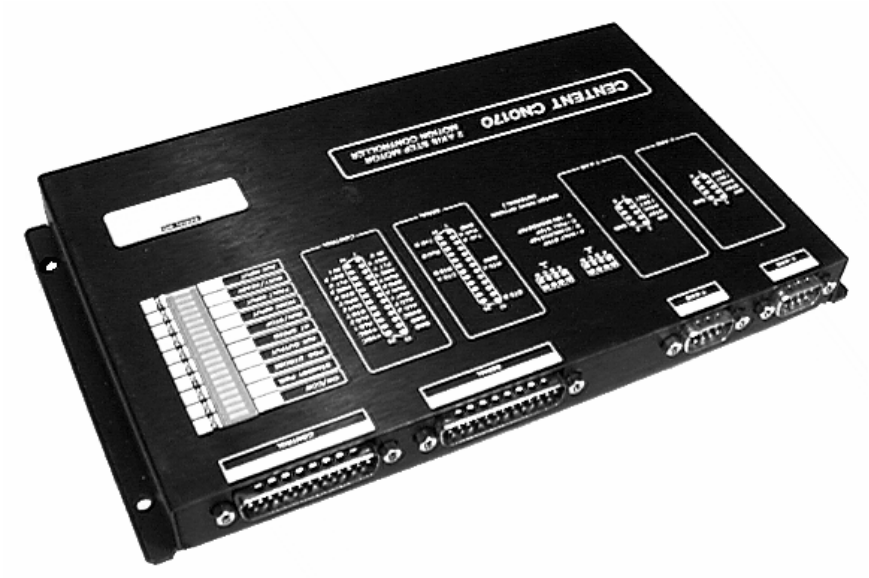
SOFTWARE MANUAL
CN0170



TWO AXIS
MOTION CONTROLLER

CENTENT
COMPANY
3879 SOUTH MAIN STREET 714-979-6491
SANTA ANA, CALIFORNIA 92707-5710 U.S.A.

This manual contains information for installing and operating the following Centent Company product:

- CN0170 Motion Controller

CONTENTS

SERIAL PORT INITIALIZATION

All programming information to the CN0170 passes through the serial port 'A'. Instruction strings are comprised of printable ASCII characters. The CN0170 must be connected to a personal computer or a terminal to program it. It may then be operated interactively with the host computer or terminal, or as a "stand alone" device, unplugged from the host.

The CN0170's RS-232 serial port is configured for eight data bits, no parity and one stop bit. The baud rate is automatically selected by the CN0170 to match the host computer. Any standard baud rate from 150 to 9600 may be used. The following steps set the CN0170's baud rate to match that of the host computer:

1. RESET the CN0170. Momentarily remove power from +VDC (Control Connector, pins 24, 25), to reset.

2. Wait 2 to 3 seconds after returning power; until RTS (Serial Connector, pin 8) goes active.

3. Send a carriage return <CR> at any standard baud rate between 300 and 9600 baud.

The CN0170 measures the baud rate from the <CR> and sets it's serial ports accordingly. The CN0170 will then transmit the following message on Port A:

U"n"<CR> {"n" is the Unit Number 0-9}

If two or more CN0170s are daisy-chained together, each unit will reply in turn with its own U"n" message.

If the host does not receive the Unit Number message an error has occurred. Possible sources of the problem are...

1. The first character sent by the host was not a carriage return.

2. The baud rate of the host does not conform to the standard rates available.

3. A serial port wiring error exists. See the HARDWARE MANUAL, CN0170 TWO AXIS MOTION CONTROLLER; Serial Connector.

Anytime the CN0170 is reset or powered-up, the first character sent to it must be a carriage return <CR>. Failure to do so will result in an improper baud rate selection.

If multiple CN0170s are daisy-chained together, all units will operate at the same baud rate.

DATA ENTRY & SYNTAX

The CN0170's operating system features a proprietary command language for motion control. The host sends the programming instructions and controls the program operation. All operating commands and instructions are comprised of ASCII character strings utilizing logical mnemonics. The program may be written and maintained with any common text editor or generated as print strings by high level program languages.

Commands or instructions always begin with a letter unique to it, followed by operators and parameters where required. Motor instructions begin with the motor identifier 'X' or 'Y', to specify the axis, followed by the parameter identifier and the numerical data if any. A terminating character completes the instruction or command. Carriage return <CR>, line feed <LF> and semi-colon ";" are all recognized as terminators. Multiple terminators (carriage return, line feed) are acceptable. The program examples that follow in this manual do not show a visible terminator. This is done for clarity, a terminator is always required.

Upper or lower case characters are accepted for axis or parameter identifiers. Numerical data may be sent with or without a decimal point. Leading zeros are ignored. Fractional data following a decimal point is rounded off to the nearest resolution applicable to the parameter in question.

The host may spell out the parameters for clarity if desired. The following program fragment:

```
XV=200,3333.3
XA=10000
XR10
XC\L
```

Is equivalent to:

```
X Velocity = 200, 3333.3
X Acceleration = 10000
X Resolution 10
X Curve\Linear
```

The first example minimizes typing while the second makes for a more readable program listing. There is no penalty in either instruction execution time or CN0170 memory requirement for the second example.

The CN0170 also accepts numerical data entered in hexadecimal format (in ASCII). Hexadecimal coded numbers are preceded by a '0' if the first character is a letter, and the hex number must be followed by the 'H' character (upper case required).

Entries for motor position, velocity or acceleration given in hex provide direct access to the motor parameter registers. They are not a straight

conversion of a decimal entry. The following table describes the CN0170's internal registers for position, velocity and acceleration.

PARAMETER	*REGISTER*	*RESOLUTION*	*UNITS*
POSITION	4 bytes	1/1024	STEPS
VELOCITY	2 bytes	1/4	STEPS/SECOND
ACCELERATION	2 bytes	64	STEPS/SECOND^2

The following gives examples of position, velocity and acceleration instructions in decimal and hexadecimal formats:

decimal	*Conversion*	hexadecimal
X=1000	*(1000)(1024)=1024000=0FA000 hex*	X=0FA000H
XV=200	*(200)(4) = 800 = 320 hex*	XV=320H
XA=12800	*(12800)/64 = 200 = 0C8 hex*	XA=0C8H

Responses from the CN0170 for position, velocity or acceleration are always returned to the host in the hexadecimal format, giving direct access to the CN0170's internal registers. For example, if the host queries the position register for the X-axis by sending:

XP?

The Cn0170 might respond:

XP=00100400h

Indicating the X-axis is located at 1025 full steps from the Home position. Notice the "h" is returned by the CN0170 in lower case format. If the decimal value is required, the host must do the conversion.

When the CN0170 receives an instruction that contains syntax errors, the entire string is rejected. Illegal ASCII strings are enclosed in quotation marks and echoed back to the host, indicating that the instruction was not understood and thus ignored. If the host were to send:

XY?

Perhaps intending to ask "XV?". The CN0170 would reply:

"XY?" ?

This indicates that the CN0170 could not interpret the instruction.

DATA BUFFERS

Buffering occurs at many levels to assure smooth operation of the CN0170 controller in conjunction with the host and other CN0170s in the system. Each RS232 transmitter and receiver has its own buffer. This prevents interleaving of data passed up or down the daisy chain by the various units in the system. These buffers are 256 characters in length.

If a CN0170's Unit Number does not match the selected number, data coming in on the Port A receiver is passed through, unaltered, to the Port B transmitter buffer. When the unit number matches the selected (active) number the data received on Port A is trapped by that CN0170 and goes no further. The data is then converted to binary formats and directed to the Program Queue (Program Mode) or Immediate Queue (Immediate Mode) as required.

Once instructions are placed in the Program Queue or Immediate Queue, they are available to the motor service routines that run concurrently with the operating system. The operating system distributes the instructions to staging areas for each motor. There are separate queues for immediate and program operation. This allows switching operation mode from Operate to Immediate without loss of instructions.

The host may interrupt a program at any time, perform Immediate Mode instructions, and then return to program operation. When the command is received to change modes during program operation, the CN0170 completes the current instruction before switching to another mode.

The queues for immediate operation will hold approximately twenty instructions. This means the host can send at least twenty instructions to a motor without blocking instructions to the other motor, regardless of the time required to execute the instructions. The staging areas for program operation will hold at least eight instructions each. Some care must be taken in program design for instruction distribution if the motors are to function individually. For coordinated instructions between motors the queues will fill and empty simultaneously.

At high baud rates, the queues may fill faster than the CN0170 can process the data. This can be controlled either by hardware handshaking (see RTS on Overflow, page 6) or by software (see Warning on Overflow, page 6).

CN0170 COMMANDS

Commands control the operation of the CN0170 and it's relationship to other units in the system. Commands are the highest order input from the host to the controller. They select modes of operation, start and stop the program, assign and select unit numbers, and set or read the clock/calendar.

The CN0170 is always in one of three modes; Immediate, Program or Operate. The Immediate Mode is the default mode on the first power-up of the CN0170. Subsequently, the CN0170 will come up in whatever mode it was in when power was removed.

Normally commands do not load to the Program Queue, however some exceptions are possible. Program operation commands sent in the Program Mode ("G","Gn","H","Q","K") will be inserted in the Program Queue. Execution of these commands will not occur immediately, but rather during the course of program operation (in the Operate Mode). The query commands (".U?","T?","D?") may also be placed in the Program Queue.

IMMEDIATE MODE SELECT M1

In Immediate Mode, instructions are executed as soon as they are received. Operating parameters sent in this mode take effect immediately.

Instructions are placed in the Immediate Queue. As the CN0170 processes and completes instructions, the next instruction moves from the Immediate Queue into staging areas for each motor and for Input/Output operation. The instructions are discarded after execution. Execution time will depend on the size of the motor movement instructions and the operating parameters in effect.

As an example, using existing motor parameters, move only the X motor 200 steps clockwise, then move only the Y motor 1000 steps counter-clockwise. The host would to send the following character sequence:

<CR>	*sets the baud rate after power-up.*
M1	*sets the IMMEDIATE mode.*
X+200&Y+0	*POINT-TO-POINT instruction for the X motor with a dimension of 200 steps and a Y motor dimension of zero steps. The X motor begins to move immediately after the terminator character is received.*
X+0&Y-1000	*POINT-TO-POINT instruction for the Y motor with a dimension of 1000 steps and an X motor dimension of zero steps. The Y motor will not begin to move until the X motor has completed the move from the previous instruction. This is because the motors are "linked" in their operation by the "&".*

The Immediate Queue holds a minimum of twenty instructions. If instructions are sent faster than the motor can complete them, the queue will fill. The host may select one of two command parameters to prevent loss of instructions due to queue overflow. They are M1\+ (RTS on Overflow) and M1\- (Warning on Overflow).

RTS ON OVERFLOW M1\+

This utilizes the RS232 hardware handshake lines to prevent instruction overflow in the CN0170. The rate at which instructions are processed from the Immediate or Program Queues is determined by the operating parameters of the motors and the dimensions of the movement instructions. The RTS output is used to inhibit the host from transmitting until sufficient memory is available for another instruction.

This method provides ease of operation, since the serial interface controls the data transmission to the CN0170. No testing is required by the host's software to start and stop transmission of data.

The disadvantage is that the host is prohibited from using the serial interface to send emergency commands (KILL or QUIT), or to communicate with other CN0170s in the system if the Immediate Queue is full.

WARNING ON OVERFLOW M1\-

When this command is active, a message is issued on serial Port A to the host to indicate that the Program Queue or the Immediate Queue is nearly full. This informs the host that the queue is within one or two instructions of overflowing.

When the "Q*" warning message is received, the host must suspend the transmission of instructions. The CN0170 can receive two additional instructions after this warning is issued. Ignoring the warning of overflow will result in data being overwritten, and the instructions lost or corrupted.

When the Immediate Queue empties, the CN0170 will signal the host by sending the "*Q" message. The host may then resume transmission of instructions to the CN0170. No "*Q" message is sent unless a "Q*" message preceded it. This prevents nuisance messages every time the queue empties.

The issuing of the overflow message does not affect the status of the RTS output of the CN0170's RS232 port. This allows the host to maintain communication with other units in the daisy chain. It is also possible to send commands to the CN0170 with the Immediate Queue full.

PROGRAM MODE SELECT M2

The Program Mode is used to download instructions and parameters to the CN0170. Execution of these instructions will not occur until the CN0170 is placed in the Operate Mode (M3) and a Go command is issued.

It is important to be aware of the current mode when selecting the Program Mode. M2 is selected either from M1 or from M3. These two possibilities yield very different results.

M1 → M2 Clears the Program Queue for a new program download. All instructions currently in the Program Queue will be erased.

M3 → M2 leaves the existing program intact; subsequent download will append to the end of the Program Queue.

The CN0170 has no program editing facilities. Modification of the program, other than appending to the end will require a complete program download. This is done by sending the following:

M1;M2

To add instructions to the end of the program without losing existing instructions, the host must enter the Program Mode from the Operate Mode (M3). This may be done by sending the following:

M3;M2

The program may be repeatedly expanded in this fashion until the Program Queue overflow message is issued by the CN0170.

The CN0170 compresses the incoming data before storage in the Program Queue. An instruction or parameter will consume between one and ten bytes of memory. The Program Queue contains approximately 30,000 bytes, so a minimum of 3000 program instructions can be stored.

Relative motor move instructions generally consume less program space than absolute ones; the dimensions of the data need not reference the entire position register. Some savings in program space may be achieved by converting absolute moves to relative ones. If the program is still too large for the queue, the host will have to break it up into several parts and reload the Program Queue during operation.

The CN0170 will notify the host of an impending memory queue overflow by sending the following message:

Q*

If the "Q*" message occurs in Program Mode the host must complete the program within one or two additional instructions.

When the download is complete the CN0170 is placed in another mode, usually the Operate Mode (M3). A GO command is then required to begin program execution.

Once the program is stored in the CN0170, the power may be removed without affecting the contents of the Program Queue. The program and operating parameters for motor operation are maintained by battery backup. The contents of the Program Queue will remain intact until one of the following two events occur.

1. The CN0170 detects data corruption during its power-up cycle, usually due to battery failure. The Program Queue is cleared and the system returns to the default operating parameters.

2. The host enters the Program Mode (M2) from the Immediate Mode (M1). The Program Queue is cleared of all prior instructions on this mode transition.

Operate Mode must be selected before the stored program can begin execution. A program must be downloaded from the host in the Program Mode (M2) before the Operate Mode will function.

Program execution can proceed with or without the host computer being present. If the CN0170 is to be operated without a host, IN1 must be specified as a GO command by sending the "P1\G" parameter, and the unit must be placed in the Operate Mode before the host is disconnected.

MODE QUERY M?

To interrogate the CN0170 for its current mode, send the following:

M? *what is the current mode?*

The reply might be:

M1 *the CN0170 is in Immediate Mode*

GO G

The GO command initiates execution of the program stored in the CN0170 Program Queue. The CN0170 must be in the Operate Mode for the Go command to take effect.

If the CN0170 remains connected to the host computer, program execution can be controlled over the RS232 interface. See the GOn, Halt, Quit and Kill commands.

When all program instructions have been completed, the program resets and waits for another Go command or other action from the host. If a program has been interrupted by the Halt command, the Go command will cause program execution to resume.

The IN1 hardware input may be designated as a Go command, allowing the CN0170 to operate with the host removed. To configure IN1 as a Go send:

P1\G

This parameter may be entered in the Program Mode (M2) or the Immediate Mode (M1). With this parameter active, taking IN1 low (momentarily) starts the program. Holding IN1 low will cause the program to repeat upon it's completion. For very short programs, a "one shot" circuit may be required to prevent false program starts.

GO n $0 < n < 65,535$ Gn

The GO n command specifies 'n' number of motor movement instructions, motor query instructions or input/output instructions to execute. Motor parameter instructions, other than queries, are not included in the value of 'n'. All input/output instructions are included.

The value of "n" goes to an internal counter called the Go Counter. Sending a simple Go command (with no 'n' parameter) disables the counter, allowing the program to execute without interruption.

The Go Counter, if active, will decrement each time the program executes an eligible instruction, as specified above. When it reaches zero program execution will halt. A Go Command or Go n Command resumes program execution.

This command is useful in testing or de-bugging a program by single stepping through it. By sending:

G1

The CN0170 will execute the next motor movement or input/output instruction and halt, waiting for another command.

The GO n command may be placed in the program by sending it during the Program Mode (M2). When the program encounters this command during Operate Mode (M3), program execution will halt after 'n' moves. This permits the program to pause after a move sequence and await user input. Consider the following program fragment:

```
XL+; YL+              set I/O linkage for X and Y
G2                    set Go Counter = 2
X+1000 & Y+500        one motor move
P3=1                  one I/O instruction

(program will halt here and wait for user input)
```

Since the X-axis and Y-axis moves are linked in the above example by the "&" operator, it is considered one move. The following program fragment will behave in a similar fashion:

```
XL+; YL+              set I/O linkage for X and Y
G3                    set Go Counter = 3
X+1000                one X-axis motor move
Y+500                 one Y-axis motor move
P3=1                  one I/O instruction

(program will halt here and wait for user input)
```

Note that this example requires a G3 instead of a G2, due to the X-axis and Y-axis instructions being separated into individual moves.

PROGRAM AUTO REPEAT

The Go 'n' command may also be used to cause the program to repeat automatically. The value of the Go Counter is not reset at the end of the program, but rather carries over and if not zero, will cause the program to start again. Consider the following program segment:

G	*disables the Go Counter; all moves execute*
X+0,Y+1000	*this move and following ones (not shown)*
…	*will execute without stopping*
G2	*enables the Go Counter, sets to 2*
X+1000	*decrements the Go Counter to 1*
<end>	*the program ends with no additional moves*

Since the Go Counter is enabled and is not zero when the program ends, it will automatically start again without a 'Go' command. In this example, program operation will continue indefinitely until interrupted by a Halt, Quit or Kill command. It is not permitted that a Go n command be the last line in the program.

If the G instruction in the example above is omitted, the Go Counter will not be disabled on restart. The X+0, Y+1000 will execute, the Go Counter then decrements to zero, and the program halts.

The Go n command can be used in conjunction with the P1\G parameter to stop and start the program when the CN0170 is operating with no host connected to the serial port. Consider the following program fragment.

P1\G	enable P1 as a hardware Go
G1	set Go Counter to 1
X+100	operation halts after this instruction
P3=1	turn output on (after P1 input is sensed)
X+0 & Y+1000	move Y-axis only
G1	set Go Counter to 1
X-100	operation halts again
P3=0	turn output off (after P1 input is sensed)

By placing a G1 before a single axis move (or a G2 before a 2 axis move) the program must halt. P1 controls the restart of the next program segment.

HALT H

This command causes program execution to stop after the current instruction is finished. The program location is preserved. When a Go command is issued, the program execution will resume at the next instruction in the program.

Sending a Halt command in the Program Mode (M2) will store the command in the Program Queue. The command will execute as it occurs in the Program Queue. Because of the staging areas and queue structure, the Halt may not occur exactly when expected. Therefore it is preferable, when operating in Mode 3, to use the Go n command to stop program execution. See the previous section for details.

QUIT Q

This command causes program execution to stop before the conclusion of the current instruction. Axis outputs will abandon the moves in progress and decelerate to zero at the programmed rate of acceleration. Position registers will remain valid, but will not have the values expected if the moves had been allowed to complete. The program location is not saved. Issuing a "G" after the "Q" will reset the program and begin execution.

Sending a Quit command in the Program Mode (M2) will store the command in the Program Queue. The command will execute as it occurs in the Program Queue.

MODULO-QUIT XM+ or XM-
YM+ or YM-

This parameter, available for the X and/or the Y-axis, is used to specify a "Modulo-Quit". As with other parameter instructions or commands, no action is taken upon receipt of this command. Modulo-Quit defines how the CN0170 will respond when the "Q" (Quit) command is received. XM+ (or YM+) enables Modulo-Quit, XM- (or YM-) disables it. The parameter will remain in effect until changed. Sending a XM+ (or YM+) does not initiate a Quit; you must send a "Q" to initiate the Quit command.

When "Modulo-Quit" is active and a "Q" command is issued, the CN0170 will attempt to stop at the first possible position that is a full step motor revolution modulus (200 full steps) from the end position of the move in progress.

If the axis cannot decelerate in time, as specified by the acceleration parameter, to stop at the first modulus position, it will continue decelerating for as many revolutions as are necessary to stop the motor at a modulus position. Setting the acceleration as high as possible will assure that the motor stops as soon as possible.

You may activate Modulo-Quit for either or both axis. If only one is active, that axis will stop at a modulus position, while the other axis will stop immediately.

Modulo-Quit will function during linear interpolation if the XM+ parameter is set. The stop position will be determined by the axis taking the larger

move. X+1000, Y+500 will position relative to the X-axis, but X+500, Y+1000 will position relative to the Y-axis.

You may switch between Modulo-Quit and normal Quit anytime. Note however that a "Q" command clears all pending instructions. If a "Q" command (Modulo or normal) is in progress while a Quit parameter (XM-, XM+, YM-, YM+) is sent to the CN0170, the parameter is lost and will not take effect. Once a "Q" is sent, it is not possible to change the type of Quit (Modulo or normal) until the axis have stopped.

KILL K

The KILL command forces both motor outputs to zero instantly. The current instructions terminate immediately and The Program Queue is reset. The KILL command functions as an emergency stop for the CN0170.

After a Kill command has been executed, the values in the Position Registers is suspect. A Home instruction is usually required after a Kill command before normal operation can continue.

> *Motor position information after a KILL command may be incorrect as a consequence of stopping an axis without deceleration ramping.*

KILL (or QUIT) commands to the CN0170 will clear all staging areas as well as the Immediate Queue. The Program Queue will be reset.

Sending a KILL command in the Program Mode (M2) will store the command in the Program Queue. The command will execute as it occurs in the Program Queue.

The KILL command is unique in that it is relayed on the serial interface to other CN0170s on the daisy chain. These units will also shut down in the manner described, bringing the entire system to a stop with a single Kill command.

IF THE SYSTEM USES ONLY ONE CN0170, YOU MAY IGNORE THE FOLLOWING SECTION.

UNIT NUMBERS

Unit numbers are required if more than one CN0170 is connected to a host utilizing a single serial port. Unit numbers provide a means of addressing a particular CN0170 to be the recipient of data sent by the host. The unit number also identifies which CN0170 is sending information back to the host.

The host assigns each unit in the system a unique unit number, and selects the unit to receive the data it is sending. One and only one unit is

selected at a time. This unit receives the commands and instructions the host is sending. The host must precede all unit commands with the "." (period) character.

A CN0170 will only 'trap' instructions and commands if it is the selected unit. If it is not, all data are sent on the Port B serial port to the next unit in the daisy chain.

If a unit number is selected that has not been assigned to any controller in the system, the data will pass through all controllers and out the Port B serial port of the last CN0170 in the chain. This allows another device to be attached to the last CN0170 and function normally, provided it can operate at the same baud rate as the CN0170s.

All messages originating from a CN0170 in the daisy chain are preceded by it's unit number, unless it is UNIT 0. The host uses this data to determine which controller in the system sent the message the host receives.

UNIT ASSIGNMENT			.U=n
	0<n<9	0<m<9	.Um=n
	0<n<9	0<m<9	.U+m=n

The first CN0170 in the daisy chain (nearest to the host's serial port) may be assigned a Unit Number of '0'. This can be done by sending:

.U=0

All other CN0170s may be sequentially numbered by sending Relative Assignment numbers. For example, if there are four CN0170s daisy-chained to the host computer, send:

.U=0
.U+1=1
.U+2=2
.U+3=3

The first CN0170 is assigned a Unit Number of '0'. When it receives the first Relative Assignment command of '.U+1=1', it subtracts one from it, converting it to a Direct Assignment of '.U=1', which it passes to the next CN0170 in line.

The second CN0170 uses '.U=1' as its Unit Number assignment. The next Relative Assignment command '.U+2=2' is converted to 'U+1=2' by the first CN0170, '.U+0=2' by the second CN0170 and interpreted as a Direct Assignment of '.U=2' by the third CN0170 in line.

This process repeats, leaving the four CN0170s with Unit Numbers 0, 1, 2 and 3.

Once the unit number for a controller is known by the host it may be reassigned directly. For example:

.U2=5

This message would result in the third unit in line being reassigned to Unit 5.

It is not necessary for the first unit in the daisy chain to be Unit 0 or for the units to be numbered consecutively. The numbering may be arbitrary and in any order.

Once all units in the system have been assigned unique numbers, they may be rearranged physically in the daisy-chain. The units will still receive the commands and instructions intended for them.

If two controllers have the same unit number assignment, the unit physically nearest the host will trap all the commands and instructions sent while that number is selected. The second unit with the same number will not receive any instructions.

UNIT SELECTION .Un

Commands, instructions or parameters destined for a particular CN0170 in multiple unit installations must be preceded by selecting its Unit Number.

Only one unit may be selected at a time. The selected unit will receive all commands and instructions until another unit is selected. The procedure is to select a unit, send it data, select another unit, send it data, etc.

For example, a motor instruction to move the X motor of UNIT 2 three hundred steps clockwise would be sent as:

.U2
X+300

All subsequent messages sent by the host will apply to unit 2 until a different Unit Number is sent.

All messages or responses to host requests from CN0170s in multiple unit installations will be preceded by the responding unit's Unit Number, unless it is Unit 0.

QUERY UNIT NUMBER .U?

The Query Unit command requests unit numbers from all units in the daisy chain. This is useful for checking how many CN0170s are on line and what their Unit Numbers are.

Sending the Query Unit command generates the following response from the four units in the Unit Assignment examples above:

[U0]
[U1]
[U5]
[U3]

REAL TIME CLOCK

The CN0170 has a twenty-four hour Real Time Clock and calendar that runs while the unit is powered up. No provision is made for maintaining the clock when power is shut off. The host may set the clock and calendar after baud rate setting has been completed.

SET TIME T=hh:mm.ss

The seconds entry is optional. If no seconds are sent, the controller sets seconds to 00. For example, to set a time of 1:32.00 PM, send:

T=13:32

SET DATE D=mm/dd/yy

The CN0170 maintains its internally calendar as long as power to the controller is maintained. Leap years are accounted for.

To set the date to Jan 2, 1997, send:

D=01/02/97

QUERY TIME T?

The host can read the clock at any time by sending the Query Time message. The reply is in the form "hh:mm.ss", where hh is hour, mm is minutes and ss is seconds. If the host queries:

T?

The response (assuming it is 8:45 AM and 11 seconds) would be:

08:45.11

This command may also be placed in the Program Queue by sending it in the Program Mode (M2).

QUERY DATE D?

The host can read the date at any time by sending the Query Date message. The response is in the form "mm/dd/yy". Sending:

D?

The response (on Feb. 15, 1997) would be:

02/15/97

This command may also be placed in the Program Queue by sending it in the Program Mode (M2).

MOTOR MOVEMENT INSTRUCTIONS

Motor instructions are of two types, motor positioning and motor parameters. Parameter instructions cause no motor movement, but rather set up the performance envelope for subsequent operation. Positioning or motion instructions directly cause the movement of one or both motors. This motion occurs in accordance with the existing parameter values. Motion instructions may treat the motors individually or coordinate their motions for linear interpolation, circular interpolation or ratio moves.

An example of a motor positioning instruction would be to move a number of steps and then stop. Examples of operating parameters would be the rate of the acceleration and the maximum velocity for the motor.

HOME XH or YH or XYH

The Home instruction reconciles the motor to a known physical location. This location is hardware set via a switch or other sensor activating the HME input on the CN0170. The motors may be homed individually or simultaneously. The current velocity, acceleration rate and acceleration profile parameters are used for the Home instruction. To Home the X-axis send:

XH

To Home both axis send:

XYH

Upon completion of the homing sequence the Position Register is set to zero. If another value is required, send the Set Position instruction after the Home has executed.

The Home instruction utilizes two special parameters. These are Home Direction and Step Resolution. Home direction is usually counter-clockwise. Step Resolution is determined by the resolution of the step motor driver in use.

SPEED CONTROL X++ or X--or Y++ or Y--
n= 0 to 16,383.75 X++n or X--n or Y++n or Y--n

The Speed Control instruction causes the motor to operate as a speed control. A clockwise direction of rotation is set by the "++" characters while counter-clockwise rotation is set by "--". To operate the X-axis as a speed control with the direction clockwise, send:

X++

The velocity and acceleration of the motor is determined by the values in effect at the time. Since no target position is specified, the velocity and acceleration parameters may be changed on the fly, with the motor moving. The motor will servo to the new value.

The Position Register is maintained for the motor during Speed Control Operation. If the motor runs long enough, the position registers will overflow or underflow.

A temporary Maximum Velocity Parameter may be included in the Speed Control instruction. When the next motor instruction is sent the motor reverts to the normal Maximum Velocity Parameter. To operate the X motor at a velocity of 5000 steps per second in the counter-clockwise direction, send:

X--5000

The motor will accelerate to 5000 steps per second. It will run at this speed until a new instruction is sent.

JOYSTICK OPERATION n is 0 - 7 XJn or YJn

The Joystick instructions place the motor under the control of the analog voltage applied to the ALG input. To place the X-axis in Joystick Mode 4 send:

XJ4

There are seven different Joystick Operation Modes. The value of 'n' in the instruction selects which of these is to be used.

n=1 Motor speed is directly proportional to the analog voltage. The function is linear.

n=2 Same as n=1 except there is a 12 percent dead-band. This means the analog voltage must be 12 percent or more of the full-scale value before the motor moves.

n=3 Same as n=2 except the dead-band is 25 percent of the full-scale value.

n=4 The same as n=1 except that the motor speed to analog input function is logarithmic. This means that a given change in voltage results in a small change in speed at low velocities, and a large change in speed at high velocities.

n=5 Same as n=4 except there is a 12 percent dead-band. A 12 percent of full-scale voltage is required to begin to move the motor.

n=6 The same as n=4 except the motor speed to voltage function is more strongly logarithmic.

n=7 Same as n=6 except there is a 12 percent dead-band.

n=0 The motor stops and ignores the ALG input. Sending a positioning instruction (other than a Joystick instruction) will also disable the ALG input.

The appropriate value for 'n' depends on the application. Dead- band is used to prevent unwanted motor movement by decreasing sensitivity in the center of joystick travel. The logarithmic functions are used where ease of accurate manual positioning as well as high speed slewing is required.

The full-scale voltage of the ALG input is selected by the internal Range Option Header. See the Control Connector section, Alg X, Y, in the HARDWARE MANUAL, CN0170 TWO AXIS MOTION CONTROLLER.

An analog input voltage above half-scale moves the motor in a clockwise direction; a voltage under half-scale moves the motor in a counter-clockwise direction.

The Acceleration Rate and Profile Parameter for the axis determines the rate of acceleration or deceleration in response to a speed change dictated by the ALG input. If the acceleration rate is set too high, the motor will tend to stall when the joystick is moved too rapidly. If the acceleration rate is set too low, the joystick will appear sluggish in its response to movement.

POINT-TO-POINT POSITIONING

A Point to Point instruction positions the motor to a specific location, as maintained by the internal Position Register. A Point to Point instruction may be Absolute or Relative.

When a Point-To-Point instruction is issued, the motor begins at Base Speed, and accelerates, using the Acceleration Rate and Profile, toward the Maximum Velocity. If the move is large enough to allow the motor to reach maximum velocity, acceleration will cease and the motor will move at constant velocity. Deceleration will mirror acceleration and the motor will reach base speed at the precise moment the targeted position is realized.

Point-To-Point moves for the X and Y-axis may be combined. The '&' denotes a combined X and Y move. The X move must always be the first axis specified in the instruction. The motors start together, using their

individual parameters for velocity and acceleration. One axis may complete its move before the other, based on these parameters and the lengths of the moves required. Instructions that follow for either axis will not begin until both motors have come to rest. Consider the following two program fragments:

```
X=2000 & Y=3000
X=3000
```

X and Y will begin to execute their first instructions simultaneously. The second X-axis move will not begin until both axis have completed the combined instruction. However...

```
X=2000
Y=3000
X-3000
```

X starts first and Y begins a short time later. This may or may not be significant, depending on the application. The second X-axis move will begin as soon as the first X-axis move finishes, with no regard for the Y-axis move.

POINT-TO-POINT ABSOLUTE X=n or Y=n

Absolute moves are specified by the "=" operator. The data is expressed as a position 'n' steps from the zero location of the Position Register. The number of steps the motor will move is the difference between the Position Register and the value of 'n'.

The direction of the move is clockwise if 'n' is greater than the motor position register. It is counter-clockwise if 'n' is smaller. The motor position register will be equal to 'n' after the Absolute point-to-point instruction executes. To position the X-axis 5000.5 full steps from the zero or Home position send:

```
X=5000.5
```

Assume the X motor position is 3,500.333. To move the X motor to 4,000, send:

```
X=4000
```

The motor will move clockwise 499.667 steps (4000-3500.333). If the instruction is sent again, the motor will not move because it is now located at 4000 (4000-4000=0).

To move the X and Y motors to 1500, send:

```
X=1500
Y=1500
```

The X motor will begin to move first. The Y motor will not move until its carriage return character is received. To start the motors simultaneously send:

X=1500 & Y=1500

The X and Y motors will stop independently, according to their individual parameters for velocity and acceleration.

POINT-TO-POINT RELATIVE X+n or Y+n
X-n or Y-n

The '+' or '-' operators indicate a relative move. In a Relative Move, the motor moves 'n' number of steps clockwise if the operator is plus and counter-clockwise if the operator is minus. To take one clockwise microstep on the X-axis using a 10 microstep drive; send:

X+.1

To move the X motor clockwise .875 steps and the Y motor counter-clockwise 123.45 steps, send:

X+.875
Y-123.45

After the move, the X Position Register will be greater by .875 steps and the Y Position Register will be 123.45 steps less than it was before the move.

Moves involving both motors may have mixed Relative and Absolute entries. For example to make a combined move of 3.5 steps clockwise for Y, and absolute position 2000 for X, send:

X=2000 & Y+3.5

Assume the X motor position is 1000. The X motor will move 1000 steps clockwise and the Y motor will move 3.5 step clockwise regardless of the value of the Y-axis Position Register before the move. The motors will start to move simultaneously, but the Y motor will be in position first since its move is much shorter than the X motor's move. However, the instruction is not complete until the X motor finishes.

In multiple motion controller applications the Unit Number must precede the instruction or string of instructions sent to that unit.

For example, assume three CN0170s are daisy-chained together and assigned Unit Numbers 0,1,2. For Unit 0, move the X motor 100 steps and the Y motor 200 steps. For Unit 1, move the X motor 300 steps and the Y motor 400 steps. For Unit 3, move the X motor 500 steps and the Y motor

600 steps. All the moves will be Point-To-Point Relative and in a clockwise direction. To do this, send:

.U0	*select Unit 0*
X+100;Y+200	*instructions for the unit*
.U1	*select Unit 1*
X+300;Y+400	*instructions for the unit*
.U2	*select Unit 2*
X+500;Y+600	*instructions for the unit*

Until a new Unit Number is issued, all commands, instructions and parameters go to the currently addressed unit; Unit 2 in the above example.

LINEAR INTERPOLATION	(ABSOLUTE)	X=n,Y=m
	(RELATIVE)	X+n,Y-m
	(MIXED)	X=n,Y+m

The ',' operator indicates a linear move The Linear Interpolation Instruction is similar to the combined Point-To-Point Instruction, with two major differences.

1. Linear Interpolation instructions start and stop both motors at the same time.

2. Linear Interpolation uses the Vector Velocity Parameter. The X and Y motor velocities are trigonometrically derived from the Vector Velocity. Linear Interpolation uses Vector Velocity instead of axis velocity.

For XY plane applications, this results in a linear path between the start and end co-ordinates, and a velocity independent of direction. Linear Interpolation finds its main use in applications where path accuracy and velocity is important, such as contouring and shape following applications.

In the following Linear Interpolation example, a triangle whose points have the following co-ordinates must be traced:

POINT	X	Y
A	2,000	10,000
B	3,000	5,000
C	7,000	8,000

Assuming X and Y are located at point A, send:

Error! No topic specified.

X=3000,Y=5000
X=7000,Y=8000
X=2000,Y=10000

The following relative moves would result in the same movement profile:

X+1000,Y-5000
X+4000,Y+3000
X-5000,Y+2000

The motors move from point A to B to C and back to point A. The speed at which the motors move between the points is constant. On approaching a point, the motors decelerate to the Vector Base Speed, then accelerate again to the Vector Maximum Velocity en route to the next point.

The advantage of using absolute data co-ordinates is the program can return to the triangle point co-ordinates from any initial location without calculating the offsets. The advantage of Relative Instructions is the program can trace an outline the same shape as the triangle at any initial location for point A without changing the data values.

When the CN0170 is executing a string of Linear Interpolation Instructions, the motors accelerate to the vector maximum velocity at the beginning of each line segment, and decelerate to the vector base speed approaching the end of each line segment. If these line segments form a path that must be followed, the velocity at which this path is traced will necessarily vary.

It is not possible to maintain a constant velocity and make abrupt changes of direction, such as a right-angle turn. Attempting to maintain constant velocity and accurate path tracking results in rates of acceleration beyond the capability of the motors.

If a constant velocity is maintained, then corners have to be rounded at a radius sufficiently large so that the motors' maximum rate of acceleration is not exceeded. If accurate path tracking is maintained, then it is necessary to decelerate to the base speed as the corner is reached.

The Linear Interpolation Instruction in the CN0170 opts for accurate path tracking as being of primary importance, so the Vector Velocity will vary.

If an application does require constant velocity, then the vector base speed should equal the vector maximum velocity. Then all moves will be made at base speed.

CIRCULAR INTERPOLATION	(ABSOLUTE)	X=n,Y=m ^ X=n,Y=m
	(RELATIVE)	X+n,Y-m ^ X-n,Y+m

(MIXED) X=n,Y+m ^ X+n,Y=m

The '^' operator indicate a circular move. The Circular Interpolation Instruction co-ordinates the X and Y axis so that their combined motion follows an arc or a circular path. The CN0170 utilizes the 'three point theorem' to define the arc. The three point theorem states that for any three given points on a plane, there is an arc that will fit through those points.

The first point for the arc is implied; it is the current position of the X-axis and the Y-axis. It does not have to be specified. The second point can be located anywhere on the arc so long as it is between the starting point and the end point. It need not be the precise midpoint between the start and end points. The third point of the arc (end point) marks the co-ordinates of the X-axis and the Y-axis at the conclusion of the instruction.

As with Linear or Point-To-Point, co-ordinates may be entered as absolute, relative or a combination of absolute and relative values. Note that when relative values are used for either co-ordinate the reference is always to the current location. Relative values for point three is in reference to point one, not point two.

Circular Interpolation, like Linear Interpolation, utilizes the Vector Velocity Parameters. These Vector Parameters set the tangential velocities in the same manner as their Linear Interpolation counterparts. The Vector Parameter units of measure are the same as Linear Interpolation except they are measured along an arc or circumference instead of a straight line.

The Linear Ramp is always selected as the profile for acceleration during Circular Interpolation. If another profile has been selected by the host, it will be ignored for Circular Interpolation, but will remain in effect for other moves.

In the following example, an arc with a radius of 1000 full steps is traced clockwise from 0° to 180°. Assume the X-axis is located 675 steps from the Home position and the Y-axis is located 2700 steps from the Home assume maximum vector velocity is 1000 full steps per second. To do this, send:

X+1000,Y-1000 ^ X+0,Y-2000

Error! No topic specified.

The CN0170 will generate an arc in a clockwise direction. The arc segment is π times 1000; 3141.59 full steps long. Ignoring acceleration/deceleration time, the move will take π seconds to complete. At the conclusion of the move, the X axis position will be 675 and the Y-axis will be 700 steps from the Home location.

FULL CIRCLES:

The three-point method for generating arc segments does not provide for complete circle movement. The Circular Interpolation Instruction defines two special cases for generating a full circle, in a clockwise or counter-clockwise direction. This is done by making two of the three co-ordinates concurrent.

A full clockwise circle is generated when the co-ordinate before the "^" is equal to the starting point. The co-ordinate after the "^" is located on the opposite side of the circle. The straight-line distance between these co-ordinates is the diameter of the circle.

A full counter-clockwise circle is generated when the co-ordinates on both sides of the "^" are equal. These concurrent points are located on the opposite side of the circle from the start point. The straight-line distance between them and the start point is equal to the diameter of the circle.

If both co-ordinates are equal to the starting point, or the co-ordinate after the "^" is equal to the starting point, the CN0170 will consider it a null move. The instruction will be received but no motion will take place.

Error! No topic specified.

As an example, to generate a clockwise circle with a diameter of 1000 full steps and a start angle of zero degrees, send:

X+0,Y+0 ^ X+0,Y-1000

The co-ordinate before the "^" is equal to the start point, indicating a clockwise circle will be generated. The start angle is perpendicular to the line tracing the diameter from the start point to the second co-ordinate.

Since the X-axis component of this line is zero and thus parallel to the Y-axis, the start angle is zero degrees.

Error! No topic specified.

As another example, generate a counter-clockwise circle with a diameter of 1000 full steps and a start angle of 45°. Send:

X+707.1,Y-707.1 ^ X+707.1,Y-707.1

The co-ordinates are equal, instructing the CN0170 to generate a full circle, counter-clockwise. Because this point is on the opposite side of the circle, it must be at the 135° location (45° + 90°).

The values for the co-ordinates are generated as follows:

$$X = 1000 \sin 135° = (1000)(.707) = 707.1$$
$$Y = 1000 \cos 135° = (1000)(-.707) = -707.1$$

CIRCULAR INTERPOLATION PERFORMANCE:

Circular Interpolation is the most complex instruction executed by the CN0170 and requires some special considerations for best performance. Maximum Velocity must be considered when executing Circular Interpolation Instructions. The CN0170 maintains a constant tangential velocity determined by the value of the Vector Maximum Velocity parameter. As progressively smaller radius arc segments or circles are generated, increasing rates (radians per second) are required. While the arc or circle is being generated, the motor undergoes acceleration and deceleration equal to the first derivative of its velocity, which is sinusoidal with time. For a given velocity the torque demands due to acceleration is inversely proportional to the required radius. Ultimately the motor will stall if too small of a radius is required at too high of a speed.

The Accelerate Rate Parameter sets the tangential rate of acceleration around the generated arc segment or circle in steps per second squared. Its purpose is to accelerate the motor to, or decelerate it from the tangential velocity set by the Vector Maximum Velocity parameter. As such, it cannot decrease the high rates of acceleration and deceleration required by high speeds around small radii. This potential problem is best dealt with by the host. The value of Vector Maximum Velocity may be temporarily adjusted downward if a radius less than a certain size is required.

Because of the nature of the calculations required by the CN0170 while executing Circular Interpolation Instructions, the minimum radius is 1/128 of a full step while the maximum radius is 131,072 steps. Larger values of radius default to a value of infinity, and the resulting arc is executed as if it were a Linear Interpolation Instruction. The move is taken in a straight-line path from the present position to the location of the third point.

The CN0170 generates arcs by repeatedly calculating the X and Y co-ordinates of where the axis should be on the arc per unit of time. It then finds the difference between the new location and the current location. This difference is the distance the axis moves during a unit of time.

The calculations are repeated 128 times a second, consequently the CN0170 generates 128 straight-line segments connecting evenly spaced points on the arc. The result is that the arc or circle is best considered as being an 'n' sided polygon, where 'n' is the number of line segments or facets.

The number of facets generated is fixed at 128 per second, so the number of facets per circle will depend entirely on the radius of the circle and on the speed at which the circle is generated.

The speed at which the circle is generated is set by the Vector Maximum Velocity Parameter and to a lesser extent by the X-axis Accelerate Rate Parameter.

The time required to generate an arc of a given radius is inversely proportional to maximum velocity therefore the number of facets along the arc is also inversely proportional to maximum velocity.

The circumference or arc length is proportional to the radius, therefore the number of facets on the arc is proportional to the radius; for a given maximum velocity. For instance, a circle with a circumference of 1000 full steps, (a radius of 159.155 steps) is generated in one second if the maximum velocity is 1000 steps per second. The circle is approximated as a 128-sided polygon since the CN0170 generates 128 line segments per second.

At the other extreme, a circle with a radius of 100,000 full steps and a maximum velocity of 100 full steps per second is generated in 6283.18 seconds and has 804,248 facets or sides to it. (128*2*100,000/VMAX)

MOTOR PARAMETER INSTRUCTIONS

Operating parameters are utilized by all motor movement instructions. They are set before movement instructions are issued. Motor parameter instructions do not directly result in movement of the motors; they define the performance envelope for motion instructions sent later.

SET AXIS VELOCITY	b=base, m=max.	XV=b,m or YV=b,m
	b=0 to 16,383.75	XV=b or YV=b
	m=.25 to 16,383.75	XV=,m or YV=,m

To set the axis velocity parameters send the base velocity first, followed by the maximum velocity. The base speed or velocity is the step rate from which acceleration begins. The maximum speed parameter defines the maximum step rate permitted for that motor.

The range of base velocity is 0 to 16,383.75 steps per second. The resolution of the base velocity register is 1/4 step per second.

The maximum velocity is determined by the physical requirements of the system utilizing the motor, or is limited by the high-speed performance of the motor itself. The range is from .25 to 16,383.75 steps per second. As with base speed, the speed resolution is in increments of .25 steps per second. A maximum velocity of zero is not allowed.

Often it is not necessary to begin acceleration from zero. Typically, 200 to 500 steps per second is a realistic base velocity for light loads. The maximum Base Speed is equal to the motor's error free start stop speed while driving the application's load. Large inertial loads generally require a lower base speed while low inertial motor loads and small motors can start at a higher base speed.

To set a base speed of 450 steps per second and a maximum speed of 15,000 steps per second on the X motor, send:

XV=450,15000

Subsequent X-axis movement instructions will use these parameters for velocity.

The host may send the base or maximum velocity parameters individually. To set a base speed of 500 steps per second for the X motor, send:

XV=500

This will set the X-axis Base Velocity. The Maximum Velocity parameter will not be affected by this instruction. To set the Maximum Velocity to 3333.33 steps per second, and leave the current Base Velocity unchanged, send:

XV=,3333.33

The CN0170 would round off the value to the closest resolvable speed, which in this case would be 3333.25 steps per second. Base Velocity will remain as before the instruction.

The controller will also accept velocity parameters in hexadecimal format. This gives the host computer direct access to the internal velocity registers of the CN0170. All velocity registers are 2 byte registers with 1 bit equal to .25 steps per second.

SET VECTOR VELOCITY	b=base, m=max.	XYV=b,m
	b=0 to 16,383.75	XYV=b
	m=.25 to 16,383.75	XYV=,m

The Vector Velocity Parameters apply when both motors are moved by linear or circular interpolation instructions. The Vector Velocity Parameters trigonometrically adjust the X and Y motor step rates so that the vector sum of the two step rates is independent of the angle of motion. In applications where an X/Y table is used, this results in a table velocity independent of the angle of table motion.

The Vector Velocity Parameters are closely modeled after the X and Y velocity parameters. The same range and resolution limits apply.

The values for Vector Velocities are determined by the weaker of the two motors. Motion along the direction of the weaker motor's axis requires it to bear the full value of vector velocity. This effectively places the weak motor's limits on the value of the Vector Velocity parameters.

To set a vector base speed of 250 steps per second and a maximum vector speed of 1750 step per second, send:

XYV=250,1750

The Vector Velocity Parameters are applicable only during execution of linear or circular interpolation instructions. The individual X and Y-axis velocity parameters apply for all other motion instructions.

QUERY VELOCITY XV? or YV? or XYV?

The host computer can interrogate the CN0170 at any time for the current velocity of a motor. The response will be the motor's step rate at the moment the request was received. The answer is always given in hexadecimal format. During Circular or Linear Interpolation the vector velocity may be requested. The response will be the vector sum of the X and Y motors' step rates. To query the Y motor's step rate, send the following:

YV?

The CN0170 may respond with:

YV=0D485h

This would indicate the motor's step rate at the time the Query Velocity was received was 13,601.25 steps per second. If the motor were accelerating or decelerating, the response would have been accurate only for the moment the query was received. By the time the response was prepared and transmitted to the host computer, the motor's velocity could have changed substantially. The answer would remain accurate if the motor was not accelerating or decelerating.

VELOCITY TRIGGER XV+ or YV+
XV- or YV-

The Velocity Trigger notifies the host computer when a motor has achieved maximum velocity. The Velocity Trigger is enabled by a plus sign '+' in the parameter, while a minus sign '-' disables it. To enable the Velocity Trigger for the X-axis, send:

XV+

When the motor reaches maximum velocity, the CN0170 will respond:

XV

SET ACCELERATE RATE n=64 to 4,194,304 XA=n or YA=n

This instruction sets the rate of acceleration and deceleration for a motor when it has to change speed. The rate of acceleration is expressed in steps per second squared, and ranges from a minimum of 64 to a maximum of 4,194,304 steps per second squared. The resolution is 64 steps per second squared. A zero rate of acceleration is not allowed. The host may provide the acceleration rate in either decimal or hexadecimal notation. The acceleration registers are 2 bytes in length.

Like velocity, the choice of a value for the Accelerate Rate is determined by system and application considerations. The rate must be high enough so as not to impede the operation of the system, yet not be so high that the motor is unable to supply the transient torque that acceleration requires.

If the X motor is to be linearly accelerated to 2000 steps per second from a base speed of 500 steps per second in 0.5 seconds, the Accelerate parameter is 2000-500/0.5 or 3000 steps per second squared. To set the Accelerate Rate parameter to 3000, send:

XA=3000

Or the same acceleration rate sent in hex format (3000/64 = 47 = 2FH):

XA=2FH

Because the Acceleration Resolution is 64 steps per second squared, the CN0170 rounds off to the closest resolvable value of 3008 steps per second squared.

QUERY ACCELERATE RATE XP? or YP?

The host computer can request the contents of the Accelerate Rate Register at any time. The answer is always given in hexadecimal format. To request the acceleration rate of the X motor, send:

XA?

The controller's response might be:

XA=00BBh

This would indicate that the accelerate register has a value equal to 11,968 full steps/sec^2.

SELECT STORED PROFILE n=1 to 16 XCn or YCn

Step motor torque is not constant with speed. If it were, the ideal rate of acceleration would be a constant and motor speed would be proportional to time. The result would be the Linear Ramp Acceleration Profile.

The torque required to overcome the moment of inertia is proportional to the rate of acceleration. Since motor torque tends to decrease with speed, a constant rate of acceleration does not yield the shortest acceleration time.

Because the rate of acceleration is determined by the motor's available torque at the speed being accelerated to, considerable low speed torque goes unused. This situation can be improved upon by making the rate of acceleration a function of motor speed, and therefore proportional to the motor torque available at that speed.

The method the CN0170 employs to achieve this is to store 16 composite motor speed/torque curves in its firmware. These speed versus torque functions have been generated using Centent Step Motor Drives running representative motors on a computerized dynamometer test stand. The curves are stored in look-up table form and are normalized as a percentage of holding torque versus speed.

The Acceleration Curve Table values act as multipliers of the Accelerate Rate parameter. Because the table values range between 0 and 1, the product is equal to or less than the Acceleration Parameter. This product is the actual rate of acceleration.

The 16 stored Acceleration Curve Tables form a family of curves in ascending order of performance. Performance in this context is defined as

meaning the rate at which motor torque decreases with speed, and the speed at which this torque decrease begins.

The lowest performance curve is suitable for high inductance motors operated from a low voltage power supply. The highest performance curve is suitable for low inductance motors operated from a high voltage power supply. Other motor inductance and power supply voltage combinations are satisfied by one of the middle performance curves.

The Acceleration Curve number that best matches the motor, motor driver and power supply combination may have to be determined empirically. This is particularly true if the user does not have the speed vs. torque data for his particular system.

SELECT LINEAR RAMP XC\L or YC\L

Selecting Linear Ramp will result in constant rate of acceleration. The rate of acceleration will be equal to the Acceleration Parameter, and will be independent of motor speed.

SELECT PARABOLIC CURVE XC\P or YC\P

Selecting Parabolic Profile will result in a parabolic rate of acceleration.

SELECT USER PROFILE XC\U or YC\U

The User Profile permits the user to select his own acceleration rate versus speed parameters. The user-entered parameters are utilized in exactly the same fashion as the other profiles. When programming the User Acceleration Curve, the user must supply the curve table values.

The acceleration curve look-up table requires 16 values to be entered in hexadecimal format. These values are used to multiply the Acceleration Parameter and must range between 00 and 0FFh.

The first value sent is utilized from 0 to 1024 steps per second, the second value sent is utilized from 1024 to 2048 steps per second and so on. The last value sent is utilized from 15,360 to 16,384 steps per second, which is the maximum speed.

As the motor accelerates, the four most significant velocity bits form a pointer into the User Acceleration Curve Table. The pointed to value is used as the multiplier for the Accelerate Rate Parameter, forming the current rate of acceleration. The next value is selected in turn as the velocity increases. During deceleration, this process is undone and a mirror image deceleration curve is generated.

ENTER USER PROFILE	i is 1 to16 n=00 to 0FFh	XC(i)=n YC(i)=n

This instruction allows the host to enter values for the user provided acceleration profile. All 16 values should be sent before selecting the User Profile for acceleration.

Assuming a User Acceleration Curve that is constant until 5100 steps per second, then decreases linearly to zero at 16,384 steps per second. To enter this curve for the X motor, send:

XC(1)=0FFh	*(99%)*
XC(2)=0FFh	*(99%)*
XC(3)=0FFh	*(99%)*
XC(4)=0FFh	*(99%)*
XC(5)=0FFh	*(99%)*
XC(6)=0FFh	*(99%)*
XC(7)=0FFh	*(99%)*
XC(8)=0E6h	*(90%)*
XC(9)=0CDh	*(80%)*
XC(10)=0B3h	*(70%)*
XC(11)=9Ah	*(60%)*
XC(12)=80h	*(50%)*
XC(13)=66h	*(40%)*
XC(14)=4Dh	*(30%)*
XC(15)=33h	*(20%)*
XC(16)=1Ah	*(10%)*

If the Accelerate Parameter for the X motor had been 640 steps per second squared, the rate of acceleration from 0 to 7168 steps per second would have been 640 * .99 or 640 steps/sec^2. Past 7168 steps per second, the rate of acceleration would have decreased to 640 * .90 or 576 steps/sec^2 until 8192 steps per second (8 * 1024 = 8192). Every additional 1024 steps per second gain in speed would reduce the rate of acceleration 10 percent until at maximum speed the rate of acceleration would be 640 * .1 or 64 steps/sec/sec.

QUERY ACCELERATE CURVE	XC? or YC?

The host computer can ask which acceleration profile is active for a motor at any time. The controller will respond with the curve number of the stored profile, or a letter to indicate linear, parabolic or user defined acceleration profile.

To request the acceleration profile of the X motor, send:

XC?

The controller's response might be:

	XC=3	stored curve #3 is active
or...	XC=L	linear ramp is active
or...	XC=P	parabolic profile is active
or...	XC=U	user profile is active

SET POSITION XP=n or YP=n

The Home Instruction automatically resets the internal Position Register to zero. The host may force the Position Register to an arbitrary value. To set the X-axis Position Register to 2,000,000.333 steps, send:

XP=2000000.333

The CN0170 will round the value off to the closest 1/1024 of a full-step, which would be 2,000,000 and 345/1024 full-steps.

If it is necessary to position the Home switch somewhere other than the end of travel of the axis, this instruction will justify the Position Register to the physical location.

The CN0170 totals motor steps in a 4 byte internal position register. This position register is constantly updated regardless of what mode or instruction is being executed. Position is maintained to a precision of 1/1024 of a full-step, with a full-scale range of 4,194,304 full-steps.

A clockwise motor direction adds steps to the position register while counter-clockwise moves subtracts steps from the register. The position register accepts only positive signed numbers. Any attempt to total below zero results in register underflow.

The Set Position Register Instruction may also be given in hexadecimal format. Position registers are 4 byte registers with 1 bit equal to 1/1024 of a full step.

QUERY POSITION XP? or YP?

The host computer can request the current position of a motor at any time. If the motor is at rest, the answering response will accurately reflect the motor position and indicate the motor is stopped. The answer is always given in hexadecimal format. To request the position of the X motor, send:

XP?

If the motor is stopped, the answering response might be:

X=00C0E6B6h

This would indicate that the position register has a value equal to 12345.678 full steps. The equal sign '=' indicates the motor is at rest, consequently the position is accurate. If the motor is moving, the answer will reflect the position and the direction of the motor at the time the request was received.

X+00C0E6B6h

The plus sign '+' indicates the motor was moving in a clockwise direction at the time the Query Position was received by the CN0170. A minus sign '-' indicates a counter-clockwise motor direction at the time of the position request. If a motor is moving at the time a Query Position is issued, the position indicated in the answering response will be accurate for the moment the query was received by the controller. By the time the response is generated and sent to the host computer, the motor may have moved a substantial number of additional steps.

The Query Position instruction is particularly useful in interactive applications using the Joystick Instruction. The user would use the Joystick to empirically arrive at XY locations. The Query Position instruction would be used by the host to quantify and log those positions in its memory.

POSITION TRIGGER

XP+ or YP+
XP- or YP-

The Position Trigger notifies the host when a move has completed. The Position Trigger is enabled by a plus sign '+' in the instruction. The CN0170 will generate a Position Trigger message at the conclusion of each move, as long as the trigger is enabled. To enable the Position Trigger for the X-axis send:

XP+

There will be no immediate response from the CN0170. When the next movement instruction for the X-axis concludes the response will be:

XP

The Position Trigger is disabled by the sending the instruction with '-' instead of '+'. To disable the Position Trigger for the Y-axis send:

YP-

The CN0170 will not generate a Position Trigger for the Y-axis moves after it receives this instruction.

RATIO OPERATION

Ratio Operation sets up one axis to follow the motion of the other. The follower axis will run at a predefined fraction of the control axis velocity. The follower axis may be programmed to move in the same or opposite direction as the controller axis. Either X or Y-axis may serve as the controller axis.

The Enter Ratio instruction provides the fractional rate of velocity for the follower axis. This is expressed as a decimal fraction. The Enable Ratio instruction specifies the direction and enables Ratio Operation.

A motion instruction for the control axis automatically creates a related motion in the follower axis. This includes Speed Control, Joystick or Home instructions. During Speed Control or Joystick operation the ratio may be changed on the fly. For other motion operation, new ratio instructions will not take effect until the completion of the move.

The follower axis will use its own parameters for acceleration rate and profile when tracking the control axis. To achieve an exact ratio of position displacement between the follower and the control axis, it is necessary to set their acceleration rates and profiles to the same values.

ENABLE RATIO Y/X+ or X/Y+
Y/X- or X/Y-

The follower axis is the character preceding the "/". The control axis is the character following the "/". The "+" or "-" character indicates whether the follower's direction of rotation is the same as, or opposite to that of the control axis. For example, to enable the Y-axis to mirror the X-axis, always moving in the opposite direction, send:

Y/X-

The ratio (always less than 1) of the velocity of the follower axis to that of the control axis is defined by the Enter Ratio Scalar Instruction. Any motor motion instruction sent to the follower axis will terminate Ratio Operation.

ENTER RATIO Y/X=.n or X/Y=.n

This instruction provides the scalar value for tracking in the Enable Ratio Instruction. The follower axis is the character preceding the "/". The control axis is the character following the "/". The fractional value ".n" is the speed ratio of follower to control axis. This ratio is always less than one. For example, to specify that the X-axis is to follow the Y axis at 50% velocity, send:

X/Y=.5

This does not cause the X-axis to actually enter the Ratio Mode. To do this you must Enable Ratio for the X-axis. The Enter Ratio value need not be

re-entered every time ratio operation is enabled. The Ratio may also be entered in hexadecimal format, as a 4 byte value from zero to 0FFFFFFFF hexadecimal. For example, to specify in hex for the Y-axis to follow the X-axis at 75% velocity, send:

Y/X=0C0000000H

The "H" character must be sent in upper case.

QUERY RATIO X/Y? or Y/X?

The host may query the value of the Ratio at any time. The reply from the controller is always returned in hexadecimal notation. The internal ratio registers of the CN0170 are 32 bit (4 byte) values.

If for example, if the host asks…

Y/X?

the reply might be:

Y/X=80000000h

In the above example, the Y-axis will run at 50% of the X-axis velocity. The answer does not indicate whether Ratio Operation is enabled or not.

SET HOME DIRECTION XH+ or YH+
XH- or YH-

The Home Direction Parameter defines the direction in which the home position switch is located. The Home Instruction uses this parameter to determine how to begin the process of homing the motor. A minus sign '-' indicates a home switch at the '0' end of axis travel. To set the X-axis for homing at the zero end of the travel send:

XH-

The Home Instruction will begin operation with a counter-clockwise rotation of the motor shaft.

The '+' variable indicates home switch located at the maximum position of axis travel. The Home Instruction will begin with a clockwise motor rotation if the '+' variable is used.

SELECT STEP RESOLUTION n= 1, 2, 10, or 125 XRn or YRn

The Home algorithm positions the motor at the home location within one increment of motion. This means within one full-step if 'n' equals 1, to within 1/125 of a full-step if 'n' equals 125.

During the final phase of the Home instruction, the HME input is sampled 256 times a second for a valid home location while the motor moves at a rate equal to one increment of motion per sample period.

This step rate ranges from 256 full-steps per second for 'n'=1 to 2 full-steps per second for 'n'=125. The result is a considerable saving in execution time if the value of 'n' matches the motor drive step resolution. To set the X-axis resolution for a 10 microstep driver, send:

XR10

For a more complete description of the Home algorithm, see HME X, Y in the HARDWARE MANUAL, CN0170 TWO AXIS MOTION CONTROLLER.

I/O LINK

XL+ or YL+
XL- or YL-

Either or both motors may be linked to the input/output channels by using the "XL+" or "YL+" instruction. When this parameter is active, input/output activity occurs only after motor instructions have completed. "XL-" or "YL-" instructions free the I/O channels of dependence on the state of the motor(s).

Consider the following two program fragments...

```
XL-          (free I/O from X motor)
X+1000       (move X motor 1000 full steps CW)
P3=1         (turn output on)

XL+          (link I/O to X motor)
X+1000       (move X motor 1000 full steps CW)
P3=1         (turn output on)
```

The behavior of the controller will be very different in these two examples. In the first, P3 will turn on virtually simultaneously with the starting of the X motor motion. In the second example however, the motor will complete the 1000 step move before setting the output channel. Normally, it is desirable to set the "XL+","YL+" parameters when using the Operate Mode. Immediate Mode may require the "XL-","YL-" parameters for direct control of the I/O channels by the host.

SET HANDSHAKE

XB+ or YB+
XB- or YB-

The Handshake Parameter determines the action of the STB output after a motor movement instruction has completed. To turn hardware handshake on for both axis send:

XB+; YB+

With hardware handshake turned on, the STB output remains active until the BSY input is taken active. The STB output acts as a means of initiating an external process when the axis has finished a move. The axis may not begin the next move until the process is finished.

The BSY input going active informs the CN0170 that the external process has commenced. The STB output goes inactive when the BSY input goes active. When the BSY input goes inactive, the CN0170 will move the axis to the next position, if an instruction is pending.

To turn handshake off for both axis send:

XB-; YB-

With handshake turned off, the STB output goes active for 1/256 of a second at the conclusion of a move. If the BSY input does not go active within that time, the next move, if one is pending, is free to commence. Since the BSY input is not connected to anything when handshake is turned off, operation will proceed without interruption.

SET STANDBY DELAY n= .004 to 256 XI=n or YI=n

The CN0170 provides the ISET output for each motor that functions as a Standby Current Output. The purpose of this instruction is to reduce power dissipation while that motor is idling. Idling is defined as no motor steps being issued for 'n' seconds to the motor by the CN0170.

The Standby Current Parameter acts as a timer. If steps are sent to the motor before it times out, the ISET output does not activate and motor current remains at its nominal value. If activated, the ISET output goes inactive 4 milliseconds before any new steps are sent to the motor.

A zero value for 'n' disables the Standby Current feature. The motor current remains at its nominal value regardless of how long it has been stopped.

To program the X motor to go to standby after 5 seconds idling and the Y motor to maintain full current during idle, send:

XI=5
YI=0

Alternately, the user could wire the 'Y' motor drive so that its idle current input was inactive.

INPUT/OUTPUT INSTRUCTIONS

The CN0170 has two general-purpose digital output channels, two general-purpose digital input channels, a dedicated analog channel for each axis and two general-purpose analog input channels. All of these are available on the Control Connector. An eight bit Auxiliary Input/Output Port is also available on an internal header. See HARDWARE MANUAL, CN0170 TWO-AXIS MOTION CONTROLLER, AUXILIARY INPUT/OUTPUT CONNECTOR for details.

The following table shows the relationship between the software instructions and the hardware connections.

Software reference	*Hardware reference*	*Control Conn.*	*function*
P1	IN 1	14	digital input, general purpose
P2	IN 2	1	digital input, general purpose
P3	OUT 1	8	digital output, general purpose
P4	OUT 2	20	digital output, general purpose
A\X	ALG X	10	analog input, X-axis
A\Y	ALG Y	9	analog input, Y-axis
A1	ALG 1	23	analog input, general purpose
A2	ALG 2	22	analog input, general purpose
P			digital input/output, auxiliary

SET OUTPUT n=3 (OUT1) or n=4 (OUT2) Pn=0 or Pn=1

The general purpose output instructions may be sent directly (M1) or loaded into the Program Queue (M2). An output turned on will stay on until it is turned off. On power-up both outputs are turned off. To turn OUT1 on, send:

P3=1

The OUT1 will go active (low) within four milliseconds of the receipt of this instruction. To turn OUT2 off, send:

P4=0

QUERY DIGITAL I/O n=1 (IN1) or n=2 (IN2 Pn?
n=3 (OUT1) or n=4 (OUT2)

To query the state of IN 1 send:

P1?

The response from the motion controller will be:

P1=0 *indicating P1 is 'off'*
or
P1=1 *indicating P1 is 'on'*

P1 DEFINE P1\G
P1\I

In addition to functioning as a general purpose input P1 may be used to start program operation in M3. This allows the CN0170 to operate as a stand-alone controller with the host removed. To enable this feature send:

P1\G

This is equivalent to sending the "G" command over the serial interface. If the program contains Gn or Halt Commands, the P1 input may be used to restart the program. If the input is held active, the program will restart automatically.

To lock out the P1 input for program control redefine it as a general purpose input by sending:

P1\I

Define P1 commands may be embedded in the program to enable or disable the input for program control. Be advised however that if the input is not defined as a Go the program will not begin without a command over the serial interface.

QUERY JOYSTICK INPUTS A\X? or A\Y?

The analog voltage on the JOY X, Y inputs may be queried by the host. The response is returned as two ASCII characters representing the hexadecimal value of the analog voltage, followed by the 'h' character, designating hex format.

If the analog input voltage range is 0 to 5VDC, then '00' will represent 0VDC and 'FF' will represent 5VDC. If the analog input voltage range is 2 to 3VDC, then '00' will represent 2VDC and 'FF' will represent 3VDC.

If there is 3.75 VDC on the ALG X input and the input voltage range is set to 0 to 5VDC, sending:

A\X?

will result in the following response:

A\X=C0h

This is because 256 * 3.75 / 5 = 192 decimal or C0 hexadecimal. If the input voltage range had been set to 2VDC to 3VDC instead, the response would have been "A\X= FFh". This is because 3.75VDC is greater than the upper limit of the input voltage range of 3VDC. Any voltage above the upper limit of the input voltage range results in a response of 'FFh'. Any voltage below the lower limit of the input voltage range results in a response of '00h'.

QUERY ANALOG INPUTS n=1 (ANA1) or n=2 (ANA2) An?

The analog inputs ALG1 and ALG 2 may be queried by the host computer. The response to the query will be the same as it is for the joystick analog inputs described in the previous section.
Sending the query:

A1?

The response might be:

A1= 33h

AUXILIARY INPUT/OUTPUT PORT

The Auxiliary Port provides eight additional inputs or outputs. The Auxiliary Port is accessed by an internal header; it is not routed to the Control Connector. See the HARDWARE MANUAL, CN0170 TWO-AXIS MOTION CONTROLLER for header location and pin assignments.

The Auxiliary Port is actually a direct connection to a Z80 PIO, and may be programmed for Input, Output, or Bit Control. For Bit Control, each bit is individually designated as Input or Output.

For Input or Output modes, the Strobe input and Ready output control lines are active. Strobe and Ready are not used for Bit Control. See a Z80 PIO Technical Manual for details on how the Strobe and Ready lines function.

The following instructions provide access to the Auxiliary Input/Output Port.

SET AUXILIARY PORT TO INPUT MODE P\I

To program the Auxiliary Port for Input send:

P\I

The Query Auxiliary Port instruction gives the status of the port to the host. If, for example, the reply is "P= 55h", bits 7, 5, 3, 1 are low and bits 6, 4, 2, 0 are high at the time the request was received by the CN0170.

SET AUXILIARY PORT TO OUTPUT MODE P\O

To set the Auxiliary Port for Output send:

P\O

The Auxiliary Output Direct, And, Or and Xor instructions determine the state of the Auxiliary Port in Output mode.

SET AUXILIARY PORT TO BIT CONTROL MODE P\hhH

This instruction is used to set the eight bits (0-7) to any desired combination of inputs and outputs. To determine the value for the 'hh' parameter, first form an 8 bit binary number with '0' representing an output and '1' representing an input. Convert this to hexadecimal, add a zero to the front if the first character is 'A' through 'F' and add an 'H' to the

end. All characters must be sent as upper case. For example, to specify the odd bits as outputs and the even bits as inputs, send:

P\0AAH

This is arrived at by converting 01010101(binary) to 0AA (hexadecimal) and adding the "H" for hexadecimal. For another example, to set channel 7 to an input and the rest as outputs send:

P\80H {10000000 binary}

The Auxiliary Output Direct, And, Or and Xor instructions determine the state of the Auxiliary Port outputs in Bit Control Mode. The Query Auxiliary Port instruction reports the status of the Auxiliary Port inputs to the host.

QUERY AUXILIARY PORT P?

This instruction reports the status of Auxiliary Port. To request the status of the Auxiliary Input/Output Port send:

P?

The answer is returned in hexadecimal format. The CN0170's reply reflects the state of both Input and Output bits. If, for example, the answer is:

P= 3Ch

Bits 2-5 are high and the rest are low. The 'h' is returned in lower case.

AUXILIARY OUTPUT DIRECT P=nnH

This instruction is used to directly turn Auxiliary Output bits on and off. For example, to turn all output bits on, send:

P=0FFH

If the Auxiliary I/O Port is set to Input Mode, the Auxiliary Output Direct instruction will have no effect. If the Mode is Bit Control, only the Output bits will be affected.

AUXILIARY OUTPUT AND P&nnH

This instruction performs a logical And on the current state of the Auxiliary Port and the value given in 'nn'. The new value is output to the Auxiliary Port. For example, to turn Bit 5 off and leave the other Outputs unchanged, send:

P&0DFH

If the Auxiliary I/O Port is set to Input Mode, the Auxiliary Output And instruction will have no effect. If the Mode is Bit Control, only the Output bits will be affected.

AUXILIARY OUTPUT OR P|nnH

This instruction performs a logical Or on the current state of the Auxiliary Port and the value given in 'nn'. The new value is output to the Auxiliary Port. For example, to turn Bit 2 on and leave the other Outputs unchanged, send:

P|04H

If the Auxiliary I/O Port is set to Input Mode, the Auxiliary Output Or instruction will have no effect. If the Mode is Bit Control, only the Output bits will be affected.

AUXILIARY OUTPUT XOR P||nnH

This instruction performs a logical Xor on the current state of the Auxiliary Port and the value given in 'nn'. The new value is output to the Auxiliary Port.

For example, to invert the state of all Output bits send:

P||0FFH

If the Auxiliary I/O Port is set to Input Mode, an Auxiliary Output Xor instruction will have no effect. If the Mode is Bit Control, only the Output bits will be affected.

GETTING STARTED

The following program sample initializes the CN0170 to operate two motors at moderate performance levels. The instructions will be placed in the Program Queue for operation in M3 (Operate Mode). Apply power to the CN0170 for a power-on reset. Wait a second or two, and then send the following lines from the host:

<CR>	sets the baud rate to match the host
M1\+	select RS232 hardware handshake
M1	select Immediate Mode
P1\G	define Input 1 as a Go Command
M2	enter M2 from M1 to clear Program Queue
XL+; YL+	link I/O instructions to motor moves
XB-; YB-	disable STB interlocks
XP-; YP-	disable Position Triggers
XV-; YV-	disable Velocity Triggers
XM-; YM-	select normal Quit for both axis
XH-; YH-	select counter-clockwise Home direction
XR10; YR10	set motor resolution to 10 microsteps
XI=1; YI=1	enable Standby Current after 1 second
XV=200,2000	set X-axis Base and Maximum Velocity
YV=200,2000	set Y-axis Base and Maximum Velocity
XYV=200,2000	set Vector Base and Maximum Velocity
XA=10000	set the X-axis Acceleration Rate
YA=10000	set the Y-axis Acceleration Rate
XC\L; YC\L	se acceleration profiles to linear

The above represents the initial parameter setting section of the program. Motor positioning instructions may now be sent. The first instruction for each motor is likely to be a Home instruction. This moves the motors to their home reference location and resets the Position Registers.

XYH	move both motors to the Home Location.

The program may now proceed with positioning, Input/Output and parameter instructions as required. The final Command will take the CN0170 out of Program Mode and place it in the Operate Mode.

M3	select Operate Mode; wait for a Go Command

Operation may be controlled by the host over the serial interface. See the sections for Program operation; Go, Halt, Quit or Kill. If the host is removed from the system, Input 1 may be used as a hardware switch activated Go instruction.