

PIC-SERVO CMC

Coordinated Motion Control I.C.

*This document is an addendum to the standard **PIC-SERVO/PIC-ENC** chipset data sheet. You will need to review that data sheet prior to reading this documentation.*

1.0 OVERVIEW

The **PIC-SERVO CMC** is an enhanced version of the standard **PIC-SERVO** motion control chip, augmented with special features for supporting the coordinated motion of several motors. While circular interpolation or electronic gearing are the most common examples of coordinated motion control, the **PIC-SERVO CMC** will allow *any* arbitrary path to be executed. Furthermore, the motion up to 32 axes can be coordinated.

The **PIC-SERVO CMC** is based on the Microchip PIC16C76 processor, and is pin and software compatible with the standard **PIC-SERVO**. It is sold as a chipset including the same **PIC-ENC** encoder counting chip used with the standard **PIC-SERVO**. Users of the standard **PIC-SERVO** should be able to drop in the **PIC-SERVO CMC** with no change to their hardware or software.

In addition to the support for coordinated motion control, the **PIC-SERVO CMC** includes a number of other enhancements, including relative moves, the ability to reset the position counter relative to the home position, the ability to directly read the servo position error, and improved index pulse latching.

2.0 ARCHITECTURE

The **PIC-SERVO CMC** contains a path point buffer with room for 96 entries. Each entry is a goal position for the motor. When the **PIC-SERVO CMC** enters its special path mode, it will automatically move from one point to the next at a user selectable rate of either 30 or 60 Hz¹. The **PIC-SERVO CMC** moves the motor between goal points at a constant velocity such that it always arrives at the next path point in exactly 1/30th or 1/60th of a second. When sets of path points are downloaded into multiple **PIC-SERVO CMC** controllers, and then the paths started simultaneously, the individual axes will execute their paths with exact² synchronization.

The path point buffer has room for about 3 seconds worth of motion for a 30 Hz path and about 1.5 seconds for a 60 Hz path. Typically, the host computer downloads the first part of a path to the **PIC-SERVO CMCs**' buffers and then starts the path mode. As the buffers become depleted, additional path points are dynamically added while the axes are still in motion, until the path is complete. The timing requirements for the host require that it be

¹ In "fast path" mode, the path rates can be selected as either 60 or 120 Hz. The default "slow path" mode is more than adequate for most applications, and requires less communications overhead.

² The exactness of the synchronization is subject to crystal frequency accuracy and other timing factors discussed later.

able to dynamically download new path points before the path point buffers empties completely. With a path point buffer size of 1.5 or 3 seconds, even a non-real time host, such as a PC running Windows, can easily keep up with the task of re-filling the path point buffers as needed.

The actual multi-axis paths which are downloaded into the **PIC-SERVO CMC**'s path point buffers are calculated by the host computer. (Example software for calculating various types of paths is available from www.jrkerr.com.) In addition to creating the geometry desired (arcs, lines, etc.), the paths should be smooth, adhering to the physical acceleration and velocity limits of the motors being controlled. Because the host computer actually creates the paths, any path the user can create can be executed, and paths can involve up to 32 axes. Most typically, coordinated straight line motions, 2-axis circular motions, or S-curve profiling motions are created.

Note that motions created with the path mode are independent of any acceleration or velocity values loaded using the Load Trajectory command.

Path Mode Commands

The following commands and functions of the **PIC-SERVO CMC** are used to implement coordinated motion control:

Add Path Points This is a new command added to the **PIC-SERVO** command set which allows you to add from 1 to seven path points at a time to the path point buffer. The same command is also used to start execution of the path mode. New path points can be added to the path point buffer while in path mode to create arbitrarily long paths.

Stop Motor In addition to its normal functions for the standard **PIC-SERVO**, any Stop Motor command will cause the **PIC-SERVO CMC** to terminate the path mode and clear any path points from the path point buffer. Using the "stop abrupt" option may cause the motor to jerk to a stop, but the "stop smooth" command can be used to decelerate to a stop using the previously loaded acceleration value. (In addition to the Stop Motor command, any Load Trajectory command will also terminate the path mode and clear the path point buffer.)

Additional Status Data Two new types of status data are available for monitoring path mode execution. Firstly, the current number of path points left in the path point buffer can be added to the standard status packet using the Define Status or Read Status commands. (While in path mode, this number will decrease as the path executes, or increase as new points are added.)

Secondly, Bit 6 of the Auxiliary Status Byte indicates if you are currently in path mode. This bit is set when an Add Path Points command with no additional path point data is sent, and it is cleared when the path point buffer runs out, or if a Stop Motor or Load Trajectory command is sent.

Fast Path Mode Option By default the **PIC-SERVO CMC** executes path points at a rate of 30 or 60 Hz as set using the Add Path Points command. An additional control bit for the I/O Control command can be used to select a “fast path” mode which switches the allowable path frequencies from 30/60 Hz to 60/120 Hz.

Miscellaneous Enhancements

The **PIC-SERVO CMC** contains a number of enhancements in addition to the path mode capabilities:

Relative Zero Position The position counter in the **PIC-SERVO CMC** can optionally be reset relative to the last position captured in the home position register. This relieves the user from having to calculate goal positions relative to the home position. Please refer to the Command Reference below.

Relative Moves The **PIC-SERVO CMC** can optionally be commanded to move relative to its current command position. This relieves the user from the additional calculations and bookkeeping otherwise required to implement relative moves. Please refer to the Command Reference below.

Servo Position Error The servo position error can now be read directly from the **PIC-SERVO CMC**. By adding the position error to the actual motor position, users can calculate the current command position. . Please refer to the Command Reference below.

Universal Reset Address Sending a Hard Reset command to the address 0xFF will reset the **PIC-SERVO CMC** independent of its programmed individual or group addresses. This relieves users from having to reset the group addresses of all modules back to the default of 0xFF before issuing a Hard Reset command.

Improved Index Latching The **PIC-SERVO CMC** homing function will trigger on any index pulse longer than 120 nanoseconds. Note, however, that the positions are still only updated every 512 microseconds, and that the most accurate homing will require a motor speed of less than 1 encoder count per 512 microseconds.

3.0 Path Accuracy

The path accuracy of the **PIC-SERVO CMC** is more than adequate for most CNC machine control or robot control applications. For very high speed or very high accuracy applications, however, there are two types of path errors to consider: absolute path errors and timing errors.

Absolute Path Errors

Absolute path accuracy is the accuracy with which a series of calculated path points with straight line segments between them matches the actual curved path desired. For example, a circle which is approximated by only 5 path points will form a pentagon rather than a circle. The maximum error between the side of the pentagon and the circle may be quite large. A larger number of path points will produce a smaller error. In general, accuracy

of an approximated path will be a function of the number of path points used, and the radius of the curve.

Because the **PIC-SERVO CMC** uses a fixed number of points per second, moving more slowly will result in a more accurate path than moving quickly. Also, a 60 Hz path will be more accurate than a 30 Hz path. The main advantages of using a slower path, however, are that fewer path points need to be calculated, less data needs to be sent to the controllers, and the path point buffer will last twice as long .

The maximum absolute path error can be approximated by the formula:

$$\text{Error} = R \times (1 - \cos(V / (2 \times F \times R)))^\dagger$$

where R is the radius of the curve (in inches), V is the velocity of the motion (in inches/sec), and F is the path point frequency (30, 60 or 120 Hz). For example, a one inch diameter circle with a velocity of 1 inch per second and a path frequency of 30 Hz would have a maximum error of 0.00028 inches. If a frequency of 60 Hz is used, the maximum error drops to 0.000069 inches.

Timing Errors

If the timing of multiple axes is not perfectly synchronized, there will be a deviation from the desired path from the fact that one axis will be ahead or behind in time. The exact deviation will depend on the path geometry.

The first type of timing error results from multiple axes not starting at exactly the same time. When a “start path” command is issued to a group of controllers, they will all start within +/- 0.00025 seconds of one another.

The second type of timing error results from inaccuracies in the frequencies of the oscillators running on each **PIC-SERVO CMC** controller. (If all **PIC-SERVO CMC** chips are timed from the same oscillator, this error is zero.) Typical oscillator variations (for the same operating temperature) are about 10 parts per million. Therefore, after running a path for 10 seconds, for example, the timing error would be about +/-0.0001 seconds.

By adding both of these timing errors together, and then multiplying by the path velocity, we get the total distance that one axis can be ahead of another axis. For a 10 second motion, while moving at 1 inch per second, we could have one axis moving ahead of another by at most 0.00035 inches. The actual worst case deviation (moving along a 45 degree angle) will produce an error from the ideal path of 0.00025 inches. Over a total distance of 10 inches traveled, this gives a basic accuracy of ±0.000025 inches per inch of travel. Other examples, of course, will produce different accuracy figures.

[†] The cosine function should be executed for an angle in radians.

Note that errors due to timing only accumulate during a coordinated motion and are, in essence, reset with each new move. Therefore, if errors due to timing do become a problem, the paths should be broken up into shorter moves.

4.0 PIC-SERVO CMC COMMAND REFERENCE

The **PIC-SERVO CMC** has essentially the same command set as the standard **PIC-SERVO** with the addition of one command for adding path points and starting the path mode. Enhancements have also been made to several of the standard **PIC-SERVO** commands. You will need to refer to the standard **PIC-SERVO** data sheet in addition to the command descriptions below.

4.1 New or Modified Commands

Reset Position *New: Reset relative to the home position*

Command value: 0x0

Number of data bytes: 1

Command byte: **0x10**

Data byte: Bit 0 = 0 - define current position to be zero
Bit 0 = 1 - define position in home register to be zero
Bits 1-7 - don't care (clear to zero)

Description:

Resets the 32 bit encoder counter to 0. Also resets the internal command position to 0 to prevent the motor from jumping abruptly if the position servo is enabled. Do *not* issue this command while executing a trapezoidal profile motion. Bit 0 of the data byte used to determine whether the current position will be defined as zero, or whether the position stored in the internal home position register is defined as zero. To maintain compatibility with earlier versions, a command byte of 0x00 can be used with no additional data byte sent, and by default, the current motor position will be defined to be zero. (Please refer to the “set home mode” command in the **PIC-SERVO** data sheet.)

Define Status *New: Receive position error and number of path points as status data*

Command value: 0x2

Number of data bytes: 1

Command byte: **0x12**

Data bytes:

1. Status items: (default: 0x00)

Bit 0: send position (4 bytes)

1: send A/D value (1 byte)

2: send actual velocity (2 bytes - no fractional component)

3: send auxiliary status byte (1 byte)

4: send home position (4 bytes)

5: send device ID, version number (2 bytes)

(**PIC-SERVO CMC** device ID = 0, ver = 5)

6: send position error* (2 bytes)

7: send number of points in the path buffer* (1 byte)

Description:

Defines what additional data will be sent in the status packet along with the status byte. Setting bits in the first data byte will cause the corresponding additional data bytes to be sent after the status byte. The status data will always be sent in the order listed. For example if bits 0 and 3 are set, the status packet will consist of the status byte followed by four bytes of position data, followed by the aux. status byte, followed by the checksum. The status packet returned in response to *this* command will include the additional data bytes specified. On power-up or reset, the default status packet will include only the status byte. (See section 5.3 for a definition of the status byte and the auxiliary status byte.) *Features listed with a * are only available after a stop command has been issued with the “enable advanced features” bit set. (Please refer to the “stop motor” command.)*

Read Status *New: Receive position error and number of path points as status data*

Command value: 0x3

Number of data bytes: 1

Command byte: **0x13**

Data bytes:

1. Status items:

Bit 0: send position (4 bytes)

1: send A/D value (1 byte)

2: send actual velocity (2 bytes - no fractional component)

3: send auxiliary status byte (1 byte)

4: send home position (4 bytes)

5: send device ID, version number (2 bytes)

(**PIC-SERVO CMC** device ID = 0, ver = 5)

6: send position error* (2 bytes)

7: send number of points in the path buffer* (1 byte)

Description:

This is a non-permanent version of the “define status” command. The status packet returned in response to *this* command will incorporate the data bytes specified, but subsequent status packets will include only the data bytes previously specified with the “define status” command. *Features listed with a * are only available after a stop command has been issued with the “enable advanced features” bit set. (Please refer to the “stop motor” command.)*

Load Trajectory *New: Ability to specify relative motions.*

Command value: 0x4

Number of data bytes: $n = 1-14$

Command byte: **0xn4**

Data bytes:

1. Control byte:

Bit 0: load position data ($n += 4$ bytes)

1: load velocity data ($n += 4$ bytes)

2: load acceleration data ($n += 4$ bytes)

3: load PWM value ($n += 1$ bytes)

4: servo mode - 0 = PWM mode, 1 = position servo

5: profile mode - 0 = trapezoidal profile, 1 = velocity profile

6: in velocity & PWM modes: 0 = FWD direction, 1 = REV direction
in trapezoidal mode:

0 = position data is absolute, 1 = position data is relative

7: start motion now

Description:

All motion parameters are set with this command. Setting one of the first four bits in the control byte will require additional data bytes to be sent (as indicated) in the order listed. The position data (range* $\pm 0x7FFFFFFF$) is only used as the goal position in trapezoidal profile mode. The velocity data (range $0x00000000$ to $0x7FFFFFFF$) is used as the goal velocity in velocity profile mode or as the maximum velocity in trapezoidal profile mode. The acceleration data (range $0x00000000$ to $0x7FFFFFFF$) is used in both trapezoidal and velocity profile mode. The PWM value (range $0-0xFF$), used only when the position servo is not operating, sends a raw PWM values directly to the amplifier. (The PWM value is reset to 0 internally on any condition which automatically disables the position servo.)

Bit 4 of the control byte specifies whether the position servo should be used or if the PWM mode should be entered.

Bit 5 specifies whether a trapezoidal profile motion should be initiated or if the velocity profiler is used. Trapezoidal profile motions should only be initialized when the motor velocity is 0. (Bit 0 of the status byte indicates when a

* While the position may range from $-0x7FFFFFFF$ to $+0x7FFFFFFF$, the goal position should not differ from the current position by more than $0x7FFFFFFF$.

trapezoidal profile motion is complete, or in velocity mode, when the command velocity has been reached.)

Bit 6 indicates the velocity or PWM direction when velocity or PWM modes are selected, but in trapezoidal mode, it indicates if the position data is absolute or relative to the current command position.

If bit 7 is set, the command will be executed immediately. If bit 7 is clear, the command data will be stored in a buffer and it will be executed when the next Start Motion command is issued.

Example: To load only new position data and acceleration data but not start the motion yet, the command byte would be 0x94, the control byte would be 0x15, followed by 4 bytes of position data (least significant byte first), followed by 4 bytes of acceleration data.

The ability to specify relative moves when in trapezoidal profile mode is only available after a stop command has been issued with the “enable advanced features” bit set. (Please refer to the “stop motor” command.)

Stop Motor *New: Enable Advanced Features bit allow strict compatibility with other versions of the **PIC-SERVO**.*

Command value: 0x7

Number of data bytes: 1 or 5

Command byte: **0x17 or 0x57**

Data bytes:

1. Stop control byte

Bit 0: Amplifier enable

1: Turn motor off

2: Stop abruptly

3: Stop smoothly

4: Stop here

5: 1 = enable advanced features,

0 = strict v.4 compatibility

6, 7: Clear to 0

2-5 Stopping position (only required if bit 4 above is set)

Description:

Stops the motor in the specified manner. If bit 0 of the Stop Control Byte is set, AMP_EN will be set; if bit 0 is cleared, AMP_EN will be cleared, regardless of the state of the other bits.

If bit 1 is set, the position servo will be disabled, the PWM output value will be set to 0, and bits 2, 3 and 4 are ignored.

If bit 2 is set, the current command velocity and the goal velocity will be set to zero, the position servo will be enabled, and velocity mode will be entered. If the velocity servo was previously disabled, the motor will simply start servoing to its current position. If the motor was previously moving in one of the profiling modes, it will stop moving abruptly and servo to its current position. This stopping mode should only be used as an emergency stop where the motor position needs to be maintained.

A more graceful stop mode is entered by setting bit 3 - this sets the goal velocity to 0 and enters velocity mode, causing the motor to decelerate to a stop at the current acceleration rate.

If bit 4 is set, the motor will move to the specified stopping position abruptly with no profiling. This mode can be used to cause the motor to track a continuous string of command positions. Note that if the stopping position is too far from the current position, a position error will be generated.

Only one of the bits 1, 2, 3 or 4 should be set at the same time. Note: the Stop Motor command must be issued initially to set AMP_EN (if used) before other motion commands are issued.

Bit 5 controls whether the advance features of the **PIC-SERVO-CMC** are enabled. Advanced features will stay enabled regardless of subsequent stop commands, until the chip is reset. Note that even with the advanced features bit set, the **PIC-SERVO CMC** should be compatible with most code written for the standard **PIC-SERVO**; the advanced features option is designed to prevent unwanted advanced features from being executed inadvertently by older host code which may have set certain bits which were previously defined as “don’t care”.

Add Path Points *New Command: Adding points to the internal path point buffer allows users to create custom multi-axis motion profiles and execute them synchronously.*

Command value:	0xD
Number of data bytes:	$n = 0, 2, 4, 6, 8, A, C$ or E
Command byte:	0xnD
Data bytes:	
0,1	Incremental data for path point 1 ($n \geq 2$)
2,3	Incremental data for path point 2 ($n \geq 4$)
.	
.	
12, 13	Incremental data for path point 7 ($n = 0xE$)
-or-	
none	Starts execution of path point mode ($n = 0$)

The Add Path Points command allow the user to add path points to the **PIC-SERVO CMC**'s internal path point buffer. Up to 7 path points may be added at with a single command,

and multiple commands may be issued to load a total of 96 path points. (Bit 7 of the Read Status or the Define Status control byte can be used to have the number of path points in the buffer returned in the status packet.) Path point data is specified as a 16 bit integer, with 14 of those bits specifying the *absolute distance from the previous path point to the current path point*³. The additional 2 bits are used to specify: 1) the direction of the incremental position data (*i.e.*, whether the path point is in front of or behind the previous point), and 2) if it should take 1/30th second or 1/60th second to get to the path point from the previous path point. Normally, just one timing mode is used in a path, but the timing can be mixed. The data format for each pair of data bytes is as follows:

30 Hz Path:

$P_{13} P_{12} P_{11} P_{10} P_9 P_8 P_7 P_6$ <i>most significant byte</i>	$P_5 P_4 P_3 P_2 P_1 P_0 F D$ <i>least significant byte</i>
---	--

60 Hz Path:

$P_{12} P_{11} P_{10} P_9 P_8 P_7 P_6 P_5$ <i>most significant byte</i>	$P_4 P_3 P_2 P_1 P_0 0 F D$ <i>least significant byte</i>
--	--

where $P_0 - P_{13}$ are the 14 bits of incremental position data, F is the path frequency (0 = 60 Hz, 1 = 30 Hz) and D is the direction (0 = forward, 1 = reverse). Note that for a 60 Hz path, the position data bits are shifted to the left, and bit 2 is always zero. As with other all other types of multi-byte data, the least significant byte is always sent first.

To actually start execution of the path in the path point buffer, an Add Path Points command is issued with no path point data included. To start several **PIC-SERVO CMC** controllers simultaneously, the Add Path Points command (with no path point data) should be sent to the group address of the group containing all the **PIC-SERVO CMC** controllers involved.

If “fast path” mode has been selected using the I/O Control command, bit F will be set to 0 for 120 Hz or 1 for 60 Hz, and the path point data will have the following format:

60 Hz Path (fast path mode):

$P_{12} P_{11} P_{10} P_9 P_8 P_7 P_6 P_5$ <i>most significant byte</i>	$P_4 P_3 P_2 P_1 P_0 0 F D$ <i>least significant byte</i>
--	--

120 Hz Path (fast path mode):

$P_{11} P_{10} P_9 P_8 P_7 P_6 P_5 P_4$ <i>most significant byte</i>	$P_3 P_2 P_1 P_0 0 0 F D$ <i>least significant byte</i>
---	--

Section 5.0 gives a concrete example for generating and downloading a path.

³By using incremental position data rather than absolute position data, the amount of path data which will fit in the **PIC-SERVO CMC**'s buffer is doubled.

I/O Control

Command value: 0x8
Number of data bytes: 1
Command byte: **0x18**
Data bytes:

1. I/O control byte
 - Bit 0: Output value of Limit1
 - 1: Output value of Limit2
 - 2: Direction of Limit1 (0 = output, 1 = input)
 - 3: Direction of Limit2 (0 = output, 1 = input)
 - 4,5: unused, set to 0
 - 6: Fast path mode (0 = 30/60 Hz, 1 = 60/120 Hz)
 - 7: unused, set to 0

Description:

Controls whether the limit1 and 2 signals are inputs (default) or outputs and sets the output values. This command is also used for selecting “fast path” mode.

Hard Reset *New: A reset command sent to the address 0xFF will always be executed.*

Command value: 0xF
Number of data bytes: 0
Command byte: **0x0F**
Description:

Resets the control module to its power-up state. *No status packet will be returned.* Typically, this command is issued to a group address for a group containing all the modules on the network, thus resetting all modules simultaneously. (Resetting individual modules can be very tricky.)

Unlike other NMC controllers, the **PIC-SERVO CMC** will execute this command sent to the address 0xFF, even if its group address has been set to some other value. This eliminated the need to set the group address for the controllers back to the default of 0xFF before issuing a Hard Reset command.

Set Homing Mode

Warning: The Set Homing Mode command options to Stop Abruptly or Stop Smoothly automatically when homing is triggered do not function when operating in coordinated path control mode. If you need to stop automatically when limit switches are triggered while in path mode, use the Motor Off option. Once you have detected the motor has turned off, use the Stop Motor command with the Stop Abrupt option to re-enable the servo.

In velocity and trapezoidal position modes, all automatic stopping modes associated with the homing command work as expected. Please see the **PIC-SERVO/PIC-ENC** data sheet for details on the Set Homing Mode command.

4.2 Status Byte Definitions

Bit 6 of the Auxiliary Status Byte (undefined for the standard **PIC-SERVO**) is defined as the “path mode” bit. This bit is set while the **PIC-SERVO CMC** is currently executing the path in its path point buffer. This bit is cleared when the path point buffer is emptied, or when a Stop Motor or Load Trajectory command is sent.

5.0 Path Mode Example

Table 1 below shows all the data for a smooth trapezoidal path for a single motor moving a carriage from a position of 0.0 to 2.0 inches. A path frequency of 30 Hz is used. (The path point numbers in boldface indicate the periods of acceleration or deceleration.) The last two columns of the table contain (in hex format) the exact position data with the frequency and direction bits set. Assuming that the address for the corresponding motor controller is 0x01, the command string to download the first seven path points would be as follows:

0xAA		Header byte
0x01		Address byte
0xED		Add Path Point Command byte (14 bytes of data)
0x5A	0x00	Path point 1
0xB6	0x00	Path point 2
0x0A	0x01	Path point 3
0x66	0x01	Path point 4
0xBE	0x01	Path point 5
0x1A	0x02	Path point 6
0x6E	0x02	Path point 7
0xBB		8-bit Checksum (does not include header)

Note that less than 7 path points can be added, as long as the upper nibble of the command byte, indicating the number of additional data bytes, is adjusted accordingly. After several path points have been added, you can actually start the path execution by issuing the command string:

0xAA		Header byte
0x01		Address byte
0x0D		Add Path Point Command byte (no additional data)
0x0E		8-bit Checksum (does not include header)

The path will continue to execute as long as new path points are added before the **PIC-SERVO CMC** reaches the last point added.

Path Mode Example: Single Axis Trapezoidal Motion

Path Length (inches) 2.00
 Encoder counts per inch 10000.00
 Maximum Velocity (in/sec) 1.00
 Acceleration (in/sec/sec) 2.00
 Path Frequency (Hz) 30.00
 Tick Time 0.033
 Maximum Velocity (counts/tick) 333.333
 Acceleration (counts/tick/tick) 22.22
 Starting Position (counts) 0.00
 Goal Position (counts) 20000.00

<i>Path Point</i>	<i>Current Velocity (counts/tick)</i>	<i>Position (counts)</i>	<i>Integer Position (counts)</i>	<i>Distance from Prev. Point (counts)</i>	<i>Most Significant Byte (binary)</i>	<i>Least Significant Byte (binary)</i>	<i>Most Significant Byte (hex)</i>	<i>Least Significant Byte (hex)</i>
(start)	0.00	0	0					
1	22.22	22.22	22	22	00000000	01011010	00	5A
2	44.44	66.67	67	45	00000000	10110110	00	B6
3	66.67	133.33	133	66	00000001	00001010	01	0A
4	88.89	222.22	222	89	00000001	01100110	01	66
5	111.11	333.33	333	111	00000001	10111110	01	BE
6	133.33	466.67	467	134	00000010	00011010	02	1A
7	155.56	622.22	622	155	00000010	01101110	02	6E
8	177.78	800.00	800	178	00000010	11001010	02	CA
9	200.00	1000.00	1000	200	00000011	00100010	03	22
10	222.22	1222.22	1222	222	00000011	01111010	03	7A
11	244.44	1466.67	1467	245	00000011	11010110	03	D6
12	266.67	1733.33	1733	266	00000100	00101010	04	2A
13	288.89	2022.22	2022	289	00000100	10000110	04	86
14	311.11	2333.33	2333	311	00000100	11011110	04	DE
15	333.33	2666.67	2667	334	00000101	00111010	05	3A
16	333.33	3000.00	3000	333	00000101	00110110	05	36
17	333.33	3333.33	3333	333	00000101	00110110	05	36
18	333.33	3666.67	3667	334	00000101	00111010	05	3A
19	333.33	4000.00	4000	333	00000101	00110110	05	36
20	333.33	4333.33	4333	333	00000101	00110110	05	36
21	333.33	4666.67	4667	334	00000101	00111010	05	3A
22	333.33	5000.00	5000	333	00000101	00110110	05	36
23	333.33	5333.33	5333	333	00000101	00110110	05	36
24	333.33	5666.67	5667	334	00000101	00111010	05	3A
25	333.33	6000.00	6000	333	00000101	00110110	05	36
26	333.33	6333.33	6333	333	00000101	00110110	05	36
27	333.33	6666.67	6667	334	00000101	00111010	05	3A
28	333.33	7000.00	7000	333	00000101	00110110	05	36

29	333.33	7333.33	7333	333	00000101	00110110	05	36
30	333.33	7666.67	7667	334	00000101	00111010	05	3A
31	333.33	8000.00	8000	333	00000101	00110110	05	36
32	333.33	8333.33	8333	333	00000101	00110110	05	36
33	333.33	8666.67	8667	334	00000101	00111010	05	3A
34	333.33	9000.00	9000	333	00000101	00110110	05	36
35	333.33	9333.33	9333	333	00000101	00110110	05	36
36	333.33	9666.67	9667	334	00000101	00111010	05	3A
37	333.33	10000.00	10000	333	00000101	00110110	05	36
38	333.33	10333.33	10333	333	00000101	00110110	05	36
39	333.33	10666.67	10667	334	00000101	00111010	05	3A
40	333.33	11000.00	11000	333	00000101	00110110	05	36
41	333.33	11333.33	11333	333	00000101	00110110	05	36
42	333.33	11666.67	11667	334	00000101	00111010	05	3A
43	333.33	12000.00	12000	333	00000101	00110110	05	36
44	333.33	12333.33	12333	333	00000101	00110110	05	36
45	333.33	12666.67	12667	334	00000101	00111010	05	3A
46	333.33	13000.00	13000	333	00000101	00110110	05	36
47	333.33	13333.33	13333	333	00000101	00110110	05	36
48	333.33	13666.67	13667	334	00000101	00111010	05	3A
49	333.33	14000.00	14000	333	00000101	00110110	05	36
50	333.33	14333.33	14333	333	00000101	00110110	05	36
51	333.33	14666.67	14667	334	00000101	00111010	05	3A
52	333.33	15000.00	15000	333	00000101	00110110	05	36
53	333.33	15333.33	15333	333	00000101	00110110	05	36
54	333.33	15666.67	15667	334	00000101	00111010	05	3A
55	333.33	16000.00	16000	333	00000101	00110110	05	36
56	333.33	16333.33	16333	333	00000101	00110110	05	36
57	333.33	16666.67	16667	334	00000101	00111010	05	3A
58	333.33	17000.00	17000	333	00000101	00110110	05	36
59	333.33	17333.33	17333	333	00000101	00110110	05	36
60	333.33	17666.67	17667	334	00000101	00111010	05	3A
61	311.11	17977.78	17978	311	00000100	11011110	04	DE
62	288.89	18266.67	18267	289	00000100	10000110	04	86
63	266.67	18533.33	18533	266	00000100	00101010	04	2A
64	244.44	18777.78	18778	245	00000011	11010110	03	D6
65	222.22	19000.00	19000	222	00000011	01111010	03	7A
66	200.00	19200.00	19200	200	00000011	00100010	03	22
67	177.78	19377.78	19378	178	00000010	11001010	02	CA
68	155.56	19533.33	19533	155	00000010	01101110	02	6E
69	133.33	19666.67	19667	134	00000010	00011010	02	1A
70	111.11	19777.78	19778	111	00000001	10111110	01	BE
71	88.89	19866.67	19867	89	00000001	01100110	01	66
72	66.67	19933.33	19933	66	00000001	00001010	01	0A
73	44.44	19977.78	19978	45	00000000	10110110	00	B6
74	22.22	20000.00	20000	22	00000000	01011010	00	5A
75	0.00	20000.00	20000	0	00000000	00000010	00	02

Table 1 - Data for a Trapezoidal Motion

6.0 HOST SOFTWARE

The **J.R. Kerr Automation Engineering** web site (www.jrkerr.com) provides two levels of software support for the **PIC-SERVO CMC** for Windows 95/98/NT/2000. The first is a new version of the NMC communications DLL, NMCLIB02.DLL. (We also provide the C source code for this DLL.) This DLL includes support for all the standard **PIC-SERVO** and **PIC-I/O** functions as well as new functions for adding path points and monitoring path execution.

The second level of software support is through several example programs demonstrating the complete implementation of coordinated motion control. The first example, CIRCDEMO is for the simple execution of a circular path, including acceleration and deceleration at the beginning and end of the circle.

Two additional example programs are provided based on Pathlib.dll which contains a collection of 2 or 3-axis path generation functions. These path generation functions allow the user to create complex paths from line segments and circular arc segments. The first of these examples, PATHDEMO is a simple demonstration of the use of path generation functions. The second example, PSCNC, is a 3-axis milling machine controller with G-Code Interpreter. The C source code for these programs and for Pathlib.dll are provided as well.

Please note that these programs are provided as programming examples only, and are not supported as working applications.