

CNC Control With **PIC-SERVO SC** Servo Controllers

~~~ Using Step & Direction Mode ~~~

1 Overview

PIC-SERVO SC based servo controller boards can be used for CNC applications in one of two ways: 1) Path control mode with serial communications, or 2) Step & Direction mode with motions controlled by discrete digital step and direction signals. Both of these modes rely on the servo motor control capabilities of the **PIC-SERVO SC**.

Path control mode is simpler and offers more flexibility, but it requires CNC software drivers specifically written for the **PIC-SERVO SC**. Step & Direction mode, however, is more universal, and can be used with any commercial CNC software which generates digital step and direction outputs. This application note will show you how to set up your **PIC-SERVO SC** controllers for use with Step & Direction mode.

This application note will also describe using the PSDRO application which allow you monitor your axis positions even when the motor drivers are disabled – a unique feature of **PIC-SERVO SC** based controllers.

2 What You Need

You will need the following items for a complete CNC control system:

- 1 **PIC-SERVO SC** controller per motor
 - Use the KAE-T0V10-BDV1 for brush-type motors 3 amps cont., 6 amps peak
 - Use the KAE-T0V10-BD3PHV1 for brushless or brush-type motors 6 amps cont., 7 amps peak.
- 1 NMC communications cable per controller board (p/n KAE-CC20-AC)
- 1 RS485 communications adapter
 - Use **Z232-485** adapter if your PC has an RS232 (COM) port
 - Use **USB-COMi** adapter if your PC only has USB ports
- 1 DB9 male - DB9 female straight serial cable (only if using the **Z232-485** adapter)
- 1 Logic power supply: 7.5 - 12vdc, 500 ma (p/n KAE-LPS-AC)
- 1 Motor power supply: 12 - 48vdc, with enough current for all your motors
- 1 Brushless or DC brush-type motor (with quadrature encoder) per axis
(Brushless motors must also have hall-effect or optical commutation sensors.)
- NMCTest program from jrkerr.com/software.html for setting up the controllers

If your PC has an RS232 port, it is also possible to use a “cheater cable” instead of the **Z232-485** Adapter board. Please see Appendix A for details.

3 Initial Connections

*If using an RS232 port and the **Z232-485** adapter:*

Start off by removing jumpers JP3, JP4 and JP5 from all **PIC-SERVO SC** boards. Plug one end of the DB9 cable into your PC's unused COM port and the other into **Z232-485** adapter. Connect

JP1 of the **Z232-485** adapter to JP2 of the first **PIC-SERVO SC** board using a 10 pin NMC communications cable. Connect JP2 of each additional **PIC-SERVO SC** board to JP1 of the previous board in the chain. On the very last **PIC-SERVO SC** board in the chain, re-insert the jumpers JP3, JP4 and JP5.

If using an RS232 port and a “cheater cable”:

Start off by removing jumpers JP3, JP4 and JP5 from all **PIC-SERVO SC** boards. Plug one end of the cheater cable into your PC’s unused COM port and the other JP2 of the first **PIC-SERVO SC** board. Connect JP2 of each additional **PIC-SERVO SC** board to JP1 of the previous board in the chain using a 10 pin NMC communications cable. On the very last **PIC-SERVO SC** board in the chain, re-insert only the jumper JP3.

*If using a USB port and the **USB-COMi** adapter:*

Start off by removing jumpers JP3, JP4 and JP5 from all **PIC-SERVO SC** boards. Plug the **USB-COMi** adapter into your USB port. (Follow the instructions for configuring the **USB-COMi** adapter as either COM5 or COM6.) Connect the 10 pin cable from the **USB-COMi** adapter to JP2 of the first **PIC-SERVO SC** board. Connect JP2 of each additional **PIC-SERVO SC** board to JP1 of the previous board in the chain. On the very last **PIC-SERVO SC** board in the chain, re-insert the jumpers JP3, JP4 and JP5.

Connect the logic power supply to connector JP8 on just one of the **PIC-SERVO SC** boards. The NMC communications cables will distribute the logic power to all of the other **PIC-SERVO SC** boards. Connect the motor power supply, your motor and encoder (and hall sensors if using brushless motors) to the **PIC-SERVO SC** boards. Please refer to the **PIC-SERVO SC** or **PIC-SERVO SC 3PH** board data sheets for details on making these connections.

4 Configure Controllers

Check Encoder. You will now use the NMCTest program to set the operating parameters for the **PIC-SERVO SC** boards. At this point, it is best to have your motors disconnected from your CNC machine. First turn on the logic power supply and then run NMCTest.exe. The list on the left side of the control panel should list all of your connected **PIC-SERVO SC** controllers. Click on each controller, one-by-one, and spin the motor shaft by hand. Make sure that the position display changes when you move the shaft, indicating that the encoder is connected properly. At this point, you should also verify the number of encoder counts registered per revolution of your motor.

Check Motor Connections. You can now try doing some test motions with each motor. Turn on the motor power supply. For each controller, click on the Enable Amplifier checkbox, and then click on the STOP! Button. This will enable the position servo, and the motor should try holding its position. If the motor moves rapidly and then stops, this means that the motor leads are probably reversed. (If you are using the **PIC-SERVO SC 3PH** board with a brushless motor, please refer to the data sheet for that board for tips on connecting your motor windings.) Next, try entering a command position value of something like 2000 (more if your motor has a gearhead),

select the position mode option and click on the GO button. The motor should move to somewhere near that position. (How near will depend on our servo gains.)

Tune Servo Gains. Once you have determined that each motor is connected properly, you can tune the servo gains to get the optimal servo performance for your motors and mechanism. The ideal gains will depend on many factors, including the motor characteristics, your power supply voltage, and your machine's dynamic properties.

To tune the servo gains, your motors should be connected to the machine drive screws, either directly, or through gearing such as a toothed belt drive. (For servo motors, you usually get the best performance with a 2:1 to 5:1 reduction between your motors and drive screws. Please see Appendix B for more information.)

Start tuning the servo parameters by setting the K_i and K_p parameters to zero. Start increasing the value of K_d until the motor starts to hum or vibrate. (A value of 1000 is a good starting point.) At this point, reduce the value of K_d by about 30%.

Next, start increasing the value of K_d until the motor starts to hum or vibrate. You may have to tweak the motor shaft by hand a little to get it to start vibrating. When you get to this point, reduce the value of K_d by about 30%.

Now, try executing a few test motions by setting the command position to different values. Make sure that the motor moves smoothly without vibration.

The next step is to tune the integral term, K_i . You will notice that with $K_i = 0$, the motor never quite gets to the position you command. If you set a non-zero value of K_i , you will see that the motor position will eventually creep up to the actual command position. The larger the value of K_i , the more quickly the position error is reduced. For large values of K_i , however, the motor will actually overshoot the goal position before creeping back to the correct position. You will have to find an acceptable balance between the overshoot and the settling time.

The value IL is used to limit the integration wind-up. It keeps the integration error from growing very large and creating large overshoots. For small values of IL , the integration part of the servo will not be able to overcome the friction in your drive train to zero out the error. You should start with a value of $IL = 0$ and increase it until the position error finally creeps to zero. Then, increase IL by about 30% above that.

Lastly, the parameter EL is the maximum allowable position error limit. If the position error rises above this limit, the servo will automatically be disabled. In practice, you will want this value set much higher than any actual desirable position error, because you do not want one axis to stop in the middle of a cut, even if the error is higher than you would like. However, by lowering this parameter during testing, you can get an idea of what the maximum error might typically be. Try setting the velocity and acceleration values to what you might typically use for a fine finish cut. Try several sample motions with ever decreasing values of EL . At some point,

the Servo On indicator will turn off, indicating you have exceeded the position error limit. By optimizing the other gain values, you can increase the accuracy of your system.

For setting the other available servo parameters, please refer to the **PIC-SERVO SC** chip data sheet.

Configuring for Step & Direction Inputs. By default, the **PIC-SERVO SC** powers up with all gains set to zero, and it ignores step and direction inputs. This next section describes how to set up the controllers so that the power-up with all parameters set, the servo enabled, and ready for step and direction signals.

The first thing to do determine a good value for the step multiplier value, SM. In Step & Direction mode, each step pulse will increment or decrement the motor's command position by some number of steps, SM. In general, you will want to use the smallest possible value of SM to get the highest resolution. However, if your CNC software can only generate a limited number of steps per second, you will want to increase SM to be able to achieve higher speeds. For example, if your CNC systems can only produce a maximum of 20,000 steps per second, but you want your motors to spin at up to 200,000 encoder counts per second, you should set SM = 10.

With the last of the servo parameters set, you can now configure the **PIC-SERVO SC**'s EEPROM to save these parameters and also to wake up in the proper mode. Click on the Configure EEPROM button. In the window that pops up, you should select the following options:

- Restore current address on power-up
- Enable amplifier on power-up
- Enable the servo on power-up
- Enable the step / direction inputs

If you are using the **PIC-SERVO SC 3PH** board (with brush-type or brushless motors), you will also want to select:

- Enable 3-phase commutation

Now click on the Save Data in EEPROM button. This will both save the data and reset the particular controller.

Repeat this process for each **PIC-SERVO SC** controller. Terminate the NMCTest program and then try power-cycling the logic power. When you turn the logic power back on, each motor should be servoing to a fixed position. (Note that the motor power must be turned on before the logic power is turned on.)

Note that if you run the NMCTest program, the parameters that you have saved in EEPROM do not appear to be restored. This is because the **PIC-SERVO SC** controllers revert to their default state whenever they receive a software reset command.

5 Connecting to your CNC Control System

At this point, the **PIC-SERVO SC** boards are configured to look just like a stepper motor driver. For each motor, you can connect the TTL level step and direction signals from your CNC control system to the step and direction inputs of the **PIC-SERVO SC** boards. The following table indicate where these connections should be made.

Signal	PIC-SERVO SC board	PIC-SERVO SC 3PH board
Step (input)	JP9, pin 4	P4, pin 1
Direction (input)	JP9, pin 5	P4, pin 2
Enable (input)	JP9, pin 6	P4, pin 3
Error (output)	JP9, pin 7	P4, pin 4
GND	JP9, pin 8	P4, pin 5

In addition to the step and direction signals, you will notice that there is also an Enable input signal and an Error output signal. The **PIC-SERVO SC** controller will remain in a reset state as long as the Enable input is low. To enable the controller, raise this input to +5v. (This input may also be left unconnected for normal operation.) If your CNC control system has an enable output for enabling/disabling the motor drivers, you connect it to this pin.

The Error output will go HI if the servo is disabled for any reason, such as on a servo position error or if the motor power is turned off. If your CNC control system has a motor fault input, you can connect it to the Error output.

At this point, you will want to play with your CNC control system parameters to set up the velocities and accelerations for each axis.

6 Using the PSDRO Software

Most stepper based CNC control systems require that the motors be powered at all times in order to keep track of the motor's position. Servo based systems, however, use the motor's encoder to keep track of the position, independent of whether the motor is being driven or not.

In many circumstances, it is very convenient to be able to disable the motor drivers, turn the cranks by hand, and still keep track of the motor positions. Unfortunately, many servo drives designed to be used with step and direction signals do not allow you to monitor the motor positions independently. With the **PIC-SERVO SC** controllers, however, you can still use the serial interface to monitor the motor positions while the drivers are either disabled or enabled.

The Windows based PSDRO.EXE program provides a digital read-out of the position for up to three **PIC-SERVO SC** controllers. For each axis, it has a position display, a reset position button, and a servo disable/enable button. You can also set the position for each motor to a specific value to match the position of your CNC control system.

When you first run PSDRO.EXE, it will detect that the configuration file PSDRO.DAT is missing, and prompt you to configure the program. The configuration window allows you to select the COM port used for communications, the desired units (inch or mm), and the number of encoder counts per inch or mm of travel. You can use a negative number of encoder counts per inch or mm to change the sign of the position display.

You can also click on the configuration button at any time to change the configuration parameters on-the-fly.

Each of the displayed motor positions can be cleared to zero to match the readout of your CNC control software. You can also set the positions to specific values. Clicking on the position display for an axis will turn the display yellow, allowing you to edit the value. Type in the new value and then click on the “Set” button to set the value. (Click on “Cancel Edit” if you do not want to change the value.)

The “Enable” / “Disable” buttons allow you to enable or disable the servo drivers, allowing you to operate the drive screws by hand. The program automatically detects if the servo is enabled or not, and labels each button “Enabled” or “Disabled” as appropriate. (If you are not able to enable an axis, it may be that the motor power supply is not turned on.) You should only use these buttons to disable an axis – using your CNC control software to disable an axis through the hardware enable pin will prevent the program from reading the position.

Appendix A – Making a “Cheater Cable”

Normally, it is recommended that you use a special adapter, such as our **Z232-485** board, to convert RS232 signals to full-duplex RS485. However, for temporary use or if you have very short cables, it is possible to connect RS232 signals directly to RS485 with a “cheater cable”.

An RS232 “logic 1” output has a nominal voltage of -12v, and a “logic 0” output has a nominal voltage of +12v. An RS232 input uses a threshold of about <1v for a logic 1 signal and a threshold of about >3v for a logic 0 signal.

RS485 uses differential outputs TX+ and TX-, where the (+) signal is non-inverted and the (-) is inverted.. For a logic 1, TX- = 0v and for a logic 0, TX- = +5v.

The RS485 differential inputs will read a logic 1 if RX+ > RX-, or a logic 0 if RX+ < RX-. The RX inputs can range in value from -12v to +12v, and an unconnected RX input will float to a value of about 2.5v.

With all of this considered, you will see that connecting the RS232 TX to RS485 RX-, and connecting RS232 RX to RS485 TX- will actually result in the correct translation of the signals. To build a cheater cable to work with a standard PC 9-pin com port, you should make the following connections:

RS232 (DB9 female)

TX pin 3
RX pin 2
GND pin 5

RS485 (10 pin IDC connector)

RX- pin 2
TX- pin 4
GND pin 10

Appendix B – Maximizing Motor/Controller Performance

Servo motors operate very differently and their performance is specified very differently from stepper motors. Very typically, you might select a 125 oz-in NEMA 23 stepper motor for use on a small desktop milling machine, with the motor shaft connected directly to a 10 or 20 thread/inch drive screw. The temptation when using a servo motor is to also specify a 125 oz-in torque rating. In fact, the required torque for a servo motor will be much less. First, you must note that there are several hidden aspects to stepper motor torque ratings:

1. Stepper motors are rated in terms of their “holding torque”, or the maximum allowable torque when the motor is stationary. However, once the motor starts moving, the actual maximum torque is only about 75% of the holding torque.
2. When moving at high speeds, stepper motor torque drops off dramatically, depending on the specifics of the driver. At 1000 RPM, the maximum torque may only be 20% of the rated holding torque.
3. Because of friction and other torque variations, the instantaneous torque load may be 50% higher than the average torque load. Therefore, the maximum motor torque *available at the maximum operating speed* should be about 30% higher than the average torque load.

Given these caveats, a 125 oz-in rated stepper motor may only reliably drive a 20 oz-in load. Note, however, that 20 oz-in of torque on a 10 TPI drive screw will provide about 40 lb of thrust, assuming a 50% drive efficiency, which is ample for most small CNC machines.

Servo motor operate very differently:

1. Servo motors have a fairly constant continuous torque rating over a wide range of operating speeds.
2. The typical maximum speed for a servo motor is about 3000 – 6000 RPM. The maximum power is delivered at about ½ the maximum speed, or at 1500 – 3000 RPM.
3. The instantaneous torque delivered by a servo motor can be several times the rated torque. Usually the instantaneous torque is limited by the maximum driver current, which is often about twice the continuous current rating.

For practical reasons, it is usually undesirable to run the drive screws on a CNC machine any faster than about 1000 RPM. Therefore, to achieve the maximum power output from your motors, it is better to connect your motors to the drive screws through a geared or toothed belt drive reduction. A reduction in the range 2:1 to 5:1 is typically used, depending on the particulars of your motors and drive screw pitch.

This reduction also has the advantage of increasing the resolution of the motor's encoder, and in turn, the accuracy of the control.

Lastly, for a given size (and power rating) of servo motor, you can usually select between a number of windings with trade-offs between the voltage and torque. To maximize the performance of your system, you will want to match the motor to the amplifier as closely as possible. In particular, you will want to use a motor with a rated winding voltage close to the maximum allowable driver voltage. This, in turn, maximizes the available motor torque.

The standard **PIC-SERVO SC** controller board has an output of 1/5 HP, which is more than enough power to drive a small CNC machine. The **PIC-SERVO SC 3PH** board has an output closer to 1/3 HP, which is adequate for driving even a Bridgeport sized machine if paired with the proper motor and drive train.