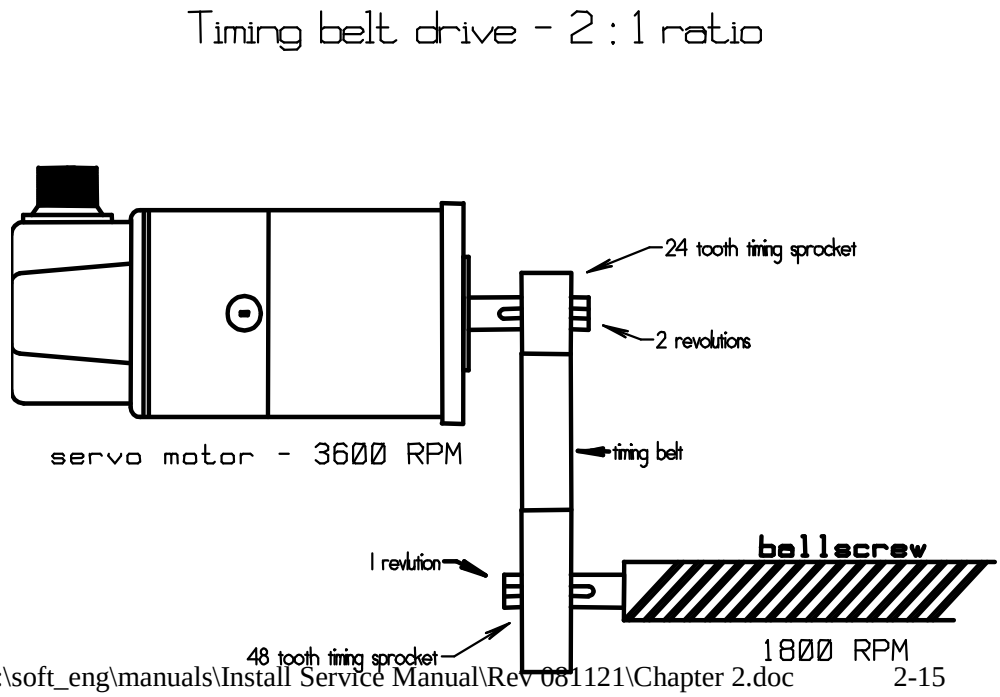
KEY:

tpi =	Turns per inch of the ball screw (given)
pitch =	Pitch of the ball screw (given)
ratio =	The gear ratio of the pulleys (calculated)
revs/inch =	Motor revs/inch (calculated)

Example: See Diagram 2 on next page.

tpi = 5
ratio = 48 / 24 = 2
Motor revs/inch = **tpi** * **ratio**
Motor revs/inch = 5 * 2
Motor revs/inch = 10

Note that this method assumes that the pitch and gear ratios are known. It is not uncommon that slight corrections based upon a measured test produce more accurate positioning throughout the entire axis travel. However, entering a Motor revs/in value based upon stated specifications should eliminate any gross errors.



Method 2 - Measured, assuming you have an accurate measuring method:

(NOTE: This method may also be used if the pulley ratio is unknown.

1. Secure a 10" gauge block to the table.
2. Indicate one end of this block and set a part zero. Use probing cycles if you have this option.
3. Find (or probe) the opposite end of the block and record the distance the DRO displays.
4. Using the formula below, calculate the new motor revs/inch value:

$$\frac{\text{DRO (displayed distance)}}{\text{Actual distance (measured)}} \times \text{Current ratio setting} = \text{New ratio setting}$$

Example:

$$\frac{10.0000 \text{ in.}}{9.9925 \text{ in.}} \times 5.0000 \text{ tpi} = 5.00375$$

Finish

1. From the Machine Configuration, press <F2>Motor and use the arrow keys to change the values Motor revs/inch. Press <F10> Save to accept the changes.
2. Repeat the measurements again to verify that the new settings are correct.

Encoder counts/rev

The counts per revolution of the encoders on your servomotors.

Lash compensation

The amount of backlash in the axis. This occurs when the table loses distance due to loose parts during direction reversals. For additional information, see Tech Bulletin TB037.

How To - Measuring, and setting Backlash:

The example below will be for the X-axis

1. Extend the Quill a few inches.
2. Set up the dial indicator parallel to the X-axis and perpendicular to the Z-axis such that the plunger will come in contact with the quill. Be sure the quill is free from chips and grease. See diagram 1.
3. Slow jog the X-axis until the quill just touches the plunger on the dial indicator and then stop.
4. Switch to Incremental Jog mode, and set the increment to .01" (press the <x 100> key)
5. Continue jogging the X-axis, moving the plunger incrementally toward the quill three (3) times. Observe and record the dial indicator ***start*** position. DO NOT TOUCH THE DIAL INDICATOR or the stand.
6. Continue to jog the X-axis moving the plunger incrementally toward the quill ten (10) times.
7. Jog the X-axis IN THE OPPOSITE DIRECTION ten (10) times. Observe and record the dial indicator ***finish*** reading. With no mechanical backlash, the ***start*** and ***finish*** values will be the same position. Subtract ***Finish - Start = Lash.***
8. Enter the Lash comp. value for the X-axis and save your changes. Repeat this procedure for all axes.

Notes:

- Measure the backlash several times until you achieve the same results consistantly.

- Measure backlash where the machine is to be operating the most (usually the center of travels). If the jobs that are being run use most of the travel envelope, measurements can be averaged.

- If you measure more than .0015" of lash, the machine should be mechanically adjusted or repaired. Although the control will attempt to compensate for excessive lash (>.0015"), it cannot completely eliminate the underlying mechanical problems that result in unsatisfactory performance.

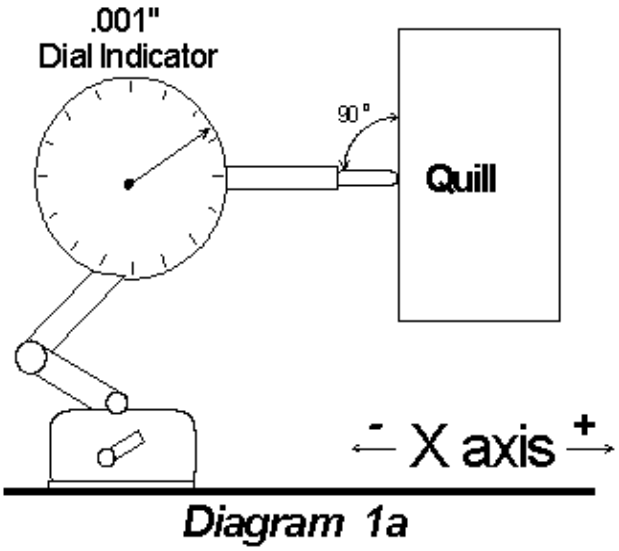


Diagram 1a

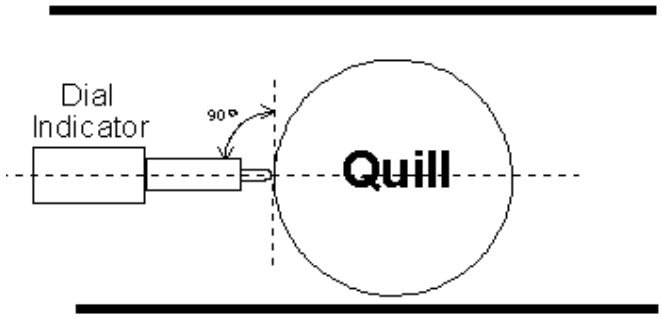


Diagram 1b. Top View
Keep the dial indicator square to the Quill

Note: The following two fields not only tell the control which PLC inputs the switches are wired to, but also affect the way the machine homes upon powerup.

Limit

These fields represent the PLC input numbers corresponding to any physical limit switches that you may have on your machine. If no limit switches are installed, these fields should be set to 0.

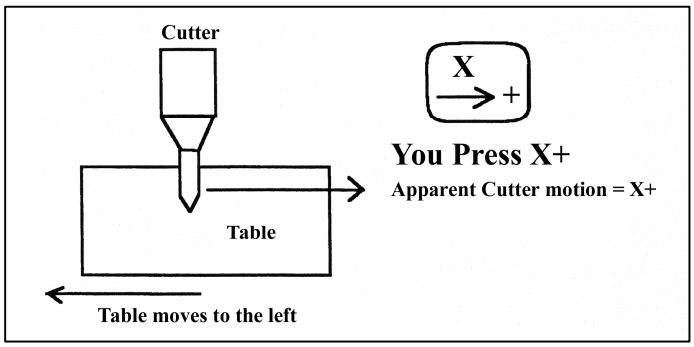
Home

These fields represent the PLC input numbers of any Home Switches you may have installed on your machine. Home switches are used when you want the machine to automatically home to a position other than the travel limits. If your machine does not have home switches, however, and you wish to use the limit switches to home the machine, these fields should be set to the Limit Switch values. If no home or limit switches are installed, these fields should be set to 0.

*** NOTE**: For Reference Mark homing, enter a zero for the axes that have a reference mark, and a nonzero value for the axes that will home to a switch.
*** NOTE**: If Home Switch has been chosen as the homing method, these fields MUST have a nonzero value in order to home properly.
*** NOTE**: The Home Switch should never be physically located beyond the Limit Switch.

Direction reversed

Used to match the +/- reference of your machine to the control electronics. Toggle this value if you actually move in the X direction (reverse) when you jog X+.



Screw Compensation

This value indicates whether ball screw mapping compensation is enabled. When enabled a preset ball screw map compensates for error along the entire ball screw.

F3 - Find Home

Press <F3> to move an axis to its plus or minus home switch. This function will not work if the machine homing is set to Jog in the Control Configuration screen.

F4 - Set Home

Press <F4> to set Machine Home for an axis at its current position. This is usually performed after Find Home. This operation should not be used to set the part zero position. To set the part zero position, use the Part Setup screen.

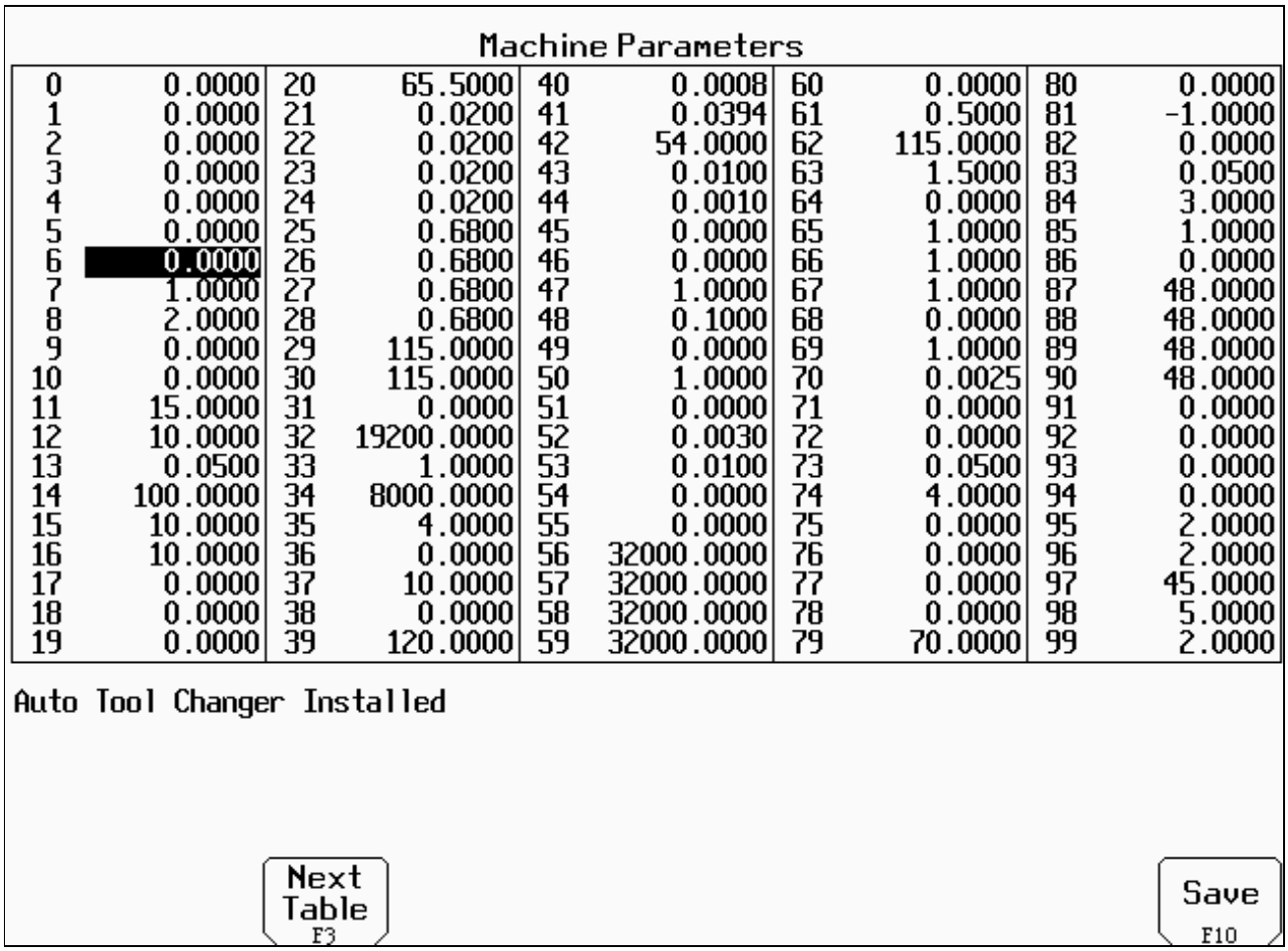
F5 – Manual Ball screw Compensation

This option lets you edit the ball screw compensation tables.

* **WARNING**: The ball screw compensation tables **should not** be changed without contacting your dealer. Corrupt or incorrect values could adversely affect the accuracy of the positioning of your machine.

Machine Parameters

(F3 from Configuration)



Pressing <F3> from the configuration screen will display the machine parameters screen. This screen provides you with a method of changing various parameters that are used by the control.

If you wish to change a field, use the arrow keys to move the cursor and select the desired field. Type the new value and press <ENTER>. When you are done editing the fields, press <F10> to accept any changes you have made and save them. Press <ESC> to return to the previous screen (Setup). A short description of the parameter will appear below the table. In the screen example above, parameter 6 determines whether an Auto Tool Changer is installed.

<F3> Next Table will toggle the display parameters between parameters 0-99 and parameters 100-199.

Bit-mapped parameters

Certain control parameters are defined by bit-mapped values. In order to change these parameters you must understand how bit mapping works. A bit-mapped parameter is stored as a number, representing a 16 bit value in the control. If a certain bit needs to be turned on, that bit's binary value must be added to the parameter value, if the bit needs turned off, its binary value must be subtracted from the parameter value. The values for each of the 16 bit's can be seen in the table below.

Bit-Mapped Parameter Bit's																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Value	32768	16384	8192	4096	2048	1024	512	256	128	64	32	16	8	4	2	1

To set bit-mapped parameters simply add together the bit values that you need to have enabled.

Examples:

Parameter value	**Bit number and settings**															
	15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
1	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	ON
11 < 8+2+1	X	X	X	X	X	X	X	X	X	X	X	X	ON	X	ON	ON
24 < 16+8	X	X	X	X	X	X	X	X	X	X	X	ON	ON	X	X	X

The following parameters are currently defined:

Parameter	Definition	Default setting
0	E-Stop PLC Bit	0
1	Y jog key orientation	0
2	G-Code Interpretation Control	0
3	Modal Tool and Height Offset Control	0
4	Remote File Loading Flag	0
5	Suppress Machine Home Setup	0
6	Auto Tool Changer Installed	0
7	Display Colors	0
8	Available Coolant System(s)	2
9	Display Language	0
10	Macro M-Function Control	0
11	Touch Probe PLC Input	0
12	Touch Probe Tool Number	0
13	Probing Recovery Distance	0.05
14	Fast Probing Rate	n/a
15	Slow Probing Rate	n/a
16	Probing Search Distance	10
17	Tool Detector Reference Number	0
18	PLC Input Spindle Inhibitor	0
19	MPG modes	0
20	Ambient Temperature	72
21-24	Motor Heating Coefficients	Refer to text
25-28	Motor Cooling Coefficients	Refer to text
29	Warning Temperature	150
30	Limit Temperature	180

31	Spindle Speed Output Port	0
32	Spindle Serial Port Baud Rate	19,200
33	Spindle Motor Gear Ratio	1
34	Spindle Encoder Counts/Rev	8,000
35	Spindle Encoder Input	4
36	Rigid Tapping Enable/Disable	0
37	Spindle Deceleration Time	10
38	Multi-Axis Max Feedrate	0
39	Feedrate Override Knob Limit	120
40	Basic Jog Increment	0.0001”
41	Handwheel 100x Speed, User Jog Increment	0.25”
42	Password for Configuration Menus	0
43	Automatic tool measurement options	0
44	TT1 PLC input #	0
60	Digital Filter Size	1
61	High Power Stall Timeout	0.5
62	High Power Stall PID Limit	115
63	High Power Idle PID Multiplier	1.5
64	Fourth Axis Pairing	0
65-67	Spindle Gear Ratios	1
68	Minimum Rigid Tapping Spindle Speed	100
69	Duration For Minimum Spindle Speed	1.25
70	Offset Library Inc/Decrement Amount	.001"/.02mm
71	Part Setup Detector Height	0
72	Data M-Function Options	0
73	Peck Drill Retract Amount	0.05
74	M-Function executed at bottom of tapping cycle	4
75	Axis Summing Display Control	0
76	Manual Input Unrestricted Distance	0
77	Manual Input Movement Tolerance	0
78	Display of Spindle Speed	0
79	Auto Brake Mode PLC Bit for Uniconsole-2	70
80	Voltage Brake Applied Message Frequency	1
81	Air Drill M-Function (executed instead of Z movement in drilling)	-1
82	Spindle Drift Adjustment	108
83	Deep Hole Clearance Amount	0.05
84	M-Function executed at return to initial point of tapping cycle	3
87-90	Autotune Accel Time and ka.	48
91-94	Axis Properties	0
95-98	Autotune Move Distance	2
99	Cutter Compensation Look-ahead	2
100	Intercon comment generation	0
101	Intercon clearance amount	0.1
102	Intercon spindle coolant delay	3.0
103	Intercon corner federate override	50.0
104	Intercon modal line parameters	0
105	Intercon modal arc parameters	0
106	Intercon modal drilling cycle parameters	0
115	Intercon Help	0
120	Probe stuck clearance amount	0.10”
121	Grid digitize prediction minimum Z pullback	0.002”
122	Grid digitizing deadband move distance	0.0002”

123	Radial digitizing clearance move	0
124-127	PLC inputs for jogging	0
128	Handwheel MPG mapping	0
129	Handwheel MPG display control	0
130	Z axis on/off selection	0
131	4th axis on/off selection	0
132	5th axis heating coefficient	Refer to text
136	5th axis cooling coefficient	Refer to text
140	Message log priority level	1
141	Maximum message log lines	1000
142	Message log trim amount	1000
143	DRO properties (load meters, 4/5 digits, DTG)	0
144	Comparison rounding	0
145	Advanced macro properties (fast branching)	0
146	Feed hold threshold for feed rate override	0
150	Run-Time Graphics	0
152	5th axis Autotune accel time and Ka	48
156	5th axis Autotune move distance	2
160	Enhanced ATC	0
161	ATC Maximum Tool Bins	0
166	5th axis properties	0
170-179	XPLC parameters	0
180	File Transfer COM Port	0
181	File Transfer Baud Rate	19.2
182	File Transfer Data, Parity and Stop bit settings	801
183	File Transfer Flow Control Setting	0
184	File Transfer COM timeout	10
185	File Transfer Serial Port Option	0
188-199	Aux key functions	0

Parameter 0 – E-Stop PLC Bit

This parameter specifies the PLC bit to which the physical Emergency Stop switch is connected. It is mainly used for ATC applications that use custom plc messages.

Examples:

PLC Type	ESTOP Input on PLC	Parameter Value
RTK3	Input 11	11
PLCIO2	Input 11	11
DC3IO	Input 11	11
Servo3IO	Input 1	1

Parameter 1 – Y jog key orientation

This parameter is a 3 bit field where bit 0 is not used in the mill software. Bit 1 sets the direction of movement that the Y+ and Y- jog keys would cause and bit 2 will swap the X and Y jog keys. This should always be set to 0 except for very special applications.

Bit	Function Description	Parameter Value
0	Not Used	
1	Flip movement direction of Y jog keys?	Yes=2, No=0
2	Exchange X axis and Y axis jog keys?	Yes=4, No=0

Parameter 2 - G-code Interpretation Control

This parameter is a 3-bit field that controls optional interpretation of several G-codes. The following table shows the functions performed by the value entered in this parameter:

Bit	Function Description	Parameter Value
0	Arc centers I, J, K are absolute in G90 mode?	Yes = 1 No = 0
1	Allow Z being specified alone to be sufficient to trigger execution of a canned tapping or drilling cycle to be executed?	Yes = 2 No = 0
2	Interpret dwell times associated with G4, G74, G82, G84, and G89 as milliseconds rather than seconds?	Yes = 4 No = 0
3	Slaving rotary axis feedrate to non-rotary axis feedrate	Yes = 8 No = 0
4	Selects the center for scale, mirror and rotate. By default the center will be 0,0,0. Add 16 to this parameter to make the center of scale, mirror and rotate the current position.	Yes = 16 No = 0

Parameter 3 - Modal Tool and Height Offset Control

This parameter is a 2-bit filed. Bit 1 controls whether or not the last tool and height offset activated during a job run will remain active after the job is complete as well as controling the Tool status display in the Status Window.

Bit	Meaning	Parameter Value
0	Tool and Height Offset numbers will be modal and remain active between jobs. Tool Status display will remain active even when job is not running. Tool Status display will show the current T and H number.	Yes = 0 No = 1
1	Reference tool position set to Z home	Yes = 1 No = 0

Parameter 4 - Remote File Loading Flag

This parameter controls the action of the Load Job menu when CNC job files are selected from drives letters higher than C. These drives (i.e. drives D, E, F, etc.) are presumed to be network, Interlink drives or extra hard drives.

Value	**Meaning**
0	Job files are not copied or cached. They are run from whichever drives they reside on.
1	Job files are copied to the C drive (c:\cnc10\ncfiles) when they are loaded. The local copy is used when the job runs.
2	Turn on file caching. Job files are temporarily cached on the C drive. The cached copy is used while the job is running. The cached copy is deleted when the next job is loaded or when Parameter 4 changes to a 0 or 1. Digitize files are cached as the machine is digitizing. When digitizing is complete, the resulting file is copied to the digitize directory specified in *pathm.ini*.

File caching is useful for machines with both a flash card and a hard drive. By caching job files from the hard drive on the flash card, the hard drive is not used while the job is running. As a result, the life of the hard drive is extended and the flash card does not fill up with job files.

Parameter 5 - Suppress Machine Home Setup

This parameter controls machine homing upon startup of the control. The following table details the functions controlled by this parameter:

Bit	Function Description	Parameter Value
0	Suppress the requirement to set machine home before running jobs?	Yes = 1 No = 0
1	Display router bit map at homing screen	Yes = 2, No = 0
2	Disable stall detection when CNC10 first starts.	Yes = 4, No = 0

Bit 0 suppresses the requirement to set machine home before running. If bit 0 of Parameter 5 is 0, machine home must be set before jobs may be run. If bit 0 of Parameter 5 is 1, machine home is not requested or required, but Graphing and running of jobs will not work until the Z axis is homed.

NOTE: Parameter 5 Bit 0 is separate from the "Machine Home at Powerup" flag in the Control Configuration Screen. Parameter 5 Bit 0 determines **whether** you must home the machine; the "Machine Home at Powerup" flag determines **how** you will home the machine, if you must do so.

Parameter 6 - Automatic Tool Changer Installed

This parameter tells the control whether you have an automatic tool changer installed on your machine. This field affects the action of M6 in your CNC programs. See M6 under M functions in Chapter 13. It also affects whether the ATC key is present in the Tool Offset Setup and whether to save the last tool change number in the job files.

Value	Meaning
0	Auto Tool Changer NOT Installed
1	Auto Tool Changer Installed

Parameter 7 - Display Colors

This parameter determines what combination of colors will be used for display. If you have a color display, set this parameter to 0. If you have a monochrome display (especially a monochrome LCD panel) set this parameter to 1.

Parameter 8 - Available Coolant Systems

This parameter is used by Intercon to determine what coolant systems are available on the machine. It should be set as follows:

Value	Meaning
1	Mist Coolant (M7) only
2	Both coolant systems
3	Flood Coolant (M8) only

Parameter 9 - Display Language

This parameter determines what language will be used for menus, prompts and error messages.

Value	Meaning
0	English
1	Spanish
2	French
3	Traditional Chinese
4	Simplified Chinese

Parameter 10 - Macro M function handling

This parameter is a 4-bit field that controls various aspects of M functions. The following table shows the functions performed by the value entered in this parameter. The default value is 0.

Bit	Function Description	Parameter Value
0	Display M & G codes in M function macros?	Yes = 1, No = 0
1	Step through M function macros in Block Mode?	Yes = 2, No = 0
2	Brushless motor option: Decelerate smoothly to stop (pause) on M105 and M106? (Digitizing and Probing moves.) "Yes" means decelerate smoothly. Choosing "yes" takes more time on each probing move and is slightly less accurate. "No" means hard stop. "No" is faster and slightly more accurate but can cause excessive vibration on brushless systems.	Yes = 4, No = 0
3	Move to Z home on M6?	No = 8, Yes = 0

Parameters 11-17 - Touch Probe Parameters

These parameters control touch probe and tool detector operation. Refer to the DP-4 Touch Probe manual.

Parameter 18 - PLC Input Inhibit Parameter (M39 and M400 only)

This parameter stores the input for the control's PLC I/O unit for the Spindle Inhibit feature. A positive value must be entered if a "normally closed" probe is to be used with the control. A negative value must be entered if a "normally open" probe is to be used with the control. The absolute value of Parameter 18 will directly reflect the PLC input the Spindle Inhibit is wired to. When this parameter is set, Digitizing and Probing cycles will not run unless a probe or touch-off block is connected. This parameter is used to prevent the tool or probe from crashing into the table. The default for this parameter is 0, which disables this feature.

Parameter 19 - MPG modes

The MPG is a hand-held device that is used as an alternate way of jogging the machine. This parameter defines the MPG's mode of operation.

Bit	Function Description	Parameter Value
0	Enable MPG when powering up control?	Yes = 1, No = 0
1	MPG speed limit	x100 = 2, x10 = 0

Parameters 20-30 - Motor Temperature Estimation

These parameters are used for motor temperature estimation. Parameters 20, 29 and 30 correspond respectively to the ambient temperature of the shop, the overheat warning temperature, and the job cancellation temperature, all in degrees Fahrenheit. Parameters 21 through 24 are the heating coefficients for each of the four axes. Parameters 25 through 28 are the cooling coefficients for each of the four axes.

DC Brush Motors and Drives						
Parameter	Axis	Values	Values	Values	Values	Values
Servo Drive		8A Drive, 15 in/lb motors	12A Drive, 29 in/lb motors	15A Drive, 29 in/lb motors	15A Drive, 40 in/lb motors	25A Drive, 40 in/lb motors
20	N/A	72	72	72	72	72
21	X	0.028	0.02	0.027	0.03	0.04
22	Y	0.028	0.02	0.027	0.03	0.04
23	Z	0.028	0.02	0.027	0.03	0.04
24	4TH	0.028	0.02	0.027	0.03	0.04
25	X	0.68	0.68	0.68	0.68	0.68
26	Y	0.68	0.68	0.68	0.68	0.68
27	Z	0.68	0.68	0.68	0.68	0.68
28	4TH	0.68	0.68	0.68	0.68	0.68
29	ALL	150	150	150	150	150
30	ALL	180	180	180	180	180
AC Brushless Motors and Drives						
Parameter	Axis	Values	Values	Values		
SD Drive		SD3, SD1 750 W motors	SD3, SD1 1,2 KW Motors	SD1 3, 4 KW motors		
20	N/A	72	72	72		
21	X	0.05	0.04	0.04		
22	Y	0.05	0.04	0.04		
23	Z	0.05	0.04	0.04		
24	4TH	0.05	0.04	0.04		
25	X	0.68	0.68	0.68		
26	Y	0.68	0.68	0.68		
27	Z	0.68	0.68	0.68		
28	4TH	0.68	0.68	0.68		
29	ALL	150	150	150		
30	ALL	180	180	180		

Parameter 31 – Spindle Speed Output Port

Parameter 31 determines the destination for the raw spindle speeds generated and output by the Control. Below are the possible values for this parameter. Note that if your machine uses a serial type spindle controller, you should not set this parameter to 0.

Value	Meaning
-1	12 bits to PLCIO2, RTK3 and Koyo via the CPU10
0	8 bits to CPU10 controlled spindle and PLC 15/15
1	12 bits to COM1 serial port
2	12 bits to COM2 serial port

Parameter 32 - Spindle Vector Drive Serial Port Baud Rate

The baud rate (e.g. 9600, 19200, etc.) of the serial port at which the control should communicate with the SPIN232 board. This parameter has meaning only if Parameter 31 is set to 1 or 2, for COM1 or COM2 spindle speed output.

Parameter 33 - Spindle Motor Gear Ratio (Baldor Vector Drive Only)

The gear or belt ratio between the spindle motor and the chuck in high gear range. Should be greater than 1.0 if the motor turns faster than the chuck and less than 1.0 if the chuck turns faster than the motor. Note: this value applies to high range. The ratio between high range and lower ranges is established by the gear ratio parameters (65-67).

Parameter 34 - Spindle Encoder Counts/Rev

This parameter controls the counts/revolution for the spindle encoder. If the encoder counts up when running CW (M3), the value of this parameter must be positive. If the encoder counts up when running CCW (M4), the value of this parameter must be negative.

Parameter 35 – Spindle Encoder Input

This parameter specifies the axis input to which the spindle encoder is connected. Input from the spindle encoder is required for the spindle-slaved movements used in the Rigid Tapping cycles. So, if Rigid Tapping is used, this parameter must be set to the correct value. Otherwise, this value is generally ignored. A value of 2 means the third encoder input; a value of 3 means the fourth encoder input; a value of 4 means the fifth encoder input. A value of 5 is used for the 6th axis encoder input, this is used on SD3 based systems.

Spindle Encoder Plugged into?	DC System Value	AC System Value
CPU10 Encoder input 1	N/A	N/A
CPU10 Encoder input 2	1	17
CPU10 Encoder input 3	2	18
CPU10 Encoder input 4	3	19
CPU10 Encoder input 5	4	20
CPU10 Encoder input 6	N/A	21
SD3 spindle encoder input	N/A	5

Parameter 36 - Rigid Tapping Enable/Disable

This parameter is a 3-bit field that enables or disables Rigid Tapping and its options. Bit 1 and 2 have no meaning unless bit 0 is turned on.

Bit	Function Description	Parameter Value
0	Enable Rigid Tapping?	Yes = 1, No = 0
1	Suppress sending "Wait for Index Pulse" during Rigid Tapping?	Yes = 2, No = 0
2	Allow Spindle Override during Rigid Tapping?	Yes = 4, No = 0

Parameter 37 - Spindle Deceleration Time

This parameter is used in conjunction with parameter 36 when rigid tapping is enabled. This sets the amount of time required for the spindle to decelerate before it switches direction during a rigid tapping operation.

Parameter 38 - Multi-Axis Max Feedrate

This parameter is used to limit the feedrate along all commanded move vectors. This parameter can be used to limit the speed of multi-axis moves on machines that may have enough power to move a single axis rapidly, but starve out of power on 2 or 3 axis rapid moves. A zero in this parameter will disable this feature.

Parameter 39 - Feedrate Override Percentage Limit

This parameter is used for limiting the upper end of the Feedrate Override Knob percentage to a value from 100% to 200%. This parameter can be used to restrict the Feedrate Override Knob effect on machines with maximum rates over 200 in/min. The Feedrate Override Knob percentage is normally allowed to go to 200%. However, on machines with high cutting speeds, if the knob is turned up to 200%, it creates overshoots on corners. If this parameter is set to something like 110, it will stop the Feedrate Override Knob from exceeding 110% and thus causes the overshoots to disappear.

Parameter 40 - Basic Jog Increment

This parameter holds the basic jog increment (0.0001" or 0.002 mm by default). This value is used by the x1, x10, and x100 jog keys (0.0001, 0.001 and 0.01 on older consoles). It also specifies the distance per click for handwheels (MPG).

Parameter 41 - Handwheel 100x Speed, User Jog Increment

On newer consoles, this parameter holds the actual handwheel speed in 100x mode. For normal 100x operation it should be 100. On some systems 100x is way too fast and this value is set to a more reasonable value such as 20 or 30. On older consoles, this parameter holds the user jog increment (0.250" or 1.0 mm by default). The 0.250 jog key on older consoles uses this value.

Parameter 42 – Password for Configuration Menus

This parameter determines the password that the user must enter in order to gain supervisor access to the configuration menus.

Value	Meaning
54.0	No password required for supervisor access; the user is not prompted for a password
ABCD.ABCD	Password is 4 digits represented by “ABCD”
Any other number	Password is “137”

Parameter 43 – Automatic tool measurement options

This parameter is a 2-bit field that is used to configure properties of the TT1.

Bit	Function	Parameter Value
0	The height of the tool detector (parameter 71) will be subtracted from the measured height of the tool.	Yes = 1; No = 0
1	Which PLC input to use for the Tool Z reference measurement	Use Parameter 11 = 0 Use Parameter 44 = 2

Parameter 44 – TT1 PLC input number

PLC input number that the TT1 is wired into on the PLC. If a shared PLC input is used for the TT1 and the DP4 probe, then the value can be left at zero or set to the same value as parameter #11.

Warning: If using a different PLC input for the TT1 and DP4, when setting the Z reference in the tool library with the DP4 make sure not to use a ruby probe tip. The TT1 is continuity based and the ruby tip is not conductive!

Parameter 56 – Feedrate Override Display Properties

This parameter is a 3-bit field that is used to define how the federate override is displayed in the status window.

Bit	Function	Parameter Value
0	Not used	
1	Display programmed rate not actual	Yes = 2; No = 0
2	Display a bar meter of percentage	Yes = 4; No = 0

Parameter 60 - Digital Filter Size

This parameter defines the PID output filter size for the motor outputs. This parameter is meant to provide a software filter where no hardware filter exists in order to slow down the PID output frequency (normally 4000 times/sec.), or to supplement a hardware filter that appears to be inadequate. It is the number of samples to average the PID output over. For example, a value of 2 says to average the PID output over 2 samples, which would reduce the PID output frequency to 2000 (4000/2) times/sec. The default value of this parameter is 1 (no averaging).

Parameters 61-62 - Stall Detection Parameters

The M-Series control will detect and report several stall conditions. The low power stall occurs if the control has been applying a specified minimum current for a specified time, and no encoder motion has been detected. This may indicate a loose or severed encoder cable. A high power stall occurs if the control has been applying at least 90% current for a specified time, and no motion greater than 0.0005" has been detected. This may indicate a physical obstruction.

Parameter 61 is the time limit, in seconds, for a high power stall. The default is 0.5 seconds.
Parameter 62 is the PID output threshold for a high power stall. The default is 115.

Parameter 63 - High Power Idle PID Multiplier

This parameter holds the value of a constant used for motor temperature estimation when an axis is not moving and no job is running, but there is power going into the motor to maintain its position. The default value is 1.5. This temperature estimation is intended to detect early if an axis is stopped against some abnormal resistance, such that it will probably overheat later.

Parameter 64 – Fourth/Fifth Axis Pairing

This feature enables the 4^{th}, 5^{th} or 4^{th} and 5^{th} axes motors to be run in a paired fashion with any of the other axes. This is intended to drive 2 screws on opposite sides of a table (probably a router table or gantry system). Set this parameter to 0 (default) to indicate that no other axis is paired with the 4^{th} or 5^{th} axis. In order to pair both the 4^{th} and 5^{th} axes on the same system add the 4^{th} axis value with the 5^{th} axis value. Example: 4^{th} axis paired with the X axis and 5^{th} axis paired with the Z axis a value of 21 would be entered into parameter 64.

Value	Meaning
0	No Pairing (Default)
1	Pair 4^{th} axis with X Axis
2	Pair 4^{th} axis with Y Axis
3	Pair 4^{th} axis with Z Axis
17	Pair 5^{th} axis with X Axis
18	Pair 5^{th} axis with Y Axis
19	Pair 5^{th} axis with Z Axis
20	Pair 5^{th} axis with 4th Axis

Parameters 65-67 - Spindle Gear Ratios

These parameters tell the control the gear ratios for a multi-range spindle drive. Up to four speed ranges are supported; high range is the default. Parameters 65-67 specify the gear ratio for each lower range, relative to high range. For example, if the machine is a mill with a dual range spindle, and the spindle in low range turns 1/10 the speed it turns in high range, then parameter 65 should be set to 0.1.

Parameter 65 is the low range gear ratio.
Parameter 66 is the medium-low range gear ratio.
Parameter 67 is the medium-high range gear ratio.

These parameters work in conjunction with the PLC program, which uses the states of INP63 and INP64 to signal to the CNC10 software which range is in effect, according to the table below.

	Spindle Range			
PLC INPUT	High Range	Medium High Range	Medium Low Range	Low Range
INP63	0	1	1	0
INP64	0	0	1	1

Parameter 68 – Minimum Rigid Tapping Spindle Speed

This parameter holds the value that the spindle slows down to from the programmed spindle speed towards the end of the tapping cycle. The lower the value, the more accurately the Z axis will land on target, but at the expense of possibly stalling the spindle motor which in turn will cause Z to fall short. If this value is too large, the off –target error increases. The suggested starting value is 100 RPM.

Parameter 69 – Duration for Minimum Spindle Speed Mode

This is the duration of time, in seconds, that the control will stay at minimum spindle speed. If the number is too small, overshoot will occur. If the number is too large, the user waits longer for the hole to be tapped at the slow speed specified by parameter 68. The suggested starting value is 1.25 seconds.

Parameter 70 - Offset Library Inc/Decrement Amount

Sets the increment and decrement amount used in the offset library.

Parameter 71 – Part Setup Detector Height

If this Parameter is set to a non-zero value, it indicates that the F3/Auto feature in part setup should be available using the tool detector (TT1) instead of the probe. The value in this parameter is the height of the detector. A value of 0 disables this feature.

When this feature is enabled:
a) Probe detection (Parameter 18) is not checked
b) The tool number and/or edge finder diameter entered by the operator are used; Parameter 12 is ignored.
c) The value from Parameter 71 is added to (or subtracted from, depending on approach direction) the part position.

Parameter 72 – Data M Function Options

The setting of this parameter affects the operation of the data M functions M122 and M123.

Bit	Function Description	Parameter Value
0	Suppress output of axis labels by M122?	Yes = 1, No = 0
1	Insert commas between positions/values with M122 and M123?	Yes = 2, No = 0
2	Suppress spaces between positions/values outputted by M122 and M123?	Yes = 4, No = 0

Parameters 73, 74 - Canned Cycle Parameters

P74 specifies the number of the M-function that is executed at the bottom of the G74 or G84 tapping cycle.
P73 specifies the retract amount used during a G73 peck drilling cycle.

Parameter 75 – Summing Display Control

This parameter indicates which axes are to be summed and how the results are to be displayed on the DRO. The parameter can contain up to four digits. The position and value of each digit has special significance as indicated in the tables below:

Parameter Digit Position	Axis Display Controlled
1's Column	Axis 1
10's Column	Axis 2
100's Column	Axis 3
1,000's Column	Axis 4

Digit Value	Meaning
0	Summing off
1 – 4	Axis to Sum
5	(reserved)
6	Disable display
7	Display if moved
8	Display if other moves
9	(reserved)

Here are some examples using the axis summing display parameter:

Desired Display	Parameter
Sum Z axis (3) with M (4), display sum in Z DRO position	400
Sum Z axis (3) with M (4), display sum in M DRO position.	3000
Sum Z axis (3) with M (4), display sum in Z DRO position, and suppress M display.	6400
Sum Z axis (3) with M (4), display sum in M DRO position, and suppress Z display.	3600
Sum Z axis (3) with M (4), display sum in Z DRO position, and show M only if M moves.	6300
Sum Z axis (3) with M (4), display sum in Z DRO position, show M if either Z moves.	7400

The DRO will display both labels when displaying a summed axis. For example, "ZM" or "MZ" depending on where the sum is displayed.

Parameter 76 – Manual Input Unrestricted Distance

This parameter is intended to be used with Z axis summing. It defines the maximum distance from the summed axis start of travel in which manual movements can occur without causing a fault. Use a negative value to specify a distance from the minus travel limit, a positive value for a distance from the plus travel limit.

When used with manual drilling, for example, setting this parameter will allow the operator to keep a hand on the quill at all times and even begin pulling on the quill in anticipation of a programmed stop.

Setting this value to zero will cause a fault if there is any manual movement.

To completely disable manual movement restrictions, set this parameter to a value exceeding the total travel of the summed axis.

Minimum = -99999.9999, maximum = 9999.9999, default = 0, typical = +/- 1.0 inch or +/- 20.0 mm

Parameter 77 – Manual Input Movement Tolerance

This parameter specifies the manual movement tolerance while a job is running. It is intended for use with a quill locking mechanism. It allows the lock to distort and/or slip a small amount when under stress. If the quill moves more than the given tolerance, the job will stop with a fault. A typical setting for Parameter 77 is 0.005 inches.

Parameter 78 – Display of Spindle Speed

This parameter specifies how the spindle speed is determined and displayed in the CNC10 status window. When set to 1.0, the spindle speed is determined from reading the encoder feedback from the axis specified according to parameter 35, which has the number of encoder counts/revolution specified in parameter 34. When set to 0.0, the displayed speed is not measured- the speed is calculated based upon the set speed, spindle override adjustment, and gear range.

Parameter 79 – Auto Brake Mode PLC Bit for Uniconsole-2

This parameter specifies which PLC bit signals the state of automatic brake mode when using the Uniconsole-2 console type. For other console types, it has no effect. This parameter can be changed to allow the Auto Brake mode key to be located in different positions on the Uniconsole-2 jog panel. The PLC program must be updated to reflect any change in this parameter.

Parameter 80 – Voltage Brake Message Frequency

This parameter specifies the number of time the "450 Voltage brake applied message has to occur before we show it in the message window and message log. A value of 0 or 1 will display the message for every instance that it occurs.

Parameter 81 – Canned Cycle Parameter

P81 (when not equal to –1.0) specifies the M-function to be called in place of Z axis movement during a G81 drilling cycle.

Parameter 82 – Spindle Drift Adjustment

This value is the number of degrees that the spindle will take to coast to a stop if it is cut off while it is spinning at the spindle speed specified by parameter 68.

Parameter 83 – Canned Cycle Parameter

P83 specifies the clearance amount used during a G82 deep hole drilling cycle.

Parameter 84 – Canned Cycle Parameter

P84 specifies the number of the M-function that is executed after the return to the initial point of a G74 or G84 tapping cycle.

Parameters 87-90 - Autotune Accel Time and Ka

These parameters are used by autotune. Increasing the value will lengthen acceleration time and reduce the ka value given by autotune. Lowering the value will decrease the acceleration time and increase Ka. First set the parameters and then run autotune. The default value is 48. The maximum value is 64 and the minimum value is 1.

Parameters 91-94 – Axis Properties

These parameters may be used to set various axis properties. These parameters correspond to X, Y, Z and the fourth axis, respectively.

Bit	Function Description	Parameter Value
0	Rotary/Linear Axis Selection	Rotary Axis= 1, Linear Axis= 0
1	Rotary Display Mode	Wrap Around = 2, Show Rotations = 0
2	Suppress direction check when doing Tool Check?	Don't Check = 4, Check= 0
3	Suppress park function?	Don't Park = 8, Park = 0
4	NOT USED ON MILL	
5	Linear Display of Rotary Axis	Linear Display = 32, Default Rotary = 0
6	4th Axis works like Z axis	Yes = 64 No = 0
7	NOT USED ON MILL	
8	Tilt rotary axis	Yes = 256; No = 0

Notes on Bit 0: Turning this bit on will cause the DRO display for the affected axis to be displayed in degrees. Also this information is used by Intercon to make rotary axis support available (by setting parameter 94 to 1, indicating that the fourth axis is rotary). This bit is also used when performing inch/mm conversions: values for a rotary axis will not be converted since they are assumed to be in degrees regardless of the system of linear units.
Notes on Bit 1: This bit has no effect unless Bit 0 (mentioned above) is turned on. When this bit is turned on, a "Wrap Around" display is shown on the DRO. A "Wrap Around" Rotary Display is a display in degrees without the number of rotations shown. If this bit is turned off, the number of rotations away from 0 degrees will be shown alongside the degree display.
Notes on Bit 2: This bit will only affect the Z axis. It controls whether or not a direction check will be performed when the Tool Check button is pressed. If this bit is turned on, direction checking is turned off, and thus, there is a possibility for the Z axis to move downward unexpectedly, depending on the Z value of Return Point #1 (G28). Therefore, it is best in most cases to leave this bit turned off to allow direction checking to be turned on (value = 0).
Notes on Bit 5: This setting overrides only the DRO display options for an axis that has bit 0 set (including the Rotary Display Mode – bit 1) so that the display does not reflect a degree symbol or any indication of the number of rotations, but appears as a linear axis.
Notes on Bit 6: This bit only works on the in Parameter 94 (4th axis). Setting this bit will cause the 4th axis to respond to Z axis only commands just like the Z axis, for example issuing an M25 with this bit set will cause the Z and 4th axes to go the home (G28) position.

Parameters 95-98 - Autotune Move Distance

These parameters hold the maximum distance that the control will move each axis in either direction from the starting point when Autotune is executed. The default value for these parameters is 2.0 inches.

Parameter 99 – Cutter Compensation Look-ahead

This parameter sets the default number of line or arc events for the G-code interpreter to scan ahead when Cutter Compensation (G41 or G42) is active. Values of 1 to 10 are allowed for this parameter.

Parameters 100–115 - Intercon parameters

These parameters are some of the Intercon setup parameters. See Chapter 10 for more information about these parameters. Changing values will change Intercon settings and may effect the output of the G-code program if it is re-posted.

Parameter 120 – Probe Stuck Clearance Amount

This parameter specifies the distance that digitizing or probing functions will move to try to clear a stuck probe condition. A stuck probe condition exists when the probe detects a point and then moves away but the probe input has not changed. It is recommended that this parameter is not changed from its default value without consulting a qualified technician.

Parameter 121 – Grid digitize prediction minimum Z pullback

This parameter specifies the minimum distance the Z axis will move upward when pulling back from a surface. The digitizing function attempts to predict the slope of a part surface because time is saved when the Z axis does not have to travel upward to the starting Z depth for every digitized point. When probe contact is made traversing in the XY plane, this parameter specifies the minimum distance the Z axis moves upward before attempting another XY plane move. Smaller values are better when the surface being digitized has smooth curves. Larger values are better for surfaces that have steep walls. It is recommended that this parameter is not changed from its default value without consulting a qualified technician.

Parameter 122 – Grid digitizing deadband move distance

This parameter specifies a deadband distance used for internal calculations when doing a clearance move. It is recommended that this parameter is not changed from its default value.

Parameter 123 – Radial Clearance Move

This parameter only applies to radial digitizing and determines what type of positioning move the digitizing probe will make should it encounter an unexpected probe contact with the surface of the part during Radial Digitizing.
Unexpected probe contact is defined as probe contact occurring while the probe is traversing towards the user defined center point.
With Parameter 123 set to 0: When the probe encounters an unexpected probe contact, the digitizing program stops data collection. The control then prompts the operator to jog the probe to a clear position. This can be any place inside the digitizing radius and above the part, that the probe stylus has a clear path to the defined center position. To restart data collection press “Cycle Start”. The probe moves in the XY plane from the position the operator placed it to the center position defined in the radial setup menu. After reaching the center position, the probe will feed down to the Z axis position it was at when the data collection was interrupted. The digitizing run will resume with the probe approaching from the defined center position.
With Parameter 123 set to 1: When the probe encounters an unexpected probe contact, it will automatically move (with probe detection turned off) to the maximum Z height, then moves the X and Y-axis to the defined center position. The probe will then move to the Z position it was at when the unexpected contact occurred. It will Then move from the defined center position, towards the measurement position it was trying to approach when the unexpected probe contact occurred and continue digitizing.
With Parameter 123 set to 2: When the probe encounters an unexpected probe contact, it will automatically move back to the defined center position (with probe detection turned off), at its present Z height. It will then move from the defined center position, towards the measurement position it was trying to approach when the unexpected probe contact occurred and continue digitizing.
Settings 1 and 2 should only be used with extreme caution because probe detection during some positioning moves is turned off, and damage to the probe or work piece could occur!

Parameters 124-127 PLC Inputs for Jogging

Parameters 124 – 127 allow up to 4 PLC inputs to be used for jogging of the first 2 axes on the control. The first 2 digits (1’s and 10’s) of the parameter specify the axis and direction, the 3rd and 4th digits specify the PLC input being used.

1’s and 10’s digit	Function
40	Jog first axis minus
41	Jog first axis plus
42	Jog second axis minus
43	Jog second axis plus

For example: A value of 140 in parameter 124 will cause the first axis to jog minus when the PLC input is closed and stop jogging when the PLC input is opened, A value of 1143 in parameter 127 the second axis to jog minus when the PLC input is closed and stop jogging when the PLC input is opened.

Parameter 128 – Handwheel (MPG) Mapping

This parameter selects how the axes are paired for handwheel operation. Each digit in the displayed number represents an axis. The first axis is at the far right. The value of each digit represents the companion axis, 1 to 5. A zero digit means no pairing. The table below shows how the digits are mapped to axes:

Axis:	5	.	4	3	2	1
Parameter value	0	.	0	0	0	0

	Axis/Companion					
Example Value	5	4	3	2	1	Comments
0.0000						No pairing.
0.1000		1			4	Axes 1 & 4 paired.
0.0043				4	3	Axes 1 & 3, 2 & 4 paired.
0.2100		2	1			Axes 1 & 3, 2 & 4 paired.
0.0021				2	1	Invalid – does nothing. Axes are paired with themselves.

Only manual axes that are paired with powered axes will produce a valid configuration. Manual axes specified by Parameter 128 must be properly configured as handwheel axes in the Motor Parameters screen of the Machine Configuration. See the Machine Configuration section earlier in this chapter.

Parameter 129 – Handwheel (MPG) Display

By default, manual axes paired by Parameter 128 are not displayed in the DRO. This parameter can force display of the manual axis in the DRO, if desired. The parameter has the same axis mapping for each digit as shown in Parameter 128. To display an otherwise hidden manual axis, set the digit corresponding to the axis number to a "1". For example, "0.1000" would display axis 4, if it is a manual axis that is paired with some other powered axis.

Parameter 130 – Z axis on/off selection (see Parameter 131)

Parameter 131 – 4th axis on/off selection (only uses 1's and 10's digit)

These parameters control the display of the 3rd and 4th axes, respectively. The tens digit of the parameter value specifies the label of the affected axis when it is enabled, with values 1-9 corresponding to axis labels ABCUVWXYZ. The ones digit specifies the label of the axis when it is disabled, with 0.0 meaning the axis is not switchable, 1.0 meaning it turns off (N), a 2.0 meaning manual (M), and a 3.0 meaning 2-axis Z (@). P130 also supports additional modes depending upon the value of the hundreds digit. For example, a value of 92 will toggle between a 3rd axis Z and a 3rd axis M and power off just the Z axis. A value of 192 will toggle the 3rd axis between Z and M and power off all axes. A value of 392 will toggle the 3rd axis label between Z and M and power off all axes and receive its positions from the 4th axis encoder input. When P130/P131 is configured for axis switching, the Setup menu displays function keys <F5>/<F6> to switch the axes.

Parameter 130 Options:

Hundred's Digit	Function Description	Parameter Value (add
Bit1	Axis motor power when switching to two axis mode.	1 = power all axes off, 0 = power 3rd off only
Bit2	Use 4th encoder input for scale input	2 = Use 4th encoder input, 0 = no manual input
Bit3	Use 5th encoder input for scale input	4 = Use 5th encoder input, 0 = no manual input

Enabled Axis:	A	B	C	U	V	W	X	Y	Z
Ten's Digit	1	2	3	4	4	6	7	8	9

Disabled Axis:	N	M	@
One's Digit	1	2	3

Parameters 132 – 5th Axis Heating Coefficient

This parameter sets the heating coefficient for the 5th axis. See parameters 20-30 for more information.

Parameters 136 – 5th Axis Cooling Coefficient

This parameter sets the cooling coefficient for the 5th axis. See parameters 20-30 for more information.

Parameter 140 – Message log priority level

This parameter controls the messages that are written to the message log, which can be accessed through the <F9> Logs function in the Utilities menu. With the Log Level set to 1, CNC10 logs numbered error messages and most other messages except "Moving...", "Jogging...", "Stopped", etc. At Log Level = 9, all messages are logged including user prompts. Message logging can be disabled be setting this parameter to –1.

Parameter 141 – Maximum message log lines

This parameter is the number of lines that will be kept in the message log. If this parameter is set to 10,000, for example, the newest 10,000 messages will be retained. CNC10 will delete the oldest messages, trimming the log file to the given number of lines at startup and periodically while CNC10 is in an idle state. Parameter 142 controls the frequency of the log cleanup.

Parameter 142 – Message log trim amount

This parameter is the number of additional lines above the minimum that can be added to the log before it is reduced to the minimum size. Setting this parameter to a lower value will cause the log file to be trimmed to its minimum size more often. The higher the value, the less often the log will be trimmed. The speed of the disk drive and total size of the log file at the time it is trimmed will determine how long the log cleanup takes. Under most circumstances, using 10,000 and 1,000 for parameters 141 and 142 will provide a reasonable and useful log size with no noticeable effects on performance. If parameters 141 and 142 are set to excessively high values, the message "Trimming excess lines from log file" will be presented. This message will appear at startup and very infrequently when CNC10 is idle. Normal operation can proceed after the message disappears. If the delay is unacceptable, reduce the values of parameters 141 and 142.

Parameter 143 – DRO Properties (load meters, 4/5 digits, DTG)

This parameter controls the display of the axis load meters and 4/5 digit DRO precision.

Bit	Function Description	Parameter Value
0	Enable Load Meters	Enable = 1, Disable = 0
1	Load Meter Outline	Enable = 2, Disable = 0
2	DRO 4/5 Digit Precision	5 digits = 4, 4 digits = 0
3	Mini DRO (Distance to Go)	Enable = 8, Disable = 0

Use a value of 3 to display load meters with outlines. The axis load meters will be colored green for values that are up to 70% of maximum power output, yellow for values between 70% and 90%, and red for values between 90% and 100%. The axis load meters appear below the DRO for each axis (see Chapter 1).

Parameter 144 – Comparison Rounding

This parameter determines the built in rounding for the comparison operators ('EQ', 'NE', 'LT', 'GT', etc.) in expressions. Rounding of comparison arguments is necessary due to extremely small errors that are part of every floating point calculation. The result of such errors is that two floating point values are rarely exactly equal. The value of parameter 144 represents the precision of comparison in places after the decimal point. If the parameter is set to 9.0, for example, then comparison operators will declare two numbers that differ in value by less than 0.0000000005 as being equal. The value 0.0 is a special value that turns comparison rounding off. When comparison rounding is off, it is up to the G code programmer to build the precision into conditional statements, for example "IF ABS[#A - #B] LT 0.00005 THEN GOTO 100". When comparison rounding is off, the "EQ" usually returns "false". If parameter 144 is set to 9, the programmer can shorten the previous example to "IF #A EQ #B THEN GOTO 100".

Parameter 145 – Advanced Macro Properties (Fast Branching)

This parameter turns fast branching on (1) and off (0). The other bits of this parameter are reserved for future use.

If fast branching is disabled, CNC10 searches forward in the program for the first matching block number and resumes searching, if necessary, from the top of the program. For this reason, backward branches take longer than forward branches and backward branch times depend on the total program size. If the program is sufficiently large, use of the GOTO statement could introduce temporary pauses.

When fast branching is enabled, CNC10 remembers the locations of block numbers as it finds them during program execution. Backward branches always take place immediately. The first forward branch to a block not yet encountered will take additional time as CNC10 searches forward for the block number; however, subsequent forward branches to that block number will take place immediately. The trade-off for using fast branching is that all line numbers at a given level of program or subprogram must be unique and programs will use more memory (approximately 16kilobytes of memory for every 1000 block numbers in the program.)

Parameter 146 – Feed Hold Threshold for Feed Rate Override

This parameter sets the lowest value permitted as the feed rate override percentage before feed hold is engaged. Feed hold will be released when the override percentage is greater than this value.

Parameter 150 – Run-Time Graphics

This parameter controls the default value of the Run-Time Graphics option in the Run Menu. If this parameter is set to 0.0, the RTG option in the Run Menu defaults to OFF when CNC10 is started. If the parameter is set to 1.0, the RTG option defaults to ON when CNC10 is started.

Parameters 152 – 5th Axis Autotune Accel Time and Ka

This parameter sets the autotune accel time and Ka for the 5th axis. See parameters 87-90 for more information.

Parameters 156 – 5th Axis Autotune Move Distance

This parameter sets the autotune move distance for the 5th axis. See parameters 95 – 98 for more information.

Parameter 160 – Enhanced ATC

This parameter controls enhanced automatic tool changer (ATC) options. A value of 1 indicates a nonrandom type of ATC and a value of 2 indicates a random type ATC. A value of 0 disables enhanced ATC features. A warning is displayed when attempting to enable enhanced ATC features as these features work in conjunction with specific PLC programs. The enhanced ATC option has the following characteristics:

The beginning of an M6 call, whether it be a customized CNC10.M6 routine or not, flags the job file, setting the ATC error flag field to 1.
The end of an M6, whether customized or not, performs the following:
(a) The atc error flag is set to zero.
(b) The tool number displayed on the screen is updated and this value is saved in the CNC10.JOB file.
(c) The tool library bin fields are updated in this manner:
If there was a valid tool in the spindle at the start of the M6, then the tool library bin field for this tool will be updated with either the “putback” field for that tool (if nonrandom type) or the current ATC carousel position (for random type). For both random and nonrandom types, the “putback” field is set to 0. The “putback” field is an internal field for each tool in the tool library. It can be displayed by using the CNC10CONV utility with the –dt option to display the tool library.
For nonrandom types, the new tool now in the spindle will have it’s “putback” field updated to the current ATC carousel position.
For both random and nonrandom types, the new tool now in the spindle has the bin field set to 0.

The current ATC carousel position is constantly monitored. When there is a change, the ATC bin field in the CNC10.JOB file is updated and the file is saved. The ATC carousel position is read from PLC bits OUT41-OUT48, which should be written by the PLC program in a binary format (not BCD).
At the start of running a job, to include MDI mode, the ATC error field is checked. If this field is 1, then a warning message is displayed with a prompt to either clear the fault by entering a 'Y' or canceling the job by pressing some other key.
A tool change is not performed if the requested tool is already in the spindle.
An M107 command sends the bin number for the specified tool number, not the tool number.
For random types, tool changes in Intercon are posted as a tool change (Tnn M6) followed by a pre-fetch command for the next tool in the program (Tn2 M107). This allows the PLC program to rotate the tool carousel to the next tool while a job continues with the current tool.
For random types, a job search for a tool number will look for lines of the form Tnn M6, i.e., the search bypasses lines of the form Tnn M107, which are just pre-fetch commands.
The tool library allows editing of the bin fields to specify which carousel bin number the tools are stored in.

Parameter 161 –ATC Maximum Tool Bins

This parameter sets the number of tool changer bins (carousel positions) used with the enhanced ATC option described above. PLC programs are responsible for reading this value. The tool library interface uses this parameter to validate bin fields and perform initialization of the bin fields.

Parameters 166 – 5th Axis Properties

This parameter sets the axis properties for the 5th axis. See parameters 91-94 for more information.

Parameters 170-179 – XPLC Parameters

These parameters are accessed by the XPLC through LP0 - LP9 commands. Please see chapter 5 page 54 for more information regarding these parameters.

Parameters 180 – File Transfer COM Port

This parameter specifies which COM port will be used for file transfer. Accepted values are 0 disabled and 1-4 for COM1 – COM4. Setting this parameter to an accepted value other than 0 will provide an <F6> Download and an <F7> Upload key in the Load Job Menu.

Parameters 181 – File Transfer Baud Rate

This parameter set the maximum file transfer rate for serial communication. The value of this parameter is in KBaud and has a range of 1.2 to 115.2. The default is 19.2Kbaud. The longer the serial cable the lower the baud rate that can be used for file transfer.

Parameters 182 – File Transfer Bit Parameters

This parameter sets the number of data bits, type of parity and the number of stop bits for the serial communication file transfer. The default value is 801 for 8 data bits, no parity and 1 stop bit.

Digit	Function	Value
1's	Stop bits	1 or 2 stop bits accepted
10's	Parity	0 = No Parity; 1 = Even Parity; 2 = Odd Parity
100's	Data bits	5 – 8 data bits accepted

Parameters 183 – File Transfer Flow Control

The setting of this parameter determines the COM port file transfer flow control.

Value	Meaning
0	No Flow Control
1	Software (XON/XOFF) Flow Control
2	Hardware (CTS/RTS) Flow Control

Parameters 184 – File Transfer Timeout

This parameter is used to the timeout for downloads. When you press the <F6> Download button in the load job menu you have this ampunt of time to start the file transfer or it will timeout. The default value of this parameter is 10 sec, but can be set from 6 secs to 600 secs (10 mins).

Parameters 185 – File Transfer Options

This is a 2 bit parameter to set file transfer options.

Bit	Function	Value
0	Ignore CR on downloads	1= Yes; 0 = No
1	Translate NL (new line) to CR on upload.	1= Yes; 0 = No

Parameters 188-199 – Aux Key Functions

These parameters are used to assign a function to aux keys 1-12. The following is the list of possible functions that can be executed when an aux key is pressed.

Function	Parameter Value
No Function	0
Input X Axis Position	1
Input Y Axis Position	2
Input Z Axis Position	3
Set Absolute Zero	4
Set Incremental Zero	5
One Shot - Drill	6
One Shot - Circular Pocket	7
One Shot - Rectangular Frame	8
One Shot - Frame	9
One Shot - Face	10
Execute M Code file	*m*11*
Free Axes	14
Power Axes	15

Function	Parameter Value
XYZ Set Absolute Zero	16
One Shot - Drill Bolt Hole Circle	17
One Shot - Drill Array	18
Jog Axis 1 (+)	21
Jog Axis 2 (+)	22
Jog Axis 3 (+)	23
Jog Axis 4 (+)	24
Jog Axis 5 (+)	25
Jog Axis 1 (-)	31
Jog Axis 2 (-)	32
Jog Axis 3 (-)	33
Jog Axis 4 (-)	34
Jog Axis 5 (-)	35

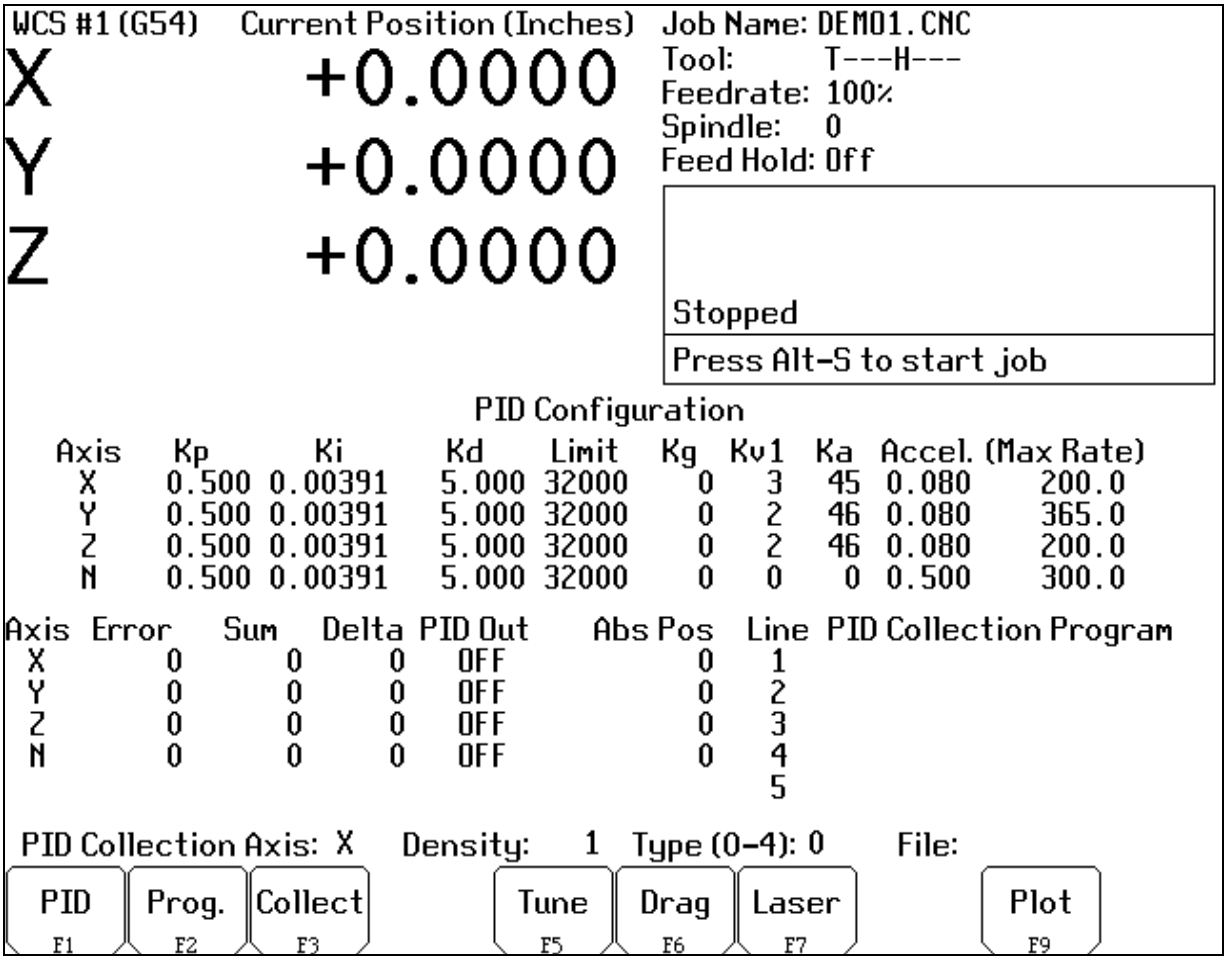
The Input Axis Position functions must be used with the Set ABS/INC Zero functions. After entering the desired value at the input field provided by the Input Axis Position function, press an aux key assigned either the function Set ABS Zero or Set INC Zero.

* *m* is the number of the M code to execute. For example, if the parameter value is set to 7211, the file CNC10.M72 will be executed when the Aux key was pressed,

All remaining parameters are reserved for further expansion.

PID Configuration

(F4 from Configuration)



Pressing <F4> from the Configuration screen will bring up the PID Configuration screen. The PID Configuration screen provides qualified technicians with a method of changing the PID dependent data to test and configure your machine. The PID Parameters **should not** be changed without contacting your dealer. Corrupt or incorrect values could cause damage to the machine, personal injury, or both.

F1 - PID Parameters -

(These parameter values should be recorded on the Control Parameters page at beginning of manual.)

This option is for qualified technicians **ONLY**. Altering these values will cause **DRAMATIC** changes in the way the servo system operates, leading to possible machine damage. **DO NOT** attempt to change these parameters without first contacting your dealer.

*** NOTE**: Some of these values are set automatically by the Autotune option. (See F5 – Autotune)

Kp, Ki, Kd, and Limit

The chart below lists the optimum value for each of these parameters based on motor size and drive current rating. Kp and Ki should NEVER be changed under any circumstances. Kd may be changed in some rare cases. Contact technical support for your machine before attempting to change these parameters to avoid serious damage to the machine.

Motor/drive amp combination	K_p	K_i	K_d	Limit
Redcom 17 in/lb 9A	1	0.00391	10	32,000
SEM 29 in/lb 9A	1	0.00391	20	32,000
SEM 29 in/lb 12A	1	0.00391	20	32,000
SEM 29 in/b 15A	1	0.00391	15	32,000
SEM 40 in/lb15A	1	0.00391	10	32,000
SEM 40 in/lb 25A	1	0.00391	10	32,000

To check your drive's Amp rating, look on the drive cover for a label with 2T, 3T, or 4T written on it.
2T = 25amp 5T = 9 amp
3T = 15 amp 6T = 6 amp
4T = 12 amp

Kg

Kg is called the Gravity Constant. It indicates an imbalance in an axis. A high Kg value means that the motor is working harder to move the axis in one direction than the other. Typical Kg values range from -5 to +5 for X and Y-axis and -10 to +10 for Z-axis.

Causes of high Kg

1) A bed mill with an incorrect weight on the counter balance.
2) A loose gib. (See TB037)
3) A failing support bearing.

In version 6.0 and above, a drag plot will show you the drag in both directions; the "gap" or difference between the two lines in the plot is the Kg value.

Kv1

Kv1 describes the amount of current it takes to slow jog the axis. By knowing the amount of current it takes to slow jog the axis, we can know the amount of friction or mechanical resistance that axis has. Common values for Kv1 on a knee mill are 10 - 20 on the X, 12 - 22 on the Y (higher because it carries the weight of the X) and 3 - 10 on the Z.

Software
Version 6.0 and above: You can see a plot of Kv1 over an entire axis by doing a drag run ("Drag" in the PID screen) and then plotting it.
5.27 and earlier: Slow Jog the machine and observe the PID OUT parameter in the PID screen and take a visual average of the numbers you see being displayed. You may get different results depending on the direction you are jogging. (See Kg)

Note: Remember that a Kv1 of 10 on a machine equipped with a 15 amp drive is not the same as a machine with a 12 amp drive. To figure out how much current is required to slow jog the machine, take Kv1 and divide by 128 and then multiply the result by the amp rating of the servo amplifier. Example: Kv1 = 35, Servo amp is a 3T or 15 amp drive (see Table 1 for turn to amp conversions) therefore 35/128 = .2734 multiplied by 15 = 4.1 amps! Way too much power to slow-jog an axis.

Ka

Ka describes the current necessary to accelerate the axis to its maximum velocity at the acceleration rate set by Autotune.

Example
If Ka is 54, Max Rate is 330, and Accel time is .245 seconds, it takes 54/128 times 15 amps = 6.3 amps of continuous current for .245 seconds to accelerate that axis to 330 inches per minute.

Accel

Accel is the time for the axis to reach maximum velocity. It is set by Autotune to be as fast as possible. Sometimes a customer may desire a slower rate. For example an Accel rate of .1 second is very fast and the machine will appear to be "jerky". You can change the Accel rate to a slower setting but Ka must be manually changed at the same time. If Autotune produced a .1 second Accel time and a Ka of 60 and you want to increase the Accel time to .2 seconds you would manually double the Accel value and divide the Ka value by 2. The new values would be an Accel of .2 and a Ka of 30. This will produce a noticeable difference.

Max Rate

Max Rate is 80% of the maximum physical speed on the axis. It is set at 80% to compensate for environmental changes. For example, without this, on a hot day when the lube is "loose" Autotune might find a value of 400 IPM. In a few months it may be very cold in the machine shop and the lube will be like molasses. Now the control will try to run at 400IPM and, unable to, will most likely give a position error message.
*Note: This parameter can only be set by Autotune from the PID Configuration menu. To set this parameter manually, see Jog Configuration section above.

Deadstart

Deadstart has to do with direction reversal of an axis. Autotune does a good job of setting the Deadstart. The deadstart usually never has to be changed on a Milling machine. Sometimes very light wood routing tables with very low resistance and low inertia can benefit with a Deadstart change along with other "hand tuning." Call in if you have this case.

WARNING: Improper PID values can ruin the machine, cause personal injury, and/or destroy the motor drives!!!

F2 - PID Collection Program

This option allows qualified technicians to test the PID parameters by entering up to 5 lines of G-codes to be executed with the Collect Data command below.

F3 - Collect Data

This option allows qualified technicians to collect data on the movement of one of the motors. It uses the values located in the axis and density fields at the bottom of the screen and the PID collection program to collect the data. When this option is selected, the controller executes the PID collection program and collects data on the selected axis. The data is saved using the file name entered at the file prompt at the lower right hand side of the screen. The information in the lower left hand side of the edit window provides information to qualified technicians about the selected axis.

F5 - Autotune

Autotune automatically sets the following parameters: Kg, Kv1, Ka, Max rate, Accel/Decel time, and Dead Start. A full description of these values along with their relationship to the machine and Autotune is given above. Autotune characteristics can be changed via Machine Parameters.

Note:

When performing an Autotune on an **AC Brushless** machine for the first time you should set the Autotune Move Distance parameters 95 through 98 for a short distance, then increase until the Max Velocity levels off. Start with 0.5 inches, and then increase the parameter number until the Max Velocity does not increase significantly. Autotune will overshoot this distance by 2 times, so if the distance is set to 1 inch expect the movement to go out to 3 inches. Care should be used in setting the starting point for Autotune so not to crash the machine.

If the machine has low friction and high mass, the drive may stop due to overcurrent. If this happens the max velocity and the jog rates may be set to zero by the Autotune. Reset these numbers to the previous values and move back to the starting position. Decrease the travel distance parameter and try again.

F6 - Drag

This feature can be used to determine whether your machine is binding anywhere along the axis travel. Binding can occur as the result of poor lubrication, gibs too tight, debris in the ways, or misalignment in the bearings. Having the availability to readily check the drag values graphically on the machine allows for preventative maintenance to ensure proper operation of the CNC machine. For additional information, reference Tech Bulletin TB047.

Steps for the drag test:

(1) First, go to the proper screen as follows: <F1> Setup, <F3> Config, type 137 at the password prompt and <Enter>; then <F4> PID and <F6> Drag. Choose the axis you want to check by pressing <F1>. Hit the green <CYCLE START> button on the jog panel to start the procedure.

(2) The drag diagnostic program will command the machine to go to a limit switch along the axis chosen. The machine will reverse direction, moving to the opposite limit switch and then reverse again moving to the starting limit switch. Approximately 200 torque values for the chosen axis will be recorded during the drag procedure. Data is saved into DRAG_X.OUT, DRAG_Y.OUT, or DRAG_Z.OUT in the CNC10 directory, depending upon the axis label for the axis under test.

(3) To display the data, go to the main menu and use <F2> Load, highlight [CNC10], press <F10> to enter that directory. Highlight the file for the axis you want. Press <F8> Graph to display data.

(4) Data displayed should remain between the red horizontal lines that bound the acceptable operating range, which is 25% of the maximum power that can be supplied to the motor. If drag exceeds 25% of the maximum power anywhere in the axis, the message "*excessive axis drag*" will be displayed on-screen. Refer to Figure 1.

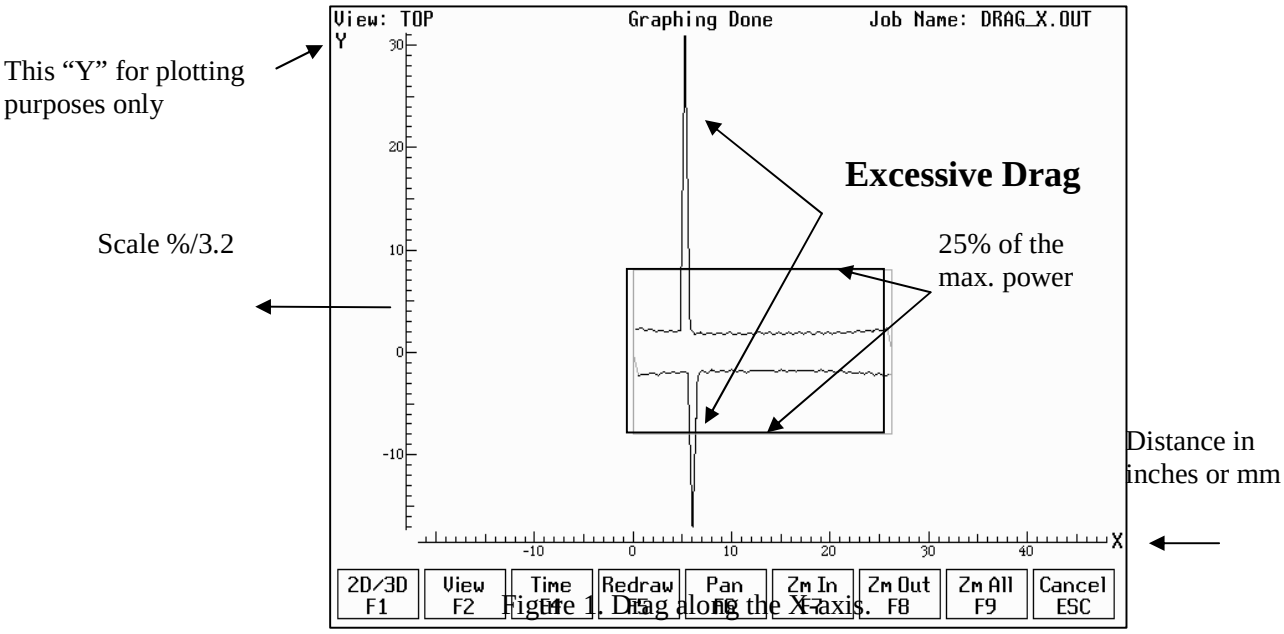


Figure 1. Drag along the X-axis.

F7 – Laser

This option is used to make automated laser measurements and create or adjust the ball screw compensation tables accordingly. Additional information can be found in Tech Bulletins TB048, TB070.

It is very **important** to run Autotune and Drag mapping for checking the status of the machine before proceeding with laser compensation. Be certain drag and PID values are within reasonable ranges before you continue with the laser mapping procedure.

First, install the laser measurement software on a separate PC or laptop. The laser equipment supplier provides the installation disks. A LDDM icon will be created after complete the software setup. Then, make sure all laser cables are connected and the serial cable is connected to your PC (laptop) serial port. Turn on the power for the laser module and double click on the [LDDM] icon.

Setting up for the X axis

Move the X-axis to Home position. Double click on the [Linear] icon and check on the [Intensity] to help align the laser beam. Set up the laser (with mirror assembly) on the table so the beam is aiming in the X+ direction. Adjust the beam to run parallel to the X-axis. Mount the mirror on the head in a drill-chuck and set the window perpendicular to the beam. Move the window into the beam, in the YZ plane until you see the beam reflecting back into the laser.

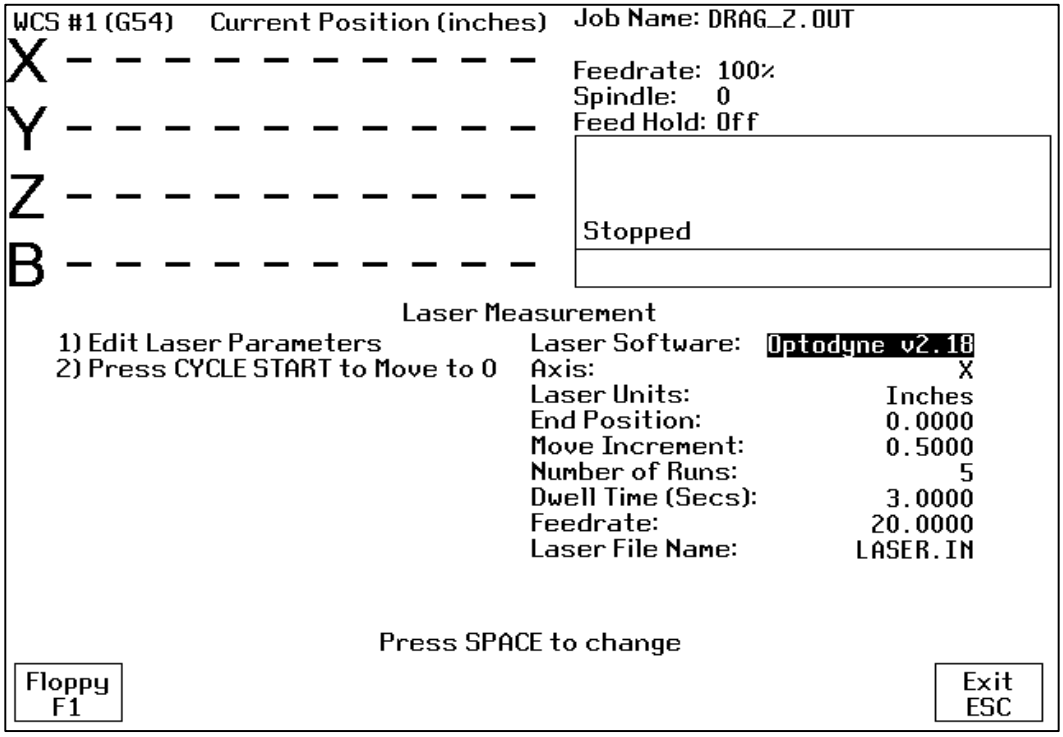
(1) At the X- end of travel, move in the Y and Z-axes until intensity is 100 %.
(2) Begin to move in the X+ direction (do not move in Y or Z) and watch for the beam to move outside the window.
(3) If and when the beam moves outside of the window, use the alignment screws on the laser to bring it back in and maximize the intensity reading. When the intensity reading is at 100%, repeat step (2) until you get 100% intensity at Max X+ travel.
(4) Repeat steps (1) to (3) until a 100% intensity reading is measured from both ends of the axis.

Data Collection

On the PC, click Setup. Fill in the identification info. Choose the X-axis, Set start to 0. For End use End position off the Control screen. Number of points is (End Position / Move Increment)+1 or in our example, (34.5/.5)+1=70. Make sure the number of runs on the PC matches the control. Make sure the following check boxes are NOT checked:

Forward only, Positions Shown as below, and both ATC boxes. Click on the Auto button. Set Target Window to 0.1 inch, Trigger Dwell to 0.9 second, and Velocity Threshold to 0.1. Make sure Backlash is not checked. Click save, then OK to get out of the setup and then Reset.

At the CNC10 control the screen should look like this: **Laser Software** should be set as Optodyne v2.18 by using <Space bar>. **Axis** should be X. **Laser Units** should be inches. **End Position** should be set as your X travel rounded down to the nearest half inch. (e.g. 34.87 inches travel would be 34.5 inches.) **Number of runs** - use 1 on the first trial, 2 on second trial, etc. **Dwell Time** should be 3.0, **Feedrate** should be 20.0, and **Laser File Name** should be KnnnnX.in, where nnnn is the machine serial number. After pressing the <Cycle Start>, the X-axis will go to its home position.



At the PC, press [Reset] to reset the zero position for the laser. Click [Start] after hearing 2 beeps or seeing a green light on the PC screen, then press <Cycle Start> on the Control. You should hear a beep from the PC every time the machine pauses. Watch the PC to verify it is collecting all points. When it is finished, click OK then save as A:\KnnnnX.in. You may want to also save it on the C drive with a filename like C:\lddm\KnnnnX.in for later analysis. After LDDM writes the file to disk, take the disk out and put it into the floppy drive of the CNC10 control. It will have a message asking for the disk. Press a key on the control. After the file is read by the control, changes in backlash and turns ratio values will be displayed. Then, run the procedure again to verify the position accuracy of the X-axis.

Repeat the same procedure for the other axes

Printing the Laser Output File

System requirements – CNC10 Windows Demo software,

(1) Set screen to B/W by setting parameter #7 to 1.

(2) Use <F2> Load and <F8> Graph to view the files listed below from the CNC10 directory

i. CNC10_X.OUT
ii. CNC10_Y.OUT
iii. CNC10_.OUT

(3) Press <Print Screen>, putting the contents of the screen into the clipboard.
(4) Open Microsoft paint and paste contents of clipboard. Invert the black background colors to white color.
(5) Print out a hard copy of the Laser file.

F9 - Plot

This option is used by qualified technicians in order to plot data.

Chapter 3

Integration Testing

Integration Testing and Checkout

Electrical cabinet:

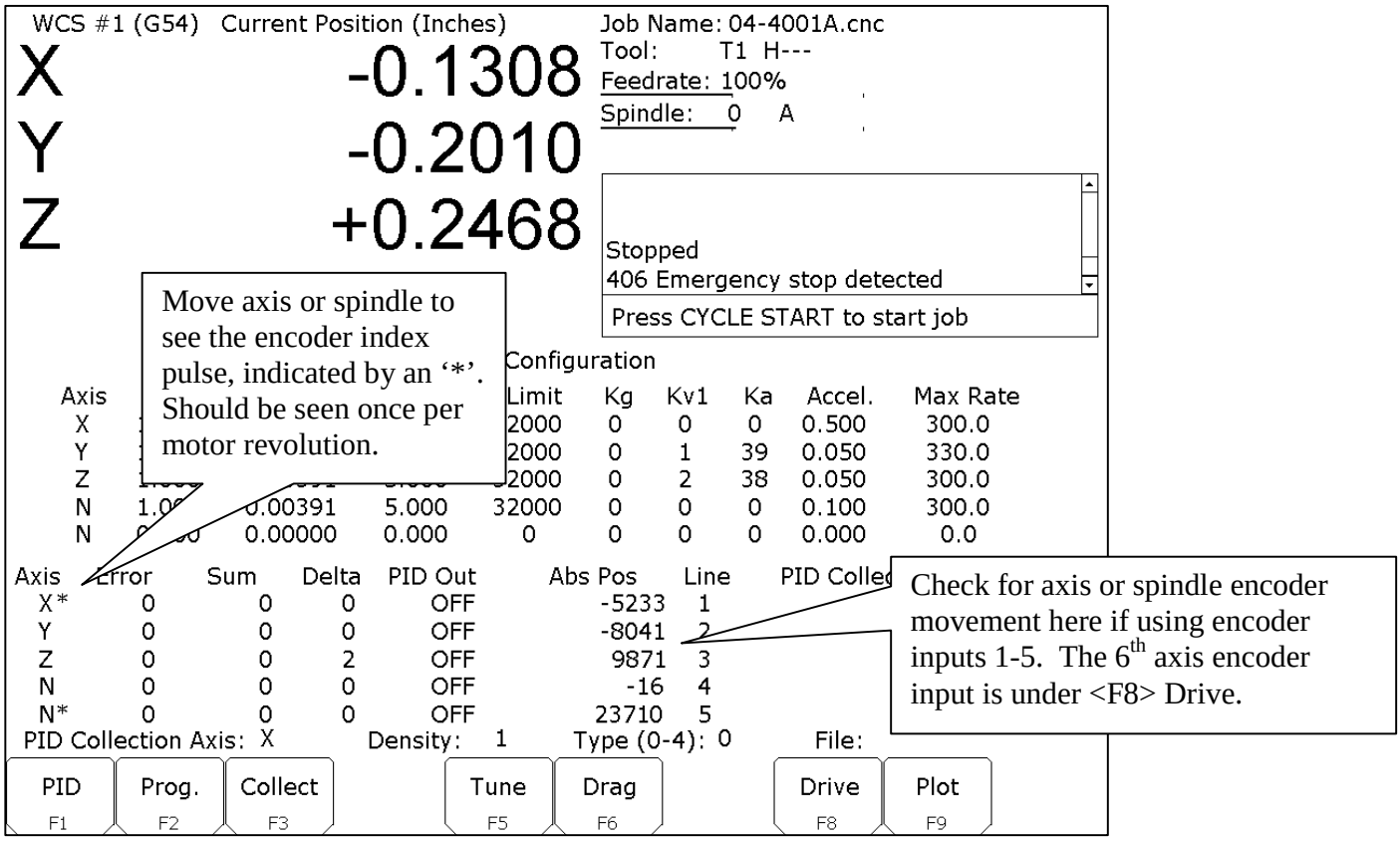
- ☐ Make sure that all holes in the cabinet are plugged so that while in operation no coolant, lube oil or chips can get into the cabinet or console.
- ☐ Make sure that all brackets, strain relief's and hardware are tight.
- ☐ Check all wire connections. (Screws tight, not clamped on the insulation, all strands in connection and not to much bare wire showing)
- ☐ If the machine was shipped after initial checkout, recheck connections for tightness. Some connections may have come loose during machine transportation.

Power up:

- ☐ Turn on main disconnect (**not the control**) check for proper voltage at the control input power fuses or circuit breaker.
- ☐ Power up control (**E-Stop pressed**)
- ☐ Enter DEMO code to enable control software
- ☐ Check for proper voltage at contactor or inverter input
- ☐ Program inverter (can be done later)

Encoder inputs: (**E-Stop still pressed**)

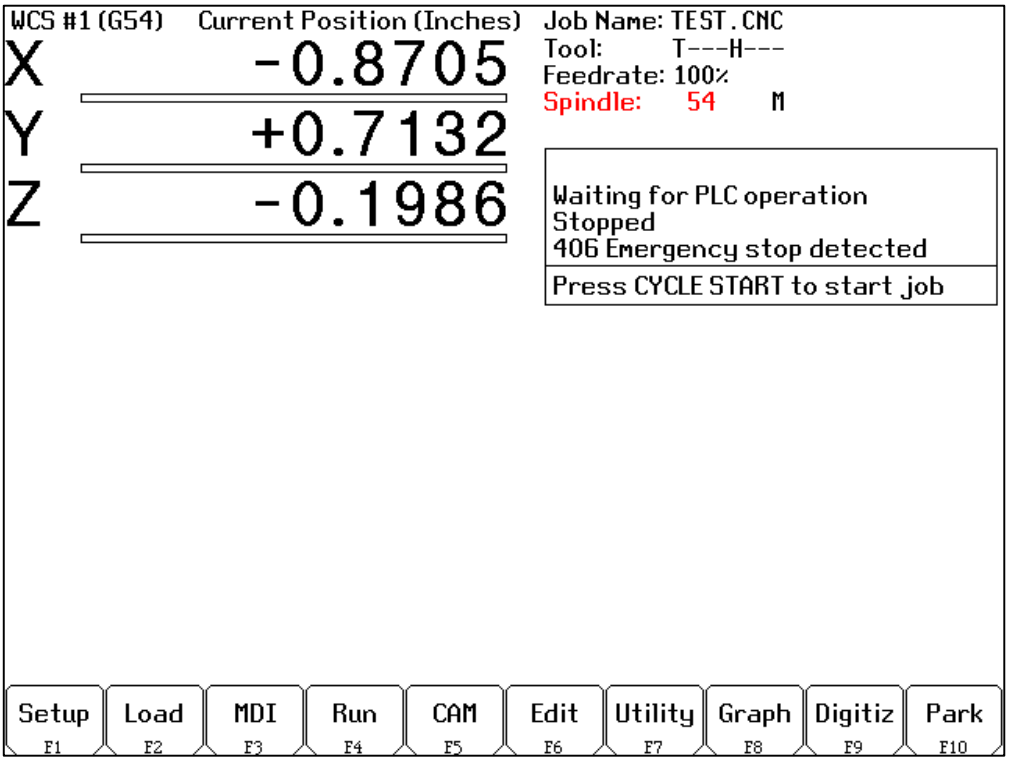
- ☐ Check axis motor encoder inputs
  a. Go to the PID menu by:
  Pressing <F1> Setup, <F3> Config, enter <137> (password), then <F4> PID



b. Manually move each axis in the plus and minus direction while watching the “Abs Pos” in PID menu for plus and minus counts.

☐ Spindle Encoder (Another method with no check for index pulse)

a. Go to the Parameters menu by:
Pressing <F1> Setup, <F3> Config, enter <137>, then <F3> Parms
b. Set parameter #78 to a 1
c. Press <Esc> until you are at the main menu.



d. Manually turn spindle and check Spindle in the status window for movement.

Air Pressure:

☐ Check air connections and pressure settings for:

a. ATC systems (If applicable)
b. Spindle brake (If applicable)

Axis Movement and Limit Switches:

☐ **Release E-Stop**

☐ Jog each axis (+/–) check for proper travel direction. Turn on direction reversal in “Motor Parameters” menu if necessary. Changes regarding direction reversal can be found in chapter 2 “Software Setup” page 14

☐ Limit switches connections:

a. In fast continuous jog move an axis in the - (minus) direction and manually trip the corresponding limit switch for that axis. After tripping the - (minus) limit switch the following should occur. The axis motor will stop and the “**_- limit (#_) tripped**” message will appear.
b. With the limit switch still tripped try to jog in the + direction. The axis motor will not move or will move slightly and the limit tripped message will repeat itself.
c. Switch to SLOW jog mode. The axis will now move in the + direction.
d. Release the limit switch. The “**Limit switch cleared**” message will appear
e. Repeat for the + direction
f. Repeat a-e for each axis that has limit switches.

Machine Homing:

- ☐ If homing to switches verify they are set in the "Motor Parameters" menu
- ☐ Press <Cycle Start> to home the machine.
- ☐ The type of homing is found in the "Control Configuration" menu, see chapter 2 "Software Setup" page 8 for changes.

Machine configuration:

- ☐ Verify turns ratio using a dial indicator or other measuring device and moving each axis in MDI or incremental jog. Procedures for modifying turns ratio can be found in chapter 2 "Software Setup" page 14
- ☐ Verify software travel limits are set, see chapter 2 "Software Setup" page 12
- ☐ Verify Max spindle speed and Min spindle speed is set if necessary, see chapter 2 "Software Setup" page 8 for more info.
- ☐ Run Autotune, in the PID menu, to set the Accel/Decel and Max Rate.
- ☐ Run Drag Plot, in the PID menu, to check for smooth machine travel (No obvious bind points)

Lube pump operation:

- ☐ Verify lube oil is getting to all parts of the machine.
- ☐ Set the lube timer if needed.

Spindle operation:

- ☐ Check manual on/off and correct direction CW/CCW
- ☐ Use MDI to check M3/M4/M5 automatic spindle operation (if applicable)
- ☐ Use tachometer to set spindle RPM.
- ☐ Check high/low range switch operation. (If applicable)
- ☐ Check spindle brake operation. (If applicable)

Coolant operation:

- ☐ Check manual on/off flood and/or mist.
- ☐ Use MDI to check M7/M8/M9 automatic coolant operation.

Tool changer operation: (Carousel Style) ***Needs Revised for new PLC programs***

- ☐ Verify parameters 160 and 161 are set correctly. See software section chapter 2 page 38
- ☐ Verify parameter 178 is set correctly. Machine Parameter 178. This is a bitmapped parameter that can switch the state of some of the inputs. The value of these bits determines what the default state of the input will be. The bits that are related to the ATC system are:

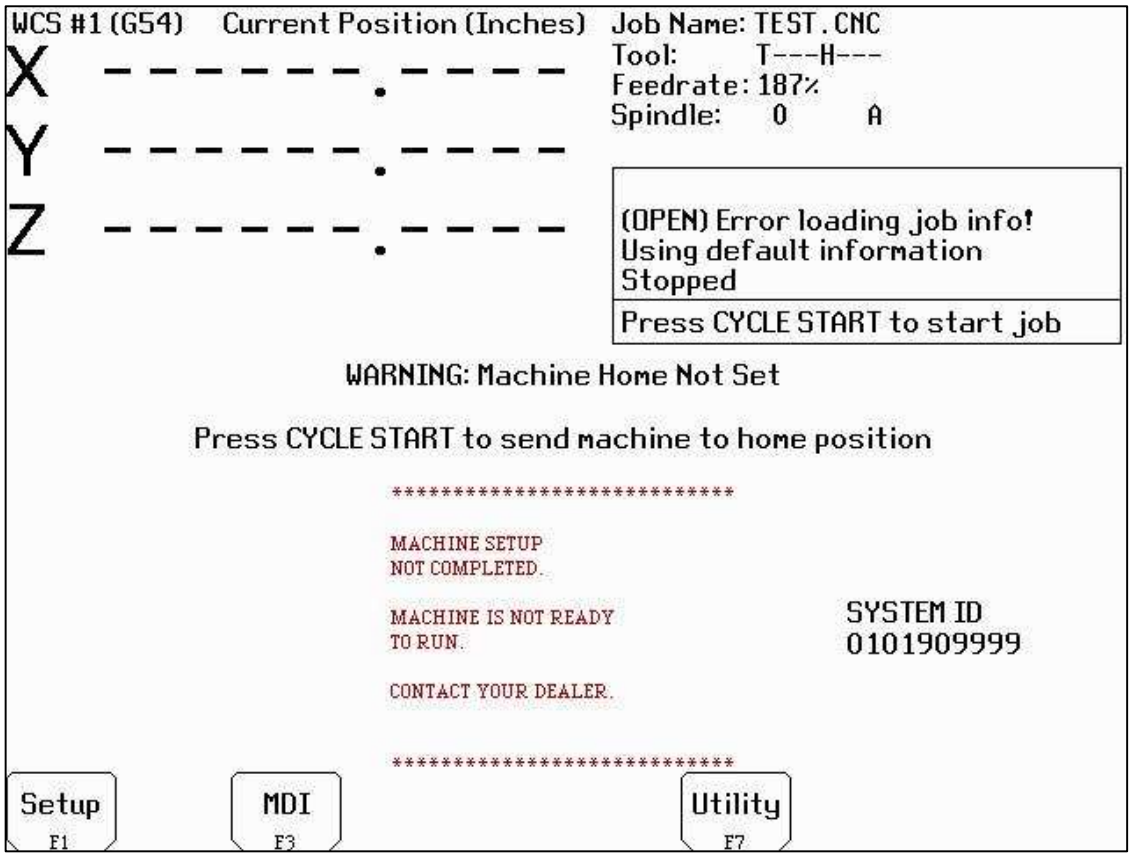
AIR_SIGNAL	P178 Bit 2 (4)	(Setting this bit will make AIR_SIGNAL normally open)
TOOL_COUNTER	P178 Bit 3 (8)	(Setting this bit will make TOOL_COUNTER default state closed).
INV_ZERO_SPEED	P178 Bit 5 (32)	(Setting this bit will make INV_ZERO_SPEED normally open).
CHILLER_FAULT	P178 Bit 10 (1024)	(Setting this bit will make CHILLER_FAULT normally open.

- ☐ These settings are configured during control system integration and should not be changed.
- ☐ Using the tool index keys on the jog panel check that the carousel rotates in the correct direction. Tool Index + should count up and Tool Index – should count down
- ☐ In the tool library Press <F3> Init and follow instructions.
- ☐ Z axis positioning, In MDI enter "M21" and check that the Z axis position is correct for a tool change, then enter "M22" and verify the Z axis moves up to a clear position for carousel rotation.
- ☐ Spindle orient, In MDI enter "M19" and verify the spindle orients to the correct position.
- ☐ Carousel in/out operation, With Z axis clear in MDI enter "M80" to move the tool carousel in check position, then enter "M81" to move the carousel back out.
- ☐ Tool release, In MDI enter "M15" to check the tool release and air blow on, then enter "M16" to clamp the tool and turn off the air blow.

- ☐ In MDI enter “M6 T1”, verify that the carousel positions to the correct tool number and the tool change completes with no errors.

Run System Test

System Test is a new feature released in software versions 8.23 (DOS) and 1.10 (CNC10). The purpose of the System Test is check basic machine functionality and to verify key parameters are set correctly. If the System Test has not run completely and passed, a message will be displayed on the screen at power up (see below). This message will continue to be displayed at power up, until the System Test has been run successfully.



Warning message displayed until system test is completed, before homing

Requirements before running System Test

3. Before running the System Test the control should be fully integrated and working correctly. Which means; the machine homes correctly, autotune has been run, drag tests completed, travel limits are set, and spindle and coolant functions are working. ATC systems need to have run successful tool changes as well.

4. Machine table should be clear so it can run through all travel limits without running into anything.

5. The spindle should be empty and the spindle face must not be able to run into the worktable at the bottom of its travel. **DO NOT START** the system test with a tool in the spindle.

Addition requirements for ATC systems

6. Position a TT1 on the worktable and connect the TT1 cable. Jog the worktable to position the spindle over the TT1 then press <F1> Setup, <F1> Part, <F9> WCS Table and <F1> Return. Enter the X and Y values displayed on the screen in return point #3 and set the Z value to zero. Also make sure the spindle face will not run into the TT1 at the bottom of its travel (if it doesn't clear, do **NOT** run the System Test).

7. Insert at least four tools into the tool carousel. Load one into bin #1 and another into the maximum bin number (ex. Load into bin 16 if a 16 – Tool ATC). The other two tools should be loaded into the carousel to balance it out. In the case of a 16 - Tool ATC load bins 5 and 10. The tools must have lengths that differ by at least 0.010 inches (0.25 mm). Do **NOT** use fly cutters, diamond tip tools or none conductive tools.

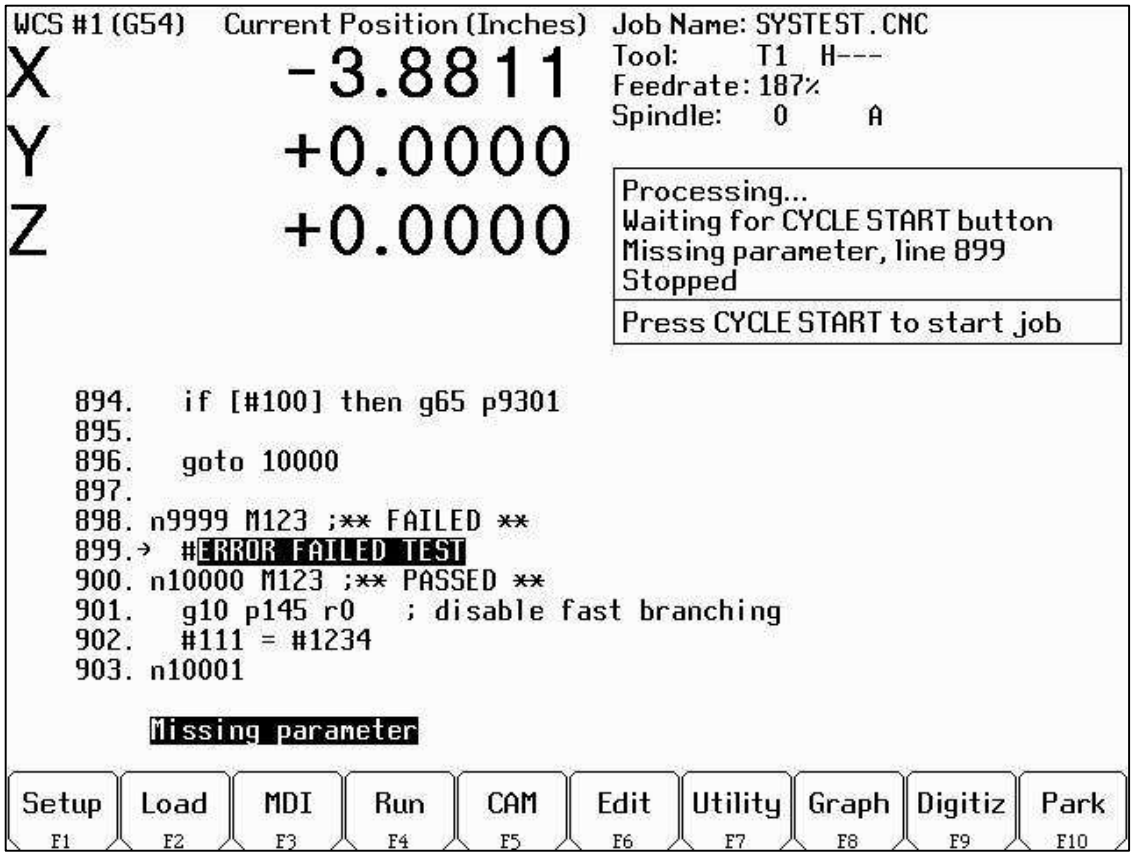
Running the System Test

3. To run the System Test, press <F1> Setup, <F3> Config, enter the correct password, and then press <F5> Test. A message will be displayed prompting the user to press Cycle Start to run the test or ESC/Cycle cancel to exit without running the System Test. If the test detects an error at any time the program stops and an error on the screen appears (See figure 3). Load systest.out and edit to see exactly what failed (See figure 2).

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;     Checking travel limits and home set
***  FAILURE:  AXIS 1 INVALID TRAVEL LIMITS
***  FAILED  ***
```

Figure 2: Systest.out example of failure message

4. System Test starts by moving the Z-axis to home. Then checks parameter 6 and 160, if P6 is set to 1 and 160 is set to a non-zero value the System Test will run all tests.



5. The travel limits and machine home are verified that they are set. Travel limits are not tested for axes that are labeled as M, N, or S, or that have the rotary bit set in parameters 91-94 (axes 1-4) and or 166 (5th axis).

Figure 3: Example of what you'll see when a test fails.

6. At this point, if the system is an ATC system a prompt will be displayed asking if you want to run all tests or just a certain one. This way if you failed the test previously and only want to run a certain test to verify you fixed the problem. It's possible without running the entire problem and risk faulting out again.

7. If full test has been selected, the next test is the TT1 test of ATC systems. If not an ATC system the test will jump to number 6. After machine home and the travel limit have been checked the following G-code message will be displayed on the screen:

```
5.      ; * Connect TT-1 and verify the XY location of the TT-1
6.      ; * is set in return point #3.  Verify that the spindle
7.      ; * face is above the TT-1 when at the Z minus travel
8.      ; * limit.  After pressing CYCLE START,
9.      ; * trigger the TT-1 twice.
10.     M0
11.     if [#9044 != 0] the #[a] = #9044
12.     if [#9044 == 0] the #[a] = #9011
13.     M123 L1 ; TT-1 set to be input
14.     M123 q0 p#a
15.     M101/#a
```

After pressing Cycle Start, the TT1 needs to be manually triggered twice. The best way to do this is by touching an electrically conductive material to the worktable and the top of the TT1. A right-angled allen wrench about six inched long works well.

How to trigger TT1

3. Before triggering the TT1 you will notice in the message window the message "Waiting for input #15 (M101)". If the TT1 parameters are set correctly the program will stop at a M0 after triggering the TT1. If the program doesn't reach the second M0 shortly after triggering the TT1, stop the program. Press <Alt + I> to turn on Live PLC debugging and trigger the TT1 to see which input is actually toggling and then change parameter 44 to the input being tripped. If no input is tripped check the TT1 wiring.

4. The quick home switch test is now run to see if the machine can be homed reliably and that the home switches are not too close to the index pulse of the motors.

5. ATC spindle test, which is made up of three parts:

6. First parameter 78 (Spindle Speed Display) is set to 1.0 to enable live measurement and calculated spindle speed to be displayed in the CNC status window. Parameter 36 (Rigid Tapping Enable/Disable) is set to 1.0 to enable rigid tapping and parameter 33 (Spindle Motor Gear Ratio) is checked to see if it's set to 1.0. If not set to 1.0, it will be changed to a 1.0 and a warning will be logged.

7. The program turns the spindle on for a few seconds and looks for any change in the absolute encoder position of all axes. If no changes are detected, the test will error out and log the failure. In which case the spindle encoder wiring and connections need to be checked. If movement is detected, parameter 35 is verified that it's set correctly. A wrong value will result in a failure; the failure message in the systest.out file will indicate the correct value. Change P35 and run test again. A quick check for spindle direction is made, where the spindle is turned on with an M3 to see in the spindle is counting up or counting down. If counting up nothing happens, counting down will result in a minus sign being added or removed from parameter 34 (Spindle Encoder Counts/Revolution) and a warning logged.

8. The test now records the average spindle deceleration time from 2000RPM.
Then all spindle speeds from 100RPM up to the maximum spindle speed set in the Control Configuration are checked and recorded in systest.out. The spindle speeds are commanded and then the actual speeds are recorded after several seconds. The commanded speeds are compared to the actual and must be within 3% to pass. If a failure occurs check P34 is set correctly and that the inverter has been programmed for the correct maximum output frequency. To check parameter 34 turn the spindle 10 times by hand and divide the difference in encoder counts by 10. The absolute

encoder position can be read off the PID screen.

9. The ATC tapping test is a quick and easy test. It starts at machine home and taps down three inches at 1000RPM. The Z-axis movement should be very smooth, however if you notice a jerky motion it is probably due to P34 having the wrong sign (+/-). For example, if P34 is set to 4096 and you have jerky motion on Z-axis change P34 to –4096. Another problem could occur with a "Z axis cannot follow the spindle" error. This means you have the wrong number of counts in P34 (i.e. 8000 instead of 4096), although the correct value for P34 should have been determined in spindle test.

10. Machine home switch and travel limit test. First off, the way the machine home tests are conducted and analyzed depends on the values stored in the plus/minus home switch fields in the Machine Configuration – Motor screen.
There are two checks made before the test begins. The program first checks the "Machine home at pwrup" setting in the Control Configuration. If set to "Jog" the test is skipped and a warning is logged in systest.out. In most cases it should be set to "Home switch" or "Ref Marks". The second check looks for any axis that has at least one home switch input defined except for, rotary configured axes or axes with an "M", "N", or "S" label. Reference Mark homing will have both plus/minus home switches set to zero. So any axes that fail this check will generate a warning in the systest.out file.

Review of homing process when homing to a switch

1.) The axis moves toward the home switch at the slow jog rate until the home switch is tripped.
2.) The axis moves away from the switch until the home switch closes.
3.) The axis moves further away by a small amount (0.0025 inches).
4.) The axis moves in small increments away from the switch until the encoder index pulse is located.

Axes with a minus home switch defined will have multiple measurements taken of the home position (found by using an M91/ command) and the position after step (3) in the homing procedure. The measurements are logged along with error measurements and other data. To pass the test, all home position measurements must be within 0.0005 inches of the first measurement. If the test fails and the error is within 0.0010 inches of a motor revolution, the follow message will be logged "SUSPECTED LIMIT SWITCH PROBLEM". Otherwise, "SUSPECTED ENCODER INDEX PROBLEM" will be logged.

Now the distance between the home position and the off-switch position (position after step 3) are checked to ensure that they are not within 1/8 or greater than 7/8 of a motor revolution. If within 1/8 or greater than 7/8 of a motor revolution a failure occurs and a "Home position too close to encoder index pulse" message is logged. To correct this failure, move the limit switch or trip dog. If unable to move, decouple the motor and rotate motor shaft 45°.

If an axis has a plus home switch set the same measurements and analysis are performed on it.

Once all plus and minus home switch tests are complete for every axis the travel limits are checked. When there are two home switches, the overall travel defined in the Machine Configuration – Jog parameters must be less than or equal to the distance between the minus and plus home positions, but within 1.0 inch (25.4mm). For example, the distance between the plus and minus home positions is 24.135 inches, which means the travel limit must be within 23.135 – 24.135 inches to pass the test.

In the case where there is one or no home switches set for an axis, two moves are made. One from the home position out to the minus travel limit and another from the home position out to the plus travel limit. If the travel limits are too long, the test will be stopped by either a "Full power without motion" or " limit tripped" message.

After the travel limits are tested and have passed, all axes move to the center of the travel limits.

3. ATC Tool Changing Test starts by performing a series of tool changes. These tool changes are done in such a way to alternate carousel directions and maximize carousel travel. For example, a 16 – Tool setup would run a series of tool changes like T1, T8, T2, T9, T3, T10 etc., and would continue in this manner until 48 tool changes (3x the maximum number of bins) have been completed.

   After every tool change, the tool height is checked using the TT1. The first time a tool is measured, the Z-axis machine position is recorded. On subsequent tool checks, the machine position is compared to the initial recorded position. The test will fail if subsequent tool measurements are not within 0.005 inches of the initial position.

   Assuming a few tool changes have been completed, some possible causes for the wrong tool being picked up are:

   - Parameter 161 (ATC Maximum Tool Bins) is set wrong. Ex. P161 = 16 but the tool carousel holds 20 tools.
   - Not using a brake motor for the carousel or there is a faulty brake motor.
   - The state of the tool counter sensor after a tool change is not electrically open. This can be caused when the wrong type of tool counter sensor (Normally Open or Normally Closed) is matched to the mechanical setup of the counting mechanism.
   - The tool counter sensor or wiring is faulty.
   - Electrical noise.

Completed System Test

If the control runs through the complete test and passes everything the systest.out file will say passed and the "Machine setup not completed" message will no longer appear at machine power up.

Note: If you run the system test after it's already passed, the systest.out file will be wiped out and the control will need to be run through the complete test again.

Machine run in:

- ☐ Run a test G code program for a minimum of 48 hours that will exercises all machine functions.

Chapter 4

Troubleshooting and Diagnostics

CNC10 Errors Messages

These error messages are seen in text mode after CNC7 has tried to start or is exiting. Error initializing CPU7/10...

1. CNC10.PLC file read error...
2. Return code 63
3. Return code 64

CNC7 Abnormal Stops – Faults Error Messages

Abnormal stops are detected in the following order: PLC, servo drive, spindle drive, lube, ESTOP. This means that if both the servo drive and the spindle drive have faulted, the servo drive fault message would appear.

1. 401: PLC failure detected
2. 404: Spindle drive fault detected
3. 405: Lubricant level low
4. 407: limit () tripped
5. 409: _ axis lag
6. 410: _ axis position error
7. 411: _ axis full power without motion
8. 412: _ axis encoder connection is bad
9. 413: CPU Failure #01: power down
10. 414: CPU Failure #02: power down
11. 420: _ axis motor overheating
12. 421: Motor(s) too hot: job canceled
13. 422: Jog Panel Offline
14. 424: Feedrate Override Offline
15. 426: Spindle Override Offline
16. 428: MPG Offline
17. 430: CPU7 PIC Offline
18. 432: External PLC Offline
19. 434: _ axis idling too high: Releasing power
20. 435: _ axis runaway: Check motor wiring
21. 436: Servo drive shutdown
22. 439: _ axis servo drive processor failure
23. 441: _ axis over voltage *AC brushless systems only.*
24. 442: _ axis under voltage *AC brushless systems only.*
25. 443: _ axis commutation encoder bad *AC brushless systems only.*
26. 444: _ axis over temperature detected *AC brushless systems only.*
27. 445: _ axis over current detected *AC brushless systems only.*
28. 446: _ axis synchronization failure *AC brushless systems only.*

Problems that do not generate an Error message:

1. Spindle does not come on.
2. Hit a limit and I can't jog off it
3. Axis jogs the wrong way
4. Machine loosing position
5. Machine won't home out
6. Won't cut a round circle
7. Leaves marks on circles
8. Noisy motors
9. Coolant not coming on
10. Control not booting (does not run autoexec.bat)
11. Keyboard or computer lockups
12. Machine not homing in correct direction
13. Spindle running in wrong direction
14. Tool carousel turning in wrong direction.
15. Rigid Taping does not work

Error initializing CPU7...

Hex file CPU10 error identifier for return code 63

Symptoms	Possible Problem	Troubleshoot / Cure
See return code 63		*If the error was in loading the indicated file the line number that it was trying to load will be displayed*

CNC7.PLC file read error…

PLC program error identifier for return code 63

Symptoms	Possible Problem	Troubleshoot / Cure
See return code 63		

Return code 63

Exit code from CNC7 because of an error during startup.

Symptoms	Possible Problem	Troubleshoot / Cure
CPU10 error loading file	❑ Bad or damaged hex file being sent to CPU10 ❑ CPU10 not seated properly in PCI computer slot. ❑ Bad or damaged PLC program (CNC10.PLC) being sent to CPU10 ❑ Hard drive problems or other hardware problems	Replace bad file ❑ DC Brush systems use CNC8.HEX ❑ AC Brushless systems use CNC9.HEX Reinstall PLC program(s) (CNC10.PLC) Reseat CPU7/9 in computer
Missing file	CNC8, CNC9 hex file or PLC program (CNC7.PLC) not found while trying to load.	Replace missing file ❑ DC Brush systems use CNC8.HEX ❑ AC Brushless systems use CNC9.HEX Install PLC program(s)

Return code 64

Exit code from control software to indicate an invalid math function was trying to be performed.

Symptoms	Possible Problem	Troubleshoot / Cure
File read error	❑ An invalid value was stored in one of the control configuration files (corrupt file). ❑ Bad hardware (Hard drive, memory, motherboard, ect.)	Backup and then delete one at a time the following files ❑ CNC7.TEM ❑ CNC7.JOB ❑ CNC7.WCS

401: PLC Failure Detected

Input 16 and output 16 are used as a check to ensure PLC operation. Output 16 is set by Input 16, so if input 16 is not correct a "**401: PLC Failure Detected**" error is generated. **Press E-stop to clear**.

Symptoms	Possible Problem	Troubleshoot / Cure
Missing Logic power	No 110VAC to logic PS No 5VDC, 12VDC to PLC	Check voltages ❑ See figure 3 for Servo3IO ❑ See figure 6 for H63 on PLC 15/15 ❑ See figure 7 for PLCIO2 ❑ See figure 8 for RTK2 ❑ See figure 9 for PLC 3/3
Fibers optic cables	Fiber cables plugged in wrong, broken or not plugged in at all.	Check Fiber cables and connections ❑ See figure 1/2 for Servo3IO ❑ See figure 6 for PLC 15/15 ❑ See figure 7 for PLCIO2 ❑ See figure 8 for RTK2 ❑ See figure 9 for PLC 3/3
PLC program	PLC program set output 16	Verify that the PLC program is not using output 16.

404: Spindle Drive fault detected

If the PLC receives a fault signal from the spindle inverter or overload contactor output 65 is set and a "**404: Spindle drive fault detected**" error is generated. **Press E-stop to clear**.

Symptoms	Possible Problem	Troubleshoot / Cure
Wrong inverter settings	Inverter has not been setup, or was not programmed properly.	Check inverter manual for proper settings
PLC program	The inverter's spindle drive fault out is opposite of what the plc program is expecting.	Switch fault output to other NO or NC fault contacts if possible. Or Modify the PLC program
Too heavy of a cut	Dull tool or programmed spindle speed or feedrate is too high	Replace tool Or Slow down feedrate and/or spindle speed
Spindle motor	Spindle Motor is miss wired, motor winding are shorted	Replace motor if bad Or See schematic for proper wiring connections.
Contactor overload protector tripped (non inverter systems)	❑ Dull tool or ❑ Programmed spindle speed or feedrate is too high ❑ Overload set too low ❑ Spindle motor is bad ❑ Spindle motor is too big for contactor	❑ Replace tool ❑ Slow down feedrate and/or spindle speed ❑ Check overload setting ❑ Verify overload rating for motor size ❑ Replace spindle motor

405: Lubricant level low

If the PLC receives a lube low signal from the lube pump output 63 is set and a "**405: Lubricant level low**" message is generated. **Press E-stop to clear**.

Symptoms	Possible Problem	Troubleshoot / Cure
Lube low alarm	❑ Ran out of lube oil ❑ Lube switch wired NO, PLC program looking for NC ❑ Low lube switch bad	❑ Fill lube pump. ❑ Wire low lube switch to NC contacts. Default for PLC program is NC. ❑ Check lube switch with ohm meter

407: __ Limit (#__) tripped

The limit tripped message is generated when one of the PLC inputs set in the motor configuration menu as a limit switch is tripped. The PLC input that was tripped is shown in the parentheses.

Symptoms	Possible Problem	Troubleshoot / Cure
Constant limit tripped message	An axis has hit a limit switch ❑ Program exceeds the travel limits, or the part zero was set incorrect. ❑ Soft travel limits not set properly Broken limit wire ❑ Limit switch wires were secured to tightly and could not move with the axis to the full travel and were damaged	Use the slow jog button to move the axis away from the end. ❑ Reset part zero to a point which will permit the CNC program move. ❑ Set soft travel limits in machine configuration menu. ❑ Ensure all limit wires have enough length to move the entire axis travel.
Axis does not home. Limit tripped message appears in message window when the home / limit switch is reached.	Home inputs are not set correctly or not set at all.	Set input numbers in the "Home –" and "Home +" columns in the "Machine Configuration" screen to match the "Limit –" and "Limit +" columns.

409 : __ axis lag

If the lag distance (allowable following error) is exceeded for more than 1.5 seconds a **"409: __ axis lag"** error occurs.

Symptoms	Possible Problem	Troubleshoot / Cure
The machine is doing a very heavy cut	❑ High drag ❑ Chip buildup ❑ The motors are undersized for the application ❑ The maximum rate or acceleration parameters are set too high.	❑ Look for and clear obstructions ❑ Install higher power motors, this may mean installing a higher power drive also ❑ Run Autotune

410: __ Position Error

When the CPU7 (motion control card) is expecting the motor to turn, and not getting a correct encoder response for >.25", a **"__ 410: Position error"** results.

Symptoms	Possible Problem	Troubleshoot / Cure
Max. rate in the parameters is set too high	The motor may not be capable of maintaining the maximum rate.	❑ Run Autotune. ❑ Try commanding slower speeds to check for a rate-related position error.
Binding or mechanical problems such as excessive lash	The CPU7 board has detected >.25" error on an axis	❑ Run drag plot to help find bind points in axis travel. ❑ Slow down the feedrate and watch for errors. ❑ Un-belt or uncouple the motor and watch for errors.
Bad encoder	Coolant, fluid, or dirt penetrating the encoder housing seal causing the encoder to give erroneous encoder counts to the CPU7 card	❑ Replace motor with a known good motor ❑ Swap motor from another axes to verify bad encoder.
Bad encoder cable	Broken wire, or connection at the pins of the cable	Move the suspected encoder cables to another axis's motor. ❑ See figure 3: Servo3I/O drive ❑ See figure 4: Servo1 drive ❑ See figure 5: QUAD drive *NOTE: You must swap motor and encoder cables together*
Encoder connectors are on the wrong axis	One motor is turning but the position displays on another axis	Check for proper encoder hookup. ❑ See figure 3: Servo3I/O drive ❑ See figure 4: Servo1 drive ❑ See figure 5: QUAD drive
Motor power wiring is backwards	The CPU7 board is getting backwards encoder counts	Watch the DRO display. If the DRO is always increasing or decreasing regardless of the direction jogged, the motor wires at the drive are reversed. ❑ See figure 3: Servo3I/O drive ❑ See figure 4: Servo1 drive ❑ See figure 5: QUAD drive
Bad CPU7 board	The CPU7 board is losing encoder counts and giving false DRO reading (very rare!)	Replace CPU7

411: __ Full power without motion

When 90% power (PID) is applied to an axis and no motion is detected for the time specified in parameter 61 (default is 0.5 sec.), a **"411: __ Full power without motion"** error occurs.

Symptoms	Possible Problem	Troubleshoot / Cure

Servo drive logic power missing DC brush system	Blown fuse, bad connection or no connection. No 110VAC to logic PS	Check voltages going to the drive SERVO3IO see figure 3 ❑ Check H2 or H3 for 5VDC and 12VDC SERVO1 see figure 4 ❑ Check H8 for 5VDC, +12VDC and –12VDC. Quad drive see figure 5 Check H1 for 5VDC, +12VDC and –12VDC See schematic for proper wiring connections.
Servo drive power missing DC brush system	❑ Blown fuse, bad connection or no connection. ❑ Flood or spindle overload contactor tripped. ❑ E-Stop circuit miss wired or DC power supply not working.	Check voltages going to the drive SERVO3IO see figure 3 ❑ Check H16 for ~110VDC SERVO1 see figure 4 ❑ Check PIN 9 and 10 of H4 for ~110VDC Quad drive see figure 5 ❑ Check PIN 9 and 10 of H4 for ~110VDC See schematic for proper wiring connections.
Axis is against a physical stop.	Program exceeds the mechanical travel limits, or the part zero was set incorrect.	Use the slow jog button to move the axis away from the end. ❑ Reset part zero to a point which will permit the CNC program move. ❑ Set soft travel limits in machine configuration menu. ❑ Check limit switches for proper operation if installed.
Binding or mechanical problems	❑ Worn bearings or ballscrew ❑ Malfunctioning lube system ❑ Ballscrew was improperly aligned with the axis travels	❑ Check lube system for lube oil and plugged lube lines. ❑ Check bearings for excessive play. ❑ Run drag plot to help find bind points in axis travel.
Loose Encoder	Set screws loose that hold encoder onto motor shaft.	❑ Tighten encoder set screws

Fiber Optic DC brush systems	Fiber Optic cables were plugged in wrong, broken or not plugged in.	For fiber optic connections: ❑ See figure 1 for M39 SERVO3 IO ❑ See figure 2 for Uniconsole SERVO3IO ❑ See figure 4 for Servo1 ❑ See figure 5 for Quad drive See schematic for proper wiring connections.
Limit Switch problem on DC brush systems	Broken limit switch or wire and the limit switch inputs are not entered into machine configuration menu.	❑ Check limit switches for proper operation and wire continuity. ❑ Set PLC limit inputs in machine configuration menu to see limit tripped message.
Servo motor not getting power from servo drive.	Motor not connected, bad cable or connection.	❑ Check motor connections and cables. See schematic for proper wiring connections.
Servo motor bad	Servo motor is shorted, coolant inside motor, motor was undersized and burned up.	❑ Replace servo motor
Servo drive bad	Servo motor shorted, motor cable shorted.	❑ Replace servo drive

412 : __ axis encoder connection bad

If an axis is enabled and CPU7/9 does not detect a differential encoder signal then a **"412: __ axis encoder connection bad"** error is generated.

Symptoms	Possible Problem	Troubleshoot / Cure
Loose or severed encoder cable	Broken wire, or connection at the pins of the cable. Hanging cable was caught during machine operation and broken or pulled.	Move the suspected encoder cables to another axis's motor. ❑ See figure 3: Servo3I/O drive ❑ See figure 4: Servo1 drive ❑ See figure 5: QUAD drive *NOTE: You must swap motor and encoder cables together*
Bad encoder	❑ Coolant or lube oil got into motor encoder housing. ❑ Incorrect type encoder	❑ Replace encoder ❑ Use quad differential type encoder, 2000 line recommended

413 : CPU Failure #01: power down

A time out occurred sending a command to the Z80 (64180) processor on the CPU7

Symptoms	Possible Problem	Troubleshoot / Cure
CPU7 not seated properly.	CPU7 card not screwed into computer ISA slot and came loose in operation.	Reseat and securely fasten CPU7 into computer ISA slot.

Z80 Processor failure	Exposed CPU7 shorted chips or coolant.	Replace CPU7 and keep all doors closed and covers on control.

414 : CPU7 Failure #02: power down

Communication with the DSP (2101/2105) on the CPU7 has been lost.

Symptoms	Possible Problem	Troubleshoot / Cure
Encoder index pulses flooding CPU7	Bad or noisy index pulse on one of the axis motors	Disable and then disconnect each axis motor one at a time. See which axis is causing the problem. ❑ Check for proper wiring and grounding on the encoder cable. ❑ Replace encoder
DSP failure	Exposed CPU7 shorted chips or coolant.	Replace CPU7 and keep all doors closed and covers on control.

420 : __ axis motor overheating

CNC7 estimates that the indicated axis motor has reached the warning temperature set in parameter #29

Symptoms	Possible Problem	Troubleshoot / Cure
Tight or sticky machine	Poor lube system, or mechanical adjustment	❑ Check lube system for proper operation ❑ Have a qualified machine mechanic adjust your machine gibbs.
Unbalance Z axis	The Z axis counter balance does not match the weight of the Z axis head closely enough	Add or remove weight from the counter balance to match head weight more closely.

421: Motor(s) too hot: job canceled

CNC7 estimates that an axis motor has reached the limit temperature set in parameter #30.

Symptoms	Possible Problem	Troubleshoot / Cure
Tight or sticky machine	Poor lube system, or mechanical adjustment	❑ Check lube system for proper operation ❑ Have a qualified machine mechanic adjust your machine gibbs.
Unbalance Z axis	The Z axis counter balance does not match the weight of the Z axis head closely enough	Add or remove weight from the counter balance to match head weight more closely.

422 :Jog Panel Offline

CPU7/9 has lost communication with the jog panel.

Symptoms	Possible Problem	Troubleshoot / Cure
Jog cable and/or connectors loose or broken.	Loose or dangling cables pulled loose or broken by machine.	❑ Retighten connector screws to hold cable connectors in place ❑ Replace any damaged jog panel cables.
Job panel dead	Coolant shorted jog panel Severed cable shorted jog panel	Replace jog panel

428: MPG Offline

The MPG button has been turned on and the CPU7/9 can not communicate with the MPG hand wheel.

Symptoms	Possible Problem	Troubleshoot / Cure
MPG button pressed on jog panel	❑ Forgot to plug in the MGP ❑ Bad MGP cable/connector.	❑ Plug in the MPG ❑ Replace MGP

432: External PLC Offline

If the CPU7/9 has lost communication with the Koyo (PLC Direct) PLC then a “432: External PLC offline” error is generated.

Symptoms	Possible Problem	Troubleshoot / Cure
Duel PLC selected in the control configuration menu	Wrong PLC type selected in menu.	Select proper PLC type in control configuration menu.
Koyo PLC lost power	Blown fuse or broken wire	Check Koyo PLC power. See schematic for proper wiring connections.

434: __idling too high: releasing power

The indicated axis is stopped with power on, and an unusually high PID output is required to hold position.

Symptoms	Possible Problem	Troubleshoot / Cure
Tight or sticky machine	Poor lube system, or mechanical adjustment	❑ Check lube system for proper operation ❑ Have a qualified machine mechanic adjust your machine gibbs.

Unbalance Z axis	The Z axis counter balance does not match the weight of the Z axis head closely enough	Add or remove weight from the counter balance to match head weight more closely.

435: __ axis runaway: check motor wiring

The indicated axis was moving more than 120 in/min (3048mm/min) while power was supposed to be off.

Symptoms	Possible Problem	Troubleshoot / Cure
Motor power wiring is backwards	The CPU7 board is getting backwards encoder counts	Watch the DRO display. If the DRO is always increasing or decreasing regardless of the direction jogged, the motor wires at the drive are reversed. ❑ See figure 3: Servo3I/O drive ❑ See figure 4: Servo1 drive ❑ See figure 5: QUAD drive
E-stop was pressed while doing a rapid	Max rates and and/or Accel value set too.	Run autotune
Noisy encoder or cable	Non-shielded cable being used Cable shield not grounded properly	❑ Check for proper wiring and grounding on the encoder cable. ❑ Replace encoder

436: Servo drive shutdown

On DC brush systems, the servo drive has detected an error. This error generates a signal from the servo drive to the PLC which will set PLC output bit 64. **Press E-stop to clear**.

Symptoms	Possible Problem	Troubleshoot / Cure
Servo drive detected an over-current condition	Sudden hard stop ❑ Ran into machine physical hard stop ❑ Z axis rapid into stock or table Motor or drive FETs shorted from ❑ Motor contaminated by coolant or lube oil ❑ Drive blown by bad motor ❑ Drive blown by chips or coolant because it is exposed or power cabinet door is open while running	Check and/or verify ❑ Limit switches ❑ Soft travel limits ❑ Part Zero (use backplot) ❑ Part Program (use backplot) Replace motor and/or drive
Servo drive detected an over-voltage condition	Input power spikes	For persistent over-voltage problem input power should be monitored by your local electric company

439: __axis servo drive processor failure

The servo 4 drive logic power failed or the processor failed.

Symptoms	Possible Problem	Troubleshoot / Cure
Fibers optic cables	Fiber cables plugged in wrong, broken or not plugged in at all.	Visually check that the fiber cables are not broken and are plugged in. ❑ See Figure 10 for Drive to CPU9 ❑ See figure 11 for Drive to Uniconsole panel
No V bias	Broken wire, blown fuse or power supply not connected.	See schematic for proper wiring connections.
CNC9.HEX running on a DC brush/CPU7 system	Update or install put a CNC9.HEX file on the system	If and only if this is a DC BRUSH/CPU7 system ❑ Delete CNC9.HEX from the hard drive.

441: __ axis over-voltage

AC brushless systems only. Drive input power was detected to be greater than 350VDC for more than 2. seconds

Symptoms	Possible Problem	Troubleshoot / Cure
Input power spikes	Lightning strikes or other weather related problems	For persistent over-voltage problems input power should be monitored by your local electric company
Wired wrong	Replacement drive was not hooked up the same as original	See schematic for proper wiring connections.

442: __ axis under-voltage

AC brushless systems only. Drive input power was detected to be less than 80VDC for more than 2.

Symptoms	Possible Problem	Troubleshoot / Cure
Input power	Brownouts Weather related problems	For persistent under-voltage problems input power should be monitored by your local electric company

443: __ commutation encoder bad

AC brushless systems only. Bad connection from encoder.

Symptoms	Possible Problem	Troubleshoot / Cure
U,V,W lines from encoder returned a value not in the range of 1 – 6	❑ Encoder wiring error ❑ Bad encoder	Check encoder wiring Replace encoder

444: __ axis over-temperature detected

AC brushless systems only. Sensor on heat sink detected a temp greater than 160F (71C)

Symptoms	Possible Problem	Troubleshoot / Cure
Tight or sticky machine	Poor lube system, or mechanical adjustment	❑ Check lube system for proper operation ❑ Have a qualified machine mechanic adjust your machine gibbs.

Unbalance Z axis	The Z axis counter balance does not match the weight of the Z axis head closely enough	Add or remove weight from the counter balance to match head weight more closely.

445: __ axis over-current detected

AC brushless systems only. An over-current was detected by the drive on the indicated axis.

Symptoms	Possible Problem	Troubleshoot / Cure
Sudden hard stop	❑ Ran into machine physical hard stop ❑ Z axis rapid into stock or table	Check and/or verify ❑ Limit switches ❑ Soft travel limits ❑ Part Zero (use backplot) ❑ Part Program (use backplot)
Motor shorted		Replace motor

446: __ axis synchronous I/O error

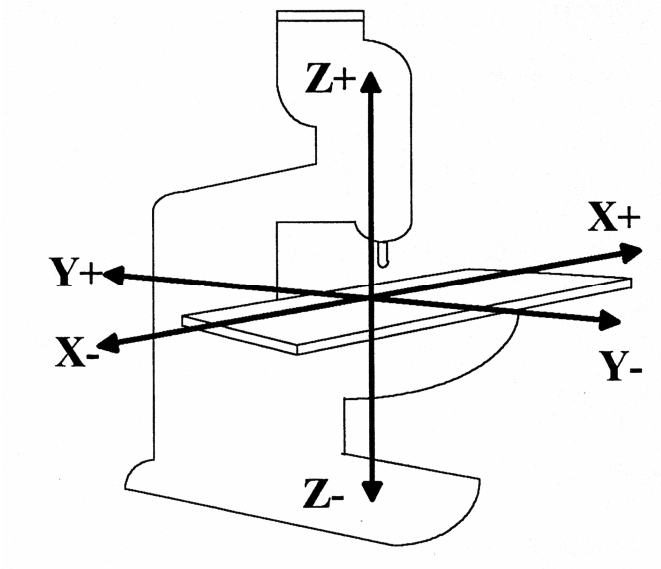
AC brushless systems only. A communication checksum error was detected between the CPU9 and drive.

Symptoms	Possible Problem	Troubleshoot / Cure
Fiber optics cables	Cables too long Fiber end not fully polished or dirty	Shorten cables Polish and/or clean fiber ends.

NON-Error message faults

Symptoms	Possible Problem	Troubleshoot / Cure
Spindle does start	A limit switch is tripped or a limit switch wire is broken and the limit inputs are set to 0 in the motor parameters	❑ Set limit switch PLC numbers in motor configuration menu to get proper message. ❑ Clear the limit switch.
Spindle starts in manual mode, but not in automatic mode (M3/M4)	Check for Hi/Low spindle range in options menu in the Utilities.	❑ Enter unlock to enable option
Hit a limit and I can't jog off it	Fast jog selected	❑ **Slow** jog in opposite direction of tripped limit to get off
Axis will not move in slow, continuous jog in either direction when limit is tripped (motor may jump slightly).	Drive limits do not match software limits	❑ If software limits are set correctly, swap hardware limit wiring for – and +. Then change software limit input numbers to compensate for the wiring change.
Axis jogs toward limit but not away (stuck on limit switch). Limit messages are correct in the message window.	Limit switch connections to the drive are reversed.	❑ Swap hardware limit wiring for – and +. Then change software limit input numbers to compensate for the wiring change.
Axis jogs the wrong way	Direction reversal in the motor parameters not set properly.	❑ See figure 1 for axis movement conventions ❑ Set Direction reversal in motor parameters for proper direction
Machine loosing position	Encoder slipping on motor shaft. Machine is not grounded properly and noise is getting into encoder cables. Using unshielded encoder cables. Encoder loosing counts (bad encoder) Pulley slipping on motor shaft or ball screw.	❑ Check set screws on encoder and tighten if needed ❑ Make sure machine is grounded ❑ Replace unshielded encoder cables with shielded ones. ❑ Replace encoder ❑ Check that pulleys are keys and set screws tight
Machine won't home out	Not all axis being homed in CNC7.HOM.	❑ Add missing axis in CNC7.HOM
Won't cut a round circle or Leaves marks on circles	Turns ratios set wrong Improper lash adjustment Improper gibb adjustment Worn bearings or ballscrew	❑ See chapter 2 Motor Parameters for setting turns ratio ❑ See chapter 2 Motor Parameters for measuring backlash ❑ Check for excessive play in bearings ❑ Have a qualified machine mechanic adjust your machine gibbs.
Noisy motors	PID parameters modified Bad motor	❑ See chapter 2 PID Configuration for correct PID settings

Control not booting	No power to control PC	❑ See schematic for proper wiring connections.
Keyboard or computer lockups	Machine not grounded Loose keyboard cable	❑ Make sure machine is grounded
Machine not homing in correct direction	CNC7.HOM file Direction Reversal	❑ Switch M91/M92 to home in opposite direction ❑ See figure 1 for axis movement conventions ❑ Set Direction reversal in motor parameters for proper direction
Spindle running in wrong direction	Spindle motor hookup	❑ Reverse 2 of the 3 power leads to the spindle motor
Tool carousel turning in wrong direction	Carousel motor hookup	❑ Reverse 2 of the 3 power leads to the spindle motor
Limit tripped message does not match limit being hit.	Limit switch inputs are not set correctly in the Machine Configuration screen	❑ Correct input numbers in the “Limit –“ and “Limit +” columns in the “Machine Configuration” screen.
Rigid tapping does not work	No spindle encoder input No index pulse from the spindle encoder	❑ See chapter 3 Encoder inputs. ❑



The CUTTER, not the TABLE, defines the direction of motion.

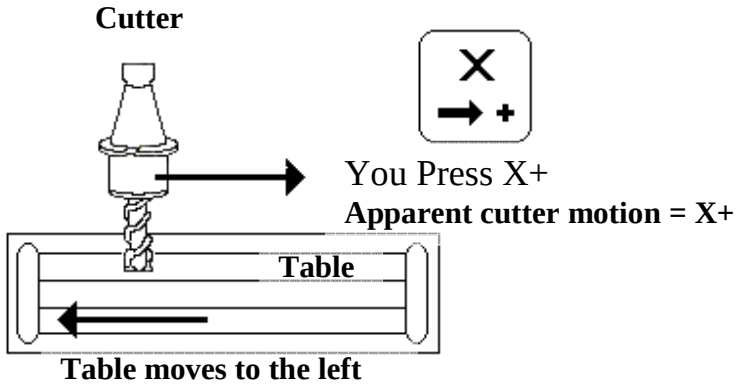


Figure 1: Direction conventions

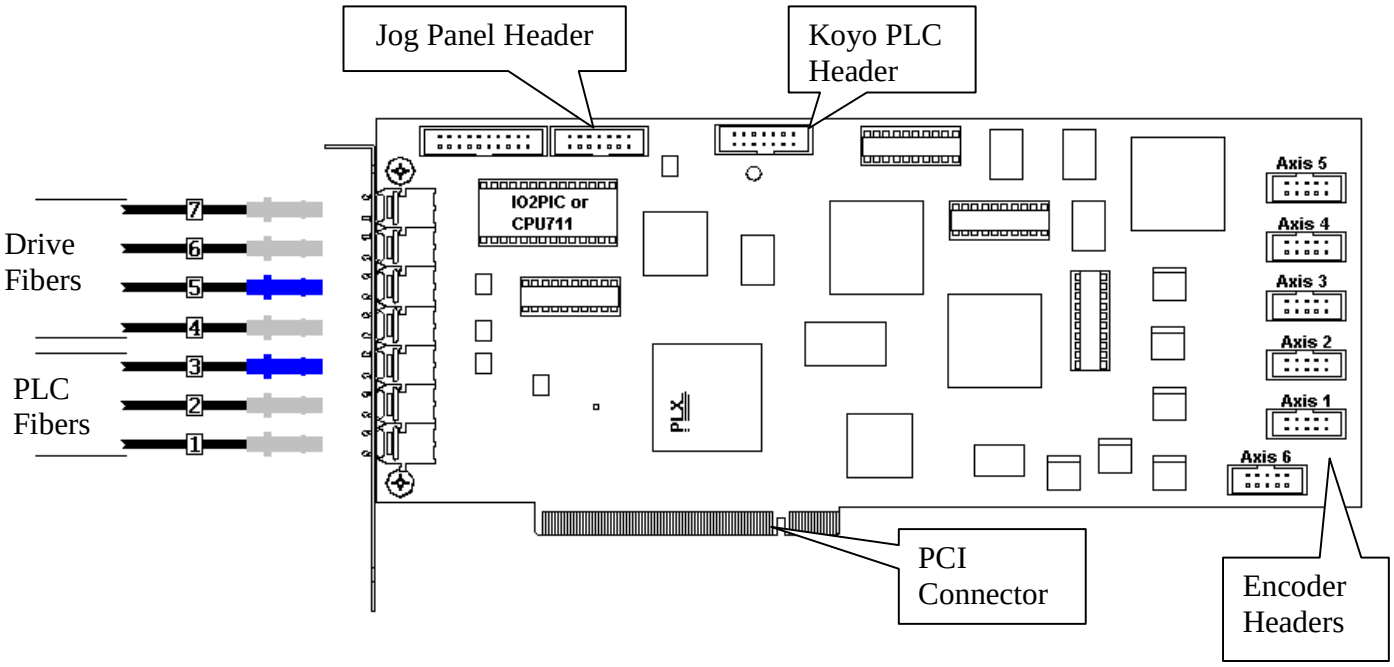


Figure 2 CPU10 Connections

Chapter 5

PLC Programming Manual

Introduction

This manual is intended for machine builders, technicians, dealers, and others who need to "look under the hood" of the PLC system.

The PLC (Programmable Logic Controller) is the portion of the control system, which deals with accessory equipment; that is, anything installed on the machine beyond the servo motors themselves. Spindle control, coolant control, limit switches, fault signals, pushbuttons, indicator lights, etc. are all controlled through the PLC.

The PLC system has two possible configurations. The first configuration consists of a hardware I/O board which provides the electrical interface to all the switches and relays (RTK3, PLCIO2, RTH2, PLC15/15, ect.), the jog panel with its keys and LEDs, and a software logic program which is executed on the CPU10 controller board (CNC10.PLC Program).

In the first configuration (see figure 1) the PLC program, which is executed on the CPU10, is contained in the file CNC10.PLC. The control software sends CNC10.PLC to the CPU10 on startup. Once running on the CPU10, the logic program receives inputs from the PLC I/O board, from the jog panel, and from CNC programs. It sends outputs back to all three places: for example, turning on output relays on the I/O board; turning on LEDs on the jog panel; and sending signals such as Feed Hold to the CNC processor.

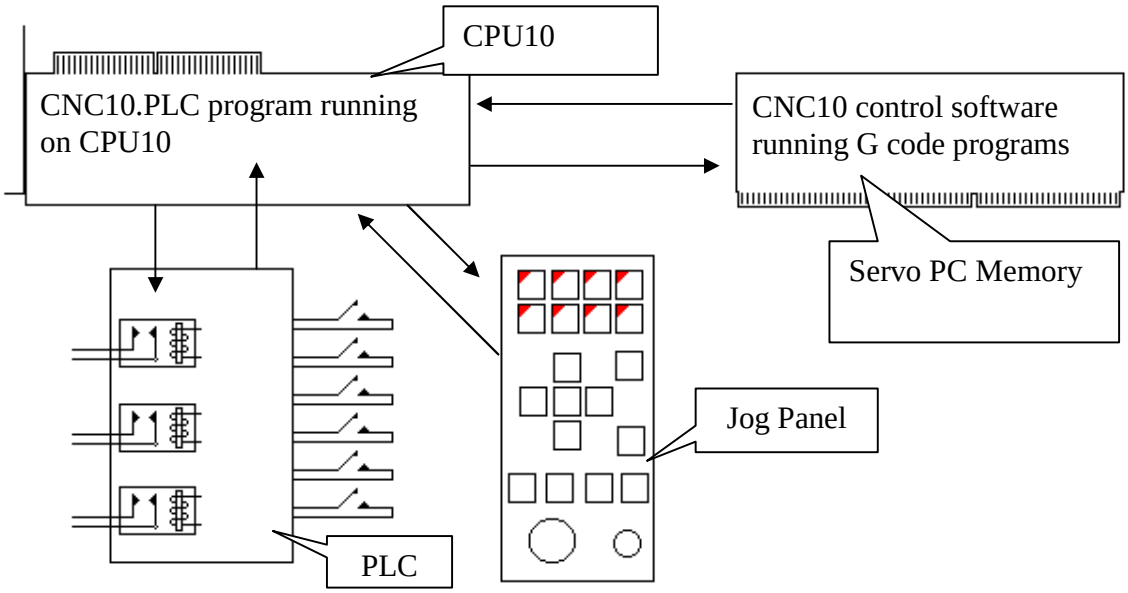


Figure1: PLC I/O Configuration 1

The first PLC configuration, using the CNC10.PLC program only, is limited in size and functionality and cannot control complex machine hardware such as a tool changer, which may

require timers, counters, ect. For machines that need more PLC program functionality a second PLC configuration is used. The second configuration consists of all elements of the first configuration plus a second PLC program (PC.PLC) running in the servo PC memory. CNC10.PLC program is still running on the CPU10 and is used in conjunction with the PC.PLC. The reason we need two PLC programs running is the new PC.PLC program doesn't have access to all of the I/O so the CNC10.PLC program needs to be there to control those I/O bits. A minimal CNC10.PLC program can be made by echoing memory bits to the outputs you need to control. Then you can control those memory bits in the PC.PLC program thus making it possible to control virtually everything needed in the PC PLC program.

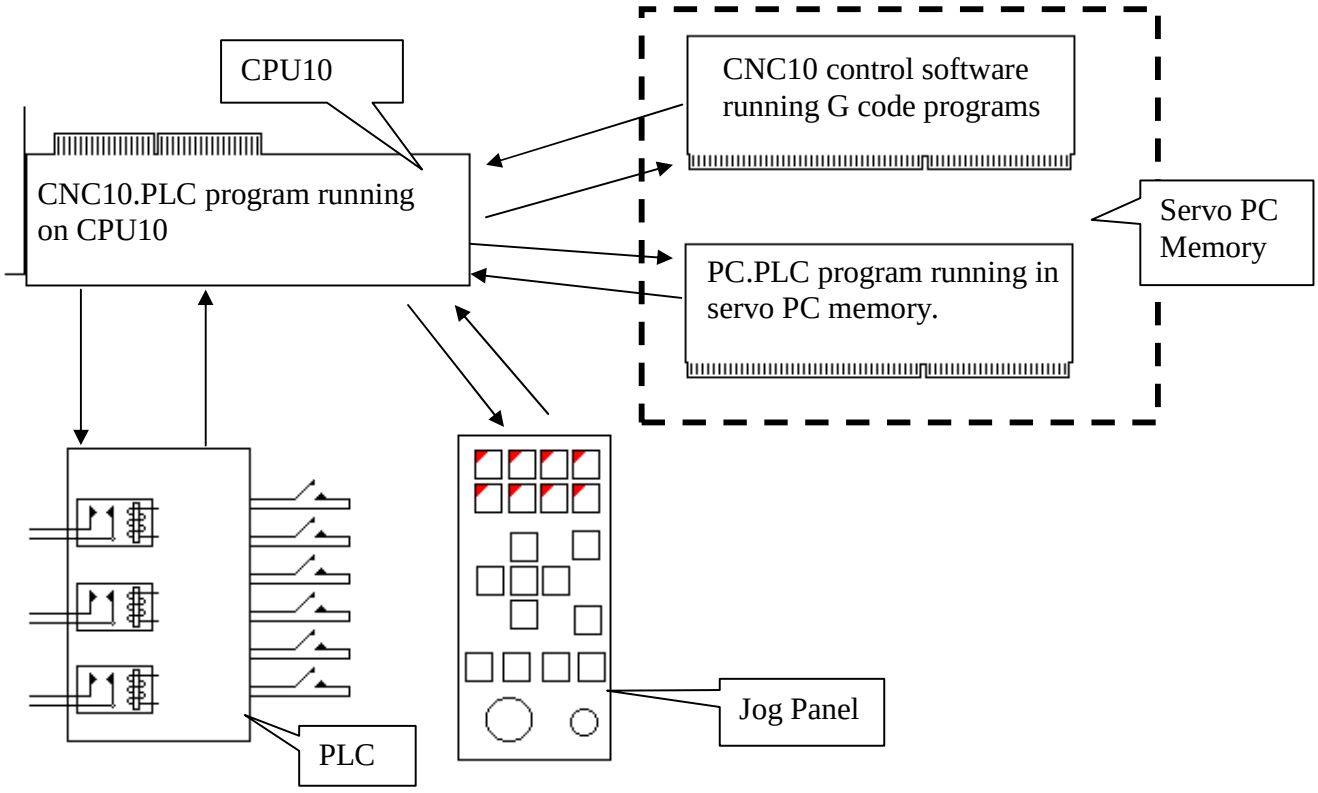


Figure 2: PLC I/O Configuration 2

While the PLC program has access to many features of the jog panel, it has no access to the jogging controls themselves. The axis jog buttons, Fast/Slow and Inc/Cont buttons, the jog increment buttons, and the MPG handwheel are all directly handled by code on the CPU7.

The first section of the PLC manual will explain programming for the CNC10.PLC program, then move into more complex issues and the PC.PLC or extended PLC programming.

Language

The PLC programming language is a simple statement-based logic language. At first it may seem far removed from the ladder logic diagrams familiar from other PLC systems. However, each assignment statement in a PLC program is equivalent to one rung of a PLC ladder. One or more inputs are logically combined to produce one output.

The PLC supports four logical operators: AND, OR, XOR, and NOT. They may be abbreviated using the symbols & (AND), | (OR), and / (NOT). There is no abbreviation for XOR. Parentheses may be used for logical grouping. In the absence of parentheses, the operators follow normal mathematical precedence: NOT, AND, XOR, OR.

The "=" sign is used to store a logical result (an input of combination of inputs) in an output location.
These operators are used to manipulate 240 data bits: 80 inputs, 80 outputs, and 80 memory locations. Many of the inputs and outputs are permanently mapped to particular hardware devices. The memory locations are not mapped to any hardware, but some are reserved for special purposes.

The data bits are given default names such as INP1, INP2, INP80, OUT1, OUT41, MEM27, etc..

Typically the first part of a PLC program file assigns alternate names to these default tokens. For example, the line

```
Emergency_stop  IS INP11
```

tells the PLC program compiler that the name Emergency_stop is equivalent to INP11. You can then use the more intuitive name for the remainder of the program.

It is important to distinguish between "IS", which assigns alternate names, and "=", which actually stores computed bit values. A line using "IS" doesn't actually do anything in the PLC program; it simply provides information to the compiler about the names you will be using. A line using "=" has a real effect each time the program runs.

Let us look at a simple latched input structure: suppose that we have a relay connected to OUT1 which controls a work light. We want the Aux1 key on the jog panel to turn the light on, and the Aux2 key to turn it off. In the definitions sections of the program we would have:

```
Aux_1_key IS INP49
Aux_2_key IS INP50
```

and

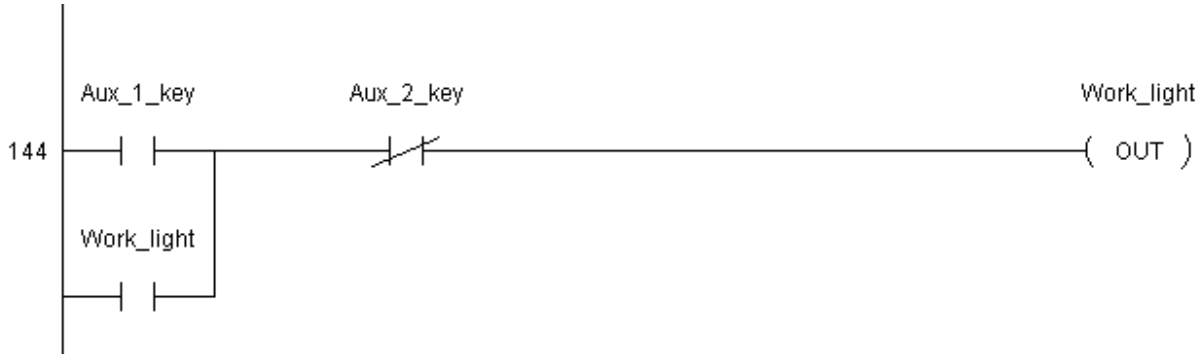
```
Work_light IS OUT1
```

In the statement section which follows we would have

```
Work_light = ( Work_light OR Aux_1_key ) AND / Aux_2_key
```

This statement says that the relay will be on if it was already on or if the Aux1 key is pressed, as long as the Aux2 key is not pressed.

In traditional ladder logic this would appear as follows:



The PLC compiler is not case sensitive. However, it is a good idea to write your PLC programs using consistent capitalization. This will make them easier to read and easier to search.

The parser is quite basic. Tokens (names, operators, and parentheses) are separated by white space (spaces, tabs, or line breaks). Names may contain almost any character. However, your programs will be more readable if you stick to alphanumeric characters and the underscore character. For example, the compiler will permit a definition like:

```
High/Low_range IS INP64
```

but the slash character (/) in the name is distracting and confusing.

All functional statements are assignments, using the '=' sign. The destination bit (e.g. an output coil) appears on the left side of the '=' sign, in the first column of the program source file. If a statement is too long to conveniently fit on one line of the program file, you can break it across multiple lines by indenting each subsequent line from the first column (e.g. by starting the line with spaces or tabs):

```
Spin_stop          = Spindle_stop_key OR
                           Stop OR Limit_tripped OR
                     ( Auto_spin_mode AND / AutoStart )
```

Comments may be included in the program source file, set off by the ';' character. The PLC compiler will ignore any text which follows the ';'. Comments are useful for explaining the intent of your PLC logic and for separating different sections of the program for readability. Comments may be on lines of their own, or may be added to the end of PLC program lines.

Any bit -- whether physically an input, an output, or a memory location -- may be assigned with the "=" operator. You are not limited to just writing to outputs.

PLC program source files are translated into the control's internal format using the PLC compiler utility, PLCCOMP. PLCCOMP and related tools are located in the C:\PLC directory on the control's hard drive.
Typically all PLC programming is done at the MS-DOS prompt. The control software, probably wisely, does not provide any readily accessible, user friendly screen to allow you to change the PLC programming.

Tutorial Example

Suppose you have an M400 control with an inverter drive for the spindle, and a spindle air brake. The default PLC logic will apply the air brake the instant the spindle-run relay is switched off, in spite of the fact that the inverter needs several seconds to decelerate the spindle. This leads to a situation where the inverter is driving the spindle motor forward against the brake, in order to maintain the programmed deceleration ramp.

One way to resolve this problem is to reprogram the PLC so that the brake is disabled whenever the spindle is turned on. The operator would then have to press the Brake key on the jog panel after the spindle has stopped if he wants to apply the brake.

<Ctrl-Alt-X>
C:\CNC10> **CD \PLC**
C:\PLC> **COPY M400.SRC K6789.SRC**
C:\PLC> **EDIT K6789.SRC**

At the beginning of the file we find a block of comments similar to the following:

```
; * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; * File:    m400.src
 .
 .
 .
; * Modifications:
; * 09/10/97 KSD Added Vector drive support by echoing
; *              SpindleRelay and CCW_relay to
; *              MEM78        and MEM79
; * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

Although it is not strictly necessary, it is a very good idea to update these comments to reflect the changes we are making:

```
; * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; * File:    K6789.src
 .
 .
 .
```

```
; * Modifications:
; * 09/10/97 KSD Added Vector drive support by echoing
; *              SpindleRelay and CCW_relay to
; *              MEM78        and MEM79
; * 07/25/01 MBL Revised brake logic to disable brake
; *              mode whenever spindle runs.
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;
```

Now page down (or search for “Brake”) to find the section of the program which controls the brake mode and output:

```
;
; Select Brake Mode
;
Brake_key_hit       = Brake_key AND / Last_brake_key
Last_brake_key      = Brake_key
      Brake_mode          = ( Brake_mode XOR Brake_key_hit )
                              OR / Already_run
;
; Turn on the brake if spindle not running and in auto mode
;
Brake               = Brake_mode AND / Spindlerelay
Spindle_brake_LED   = Brake_mode
```

When Brake_mode is 1, the brake is in “Auto” mode, and is switched on immediately whenever the spindle is switched off. When Brake_mode is 0, the brake is left off.

We can change this so that Brake_mode is set to zero any time the spindle is running. In this case it is no longer necessary to check for spindle running before applying the Brake.

```
;
; Select Brake Mode
;
Brake_key_hit       = Brake_key AND / Last_brake_key
Last_brake_key      = Brake_key
Brake_mode          = ( ( Brake_mode XOR Brake_key_hit
                            ) OR / Already_run )
                           AND / SpindleRelay
;
; Turn on the brake if spindle not running and in auto mode
;
Brake               = Brake_mode
Spindle_brake_LED   = Brake_mode
```

Note the parentheses in the Brake_mode expression. They are required because AND has a higher precedence than OR. Note also the spaces around the parentheses. The PLC compiler needs them to separate tokens. If we wrote the Brake_mode expression as:

```
        Brake_mode          = ((Brake_mode XOR Brake_key_hit)
                              OR / Already_run)AND
                              / SpindleRelay
```

the PLC compiler would give us error messages, saying that it did not recognize “((Brake_mode”, “Brake_key_hit)” and “Already_run)”. In other words, the compiler would think that the parentheses were parts of the names, and therefore that the names were

new and different ones from what we had defined earlier in our program. So always put spaces around parentheses.

File/Save
File/Exit
C:\PLC> **PLCCOMP /I K6789**

```
PLCCOMP v. 7.17 - PLC compiler

Compilation successful
```

C:\PLC> **CNC10M4**

The /I (install) switch tells PLCCOMP to copy the compiled PLC program over to the CNC10 directory as CNC10.PLC.

If we had made any mistakes in our PLC source file, PLCCOMP would report the errors, and would not produce any compiled program. For example, if we had left out the spaces around the parentheses in the example above, we would see something like this:

```
PLCCOMP v. 7.17 - PLC compiler

Unknown item: ((BRAKE_MODE on line 238

Unknown item: BRAKE_KEY_HIT) on line 238

Unknown item: ALREADY_RUN) on line 238
Missing operator on line 239
```

PLC Hardware

We have produced a wide assortment of PLC input/output boards over the years. Programming is done in the same way regardless of the board installed; the hardware merely determines which inputs and outputs are physically available, and sometimes how you can use them. You must set the “PLC Type” selection on the CNC10 Control Configuration screen to match the PLC hardware, see chapter 1 for PLC hardware descriptions.

Locating Source for Current PLC Program

If you are working on an older control, or one which has seen custom work, you may not be certain which PLC program is actually in use. Is it a “stock” PLC program, or has someone modified it? Is it based on an earlier version of a standard PLC program?

Before doing any custom work of your own, you want to be sure you have the source file for the PLC program which has been running on the control. Otherwise you risk breaking features that are important to the customer.

Since approximately software version 5.10, the compiled CNC10.PLC file has contained comments indicating the source file from which it was compiled. The first step, then, is to look there:

<Ctrl-Alt-X>
C:\CNC10> **EDIT CNC10.PLC**

```
; PLC program compiled by PLC compiler PLCCOMP v. 5.26
; Source file: M400.SRC
; Date:        10 September 1997
; Time:        09:01:23
;
9B 20 FC 21 FC F2 A2 FC F1 FD
A7 01 FC 00 FC F2 03 FC F2 02 FC F2 05 FC F2 04 FC F2 FD
 .
 .
 .
```

Note that the date and time listed show when the file was compiled, not the date and time when the source file was last modified. More recent versions (since approximately software version 6.03) list the source file date and time as well as the compilation date and time:

```
; PLC program compiled by PLC compiler PLCCOMP v. 6.03
; Source file: F:\SOFTWARE\PLC\CUSTOM\M400B.SRC
; File Date:  9-20-99 11:30a
; Compiled:   7-14-01  3:40p
;
A5 01 FC 00 FC F2 03 FC F2 02 FC F2 05 FC F2 04 FC F2 FD
A4 20 FC 21 FC F2 FD
 .
 .
 .
```

File/Exit

If we did not find a useful comment at the beginning of the CNC10.PLC file, the next step is to compare its file time, date, and size to a compiled PLC file we may find in the PLC directory.

C:\CNC10> **DIR CNC10.PLC**

```
Directory of C:\CNC10

CNC10     PLC         1,675  06-09-94  5:40p
        1 file(s)          1,675 bytes
```

C:\CNC10> **CD \PLC**

C:\PLC> **DIR *.??C**

```
Directory of C:\PLC

M40-N3B  SRC        12,854  06-09-94  5:39p
     M40-N3B  PLC         1,675  06-09-94  5:40p
```

```
           2 file(s)         14,529 bytes
```

Here we see that there is a file in the PLC directory, M40-N3B.PLC, which appears to be the same as the running CNC10.PLC file. We could make certain using the DOS file compare utility FC:

C:\PLC> **FC M40-N3B.PLC \CNC10\CNC10.PLC**

```
Comparing files M40-N3B.PLC and \CNC10\CNC10.PLC
FC: no differences encountered
```

Furthermore, we see we have a matching M40-N3B.SRC file which was last modified shortly before the PLC file was compiled. We can safely assume it is the source of the active PLC program.

If there are source files in the PLC directory, but no compiled .PLC files, you can always compile each source file (without installing!), then use FC to compare the resulting .PLC files to the CNC10.PLC file in the CNC10 directory. If there are no differences outside of the header comments, then you have found your source file.

If all of those methods fail, then the current PLC source is not stored on the control. Your best bet at that point is to catalog the jog panel type, PLC board type, I/O connections, and control features, then choose the most appropriate PLC program source you can find. It is a good idea in this case to preserve the old CNC10.PLC file for reference.

PLC Memory Map

Inputs	
1 - 15	Physical inputs on PLC board (0 = closed, 1 = open)
16	Integrity signal (1 = okay)
17 - 32	Inputs from PLCIO2, RTK3 or Koyo PLC (otherwise unused)
33 - 48	M function requests (M94 and M95)
49 - 58	AUX keys (normally 0; 1 when pressed)
59 - 62	PIC inputs
63	Mid range

64	High/low range
65	CNC program running
66 - 75	Jog panel keys
77	Pause (Feed Hold) (0 = pause, 1 = run)
78	Block Mode key
79	Rapid Override key
80	Disable spindle override (force 100%)

Outputs	
1 - 15	Relays on PLC board (0 = off, 1 = on)
16	Integrity signal (1 = fault)
17 - 24	Analog spindle speed (8 bits, for use on RTK2)
25 - 28	Extended resolution analog spindle speed (12 bits total, for use by Koyo)
29 - 40	Used with PLCIO2, RTK3 or Koyo PLC
41 - 48	BCD tool number
49 - 58	AUX LEDs
59 - 62	PIC outputs
63	Low lube fault (1 = fault)
64	Servo drive fault (1 = fault)
65	Spindle fault (1 = fault)
66 - 73	Jog panel LEDs (1 = on)
75	Stop (1 = any fault)
76	PLC operation indicator (1 = busy)
78	Single block mode (1 = on)
79	Rapid override (1 = enabled)

Memory	
1 - 40	General purpose
41 - 48	Messages, or general purpose
49	XPLC in use if set
50 - 72	Messages, or general purpose
73 - 77	Reserved
78	Copy of Spindle_relay, for PC spindle control
79	Copy of CCW_relay, for PC spindle control
80	M function macro indicator (v5.27 and earlier)

What does a 1 or 0 mean?

For outputs, the logic is quite simple: if the PLC output bit is 0, then the output is off (relay open, LED off, etc.); if the output bit is 1, then the output is on (relay closed, LED lit, etc.).

For inputs, the logic varies:

For physical inputs (INP1 through INP15), 0 indicates that the switch is closed (the input point is being pulled to ground); 1 indicates that the switch is open (the input point is maintained at 5V by the internal pull-up).

For inputs sent from a Koyo PLC (INP4 through INP15 and INP17 through INP32), the 0 or 1 state is as sent by the Koyo PLC program.

For M function request bits (INP33 through INP48), M94 sets the bit to 1 and M95 sets the bit to 0.

For jog panel keys, 0 indicates that the key is not pressed; 1 indicates that the key is currently pressed.

When you name the inputs in your PLC program, keep the normal states in mind. For example, if you connect a normally-closed inverter fault signal to INP14, you might want to name it "Spindle_fault", because it will be 0 (closed) if all is well, and 1 (open) in the event of a fault. If on the other hand you are using a normally-open fault signal, then you should name it something like "Spindle_okay", since 0 (closed) indicates a problem and 1 (open) indicates that all is well.

This naming guideline can be a little more awkward in some cases: notably for "start" buttons which are wired to PLC input points. Fail-safe design requires that start buttons be normally open, but that means the inputs will show up as 1 (open) when idle, and 0 (closed) when the button is pressed. Strictly speaking, you should name the input something like "Remote_start_not_pressed". In such cases, though, you might conclude that a shorter name, like "Remote_start", is more readable even if it is not literally correct.

Some Special Locations

INP63 - Mid_range
INP64 - High_low_range

These inputs define the current spindle gear range. A two-speed machine need only use INP64 (0 = high range, 1 = low range). A three or four-speed machine uses both inputs, as follows:

Range	INP63	INP64
Low	0	1
Low-Mid	1	1

High-Mid	1	0
High	0	0

Both the CPU7 and CNC10 use these inputs to compute motor speed for a desired spindle RPM, and to compute display RPM from motor speed. Typically these inputs are copied from physical input switches: e.g. a range detect switch under the back gear lever which is closed (0) in high range and open (1) in low range. These bits could also be set by internal PLC logic which is controlling an automatic gear changer.

INP65 - CNC_program_running

This input tells the PLC program when a job is in progress. It will be set (1) whenever a CNC program is running. It will also be set in MDI mode and during homing, digitizing, probing, and other automatic machine movement. It will not be set if the operator is merely jogging an axis.

The standard PLC programs use INP65 to run the lube pump during jobs, and to cancel all automatic M functions when the job ends or is canceled.

INP77 - Pause

This bit reflects the FEED HOLD state. It would be more appropriately named “Run”, since a value of 0 means the machine is paused, and a value of 1 means the machine is allowed to run.

Standard PLC programs do not use this bit, but custom applications can use it to implement additional FEED HOLD buttons; to activate FEED HOLD when a guard or cover is opened; or to allow or disallow certain operations during FEED HOLD.

OUT75 - Stop

This bit indicates a fault condition. When it is set, the CPU7 will immediately cancel any running CNC job, and will release power to the servo motors. The standard PLC programs set this bit in response to the five standard faults: servo drive fault; spindle fault; PLC failure; low lube; and emergency stop. Note that a low lube fault is held off as long as INP65 (CNC_program_running) is set, allowing the current job to finish.

Custom applications may set the Stop bit as needed to force a fault. The on-screen message will be determined by four reserved fault bits:

OUT16 - PLC_fault_out
OUT63 - Lube_fault_out
OUT64 - Drive_fault_out
OUT65 - Spindle_fault_out

If one of these bits is set, CNC10 will display either the matching custom message (see Custom On-Screen Messages below) or the default message for that fault if no custom message is defined.

If none of the four known faults is indicated, then the control will assume it is just E-stop, and display “Emergency stop detected”.

If your application requires a special fault condition, but does not require all four standard faults (e.g. there is no auto lube system) then you can redefine one of the standard faults (e.g. set OUT63 in the event of trouble, and define a custom message to go with it).

If you need a special fault condition, and still need to use all four standard faults, then you will have to live with the display of “Emergency stop detected” when your fault occurs. You can still get a custom message to appear in addition, by setting one of MEM41-MEM72 and defining a message for that bit.

OUT76 - PLC_op_signal

This bit allows the PLC program to tell the CNC processor that some PLC-related operation is not yet complete, and therefore that the CNC job should be paused awaiting completion.

The standard PLC programs set this bit whenever the CNC program attempts to start the spindle in automatic mode (using M3 or M4), but the operator has switched to manual spindle control (using the Spindle AUTO/MAN button on the jog panel). In this case the message “Waiting for PLC operation (M3)” will be displayed. OUT76 will be cleared and the job will proceed only when the operator switches back to auto spindle control.

OUT76 is set to 1 automatically at the beginning of each PLC program scan. If there are no conditions in your PLC program which might need to set OUT76, you will still need an assignment in order to clear it back to 0.

```
PLC_operation = Zero
```

MEM41- MEM48 and MEM50- MEM72

These bits can be used to trigger custom messages in the status window. See Custom On-Screen Messages below. When not being used for that purpose, they can be used freely like any other memory bits.

MEM49

MEM49 is used by the XPLC program to communicate to the standard program that the XPLC program is being used.

Custom On-Screen Messages

There are several ways you can make custom messages appear in the CNC10 status window.

In some cases you substitute your message for a default message the control would otherwise have displayed in the course of its operation. In other cases you specify a message to be displayed “asynchronously”, without regard to what the control is currently doing.

Custom messages are stored in the file CNC10XMSG.TXT, in the CNC10 directory. Each message occupies two lines:

```
MEM41
“Unplug the probe!”
```

The first line specifies the PLC bit which triggers the message; the second line is the message itself, in double quotes.

Waits

Whenever the CNC processor is executing an M100 or M101 (waiting for any input) or M0 (waiting for INP75 - CYCLE START) you can substitute your own message by associating your message with the relevant bit. For example, the following definitions would cause appropriate messages to be displayed while executing M80 and M81 operations on a tool changer mill running the ATC6 PLC programs:

```
INP27
“Waiting for carousel to advance”
INP28
“Waiting for carousel to retract”
```

Limits

Whenever a limit switch (any switch number specified on the Machine Configuration screen) is tripped, the control displays a message such as “Y+ limit (#3) tripped”. You can replace this message with your own by associating a message with the limit switch number (e.g. INP3).

Faults

As mentioned previously, you can associate messages with INP16, INP63, INP64, and INP65 to replace the default messages for PLC failure, low lube, servo drive fault, and spindle fault.

Asynchronous Messages

If you want your message to appear whenever a condition is present, regardless of what the CNC processor might be doing at the moment, you can associate it with a memory bit ranging from MEM41 to MEM72. Your message will be displayed any time the bit is set.

Common Expressions

Unconditional Output

It is often useful to set or clear a PLC bit without regard to any inputs. For example, a PLC program which does not detect any fault conditions (e.g. on a machine with no PLC I/O board installed) might clear the fault indicator with the line:

```
Stop = Zero
```

Likewise, a PLC program for a control with no Rapid Override key (e.g. M15 models 1 through 7) might enable Rapid Override mode with the line:

```
Rapid_override = / Zero
```

Many PLC programs give the name “Zero” to one of the memory bits (often MEM11). There is nothing special about this location. All memory bits are initialized to 0 at power up. If you never change one, then you can use it for unconditional outputs like this.

If you change PLC programs after power up, it is possible that the earlier program set a memory bit that the new program would like to assume is 0. You can explicitly clear or set bits with lines such as the following:

```
Zero = Zero AND / Zero        ; always cleared
One = One OR / One            ; always set
```

Latched Fault

The latch and reset expression we saw earlier is used with fault conditions. If a fault occurs, we want to catch it and report it even if the problem was only momentary. Once we have seen the error message and fixed the problem, then we can clear the fault by pressing and releasing Emergency Stop.

```
PLC_fault_out       =  / PLC_ok OR ( PLC_fault_out AND
                                   / E_stop )
```

The PLC fault output will be set if there is a problem with the PLC integrity signal on INP16. Once set, it will remain set until the problem is fixed (INP16 comes back on) and Emergency stop is pressed.

Detecting Changed Inputs and Toggling an Output

We often want a single input to switch an output either on (if it was off) or off (if it was on). We can do this easily with the XOR operator. However, we need to ensure that the output changes state only one each time the input is activated. If we wrote:

```
Work_light = Work_light XOR Aux_1_key
```

Then the light would switch rapidly on and off as long as the key was held down.

To make this work properly, we need to detect when the key is pressed and toggle the output only in that one scan of the PLC program. We can do this by storing the previous state of the key in a memory location.

```
Last_aux_1 IS MEM23
Aux_1_hit  IS MEM24
     [...]
```

```
Aux_1_hit  = Aux_1_key AND / Last_aux_1
Last_aux_1 = Aux_1_key
Work_light = Work_light XOR Aux_1_hit
```

If the key is down now, and was not down in the previous scan, then the "Aux_1_hit" memory location will be set for this one scan, and the output will toggle just once.

Initializing Bits on First Scan

Another common need is to set a bit on the first scan of the PLC program, then let it change according to normal logic. We can detect the first scan by setting a memory location as the last operation in our program. If that location is set, then we know we have been through the program at least once. If it is not set, then we know this is the first scan.

At the end of the program:

```
Already_run = / Zero
```

or

```
Already_run = Already_run OR / Already_run
```

Then, earlier in the program, we can refer to this when we want to set some mode explicitly on startup. We saw this previously in the spindle brake example, where the automatic braking mode is set on startup, then toggled with the BRAKE key on the jog panel:

```
Brake_mode          = ( Brake_mode XOR Brake_key_hit ) OR / Already_run
```

Application Examples

Remote FEED HOLD

Some operators, particularly those who also use older M10 and M40 controls, find the FEED HOLD key on M15/M39/M400 jog panels too small, and hard to locate quickly.

It is fairly simple to install an external pushbutton as an auxiliary FEED HOLD button. Unlike the button on the M40, it cannot latch: it must be a momentary button, since CYCLE START is required to restart out of FEED HOLD.

Assume we have an illuminated red pushbutton with a normally-closed contact block (e.g. a Cutler-Hammer E22TB2X4B). We wire the switch contacts to INP9, and wire the light through OUT8.

In the definitions section we have:

```
Remote_pause        IS INP9    ; 0 = idle      1 = pressed
```

and

```
Feed_hold_light     IS OUT8    ; 0 = off       1 = on
```

In the program we have:

```
;
; Turn on feed hold if remote feed hold button is pressed
;
```

```
Pause = Pause AND / ( Remote_pause AND CNC_program_running ) OR
Cycle_start
Feed_hold_light = / Pause
```

Recall that INP77 (“Pause”) is 0 to pause, 1 to run.

The first statement changes Pause to 0 if the remote pause button is pressed while a job is running, and changes it back to 1 if CYCLE START is pressed. The “OR Cycle_start” clause is not necessary if we only have the built-in CYCLE START button: CPU7 will automatically cancel FEED HOLD when that button is pressed. However, if we have a remote CYCLE START, as shown below, CPU7 will not automatically recognize it as canceling FEED HOLD. In that case we have to do it explicitly as shown above.

The second statement turns on the light any time we are in FEED HOLD.

Remote CYCLE START

Many operators also like to have an additional CYCLE START button in a handheld pendant or mounted on the machine head. If you have installed a remote FEED HOLD as described above, it is particularly convenient to have a remote CYCLE START located nearby for restarting out of FEED HOLD.

Assume we have an illuminated green pushbutton with a normally-open contact block. We wire the switch contacts to INP8, and wire the light through OUT7. The light will be an in-cycle indicator, illuminated any time a job or other automatic cycle is running.

In the definitions we have:

```
Remote_start        IS INP8    ; 0 = pressed   1 = idle
and
In_cycle_light      IS OUT7    ; 0 = off       1 = on
and
Last_remote_start   IS MEM30
```

Fail-safe design demands that we use a normally-open switch contact for any START button. Unfortunately this results in "reversed" logic in the PLC program. INP8 will be 1 when the button is not being pressed, and will change to 0 when the button is pressed.

The logic for combining the remote start button with the existing input for the built-in CYCLE START button is fairly complicated. I will not attempt to explain here how it is derived. Also, normally-open inputs like this one must usually be explicitly set in the first scan of the PLC program.

In the program, then, we have:

```
;
;  Set Remote_start to 1 (not pressed) on first pass through PLC program
;
Remote_start = Remote_start OR / Already_run
     ;
```

```
;  Combine Start and Remote_start into Start
;
Cycle_start =  ( / Cycle_start AND / Remote_start AND /
Last_remote_start )
            OR (   Cycle_start AND   Remote_start AND /
Last_remote_start )
            OR (   Cycle_start AND / Remote_start )
Last_remote_start = / Remote_start
;
In_cycle_light = CNC_program_running
```

The light, of course, is optional.

Remote CYCLE CANCEL

Remote Cancel logic is a little simpler than Remote Start, because the result (in INP74) only needs to stay on for a single scan.

```
Rem_cancel          IS INP9   ; 0 = idle, 1 = pressed
[...]
Cycle_cancel        IS INP74  ; 0 = continue, 1 = cancel
[...]
Last_remote_cancel  IS MEM26
Last_cancel         IS MEM28
[...]
;
;  Combine local and remote cancel into cancel; clear after one cycle
;
Cycle_cancel =  ( / ( Cycle_cancel AND Last_cancel )
                  AND Remote_cancel AND / Last_remote_cancel )
                OR ( Cycle_cancel AND / Last_cancel )
;
Last_remote_cancel = Remote_cancel
Last_cancel = Cycle_cancel
```

Cycle_cancel will be set if the remote cancel button was just pressed, or if the built-in cancel button was just pressed; it will be cleared again once it has been set for one scan.

AUX key to toggle an output

The standard PLC programs which ships on M39 and M400 controls use the four Aux keys as on-off toggles. For example, Aux1 normally controls OUT6. Pressing the key switches the output from off to on, or from on to off.

The code to implement this is as follows:

```
Aux_1_out           = Aux_1_out XOR ( Aux_1_key AND / Last_aux_1 )
Aux_1_LED           = Aux_1_out
Last_aux_1          = Aux_1_key
```

The output retains its last value unless the key was just depressed. In that case the output changes state (1 becomes 0, 0 becomes 1). XOR is a convenient way to achieve this.

The LED in the Aux key indicates the current state of the output.

The last state of the key itself must be saved in a memory location so that changes can be detected.

AUX key for a momentary output

Sometimes you only want the output to remain on as long as the button is held down. Suppose you have a knee mill retrofit with the Z axis in the quill. You would like to power the knee, but cannot justify the cost of a full four-axis control package. Instead, you install a DC gear motor as a knee power feed, and use the Aux1 and Aux2 keys as knee up/down jog keys. You install a pair of relays suitable for the motor load, and wire their coils to OUT6 (knee up) and OUT7 (knee down).

In the definitions, then, you would have:

```
Knee_up_relay   IS OUT6
Knee_down_relay IS OUT7
```

In the program, you could write:

```
Knee_up_relay   = Aux_1_key
Knee_down_relay = Aux_2_key
```

Each relay would close as long as the corresponding key was held down, jogging the knee in the selected direction.

However, you would have to be sure to wire interlocks through your reversing relays. Otherwise the operator could cause a dead short by pressing both Aux keys at the same time.

A safer solution would be as follows:

```
Knee_up_relay   = Aux_1_key AND / Aux_2_key
Knee_down_relay = Aux_2_key AND / Aux_1_key
```

This has the same effect, but turns off both outputs in the event both Aux keys are pressed.

Custom M Functions

You can easily create custom M functions for a variety of applications. Any M function from M0 through M90 can be defined or redefined with an M function macro file.

Suppose you have a lathe with an air spindle for live tools. You want to use M32 to turn the air spindle on, and M33 to turn it off.

To do this, you need to find an unused bit in the range INP33 through INP48. These bits are M function requests, sent by M functions in the running CNC job to the PLC program. The first few are usually already in use for spindle and coolant control, and sometimes for a clamp or collet closer. For this example we will assume that INP38 is available.

The M94 and M95 codes control the M function request bits. M94 turns the specified request on; M95 turns the request off. With M94 and M95, the M function requests are numbered 1 through 16 (even though they are mapped to inputs 33 through 48).

Therefore, to turn on INP38 we would execute M94/6. To turn off INP38 we would execute M95/6.

We create two files in the control software directory (e.g. C:\CNC10T):

CNC10.M32:
```
; M32 - turn on air spindle
M94/6
```

CNC10.M33:
```
; M33 - turn off air spindle
M95/6
```

This is all that is needed for the control to recognize M32 and M33 as valid M functions, and to execute the command(s) in the files whenever a program calls on them.

In the PLC program we need to accept and act upon the M function request. Setting INP38 with M94 does not by itself do anything.

In the definitions we have:

```
Air_spindle IS OUT5
```
and
```
M32 IS INP38
```

In the program, we have

```
Air_spindle = M32
M32 = M32 AND CNC_program_running
```
The first line turns on the output whenever it is requested by the M functions. The

second line is necessary to ensure that the output is turned off when the program ends, even if the programmer forgot to put in a final M33 (or the operator stops the job with CYCLE CANCEL or E-stop).

Note that there is no definition for M33 in the PLC program. M33 is simply the absence of M32.

Custom M Function with AUX key toggle (manual override)

In the previous example, we assumed that the air spindle would be used only with M functions, and only during a programmed job.

Suppose we would also like to have manual control of the air spindle, and would like to be able to turn it on even if no job is running (for example, to drill a cross hole using jog panel controls).

In that case we will need a couple more definitions (explained below):

```
Last_prog_running   IS MEM28
Continue_M_functs   IS MEM29
```

And a little more work handling M32 in the program:

```
Continue_M_functs = ( CNC_program_running OR / Last_prog_running )
                        AND / Stop
Last_prog_running = CNC_program_running

M32 = ( M32 XOR ( Aux_1_key AND / Last_aux_1 )  ) AND Continue_M_functs

Air_spindle = M32
Aux_1_LED = M32
```

The "Continue_M_functs" indicator says that dual-mode M functions (automatic and manual) should continue to run unless the job just ended or a fault condition is present. This allows the spindle to stop automatically at the end of the job, while still allowing the operator to restart it while no job is running.

The M32 request bit will be set and cleared by M functions as they appear in a job. However, the request will also be toggled whenever the Aux1 key is pressed. Aux1 can then be used both to control the air spindle when no job is running, and also to override M32 and M33 during a job.

Deferring Spindle Brake Until Inverter Stops

In the first tutorial example we looked at a way to prevent the inverter from fighting the spindle air brake, by disabling the automatic brake mode whenever the spindle was running. That solution is convenient because it does not require any additional control wiring. However, it is not ideal. If the operator wants the brake to be applied whenever the spindle is stopped, he always has to press the BRAKE key; and if he pressed the BRAKE key

immediately after SPIN STOP then the brake will be applied, fighting the inverter just like it did before.

A better solution is to have the inverter inform the PLC when the spindle has decelerated to a stop. Most inverters have spare programmable outputs, and most allow you to program one of those outputs to be a zero-speed (sometimes called "baseblock") indicator.

Suppose we program an inverter output to close at baseblock, and wire that output to INP8 on our PLC board. We can then write the following PLC program:

```
Spindle_running IS INP8   ; 0 = stopped  1 = running

Brake           = Brake_mode AND / ( SpindleRelay OR
                             Spindle_running )
```

We can keep the standard logic for toggling Brake_mode on and off with the BRAKE key.

The line above simply inhibits the brake output until the spindle winds down. We might be tempted to simplify it to:

```
Brake           = Brake_mode AND / Spindle_running
```

But it is likely that there is a lag between the time the inverter receives the Run signal (SpindleRelay) and the time the zero-speed output opens. During this lag the inverter would be trying to start the spindle against the brake. Thus it is better to leave both conditions in place.

Tool Number Feedback

On a machine with an automatic tool changer, the on-screen tool number display is driven by the tool number in OUT41 through OUT48. This preserves the last automatic tool change command (actually the last M107) even if we exit and later restart CNC10.

However, suppose we implement a manual tool change (for example, using the Turret Index key on a T400 to rotate a new tool into position). In this case the machine has changed tools, but there has been no new M107; no new bits in OUT41 - OUT48; and no change in the screen display.

We would like to have our PLC program override the tool number bits after a manual index, without interfering with the tool numbers for automatic tool changes.

Assume we have a 6-station turret controlled by a Koyo PLC. The CPU7 sends the state of the Turret Index key to the Koyo, and the Koyo acts upon it to rotate the turret to a new position. When indexing is complete, the Koyo PLC knows the new position, and wants to plug that number into the tool number bits. We provide three bits of tool number information, plus a strobe bit to indicate that they should be copied into OUT41 through OUT43.

In the definitions section we have:

```
;
;  Actual tool information from Koyo
;
Actual_tool_1    IS INP17
Actual_tool_2    IS INP18
Actual_tool_4    IS INP19
Actual_tool_set  IS INP20

;
;  Tool number for ATC
;
Tool_BCD_1          IS OUT41
Tool_BCD_2          IS OUT42
Tool_BCD_4          IS OUT43
```

And in the program we have:

```
;
; Use Koyo signal to force tool number to selected pattern
;
Tool_BCD_1 = ( Tool_BCD_1 OR ( Actual_tool_set AND Actual_tool_1 ) )
             AND / ( Actual_tool_set AND / Actual_tool_1 )
Tool_BCD_2 = ( Tool_BCD_2 OR ( Actual_tool_set AND Actual_tool_2 ) )
             AND / ( Actual_tool_set AND / Actual_tool_2 )
Tool_BCD_4 = ( Tool_BCD_4 OR ( Actual_tool_set AND Actual_tool_4 ) )
             AND / ( Actual_tool_set AND / Actual_tool_4 )
;
```

This preserves the state of the Tool_BCD_x bits until Actual_tool_set is selected. Then it sets or clears each tool bit based on its corresponding bit from the Koyo.

Switching Optional Stops During Job Run, with Aux Key

Some operators request a more convenient way of turning on Optional Stops; a way to turn Optional Stops on after the job is started; and an Optional Stops feature that is not automatically canceled at the end of each job.

We can do this by bypassing the Optional Stops feature on the F4/Run menus, creating a custom macro for M1, and assigning control of the Optional Stops mode to an Aux key.

In this example we use the Aux4 key for Optional Stops control. INP38 and MEM32 are used internally between the M1 macro and the PLC program; any other available M function request and memory bit could be used instead.

The logic is that, whenever an M1 is encountered, we will stop and wait for CYCLE START if Optional Stops are on. The Aux4 key will toggle Optional Stops. The Optional Stops status is both stored and displayed in the Aux key LED.

Edit the PLC program source file as follows:

```
M1  IS INP38      ; 0 = idle   1 = opt. stop    M94/6 M95/6
                  ; (or any available request bit –
                  ;see M1 macro)

[...]

~~Aux_4_LED       IS OUT52~~
Optional_stops IS OUT52

[...]

Wait_at_M1 IS MEM32 ; <-- or any available memory bit -- see M1 macro

[...]

~~Aux_4_out       = Aux_4_out XOR ( Aux_4_key AND / Last_aux_4 )~~
Optional_stops  = Optional_stops XOR ( Aux_4_key AND / Last_aux_4 )
~~Aux_4_LED       = Aux_4_out~~
Last_aux_4      = Aux_4_key
;
; Handle optional stops whenever an M1 comes in
;
Wait_at_M1 = ( Wait_at_M1 OR ( M1 AND Optional_stops ) ) AND /
Cycle_start
;
```

Create a file CNC10.M1 as follows:

```
M94/6               ; <-- M1 input bit minus 32
M95/6               ; <-- M1 input bit minus 32
M101/192            ; <-- “Wait” memory bit plus 160
```

Add an entry to the CNC10XMSG.TXT file similar to the following:

```
MEM32
"Optional Stop - press CYCLE START"
```

The M1 macro flashes its M function request on and off. The PLC program takes this as a signal to set the "Wait_at_M1" memory bit if Optional Stops are enabled. Once the "Wait..." bit is set, it is not cleared until the operator presses CYCLE START. Thus the control will not restart from an M1 just because the operator turns off optional stops; he must also press CYCLE START one last time.

There is one minor side effect to this change. Consider a program like the following:

```
N1 G0 X0 Y0
N2 G1 F20 X1
N3 M1
N4 X2
```

With the built-in optional stops feature, if optional stops are turned off, the X axis will move continuously from X0 to X2.

With the macro-based version presented here, even if optional stops are turned off, the X axis will decelerate to a stop at X1, then resume the move to X2.

In most cases this will not matter. M1 codes are traditionally placed at tool changes and between operations, not in the middle of a cut.

Rapid Override on Early Controls

The Rapid Override feature was added to the control software in early 1994. Controls produced up to that time (M10 and M40 family) did not have a dedicated Rapid Override key on the jog panel. We modified the standard PLC programs to provide Rapid Override control using the Aux1 key:

```
Rapid_over_key IS INP49 ; AUX1_key
[...]
Rapid_over_LED IS OUT49 ; AUX_1_LED
[...]
Rapid_override        IS OUT79
[...]
Rapid_over_key_hit    IS MEM18
Last_rapid_over_key   IS MEM19
[...]
;
;   Toggle Rapid Override mode with AUX1 key
;
Rapid_over_key_hit      = Rapid_over_key & / Last_rapid_over_key
Last_rapid_over_key     = Rapid_over_key
Rapid_override          = ( Rapid_override XOR Rapid_over_key_hit )
                          OR / Already_run
Rapid_over_LED          = Rapid_override
```

Since most operators prefer to have Rapid Override enabled, the “OR / Already_run” clause is used to turn the mode on during the first scan of the PLC program. Thereafter it is toggled off or on each time the jog panel key is pressed.

The M400 and M39 jog panels have a dedicated Rapid Override key, mapped to INP79. The LED indicator in this key is mapped to OUT79, so it reflects the Rapid Override mode automatically, without any assignment needed in the PLC program. Otherwise, the logic in an M39 or M400 PLC program is identical to that above.

Some early M40 PLC programs have Rapid Override control, but do not turn it on automatically on the first scan. As a result, operators are often unaware of the feature. It is a fairly simple matter to edit or replace the PLC program to enable the Rapid Override mode on the first scan.

Some router tables used an M40 style jog panel mounted in a walk around pendant. Because most of these machines did not have PLC boards, and therefore did not have automatic spindle or coolant control, the pendants cover up the right half of the jog panel with a metal plate. Often the underlying keypad does not even have snap domes installed on that side.

In that case, there is no Aux1 key to be used for Rapid Override control. The best solution is to simply turn the Rapid Override mode on unconditionally. Again, this is a simple edit:

```
Rapid_override = / Zero
```

The jog panels on CNCDROs and M15s through model 7 have neither a Rapid Override key nor Aux keys, so their PLC programs should use the same line to turn on Rapid Override mode unconditionally.

PLC Diagnostics

Press <ALT I> at he main menu of CNC10 to turn on PLC diagnostics. Red and Green indicators will display the status of the current state of the PLC inputs, outputs and memory locations. Using the arrow keys you can highlight different PLC bits to show the label given to the selected bit.

Interfacing with a Koyo DirectLogic PLC

We provide an optional interface which allows you to connect the CPU7 to a Koyo DirectLogic DL205 PLC using a D2-240 CPU. The DL205 is a modular PLC system supporting up to 256 I/O points with a wide array of input and output devices.

The connection between the CPU7 and the D2-240 PLC CPU is via the tiny CPU7ADD board, plugged into a header on the CPU7, and the larger OPTIC232 board mounted in the cabinet. OPTIC232 connects to CPU7ADD via three optical fibers; it connects to the D2-240 via an RJ12 cable and optionally a two-wire interrupt signal from one of the PLC output modules.

With this system, the CPU7 sends 48 of its output bits to the Koyo CPU, and receives back 32 input bits (only 29 of which can be used, for reasons explained below).

Obviously, with perhaps 40 or 50 physical inputs, the Koyo cannot send every input to the CPU7. Fortunately, it doesn't have to. Typically the two PLC programs are written so that the Koyo does all the hard work, and only reports final results back to the CPU7.

Consider an automatic tool changer application:

To request a tool change, the CPU7 sends the desired tool number and a "start tool change" signal to the Koyo. The Koyo may have to deal with a large number of outputs (hydraulic or pneumatic solenoid valves; carousel motor starters; etc.) and inputs (tool counter switches, arm position switches, clamp/unclamp switches, etc.). However, it only needs to report back to the CPU7 with one bit indicating that the tool change is complete. It is usually not even necessary to report failure, since a properly written tool-change macro will have a timeout on the CPU7 side, forcing an error if the completion signal is not received in the expected length of time.

Although the CPU7 sends OUT1 through OUT48 to the Koyo, and received INP1 through INP32 back, it assumes that OUT1 through OUT3 and INP1 through INP3 are part of a 3/3 PLC board used along with the Koyo. Because of this, it does not update its copies of INP1 through INP3 with data from the Koyo, but instead saves those locations for the 3/3 PLC.

The 3/3 PLC should be used for any inputs where predictable timing is important. The lag time from when an input is actually triggered to when the CPU7 receives the bit change is unpredictable. It may be anywhere from 10ms to 100ms. If a Koyo input were used for a touch probe, this would lead to inconsistent probing and digitizing measurements, as the amount of movement between the time the probe tripped and the time the control stopped motion would vary.

Therefore you should always connect probe or tool detector inputs through a 3/3 PLC.

The 3/3 PLC can also be useful for electronic compatibility. Its inputs are 5VDC current sourcing inputs, while Koyo inputs are typically 24VDC sourcing or sinking. The fault output on our servo drive board is designed to connect only to a 5V sourcing input. Therefore you can connect the servo fault directly to a 3/3 PLC input, while you would have to use an intermediate relay if you wanted to connect it to a Koyo input.

When a Koyo PLC is used, it is typically equipped with an Analog output module (usually an F2-02DA-2) to control spindle speed. This provides 12-bit spindle speed resolution, instead of the 8 bits available with the RTK2 or 15/15 PLC. On the CNC10 Machine Parameters screen, set Parameter 31 to a value of -1.0 to select 12 bit spindle speed control via the Koyo. The current spindle speed request will be sent out in OUT17 through OUT28.

The F2-02DA-2 module provides two independent analog output channels. Since only one is needed for spindle speed control, the second may be used as a convenient solid-state output for the CPU7 interrupt signal mentioned above. This is particularly desirable if all your other Koyo output modules are relay outputs: a relay output could quickly exceed its expected service life if it had to cycle every time any input changed.

Limit Switch Interface between PLC and Servo Drive

In most control applications, the axis limit switches are connected both to PLC and to the servo drive. Connecting them to the PLC allows the CPU7 and CNC10 to "see" the switch, and therefore to know what happened and why axis motion has stopped. Connecting them to the servo drive allows the drive to inhibit motor current in the direction of the tripped switch, ensuring that the move is shut down even if software is misbehaving or the axis is out of control (e.g. failed encoder).

It is not strictly necessary to connect the limit switches to PLC inputs. With software travel limits set, the control will never hit a limit switch once machine home has been set. The control is even capable of homing to a limit switch which inhibits the drive, but does not ap-

pear on a PLC input. If you choose not to connect the limit switches to the PLC, enter zeros for the limit and home switch numbers on the Machine Configuration screen.

You will still be able to jog into a limit switch before machine home has been set. If you do, the message will be “full power w/o motion” or “position error”, not “limit tripped”. You will still be able to slow jog off the switch as usual.

If you have an unbalanced axis that tends to “fall off” the tripped limit switch, and you want to use that switch as a home, you should connect the switch to a PLC input as well as the drive inhibit. The software logic for homing to a drive inhibit alone is prone to “full power w/o motion” stalls if the axis drops back after hitting the switch.

The method for wiring the limit switches to both PLC inputs and drive inhibits depends on the PLC and servo drive hardware. The most common examples are as follows:

RTK2 and Servo Drives

The DC servo drives (SERVO1 and QUADDRV) use 5VDC pull-down (current sourcing) inputs for the limit switch inhibits. Since these are electrically identical to the inputs on our PLC I/O boards, the limit switches can be connected to both devices in parallel.

On the RTK2, the limit switches themselves are wired to headers H1 and H3 on the middle board (RTK2B). H3 connects the X, Y, and Z limits to inputs INP1 through INP6. Internal connections also pass these signals through to a nearby 10-pin Waldom connector. Fourth axis limit switches, if any, are wired to H1. These signals are not mapped to any PLC logic inputs but they are also passed through to the Waldom connector.

A separate cable carries the limit switch signals from the RTK2B Waldom connector to the servo drive's limit switch header. In addition to the eight switch signals, this cable has positions for +5VDC and ground, allowing the two devices to match their reference levels. In current production, only the +5 wire is carried from the RTK2 to the servo drive.

A three axis drive has a bank of eight DIP switches next to the limit switch header. The first six are limit switch defeaters: if thrown up (away from the board), the respective limit switch is disabled. Motor current will not be inhibited regardless of limit switch state. In normal operation these DIP switches should all be thrown down (towards the board) so that limits are enabled. The remaining two switches, labeled "G" and "5" select whether the ground and +5VDC signals which drive the inputs are taken from on board the drive, or from the PLC via the limit switch cable. In normal operation with an RTK2 these switches should be thrown down (towards the board) to select PLC signals.

A four axis drive has two banks of DIP switches: a bank of eight which enable or defeat the eight limit switches, just like on a three axis drive; and a separate bank of two which select +5V and ground reference signals. The latter function the same as on a three axis drive, except that the switch position is reversed: up away from the board for normal (PLC

driven) operation, and down towards the board to use the drive's on board reference signals.

In a three axis control, connections on RTK2B H1 do not matter. In a four axis control, fourth axis limit switches should be wired to H1. If there are no limit switches (e.g. the fourth axis is a 360° rotary table) then jumpers should be installed on H1 to simulate closed limit switches.

Although H1 does not directly control any PLC inputs, you could connect one or both fourth axis limits to a PLC input by jumpering from H1 to the appropriate connector. For example, you could connect the W+ limit to INP8 (terminals 7 and 8 of H5) in order to use it more reliably as a home switch on an unbalanced axis.

15/15 PLC and Servo Drives

The electrical interface between the 15/15 PLC and the servo drive is identical to that between the RTK2 and the drive, except that the 15/15 PLC does not have any pass-through header.

Instead, the servo drive limit cable connections are piggybacked with the physical switch connections on the input header H1.

In current production, both the +5VDC and ground signals in the limit switch cable are wired from H3 on the PLC to the limit header on the servo drive.

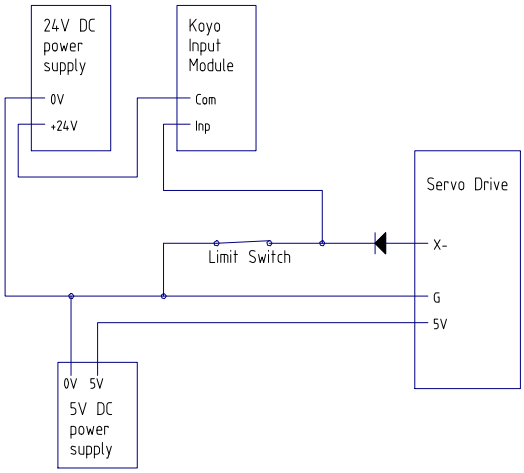
M15DRV1

Since the M15DRV1 combines PLC and servo drive on one board, the pass-through connection is made internally. Limit switches act as drive inhibits, and also appear on PLC inputs 7 through 12.

Koyo PLC and Servo Drives

Koyo PLC inputs typically operate on 24VDC, either current sourcing or current sinking. The 5VDC pull-up, which the servo drive can supply, is not enough to maintain a Koyo input.

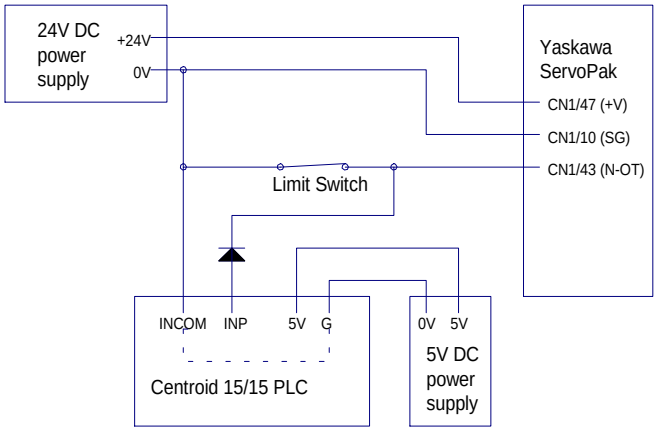
The solution to this incompatibility is to wire the limit switches to the Koyo as 24V current sourcing inputs, then connect each input to the servo drive limit header through a diode. The diode allows the limit switch input to be pulled to ground when the switch is closed, but prevents the Koyo's 24V from feeding into the drive's input when the switch is open. For this to work, the Koyo input and the servo drive ground references must be matched (connected). Typically the servo drive limit switch circuit is supplied by an external DC power supply; often the same one which powers a 3/3 PLC board and the OPTIC232 board.



RTK2 or 15/15 PLC and Yaskawa AC Drives

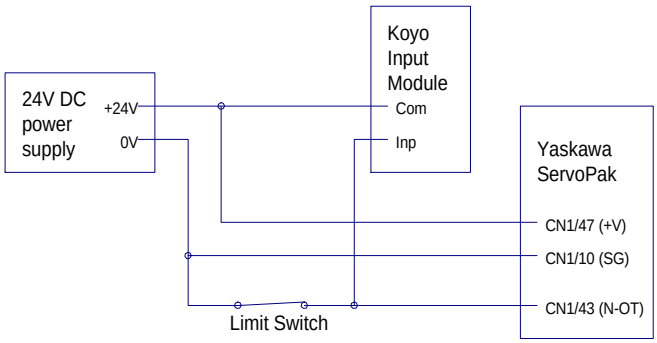
A similar electrical compatibility issue arises when using our PLC I/O board with Yaskawa or similar AC servo drives. The Yaskawa drives require 24V limit switch inputs, while our boards operate at 5V.

The solution is likewise similar: wire the limit switches to the Yaskawa drives at 24VDC, then connect them to PLC inputs through diodes. Again match the ground references by connecting the ground side of the 24V supply to the PLC input common.



Koyo PLC and Yaskawa AC Drives

Since both these devices use 24V inputs, no diodes are needed. Just connect the positive side of the 24V supply to both the PLC and the drives. Connect the negative side to the limit switches.



A note on AC servo tuning

With our DC drives, the standard PID parameters of Kp = 1.0, Ki = 0.004, and Kd = 10-20 work quite well in nearly all installations.

These gains are much too high for Yaskawa AC drives. One visible effect is that the axis will "jump" or "bang" when moving back off a tripped limit switch. Since moving on and off the limit switches is a normal part of machine homing, we would like the operation to be smooth and quiet.

Some retrofitters have reported defeating the drive inhibits in order to resolve the problem. They rely solely on the control software to enforce the limit switches. That is not the best solution. The best solution is to use lower PID gains, so that movement is smooth even when a limit switch is inhibiting current in one direction.

A good starting point for tuning is Kp = 0.2, Ki = 0.001, Kd = 2.0. The final numbers you use will vary depending on your installation and drive/motor selection.

XPLC Programming (PC.PLC)

The XPLC language is an addition to the standard Centroid programming language that provides more functionality. It must be used in conjunction with a "standard" program to solve more complex control applications that may have timing and arithmetic requirements, such as automatic tool changers. This manual assumes that one is familiar with the material in the standard PLC programming manual.

In order to write effective programs one must understand the way the standard program and the XPLC programs interact. When the phrase "standard" program is used, it refers to the program that is being executed on the motion control board. When the phrase "XPLC" program is mentioned it refers to the program that is executing on the computer, or PC.

Figure 2 shows the interaction between the standard and XPLC programs. Note that all INP, OUT, and MEM locations can be read in both a standard program and by an XPLC program. The figure below shows which program has control to write or change these bits. There are several questions that one may have at this point.

Why would any of the inputs need written or changed by a program? The answer is that not all of these INP bits are actually physical inputs. In fact, INP33-INP48 are actually M-function outputs that are controlled by M94 and M95 commands in M&G code programs.

How can operations in an XPLC program change or write bits that are controlled by the standard program? The answer is that the XPLC program must write to or change a location, often a memory bit, that it, XPLC, has access to. The standard program must read that location and write the corresponding bit. For example, suppose the XPLC program wanted to turn on the AUX1 LED on the jog panel when a timer had expired so as to alert the operator of a potential problem. The AUX1 LED is mapped to OUT49, which can only be changed by the standard program. In this case, the XPLC program would set, say MEM49, when it wanted to turn on the LED. The standard program would then set OUT49 based upon the value of MEM49. The actual code is listed below:

The lines common to both programs:

```
AUX1_LED      IS OUT49
XPLC_AUX1_LED IS MEM49
```

The standard PLC program lines:

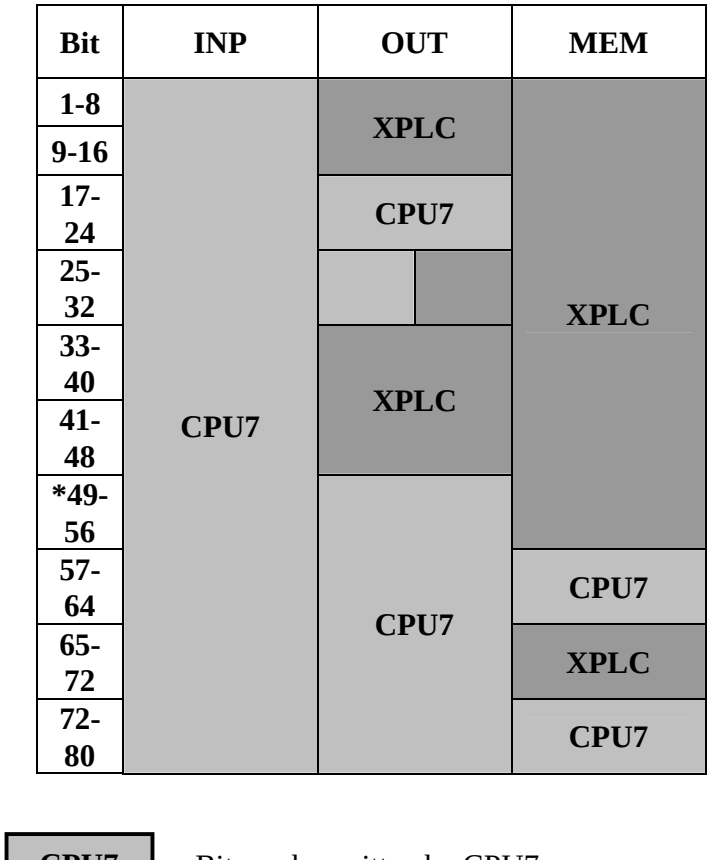
```
AUX1_LED = XPLC_AUX1_LED
```

The XPLC program lines:

```
IF T1 THEN (XPLC_AUX1_LED)
```

Not every location that can be written by a standard PLC program can be mapped to another location as there are not enough locations. It would be a very desirable goal to be able to map all these locations as it would mean only one program, the XPLC program, would ever need to be changed. Not so obvious is that even if there were enough memory locations to reach this goal, there are still underlying hardware and software conditions that require both programs to be maintained in certain situations. Later in the manual, a set of standard and XPLC programs will be provided that allow most all changes and additions to be made by changing only the XPLC program.

Figure 2. CPU7 vs. XPLC write control.



CPU7 Bit can be written by CPU7 program

XPLC Bit can be written by XPLC program

*** MEM49 is used by the XPLC program to communicate to the standard program that the XPLC program is being used. Therefore, MEM49 must be set by the XPLC program for the programs to work together.**

Compiling and naming of XPLC programs.

The XPLC program, like the standard program, is text based. The structure is less stringent than a standard program. In other words, spaces are usually not required and individual lines may be broken across several lines without problems. Blank lines are also permitted.

Just like standard plc programming, an XPLC program must be compiled and must reside in a certain directory and be given a predetermined name.

A program is compiled by supplying the name of the XPLC source program to the compiler, XPLCCOMP.EXE.

Assuming the XPLC source program was named XPLC.SRC, the following command would be used to compile the program, with **boldface** type indicating what would be typed.

```
C:\PLC>XPLCCOMP XPLC.SRC
```

If the compilation is successful, a message similar to the following is displayed:

```
XPLCCOMP v. 1.00 - XPLC compiler
Copyright 2001-2003.

Compilation successful
Program size: 1
```

XPLCCOMP will create the compiled plc file named `XPLC.PLC.` If the compilation was not successful, there will be error messages displayed. See the section **Compilation Errors** for more information.

In order for this compiled program to be used by the system, it must be named `PC.PLC` and reside in the `C:\PLC` directory. The following command can be used to accomplish this:

```
C:\PLC>COPY XPLC.PLC PC.PLC
```

The system must be rebooted for any changes in the XPLC program to take effect.

The program that executes the PC.PLC program is PCPLC.EXE, which is located in the C:\PLC directory and is automatically called at startup by commands in the AUTOEXEC.BAT file.

The XPLC program has the same syntax for definition lines and comments as a standard program. For example:

```
X_MINUS    IS INP1 ; 0 = ok, 1 = limit tripped
```

XPLC Program lines are of the form

IF <*boolean_expression*> **THEN** <*actions*>

Standard programs are of the form

<*plc_bit*> = <*boolean_expression*>

It may be helpful to see how a line in a standard program would be written using a standard program, an XPLC program, and traditional relay ladder logic (RLL).

(1) Standard program

<*bit_type*> = <*boolean_expression* >

```
Lube     = CNC_program_running and not Lube_low
```

(2) XPLC program

IF < *boolean_expression*> **THEN** <*actions*>

IF `CNC_program_running and not Lube_low  THEN (Lube)`

(3) Relay Ladder Logic (RLL) program

<*boolean_expression*> <*actions*>

Note here that an “IF” is represented graphically as |-- , with THEN being interpreted as a coil connection at the end –(.

All three of these examples accomplish the same thing, namely that if a CNC program is running AND the low lube signal is not triggered, the lube will turn on.

The **<*boolean_expression*>** is a combination of the AND, OR, XOR and NOT operations be-

tween various inputs, outputs, memory locations, timer contacts, one-shot positive differential contacts, stage bits, and comparisons between integer word types.

The boolean operators are:

AND *or* **&**
OR *or* **|**
NOT *or* **!**
XOR *or* **^**

which are used with bit type operands, and

== (equal)
!= (not equal)
<less than
<=less than or equal
>greater than
>=greater than or equal

which are used with integer type operands, namely word memory and timer current values.

XPLC Bit Types

(the nn represents a number from 1-256):

INPnn - used for inputs
OUTnn - used for outputs
MEMnn - used as a temporary storage location or memory bit

Note that these are the same as used in standard programs.

STGnn - used for stage bits, explained later
PDnn - used for one-shot positive differential (leading edge one-shot)
Tnn - used for timer contact values

XPLC Integer Types

Wnn - a number that ranges between -2147483648 to +2147483647
TMRnn - how long a timer has been on, measured in 10ms (0.01sec) increments.
FLT - an internal fault number can have one of the following values:
0 = No error
1 = Stack fault
2 = Division by Zero fault
8 = Illegal instruction
<***numerical_expression***> – can be just a plain number, such as 78, or a combination of

plain numbers and other integer types using *, /, -, and +, along with parentheses. For example W1 + W2/(3*TMR7), or 5/8, etc.

For those who may be familiar with relay ladder logic (RLL) programs, several examples with equivalent XPLC syntax are demonstrated below. For those who may not be familiar with RLL programming, the convention is that

X is used for an input type.
Y is used for an output type.
C is used for a memory bit.

The following XPLC definitions will be used in these examples.

X0 IS INP1
X1 IS INP2
X2 IS INP2
Y0 IS OUT1.

The terms **on, set, high, closed,** and **true** are normally used to denote a **logic (1)** and the terms **off, reset, low, open,** and **false** are normally used to denote a **logic (0)**. Whether a bit is "open" or "closed", "high" or "low", etc., depends upon the particular bit type. When using the aforementioned terms for logic 0 or 1, it is mainly used to differentiate between two different states.

In conventional RLL programming and hardware, an electrical input that is closed is a "1" and an input that is electrically open is a "0". In this system, however, a physical input on the PLC hardware that is electrically closed is a "0" whereas a physical input on the PLC hardware that is electrically open is a "1". In conventional RLL programming and with this system, an output that is "1" is considered on.

In order not be confusing for those who may already be familiar with RLL programming, the explanations below will assume conventional industry standard RLL conventions as noted above- just keep in mind that these industry standard conventions are opposite of the actual working in this PLC system for physical inputs on the PLC hardware.

Example 1. A simple one contact statement.


```
IF      X0                                          THEN    (Y0)
```

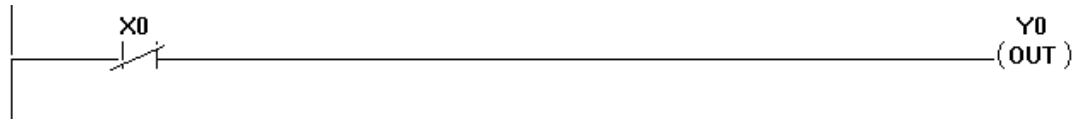
This is the most basic program statement. What it means is that if X0 is on, then turn on Y0. If X0 is off, turn off Y0. There are only two combinations that are possible using one con-

tact of logic. These combinations are in the table below. This kind of table is referred to as a "truth table".

Table 1. Truth table for example 1.

X0	Y0
0	0
1	1

Example 2. The basic NOT statement.



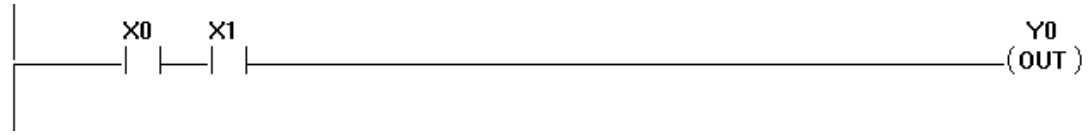

```
IF    !X0                                    THEN    (Y0)
```

This line of program has the opposite meaning. It is an example of the NOT operation in XPLC programming. What it means is that if X0 is not off, then turn off Y0. If X0 is off, turn on Y0. Again, there are only two possibilities:

Table 2. Truth table for Example 2.

X0	Y0
0	1
1	0

Example 3 – Basic AND statement




```
IF    X0 & X1                                THEN    (Y0)
```

This line of program is the basic AND operation. What it means is that if X0 is on **AND** X1 is on then Y0 will be on. Otherwise, Y0 will be off.

There are four different possibilities using logic with two contacts. These combinations and the result of Y0 are shown in the table 3 below.

Table 3. Truth table for example 3 (AND).

X0	X1	Y0
0	0	0
0	1	0
1	0	0
1	1	1

Example 4. The basic OR statement


```
IF   X0 | X1                                    THEN   (Y0)
```

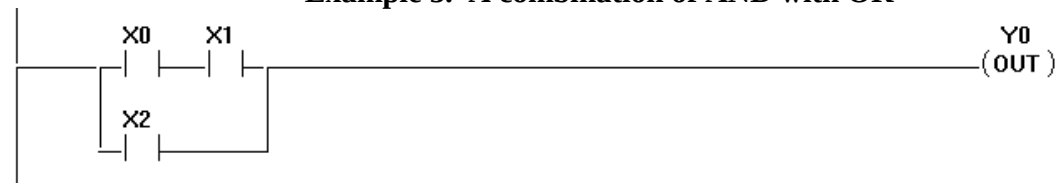
This is the basic OR operation. What it means is that if X0 **OR** X1 is on, then Y0 will be on. Otherwise, Y0 will be off.

Again, there are four different combinations that are possible. The truth table is in Table 4.

Table 4. Truth table for example 4 (OR).

X0	X1	Y0
0	0	0
0	1	1
1	0	1
1	1	1

Example 5. A combination of AND with OR




```
IF   X0 &  X1 | X2                                THEN  (Y0)
```

This example shows a simple combination of logic. When using combinations of AND, OR, XOR, and NOT, there are rules that determine the order in which operations occur. This is simi-

lar to the way there are rules when combining numbers and mathematical operations. It is easy to understand when it is realized that:

AND is like multiplication, OR is like addition, and like math, multiplication comes before addition. One could also say that multiplication has precedence over addition.

For example, imagine X0 being 2, X1 being 3, and X2 being 4. In this scenario, the line above would be written as 2 x 3 + 4 (= 10). This is helpful when trying to understand and determine what the result will be.

A program statement composed of three different contacts has eight different combinations. The table below, often referred to as a "truth" table, shows the outcomes of all these combinations.

Table 5. Truth table for example 5.

X0	X1	X2	Y0
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Example 6. Another combination of OR with AND


```
IF   X0 & (X1 | X2)                          THEN   (Y0)
```

Using the analogy to math, the way this logic is calculated is similar to the way that 2 * (3 + 4) is calculated. Note that in both the XPLC program and in the math example, parenthesis are needed to override the normal rules. Just like in math, the result is usually not the same, see Table 6 and compare it to Table 5.

Table 6. Truth table for example 6.

X0	X1	X2	Y0
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1

1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Example 7. The basic XOR statement


```
IF      X0 ^ X1                                  THEN    (Y0)
```

This example shows the XOR operation. What it means is that if one of X0 or X1 is on (1) , but not both, then Y0 will be turned on (set to 1). Otherwise, Y0 will be off (reset to 0). The truth table is below. Another way of stating this is that if both of the inputs do not have the same value, then the result is true.

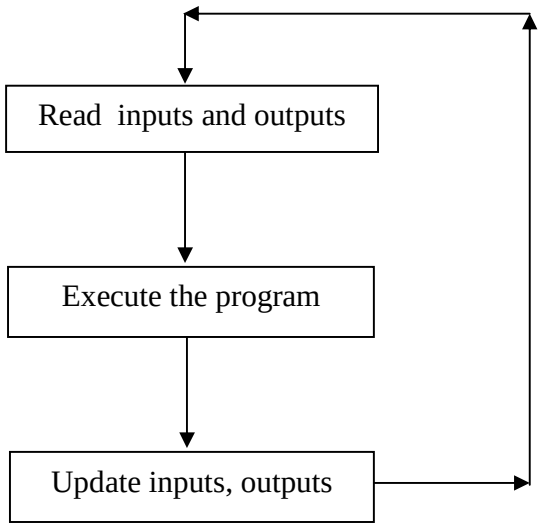
Table 7. Truth table for example 7.

X0	X1	Y0
0	0	0
0	1	1
1	0	1
1	1	0

Program execution

Before continuing with examples of some of the more advanced contact types, it will be helpful to review and understand how a program is executed. Programs are composed of a series of lines that are executed from top to bottom continuously. The XPLC execution occurs 256 times a second, that is, the entire program is executed from top to bottom 256 times a second. Thus, the first concept to understand is that the program is constantly being executed. One execution of the program from top to bottom is referred to as a **pass**. Further explanations will refer to passes, such as the first pass of the program, or the second pass of the program, etc.

The second concept to understand is how the inputs, outputs, memory, and other bits are updated **as the program executes**. As a review from the standard ??plc manual??, the complete plc program execution follows this cycle:



When the inputs and outputs are read, a snapshot of the their current state is made and placed

into a **copy buffer**. The copy buffer contains a copy of the input and output states as they were **at the start of that pass of program execution.**

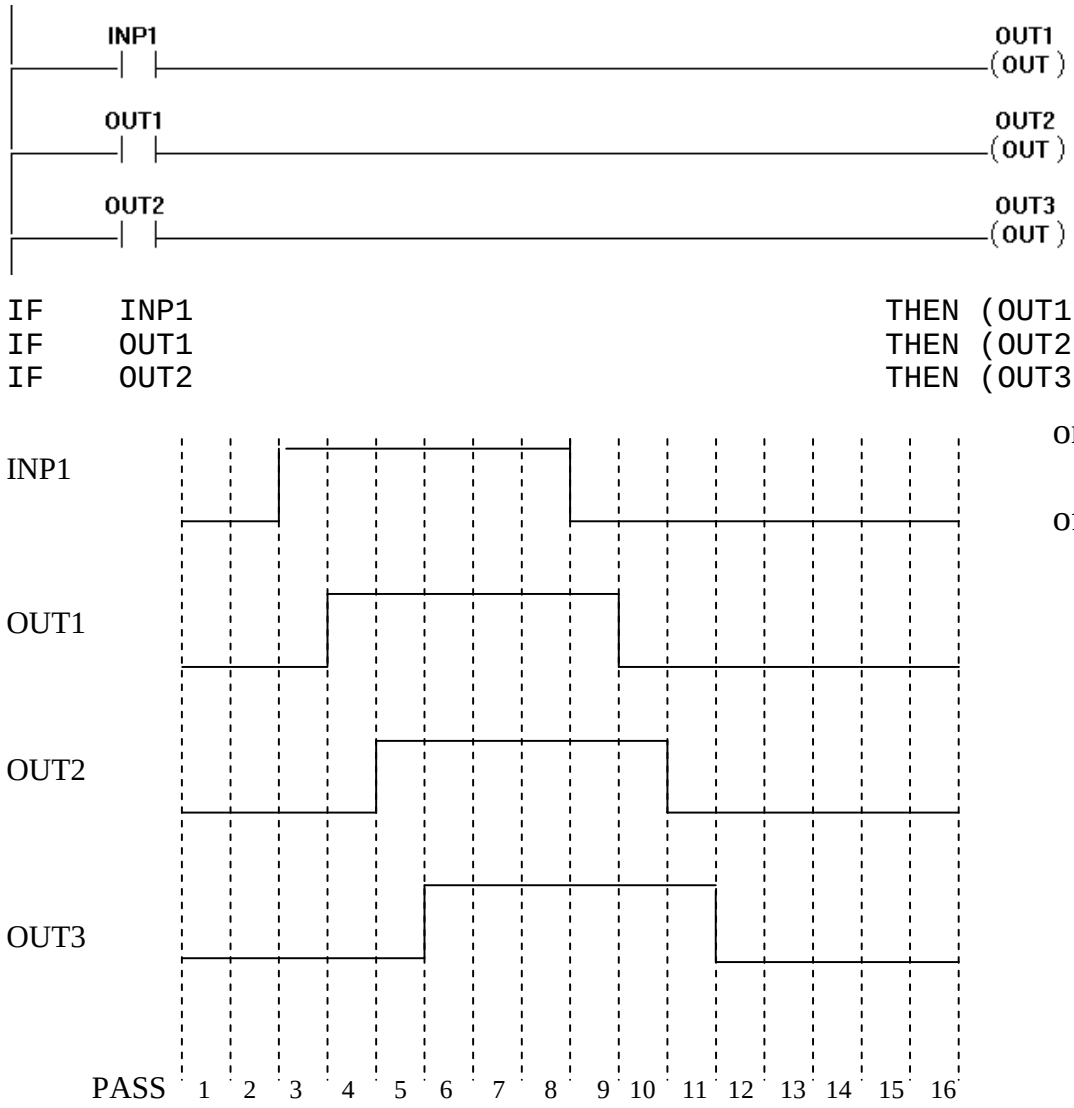
As the program is executing, <*boolean_expression*> that reference the inputs and outputs (INPnn and OUTnn) use the value from the copy buffer to determine what the state of the contact is.

There is another buffer used during execution, which is referred to as the **image buffer**. The image buffer contains what is going to be the new value of the input or output after the program has finished execution. This is the place where, during execution of the program, the new values of the inputs and outputs will be stored.

Most of the problems and errors found when developing programs are directly related to these principles, namely that program execution is continuous from top-to-bottom and that input and output states are read from the copy buffer and written using the image buffer.

For other types, such as memory bits, stage bits, and word values, there is no buffer. Program statements that write these types will **immediately** change the value. To better understand this concept, some timing diagrams are presented below. The first example shows when outputs are updated. This would be the same if the OUTs were INPs. The second example shows the timing and updating when using MEM bits.

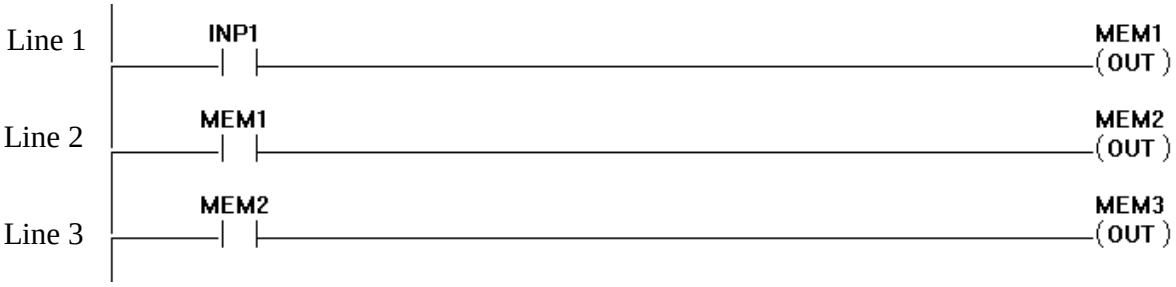
Example ??. Timing diagram of writing of outputs.



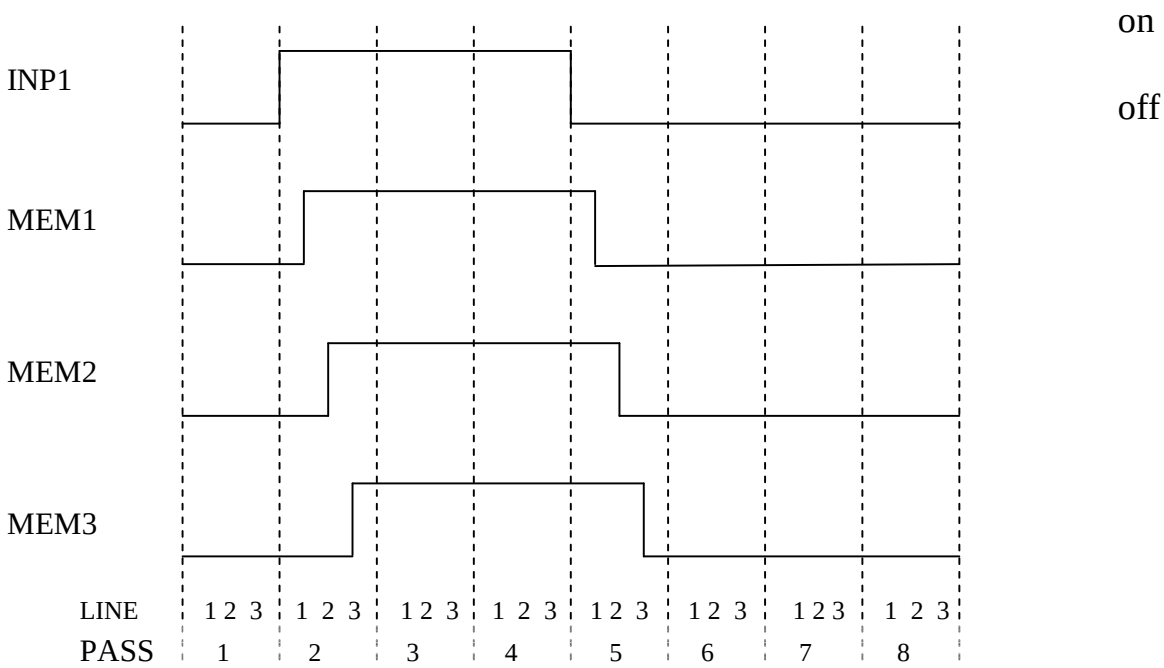
Hopefully this example has demonstrated how the inputs and outputs are read and written, and the effects this has on when they are updated.

Consider now an example of writing MEM bits. The same example would hold true if the MEM bits were replaced by STG or PD bits or for word memory assignments.

Example ??. Timing and states when writing MEM bits.




```
IF    INP1                    THEN (MEM1)
IF    MEM1                    THEN (MEM2)
IF    MEM2                    THEN (MEM3)
```

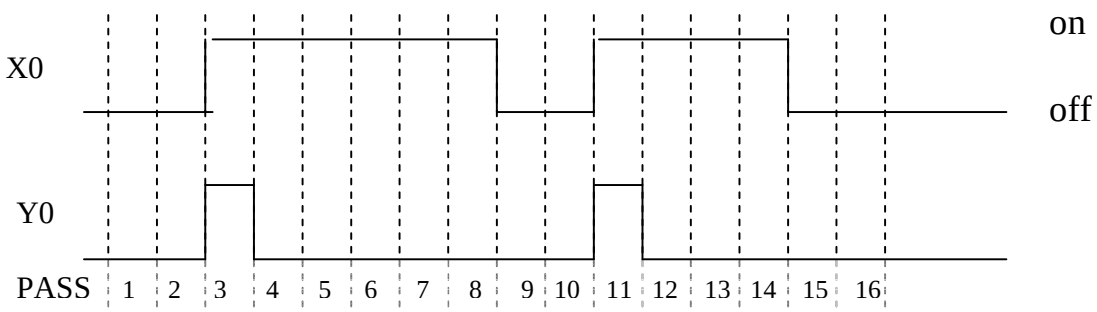


Note that the reason the MEM bit rising and falling edges are slightly skewed throughout pass #3 and pass #5 is to show that they don't actually get changed until that particular line of logic is executed.

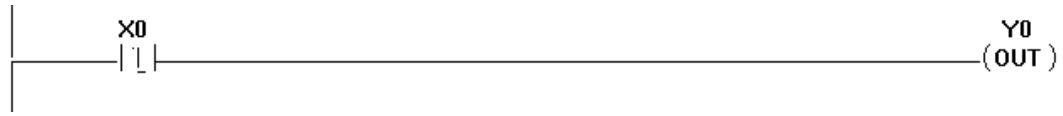
Example ??. The basic rising edge one-shot or positive differential PD


```
IF     X0                     THEN (PD1)
IF    PD1                     THEN (Y0)
```

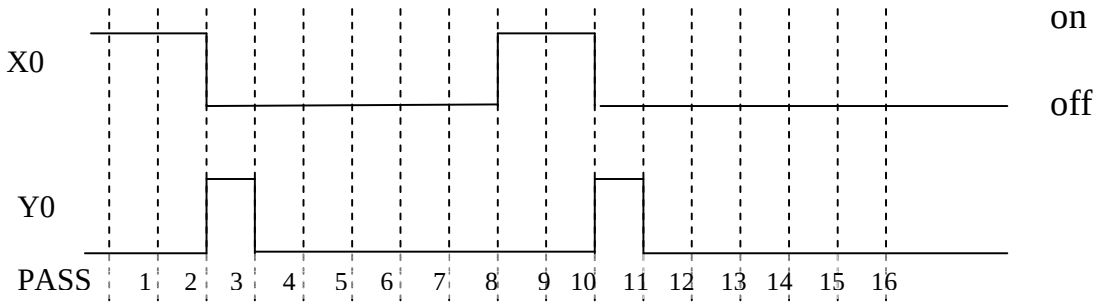
This example demonstrates the use of a positive differential, or rising edge one-shot type of contact. What it means is that if X0 is off one pass and then on the next pass, then Y0 will be turned on for one pass. Perhaps a timing diagram can better show this.



Example ??. The basic falling edge one-shot or negative differential PD




```
IF   !X0                              THEN (PD1)
IF   PD1                              THEN  (Y0)
```



One-shots are often used to toggle states without the side effect of causing rapid flickering of outputs. Specific applications of one-shots and when to use them will be covered in a later section of manual. One-shots are really nothing more than a convenience when writing programs. The same result can be obtained using just MEM bits and having the program keep track of the last state. Example ?? above can be written using MEM bits as such:

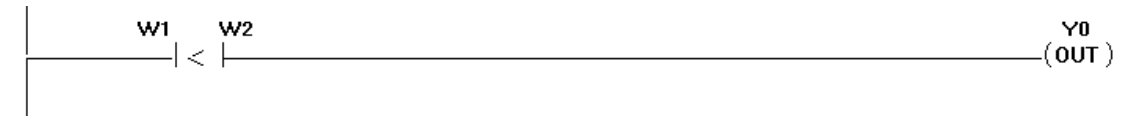
```
IF X0 & !MEM1 THEN (Y0)
IF X0         THEN (MEM1)
```

Note when using one-shots it takes another line of XPLC program to achieve the same effect as would be required in RLL programming. The format used for XPLC programming was used to cut down the number of different token types that are used. Note that an XPLC program allows an entire rung of logic to control the one-shot. A similar RLL program in this case would also require two rungs of logic.

Example ??. Basic comparison statements.


```
IF    W1 == W2                                    THEN (Y0)
```


```
IF    W1 != W2                                    THEN (Y0)
```




```
IF     W1 < W2                                    THEN  (Y0)
```


```
IF     W1 >= W2                                   THEN   (Y0)
```

The above examples demonstrate comparisons between integer type operands. What they mean is that if the comparison is true, then turn on Y0. Otherwise, turn it off. In the first program line above, if W1 = 3 and W1 = 4, then Y0 would be turned off.

Example ??. Basic calculations.

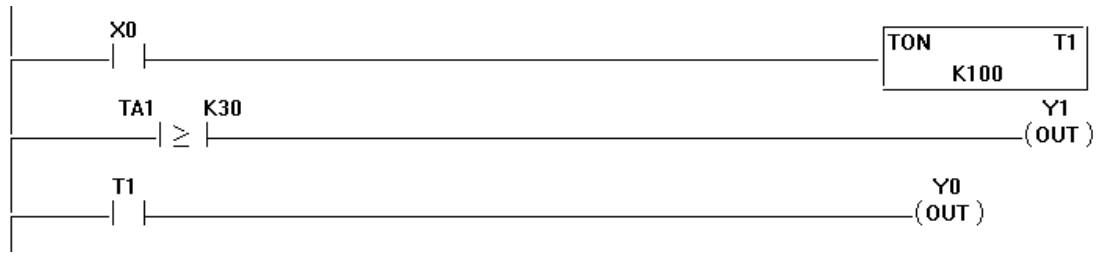
XPLC programming of mathematical operations is easier to accomplish than using RLL as it does not require the use of stacks, accumulators, and various math boxes. See below.


```
IF     X0                                THEN  W1 = 3 * 4
```


```
IF   X0                                  THEN  W1 = W1 + 1
```

When programming XPLC mathematical expressions, simply write the expression as it would be written on paper.

Example ??. Timers




```
IF    X0                                 THEN T1 = 100, (T1)
IF    TMR1 >= 30                         THEN           (Y1)
IF    T1                                 THEN    (Y0)
```

The programs above will turn on Y1 after X0 has been on continuously for at least 300 ms (0.3 seconds) in addition to turning on Y0 after 1.0 second.

Timers have four components associated with them, which are:

(1) A **preset value**. In the example above, the preset value is 100, or 1 second. The value is specified using the syntax Tnn = <*numerical_expression*> in an action statement. The preset value only needs to be set once. In a typical program, the preset value is set once in the beginning of the program.

(2) A **timer input** which, when switched on, causes the timer to start keeping track of elapsed time. If the timer input is off, the timer is reset to 0.

(3) A **current value** that returns how long the timer has been on. This is where the elapsed time is stored. The syntax for accessing the timer current value is to use a TMRnn reference in an integer comparison.

(4) A **timer contact** that is on when the current value >= preset value. The syntax for referencing a timer contact is to use a Tnn reference in a <*boolean_expression*>.

With XPLC programs there are no accumulating timers or up/down counters. However, the functionality of these typical RLL elements can be coded using a combination of word memory and timers with a bit of skill. Examples of up/down counters and accumulating timers are provided later in the manual.

Example ??. Executing multiple actions for the same boolean expression.


```
IF     X0                                    THEN (Y0),
                                                  (Y1),
                                                  (Y3)
```

In this example, when X0 is on, Y0, Y1, and Y3 are on. Otherwise, they are all off.

Understanding Stages

Stages are a feature of XPLC programming that helps write structured programs. Stages also aid in program development and maintenance. As used in XPLC programming, they are closely related to the RLL programming of master control relays (MCS and MCR coils).

Any IF statement of an XPLC program may be preceded by a STGnn. This marks all program lines after the STG, until another STG or the end of the program occurs, as belonging to

that stage.

Example ??.

```
1:  STG1
2:
3:  IF INP1 THEN (Y0)
4:  IF INP2 THEN (Y1)
5:
6:  STG2
7:
8:  IF INP3 THEN (Y2)
9:  IF INP4 THEN (Y3)
```

In the example above, lines 2-5 are part of STG1 and lines 7-9 are part of STG2.

A stage can be ON or OFF. It has an associated internal memory bit that determines whether it is ON (1) or whether it is OFF (0). This memory bit can be written (turned on and off) by using certain action statements and can be read by referencing it in a <*boolean_expression*>. For example:

```
IF INP1 THEN (STG1)
```

This is an example of how the STG status is turned on or off. If INP1 is on, then STG1 would be turned on. If INP1 is off, STG1 would be turned off.

```
IF STG1 THEN (Y0)
```

This is an example of reading a stage status. Here, if STG1 is on, then Y0 is turned on. If STG1 is off, then Y0 would be turned off.

How stages work in program execution

When the XPLC program is being executed and it encounters a STGnn, the executor marks that stage as the **active stage.** If the active stage is ON, then the execution of the program continues normally until the next stage or end of program. If the stage is OFF, the effect is that **all the <*boolean_expression*> are considered false for that stage**.

```
STG1
IF INP1 THEN (Y0)
```

In this example, if STG1 is OFF, then Y0 would be turned off **regardless of whether INP1 was on or off**. If STG1 is ON, then Y0 would be on if INP1 is on and off if INP1 is off.

STG1 is ON, by default, when the control system is initialized. STG2-STG256 are OFF at system initialization. This allows a starting point for initialization without having to explicitly turn it ON.

Note that like PD contacts, stages are a feature that can also be implemented in another way using MEM bits and additional program lines. There is no requirement that stages are even used in an XPLC program, but experienced PLC programmers use them to make their programs easier write and easier to understand and maintain.

Specific uses and application of stages are covered more in-depth later in the manual.

<Actions>

<*Actions*> are the various commands that can be executed depending upon the value of a <*boolean_expression*>. So far, most of the examples in this manual have used one specific type of action- the output coil. The ones that have not were the basic calculations and the timer example. All of the possible actions are explained below.

Output coil ( )

The output coil action has several forms. The most common form is used to turn on or off a specified bit and is used with **INP**, **OUT**, **MEM**, or **STG** bits.

```
IF INP1 THEN (OUT1)
```

What it means is that if INP1 is ON, then turn on OUT1. If INP1 is off, then turn off OUT1.

```
IF INP1 & !INP3 | MEM3 | W1 <= 3 THEN (OUT1)
```

Here, if the entire <*boolean_expression*> "INP1 & !INP3 | MEM3 | W1 <= 3" is true, then OUT1 is on. Otherwise, it is off.

When an output coil is used with a **PD** bit, as in

```
IF INP1 THEN (PD1)
```

then the meaning is:

if <*boolean_expression*> is true and on the previous pass it was false, then turn on PD. Otherwise, turn off PD.

When an output coil is used with a **T** or **TMR** bit, it means to connect the value of <*boo-*

lean_expression> to the **timer input**. Thus, if <*boolean_expression*> is true, then the timer updates the **current value** with the elapsed time. If <*boolean_expression*> is false, **the current value is set to 0**..

Other actions

All the remaining types of action statements work in this way:

if <*boolean_expression*> is true, then **execute** the command.
if <*boolean_expression*> is false, then **do not execute** the command.

SET, RST

These actions turn on and turn off **INP**, **OUT**, **MEM**, and **STG** bits.

```
IF INP1 THEN SET OUT1
```
If <*boolean_expression*> is true, then the bit is turned on.

```
IF INP1 THEN RST OUT1
```
If <*boolean_expression*> is true, then the bit is turned off.

*Note that in these two examples that if <boolean_expression> is false**, nothing happens**. It is a mistake to think that if <boolean_expression> is false, the SET command is turned into a RST command, or vice versa. The same is true for any action that is not an output coil.*

The line:

```
IF INP1 THEN (OUT1)
```

is the same as:

```
IF  INP1 THEN SET OUT1
IF !INP1 THEN RST OUT1
```

= (Assignment)

The assignment command is used to assign a <*numerical_expression*> to a word memory location **W** or to set a timer **preset value**, provided <*boolean_expression*> is true. If <*boolean_expression*> is false, no assignment is made.

```
IF INP1 THEN W1 = 60 * 10
IF INP1 THEN T1 = 500
```

In the first example, if INP1 is on, then W1 is assigned the value 600 (60 * 10).

In the second, if INP1 is on, then T1 **preset value** is assigned 500 units, or 5 seconds.

JMP STGnn

The JMP command resets the **active** stage and sets STGnn.

```
STG1

IF INP1 THEN JMP STG2
```

Here, if INP1 is on, then the JMP STG2 command will reset STG1 and set STG2. Note that if the active stage is reset execution continues normally for lines in that STG.

WTB Wnn MEMnn
WTB Wnn OUTnn

The WTB commands write the lower eight bits of the Wnn word to a series of MEM or OUT bits. The least significant bit is written to nn and the most significant is written to nn+7

```
IF 1==1 THEN W1 = 170, WTB W1 OUT41
```

After the above line is executed,

OUT48	OUT47	OUT46	OUT45	OUT44	OUT43	OUT42	OUT41
1	0	1	0	1	0	1	0

170 (Decimal) = AA (Hexadecimal) = 10101010 (Binary)

BCDWnn
BIN Wnn

The BCD Wnn command converts the value in Wnn to binary coded decimal (BCD) format. The BIN Wnn command converts the value in Wnn to binary, assuming that it was in BCD format.
LDT Wnn
LTS Wnn
LMT Wnn
LCP Wnn

All of the above commands are for support of automatic tool changers.

LDT Wnn will load the value of the last tool number into Wnn. Tool numbers are sent to the XPLC system when an M107 command is executed in an M&G code program.

LTS Wnn will load the value of the tool in the spindle into Wnn. This value comes from the CNC10.JOB file at CNC10 startup.

LMTWnn will load the value of the maximum number of tools into Wnn. This value comes from CNC10 Machine Parameter 161 at startup.

LCPWnn will load the value of the tool carousel position into Wnn. This value comes from the CNC10.JOB file at startup.

LSRWnn

This command will load into Wnn a value that can be checked to see if the CNC10 software is currently running. Its main use is to disable carousel indexing when CNC10 is not running since if the carousel position changed, CNC10 would not be able to monitor the new position and save it in the CNC10.JOB file.

```
IF 1==1     THEN LSR W1
IF W1 == 17 THEN (Disable_Tool_Indexing)
```

In this example, Disable_Tool_Indexing is a memory bit that would be used later in the XPLC program to prevent tool indexing via Aux keys.

LP0 Wnn **– LP9** Wnn

These ten commands are used to load the value of CNC10 Machine parameters into Wnn.

LP0 loads CNC10 Machine Parameter 170 into Wnn.
LP1 loads CNC10 Machine Parameter 171 into Wnn.
…
LP9 loads CNC10 Machine Parameter 179 into Wnn.

Machine Parameters 170-179 can have values between 0 – 65535.

These commands have many applications, including changing XPLC program behavior based upon certain values and toggling the logic value of an input.

Both the standard and XPLC programs may differ according to the physical hardware that constitutes the system. Such hardware considerations are mainly:

(1) PLC hardware (PLC3/3, PLC 15/15, RTK2, SERVO3IO, PLCIO2, RTK3, etc.)
(2) Jog Panel hardware: None, keyboard, M39 pendant, Uniconsole-2, etc.

A basic set of standard and XPLC source programs are presented below. These programs are targeted to a system composed of a SERVO3IO used with a Uniconsole-2 jog panel.

When listing programs and in order to aid readability, the `Courier typeface` is used for program lines and Arial typeface is used for the more meaningful comments.

The standard program.

```
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *
; *  File:    BASECPU1.SRC
; *  Purpose: STANDARD PLC program for SERVO3IO/UNICONSOLE-2
; *
; *          Works in conjunction with XPLC program BASEXPC1.SRC
; *
; *          CNC Configuration Settings
; *              Jog Panel Type:   Uniconsole-2
; *              PLC Type:         Normal
; *
; *          CNC Parameter Settings
; *              P31   = 1 or 2 (for COM1 or COM2 spindle control)
; *
; *              P177  = 0.0  Normal behavior
; *                    = 1.0  Fault Override in effect
; *
; *              P178  Bit flags
; *                    LubeNONC         ; P178 Bit 0  (1)
; *                    SpindleNONC      ; P178 Bit 1  (2)
; *                    NoReverseSpindle ; P178 Bit 7  (128)
; *                    SpinRangeNONC    ; P178 Bit 9  (512)
; *
; *              P179  Lube timer settings  (see below for full explanation)
; *                    = 0  On when running a job
; *                    MMMSS   Off for MMM minutes, On for SS seconds
; *                            if SS = 0, On for at least MMM minutes
; *
; *
; *   () Support for probing and digitizing.
; *   () Handling of both NO/NC spindle drive faults and automatic reset.
; *   () Handling of both NO/NC low lube faults.
; *   () Handling of drive faults.
; *   () Support of two limits switches each for all three axes
; *   () Support for NO/NC spindle high/low range that can
; *      optionally reverse direction.
; *   () Support for drawbar unclamping.
; *   () Control of flood coolant in both automatic (M8) and manual modes.
; *   () Control of mist coolant in both automatuc (M7) and manual modes.
; *   () Automatic/manual control of a spindle brake.
; *   () Lube pump timing and control.
; *   () Fault override logic via parameter setting.
; *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;
;  By way of standard practice, the definitions for all the INP bits are
;  grouped together as well as those for OUT, and MEM bits.
;
;  If a bit is unused in the program, it is prefixed with a u_.
```

```
;
; ***************************************************************************
;                              INPUT DEFINITIONS
; ***************************************************************************
;
;  Remember that for the SERVO3IO that an input that is electrically
;  closed is a 0.  An input that is electrically open is a 1.
;
E_stop               IS INP1    ; 0 = Normal    1 = E_STOP ** HARD CODED **
Probe_input          IS INP2    ;
Spindle_range_in     IS INP3    ;
Spindle_ok           IS INP4    ; 0 = fault     1 = ok
Probe_not_detected   IS INP5    ; 0 = probe     1 = no probe
Lube_Fault_In        IS INP6    ;
X_minus              IS INP7    ; 0 = ok        1 = Tripped
X_plus               IS INP8    ; 0 = ok        1 = Tripped
Y_minus              IS INP9    ; 0 = ok        1 = Tripped
Y_plus               IS INP10   ; 0 = ok        1 = Tripped
Z_minus              IS INP11   ; 0 = ok        1 = Tripped
Z_plus               IS INP12   ; 0 = ok        1 = Tripped
Servo_fault          IS INP13   ; 0 = Drive ok  1 = Drv Fault ** HARD CODED **
Tool_Release         IS INP14   ; 0 = pressed   1 = not pressed
Zero_Speed           IS INP15   ; 0 = at zero speed
PLC_ok               IS INP16   ; 0 = fault     1 = ok
;
; The lines above define the physical inputs that one would wire to
; on the SERVO3IO. The reason that PLC bits are defined in this way
; is because it makes the program easier to understand.
;
; The ** HARD CODED ** comment refers to the fact that on the SERVO3IO,
; INP1 must be the emergency stop input because the SERVO3IO performs special
; processing when it sees the E_stop input.  The other ** HARD CODED ** input
; is INP13.  There is no physical connection to this input as it is internal
; to the hardware.   The SERVO3IO will, however, cause INP13 to change
; to a 1 when there is a fault detected, such a sif the SYNC or DATA fibers
; are not connected.
;
;
; INP16 is reseved for PLC line check. It is an internal signal that has no
; physical comnnection.  If the PLC fiber connections CLK, RXS, or TXS
; are removed then the bit is turned off.
;
;
; INP17 - INP32 have no physical mapping for SERVO3IO.
;
u_inp17              IS INP17   ;
u_inp18              IS INP18   ;
u_inp19              IS INP19   ;
u_inp20              IS INP20   ;
u_inp21              IS INP21   ;
u_inp22              IS INP22   ;
u_inp23              IS INP23   ;
u_inp24              IS INP24   ;
u_inp25              IS INP25   ;
u_inp26              IS INP26   ;
u_inp27              IS INP27   ;
u_inp28              IS INP28   ;
u_inp29              IS INP29   ;
u_inp30              IS INP30   ;
u_inp31              IS INP31   ;
u_inp32              IS INP32   ;
;
; M94/M95 Mappings
;
;
; The M94/M95 mappings are interfaces to M&G code programming.
; Turning on and off these bits is how certain M-functions work.
;
; In M&G code programming, the command
;
;    M94/1 turns on  INP33
        ;    M95/1 turns off INP33
```

```
;
;    M94/2 turns on  INP34
;    M95/2 turns off INP34
;    ...
;    ... and so on
;    ...
;    M94/16 turns on  INP48
;    M95/16 turns off INP48
;
;  These pre-defined M-functions are:
;
;  M3 (Spindle CW)
;    M95/2
;    M94/1
;
;  M4 (Spindle CCW)
;    M95/1
;    M94/2
;
;  M5 (Spindle Off)
;    M95/1/2
;
;  M7 (Flood Coolant Off, Mist Coolant On)
;    M95/3
;    M94/5
;
;  M8 (Mist Coolant Off, Flood Coolant On)
;    M95/5
;    M94/3
;
;  M9 (Flood Coolant Off, Mist Coolant Off)
;    M95/3/5
;
;  M10 (Clamp On)
;    M94/4
;
;  M11 (Clamp Off)
;    M95/4
;
;  This program does not use a clamp but its definition
;  is included here because it has a predefined meaning.
;
;  ** Note that M6 (Tool Change) and M39 (Air Drill) commands
;  ** also have pre-defined meanings which affect one or more of these
;  ** bits.  See the M-series Operators Manual for more details.
;
M3                  IS INP33   ; Map to M94/1  M95/1 (Spindle CW)
M4                  IS INP34   ; Map to M94/2  M95/2 (Spindle CCW)
M8                  IS INP35   ; Map to M94/3  M95/3 (Flood On)
M10                 IS INP36   ; Map to M94/4  M95/4 (Rotary Clamp)
M7                  IS INP37   ; Map to M94/5  M95/5 (Mist)
M94_M95_6           IS INP38   ; Map to M94/6  M95/6
M94_M95_7           IS INP39   ; Map to M94/7  M95/7
M94_M95_8           IS INP40   ; Map to M94/8  M95/8
M94_M95_9           IS INP41   ; Map to M94/9  M95/9
M94_M95_10          IS INP42   ; Map to M94/10 M95/10
M94_M95_11          IS INP43   ; Map to M94/11 M95/11
M94_M95_12          IS INP44   ; Map to M94/12 M95/12
M94_M95_13          IS INP45   ; Map to M94/13 M95/13
M94_M95_14          IS INP46   ; Map to M94/14 M95/14
M94_M95_15          IS INP47   ; Map to M94/15 M95/15
M94_M95_16          IS INP48   ; Map to M94/16 M95/16
;
;  Jog panel AUX keys
;
Brake_key           IS INP49   ; 1 = pressed
Aux_2_key           IS INP50   ; 1 = pressed
Aux_3_key           IS INP51   ; 1 = pressed
Aux_4_key           IS INP52   ; 1 = pressed
Aux_5_key           IS INP53   ; 1 = pressed
```

```
Aux_6_key            IS INP54   ; 1 = pressed
Aux_7_Key            IS INP55   ; 1 = pressed
Aux_8_key            IS INP56   ; 1 = pressed
Aux_9_key            IS INP57   ; 1 = pressed
Mist_Key             IS INP58   ; 1 = pressed
;
; INP59 -INP62 are physical inpus located on the CPU.
; They should not be used.
;
u_inp59              IS INP59   ;
u_inp60              IS INP60   ;
u_inp61              IS INP61   ;
u_inp62              IS INP62   ;
;
;                                 High     Med-High  Med-Low  Low
;
Mid_range            IS INP63   ;    0          1        1      0
High_Low_Range       IS INP64   ;    0          0        1      1
;
; INP65 is set by CNC software when running a job or in MDI mode.
;
CNC_program_running  IS INP65   ; 0 = stopped   1 = running
;
; Spindle control related keys
;
Spindle_mode_switch  IS INP66   ; 0 = manual    1 = auto
Spindle_Dir_Key      IS INP67   ; 0 = CCW       1 = CW
Spindle_start_key    IS INP68   ; momentary     1 = start spindle
Spindle_stop_key     IS INP69   ; momentary     1 = stop spindle
;
Aux_11_key           IS INP70   ; Aux_11_Key
Aux_12_key           IS INP71   ; Aux_12_key
;
; Coolant related keys
;
Coolant_mode_switch  IS INP72   ; 0 = manual    1 = auto
Flood_Key            IS INP73   ; Flood_Key
;
; Misc keys
;
Cycle_cancel         IS INP74   ; 1 = pressed
Cycle_start          IS INP75   ; 1 = pressed
Tool_check_key       IS INP76   ; 1 = pressed
Pause                IS INP77   ; 1 = Pause
Block_mode_key       IS INP78   ; 0 = auto      1 = block mode
Rapid_over_key       IS INP79   ; Not present on a Uniconsole-2
;
;*****************************************************************************
;                           OUTPUT DEFINITIONS
;*****************************************************************************
Mist                 IS OUT1    ; 0 = off       1 = on
Lube                 IS OUT2    ; 0 = off       1 = on ** Hard Coded **
Flood                IS OUT3    ; 0 = off       1 = on
Brake                IS OUT4    ;
VFD_reset            IS OUT5    ; 0 = Normal    1 = Reset
u_out6               IS OUT6    ;
Drawbar_Sol          IS OUT7    ;
;
; OUT8-OUT15 have no physical mapping for SERVO3IO
;
u_out8               IS OUT8    ;
u_out9               IS OUT9    ;
u_out10              IS OUT10   ;
u_out11              IS OUT11   ;
u_out12              IS OUT12   ;
u_out13              IS OUT13   ;
;
; OUT14 and OUT15 are used here as memory bits.
; On other PLC hardaware, such as PLC 15/15, RTK2, PLCIO2
; They are the physical ouputs that control spindle
; direction and enable.
;
Spindle_Enable       IS OUT14   ;
      Spindle_Dir_Out      IS OUT15   ;
```

```
;
;  OUT16 is an internal bit that must be set
;  when INP16 (PLC fault INP) is set.
;
Plc_fault_out        IS OUT16   ; 0 = normal    1 = fault
;
Spindle_speed0       IS OUT17   ; Reserved for 8/12 bit spindle speed
Spindle_speed1       IS OUT18   ; Reserved     "
Spindle_speed2       IS OUT19   ; Reserved     "
Spindle_speed3       IS OUT20   ; Reserved     "
Spindle_speed4       IS OUT21   ; Reserved     "
Spindle_speed5       IS OUT22   ; Reserved     "
Spindle_speed6       IS OUT23   ; Reserved     "
Spindle_speed7       IS OUT24   ; Reserved     "
Spindle_speed8       IS OUT25   ; Reserved for 12 bit spindle speed
Spindle_speed9       IS OUT26   ; Reserved for 12 bit spindle speed
Spindle_speed10      IS OUT27   ; Reserved for 12 bit spindle speed
Spindle_speed11      IS OUT28   ; Reserved for 12 bit spindle speed
;
; OUT29-OUT40 have no physical mapping for SERVO3IO
;
u_out29              IS OUT29   ;
u_out30              IS OUT30   ;
u_out31              IS OUT31   ;
u_out32              IS OUT32   ;
u_out33              IS OUT33   ;
u_out34              IS OUT34   ;
u_out35              IS OUT35   ;
u_out36              IS OUT36   ;
u_out37              IS OUT37   ;
u_out38              IS OUT38   ;
u_out39              IS OUT39   ;
u_out40              IS OUT40   ;
;
;  IF MEM49 = 1 THEN OUT41-OUT48 ARE XPLC PROGRAMMABLE
;  IF MEM49 = 0 THEN OUT41-OUT48 ARE 2 DIGIT BCD TOOL NUMBER
;
u_out41              IS OUT41   ;
u_out42              IS OUT42   ;
u_out43              IS OUT43   ;
u_out44              IS OUT44   ;
u_out45              IS OUT45   ;
u_out46              IS OUT46   ;
u_out47              IS OUT47   ;
u_out48              IS OUT48   ;
;
;  Jog panel LEDs
;
Brake_LED            IS OUT49   ;
Aux_2_LED            IS OUT50   ;
Aux_3_LED            IS OUT51   ;
Aux_4_LED            IS OUT52   ;
Aux_5_LED            IS OUT53   ;
Aux_6_LED            IS OUT54   ;
Aux_7_LED            IS OUT55   ;
Aux_8_LED            IS OUT56   ;
Aux_9_LED            IS OUT57   ;
Mist_LED             IS OUT58   ;
;
; OUT59 - OUT62 are on the CPU
; They should not be used.
;
u_out59              IS OUT59   ;
u_out60              IS OUT60   ;
u_out61              IS OUT61   ;
u_out62              IS OUT62   ;
;
Lubricant_low        IS OUT63   ; 0 = normal    1 = low lube
Drive_fault_out      IS OUT64   ; 0 = normal    1 = fault
Spindle_fault_out    IS OUT65   ; 0 = normal    1 = fault
Spindle_Auto_LED     IS OUT66   ; 0 = manual    1 = auto
Spindle_Dir_LED      IS OUT67   ; 1 = CCW       0 = CW
u_out68              IS OUT68   ;
u_out69              IS OUT69   ;
```

```
;
Aux_11_LED           IS OUT70   ;
Aux_12_LED           IS OUT71   ;
;
Coolant_Mode_LED     IS OUT72   ; 0 = manual    1 = auto
Flood_LED            IS OUT73   ; 0 = off       1 = on
Stop                 IS OUT75   ; 0 = run       1 = stop
PLC_op_signal        IS OUT76   ; 0 = run       1 = operation in progress
Feed_Hold_LED        IS OUT77   ;
Block_mode_LED       IS OUT78   ;
Rapid_override       IS OUT79   ;

;*****************************************************************************
;                             MEMORY DEFINITIONS
;*****************************************************************************
ZERO                 IS MEM1    ;  Used as a memory bit that is always 0
PC_Brake_LED         IS MEM2    ;
PC_Aux_2_LED         IS MEM3    ;
PC_Aux_3_LED         IS MEM4    ;
PC_Aux_4_LED         IS MEM5    ;
PC_Aux_5_LED         IS MEM6    ;
PC_Aux_6_LED         IS MEM7    ;
PC_Aux_7_LED         IS MEM8    ;
PC_Aux_8_LED         IS MEM9    ;
PC_Aux_9_LED         IS MEM10   ;
PC_Mist_LED          IS MEM11   ;
PC_Aux_11_LED        IS MEM12   ;
PC_Aux_12_LED        IS MEM13   ;
Range_reverse        IS MEM16   ;
Fault_Override       IS MEM17   ;
Lube_fault           IS MEM43   ;
Probe_Fault          IS MEM44   ;
Coolant_Fault        IS MEM47   ;
PC_PLC_RUNNING       IS MEM49   ;
Auto_Spin_Mode       IS MEM50   ;
PC_Lube_Fault        IS MEM51   ;
Brake_mode           IS MEM52   ;
Auto_Coolant_Mode    IS MEM53   ;
Man_spin_dir         IS MEM55   ;
Man_spin_mode        IS MEM56   ;
Spindle_dir          IS MEM57   ;
PC_Spindle_Fault     IS MEM58   ;
PC_OUT59             IS MEM59   ;
PC_OUT60             IS MEM60   ;
PC_OUT61             IS MEM61   ;
PC_OUT62             IS MEM62   ;
PC_Stop              IS MEM65   ;
PC_Block_mode        IS MEM66   ;
PC_Rapid_override    IS MEM67   ;
Already_run          IS MEM68   ;
Already_run2         IS MEM69
Real_Spin_Dir_Key    IS MEM73   ;
Last_Spin_Dir_Key    IS MEM74   ;
Spindle_Dir_Change   IS MEM75   ;
Last_Spin_Dir        IS MEM76   ;
Last_rapid_over_key  IS MEM77   ;
;
; MEM78 and MEM79 have pre-defined meanings and are
; used for communication between the PLC program
; and the CNC software to send the appropriate
; commands to a SPIN232, which uses either
; COM1 or COM2 port for communication.  When CNC Machine Parameter
; 31 = 1 or 2, the CNC software will look at MEM78 and MEM79
; to output the requested direction and enable information.
;
; Note that the SERVO3IO has a built-in SPIN232 interface.
;
Spin232_Enable       IS MEM78   ;
Spin232_Dir          IS MEM79   ;
;
;*****************************************************************************
;*                              PROGRAM START
;*****************************************************************************
;
```

```
;  PLC_op_signal must be zero for motion control is to resume.
;  Here, we halt processing if a spindle command
;  is being executed (M3/M4) and Auto spindle mode is not selected or
;  a coolant command (M7/M8) is being executed and Auto Coolant
;  Mode is not selected.
;
;  Setting PLC_op_signal is what causes the
;  "Select Auto Spindle" or "Select Auto Coolant"
;  messages to appear in the CNC message window and for
;  program execution to pause until the signal is 0.
;
;  It will be forced to zero if the Fault_Override bit is set.
;  The XPLC program sets the Fault_Override bit which it
;  determines from CNC Machine Parameter 177 settings.
;
PLC_op_signal        = ( ( ( M3 OR M4 ) AND Man_spin_mode ) OR
                       ( ( M7 OR M8 ) AND / Auto_Coolant_Mode ) ) AND
                       / Fault_Override
;
;/////////////////  UNICONSOLE-2 SPINDLE CONTROL //////////////
;
;  A memory bit is used to simulate a "a real spindle direction key".
;  Most of the jog panel inputs are on when the key is pressed and off
;  when it is released.  On the Uniconsole-2 jog panel,
;  the spindle direction is composed of two keys that use one input.
;  The reason that the spindle direction logic is here in the
;  standard program is because only it can control the spindle
;  direction input (INP67).  Further, this bit cannot be set to a MEM
;  bit by the XPLC prOgram and written by the standard program because
;  of timing issues between the running of the XPLC and standard
;  programs.
;
;  The Uniconsole-2 jog panel firmware
;  sends a message if the spindle direction key (CW or CCW) that is
;  pressed is different than what was last sent. It initializes to CW.
;
;
Real_Spin_Dir_Key    = ( Spindle_Dir_Key & / Auto_Spin_Mode )
;
;  This line determines if there is a spindle direction change.
;  It does so on the following conditions:
;
;
;  (1) The spindle direction key has changed and the mode is manual., i.e,
;     do not allow the spindle direction key to change the direction in
;     Automatic spindle mode since the direction in automatic spindle
;     mode is determined by an M3 or M4 command.
;
;                    OR
;
;  (2) If in automatic spindle mode there has been a change between M3 and M4.
;
;  If one of these conditions is true and the program is not already changing
;  the direction, the Spindle_Dir_Change bit will flip the direction.
;
;  The "OR / Already_run" is used to simulate a direction change at
;  initial power up to get into a default CW state.
;
Spindle_Dir_Change   = ( ( ( ( Real_Spin_Dir_Key XOR Last_Spin_Dir_Key ) AND
                 / Auto_Spin_Mode ) OR
                 ( Auto_Spin_Mode & ( Spindle_Dir_Out XOR Last_Spin_Dir ) ) )
                 AND / Spindle_Dir_Change ) OR / Already_run
Last_Spin_Dir_Key    = Spindle_Dir_Key
Spindle_Dir_Key      = Real_Spin_Dir_Key OR ( Auto_Spin_Mode AND M4 )
;
;  Spindle_Dir is the commanded direction.
;  Spindle_Dir_Out us the actual direction sent to the hardware.
;  These two may be different if, for instance, the
;  spindle is being reversed in low range.
;
```

```
Last_Spin_Dir       = Spindle_Dir_Out
Spin232_Dir         = Spindle_Dir_Out
;
; The Spindle_Dir_LED is set based upon the commanded direction, NOT
; the actual direction output to the hardware. As noted above,
; if Spindle_Dir_LED were set to Spinlde_Dir_Out then the LED would
; be incorrect if the spindle direction was being reversed because
; of being in low range.
;
Spindle_Dir_LED     = Spindle_Dir
;
; The XPLC program handles spindle Auto/Man changes and
; whether the spindle is enabled.
;
Spin232_Enable      = Spindle_Enable
Spindle_Auto_LED    = / Man_Spin_Mode
;
; Here the signal into CNC software that determines the spindle range
; is set. The CNC software uses this bit to determine the proper
; speed to display on the screen.
;
; Is is based upon the physical input and the
; Range_Reverse bit. The Range_Reverse bit is set by the XPLC
; program based upon CNC Machine Parameter 178 settings.  It is used to
; set the physical spindle range input to work as either a NO/NC input.
; More information is found in the XPLC program.
;
High_Low_Range      = Spindle_Range_In XOR Range_Reverse

;/////////////////////// UNICONSOLE-2 CPU7 COOLANT CONTROL  //////////
;
; The XPLC program is controlling the coolant functions.
; These lines just turn on the LEDs since XPLC cannot set them.
;
Flood_LED           = / Flood
Mist_LED            = Mist
Coolant_mode_LED    = Auto_Coolant_Mode
;
; The lines below will turn off the M-function
; requests if a programming is not running.
; For example, if while running a program with
; spindle CW on (M3) and/or Flood (M8),
; The spindle and coolant will be turned off
; when the program ends, or the CANCEL or ESC key is pressed.
;
M3                  = M3        AND CNC_Program_Running
M4                  = M4        AND CNC_Program_Running
M8                  = M8        AND CNC_Program_Running
M7                  = M7        AND CNC_Program_Running
;
; The logic below will signal to CNC software (via Drive_fault_out)
; a drive fault condition if the Servo_fault bit indicates a fault, or
; there already is a fault and the E_stop has not been pressed.
; This manner of logic is referred to as "latching" the fault.
; Once the fault has been "latched", the E_stop must be pressed AND
; the fault bit must indicate a no fault condition before the
; fault will be cleared.
;
; This fault can be disabled by setting of the Fault_override bit.
;
;
Drive_fault_out    =  ( Servo_fault OR Drive_fault_out AND / E_stop ) AND
                       / Fault_override
;
; The XPLC program will set PC_Lube_Fault and PC_Spindle_Fault
; to signal these faults.  Here, the standard program just
; reads the MEM bits and writes the actual OUT bits
; that signal the fault. These faults will be disabled
; if the Fault_Override bit is set.
;
    Lubricant_low       = PC_Lube_Fault & / Fault_Override
```

```
Spindle_fault_out    = PC_Spindle_Fault & / Fault_Override
;
;  And the same method of control noted above is used for controlling
;  the Stop bit, Block_mode, and the AUX function key LEDs.
;  In this way, there are no changes required to the standard
;  program to control these states.
;
Stop                 = PC_Stop & / Fault_Override
Block_mode_LED       = PC_Block_Mode
Feed_Hold_LED        = Pause
Brake_LED            = Brake_mode
Aux_2_LED            = PC_Aux_2_LED
Aux_3_LED            = PC_Aux_3_LED
Aux_4_LED            = PC_Aux_4_LED
Aux_5_LED            = PC_Aux_5_LED
Aux_6_LED            = PC_Aux_6_LED
Aux_7_LED            = PC_Aux_7_LED
Aux_8_LED            = PC_Aux_8_LED
Aux_9_LED            = PC_Aux_9_LED
;
; Aux_11_LED, due to backward compatibility,
; is backwards. (1 = LED off, 0 = LED on)
;
Aux_11_LED           = / PC_Aux_11_LED
Aux_12_LED           = PC_Aux_12_LED
;**************************************************************************
;                               END OF PROGRAM
;**************************************************************************
```

The XPLC program:

```
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; *
; *  File:    BASEXPC1.SRC
; *  Purpose: XPLC program for SERVO3IO/UNICONSOLE-2
; *
; *          Works in conjunction with STANDARD program BASECPU1.SRC
; *
; *          CNC Configuration Settings
; *              Jog Panel Type:   Uniconsole-2
; *              PLC Type:         Normal
; *
; *          CNC Parameter Settings
; *              P31   = 1 or 2 (for COM1 or COM2 spindle control)
; *
; *              P177  = 0.0  Normal behavior
; *                    = 1.0  Fault Override in effect
; *
; *              P178  Bit flags
; *                    LubeNONC         ; P178 Bit 0  (1)
; *                    SpindleNONC      ; P178 Bit 1  (2)
; *                    NoReverseSpindle ; P178 Bit 7  (128)
; *                    SpinRangeNONC    ; P178 Bit 9  (512)
; *
; *              P179  Lube timer settings  (see below for full explanation)
; *                    = 0  On when running a job
; *                    MMMSS   Off for MMM minutes, On for SS seconds
; *                            if SS = 0, On for at least MMM minutes
; *
; *
; *   () Support for probing and digitizing.
; *   () Handling of both NO/NC spindle drive faults and automatic reset.
; *   () Handling of both NO/NC low lube faults.
; *   () Handling of drive faults.
; *   () Support of two limits switches each for all three axes
; *   () Support for NO/NC spindle high/low range that can
; *      optionally reverse direction.
```

```
; *   () Support for drawbar unclamping.
; *   () Control of flood coolant in both automatic (M8) and manual modes.
; *   () Control of mist coolant in both automatuc (M7) and manual modes.
; *   () Automatic/manual control of a spindle brake.
; *   () Lube pump timing and control.
; *   () Fault override logic via parameter setting.
; *
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;
;
```

Note: The listing for the XPLC program omits the definitions for INP1-INP80, OUT1-OUT80, and MEM1-MEM80 since they are identical to the standard program.

```
Spin_start          IS MEM101  ;
Spin_stop           IS MEM102  ;
Autostart           IS MEM104  ;
Autostop            IS MEM105  ;
Limit_tripped       IS MEM106  ;
Fault               IS MEM107  ;
Halt                IS MEM108  ;
Spindle_pause       IS MEM109  ;
Block_mode_key_hit  IS MEM110  ;
Spin_dir_key_hit    IS MEM112  ;
Rapid_over_key_hit  IS MEM115  ;
Brake_key_hit       IS MEM117  ;
LubeTimeExpired     IS MEM140  ;
ManFlood            IS MEM152  ;
ManMist             IS MEM153  ;
Man_Cool_Type       IS MEM162  ;
Coolant_Type        IS MEM163  ;
LubeMethod1         IS MEM164  ;
LubeMethod2         IS MEM165  ;
LubeMethod3         IS MEM166  ;
;
; In this program, CNC Machine Parameter 178 has been
; pre-defined as a bit mapped parameter that controls
; various aspects of the PLC program behavior.
;
; The LubeNONC, SpindleNONC, and SpinRangeNONC bits can be used to flip
; the respective switch input behavior between NO and NC. For example,
; the lube fault is considered to be a NC input, i.e., if the switch is
; electrically closed, then there is no fault.  When the switch opens,
; then this signals a fault.  If the lube fault signal needs to be NO,
; then the LubeNONC bit can be set.
;
; The NoReverseSpindle bit is used to indicate whether the spindle
; direction outputs should be reversed between high and low range.
; The default behavior of this program is to reverse direction as this
; convention follows a majority of the spindle gearing systems in use.
;
LubeNONC            IS MEM200  ; P178 Bit 0  (1)
SpindleNONC         IS MEM201  ; P178 Bit 1  (2)
uP178Bit2           IS MEM202  ; P178 Bit 2  (4)
uP178Bit3           IS MEM203  ; P178 Bit 3  (8)
uP178Bit4           IS MEM204  ; P178 Bit 4  (16)
uP178Bit5           IS MEM205  ; P178 Bit 5  (32)
uP178Bit6           IS MEM206  ; P178 Bit 6  (64)
NoReverseSpindle    IS MEM207  ; P178 Bit 7  (128)
uP178Bit8           IS MEM208  ; P178 Bit 8  (256)
SpinRangeNONC       IS MEM209  ; P178 Bit 9  (512)
uP177Bit10          IS MEM210  ; P178 Bit 10 (1024)
uP178Bit11          IS MEM211  ; P178 Bit 11 (2048)
uP178Bit12          IS MEM212  ; P178 Bit 12 (4096)
uP178Bit13          IS MEM213  ; P178 Bit 13 (8192)
uP178Bit14          IS MEM214  ; P178 Bit 14 (16384)
uP178Bit15          IS MEM215  ; P178 Bit 15 (32768)
;
;***************************************************************************
;                             STAGE DEFINITIONS
```

```
;*****************************************************************************

InitialStage         IS STG1
MainStage            IS STG2
LoadCNC10Parameters   IS STG3
Lubemonitor          IS STG6

;*****************************************************************************
;                              WORD DEFINITIONS
;*****************************************************************************

LubeWord             IS W12
LubeOnTime           IS W13
LubeOffTime          IS W14
;LubeOffX100         IS W15
P178Bits             IS W16
P177Bits             IS W17
TotalTime            IS W18
PresetTime           IS W19

;*****************************************************************************
;                             TIMER DEFINITIONS
;*****************************************************************************

LubeOffT             IS T2
AccumulatingTMR      IS TMR3
LubeOnT              IS T4

;*****************************************************************************
;                          PD (One-Shot) DEFINITIONS
;*****************************************************************************
CM_1Shot             IS PD1      ; Coolant Auto/Manual key pressed
MK_1Shot             IS PD2      ; Mist coolant key pressed
SD_1Shot             IS PD3      ; Spindle Direction
BM_1Shot             IS PD4      ; Block Mode key pressed
ATM_1Shot            IS PD5      ; Coolant Auto To Manual change
ES_1Shot             IS PD6      ; E_stop release
M10_1Shot            IS PD7      ; M10
M11_1Shot            IS PD8      ; M11
FK_1Shot             IS PD9      ; Flood
BRK_1Shot            IS PD10     ; Aux1
;
;  Lube Timing Related One-Shots
;
PR_1Shot             IS PD11     ; started CNC program running
NPR_1Shot            IS PD12     ; stopped CNC program running
OffTimerExpired      IS PD13     ;
OnTimerExpired       IS PD14     ;

;*****************************************************************************
;*                               PROGRAM START
;*****************************************************************************

;----------------
 InitialStage
;----------------
;
; InitialStage (STG1) is on at PLC initialization.
; Here we set PC_PLC_Running (MEM49) to indicate to
; the standard program that XPLC is being used also.
;
IF !ZERO THEN  SET PC_PLC_Running,
               SET LubeMonitor,
               SET LoadCNC10Parameters,
               JMP MainStage

;---------------------
   LoadCNC10Parameters
;---------------------
;
; Here is where the processing of the pre-defined CNC
; Machine Parameters takes place.  Note that this STG
; remains on so that changes to the CNC Machine
```

```
; Parameters can take effect without rebooting.
;
; Read the value from P179 in MMMSS format
; and break it into separate MMM and SS parts
;
; Remember that mathematical operations work
; with integers.  When dividing, there are no remainders
; and no explicit rounding up.
;
; For example, 5 / 2 = 2, 100 / 99 = 1, 99 / 100 = 0
;
;  LubeWord      = HHHMM
;  HHHMM / 100   =   HHH (LubeOffTime)
;  HHH * 100     = HHH00 (LubeOffTime * 100)
;
;                  HHHMM (LubeWord)
;                - HHH00 (LubeOffTime * 100)
;                -------
;                =    MM (LubeOnTime)
;
IF !ZERO THEN  LP9 LubeWord,
               LubeOffTime = LubeWord / 100,
               LubeOnTime  = LubeWord - LubeOffTime * 100
;
; Convert LubeOffTime to minutes
; and   LubeOnTime  to seconds
; and set timer PRESET values
;
; TMR values are in 0.01 second increments, i.e.,
; there are 100 of them in every second.
;
IF !ZERO THEN  LubeOffTime = LubeOffTime * 100 * 60,
               LubeOnTime  = LubeOnTime * 100,
               LubeOffT    = LubeOffTime,
               LubeOnT     = LubeOnTime
;
;
; Initialize Bit reversals according to param 178
; Memory Bits 200-215 hold the values in parameter 178
;
;  P178Bits            P178Bits / 256
;  FEDCBA9876543210    00000000FEDCBA98
;
IF !ZERO THEN  LP8 P178Bits,
               WTB P178Bits MEM200,
               P178Bits = P178Bits / 256,
               WTB P178Bits MEM208,
               LP7 P177Bits


;
; Set MEM bit to communicate with standard PLC whether
; the spindle range input is NO/NC and whether to Override faults.
;
IF SpinRangeNONC THEN (Range_reverse)
IF P177Bits == 1 THEN (Fault_Override)

;-----------------------------
     LubeMonitor
;-----------------------------
;
; These Lube pumps are set by CNC10 Machine Parameter 179,
; where the value is between 0 - 65535
; and is formatted as MMMSS
; Where MMM is the Off Time in minutes
; and SS is the On Time in seconds.
;
;  If SS == 0 and MMM != 0, then Method 1 is used for control.
;  If SS != 0 and MMM != 0, then Method 2 is used for control.
;  If SS == 0 and MMM == 0, then Method 3 is used for control.
;
; METHOD 1
;
```

```
;  On the start of CNC_program_running,
;  the lube pump turns on.
;
;  The lube pump is turned off when E_stop or
;  a low lube condition exists or when
;  a program has NOT been running for MMM minutes.
;
;  This type of control is intended
;  for lube pumps that have internal timers that either
;  lube immediately when power is applied and then start timing, or
;  for pumps that wait until power has been on for the set time
;  before pumping.  The former type of pump will run out of lube oil quickly
;  on short jobs if lube is only applied while CNC_program_running.
;  The latter type of pump will never lube on short job runs if lube power is
;  is only applied while CNC_program_running. A short job is defined
;  as one which completes in less time than the internal lube timer
;  is set to.
;
;  Example 1.
;   The lube pump has a timer and it is set to lube every 30 minutes.
;   Set Machine Parameter 179 = 3500.  Note that this time (MMM) should
;   be longer than the setting of the lube timer.
;
; METHOD 2
;
;  The lube turns on for SS seconds every MMM minutes.
;
;  Example 2.
;   To set the lube pump power to come on for 5 seconds
;   every 10 minutes, set P179 = 1005.
;
;  Example 3.
;   To set the lube pump power to come on for 30 seconds
;   every 2 hours, set P179 = 12030
;
; METHOD 3
;
;  The lube comes on whenever a program is being run
;  (CNC_program_running is on).  This includes MDI mode.
;
;;
;
;  Set MEM bits to indicate the method used
;
IF LubeOnTime == 0 & LubeOffTime != 0 THEN (LubeMethod1)
IF LubeOnTime != 0 & LubeOffTime != 0 THEN (LubeMethod2)
IF LubeOnTime == 0 & LubeOffTime == 0 THEN (LubeMethod3)

IF  Cnc_program_running THEN (PR_1Shot)
IF !CNC_program_running THEN (NPR_1Shot)
;
; Here the accumulated time that a CNC program has been
; running is programmed.  When the Accumulated time
; has reached the OffTime (MMM), then start the On timer.
; The Accumulated time is reset when the On timer expires.
;
IF NPR_1Shot THEN TotalTime = TotalTime + AccumulatingTMR
IF CNC_program_running & !LubeOnT THEN (AccumulatingTMR)
IF TotalTime + AccumulatingTMR > LubeOffTime THEN (LubeOnT), (LubeTimeExpired)
IF LubeOnT THEN TotalTime = 0
IF  LubeTimeExpired THEN (OffTimerExpired)
IF !LubeTimeExpired THEN (OnTimerExpired)
;
;
; Turn the lube pump ON under one the following conditions:
;
; (1) the CNC program has started running and using Method 1 or Method3
; (2) the OffTime is up and using Method2.
;
IF (PR_1Shot & (LubeMethod1 | LubeMethod3)) |
```

```
   (OffTimerExpired & LubeMethod2) THEN SET Lube
;
;
;  When lube is on, start keeping track of the off time
;  for use in Method 2.
;
IF Lube THEN (LubeOffT)
;
;  Turn the lube pump OFF under one of these conditions:
;
;  (1) Method1 and the Off timer has expired and a
;     CNC program is not running.
;  (2) Method2 and the On timer expired.
;  (3) Method3 and a CNC program is no longer running.
;  (4) A low lube fault has occured.
;  (5) E_stop is pressed.
;
IF (LubeOffT & !CNC_Program_Running & LubeOnTime == 0 & LubeOffTime != 0) |
   (OnTimerExpired & LubeOnTime != 0 & LubeOffTime != 0) |
   (NPR_1Shot & LubeOffTime == 0) |
   (Lube_Fault_In XOR LUBENONC) | E_stop THEN RST Lube

;---------------
  MainStage
;---------------

;
;////////////////// UNICONSOLE-2 XPLC SPINDLE CONTROL  ////////
IF !Man_Spin_Mode | !Already_run THEN (Auto_Spin_Mode)
IF M3 | M4    THEN (AutoStart)
IF !AutoStart THEN (AutoStop)

;-------------------------------------------------
;  Select between auto and manual spindle mode
;-------------------------------------------------
IF Spindle_Mode_Switch THEN (SD_1Shot)
IF (SD_1Shot XOR Man_Spin_Mode) THEN (Man_Spin_Mode)

;-------------------------
;     Start the spindle
;-------------------------
IF  (Man_spin_mode & Spindle_start_key) |
    (Auto_spin_mode & Autostart & !Spindle_pause) THEN (Spin_Start)

IF (Spin_Start & !Probe_not_detected) THEN (Probe_Fault)

;-------------------------
;     Pause the Spindle
;-------------------------

IF  Auto_spin_mode & ((Spindle_pause & !Spindle_start_key &
    CNC_program_running) | (!Spindle_pause & Spindle_stop_key
    & Spindle_Enable)) THEN (Spindle_pause)

;-------------------------
;     Stop the spindle
;-------------------------

IF  Spindle_Stop_key | Stop | Limit_tripped |
   (Auto_spin_mode & AutoStop) THEN (Spin_Stop)

IF (Man_spin_mode & Spindle_Dir_key) THEN (Man_spin_dir)

IF (Auto_spin_mode & M4) |
   (Man_spin_mode & Man_spin_dir) THEN (Spindle_dir)
;
;
; The direction of the physical ouput is reversed if
; the NoReverseSpindle Bit is off and the Spindle_range_in is:
;
;   open  (1) and SpinRangeNONC is RST (0)
;              OR
;       closed (0) and SpinRangeNONO is SET (1)
```

```
;
; NoReverseSpindle and SpinRangeNONC are SET/RST
; depending upon the value of P178 in CNC10 Machine Parameters.
;
; Note that even if the spindle is NOT to be reversed in low range
; that the Spindle_range_in is needed by CNC10 to determine
; the correct speed to display on the screen.
;
; Application Requirements:
;
; (1a)  The spindle must reverse in low range.
; (1b)  The spindle must NOT reverse in low range.
;
; (2a)  There is a NC HI-LO switch connected to Spindle_range_in
; (2b)  There is a NO HI-LO switch connected to Spindle_range_in
;
; (3a) The HI-LO switch is tripped when the gear change lever (or switch)
;      is in the LO position.
; (3b) The HI-LO switch is tripped when the gear change lever (or switch)
;      is in the HI position.
;
; Application P178 Settings:
;
; (1a)
;   NoReverseSpindle 0
; (1b)
;   NoReverseSpindle 1
;
; (2a & 3a) or (2b & 3b)
;   SpinRangeNONC   0
;
; (2a & 3b) or (2b & 3a)
;   SpinRangeNONC   1
;
IF Spindle_dir ^ (!NoReverseSpindle &
     (Spindle_range_in ^ SpinRangeNONC)) THEN (Spindle_Dir_Out)

IF (Spindle_Enable | Spin_start) & !Spin_Stop &
   Probe_not_detected THEN (Spindle_Enable)

IF !Spindle_Enable & !Tool_Release & !ZERO_speed THEN (Drawbar_Sol)

;/////////////  UNICONSOLE-2 XPLC COOLANT CONTROL  ////////////////////////

;-------------------------------------------
;      Toggle Auto Manual Coolant Mode
;-------------------------------------------

IF Coolant_mode_switch THEN (CM_1Shot)
IF Auto_Coolant_Mode ^ CM_1Shot OR ! Already_run THEN (Auto_Coolant_Mode)

;
; Flood coolant control
;  Toggle flood coolant on and off if coolant mode is manual
;  When switching from Auto to manual mode, turn off flood coolant
;
IF !Auto_Coolant_Mode THEN (ATM_1Shot)
IF Flood_key THEN (FK_1Shot)
IF (ManFlood ^ (!Auto_Coolant_Mode & FK_1Shot)) & !ATM_1Shot THEN (ManFlood)
IF !Stop & ((M8 & Auto_Coolant_Mode) | (ManFlood & !Auto_Coolant_Mode))
   THEN (Flood)
;
; Mist coolant control
;  Toggle mist coolant on and off if coolant mode is manual
;  When switching from Auto to manual mode, turn off mist coolant
;
IF Mist_key THEN (MK_1Shot)
IF (ManMist ^ (!Auto_Coolant_Mode & MK_1Shot)) & !ATM_1Shot THEN (ManMist)
IF !Stop & ((M7 & Auto_Coolant_Mode) | (ManMist & !Auto_Coolant_Mode))
   THEN (Mist)
```

```
;/////////////////////////////////////////
;
;      FAULT HANDLING
;
;/////////////////////////////////////////
;
; If FLT is not zero, there is an internal error
; in the execution of the XPLC program.
;
IF FLT != 0 THEN SET PLC_Fault_Out
;
; If the lube is low set the lube alarm
;
IF (Lube_Fault_In ^ LUBENONC) |
   (PC_Lube_Fault & !E_stop) THEN (PC_Lube_Fault)
;
; Check for tripped limits
;
IF (X_plus | X_minus | Y_plus | Y_minus | Z_plus | Z_minus)
   THEN (Limit_tripped)
;
;
; Set and clear spindle and servo faults
; The fault is latched and cleared when the fault
; signal is removed and E_stop is pressed.
;
; The  &Already_run2 is used to screen out the intialization passes
; which may initially indicate a fault.
;
IF ((!Spindle_ok ^ SpindleNONC) |
   (Spindle_fault_out & !E_stop)) & Already_run2
   THEN (PC_Spindle_Fault)
;
; Check for PLC fault
; The fault is latched and cleared when the fault signal
; is cleared and E-stop is pressed.
;
IF (!PLC_OK | (PLC_fault_out & !E_stop)) & !Fault_Override
   THEN (PLC_Fault_Out)
;
; If there's a fault condition then Stop
; A running program is not stopped because of
; low lube.  However, the error will occur as soon
; as the running job is stopped or completed.
;
IF PLC_fault_out | Spindle_fault_out | Drive_fault_out |
  (!CNC_program_running & Lubricant_low) THEN (Halt)

IF (Halt | Fault) & !E_stop THEN (Fault)

IF  Fault | E_stop | Coolant_Fault THEN (PC_Stop)
;
; Reset inverter (VFD) if there is a fault and the E_stop is pressed.
;
IF (!Spindle_ok XOR SpindleNONC) & E_stop THEN (VFD_reset)

;////////////////////  SINGLE/BLOCK MODE CONTROL   //////////////
;
; Toggle between single/block mode
;
IF Block_mode_key THEN (BM_1Shot)
IF (!PC_Block_mode & BM_1Shot & !CNC_program_running) |
   (PC_Block_mode & !BM_1Shot)
  THEN (PC_Block_mode)

;////////////////////// AUTO SPINDLE BRAKE CONTROL   /////////////
;
; Set auto spindle brake mode
; | !Already_run is used to turn on Auto Brake mode at initialization.
;
IF Brake_key THEN (BRK_1Shot)
IF BRK_1Shot ^ Brake_mode | !Already_run THEN (Brake_Mode)
      IF Brake_mode & !Spindle_Enable THEN (Brake)
```

```
;
;  Keep these at the end of the Main Stage and in this order
;  These are used for power up conditions
;
IF Already_run THEN (Already_run2)
IF Already_run OR ! Already_run THEN (Already_run)
```

Application Examples

All the examples in the standard manual can be converted to an XPLC program by identifying which bits the XPLC has control of and converting these lines of logic to XPLC format. The definition of plc bits should be the same in both the standard and XPLC program.

Remember the standard programs follow this pattern:

<plc_bit> = <boolean_expression>

To change it into an equivalent XPLC program statement, rewrite is as

IF *<boolean_expression>* THEN (*<plc_bit>*)

and convert any '/' (NOT) symbols to '!'.

As an example, the standard program

```
Brake_key_hit       = Brake_key AND / Last_brake_key
Last_brake_key      = Brake_key
Brake_mode          = ( Brake_mode XOR Brake_key_hit ) OR / Already_run
Brake               = Brake_mode AND / SpindleRelay
```

would be converted to the XPLC program:

```
IF Brake_key AND ! Last_brake_key                   THEN (Brake_key_hit)
IF Brake_key                                        THEN (Last_brake_key)
IF ( Brake_mode XOR Brake_key_hit ) OR ! Already_run THEN (Brake_mode)
IF Brake_mode AND ! SpindleRelay                    THEN (Brake)
```

Using an Aux key to toggle an output.

Assume that for the base programs listed above that we wish to modify the programming so that the AUX2 key on the jog panel toggles a light that is connected to OUT6. We desire the AUX2 LED on the jog panel to be on when the light is on. In this case, we can make all the changes solely to the XPLC program as such:

Definitions

```
LIGHT            IS OUT6
AUX2_KEY_PRSSED  IS PD50
```

Program

```
IF AUX_2_KEY THEN (AUX2_KEY_PRESSED)
IF LIGHT ^ AUX2_KEY_PRESSED THEN (LIGHT), (PC_AUX_2_LED)
```

Using an Aux key to momentarily turn on an ouput.

Assume that in the example above, we wish the Light and LED to be on only while the AUX2 key is being pressed. Here is how it could be done:

Definitions

```
LIGHT            IS OUT6
```

Program

```
IF AUX_2_KEY THEN (LIGHT), (PC_AUX_2_LED)
```

Implementing a Hass Indexer

A Haas Indexer can be can interfaced to a CNC system using one input, one output, and a custom M-code. The example will use an M12 custom M-function to perform the indexing operation. When the M12 command is executed in an M&G code program, OUT6 will turn on until the Haas Indexer finished signal (wired to INP6) closes

For this example, there is a need to change the standard PLC program so that the M12 command will be cancelled if for some reason the job stops running.

Definitions

```
Index_finished   IS INP6
Indexer_Out      IS OUT6
M12              IS INP38
```

Program

```
IF M12 THEN (Indexer_Out)
```

Custom M12 (CNC10.M12)

```
M94/6         ; turn on INP38
M101/6        ; wait for INP6 to close
M95/6         ; turn off INP38
```

Standard PLC program changes

```
M12 = M12 AND CNC_Program_running
```

This example will add to the above Haas Indexer example by allowing an AUX key to work the indexer. The AUX key indexing will work provided that a CNC job is not currently running.

In addition to the above XPLC program we need:

```
AUX1_HIT            IS PD1
AUX1_KEY            IS INP49
AUX_Index           IS MEM20


IF AUX1_KEY THEN (AUX1_HIT)
IF AUX1_HIT & !CNC_Program_running THEN SET Aux_Index

IF M12 | Aux_index THEN (Indexer_Out)
IF !Index_finish | Stop THEN RST Aux_index
```

A Flashing Light

The following example can be used to turn an output on and off at a certain frequency. We will assume a light is attached to this output. A memory bit will be used to turn on the flashing light.

```
LIGHT       IS OUT1
DO_FLASH    IS MEM10
FLASH_LIGHT IS STG2

IF DO_FLASH THEN (FLASH_LIGHT)

;---------------------
    FLASH_LIGHT
;---------------------
IF !T1 THEN T1=200, (T1)
IF TMR1 < 100 THEN (LIGHT)
```

In this example, the LIGHT will be on for one second and off for one second.

Programming UP/DOWN counters

While the XPLC language has no direct support for UP or DOWN counters, the same functionality can be achieved using the following programming. The basic technique is to use a one shot PD coil to increment or decrement a word memory location.

```
COUNTER_INPUT     IS INP1
COUNTER_RESET     IS MEM2
COUNT_REACHED     IS MEM3
COUNTER_1SHOT     IS PD4
COUNTER_VALUE     IS W5
COUNTER_LIMIT     IS W6
COUNTER_SETUP     IS STG7
MONITOR_COUNT     IS STG8

;-------------------------------------
         COUNTER_SETUP
;-------------------------------------
IF 1==1 THEN COUNTER_LIMIT = 10, JMP MONITOR_COUNT
;-------------------------------------
         MONITOR_COUNT
```

```
;--------------------------------------
IF COUNTER_INPUT THEN (COUNTER_1SHOT)
IF COUNTER_1SHOT THEN COUNTER_VALUE = COUNTER_VALUE + 1
IF COUNTER_VALUE >= 10 THEN (COUNT_REACHED)
IF COUNTER_RESET THEN COUNTER_VALUE = 0
```

This example omits the need to turn on the COUNTER_SETUP stage during program initialization. It also does not show how other parts of the program would respond to the COUNT_REACHED.

Turning a truth table into a boolean expression.

INP1	INP2	INP3	OUT1
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

```
IF

(!INP1 & !INP2 &  INP3) |
(!INP1 &  INP2 & !INP3) |


( INP1 & !INP2 &  INP3) |
( INP1 &  INP2 & !INP3)

THEN (OUT1)
```

The example above shows a truth table and the boolean expression that would be used in a program to implement it. By close examination, one can see the pattern that emerges. Every row in the truth table where OUT1 is 1 becomes a line of logic that is OR'ed with the other rows where OUT1 is 1. Each row is the ANDing of each INP. If the truth table was a 0 for an INP, then in the boolean expression it would have a ! (NOT) preceding it.

In RLL, the above program would be written as such:

Turning a truth table into a boolean expression as above will often result in an expression that is more complex than it needs to be. There are techniques that can be used to simplify these expressions but they are beyond the scope of this manual. Note that this particular expression can be written as:

```
IF INP2 XOR INP3 THEN (OUT1)
```

That's right. In this example INP1 has no effect on the resulting logic.

Converting a series of inputs to a number.

The conversion of a series of inputs into a useable number in a program is typically found when interfacing to an automatic tool changer.

For this example, suppose there are six inputs that represent a tool number and we wish to keep track of the current tool number. The inputs form a binary number.

```
TOOL_INPUT1   IS INP1
TOOL_INPUT2   IS INP2
TOOL_INPUT3   IS INP3
TOOL_INPUT4   IS INP4
TOOL_INPUT5   IS INP5
TOOL_INPUT6   IS INP6
TOOL_NUMBER   IS W2
TOOL_MONITOR  IS STG3


;----------------------------------------
              TOOL_MONITOR
;----------------------------------------

IF 1==1 THEN TOOL_NUMBER = 0
IF TOOL_INPUT1 THEN TOOL_NUMBER = TOOL_NUMBER + 1
IF TOOL_INPUT2 THEN TOOL_NUMBER = TOOL_NUMBER + 2
IF TOOL_INPUT3 THEN TOOL_NUMBER = TOOL_NUMBER + 4
IF TOOL_INPUT4 THEN TOOL_NUMBER = TOOL_NUMBER + 8
IF TOOL_INPUT5 THEN TOOL_NUMBER = TOOL_NUMBER + 16
IF TOOL_INPUT6 THEN TOOL_NUMBER = TOOL_NUMBER + 32
```

If the inputs were representing a BCD tool number then the last two lines in the TOOL_MONITOR stage would be

```
IF TOOL_INPUT5 THEN TOOL_NUMBER = TOOL_NUMBER + 10
IF TOOL_INPUT6 THEN TOOL_NUMBER = TOOL_NUMBER + 20
```

Another consideration to take into account when using this technique is that PLC inputs are not received into the copy buffer in parallel from the hardware. In other words, they are received one at a time or one after another. If the snapshot of the inputs is taken in the middle of this updating, the calculated tool number will be wrong. For example: when changing from a 7 (0111 binary) to 8 (1000 binary), there are four inputs that have changed. If a pass of program execution happens before all four bits have been updated then the number will be calculated incorrectly. Whether this will actually be a problem depends upon other factors of the interface.

One method that can be used as a work-around to the above problem is to not allow the number to be updated unless it is within one of the last number calculated. For example, if the number is 3, do not change the number until it reaches 4, assuming the numbers were increasing.

Shortest Path Calculation for Circular Indexing

Automatic tool changers typically consist of a certain number of tools arranged in a circular carousel as shown in the figure below. Often, the carousel can be indexed forward and reverse. The following example shows some programming that can be used for calculating the shortest distance and the direction to index.

This example will assume a motor connected to reversing contactors. The motor control will be through a SPST relay that acts as a motor on/off switch that is wired to a SPDT relay that is wired to the FWD/REV contacts of the reversing contactors. The example below assumes PLCIO2 hardware is being used. It is also assumed that there is a custom M6 program to send the tool number (with M107) and turn on INP38 to start the process.

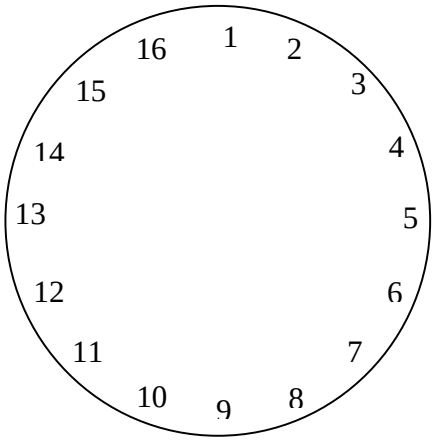
```
M6               IS INP38 ; (ToolChange) Map to M94/6  M95/6


MotorOnRly       IS OUT31 ;
MotorDirRly      IS OUT32 ; off = FWD, on = REV

M6_1SHOT         IS PD1   ;

ToolNumber       IS W1    ; the current carousel position
WantedTool       IS W2    ; the carousel position to index to
NumbeOFTools     IS W3    ; the number of carousel positions
MotorDir         IS W4    ; the motor direction (1 = forward, -1 = backward)
Distance         IS W5    ; the number of indexes required

InitialStage     IS STG1  ;
CalcDistAndDir   IS STG2  ;
Calc_B1          IS STG3  ;
Calc_B2          IS STG4  ;
IndexMotor       IS STG5  ;
IndexFinished    IS STG6  ;
```




```
;-----------------------------
     InitialStage
;-----------------------------
IF 1==1 THEN NumberOfTools = 16,
             ToolNumber = 1,
             JMP MainStage

;-----------------------------
     MainStage
;-----------------------------
IF M6 THEN (M6_1SHOT)
IF M6_1SHOT THEN LDT WantedTool
                 SET CalcDistAndDir

;-----------------------------
     CalcDistAndDir
;-----------------------------
IF WantedTool == ToolNumber
  THEN JMP IndexFinished

IF WantedTool > ToolNumber
  THEN Distance = WantedTool - ToolNumber,
  JMP Calc_B1

IF ToolNumber > WantedTool
  THEN Distance = ToolNumber - WantedTool,
  JMP Calc_B2


;-----------------------------
    Calc_B1
       ;-----------------------------
```

```
IF Distance <= NumberOfTools/2
  THEN MotorDir = 1

IF Distance > NumberOfTools/2
  THEN MotorDir = -1,
       Distance = NumberOfTools - Distance

IF 1==1 THEN JMP IndexMotor

;-----------------------------
    Calc_B2
;-----------------------------
IF Distance <= NumberOfTools/2
  THEN MotorDir = -1

IF Distance > NumberOfTools/2
  THEN MotorDir = 1,
  Distance = NumberOfTools - Distance

IF 1==1 THEN JMP IndexMotor

;-----------------------------
    IndexMotor
;-----------------------------
;
; It is assumed here that another stage is monitoring and updating ToolNumber
;
IF 1==1 THEN (MotorOnRly)
IF MotorDir < 0 THEN (MotorDirRly)
IF ToolNumber == WantedTool THEN JMP IndexFinished

;-----------------------------
    IndexFinished
;-----------------------------
;
;  At this point, the tool carousel has been indexed to the new location.
;  There should be some handshaking with the custom M6 to signal the end of the process
;
IF 1==1 THEN RST IndexFinished
```

Accumulating Timers

Like counters, XPLC programs do not directly support accumulating timers. Accumulating timers are timers whose current value does not reset to zero if the timer input is false. Accumulating timers have a separate input that is used to reset the current value.

```
ATIMER_INPUT    IS INP1
ATIMER_RESET    IS INP2
ATIMER_UP       IS MEM20
INPUT_OFF       IS PD1
ACCUMTMR        IS TMR1
ATIMER_PRESET   IS W1
TOTAL_TIME      IS W2
SUBTOTAL        IS W3
ATIMER_SETUP    IS STG1
ATIMER_MONITOR  IS STG2

;-------------------------------------
             ATIMER_SETUP
;-------------------------------------
IF 1==1 THEN ATIMER_PRESET = 100 * 60 * 15, JMP ATIMER_MONITOR



;-------------------------------------
             ATIMER_MONITOR
;-------------------------------------
IF !ATIMER_INPUT THEN (INPUT_OFF)
IF INPUT_OFF THEN SUBTOTAL = SUBTOTAL + ACCUMTMR
IF ATIMER_INPUT & !ATIMER_RESET  THEN (ACCUMTMR)
```

```
IF ATIMER_RESET  THEN SUBTOTAL = 0
IF 1==1 THEN TOTAL_TIME = SUBTOTAL + ACCUMTMR
IF TOTAL_TIME >= ATIMER_PRESET THEN (ATIMER_UP)
```

The example above programs a 15 minute accumulating timer. Whenever ATIMER_INPUT is on, the timer will start timing. When ATIMER_INPUT is off, the timing stops timing but the TOTAL_TIME contains the accumulated time. If the TIMER_RESET is on, the TOTAL_TIME is reset. When the accumulating timer reaches the preset value (programmed above as 15 minutes), the ATIMER_UP bit will be on.

Communication via CNC Machine Parameters

CNC Machine Parameters 170-179 are passed to the PLC program when they are changed using G10 codes or when the parameters are saved. The value of these parameters can be read into XPLC word memory using the LP0 – LP9 commands.

```
IF 1==1    THEN LP0 W1
IF W1 == 1 THEN (OUT1)
```

In the above program, if CNC Machine Parameter 170 was set to 1.0, then OUT1 would turn on.

In a G-code program, P170 could be turned on/off as such:

```
G10 P170 R1   ; turn on OUT1
G10 P170 R0   ; turn off OUT1
```

Note that when a G-code program is being run that the program is actually being parsed ahead of what is currently executing. Thus, if a line such as G10 P170 R1 were located near the end of the program, it could happen that the parameter change could take effect almost immediately after starting a job. The way to halt processing until a certain point is reached is to proceed it with a read of a PLC bit variable as such:

```
if #6001      ;  do not parse the program any further than this line.
G10 P170 R1   ;  turn on OUT1
```

In this way, the parameter change will not occur until the G-code execution reaches this point. #6001 is a reference to INP1. It is possible to use other values as well.

A CNC Machine Parameter can also be used to bitmap values. Since the range of valid values in the Machine Parameters is 0-65535, a single parameter can be used to turn on and off up to sixteen different bits. Refer to the program BASEXPC1.SRC which demonstrates this technique and allows a CNC Machine Parameter to configure certain inputs to work with either normally open (NO) or normally closed (NC) switch inputs without having to rewrite the program.

Using more than 16 custom M-codes

Custom M-codes usually involve using a combination of M94/M95 commands to turn on/off INP33-INP48. The PLC program will then look at INP33-INP48 to turn on/off an output. There is a problem, however, when more than 16 custom M-codes are required. Remember also that at least five are predefined for spindle, coolant, and clamp control. Presented below is a technique used to get more than 16. The basic idea is to use so many lines to form a binary M-code number and another to act as a strobe. Then the individual M-codes setup the binary number and then turn on a strobe. The following example outlines everything needed for 32 custom M-codes, using just six of the INP33-INP48 bits.

```
MFUN_STROBE      IS INP38  ; Set with M94/6
MFUN_BIT0        IS INP39  ; Set with M94/7
MFUN_BIT1        IS INP40  ; Set with M94/8
MFUN_BIT2        IS INP41  ; Set with M94/9
MFUN_BIT3        IS INP42  ; Set with M94/10
MFUN_BIT4        IS INP43  ; Set with M94/11


MFUN_1SHOT       IS PD1

MFUN_VALUE       IS W1

DO_M_FUCNTION    IS STG2
DO_MFUN_0        IS STG100
DO_MFUN_1        IS STG101
…
…
…
DO_MFUN_31       IS STG131


;
;  Locate this code somewhere in the main loop
;

IF MFUN_STROBE THEN (MFUN_1SHOT)
IF MFUN_1SHOT THEN SET DO_M_FUCNTION

;----------------------
     DO_M_FUNCTION
;----------------------

IF 1==1 THEN MFUN_VALUE = 0
IF MFUN_BIT0 THEN MFUN_VALUE = MFUN_VALUE + 1
IF MFUN_BIT1 THEN MFUN_VALUE = MFUN_VALUE + 2
IF MFUN_BIT2 THEN MFUN_VALUE = MFUN_VALUE + 4
IF MFUN_BIT3 THEN MFUN_VALUE = MFUN_VALUE + 8
IF MFUN_BIT4 THEN MFUN_VALUE = MFUN_VALUE + 16

;
;  At this point MFUN_VALUE will be 0-31
;
IF MFUN_VALUE == 0  THEN JMP DO_MFUN_0
IF MFUN_VALUE == 1  THEN JMP DO_MFUN_1
IF MFUN_VALUE == 2  THEN JMP DO_MFUN_2

…
…
…

IF MFUN_VALUE == 30 THEN JMP DO_MFUN_30
IF MFUN_VALUE == 31 THEN JMP DO_MFUN_31

;-------------------
     DO_MFUN_0
;-------------------
```

```
;
; Place actions in here
;

;-------------------
     DO_MFUN_1
;-------------------
;
; Place actions in here
;

…
…
…

;-------------------
     DO_MFUN_31
;-------------------
;
; Place actions in here
;
```

The preceding program would be used with the following CNC10.M?? codes.

CNC10.M00

```
M95/7    ;\
M95/8    ;-\
M95/9    ;-->  Set up binary pattern 00000
M95/10   ;-/
M95/11   ;/

M94/6    ;
G4 P0.2  ;     Trigger MFUN_1SHOT
M95/6    ;
```

CNC10.M01

```
M94/7    ;\
M95/8    ;-\
M95/9    ;-->  Set up binary pattern 00001
M95/10   ;-/
M95/11   ;/

M94/6    ;
G4 P0.2  ;     Trigger MFUN_1SHOT
M95/6    ;

…
…
…
```

CNC10.M31

```
M94/7    ;\
M94/8    ;-\
M94/9    ;-->  Set up binary pattern 11111
M94/10   ;-/
M94/11   ;/

M94/6    ;
G4 P0.2  ;     Trigger MFUN_1SHOT
M95/6    ;
```

Note that the actual M-codes used would typically be ones that do not already have a pre-defined

meaning. Some M functions that do not have pre-defined meanings in milling software are M12-M24, M27-M29, M31-M38, and M40-M90.

Another Way to Control Outputs in M&G code Programs

Another technique that can be used to control PLC outputs from within an M&G code program is to combine the use of a bitmapped CNC Machine Parameter with appropriate subprograms.

Assume that CNC Machine Parameter 170 is used to hold the states of 16 individual bits. The XPLC program fragment below maps P170 into MEM100-MEM115 and then MEM100-MEM115 are used to control OUT1-OUT15, and OUT29. OUT16 was skipped because it is a PLC fault indicator and OUT17-OUT28 are reserved for 12-bit spindle speed.

```
IF 1==1 THEN LP0 W1,         ; Load P170 into W1
             WTB W1 MEM100,  ; Write the lower 8 bits to MEM100-MEM107
             W1 = W1 / 256,  ; Shift the upper 8 bits to the lower 8 bits
             WTB W1 MEM108   ; Write the 8 bits to MEM108-MEM115

IF MEM100 THEN (OUT1)
IF MEM101 THEN (OUT2)
IF MEM102 THEN (OUT3)
…
…
…
IF MEM114 THEN (OUT15)
IF MEM115 THEN (OUT29)
```

And in the NCFILES directory we have the following programs:

SETBIT

```
if [#4201 || #4202] goto 1      ; skip if doing search or backplot
if #6001                        ; wait until program execution reaches this point
G10 P170 R[#9170 or (2 ^ #B)]   ; set the bit in P170
N1 ; end
```

CLRBIT

```
if [#4201 || #4202] goto 1                    ; skip if doing search or backplot
if #6001                                      ; wait until execution reaches this point
G10 P170 R[(#9170 and ~(2 ^ #B)) and 65535]   ; clear the bit in P170
N1 ; end
```

Then the individual outputs can be turned on in M&G codes like this

```
G65 “SETBIT” B1  ; turn on  OUT2
```

and turned off with

```
G65 “CLRBIT” B1  ; turn off OUT2
```

Here, B is a value from 0-15.

VOID Program Templates

There are times when it is nice to experiment with PLC programming without having to worry about fault messages appearing on the screen. The standard PLC program below, VOID.SRC, is a program that can be used to eliminate all faults.

```
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
; * File:    VOID.SRC
; * Purpose: blank program for testing
; *
; * Notes:   Forces all faults and stop bit off.
;            PLC_op_signal is forced to zero.
;
; * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

PLC_fault_out       IS OUT16   ;
Lubricant_low       IS OUT63   ;
Drive_fault_out     IS OUT64   ;
Spindle_fault_out   IS OUT65   ;
Stop                IS OUT75   ;
PLC_op_signal       IS OUT76   ;
PCPLC_running       IS MEM49   ;
Zero                IS MEM73   ;

;*******************************
;*       Program Start         *
;*******************************
;
;  Ensure the Zero MEM bit is forced to zero
;  In case it had previously been set and the
;  system was not powered off.
;
Zero                = Zero & / Zero
;
; Disable PLC, Lube, Spindle, and Axis Drive faults
;
PLC_fault_out       = Zero
Lubricant_low       = Zero
Drive_fault_out     = Zero
Spindle_fault_out   = Zero
;
; Prevent the stop bit from being set.  Normally,
; the Stop bit would be set by some other fault.
; If a different plc program was running, it may have
; set the fault.  We clear the Stop bit here so that this
; program can be compiled and installed without rebooting
; being needed to clear the fault.
;
Stop                = Zero
;
; Clear the PLC_op_signal so at startup CNC software does not “hangup”
; with a “Waiting for PLC operation” in the message window
; and then waiting for an <ESC> or <CANCEL>.
; If PLC_op_signal is 1, then trying to enter MDI
; mode in CNC software will prevent the Block? prompt
; from appearing and require the <ESC> or <CANCEL>
; to be pressed a couple of times to return to normal display.
;
PLC_op_signal       = Zero

;*******************************
;  End Program
;*******************************
```

Troubleshooting and Debugging

It is common during the development of programs to make errors. Errors can be viewed as either syntax errors or logic errors. Syntax errors are those errors that prevent a program from being compiled and are generally errors that are fixed easily. Logic errors are those errors that occur when a program behaves differently than expected.

Handling Syntax Errors

Syntax errors are best handled by fixing them in the order that they are output when the program is compiled. While the compiler will output the line number in the program where it encountered an error, sometimes the error is actually caused by something that preceded the displayed line number. The point here is do not assume that there is always something wrong with the line the compiler displays.

Handling Logic Errors

Logic errors are typically harder to find. It is possible that these types of errors are not discovered for months or even years.

There is a PLC bit watch display in the CNC software that can be toggled on and off using <ALT-I>. This will display the states of the bit numbers for INP1-INP80, OUT1-OUT80, MEM1-MEM80, and, if there is an XPLC program running, STG1-STG80. These bits can be watched to help determine the problem.

For bits that are outside the 1-80 range in the PLC debug display, it is usually best to set one of the lower MEM bits equal to that bit. For example, to monitor MEM200, write in the program:

```
IF MEM200 THEN (MEM35)
```

so that MEM200 can be monitored by viewing MEM35 in the watch display. The same techniques can also be used for monitoring timers, one-shots, or a complete boolean expression.

Common logic errors

There are several common mistakes made that cause logic errors. One of these mistakes is to reference a bit in an output coil action twice in a program. For example,

```
LUBE_FAULT   IS INP1
LEVEL_LOW    IS INP2

RED_LIGHT    IS OUT1

IF LUBE_FAULT THEN (RED_LIGHT)
…
…
…
IF LEVEL_LOW THEN (RED_LIGHT)
```

In the above example program, it is desired to turn on a RED_LIGHT whenever there is a LUBE_FAULT or if the LEVEL_LOW input was on. What happens when the program is executed is that the RED_LIGHT is on only when LEVEL_LOW is on. The RED_LIGHT never comes on when the LUBE_FAULT occurs. The reason is because the last statement that referenced RED_LIGHT will overwrite any previous changes.

The way to find this error is to search through the program for all occurrences of the particular bit. In the example above, the search would be for all occurrences of RED_LIGHT and OUT1. The reason to search for OUT1 is that it is possible it was referenced in the program without using the defined name, RED_LIGHT. Note there is no rule that plc bits must be referenced by their previously defined names.

There is a couple of ways to solve the problem with the most obvious being by replacing the two lines that reference RED_LIGHT with the following:

```
IF LUBE_FAULT | LEVEL_LOW THEN (RED_LIGHT)
```

Now, the program will work as expected.

Another common mistake is forgetting that a program is continuously being executed. Consider a short example where we desire that when the AUX1 key is pressed, then an internal counter will be incremented so as to keep track of the number of times the AUX1 key was pressed.

```
AUX1_KEY    IS INP49
COUNTER     IS W1

IF AUX1_KEY THEN COUNTER = COUNTER + 1
```

This program fragment would appear to work correctly but it does not. What happens is that COUNTER is incremented 256 times for every second the AUX1_KEY is held down. The solution to this problem is to use a one-shot to trigger the incrementing of the counter, as such:

```
AUX1_KEY    IS INP49
AUX1_1SHOT  IS PD1
COUNTER     IS W1

IF AUX1_KEY   THEN (AUX1_1SHOT)
IF AUX1_1SHOT THEN COUNTER = COUNTER + 1
```

Compilation errors

Many different errors can be displayed when trying to compile an XPLC program. The potential errors are listed below along with some brief source code examples that can cause them

General errors

"Memory error."
This error is generated if there is not enough memory available to compile the program. It is not likely that this error will be displayed, but it can be purposely caused by generating a program that has many label statements in it, to exceed the limit that would be reached by defining every PLC token once.

"Stack overflow!"
Like the memory error, it is unlikely that this error will ever be displayed. However, it can be caused by very excessive nesting within expressions.

"Error opening file *filename*"
This error is generated when XPLCCOMP cannot open a file named *filename*. It is most likely caused by specifying an input file that does not exist, but can also be caused by trying to open the .PLC file if it already exists and is a read-only file.

"Too many errors"
This message is displayed after 19 errors have been generated and displayed on the screen. This error causes compilation to be stopped.

"Malformed command line"
XPLCCOMP source_file[.ext] [output_file[.ext]]
This error is generated when there are too many or too few arguments supplied when calling the program.

Syntax errors

When errors related to the compilation process are encountered, they are displayed on the screen. The error logic used in the compiler reports only one error per line and does make a limited attempt to recover from errors, usually by discarding tokens until the next IF, THEN, or STG token is encountered. Errors are displayed in the following format:

`Error Line (`*`line_number`*`): `*`message`*` #`*`token_string`*`#`

`line_number` is the line number of the XPLC source program in which the error occurred

`token_string` is the sequence of characters that the compiler was looking at when the error occurred.

`message` is one of the messages described below.

"IF expected"

This error is generated when the compiler expects to see the IF token, but it does not. A program such as:

```
    STG1
    STG2
Error Line (2): IF expected #STG2#
```

"End of file expected"

Despite the name of this message, this error is usually generated when the compiler does not find the start of a valid rung (an IF or STG token) and then assumes that the next token is the end of the file.

"*symbol* already defined."

This error is generated by defining a symbol more than once, such as in the program:

```
    X_LIMIT  IS INP1
    X_LIMIT  IS INP2
Error Line (2): X_LIMIT already defined. #INP1#
```

"Invalid label statement"

This error is typically seen when a label statement does not define a valid PLC bit token, such as:

```
    X_LIMIT  IS  WHATEVER
Error Line (1): Invalid label statement #WHATEVER#
```

"Undefined label *identifier*"

This error happens when, during the compilation of the program, that an identifier has been found that has not been previously defined.

```
    IF LUBE_LOW THEN (OUT1)
Error Line (1): Undefined label LUBE_LOW #LUBE_LOW#
```

"STG expected"

There are two cases in which this error will be displayed, both of which are demonstrated in the program below:

```
    StageOne IS INP1
    StageOne
    IF INP1 THEN (OUT1)
    IF INP2 THEN JMP OUT2
Error Line (3): STG expected #IF#
Error Line (4): STG expected #OUT2#
```

Note that in the first case, the error is actually on line 2 but not recognized until line 3.

"THEN expected"

This error is generated when the THEN token is omitted or when there is an error trying to parse a valid <*boolean_expression*>. The program below demonstrates both cases.

```
    IF INP2 == INP2 THEN (OUT1)
    IF INP1 & INP2 JMP STG
      Error Line (1): THEN expected #==#
```

```
Error Line (2): THEN expected #JMP#
```

"= expected"

```
    IF INP1 THEN W1
Error Line (1): = expected ##
```

"W expected"

```
    IF INP1 THEN BCD OUT1
Error Line (1): W expected #OUT1#
```

") expected"

```
    IF INP1 THEN (OUT1
Error Line (1): ) expected ##
    IF (INP1 THEN (OUT1)
Error Line (1): ) expected #THEN#
```

"Expected OUT token"

```
    IF INP1 THEN WTB W1 INP50
Error Line (1): Expected OUT or MEM token #INP50#
```

"Invalid action statement"

```
    IF INP1 THEN OUT1
Error Line (1): Invalid action statement #OUT1#
```

"Invalid numerical expression"

```
    IF INP1 THEN W1 = W1 +
Error Line (1): Invalid numerical expression ##
```

"Relational operator expected"

```
    IF W1 AND INP1 THEN (OUT1)
Error Line (1): Relational operator expected #AND#
```

"One of INPn OUTn MEMn STGn expected"

This error occurs if one of the expected tokens does not appear after the SET or RST command is parsed.

```
    IF INP1 THEN SET TMR1
Error Line (1): One of INPn OUTn MEMn STGn expected #TMR1#
```

"One of INPn OUTn MEMn STGn PDn Tn TMRn expected"

This error occurs after an initial `'('` is parsed as part of an action to denote an output coil instruction but none of the expected tokens are found.

```
    IF INP1 THEN (W1)
Error Line (1): One of INPn OUTn MEMn STGn PDn Tn TMRn expected #W1#
```

"Line too long"

This error occurs when a line exceeds 1024 characters in length.

"Integer const too large"

```
    IF INP1 THEN W1 = 2147483647
```

```
    IF INP2 THEN W2 = 2147483648
Error Line (2): Integer const too large #2147483648#
```

"Integer constant overflow"

This error indicates that not only is an integer constant too large but also that it overflows what can be stored in a 32-bit unsigned integer.

```
    IF INP1 THEN W1 = 4294967295
    IF INP2 THEN W2 = 4294967296
Error Line (1): Integer const too large #4294967295#
Error Line (2): Integer constant overflow #4294967296#
```

"Token out of range"

There are only 256 of each type of PLC token and they are numbered 1-256. Any value not inside this range causes this error.

```
    IF INP0   THEN (OUT1)
    IF INP256 THEN (OUT1)
    IF INP257 THEN (OUT1)
Error Line (1): Token out of range #INP0#
Error Line (3): Token out of range #INP257#
```

"Invalid identifier"

This error is generated when an otherwise valid identifier ends in a character that cannot be part of a valid identifier.

```
    X_LIMIT@ IS INP1
Error Line (1): Invalid identifier #X_LIMIT@#
```

"Invalid character"

Whenever a character is found that is not a part of the XPLCOMP language, this error message is generated.

```
    @HOME IS OUT2
Error Line (1): Invalid character #@HOME#
```

Logic Analyzer

Introduction

There is a basic logic analyzer built into the linux versions of software since v1.15. The logic analyzer can monitor up to 32 plc bits from a combination of INP1-INP80, OUT1-OUT80, MEM1-MEM80, and STG1-STG80. On every pass of plc program execution, the logic analyzer checks to see if any of the monitored bits have changed state. If any of the monitored bits have changed state, the logic analyzer records a time stamp and the state of all the bits. Internally, the logic analyzer has a circular buffer capable of storing the last 64 state change records. The CNC software regularly communicates with the plc to obtain the data and log new records into a text file named xplclao.out.

Setup

To enable the logic analyzer, create a text file named xplc.la in the cnc10(t) directory. The format of the xplc.la file is a list of bits to watch, listed one per line starting in the first column. The format for the plc bit is a case insensitive character ('I', 'O', 'M', or 'S') followed by a number from 1-80. The order in which the bits are specified indicates the order in which they will be located in the record and will appear in the output file. See the example below for further details.

Example.

The small program below will be used for demonstration purposes. The behavior of the plc program is to monitor a fault signal. When the fault signal has been closed for at least one second, an error is flagged using a memory bit and a warning light will flash (two seconds on and one second off). The error is cleared after the external fault is opened.

```
External_Fault          IS INP1
Warning_Light           IS OUT1
Error_Flag              IS MEM1
Monitor_Fault           IS STG1
Flash_Light             IS STG2

Monitor_Fault
  if 1==1 then SET mem49
  if !External_Fault then t1 = 100, (t1)
  if t1 then SET Error_Flag, JMP Flash_Light
Flash_Light
  if !t2 then t2 = 200, (t2)
  if tmr2 < 100 then (Warning_Light)
  if External_Fault then RST Error_Flag, JMP Monitor_Fault
```

We want to monitor all the plc program logic we have written rm /cncroot/c/s The first step is to create an xplc.la file with the following contents and located in the cnc10 directory (/cncroot/c/cnc10):

I1
O1
M1
S1
S2

We start the CNC software. Initially, the external fault (INP1) is open. We then toggle INP1 briefly two times

Interpreting Output

The output file, xplclao.out, consists of header information, followed by the state change records. The state change records are listed one per line, and include a 32-bit hex string time stamp and a 32-bit binary string representing the states of the monitored plc bits. The following is a sample output xplclao.out file generated by an xplc.la file as indicated in the example above. The plc program used for this test was

```
PLC Logic Analyzer Output
-------------------------
Number of watch bits = 5
Bit0      I0    EXTERNAL_FAULT
Bit1      O0    WARNING_LIGHT
Bit2      M0    ERROR_FLAG
Bit3      S0    MONITOR_FAULT
Bit4      S1    FLASH_LIGHT
Time Stamp       State
00102E37        00000000000000000000000000001000
00102E9D        00000000000000000000000000001001
00102EF1        00000000000000000000000000001000
00102F6E        00000000000000000000000000001001
00102FBA        00000000000000000000000000001000
001030BA        00000000000000000000000000010110
001031B9        00000000000000000000000000010100
001032BA        00000000000000000000000000010110
001033BA        00000000000000000000000000010100
001034BB        00000000000000000000000000010110
001035BB        00000000000000000000000000010100
001036BC        00000000000000000000000000010110
001037BC        00000000000000000000000000010100
001038BA        00000000000000000000000000010101
001038BB        00000000000000000000000000001001
```

PLC Frequently Asked Questions

These are some common questions regarding the interaction and use of the two PLC programs that are being used in CNC10 software versions 7.50 and above.

Q. Does this FAQ's have anything to do with KOYO PLCs?

A. No. It has nothing to do with the PLC Direct by Koyo or any software having to do with it.

Q. Do I need to set the PLC type to dual in the configuration screen?

A. No. That setting is for third party PLCs only. Generally this should be set to normal.

Q. What are the two PLC programs and where are they located?

A. One is CNC10.PLC which is located in the CNC10 directory or CNC10t directory for lathes, the other is PC.PLC and is located in the PLC directory. The CNC10.PLC file is the same file that has been used on our controls for all previous versions of CNC10 software. The PC.PLC file is a new PLC file with expanded capabilities.

Q. What's the difference between the two?

A. CNC10.PLC – This PLC file runs on the CPU7 / CPU9 motion control card. It is limited in size to 765 tokens (approximately 2,800 bytes). This file can control inputs 1-80, outputs 1-80, and memory bits 1-80. It is the only PLC file that is needed for the control to function.

PC.PLC – This PLC file runs on the PC. Its file size can be much larger than the CNC10 PLC file. It has features such as stages, times, counters, and one-shots. It also has access to 255 inputs, outputs, and memory bits although there are only 80 physical inputs and outputs. With versions 8.10 and above, it can also have parameters passed to it such as number of tools in a tool changer. To use this PLC program memory bit 49 needs to be set to a 1. This tells the motion control card that the PC.PLC program will be controlling outputs 1-48, 81-255 and memory bits 1-255. Note that even though the PLC program has access to all outputs and inputs it can only write to outputs 1-48 and 81-255.

Q. How do I compile the PLC files?

A. The PLC programs use different formats and therefore need different compilers in order to work. The CNC10 PLC program uses PLCCOMP and the PC PLC program uses XPLCCOMP. Typical usage is as follows:

PLCCOMP [source file] [destination file] destination file = c:\CNC10\CNC10.plc

XPLCCOMP [source file] [destination file]destination file = c:\plc\pc.plc

Q. Why is there a need for two PLC programs?

A. The answer is the features in the new PLC program are useful in tool changer and custom applications, but the new PLC program doesn't have access to all of the I/O so the CNC10 PLC program needs to be there to control those I/O. A minimal PLC program for the CNC10 can be made by echoing memory bits to the outputs you need to control. Then you can control those memory bits in the PC PLC program thus making it possible to control virtually everything needed in the PC PLC program.

Q. Do I have to use both PLC programs?

A. No. You only need to use the CNC10 PLC program. Just make sure that you don't have a file called PC.PLC in your PLC directory.

Q. Do I need to have a PLCIO2 to use the new PLC program?

A. No. It can be used with any of our current PLCs: RTK2, PLC15/15, PLC3/3, Servo3IO, PLCIO2. It's just that the PLC programs will be limited to the physical I/O available on that PLC.

Q. What would happen if I set MEM49 and didn't use the PC.PLC program?

A. The motion control card would give up control of the outputs and memory locations and since there wouldn't be a PC PLC program running no outputs would ever turn on.

Q. Can I just use the PC side PLC program?

A. No. At this time there are still compatibility issues that make it necessary to have the CNC10 PLC program.

Q. How do I find out if my system is running both PLC programs?

A. To find this out all you need to do is look in the PLC directory. If there is a file called PC.PLC then your system is running both.

Q. How do I find out what source code was used to compile the PLC programs?

A. To do this you need to edit both the CNC10.PLC file in the CNC10 directory and the PC.PLC file in the PLC directory. At the top of each, there is a header that lists the name of the source code that each one was compiled from.

Q. Is there more information available about the PLC programs?

A. Yes. There is documentation on how to use both types of PLC programs within this chapter.

Appendix B

Rotary Table Setup