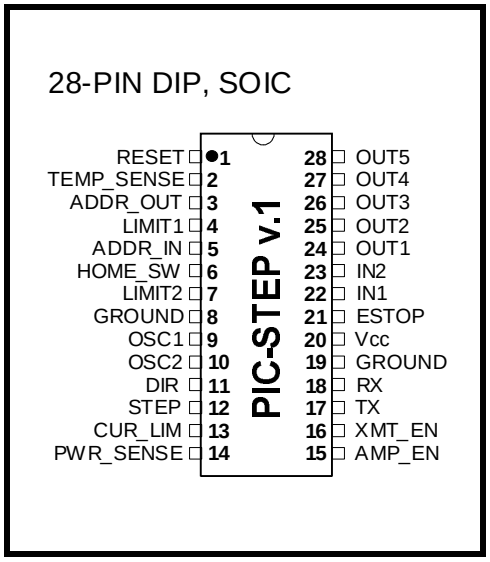
J R KERR
AUTOMATION
ENGINEERING

PIC-STEP Motion Controller

Step & Direction Signal Generator

1.0 Overview



The ***PIC-STEP*** motion controller is high-speed step and direction signal generator for use with full, half, or microstepping stepper motor drivers, with step rates up to 50,000 step/second. It supports an RS232 or multi-drop RS485 serial interface using the same NMC communications protocol as the ***PIC-SERVO*** and ***PIC-I/O*** controllers, supporting up to a total of 32 controllers. Its operating modes include profiled velocity mode, unprofiled velocity mode, trapezoidal profile position mode and unprofiled position mode. It has 2 limit switch inputs, one homing switch input, a stop input, a thermistor or A/D input, and two undedicated inputs. The driver interface includes an amplifier enable output, an analog current limit output, plus 5 undedicated outputs which can be used for amplifier control, or other purposes.

The profiled modes (velocity and trapezoidal) can specify integer velocities in the range of 1 - 250. Unprofiled velocities can be specified to within the resolution of the ***PIC-STEP***'s internal 16 bit timer. Four different speed modes allow velocities as low as 25 steps/second (even lower in unprofiled modes) and as high as 50,000 steps per second. The ***PIC-STEP*** uses a 32 bit position counter, enabling very long motions to be executed.

2.0 Specifications

2.1 Chip Description

The ***PIC-STEP*** is based on the Microchip PIC16C73 microcontroller. It is available in either a 28 pin DIP package (0.3" width) or a 28 pin SOIC package. It can be clocked from a 20 MHz crystal or TTL oscillator. It uses a supply voltage of 5v DC. Outputs and inputs are TTL and CMOS compatible. Please refer to the Microchip PIC16C73 data sheet (www.microchip.com) for complete electrical and physical specifications.

• • • CAUTION • • •

The ***PIC-STEP*** is not warranted as a fail-safe device. As such, it should not be used in life support systems or in other devices where its failure or possible erratic operation could cause bodily injury or loss of life.

2.2 Ordering Information

Part Number	*Description*
KAE-T3V1-DP	***PIC-STEP*** I.C., version 1, 0.3" wide DIP package
KAE-T3V1-SO	***PIC-STEP*** I.C., version 1, SOIC package

Table 1 - ***PIC-STEP*** part numbers

2.3 Pin Definitions

Pin Number	*Description*
1	RESET input. Pull LOW to reset the chip
2	TEMP_SENS analog input (0 - 5v). This input can be tied to a thermistor bridge which lowers its voltage signal as the temperature rises.
3	ADDR_OUT output. This output is initialized to a HIGH state, and drops and stays LOW when a Set Address command is issued.
4	LIMIT1 input. This input is used as the forward limit switch input, and can be used for homing or as a general purpose input bit as well.
5	ADDR_IN input. Communications with the ***PIC-STEP*** will be disabled until this input goes LOW.
6	HOME_SW input. This input can be used for homing or as a general purpose input bit.
7	LIMIT2 input. This input is used as the reverse limit switch input, and can be used for homing or as a general purpose input bit as well.
8	Ground
9	OSC1 - This input accepts a 20 MHz clock input, or can be connected to one side of a 20 MHz crystal.
10	OSC2 - This pin can be connected to the other side of a 20 MHz crystal, and should be left unconnected if a TTL clock input is used.
11	DIR output. This signal is used by the stepper motor driver to set the direction of motion. DIR = LOW for forward motion, DIR = HIGH for reverse motion.
12	STEP output. This output is a rising pulse with a duration of 4.8 microseconds. It is used by the stepper motor driver to increment or decrement the position by one step.
13	CUR_LIM output. This is a 40 KHz *pulse width modulated* output (magnitude = 5v) which is used to set the current limit reference voltage for the stepper motor driver. This signal should be filtered with a resistor / capacitor filter and voltage divider to produce an analog voltage of the appropriate range for your stepper driver.
14	PWR_SENSE input. A LOW value on this pin will cause the AMP_EN output to be lowered, and any motion to be terminated. It should be HIGH for normal operation.
15	AMP_EN output. This output is set HIGH to enable the stepper motor driver and LOW to disable it.
16	XMT_EN output. This output is used to disable the output driver for an RS485 transceiver. It can be left unconnected for RS232 communications.
17	TX output. This output should be tied to the transmit input of an RS485 or RS232 transceiver chip.
18	RX input. This input should be tied to the receive output of an RS485 or RS232 transceiver chip.
19	Ground
20	Vcc - Tie to +5vdc.
21	ESTOP input. A HIGH input on this pin will terminate any motion.
22	IN1 input. General purpose input pin.
23	IN2 input. General purpose input pin.
24 - 28	OUT1 - OUT5 outputs. These general purpose outputs can be tied to control inputs of the stepper motor driver to set driver specific operating modes. Outputs 1-4 are initialized LOW, output 5 is initialized HIGH.

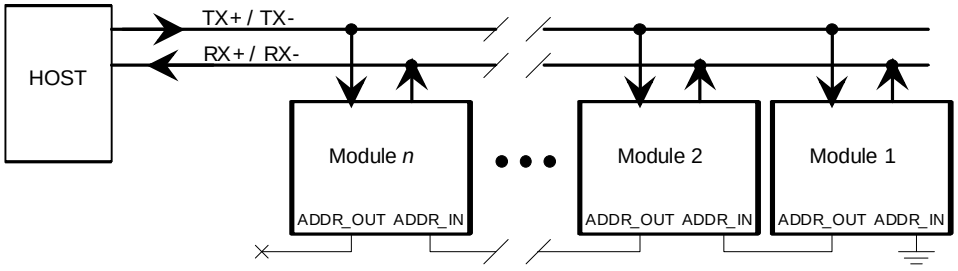
Table 2 - ***PIC-STEP*** Pin Definitions.

3.0 Communications

3.1 NMC Communication Protocol

The ***PIC-STEP*** uses the same *full-duplex*, RS485 based NMC (*Networked Modular Control*) communication protocol as used by the ***PIC-SERVO*** and the ***PIC-I/O*** controllers. It is a strict master/slave protocol, where command packets are sent to a controller module by the host computer, and a status packet is returned by the module. The default baud rate is 19,200, but it may be changed at any time to up to 115,200. The communication protocol uses 1 start bit, 1 stop bit and no parity.

Command packets are transmitted by the host over a dedicated command line. Status packets are received over a *separate* status line which is shared by all of the modules on the network. Because the host does not have to share the command line, the host communications port can be a standard RS232 port with a simple RS232 to RS485 signal level converter. The slave ports, however, must be able to disable their transmitters to prevent data collisions over the shared status line. Therefore, all NMC compatible controllers provide an XMT_EN output used for enabling or disabling an RS485 transmitter. Please refer to the sample schematic in Section 6 and to the figure below.



The command packets have the following structure:

Header byte (always 0xAA)
Module Address byte (0 - 255)
Command byte
Additional Data bytes (0 - 15 bytes)
Checksum byte (8-bit sum of the Module Address byte thru the last additional data byte)

The Header byte is used to signal the beginning of a command packet. When waiting for a new command, each module will ignore any incoming data until it sees a Header byte.

The Module Address byte is the address of the target module. The address can be an individual address, or the *group* address for the module. (See *Group Commands* below.)

The Command byte is broken up into an upper nibble (4 bits) and lower nibble (4 bits). The lower nibble contains the command value (0 - 15), and the upper nibble contains the number of additional data bytes required for that command (0 - 15). It is up to the host to insure that the upper nibble matches the number of additional data bytes actually sent.

The Additional Data bytes contain the specific data which may be required for a particular command. It is up to the host to make sure that the proper number of additional data bytes is sent for a particular command, and that the upper nibble of the command byte is equal to this number.

Once a module receives a complete packet, and the Address byte matches its address, it will verify the checksum and immediately send a status packet in return. If there is a checksum error in the command packet or any other sort of communications error (framing, overrun), the command will not be executed, but a status packet will still be returned. If there are no errors, the command will then be executed.

The status packets have the following structure:

Status byte
Additional Status Data bytes (optional)
Checksum byte (8-bit sum of all the bytes above)

The Status byte contains basic information about the state of the module, including whether or not the previous command had a checksum error. The specific bit definitions for the Status byte are in Section 5.1 below.

The Additional Status Data bytes are optional, and may contain information such as motor position, input bit values, or the module type and version numbers. Exactly which data is included in these Additional Status Bytes can be programmed using the Define Status or Read Status commands. On power-up or reset, each NMC module defaults to sending only the Status byte and Checksum byte, with no additional status data.

The first byte of a status packet is sent within 0.5 milliseconds after the last byte of the command packet has been received. The host should always wait to receive the status from one command before sending another command (except in the case of group commands described below) to prevent data collisions on the status line.

The Command Reference section below describes the data contained in the command packets and status packets.

3.2 Addressing

When multiple modules are connected to the same NMC network, they must be assigned unique addresses. This is done through the use of the ADDR_IN and ADDR_OUT signals on each NMC compatible controller. The ADDR_OUT signal from one controller is daisy-chained to the ADDR_IN signal of the adjacent controller on the network. Customarily, the ADDR_IN pin of the controller furthest from the host is tied to GND, and the ADDR_OUT signal of the controller closest to the host is left open. (See the figure above).

Unique addresses are assigned using the following procedure:

1. On power-up, all modules assume a default address of 0x00, and each will set its ADDR_OUT signal HIGH. Furthermore, a module's communications will be disabled completely until its ADDR_IN signal goes LOW. If the ADDR_OUT and ADDR_IN signals

are daisy-chained as described above, all modules will be disabled except for the module furthest from the host.
2. The host starts by sending a Set Address command to module 0, changing its address to a value of 1. A side affect of the Set Address command is that it will lower the module's ADDR_OUT signal.
3. At this point, the next module in line is enabled with an address of 0. The host then sends a command to module 0 to change its address to a value of 2.
4. This process is continued until all modules have been assigned unique addresses.

Initialization of the addresses is performed by the host each time the NMC network is powered up or reset. The host can also use this mechanism to verify that the proper number of modules are present, and that their types match those expected for a particular application.

Once addresses are set, all other operations can be executed.

3.3 Group Commands

Each NMC controller module actually has two addresses: an individual address and a group address. On power-up or reset, the individual address defaults to 0x00 and the group address defaults to 0xFF. Both the individual address and the group address are set with the same Set Address Command. Individual addresses can have any value between 0 and 255, but group addresses are restricted to values between 128 and 255.

The purpose of the group address it to be able to send a single command (such as Start Move) to a several controllers at the same time. While the individual addresses of all controllers must be unique, a group of controllers can share a common group address. When a command packet is sent over the NMC network to a group address, all modules with a matching group address will execute the command.

The issue of which modules will send a status packet in response to a group command is resolved with the distinction between group *members* and group *leaders*. When the group address for a module is set, the command will also specify if the module is to be the leader or a member of that group. If a module is a member of its group and it receives a group command (one sent to its group address), it will execute the command but *not* send back a status packet. If a module is the leader of its group and it receives group command, it *will* send back a status packet in addition to executing the command. (The status packet is just the same as one sent in response to an individually addressed command.)

For any group of modules sharing the same group address, only one should be declared the group leader.

In certain instances (as when changing the Baud rate for all modules on the network), is necessary to send a command to a group without a group leader. In this case, no status will be coming back from any controllers, and the host should wait for at least 0.5 milliseconds before sending another command to keep from overwriting the previous command.

3.4 Network Initialization

The previous subsections have hinted at various operations required for network initialization. Here is a specific list of the actions which should be taken on power-up, or after a network-wide reset† :

1. Set the host baud communications port to 19,200 Baud, 1 start bit, 1 stop bit, no parity.
2. Send out a string of 16 null bytes (0x00) to fill up any partially filled command buffers. Wait for at least 1 millisecond, and then flush any incoming bytes from the host's receive buffer.
3. Use the Set Address command, as described in Section 3.2, to assign unique individual addresses to each module. At this point, set all group addresses to 0xFF, and do not declare any group leaders.
4. Verify that the number of modules found matches the number expected.
5. Different NMC controller modules will have different type numbers and different version numbers (***PIC-STEP*** = type 3). Use the Read Status command to read the type and version numbers for each module and verify that they match the types and versions expected.
6. Send a Set Baud command to the group address 0xFF to change the baud rate to the desired value. No status will be returned.
7. Change the host's Baud rate to match the rate just specified.
8. Poll each of the individual modules (using a No Op command) to verify that all modules are operating properly at the new Baud rate.
9. Use the Set Address command to assign any group addresses as needed.

At this point you are ready to send any module specific initialization commands to the individual modules and begin operation. Note that at any time, you may use the Set Address command to re-assign group addresses.

4.0 Theory of Operation

4.1 Unprofiled Motion - The Basic Step Timer

The ***PIC-STEP*** has an internal 16 bit timer which is used to set the time between step pulses. In unprofiled operating modes, this timer is loaded with a user specified initial timer count. When the timer counts up and rolls over to a value of 0x0000, a 4.8 microsecond step pulse is generated, and the timer is reloaded with the initial timer count.

The step timer can be set to operate at one of four speed. The 1x speed is 625,000 Hz, the 2x speed is 1,250,000 Hz, the 4x speed is 2,500,000 Hz, and the 8x speed is 5,000,000 Hz. The following formulas can be used to determine the initial timer count for a desired unprofiled stepping speed, S, in steps per second:

Speed Mode	*Initial Timer Count*
1x	(65,536 - (625,000 / S)) + 2
2x	(65,536 - (1,250,000 / S)) + 4
4x	(65,536 - (2,500,000 / S)) + 8
8x	(65,536 - (5,000,000 / S)) + 16

† For most basic applications which do not use group commands or faster baud rates, only steps 1 and 3 are really required.

Table 2: Determining Initial Timer Counts for Different Speed Modes.

The initial timer count can be loaded using the Load Trajectory command described in the Command Reference section. The maximum allowable initial timer count is 65,452, and the minimum is 1. The direction of motion is specified with a Direction bit.

In conjunction with the initial timer count, a stopping position can also be loaded using the Load Trajectory command. If a stopping position is loaded at the same time as the initial timer count, the motor will move at the unprofiled stepping speed until it reaches the stop position, at which point it will stop abruptly. This mode is referred to as the *unprofiled position mode*.

Note: To change the direction of motion, a Stop Motor command must first be issued before a speed in the opposite direction is commanded.

4.2 Velocity Profile Mode

Velocity profile mode is used to smoothly accelerate from one speed to another. Instead of using initial timer count values, however, speeds are specified as integer values between 1 and 250. The direction of motion is specified with a Direction bit. The actual step rates and minimum and maximum speeds for the different speed modes appear in the table below:

Speed mode	***Step Rate Resolution (Minimum Speed) in steps/sec.***	***Max. Step Rate in steps/sec. (Speed = 250)***
1x	25	6,250
2x	50	12,500
4x	100	25,000
8x	200	50,000

Table 3: Speed Resolutions and Maximum Speeds for Different Speed Modes.

The acceleration (or deceleration) is achieved by incrementing (or decrementing) the current integer speed value by one until the goal speed is reached. An acceleration time value, in units of 0.25 milliseconds, is used to specify the time it takes to increment from one speed to the next. For example, to accelerate from a speed of 25 to a speed of 125, with an acceleration value of 4 (= 1 millisecond) would take 100 milliseconds. Speed and acceleration time values are set using the Load Trajectory command described in the Command Reference section.

Note: To change the direction of motion, a stop command must first be issued before a speed in the opposite direction is commanded.

4.3 Trapezoidal Profile Mode

Trapezoidal profile mode is used to move to a goal position by first accelerating up to a running speed, slewing at the running speed, and finally decelerating to a stop at the commanded goal position. The maximum running speed and the acceleration time are specified using the same parameters as described for the velocity profile mode. Deceleration occurs at the same rate as the acceleration. Please refer to the Load Trajectory command in the Command Reference section for details on the trapezoidal profile mode, and on the restrictions in switching between operating modes.

4.4 Homing

The ***PIC-STEP*** has built-in homing features for assisting in initializing the motor position. A special command is used to make the ***PIC-STEP*** start monitoring its limit switch inputs or its homing switch input. When one of the selected switch inputs changes state, the motor's position will be stored in an internal homing register, and motor will optionally stop in one of three specified manners: 1) decelerating to a stop, 2) stopping abruptly, or 3) turning the motor off altogether.

Once the homing condition has occurred (as determined from the homing bit in the Status byte), the home position can be read from the home position register using the Read Status command.

Note that the homing command does not actually start any movement of the motor. The homing command is first issued, and then a Load Trajectory command is used to start the motor moving in the desired direction.

4.5 Error Checking

The ***PIC-STEP*** can be set up to monitor various error conditions and stop the motor accordingly. Firstly, by default, the ***PIC-STEP*** will inhibit any forward motion if the Limit Switch 1 input is HIGH, and it will inhibit any reverse motion if the Limit Switch 2 input is HIGH. Thus, if the motor runs into a limit switch, the motion will be stopped (and optionally the motor will be turned off). Motion in the direction to move back off of the limit switch, however, will be allowed. Automatic stopping at limit switches can optionally be disabled.

Secondly, the motor will be stopped (and optionally turned off) if the ESTOP input goes HIGH. This emergency stop feature can also optionally be disabled.

Thirdly, the ***PIC-STEP*** has a power sense input which can be used to monitor the state of the power amplifier. If this input goes low, the amplifier output is lowered, and any motion is terminated.

Lastly, if the temperature sensor input is tied to a thermistor bridge (such that the input voltage goes down as the motor temperature rises), the ***PIC-STEP*** can be programmed to automatically turn off the motor to prevent overheating. An analog value of 0-255 can be programmed as a temperature threshold for automatically disabling the amplifier. The analog value can also be read directly to monitor the temperature during normal operation, and to initially determine an appropriate threshold value for your motor and thermistor. A threshold value of 0 effectively disables the thermal shutdown feature.

4.6 Current Limiting

The ***PIC-STEP***'s CUR_LIM pin puts out a 40 KHz pulse width modulated signal (5v amplitude) with a duty cycle proportional to a current limiting reference voltage. If fed through an appropriate resistor/capacitor network, it will produce a reference voltage which can be used by your stepper motor driver to set the motor current. The ***PIC-STEP*** can be programmed with two separate current limit values: a running current limit and a holding current limit, each from 0 (no current) to 255 (full current). When the motor is idle, the holding current limit will be used. When a motion is started, the ***PIC-STEP*** will automatically switch to the running current limit.

When the motor stops, the holding current limit will resume after about 0.25 seconds. Typically, the holding current limit is set at a lower level than the running current limit to reduce motor heating when the torque requirements are lower.

5.0 Command Reference

5.1 Commands

Reset Position

Command value: 0x0
Number of data bytes: 0
Command byte: **0x00**

Description:

Resets the 32 bit position counter to 0. Do *not* issue this command the motor is in motion.

Set Address

Command value: 0x1
Number of data bytes: 2
Command byte: **0x21**
Data bytes:

1. Individual address: 0-0xFF (initial value 0x00)
2. Group Address: (initial value 0xFF)

Description:

Sets the individual address and group address. Group addresses are always interpreted as being between 0x80 and 0xFF. If a ***PIC-STEP*** is to be a group *leader*, clear bit 7 of the desired group address in the second data byte; the ***PIC-STEP*** will automatically set bit 7 internally after flagging the itself as the group leader. (If bit 7 of the second data byte is set, the module will default to being a group *member*.) The first time this command is issued after power-up or reset, it will also enable communications for the next *module* in the network chain by lowering the ADDR_OUT signal.

Define Status

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 current initial timer count (2 bytes)
   3: send inputs byte (1 byte)
   4: send home position (4 bytes)
   5: send device type, version number (2 bytes)
   (***PIC-STEP*** controller device type = 3)
   6, 7: not used - clear to zero

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 inputs 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. All multi-byte data is send back least significant byte first.

The individual bits in the status byte and the inputs byte are defined in Section 5.2 below.

Read Status

Command value: 0x3
Number of data bytes: 1
Command byte: **0x13**
Data bytes:

1. Status items: (default: 0x00)
   Bit 0: send position (4 bytes)
   1: send A/D value (1 byte)
   2: send current initial timer count (2 bytes)
   3: send inputs byte (1 byte)
   4: send home position (4 bytes)
   5: send device type, version number (2 bytes)
   (***PIC-STEP*** controller device type = 3)
   6, 7: not used - clear to zero

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.

Load Trajectory

Command value: 0x4
Number of data bytes: *n* = 1-10
Command byte: **0x*n*4**
Data bytes:

1. Control byte:
   - Bit 0: load position data (*n* += 4 bytes)
   - 1: load speed (*n* += 1 bytes)
   - 2: load acceleration time (*n* += 1 bytes)
   - 3: load initial timer count (*n* += 3 bytes)
   - 4: reverse direction - 0 = forward, 1 = reverse
   - 5,6: Not used - clear to zero
   - 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* +/- 0x7FFFFFFF) is used as the goal position in trapezoidal profile mode and in the unprofiled position mode. The speed data (range 1-250) is used as the goal speed in velocity profile mode or as the maximum speed in trapezoidal profile mode. (If the goal speed is less than the minimum profile speed, the minimum profile speed will be used instead.) The acceleration data (range 1-255) is used in both trapezoidal and velocity profile mode.

If the initial timer count is loaded, a total of three additional data bytes must be sent: 2 for the 16 bit count value, followed by 1 byte which is the nearest integer speed value. If you choose to enter velocity mode and decelerate or accelerate to a new velocity, this nearest integer speed value will be used as the starting point for the ramping.

Bit 4 is used with the velocity profile mode or the unprofiled velocity mode to set the direction of motion. (Note that the direction of motion cannot be changed without first stopping the motor.) If bit 7 is set, the motion will be executed immediately. If it is not set, the command will have no effect whatsoever (and may be overwritten by another Load Trajectory command) until a Start Motion command is issued.

The mode of operation, unprofiled velocity mode, unprofiled position mode, velocity profile mode or trapezoidal profile mode, will be determined by which data is loaded using this command. In addition, there are restrictions on which modes can be entered from any given current mode. Table 4 below details which goal modes can be entered from any starting mode, and which data needs to be loaded in order to enter a specific goal mode.

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

Start Motion

Command value: 0x5
Number of data bytes: 0
Command byte: **0x05**
Description:

Causes the most recent Load Trajectory command to execute. This is useful for loading several ***PIC-STEP*** or ***PIC-SERVO*** chips with trajectory information and then starting the motions simultaneously with a group command.

Set Parameters

Command value: 0x6
Number of data bytes: 5
Command byte: **0x56**
Data bytes:

1: Operating mode
   Bits 1,0: Speed mode 00 = 8x, 01 = 4x, 10 = 2x, 11 = 1x
   2: Disable limit switch autostop
   3: Disable E-Stop
   4: Turn off motor on Limit or E-Stop
   5,6,7: Ignored, set to zero
2: Minimum profile speed (1-250)
3: Running current limit (0-255)
4: Holding current limit (0-255)
5: Thermal limit (0-255)

Description:

Sets control parameters governing the operation of the ***PIC-STEP***. This command must be issued before any motions can be executed. If this command is issued while the motor is in motion, any changes to the speed mode bits and minimum profile speed will be ignored.

Stop Motor

Command value: 0x7
Number of data bytes: 1
Command byte: **0x17**
Data bytes:

1. Stop control byte
   Bit 0: Amplifier enable
   1: not used
   2: Stop abruptly
   3: Stop smoothly
   4,5,6,7: not used - clear to zero

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 2 is set, the motor will stop abruptly at its current position.

If bit 3 is set, the motor will decelerate to a stop using the current acceleration time for the deceleration ramp. Only one of bits 2 or 3 should be set at one time.

When you stop smoothly, you are effectively setting the goal speed to zero, and when the minimum profile speed is reached, the motor will stop. Note that if, after stopping smoothly, you want to enter the trapezoidal profile mode, you will have to load a new goal velocity (along with the position) because Stop Motor command will have set the goal velocity to zero.

Note that the Stop Motor command must be issued in order to initially enable the amplifier. The amplifier can be enabled without setting any of the other stop control bits.

Set Outputs

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

1. Set Output control byte
   Bit 0: OUT1
   1: OUT2
   2: OUT3
   3: OUT4
   4: OUT5
   5,6,7: Clear to zero

Description:

Sets or clears the general purpose output pins OUT1 - OUT5.

Set Homing Mode

Command value: 0x9
Number of data bytes: 1
Command byte: **0x19**
Data bytes:

1. Homing control byte
   Bit 0: Capture home position on *change* of Limit1
   1: Capture home position on *change* of Limit2
   2: Turn motor off on home
   3: Capture home on *change* of Home Switch
   4: Stop abruptly on home
   5: Stop smoothly on home
   6,7: not used - clear to 0

Description:

Causes the controller to monitor the specified conditions and capture the home position when *any* of the flagged homing conditions occur. The home_in_progress bit in the Status byte is set when this command is issued and it is lowered when the home position has been found. Setting one (and only one) of bits 2, 4 or 5 will cause the

motor to stop automatically in the specified manner once the home condition has been triggered.

Set Baud Rate

Command value: 0xA
Number of data bytes: 1
Command byte: **0x1A**
Data bytes:

1. Baud rate divisor, BRD

sample values:

9600	BRD = 129
19200	BRD = 63
57600	BRD = 20
115200	BRD = 10

Description:

Sets the communications baud rate. All controller chips on the network must have their baud rates changed at the same time, therefore this command should only be issued to a group including all of the controllers on the network. A status packet returned from this command would be at the new baud rate, so typically (unless the host's baud rate can be accurately synchronized), there should be no group leader when this command is issued. The baud rate divisor is programmed directly into the PIC16C73's SPBRG register with the bit BRGH = 1. Please refer to the PIC16C7x data sheet for details in obtaining other baud rates.

Save Current Position as Home

Command value: 0xC
Number of data bytes: 0
Command byte: **0x0C**
Description:

Causes the current position to be saved in the home position register. This command is typically issued to a group of controllers to cause their current positions to be stored synchronously. The stored positions can then be read individually by reading the home position registers.

No Operation

Command value: 0xE
Number of data bytes: 0
Command byte: **0x0E**
Description:

Does nothing except cause a status packet with the currently defined status data to be returned.

Hard Reset

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

Resets the control module to its power-up state. No status will be returned. Typically, this command is issued to all the modules on the network. With the ***PIC-STEP***, unlike other earlier versions of NMC controllers, a Hard Reset command sent to the address 0xFF will be executed under all circumstances, even if the controller's group address has been programmed to another value. This simplifies the process of resetting the entire network of controllers.

Table 4 - Operating Mode Transition Table:

	Starting Mode				
Goal Mode	**Stopped**	**Trap. Profile**	**Vel. Profile**	**Unprofiled Velocity With Stop**	**Unprofiled Velocity, No Stop**
Stopped	---	*Use Stop Command*	*Use Stop Command*	*Use Stop Command*	*Use Stop Command*
Trap. Profile	✓ Position ○ Velocity ○ Acceleration × Tmr. Count	*Not Allowed*	*Not Allowed*	*Not Allowed*	*Not Allowed*
Vel. Profile	× Position ○ Velocity ○ Acceleration × Tmr. Count	× Position ○ Velocity ○ Acceleration ✓ Tmr. Count	× Position ○ Velocity ○ Acceleration × Tmr. Count	× Position ○ Velocity ○ Acceleration × Tmr. Count	× Position ○ Velocity ○ Acceleration × Tmr. Count
Unprofiled Velocity With Stop	✓ Position × Velocity × Acceleration ✓ Tmr. Count	× Position × Velocity × Acceleration ✓ Tmr. Count	*Not Allowed*	× Position × Velocity × Acceleration ✓ Tmr. Count	*Not Allowed*
Unprofiled Velocity, No Stop	× Position × Velocity × Acceleration ✓ Tmr. Count	*Not Allowed*	× Position × Velocity × Acceleration ✓ Tmr. Count	*Not Allowed*	× Position × Velocity × Acceleration ✓ Tmr. Count

✓ Load this parameter

× Do not load this parameter

○ Optionally load this parameter

If a parameter is not loaded, the previously loaded value will be used.

5.2 Status Byte and Inputs Byte Bit Definitions

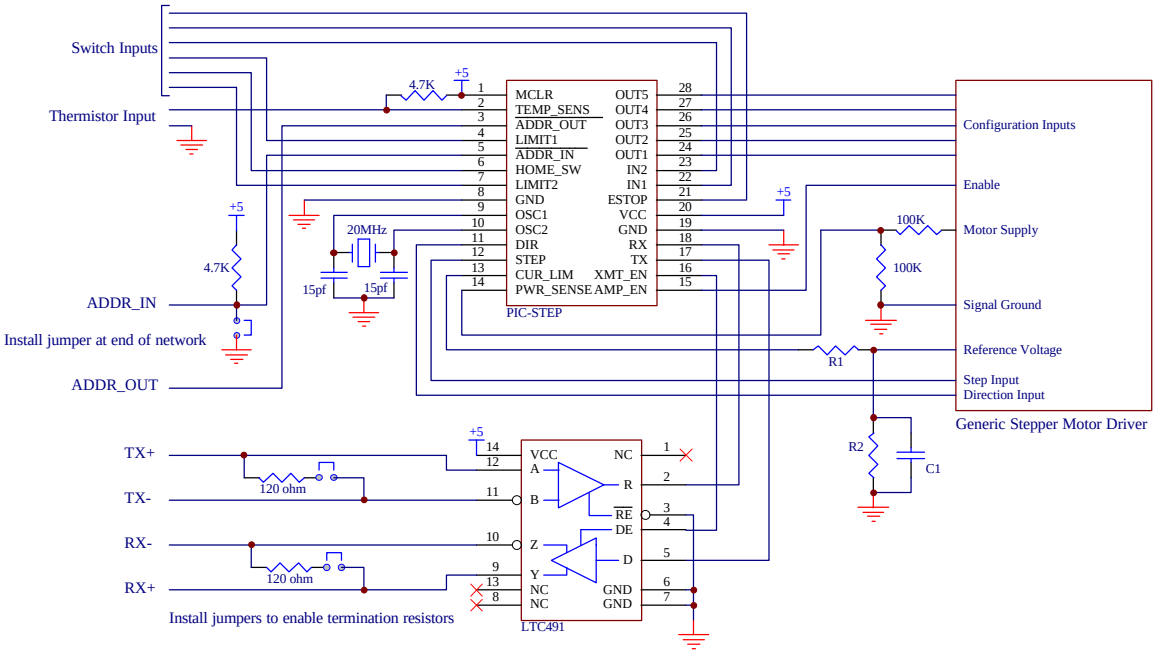
Status Byte Bit Definitions:

0 Motor is moving
1 Communications error
2 Amplifier enable output signal is HIGH
3 The power sense input signal is HIGH
4 At commanded speed
5 Velocity profile mode
6 Trapezoidal profile mode
7 Homing in progress

Inputs Byte Bit Definitions:

0 E-Stop input
1 IN1 general purpose input
2 IN2 general purpose input
3 LIMIT1 - forward limit switch
4 LIMIT2 - reverse limit switch
5 HOME_SW - homing switch input
6,7 *undefined*

6.0 Example Circuit



7.0 Additional Information

Further documentation, application notes, and example host software for the ***PIC-STEP*** and other NMC compatible controllers may be obtained from:

J R KERR Automation Engineering http://www.jrkerr.com

Complete electrical specifications for the PIC17C73B-20 microcontroller may be obtained from:

Microchip, Inc. http://www.microchip.com