5.2 COMMAND SET SUMMARY

All programming of CMC systems is accomplished through the CMC-0 master unit. The command set can be broken down into the following categories:

1. System commands
2. Axis Initialization commands
3. Movement and Profile Definition commands
4. Conditional Profile Execution and Branching commands
5. Auxiliary Position Equation commands
6. Phase-Locked Loop commands
7. I/O and Variable commands
8. Status and Position Query commands
9. Editing commands

A brief description of all command types is given below. In the descriptions, the following notation applies:

a	- denotes axis W, X, Y, or Z
b	- denotes a binary digit, 0 or 1
d	- denotes a hexadecimal digit, 0-F
V	- denotes an integer variable
n	- denotes an 8-bit number
nn	- denotes a 16-bit number
nnn	- denotes a 24-bit number
nnnn	- denotes a 32-bit number

In addition, brackets indicate that the programmer must choose one of the items within the brackets. Parentheses indicate optional items.

System commands: (Not allowed in defined profiles.)

Numbers in parentheses after the command description indicate sections where more detailed information is available about the command.

Mnemonic	Action	
CB	Clear all profiles, reset parameters to default values	
WB	Clear faults, leave profiles and parameters intact	(4.2.6)
PEb	Enable/disable parallel profile execution	(4.6)
EB	Empty command buffer	
MS	Upload memory from one CMC-0 to another	(4.8)
ML	Download memory to CMC-0 from another	(4.8)
DGb	Enable/disable diagnostic routine	(4.5)
?	Returns parameter value	(4.4.1)

Axis initialization commands: (Not allowed in defined profiles.)

These commands initialize the individual axes with user-specified control constants. Each axis used must be initialized at least once prior to executing moves. These commands are described completely in Sections 4.2.1 and 4.2.2.

(1) - used with CMC-1 Stepping Motor Controllers only.
(2) - used with CMC-2 Servo Motor Controllers only.

Mnemonic		Action	
ITa		Initialize axis with current control constants	(4.2.3)
URann		Load unit ratio	
NOannn		Load negative software O.T.	
POannn		Load positive software O.T.	
WRab		Enable/disable position register wraparound	
LDann	(2)	Load encoder line density	
GNan	(2)	Load controller gain	
ZRan	(2)	Load controller zero	
PLan	(2)	Load controller pole	
FFan	(2)	Load feedforward constant	
DBann		Load deadband constant	
IBann	(2)	Load integration band	
PBann	(2)	Load position band	
ERann	(1)	Load encoder ratio	
DFab	(1)	Enable/disable fault on deadband error	
PCab	(1)	Enable/disable position correction	
PSab	(1)	Enable/disable power-save feature	
CTan	(1)	Load correction time constant	

Movement and profile definition commands:

These commands are used to load motion parameters and execute moves. In general, they can be executed interactively or stored in a profile for later execution. Numbers in parentheses after the command description indicate sections where more detailed information is available about the command.

Mnemonic	Action	
DEn	Begin profile definition	
ED	End profile definition	
UEb	Enable/disable profile 128 execution	(5.11.1)
RMnn	Repeat statements from start of profile (n times)	
GTn	Unconditional GOTO profile n	(5.15)
GSn	Unconditional GOSUB profile n	(5.15)
HT	Halt all axes	
STZa	Immediately set at zero	(5.5)
SHZa	Find home limit switch, then set at zero	(5.5)
SIZa	Find index, then set at zero	(5.5)
SBZa	Find both, then set at zero	(5.5)
STPannn	Immediately set at position nnn	(5.5)
SHPannn	Find home limit switch, then set at position nnn	(5.5)
SIPannn	Find index, then set at position nnn	(5.5)
SBPannn	Find both, then set at position nnn	(5.5)

Mnemonic	Action	
AMa$\begin{bmatrix}\text{nnn}\\\text{Vd}\end{bmatrix}$	Load absolute move register	(5.3)
IMa$\begin{bmatrix}\text{nn}\\\text{Vd}\end{bmatrix}$	Load incremental move register	(5.3)
ACa$\begin{bmatrix}\text{nn}\\\text{Vd}\end{bmatrix}$	Load accel/decel rate	(5.3)
VX$\begin{bmatrix}\text{nn}\\\text{Vd}\end{bmatrix}$	Load trajectory velocity for XY move	(5.3)
VW$\begin{bmatrix}\text{nn}\\\text{Vd}\end{bmatrix}$	Load trajectory velocity for WZ move	(5.3)
VOXnn	Load velocity offset for XY move	(5.9)
VOWnn	Load velocity offset for WZ move	(5.9)
RX$\begin{bmatrix}\text{nn}\\\text{Vd}\end{bmatrix}$	Load radius for XY move	(5.3)
RW$\begin{bmatrix}\text{nn}\\\text{Vd}\end{bmatrix}$	Load radius for WZ move	(5.3)
IXnn	Add increment to radius for XY move	
IWnn	Add increment to radius for WZ move	
RAa(a)(a)(a)	Run absolute, all axes given	(5.6)
RIa(a)(a)(a)	Run incremental, all axes given	(5.6)
RRX	Run radius XY	(5.7.3)
RRW	Run radius WZ	(5.7.3)
RRA	Run radius, all axes	(5.7.3)
CRX	Run circular, XY	(5.7.1)
CRW	Run circular, WZ	(5.7.1)
CRA	Run circular, XY and WZ	(5.7.1)
ELa$\begin{bmatrix}\text{nn}\\\text{Vd}\end{bmatrix}$	Load ellipse ratio	(5.7.2)
SNann	Step input command to axis	(4.2.2.7)
PTAxxBannn	Load axis set point A begin position, range xx	(5.14.1)
PTAxxEannn	Load axis set point A end position, range xx	(5.14.1)
PTAxxFa	Forget axis set point A range xx	(5.14.1)
PTBxxBannn	Load axis set point B begin position, range xx	(5.14.1)
PTBxxEannn	Load axis set point B end position, range xx	(5.14.1)
PTBxxFa	Forget axis set point B range xx	(5.14.1)
SPAa$\begin{bmatrix}\text{A}\\\text{B}\end{bmatrix}a\begin{bmatrix}\text{A}\\\text{B}\end{bmatrix}$	Define CMC-0 set point A as occurrence of two axis set points	(5.14.2)
SPBa$\begin{bmatrix}\text{A}\\\text{B}\end{bmatrix}a\begin{bmatrix}\text{A}\\\text{B}\end{bmatrix}$	Define CMC-0 set point B as occurrence of two axis set points	(5.14.2)
SPCa$\begin{bmatrix}\text{A}\\\text{B}\end{bmatrix}a\begin{bmatrix}\text{A}\\\text{B}\end{bmatrix}$	Define CMC-0 set point C as occurrence of two axis set points	(5.14.2)
SPDa$\begin{bmatrix}\text{A}\\\text{B}\end{bmatrix}a\begin{bmatrix}\text{A}\\\text{B}\end{bmatrix}$	Define CMC-0 set point D as occurrence of two axis set points	(5.14.2)
SPAF	Forget CMC-0 set point A	(5.14.2)
SPBF	Forget CMC-0 set point B	(5.14.2)
SPCF	Forget CMC-0 set point C	(5.14.2)
SPDF	Forget CMC-0 set point D	(5.14.2)
TCn	Initialize teach mode for profile n	(5.15)
TMa(nnn)	Load teach point	(5.15)

Conditional execution and profile branching commands:

These commands can also be executed interactively or stored within profiles. They allow you to control profile execution through logic functions and mathematical comparisons of integer variables. See Section 5.16 for details and examples.

Note: these commands use the following notation:

A - AND function	I - User Input	> - Greater than
O - OR function	V - User Integer Variable	= - Equal to
X - XOR function	B - User Boolean Variable	<> - Not equal to
N - NOT function		

In addition, brackets indicate that the programmer must choose one of the items within the brackets. Parentheses indicate optional items.

Mnemonic	Action
WTTnn	Wait until timer is elapsed
WTMa	Wait until axis is stopped
WTMA	Wait until all axes are stopped
WTI	Wait until index pulse
WTE $\left[\begin{matrix} (N) \begin{bmatrix}B\\I\end{bmatrix} d \left(\begin{bmatrix}A\\O\\X\end{bmatrix} (N) \begin{bmatrix}B\\I\end{bmatrix} d \right) \\ Vd \begin{bmatrix}>\\=\\<>\end{bmatrix} Vd \end{matrix}\right]$	Wait until expression is true
IF $\left[\begin{matrix} (N) \begin{bmatrix}B\\I\end{bmatrix} d \left(\begin{bmatrix}A\\O\\X\end{bmatrix} (N) \begin{bmatrix}B\\I\end{bmatrix} d \right) \\ Vd \begin{bmatrix}>\\=\\<>\end{bmatrix} Vd \end{matrix}\right]$ GSn	If expression true, GOSUB profile n
IF $\left[\begin{matrix} (N) \begin{bmatrix}B\\I\end{bmatrix} d \left(\begin{bmatrix}A\\O\\X\end{bmatrix} (N) \begin{bmatrix}B\\I\end{bmatrix} d \right) \\ Vd \begin{bmatrix}>\\=\\<>\end{bmatrix} Vd \end{matrix}\right]$ GTn	If expression true, GOTO profile n

Note: Use of both integer variables and either Boolean variables or user inputs in the same conditional expression is not allowed. Also, if you wish to complement only one logical operand in an expression, then it must be the first.

Auxiliary Position Equation commands:

These commands allow you to control individual axes with the Auxiliary Position Equation. They are useful for jogging functions, encoder tracking, and joystick applications, to name a few. See Section 5.10 for details and examples.

Note: these commands use the following notation:

A - Analog input channel
N - Number
E - External encoder input
F - Internal Frequency Source
V - Integer Variable

Mnemonic	Action
FSa$\begin{bmatrix}\text{n}\\ \text{Vd}\end{bmatrix}$	Load axis frequency source value
AEab	Enable/disable axis Auxiliary Position Equation
LMa$\begin{bmatrix}\text{Ad}\\ \text{Nnn}\\ \text{Vd}\end{bmatrix}$	Load axis Auxiliary Position Equation multiplier, either analog input, constant number, or integer variable
LIa$\begin{bmatrix}\text{Ed}\\ \text{F}\end{bmatrix}$	Load axis Auxiliary Position Equation input, either external encoder or internal frequency source

Phase-Locked Loop commands:

These commands are used only for phase-locked loop (PLL) applications. See Section 5.13 for details and examples.

Mnemonic	Action
PHGn	Load PLL gain
PHZn	Load PLL zero
PHO$\begin{bmatrix}\text{nn}\\ \text{Vd}\end{bmatrix}$	Load PLL offset
PHL$\begin{bmatrix}\text{nn}\\ \text{Vd}\end{bmatrix}$	Load PLL cycle length
PHCnn	Load PLL correction limit
SPH	Set PLL counter to zero

I/O and Variable commands:

These commands are for use with the user I/O lines, integer variables, and analog inputs. Some are allowed in defined profiles, while others must be executed interactively. Those marked with a (#) are not allowed in defined profiles.

Numbers in parentheses after the command description indicate sections of the manual where more detailed information is available about the command.

Mnemonic		Action	
ST$\begin{bmatrix}B\\I\end{bmatrix}$d		Set Boolean variable or I/O pin	
RS$\begin{bmatrix}B\\I\end{bmatrix}$d		Reset Boolean variable or I/O pin	
DRdb	(#)	Set direction of I/O pin (output or input)	(4.2.4)
RAd	(#)	Report analog input value	(4.4.3)
RDB	(#)	Report Boolean variable values	(4.4.3)
RDI	(#)	Report user input values	(4.4.3)
ADdnn		Load analog channel deadband	(4.2.5)
AOd$\begin{bmatrix}nn\\Vd\end{bmatrix}$		Load analog channel offset	(4.2.5)
Vd=?	(#)	Report integer variable value	(5.11)
Vd=a$\left(\begin{bmatrix}*\\/\\+\\-\end{bmatrix}b\right)$		Arithmetic integer variable assignment	(5.11)
Vd=nnnn		Fixed number integer variable assignment	(5.11)

In the first integer variable assignment command, "a" and "b" can be:

PHE	- PLL error (in position pulses)
PHP	- PLL master position (in position pulses)
Ed	- External encoder input (represents velocity)
Vd	- Other integer variable
(N)Bd	- Boolean variable ("N" is optional, implies complement)
(N)Id	- User I/O pin ("N" is optional, implies complement)
Td	- Thumbwheel input
Ad	- Analog input
Pa	- Axis position

(#) - Not allowed in defined profiles.

Status and position query commands: (Not allowed in defined profiles.)

Mnemonic	Action	
RS	Report status of CMC-0 unit	(4.3.1)
RSa	Report status of axis	(4.3.1)
RCa	Report command position of axis	(4.4.2)
RPa	Report actual position of axis	(4.4.2)
FEa	Report following error of axis	(4.4.2)
FC	Report fault code of CMC-0 unit	(6.1.2)
FCa	Report fault code of axis	(6.1.2)

Editing commands:

These commands are used when you are "inside" a profile definition and need to scroll through it to verify its contents, delete a line, or insert a line. See Section 5.17.

Mnemonic	Action
X	Scroll to next line in profile
DEL	Delete profile line
!	Abort edit mode

5.3 PARAMETERS

5.3.1 Introduction.

The CMC-0 controller provides complete programmability of motion profile parameters. The controller saves all entered parameters in battery-backed RAM (nominal battery life = 10 years). Therefore, unchanged profile parameters need not be reloaded between movement commands or after power failure. However, if a Cold Boot (CB) command is sent, all the parameters will be returned to their default values, and the Memory Lost flag in the CMC-0 Status Register will be set. If the Memory Lost flag is set upon power up or when a Cold Boot command has not been issued, then the unit most likely has a hardware problem (see Section 6.0).

To load any parameter, send the appropriate load command with the signed BCD representation of the value. When the number is positive, the plus sign is optional. Leading zeros are also not required but will be accepted.

To find the current value of a parameter, send the appropriate load command with a "?" in place of the value. For example, sending the query "4AMX?<CR>" would return the current absolute move value for axis "X" (assuming the address of the CMC-0 is 4). The query suffix works for any of the load commands, even the ones not discussed in this section.

5.3.2 Internal Parameter Representation.

Whenever a parameter is entered by the user (user parameter), the CMC-0 converts it to an internal representation (internal parameter) which it uses for all control mathematics. The units associated with the internal parameters are generally different from those associated with the user parameters. For example, the CMC-0 treats distances (internally) in terms of position pulses (pp), even though the user may be programming distances in terms of inches or millimeters.

Table 5-1 on the next page illustrates the acceptable ranges for the internal parameter representations.

Note that these ranges set absolute limits on each parameter, no matter what the Unit Ratio (position pulses/unit) is set at for each axis.

Table 5-1 Internal Parameter Ranges

Parameter Description	Load Command	Min. Internal Value	Max. Internal Value	Units
Absolute Move	AMa $\begin{bmatrix}\text{nnn}\\ \text{Vd}\end{bmatrix}$	0	+/- 8,388,607	pp
Incremental Move	IMa $\begin{bmatrix}\text{nn}\\ \text{Vd}\end{bmatrix}$	0	+/- 65,535	pp
Acceleration	ACa $\begin{bmatrix}\text{nn}\\ \text{Vd}\end{bmatrix}$	80,000	6,500,000	pp/s/s
Trajectory Vel.	VX $\begin{bmatrix}\text{nn}\\ \text{Vd}\end{bmatrix}$	10	655,350	pp/s
	VW $\begin{bmatrix}\text{nn}\\ \text{Vd}\end{bmatrix}$	10	655,350	pp/s
Radius	RX $\begin{bmatrix}\text{nn}\\ \text{Vd}\end{bmatrix}$	0	+/- 65,535	pp
	RW $\begin{bmatrix}\text{nn}\\ \text{Vd}\end{bmatrix}$	0	+/- 65,535	pp
Radius Increment	IXnn	0	+/- 32,767	pp
	IWnn	0	+/- 32,767	pp
Velocity Offset	VOXnn	0	+/- 25,600	N/A
	VOWnn	0	+/- 25,600	N/A
Ellipse Ratio	ELa $\begin{bmatrix}\text{nn}\\ \text{Vd}\end{bmatrix}$	0	65,535	N/A
Step Input	SNann	0	+/- 30,000	pp
Wait Time	WTTnn	0	655,350	ms
Repeat Move	RMnn	0 (infinite)	32,767	N/A
Frequency Source	FSan	200	51,200	pp/s
Aux. Eqn. Mult.	LMa $\begin{bmatrix}\text{Nnn}\\ \text{Vd}\end{bmatrix}$	0	+/- 25,600	N/A
Analog Deadband	ADdnn	0	65,535	N/A
Analog Offset	AOd $\begin{bmatrix}\text{nn}\\ \text{Vd}\end{bmatrix}$	0	+/- 32,767	N/A
PLL Offset	PHO $\begin{bmatrix}\text{nn}\\ \text{Vd}\end{bmatrix}$	0	+/- 32,767	pp
PLL Length	PHL $\begin{bmatrix}\text{nn}\\ \text{Vd}\end{bmatrix}$	2,500	60,000	pp
PLL Correction Lim.	PHCnn	0	65,535	N/A
PLL Gain	PHGnn	0	511	N/A
PLL Zero	PHZnn	0	255	N/A

It is especially important to keep these limits in mind when using the integer variables to load the parameters. Whenever a variable is used to load a parameter, the controller interprets the contents of the variable in terms of the units given above, so care must be taken not to allow the contents of the variable to exceed the absolute parameter ranges. See Section 5.11 for more details about using variable parameters.

5.3.3 User Parameter Representation.

For programming simplicity, the CMC-0 allows you to define the unit of length that is most convenient for your application by specifying the Unit Ratio for each axis (refer to Section 4.2 for details). This allows you to program distances in terms of units, velocities in terms of units/sec., etc.

However, this means that if your unit of length is not 1 position pulse (i.e. Unit Ratio not equal to 1) , you must use the table below to determine the acceptable ranges for the motion parameters.

Note: In the table, "u" indicates length units.

Table 5-2 User Parameter Ranges

Parameter Description	Load Command	Min. User Value*	Max. User Value*	Units
Absolute Move	AMannn	0.0001 - 1	838.8607 - 8,388,607	u
Incremental Move	IMann	0.0001 - 1	6.5535 - 65,535	u
Acceleration	ACann	8 - 80,000	650 - 6,500,000	u/s/s
Trajectory Vel.	VXnn	.01 - 10	65.53 - 655,350	u/s
	VWnn	.01 - 10	65.53 - 655,350	u/s
Radius	RXnn	0	6.5535 - 65,535	u
	RWnn	0	6.5535 - 65,535	u
Radius Increment	IXnn	0	3.2767 - 32,767	u
	IWnn	0	3.2767 - 32,767	u
Velocity Offset	VOXnn	0	+/- 100.00	N/A
	VOWnn	0	+/- 100.00	N/A
Ellipse Ratio	ELann	0	1.0000	N/A
Step Input	SNann	0	3.0000 - 30,000	u
Wait Time	WTTnn	0	655.35	s
Repeat Move	RMnn	0 (infinite)	32,767	N/A
Frequency Source	FSan	.02 - 200	5.12 - 51,200	u/s
Aux. Eqn. Mult.	LMaNnn	0	+/- 100.00	N/A
Analog Deadband	ADdnn	0	65,535	#
Analog Offset	AOdnn	0	+/- 32,767	#
PLL Offset	PHOnn	0	+/- 32,767	pp
PLL Length	PHLnn	2,500	60,000	pp
PLL Correction Lim.	PHCnn	0	65,535	N/A
PLL Gain	PHGnn	0	511	N/A
PLL Zero	PHZnn	0	255	N/A

* - If a range is given for minimum and/or maximum values, the actual value depends on the Unit Ratio for that axis. For example, the absolute maximum user value for incremental moves is 65,535, which corresponds to a Unit Ratio of 1. However, if the Unit Ratio were set to 2, then the maximum user value for incremental moves would be 32,767.5.

- The units for analog channel deadband and offset are A/D counts.

5.4 ACCELERATION EFFECTS

To improve the overall accuracy and performance capabilities of the system, the CMC-0 controller allows you to define an acceleration rate for each axis. Since these acceleration rates may affect the motion in different ways depending on the situation, keep the following items in mind as you define and execute profiles with a CMC system:

- In cases where the combination of acceleration rates and trajectory velocities are impossible to achieve for a given move, the CMC-0 controller will use the values closest to (but not greater than) the values entered by the user.

- When chaining linear segments and/or arc segments together to form a contour, keep in mind that the controller may decelerate to a stop at the end of one segment before starting the next one, depending on the magnitude of the acceleration rate and the degree of continuity at the point where the segments join together. As the acceleration rate increases, the controller will tend not to stop the axes between segments. (See Example Profile 9, Section 5.8.2.)

  This is especially true when chaining arc segments to straight line segments or other arc segments when the Ellipse Ratio is not equal to 1. As the Ellipse Ratio gets smaller, the controller will have a greater tendency to decelerate to a stop between segments.

5.5 SETTING POSITION (and homing axes)

In many applications, the first thing you will want to do is "home" each axis to a specific location and assign a position to this location. The CMC-0 has eight commands which are dedicated to this function. They are listed below:

Mnemonic	Action
STZa	Immediately set at zero
SHZa	Find home input, then set at zero
SIZa	Find index pulse, then set at zero
SBZa	Find both, then set at zero
STPannn	Immediately set at position nnn
SHPannn	Find home input, then set at position nnn
SIPannn	Find index pulse, then set at position nnn
SBPannn	Find both, then set at position nnn

Each of these commands works in conjunction with the Auxiliary Position Equation. (This equation and its uses are discussed fully in Section 5.10, but since we need it for this next example, we'll introduce it now very briefly.)

The Auxiliary Position Equation allows you to control any axis with the equation

$$P = Ax + B$$

where:

"A" may be

1. A fixed number,
2. One of the analog inputs, or
3. An integer variable

"x" may be

1. One of the external encoder inputs, or
2. A user-selectable internal frequency source

"B" is the previous position of the axis. That is, the product "Ax" is an incremental addition to the last position of the axis.

When used with the Auxiliary Position Equation, the "set position" commands allow you to jog any of the axes to a "home" position. The example profile which follows homes the X axis using the SBZa command.

Example Profile 1: Homing Routine

```
Command           Description
 4DE1     <CR>    Define profile 1
*4FSX.5   <CR>    Load frequency source for X axis of 0.5 units/sec.
*4LIXF    <CR>    Specify frequency source as input for Auxiliary Eq'n.
*4LMXN-1  <CR>    Load multiplier of -1 for Auxiliary Eq'n.
*4SBZX    <CR>    Run until home input and index pulse occur
*4ED      <CR>    End profile definition
*4GT1     <CR>    Execute profile 1
*
```

Note: The stars (*) are the responses from the controller, and the controller is assumed to be addressed at "4".

In the above example, the X axis will begin jogging in the negative direction at a speed of 0.5 units/sec as soon as the "SBZX" command is issued. It will stop only after it has detected the home input signal followed by the index pulse of the axis encoder. It will then set its position register to zero.

The other "set position" commands work in a similar way. Note that all "set position" commands enable the Auxiliary Position Equation when they are issued, and disable it when they are finished. Also note that the axis must be stopped before any of the "set position" commands are issued or the command will be ignored.

5.6 LINEAR MOVES

Linear moves with the CMC system fall into two categories:

1. Single-axis moves (not coordinated with other axes), and
2. Multi-axis moves involving linear interpolation (coordinated)

Two definitions may be helpful:

Incremental Move -	A linear move which causes the motor to travel a specified distance from its current position.
Absolute Move -	A linear move which causes the motor to travel to an absolute location with respect to the zero position.

In short, incremental moves cause the motor to travel a specific distance (positive or negative) from its current position, whereas absolute moves cause the motor to move to a specific position, no matter how far it must travel to get there.

5.6.1 Single-Axis Moves

Single-axis moves with the CMC-0 can be accomplished in either of two ways. First, you can use the Auxiliary Position Equation to move an individual axis. This is described fully in Section 5.10.

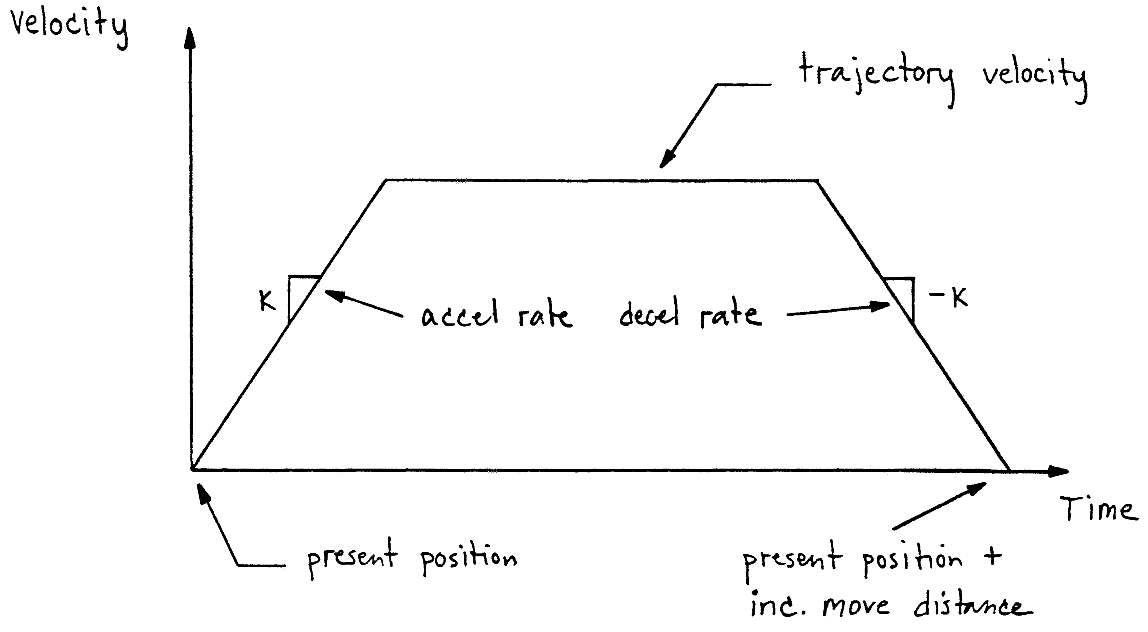
Alternatively, you may use the "Incremental Move" and "Absolute Move" commands to make trapezoidal position moves. To do this, you must:

1. Load the axis acceleration rate. (*)
2. Load the trajectory velocity. (*)
3. Load the axis incremental move or absolute move register. (*)
4. Issue the "run incremental" or "run absolute" command.

(*) - Remember that the CMC-0 saves all parameters after they are loaded, and the only way to change them is to enter new values or execute a defined profile which changes them during execution. As a result, it is not necessary to enter parameters each time you make a move if the desired values have not changed.

The following example profile shows how to program a simple incremental move for a single axis. The profile corresponds to Figure 5-1 on the next page.

Figure 5-1 Single-Axis Incremental Move



Example Profile 2: Single-Axis Incremental Move

Command		Description
4DE2	<CR>	Define profile 2
*4ACX20	<CR>	Load axis acceleration rate of 20 units/sec./sec.
*4VX5	<CR>	Load trajectory velocity of 5 units/sec.
*4IMX10	<CR>	Load incremental move distance of +10 units
*4RIX	<CR>	Issue "run incremental" command, X axis
*4ED	<CR>	End profile definition
*4GT2	<CR>	Execute profile 2
*		

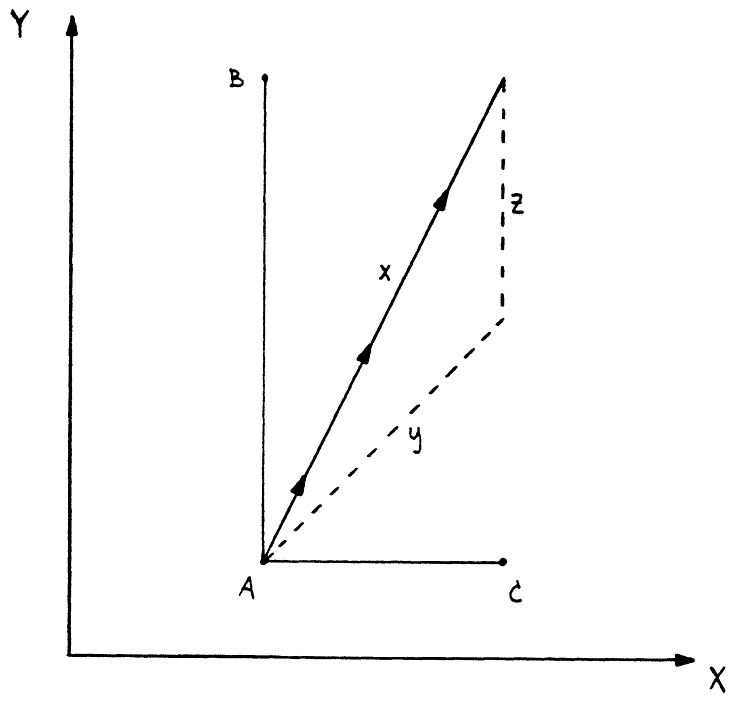
Note: The stars (*) are the responses from the controller, and the controller is assumed to be addressed at "4".

5.6.2 Multi-Axis Moves (Linear Interpolation).

Linear moves involving more than one axis use linear interpolation to coordinate the axes. This means that the positions of the axes are coordinated with each other as a function of time so that all axes reach their final positions at exactly the same time.

Figure 5-2 on the next page illustrates this idea for two axes.

Figure 5-2 Linear Interpolation: Two Axes



In the figure, you can see that the distance from A to C is smaller than the distance from A to B. If you told an "uncoordinated" controller to move the X axis from A to C and the Y axis from A to B at the same time, you might expect the trajectory path to look something like the path yz, where the X axis reaches C before the Y axis reaches B.

However, since the CMC-0 controller uses linear interpolation to coordinate the velocities of the axes, the actual path taken is path x in the figure. That is, the controller guarantees that axis X reaches C at the same time that axis Y reaches B.

The CMC-0 allows you to coordinate up to four axes in this manner using either absolute or incremental moves. The profile on the following page is an example of 4-axis linear interpolation. It moves axes X and Y from the zero position to the 5 unit mark, while moving the W and Z axes from the zero position to the 10 unit mark. In spite of the different move lengths, all four axes reach the final position at the same time.

Example Profile 3: 4-Axis Linear Interpolation

Command		Description
4DE3	<CR>	Define profile 3
*4ACX20	<CR>	Load X axis acceleration rate of 20 units/sec./sec.
*4ACY20	<CR>	Load Y axis acceleration rate of 20 units/sec./sec.
*4ACW20	<CR>	Load W axis acceleration rate of 20 units/sec./sec.
*4ACZ20	<CR>	Load Z axis acceleration rate of 20 units/sec./sec.
*4VX2	<CR>	Load trajectory velocity of 2 units/sec., XY axes
*4VW4	<CR>	Load trajectory velocity of 4 units/sec., WZ axes
*4STZX	<CR>	Set X axis position to zero
*4STZY	<CR>	Set Y axis position to zero
*4STZW	<CR>	Set W axis position to zero
*4STZZ	<CR>	Set Z axis position to zero
*4AMX5	<CR>	Load absolute move of 5 units, X axis
*4AMY5	<CR>	Load absolute move of 5 units, Y axis
*4AMW10	<CR>	Load absolute move of 10 units, W axis
*4AMZ10	<CR>	Load absolute move of 10 units, Z axis
*4RAXYWZ	<CR>	Run absolute, axes X, Y, W, and Z
*4ED	<CR>	End profile definition
*4GT3	<CR>	Execute profile 3
*		

Note: The stars (*) are the responses from the controller, and the controller is assumed to be addressed at "4".

In the example, the trajectory velocity for the WZ axes is twice as large as the trajectory velocity for the XY axes. This makes sense because the W and Z axes have twice as far to go as the X and Y axes.

However, what would happen if you tried to execute the above profile with the two trajectory velocities the same (VX = VW = 6 in the above profile, for example)? Since the W and Z axes have twice as far to go as the XY pair, you might reasonably assume that they will finish moving after the XY pair has stopped. This is not the case, though. The CMC-0 determines which pair of axes (XY or WZ) would take the longest time to complete the move with the programmed trajectory velocity, and then scales the velocity for the other axes so that they take the same amount of time.

If VX = VW = 6 in the above example, then the WZ pair would take the longest time to complete the move (since the WZ distance is twice the XY distance). Therefore, the CMC-0 internally changes the velocity for the XY pair to 3 so that all the axes finish moving at the same time.

Remember, the primary function of the CMC-0 during linear moves is to guarantee that all axes finish moving at the same time. If the trajectory velocities entered by the user do not permit this, then the CMC-0 changes them internally so that they do.

5.7 CIRCULAR MOVES

In addition to linear interpolation, the CMC-0 has circular interpolation capabilities which allow it to make circular and elliptical moves. It is also capable of generating arc segments with a maximum included angle of 120 degrees.

The commands which are used for circular interpolation are:

Mnemonic	Action
RX $\begin{bmatrix}nn\\Vd\end{bmatrix}$	Load radius for XY move
RW $\begin{bmatrix}nn\\Vd\end{bmatrix}$	Load radius for WZ move
IXnn	Add increment to radius for XY move
IWnn	Add increment to radius for WZ move
RRX	Run radius XY
RRW	Run radius WZ
RRA	Run radius, all axes
CRX	Run circular, XY
CRW	Run circular, WZ
CRA	Run circular, XY and WZ
ELa $\begin{bmatrix}nn\\Vd\end{bmatrix}$	Load ellipse ratio

Examples which illustrate the use of these commands are given in the following sections.

5.7.1 Circles.

The "CRX", "CRW", and "CRA" commands allow you to generate complete circles based on radius and velocity information. The "CRX" command uses the X and Y axes, while the "CRW" command uses the W and Z axes. The example profile given below shows how to program this type of move for the X and Y axes.

Example Profile 4: Generating a Circle

Command		Description
4DE4	<CR>	Define profile 4
*4ACX30	<CR>	Load X axis acceleration rate of 30 units/sec./sec.
*4ACY30	<CR>	Load Y axis acceleration rate of 30 units/sec./sec.
*4VX5	<CR>	Load trajectory velocity of 5 units/sec., XY axes
*4RX2	<CR>	Load radius of 2 units
*4CRX	<CR>	Generate circle, XY axes
*4ED	<CR>	End profile definition
*4GT4	<CR>	Execute profile 4
*		

Note: The stars (*) are the responses from the controller, and the controller is assumed to be addressed at "4".

The "CRA" command is similar to the "CRX" and "CRW" commands, except it causes both circles to be generated in the same amount of time. In many cases this requires the controller to scale the trajectory velocity of one axis pair in order to make both circles finish at the same time.

For example, suppose you have programmed the X and Y axes for a radius of 2 units, and the W and Z axes for a radius of 4 units. Furthermore, suppose you have defined a trajectory velocity of 5 units/sec for both axis pairs. Clearly, the WZ move would take twice as long as the XY move because it has twice as far to go. Because of this, the controller would scale the velocity of the XY move back to 2.5 units/sec so that both circles would finish together. This is completely analogous to what happens when you perform linear interpolation with more than two axes.

Note: The "CRX", "CRW", and "CRA" commands only generate circles in the counter-clockwise direction, which corresponds to positive radius values. If you enter a negative radius and try to execute one of these commands, the command will be ignored.

To generate circles in the clockwise direction you must use the "run radius" commands discussed in Section 5.7.3.

Caution: The CRX, CRW, and CRA commands use the incremental move registers to store intermediate parameters. Do not assume that values previously stored in these registers will be left unchanged after executing one of these commands.

5.7.2 Ellipses.

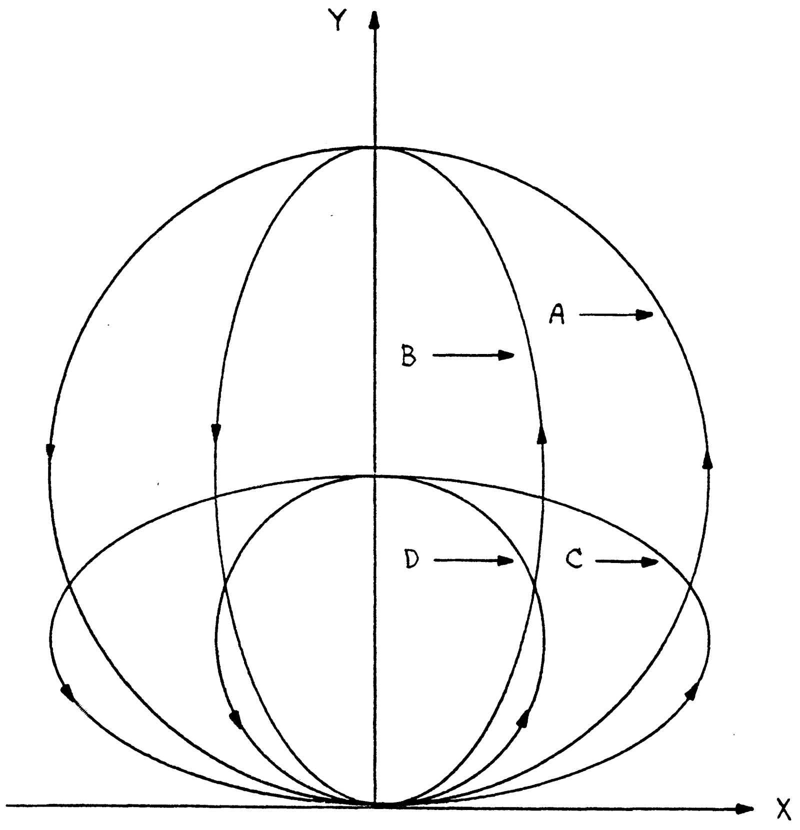
Each axis has an Ellipse Ratio associated with it which is used in conjunction with the circular interpolation commands to determine how circular the resulting moves will be. This constant is generally left at 1 for all axes so that all circle commands produce true circles. However, if you make this constant less than 1 for a given axis, then any circular moves involving that axis will produce an ellipse.

The Ellipse Ratio works by reducing the travel distance of the specified axis. For example, suppose you have defined a radius of 2 units for the XY axis pair. This is equivalent to a circle diameter of 4 units, so each axis travels a total of 4 units while moving from one side of the circle to the opposite side (if the Ellipse Ratio equals 1).

Now suppose you change the Ellipse Ratio of the X axis to 0.5. This causes the X axis to move only half the distance that it normally would when generating a full circle. The result is an ellipse which is 4 units long (in the Y direction) and 2 units long (in the X direction). Note that if you were to set the Ellipse Ratio of both the X and Y axes to 0.5, you would still get a true circle, except it would only be 2 units in diameter instead of 4.

Figure 5-3 below illustrates the general idea. Curve A is the original circle which corresponds to an Ellipse Ratio of 1 for both the X and Y axes. Curve B is the result of setting the Ellipse Ratio of axis X to 0.5, and the Ellipse Ratio of axis Y to 1. Curve C is the result of setting the Ellipse Ratio of axis Y to 0.5, and the Ellipse Ratio of axis X to 1. Finally, curve D is the result of setting the Ellipse Ratios of both axes to 0.5.

Figure 5-3 Effect of Ellipse Ratio



Note that the Ellipse Ratio affects the "run radius" commands as well as the "CRX", "CRW", and "CRA" commands.

Example profile 5 on the next page shows how to program the ellipse of curve B in the above figure.

Example Profile 5: Ellipse Profile

Command		Description
4DE5	<CR>	Define profile 5
*4ACX30	<CR>	Load X axis acceleration rate of 30 units/sec./sec.
*4ACY30	<CR>	Load Y axis acceleration rate of 30 units/sec./sec.
*4VX5	<CR>	Load trajectory velocity of 5 units/sec., XY axes
*4RX2	<CR>	Load radius of 2 units
*4ELX.5	<CR>	Load Ellipse Ratio of 0.5 for X axis
*4ELY1	<CR>	Load Ellipse Ratio of 1 for Y axis
*4CRX	<CR>	Generate circle (ellipse), XY axes
*4ED	<CR>	End profile definition
*4GT5	<CR>	Execute profile 5
*		

Note: The stars (*) are the responses from the controller, and the controller is assumed to be addressed at "4".

5.7.3 Arc Segments.

The CMC-0 controller also allows you to produce arc segments using the "run radius" commands "RRX", "RRW", and "RRA". With these commands you can chain arbitrary arcs together to form a desired contour.

To define this type of motion you must:

1. Define an incremental move distance for both axes involved.

2. Define a trajectory velocity for the move.

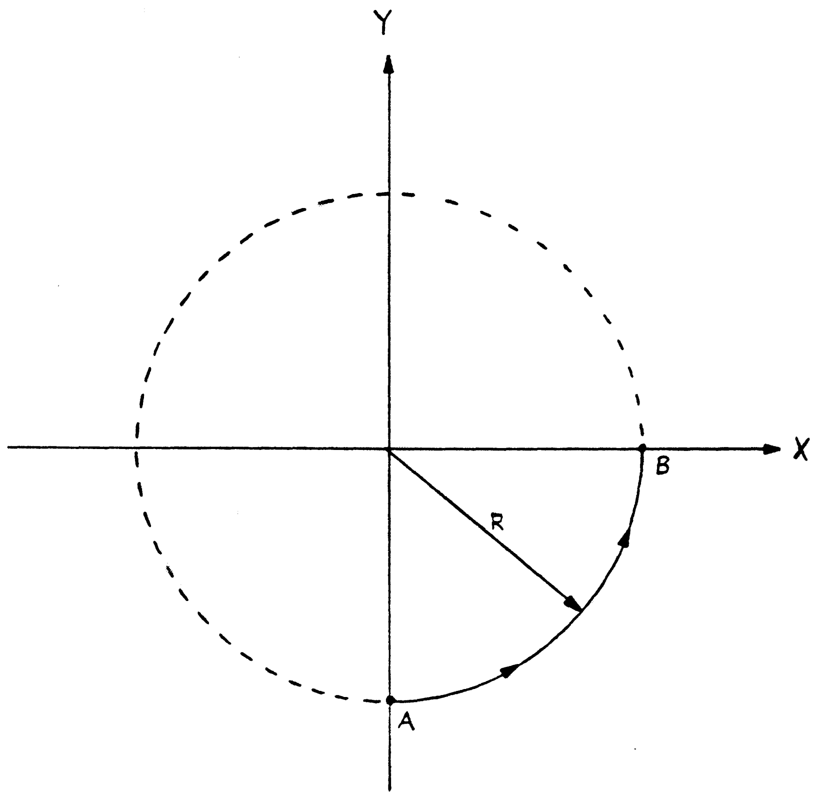
3. Define an acceleration rate for each axis.

4. Define a radius of curvature for the move. Positive radius values produce counter-clockwise motion; negative values produce clockwise motion.

5. Issue the appropriate "run radius" command.

Figure 5-4 on the next page illustrates how to define an arc. Example profile 6 shows how to execute this type of move.

Figure 5-4 Arc Segment Definition



Example Profile 6: Arc Segment Profile

Command		Description
4DE6	<CR>	Define profile 6
*4ACX20	<CR>	Load X axis acceleration rate of 20 units/sec./sec.
*4ACY20	<CR>	Load Y axis acceleration rate of 20 units/sec./sec.
*4VX5	<CR>	Load trajectory velocity of 5 units/sec., XY axes
*4IMX2	<CR>	Load incremental move distance, X axis
*4IMY2	<CR>	Load incremental move distance, Y axis
*4RX2	<CR>	Load radius of curvature of 2 units
*4RRX	<CR>	Execute arc segment move from A to B.
*4ED	<CR>	End profile definition
*4GT6	<CR>	Execute profile 6
*		

Note: The stars (*) are the responses from the controller, and the controller is assumed to be addressed at "4".

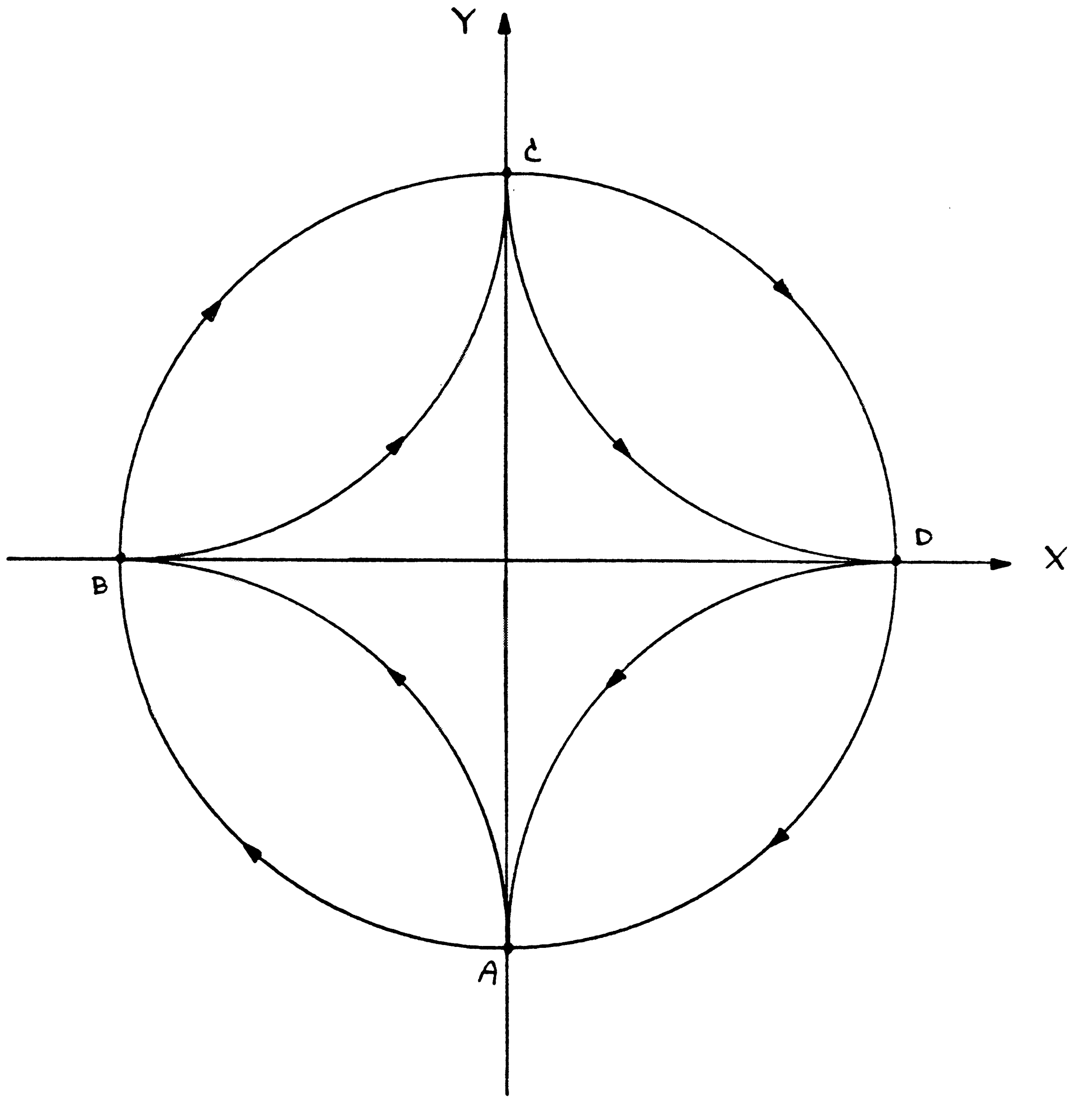
In the above example the radius of curvature is positive, which results in a counter-clockwise motion. To obtain a clockwise motion simply make the radius a negative number.

Note: If you specify a radius of curvature which causes the included angle to be greater than 120 degrees, the controller will generate a straight line between the points instead of an arc.

The next example shows how to link together four similar arcs to form a circle. Unlike the "CRX", "CRW", and "CRA" commands, which can only generate counter-clockwise circles, this example generates the circle in the clockwise direction.

Figure 5-5 illustrates how the circle is oriented in the plane. Note that when the radius is defined as negative the resultant contour is the circle (because each arc segment is generated in a clockwise direction). When the radius is defined as positive the result is the "star" pattern inside the circle (because each arc is generated in the counter-clockwise direction).

Figure 5-5 Contour Generation Using Arc Segments



Example Profile 7: Contour Generation

Command		Description
4DE7	<CR>	Define profile 7
*4ACX20	<CR>	Load X axis acceleration rate of 20 units/sec./sec.
*4ACY20	<CR>	Load Y axis acceleration rate of 20 units/sec./sec.
*4VX5	<CR>	Load trajectory velocity of 5 units/sec., XY axes
*4RX-2	<CR>	Load XY radius of 2 units, clockwise motion
*4IMX-2	<CR>	Load incremental move distance, X axis
*4IMY2	<CR>	Load incremental move distance, Y axis
*4RRX	<CR>	Execute arc segment move from A to B
*4IMX2	<CR>	Load incremental move distance, X axis
*4RRX	<CR>	Execute arc segment move from B to C
*4IMY-2	<CR>	Load incremental move distance, Y axis
*4RRX	<CR>	Execute arc segment move from C to D
*4IMX-2	<CR>	Load incremental move distance, X axis
*4RRX	<CR>	Execute arc segment move from D to A
*4ED	<CR>	End profile definition
*4GT7	<CR>	Execute profile 7
*		

Note: The stars (*) are the responses from the controller, and the controller is assumed to be addressed at "4".

5.8 COMBINING LINEAR AND CIRCULAR INTERPOLATION

5.8.1 3-Dimensional Cylindrical Moves.

The "run radius" command RRA allows you to coordinate circular motion on two axes with one or two axes of linear motion. To do this you must:

1. Define the desired arc segment or circular move for either the XY or WZ axis pair.

2. Define an incremental move for the remaining axis (or axes).

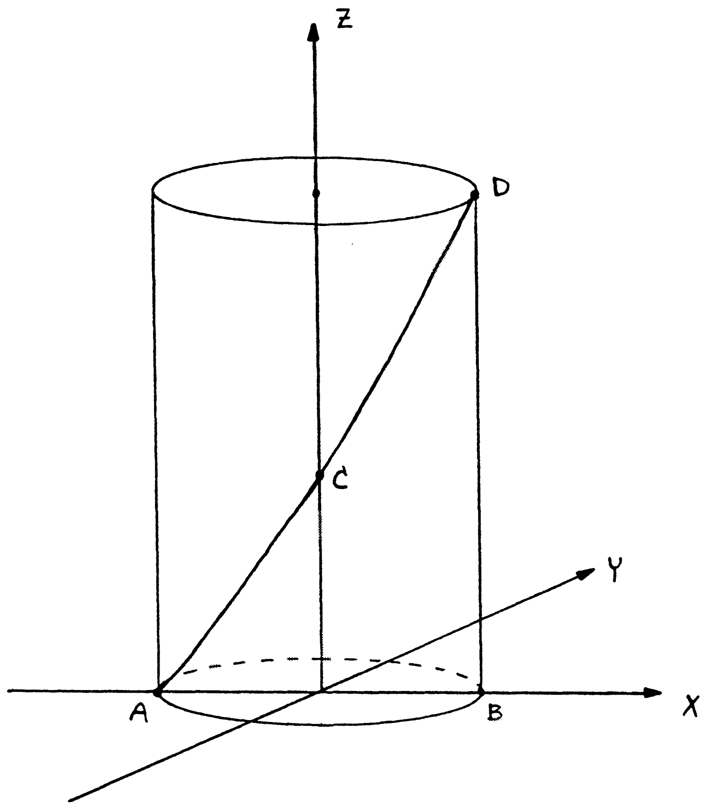
3. Set the radius to zero for the axis (or axes) which will make the linear move.

4. Execute the RRA command.

Whenever you issue a "run radius" command when the associated radius is zero, the controller executes a linear move between the two points instead of an arc. Therefore, you can easily combine both linear and circular interpolation with the RRA command.

Example profile 8 corresponds to the curve in Figure 5-6 on the next page. The circular portion of the profile (the semi-circle from A to B) is generated by the XY axis pair, while the linear portion of the profile (from the bottom of the cylinder to the top) is generated by the Z axis. The resulting motion is the curve from A to D.

Figure 5-6 3-Dimensional Cylindrical Move



Example Profile 8: 3-Dimensional Cylindrical Move

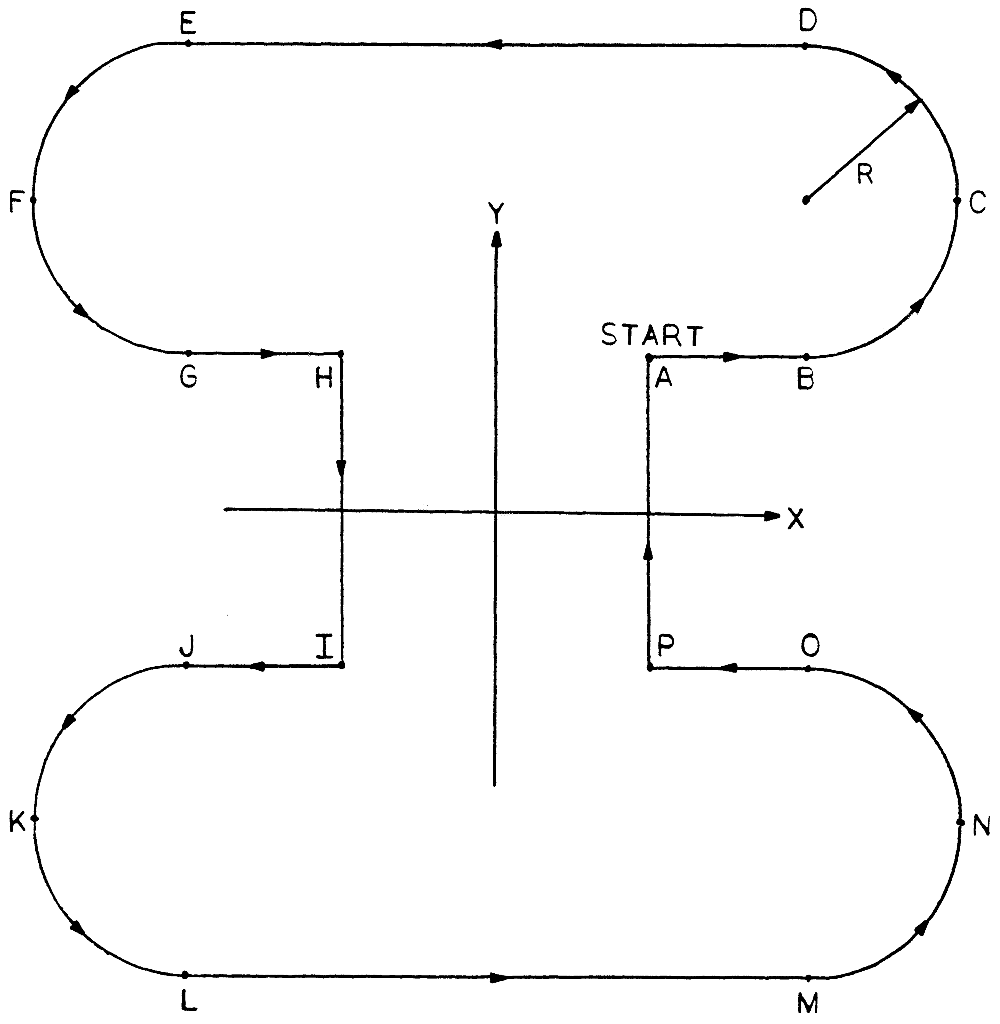
Command		Description
4DE8	<CR>	Define profile 8
*4ACX20	<CR>	Load X axis acceleration rate of 20 units/sec./sec.
*4ACY20	<CR>	Load Y axis acceleration rate of 20 units/sec./sec.
*4ACZ20	<CR>	Load Z axis acceleration rate of 20 units/sec.
*4VX3	<CR>	Load XY trajectory velocity of 3 units/sec.
*4VW3	<CR>	Load WZ trajectory velocity of 3 units/sec.
*4RX2	<CR>	Load XY radius of 2 units, counter-clockwise motion
*4RW0	<CR>	Load WZ radius of 0 units (linear interpolation)
*4IMX2	<CR>	Load incremental move distance, X axis
*4IMY-2	<CR>	Load incremental move distance, Y axis
*4IMZ4	<CR>	Load incremental move distance, Z axis
*4RRA	<CR>	Execute 3-dimensional move from A to C
*4IMY2	<CR>	Load incremental move distance, Y axis
*4RRA	<CR>	Execute 3-dimensional move from C to D
*4ED	<CR>	End profile definition
*4GT8	<CR>	Execute profile 8
*		

Note: The stars (*) are the responses from the controller, and the controller is assumed to be addressed at "4".

5.8.2 2-Dimensional Contour Blending.

The example which follows illustrates how you can join linear segments and arc segments together to form a continuous contour. The example profile generates the 2-dimensional contour shown below in Figure 5-7.

Figure 5-7 Contour Blending



Note that the controller may decelerate the axes before reaching points in the contour where the direction changes abruptly (for example, points H, I, and P) if it cannot make the transition without exceeding the programmed acceleration or velocity parameters. This feature increases accuracy by minimizing or eliminating overshoot at transition points which are not "smooth". (See Section 5.4)

Example Profile 9 on the following page defines the motion illustrated above.

Example Profile 9: Contour Blending

Command		Description
4DE9	<CR>	Define profile 9
*4ACX20	<CR>	Load X axis acceleration rate of 20 units/sec./sec.
*4ACY20	<CR>	Load Y axis acceleration rate of 20 units/sec./sec.
*4VX1	<CR>	Load XY trajectory velocity of 1 unit/sec.
*4RX1	<CR>	Load XY radius of 1 unit, counter-clockwise motion
*4IMX1	<CR>	Load incremental move for X axis of 1 unit
*4RIX	<CR>	Run incremental, X axis, A to B
*4IMY1	<CR>	Load incremental move for Y axis of 1 unit
*4RRX	<CR>	Generate arc from B to C
*4IMX-1	<CR>	Load incremental move for X axis of -1 unit
*4RRX	<CR>	Generate arc from C to D
*4IMX-4	<CR>	Load incremental move for X axis of -4 units
*4RIX	<CR>	Run incremental, X axis, from D to E
*4IMX-1	<CR>	Load incremental move for X axis of -1 unit
*4IMY-1	<CR>	Load incremental move for Y axis of -1 unit
*4RRX	<CR>	Generate arc segment from E to F
*4IMX1	<CR>	Load incremental move for X axis of 1 unit
*4RRX	<CR>	Generate arc from F to G
*4RIX	<CR>	Run incremental, X axis, from G to H
*4IMY-2	<CR>	Load incremental move for Y axis of -2 units
*4RIY	<CR>	Run incremental, Y axis, from H to I
*4IMX-1	<CR>	Load incremental move for X axis of -1 unit
*4RIX	<CR>	Run incremental, X axis, I to J
*4IMY-1	<CR>	Load incremental move for Y axis of -1 unit
*4RRX	<CR>	Generate arc from J to K
*4IMX1	<CR>	Load incremental move for X axis of 1 unit
*4RRX	<CR>	Generate arc from K to L
*4IMX4	<CR>	Load incremental move for X axis of 4 units
*4RIX	<CR>	Run incremental, X axis, from L to M
*4IMX1	<CR>	Load incremental move for X axis of 1 unit
*4IMY1	<CR>	Load incremental move for Y axis of 1 unit
*4RRX	<CR>	Generate arc segment from M to N
*4IMX-1	<CR>	Load incremental move for X axis of -1 unit
*4RRX	<CR>	Generate arc segment from N to O
*4RIX	<CR>	Run incremental, X axis, from O to P
*4IMY2	<CR>	Load incremental move for Y axis of 2 units
*4RIY	<CR>	Run incremental, Y axis, from P back to A
*4ED	<CR>	End profile definition
*4GT9	<CR>	Execute profile 9
*		

Note: The stars (*) are the responses from the controller, and the controller is assumed to be addressed at "4".

5.9 USING THE "VOX" AND "VOW" COMMANDS

The "VOX" (velocity offset, XY axes) and "VOW" (velocity offset, WZ axes) commands allow you to adjust the speed used during execution of any available run command (linear or circular) by using an analog input. VOX uses analog channel one, while VOW uses analog channel three.

The commands work as follows:

When the controller executes either the "VOX" or "VOW" command (interactively via serial input or during profile execution), the number given with the command is stored in the appropriate "velocity offset" parameter. Then, whenever a run command is executed, this value is multiplied by the appropriate analog input value and the result (units = 10 pulses/sec) is added to the programmed trajectory velocity.

For example, suppose you program into a profile the command "VOX5", and have a +/- 10 VDC potentiometer signal connected to analog channel 1. Furthermore, suppose the potentiometer signal happens to be at 5 VDC, which corresponds to 1,024 counts. (Remember that the range of each analog input channel is +/- 2,048 counts, and the range of VOX and VOW is +/- 100.00.)

Therefore, the incremental velocity which is added to the programmed trajectory velocity is 5 * 1,024 = 5,120 units. Since the units are 10 pulses/sec, the resultant change is 51,200 pulses/sec. For a system with a unit ratio of 10,000 pulses/unit, this would correspond to an increase in trajectory velocity of just over 5 units/sec. Note that the product of the analog channel value and the velocity offset parameter determines whether the incremental velocity is added to or subtracted from the programmed trajectory velocity. If the result of the multiplication is positive, the incremental velocity is added to the trajectory velocity. If the result is negative, it is subtracted.

Note: The VOX and VOW commands store parameters which, once loaded, are used by every run command. If you use these commands in certain profiles but don't want them to affect other profiles, you must reset the parameter(s) to zero (i.e. "VOX0" and/or "VOW0") in the profiles which should not use them.

Also, these commands do not allow the controller to adjust velocities continuously. The controller only makes velocity offset adjustments at the start of each run command.

5.10 THE AUXILIARY POSITION EQUATION

5.10.1 Introduction.

The Auxiliary Position Equation allows you to control any axis with the equation

$$P = Ax + B$$

where:

"A" may be

1. A fixed number,
2. One of the analog inputs, or
3. An integer variable

"x" may be

1. One of the external encoder inputs, or
2. A user-selectable internal frequency source.

"B" is the previous position of the axis. That is, the product "Ax" is an incremental addition to the last position of the axis.

This feature is very useful for:

1. Fixed-speed jogging,
2. Analog joystick applications,
3. Encoder tracking/trackball applications

The sections which follow discuss each of these applications in detail.

Remember from Section 5.1 that the Auxiliary Position Equation for each axis, when enabled, executes every five milliseconds no matter what else the controller is doing. This means that the Auxiliary Position Equation can be "active" while other profiles are executing. For example, it would be possible to affect an axis with an analog or encoder input at the same time that it was running a linear or circular profile. For this reason, it is best to disable the Auxiliary Position Equation whenever it is not in use.

5.10.2 Fixed-Speed Jogging.

To jog any axis at a fixed speed, simply specify "x" as a frequency source and "A" as a fixed number. For example, to jog an axis at a speed of 5 units per second in the positive direction, set "x" equal to 5, and "A" equal to 1. To reverse the direction of motion, simply set "A" equal to -1.

Example profile 10 shows how to program this type of move.

Example Profile 10: Fixed-Speed Jogging

Command		Description
4DE10	<CR>	Define profile 10
*4FSX5	<CR>	Load frequency source for X axis of 5 units/sec.
*4LIXF	<CR>	Specify frequency source as input for Aux. Equation
*4LMXN1	<CR>	Load multiplier of 1 for Aux. Equation
*4AEX1	<CR>	Enable Aux. Equation for axis X
*4ED	<CR>	End profile definition
*4GT10	<CR>	Execute profile 10

Note: The stars (*) are the responses from the controller, and the controller is assumed to be addressed at "4".

Note that it is the product "Ax" which determines the speed and direction of the move. For example, the above profile sets "A" to 1 and "x" to 5 to achieve a speed of 5 units per second, but the same result could have been achieved with an "A" of 0.5 and an "x" of 10.

The sign of the product determines the direction of the move. If "Ax" is positive, the axis will jog in the positive direction. If "Ax" is negative, the motion will be in the negative direction.

Finally, remember that it is the "AEab" command (where "a" is the axis and "b" is a 0 or 1) which actually enables the move. In the above example, the move does not start until the "AEX1" command is sent. To stop the axis, send the "AEX0" command or the "HT" command.

5.10.3 Jog Functions Using the Analog Inputs.

Another way to use the Auxiliary Position Equation is to use an internal frequency source as the input ("x"), and an analog input as the multiplier ("A"). This allows you to vary the speed by varying the voltage at the analog channel. This feature is typically used in applications requiring analog joystick jog functions.

You can also use Profile 128 to load an Auxiliary Position Equation multiplier which is a mathematical combination of integer variables. For example, you could load a multiplier which is the sum or difference of two analog inputs. To do this you must program Profile 128 as follows:

1. Define the desired integer variable operations.

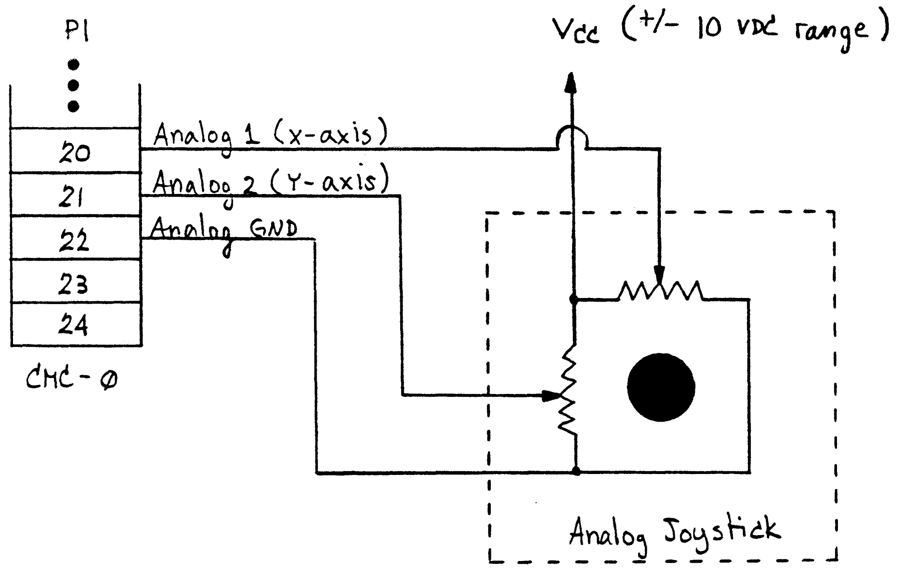
2. Load the result as the multiplier (use the "LMaVd" command).

3. Enable Profile 128 execution with the "UE1" command.

See Section 5.11 for details about integer variable usage.

The example profile on the next page shows how to program a simple analog joystick interface for two axes (an XY table, for example). Figure 5-8 illustrates proper wiring for this application.

Figure 5-8 Analog Joystick Wiring



Example Profile 11: 2-Axis Jogging Using Analog Inputs

Command		Description
4DE11	<CR>	Define profile 11
*4FSX1	<CR>	Load frequency source for X axis of 1 unit/sec.
*4LIXF	<CR>	Specify frequency source as input for Aux. Equation
*4LMXA1	<CR>	Define Analog Channel 1 as X-axis multiplier
*4FSY1	<CR>	Load frequency source for Y axis of 1 unit/sec.
*4LIYF	<CR>	Specify frequency source as input for Aux. Equation
*4LMYA2	<CR>	Define Analog Channel 2 as Y-axis multiplier
*4AEX1	<CR>	Enable Aux. Equation for axis X
*4AEY1	<CR>	Enable Aux. Equation for axis Y
*4ED	<CR>	End profile definition
*4GT11	<CR>	Execute profile 11
*		

Note: The stars (*) are the responses from the controller, and the controller is assumed to be addressed at "4".

Again, note that the product "Ax" determines the speed and direction for each axis. Since the range of "A" is +/- 7.99 (+/- 2,047/256) when using an analog input, be careful to specify an appropriate frequency value which does not cause the axis to move too fast when the analog input signal becomes large.

Also, make sure that each analog channel's parameters (deadband and offset) are initialized correctly before using it in a profile. See Section 4.2.5 for analog channel initialization procedures.

5.10.4 Encoder Tracking with the External Encoder Inputs.

You can also use the Auxiliary Position Equation to "slave" one or more motors to an external encoder. To do this, specify the input to the equation ("x") as one of the external encoder inputs instead of a frequency source. Then set the multiplier ("A") to a specific number.

If you want to track the encoder pulse for pulse, then set the multiplier to 1. However, you can also set up a "gear ratio" between the external encoder and the slave axis by selecting an appropriate value for "A". For instance, if "A" is set to 2, then the slave axis will move 2 pulses for every input pulse. This makes the slave axis run at two times the speed of the external encoder if the number of pulses per revolution is the same for both the encoder and the slave motor. This is essentially equivalent to a master-slave gear ratio of 1:2.

Achieving a desired gear ratio is somewhat more complicated if the number of pulses per revolution is different for the master and slave axes. For example, suppose you have a 1000 pulse/revolution encoder connected to one of the three external encoder channels on the CMC-0 unit (See Section 2.7 for wiring details), and you want to slave a servo motor to this encoder so that it runs at half the speed of the encoder. In addition, suppose the servo motor has an 1800 pulse/revolution encoder attached to it. In summary:

- The "master" is the external encoder (1000 pulses/revolution).
- The "slave" is the servo motor (1800 pulses/revolution).
- The desired master/slave gear ratio is 2:1.

Since the encoder line densities are different, a multiplier ("A") of 0.5 will not achieve the desired gear ratio. Instead, the multiplier must be set to 0.9, so that when one revolution of the master encoder has occurred (1000 pulses), the servo motor will have turned half a revolution (900 pulses).

The example profile which follows shows how to program this example. It assumes the master encoder is connected to external encoder channel 1, the position pulse multiplier is "1", and the slave axis is axis X.

Example Profile 12: Encoder Tracking

Command		Description
4DE12	<CR>	Define profile 12
*4LIXE1	<CR>	Specify encoder channel 1 as input for equation
*4LMXN.9	<CR>	Set multiplier to 0.9
*4AEX1	<CR>	Enable Auxiliary Position Equation for axis X
*4ED	<CR>	End profile definition
*4GT12	<CR>	Execute profile 12
*		

Note: The stars (*) are the responses from the controller, and the controller is assumed to be addressed at "4".

5.11 VARIABLE PARAMETERS

5.11.1 Introduction.

The CMC-0 controller has sixteen integer variables (V0 - VF) which can be used to load motion parameters. V1 through VF are general purpose integer variables, while V0 is reserved for use with the Phase-Locked Loop equation. The parameters which can be loaded by variables are:

- Absolute move
- Incremental move
- Acceleration rate
- Trajectory velocity
- Radius
- Ellipse Ratio
- Frequency source
- Analog channel offset
- Phase-Locked Loop offset
- Phase-Locked Loop cycle length
- Auxiliary Position Equation multiplier

Each of the sixteen variables can be assigned to one of the following:

- 32-bit signed number
- PLL phase error
- PLL phase position
- Axis position
- Analog input
- Boolean variable
- User input or output
- Thumbwheel input
- Other integer variable
- External encoder input (represents velocity)

Variables can also be assigned to arithmetic combinations of two of the above (excluding 32-bit numbers) using the +, -, *, and / math operators.

Once a variable has been assigned to one of the above, it can be used to load one or more profile parameters. For example, variable V1 could be assigned to an analog input, and then loaded as the acceleration rate of an axis. This would allow setting the acceleration rate of the axis with a potentiometer or other analog device.

Note: Integer variable statements are only calculated (updated) when the controller comes across them in a profile it is executing. These statements are not updated on a continuous basis.

However, you can use Profile 128 to load an Auxiliary Position Equation multiplier (the "A" in P = Ax + B) with an integer variable and have this variable updated continuously. Profile 128 is a special profile which can only contain integer variable operations and "LMaVd" commands for the Auxiliary Position Equation. This profile is useful for Phase-Locked Loop applications and for Auxiliary Position Equation applications which require multipliers which are not fixed numbers or single analog inputs. Profile 128 is executed every 5 milliseconds when enabled with the "UE1" command.

For example, Profile 128 could be used to continuously update an Auxiliary Position Equation multiplier with the sum of two or more analog input channels. To do this, simply define Profile 128 with the desired integer variable assignments and the appropriate "LMaVd" command. Then, when the Auxiliary Position Equation is enabled and Profile 128 is enabled with the "UE1" command, the multiplier will be continuously reloaded with the updated integer variable.

5.11.2 Variable Assignment.

Before an integer variable can be used to load a parameter, it must be assigned to a specific function. The assignment command must be in the form:

$$Vd=a\left(\begin{bmatrix} * \\ / \\ + \\ - \end{bmatrix} b\right)$$

where "a" and "b" can be any of the following:

PHE	- Phase-Locked Loop error
PHP	- Phase-Locked Loop position
Ed	- Velocity of external encoder channel (#)
Vd	- Other integer variable
(N)Bd	- Boolean variable ("N" is optional, implies complement)
(N)Id	- User I/O line ("N" is optional, implies complement)
Td	- Thumbwheel input
Ad	- Analog input
Pa	- Axis position

For example, V1=A1+A2 would assign integer variable V1 to the sum of analog input channels one and two.

Note that any operation which produces a value outside of the 32-bit signed range will result in an overflow (or underflow) condition. In these cases, the controller will fault and the fault code will be set to "Arithmetic Overflow". Note also that when performing multiplication, the first operand must be no greater than 16 bits, but the second operand can be as large as 32 bits.

The controller also allows you to assign 32-bit signed numbers to integer variables, but it does not allow you to use fixed numbers in an assignment command involving mathematical combinations of the other allowed functions.

For example, suppose you want thumbwheel digit T1 to be multiplied by ten internally and assigned to integer variable VC (C = 12). Since you cannot say VC=10*T1, you must assign another variable to the number 10, and use this variable in the final variable assignment. The result would look like this:

Command	Description
4VB=10 <CR>	Assign the number 10 to variable B
*4VC=VB*T1 <CR>	Assign 10*T1 to variable C
*	

(#) - When an external encoder Ed is assigned to a variable, the value stored in the variable is the change in the position over the previous 5 millisecond period. This is equivalent to a digital tachometer function.

5.11.3 Parameter Loading.

Once the necessary variables have been assigned, they can be used to load certain parameters. To load a parameter with a variable, simply send the appropriate load command with a "Vd" at the end, where "Vd" is the variable with which you want to load the parameter.

For example, "4VXV1<CR>" would load the trajectory velocity of the XY axis pair with the contents of variable V1.

It is very important to remember that integer variables are interpreted by the controller in the units which the controller uses internally, not the user-specified units. For example, if you use variable V1 to load an absolute move distance, and its contents are 100, then the absolute move parameter will be 100 position pulses, not 100 user-defined length units.

Refer to Table 5-3 below for the variable parameter load commands and the acceptable variable ranges.

Table 5-3 Variable Parameter Load Commands

Parameter Description	Load Command	Min. Variable Value	Max. Variable Value	Units
Absolute Move	AMaVd	0	+/- 8,388,607	pp
Incremental Move	IMaVd	0	+/- 65,535	pp
Acceleration	ACaVd	80,000	6,500,000	pp/s/s
Trajectory Vel.	VXVd	10	655,350	pp/s
	VWVd	10	655,350	pp/s
Radius	RXVd	0	+/- 65,535	pp
	RWVd	0	+/- 65,535	pp
Ellipse Ratio	ELaVd	0	65,535	N/A
Frequency Source	FSaVd	200	51,200	pp/s
Aux. Eqn. Mult.	LMaVd	0	+/- 25,600	N/A
Analog Offset	AOdVd	0	+/- 32,767	N/A
Phase Lock Offset	PHOVd	0	+/- 32,767	pp
Phase Lock Length	PHLVd	2,500	60,000	pp

The example in section 5.11.4 illustrates the use of variable parameters. In it, the controller executes ten circular moves, with the radius and velocity controlled by two potentiometers (analog inputs).

Notice how the integer variable mathematical operations are used to achieve the correct units for loading the parameters.

5.11.4 Example Using Variable Parameters.

Radius: Controlled by 0-10 VDC potentiometer connected to A1.

Velocity: Controlled by 0-10 VDC potentiometer connected to A2.

System Information:

X and Y axes are connected to an XY table. Lead screw pitch is five revolutions per inch (both axes). Encoder line density is 1000 lines per revolution (both axes). Desired unit of length is one inch; speed units are inches per second. Therefore, the Unit Ratio for both axes is 5000 pulses/inch.

Desired speed range: 0 - 5 inches/second (0 - 25,000 pulses/second).

Desired radius range: 0 - 5 inches (0 - 25,000 pulses).

The goal here is to map the 0 - 10 VDC signals into 0 - 25,000 pulses/sec for the speed, and 0 - 25,000 pulses for the radius. Since the analog inputs have a range of +/- 10 VDC and a resolution of 12 bits (+/- 2,047 counts), we want to map 0 - 2,047 counts into 0 - 25,000 pulses (or pulses/sec).

The conversion factors are:

25,000/2,047 = 12.21 pulses per A/D count for radius, and

25,000/2,047 = 12.21 pulses/sec per A/D count for speed.

We can easily use integer variable arithmetic to perform these conversions using the following variable assignments:

```
V1=25000    (Assign V1 to fixed number 25000)
V2=2047     (Assign V2 to fixed number 2047)
V3=A1*V1    (Multiply A1 by 25,000)
V3=V3/V2    (Divide result by 2,047: A1 conversion finished)
V4=A2*V1    (Multiply A2 by 25,000)
V4=V4/V2    (Divide result by 2,047: A2 conversion finished)
```

This assumes, of course, that both analog inputs have already been initialized so that 0 volts in corresponds to 0 A/D counts.

Notice that Table 5-3 on the previous page gives a minimum velocity value of 10 pulses/sec, so it would be impossible to achieve absolute zero velocity during execution of the profile.

The example profile on the next page gives the entire programming sequence for this example.

Example Profile 13: Profile with Variable Parameters

```
Command           Description
 4DE13      <CR>  Define profile 13
*4ACX20     <CR>  Load acceleration rate, X axis
*4ACY20     <CR>  Load acceleration rate, Y axis
*4V1=25000  <CR>  Assign variable V1 to fixed number 25,000
*4V2=2047   <CR>  Assign variable V2 to fixed number 2,047
*4V3=A1*V1  <CR>  Multiply A1 by 25,000
*4V3=V3/V2  <CR>  Divide result by 2,047: A1 conversion finished
*4V4=A2*V1  <CR>  Multiply A2 by 25,000
*4V4=V4/V2  <CR>  Divide result by 2,047: A2 conversion finished
*4RXV3      <CR>  Load XY radius with V3
*4VXV4      <CR>  Load XY velocity with V4
*4CRX       <CR>  Generate circle, XY axes
*4RM9       <CR>  Repeat profile 9 times
*4ED        <CR>  End profile definition
*4GT13      <CR>  Execute profile 13
*
```

Note: The stars (*) are the responses from the controller, and the controller is assumed to be addressed at "4".

5.12 THUMBWHEEL PROGRAMMING

Thumbwheel programming is essentially identical to programming with variable parameters. However, there are a couple of issues which make it somewhat more complex. This section covers the programming issues involved. For information about wiring thumbwheels to the CMC-0, see Section 2.6.

In most cases, thumbwheel inputs must be converted to appropriate units before the parameter can be loaded. For example, suppose you have a four-digit thumbwheel arrangement with a range from 00.00 to 99.99, and suppose these represent absolute move length (resolution = .01 units). Knowing that the CMC-0 interprets all variable parameters in terms of internal units (i.e.position pulses for length), we must convert the thumbwheel reading from units to position pulses prior to loading the absolute move parameter.

Suppose the desired length units are inches, and the Unit Ratio is 5000 position pulses/inch.

The equation we want to implement is:

(10T1 + 1T2 + .1T3 + .01T4) inches * 5000 pulses/inch = X, where T1-T4 are the four thumbwheel inputs.

Therefore, X is equal to (50000T1 + 5000T2 + 500T3 + 50T4) pulses.

It is often a good idea to program the entire thumbwheel conversion process as a single profile so that it can be called whenever a conversion is required. Example profile 14 illustrates the thumbwheel conversion process for the previous example.

Example Profile 14: Thumbwheel Conversion

Command		Description
4DE14	<CR>	Define profile 14
*4V1=10	<CR>	V1 = 10
*4V2=50	<CR>	V2 = 50
*4V3=T1*V1	<CR>	V3 = 10T1
*4V3=V3+T2	<CR>	V3 = 10T1 + T2
*4V3=V3*V1	<CR>	V3 = 100T1 + 10T2
*4V3=V3+T3	<CR>	V3 = 100T1 + 10T2 + T3
*4V3=V3*V1	<CR>	V3 = 1000T1 + 100T2 +10T3
*4V3=V3+T4	<CR>	V3 = 1000T1 + 100T2 + 10T3 + T4
*4V3=V3*V2	<CR>	V3 = 50000T1 + 5000T2 + 500T3 + 50T4
*4ED	<CR>	End profile definition
*		

Note: The stars (*) are the responses from the controller, and the controller is assumed to be addressed at "4".

Now, whenever you need to update the thumbwheels and load them into a parameter, simply execute profile 14, then load the contents of variable V3 into the parameter. Example profile 15 illustrates this for a simple absolute move.

Example Profile 15: Thumbwheel-Based Move

Command		Description
4DE15	<CR>	Define profile 15
*4ACX20	<CR>	Load X axis acceleration rate
*4VX5	<CR>	Load X axis velocity
*4GS14	<CR>	Convert thumbwheels, store in variable V3
*4AMXV3	<CR>	Load V3 contents into absolute move register, axis X
*4RAX	<CR>	Run absolute, X axis
*4ED	<CR>	End profile definition
*4GT15	<CR>	Execute profile 15
*		

Note: The stars (*) are the responses from the controller, and the controller is assumed to be addressed at "4".

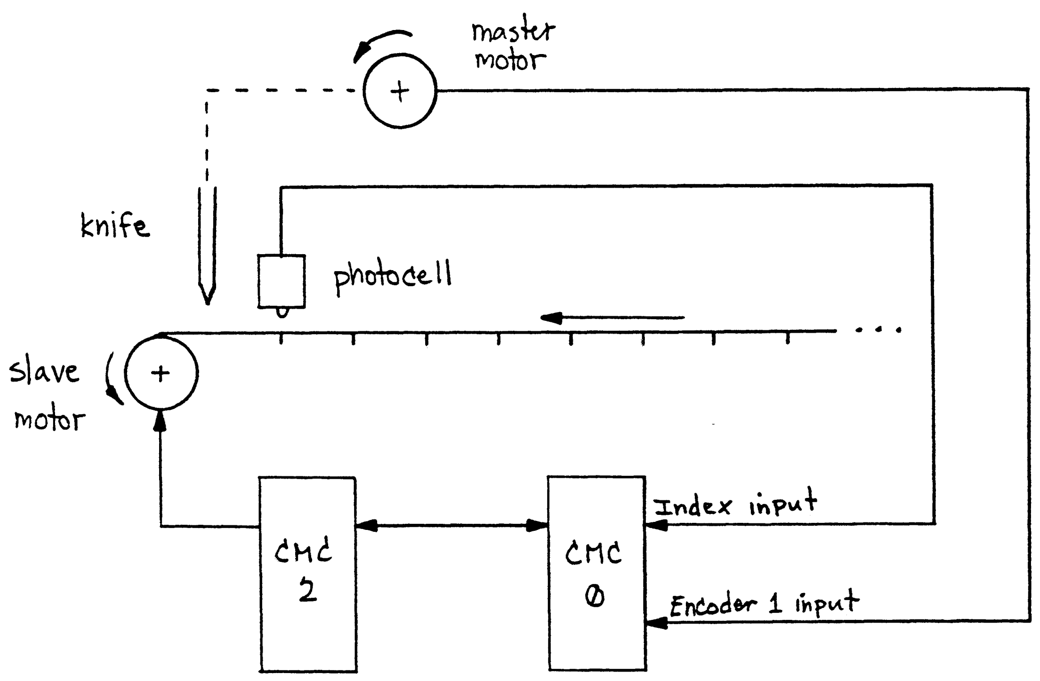
5.13 PHASE-LOCKED LOOP CONTROL

5.13.1 Introduction.

The CMC-0 allows you to phase lock an axis to an external frequency source using the Auxiliary Position Equation and the special phase-locked loop control commands. This feature is typically used in continuous-feed applications requiring the synchronization of registration marks with an external action (i.e. cutting).

Figure 5-9 illustrates a typical setup.

Figure 5-9 Phase-Locked Loop Configuration



The control system consists of the following components:

1. Master motor - generates the frequency source (encoder pulses) which the registration marks are phase locked to.

2. Slave motor - moves in ratio to the master motor such that the registration marks always occur at the same point in the cycle.

3. CMC-0 - controls the ratio between the master and slave motors such that phase lock occurs between the master motor and the registration marks.

4. CMC-2 - accepts control commands from the CMC-0 to move the slave motor.

5. Photocell - detects registration marks. Acts as feedback element for control system.

The following parameter explanations may also be helpful:

<u>PLL Cycle Length</u>

The distance in master position pulses from one registration mark to the next. This distance is also called a "cycle".

<u>PLL Offset</u>

The position (within the bounds of the cycle length) at which the registration mark should always occur. For example, if the phase length is 4000 master position pulses, and you want the registration mark to occur in the middle of each cycle, then the phase offset would be 2000 master position pulses.

<u>PLL Correction Limit</u>

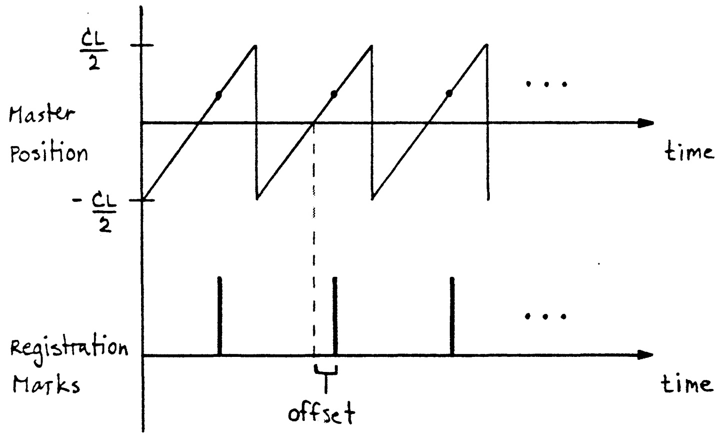
The maximum correction that the PLL equation can make during any one pass through the control loop, i.e. the maximum amount by which V0 can be changed during each loop iteration. It defaults to its maximum value (65,535), and should be left there unless the system cannot handle large instantaneous corrections.

<u>PLL Gain and Zero</u>

The compensation constants which regulate how the closed-loop system behaves. Their selection is very important; improper values will cause unstable behavior (see Section 5.13.3).

Figure 5-10 is a graphical representation of a PLL system functioning properly at constant speed.

Figure 5-10 PLL Timing Diagram



The triangular waveform represents the position of the master encoder, and is reset every time it travels a full cycle length. The pulses at the bottom of the diagram are the registration mark signals. Note that each one occurs at the same point in the cycle, i.e. at the programmed offset value.

5.13.2 Programming and Operation.

The CMC-0 reserves profile 128 for real-time updating of integer variables and Auxiliary Position Equation multipliers. This profile can only contain variable assignment commands and the "LM" command which loads the multiplier for the Auxiliary Position Equation.

When used with PLL control, its purpose is to take the ratio correction from the PLL control equation and load it into the multiplier for the slave axis Auxiliary Position Equation. Because it must continuously make this correction, this profile executes every 5 milliseconds when it is enabled. The ratio correction that it produces is stored in integer variable V0.

The procedure for setting up and running a typical PLL system is as follows:

1. Connect the photocell output to the CMC-0 "index +" input. If the output is single-ended, then connect the CMC-0 "index -" input to the "encoder tie" output of the CMC-0.

2. Connect the master frequency source/encoder to encoder channel 1 of the CMC-0.

3. Load the correct PLL cycle length in master position pulses.

4. Load the appropriate PLL gain and zero values.

5. Jog the slave axis (using the Auxiliary Position Equation) until the photocell detects a registration mark, and then stop. Use the "WTI" command to wait for the registration mark.

6. Jog the master motor to the correct position with respect to the registration mark, then set the master position to zero.

7. Set the PLL offset to zero.

8. Initialize the slave axis Auxiliary Position Equation to track external encoder channel 1, and load the multiplier with the nominal master/slave ratio.

9. Enable profile 128 execution with the "UE1" command.

10. Accelerate the master motor to normal running speed.

At this time, the system should be running in phase lock, where the phase between the master frequency source and the registration marks was determined by the operator in steps 5, 6, and 7 above.

The profiles which accomplish this function are given on the following pages.

Example Profile 16: Profile 128 Definition

```
Command           Description
 4DE128   <CR>    Define profile 128 for PLL operation
*4LMXV0   <CR>    Update slave axis ratio with PLL equation result
*4ED      <CR>    End profile definition
*
```

Example Profile 17: PLL Setup

```
Command            Description
 4DE17     <CR>    Define profile 17
*4PHG260   <CR>    Load PLL gain
*4PHZ245   <CR>    Load PLL zero
*4PHL3000  <CR>    Load PLL cycle length
*4FSX.1    <CR>    Load frequency source for X axis Aux. Posn. Eqn.
*4LIXF     <CR>    Specify frequency source as input
*4UE0      <CR>    Disable profile 128 execution
*4LMXN1    <CR>    Load multiplier for X axis Aux. Posn. Eqn.
*4AEX1     <CR>    Start slave motor jogging
*4WTI      <CR>    Wait for index pulse (registration mark)
*4AEX0     <CR>    Stop slave motor
*4ED       <CR>    End profile definition
*
```

Example Profile 18: PLL Execution

```
Command            Description
 4DE18     <CR>    Define profile 18
*4SPH      <CR>    Zero PLL master position
*4PH00     <CR>    Load offset with present master position (zero)
*4LIXE1    <CR>    X axis Aux. Posn. Eqn. input equals encoder 1
*4V0=51    <CR>    Initialize variable V0 to nom. ratio * 256 (.2 * 256)
*4LMXV0    <CR>    Load nominal master/slave ratio
*4AEX1     <CR>    Enable X axis Aux. Posn. Eqn.
*4UE1      <CR>    Enable profile 128 execution
*4ED       <CR>    End profile definition
*
```

Note that in profile 18, V0 was initialized to the desired nominal ratio times 256. This is because Auxiliary Position Equation multipliers are divided internally by 256 when they are entered by variables. For example, to load a multiplier equivalent to "1" with a user variable, the variable would have to contain 256.

The start-up sequence would then be:

1. Execute profile 17 to jog slave motor to registration mark and initialize PLL gain, zero, and cycle length.

2. Jog master motor to position which is correct with respect to the registration mark.

3. Execute profile 18 to initialize PLL offset and enable PLL operation. At this point, the CMC system is ready to phase lock the registration marks to the master axis.

Note that many variations of the above profiles are possible to accommodate specific system requirements. For example, the thumbwheel inputs can be used with integer variables to load the PLL cycle length and offset. Also, the user I/O on the CMC-0 can be used to allow operator selection between registered and non-registered modes of operation.

5.13.3 Control Constant Selection.

The PLL control constants (gain and zero) determine to a large degree the response of the closed loop control system. The default values (1 for gain, 245 for the zero) will generally be close to the desired values, but in many cases it will be necessary to alter these to achieve optimum performance. With this in mind, here are some general guidelines:

- As the nominal master/slave ratio increases, it will be possible to increase the gain and still maintain stability. If the gain becomes too large, the slave axis may oscillate around the desired velocity or run uncontrollably.

- If the system takes a long time to achieve phase lock after start-up, then increasing the gain or decreasing the zero value will help. Again, making the zero value too small may cause the slave axis to oscillate around the desired velocity. The best system performance will be obtained with the zero in the range of 230 - 255.

Note that the gain term is entered exactly like the gain term for a servo axis (CMC-2). The CMC-0 treats it as a two-byte quantity, and if the first bit in the upper byte is set, it divides the lower byte by 16 and enters the result as the gain value. Therefore, numbers between 0 and 255 are treated exactly as they are entered, e.g. loading 255 results in a gain of 255. However, numbers greater than 255 cause the value in the lower byte to be divided by 16 and then loaded as the gain.

For example, entering a gain of 257 is equivalent to a gain of 1/16, because the first bit in the upper byte is set, and the lower byte contains 1. Similarly, 258 is equivalent to 2/16 or 1/8.

If the default values for the PLL control constants cause the system to oscillate, then try decreasing the gain. If that does not work, then try increasing the zero value. If the default values make the system take a long time to achieve phase lock, then try increasing the gain first. If that does not work, try decreasing the zero value. In many cases you will have to experiment with both constants to achieve optimum performance.

To see how the system is performing in the steady-state, repeatedly load the PLL phase error into an integer variable (e.g. "V1=PHE") and read it (e.g. "V1=?"). To see how quickly it responds to instantaneous changes in phase offset, use the PHO command to change the desired phase offset by about 25 percent or so. This is equivalent to giving the system a step input command.

5.14 SET POINTS

The CMC-0 has two types of set points which are defined in different ways. The sections below describe their functions and programming commands.

5.14.1 Slave Axis Set Points.

Each slave axis has two set point output lines on pins 16 and 17 of connector P1, and each of these can be programmed to be active for up to fifty different position ranges. The commands which are associated with the slave axis set points are given below:

Mnemonic	Action
PTAxxBannn	Load axis setpoint A begin position, range xx
PTAxxEannn	Load axis setpoint A end position, range xx
PTAxxFa	Forget axis setpoint A range xx
PTBxxBannn	Load axis setpoint B begin position, range xx
PTBxxEannn	Load axis setpoint B end position, range xx
PTBxxFa	Forget axis setpoint B range xx

The set point ranges are direction sensitive. Each range is active only when the axis is moving (or was last moving) in the direction from the beginning point to the ending point. The active ranges are defined as follows:

positive motion: $B \leq \text{motor position} < E$

negative motion: $B > \text{motor position} \geq E$

For instance, to make set point "A" of axis X become active at 10 units and inactive at 11 units, issue the sequence

```
"4PTA01BX10 <CR>"
"4PTA01EX11 <CR>"
```

where "4" is the address of the CMC-0. The "01" in both of these commands is the range number associated with the defined "beginning position - ending position" pair. Remember that each line can have fifty ranges associated with it.

A range can also be programmed to start at a larger position and end at a smaller position. For example, the sequence

```
"4PTA02BX11 <CR>"
"4PTA02EX10 <CR>"
```

would cause the second range of axis X set point A to become active at just under 11 units and inactive at just under 10 units.

To change the position pair associated with a particular range, simply redefine it. To forget a set point range altogether, issue the PTAxxFa or PTBxxFa command with the appropriate range and axis specified.

Note that axis set point ranges must be defined in numerical order. For example, you cannot define ranges 1, 2, 3, and then 5. This also applies when you want to forget a range. For example, if you have ranges 1 through 5 defined and you forget range 3, then ranges 4 and 5 will also be forgotten.

5.14.2 CMC-0 Set Points.

The CMC-0 has four set points associated with it which are separate from the axis set points. Although these set points use different I/O lines, they are actually programmed in terms of the axis set points with the commands given below.

Mnemonic	Action
SPAa$\begin{bmatrix}A\\B\end{bmatrix}a\begin{bmatrix}A\\B\end{bmatrix}$	Define CMC-0 set point A as occurrence of two axis set points
SPBa$\begin{bmatrix}A\\B\end{bmatrix}a\begin{bmatrix}A\\B\end{bmatrix}$	Define CMC-0 set point B as occurrence of two axis set points
SPCa$\begin{bmatrix}A\\B\end{bmatrix}a\begin{bmatrix}A\\B\end{bmatrix}$	Define CMC-0 set point C as occurrence of two axis set points
SPDa$\begin{bmatrix}A\\B\end{bmatrix}a\begin{bmatrix}A\\B\end{bmatrix}$	Define CMC-0 set point D as occurrence of two axis set points
SPAF	Forget set point A
SPBF	Forget set point B
SPCF	Forget set point C
SPDF	Forget set point D

These set points are activated upon the occurrence of two axis set points becoming active simultaneously. For example, suppose you have an XY table, and you want to activate CMC-0 set point A when the table reaches position (X,Y) = (10,10). To do this, issue the following sequence of commands:

```
Command                   Description
 4PTA01BX10    <CR>    Define first beginning position, axis X
*4PTA01EX10.1 <CR>    Define first ending position, axis X
*4PTA01BY10    <CR>    Define first beginning position, axis Y
*4PTA01EY10.1 <CR>    Define first ending position, axis Y
*4PTA02BX10.1 <CR>    Define second beginning position, axis X
*4PTA02EX10    <CR>    Define second ending position, axis X
*4PTA02BY10.1 <CR>    Define second beginning position, axis Y
*4PTA02EY10    <CR>    Define second ending position, axis Y
*4SPAXAYA      <CR>    Define CMC-0 set point "A"
*
```

Note: The stars (*) are the responses from the controller, and the controller is assumed to be addressed at "4".

Note in the above example that two ranges were defined for both axis set points so that the CMC-0 set point would become active no matter how the XY table reached the point (10,10).

Finally, note that the CMC-0 set points C and D do not have dedicated outputs assigned to them. If defined, these set points use the user I/O lines B and C, respectively. These I/O lines must be defined as outputs prior to defining these set points.

5.15 TEACH MODE

The CMC-0 allows you to "teach" the controller different positions using the special Teach Mode commands "TCn" and "TMannn". This feature is especially useful in applications requiring an operator to define different positions (Teach Points) for different operations without having to edit the actual profile statements. In most cases it is used in conjunction with a joystick or pushbutton arrangement which allows the operator to jog the motor(s) to the desired location(s) prior to teaching the positions.

There are two steps involved in using this feature: set up, and execution. The correct sequence for setting up this feature is given below:

1. Using the "DRdb" command, define user I/O line "A" as an input, and connect a high-quality, debounced pushbutton to it. This input is used later to tell the controller to memorize the current position.

2. Define user I/O line "C" as an output, and connect an indicator lamp to it. The CMC-0 uses this output to indicate that a point has been memorized.

3. Define the profile which will use the memorized positions. Place appropriate "TMannn" statements wherever a memorized position is required. For example, if you need to teach the controller a point in the XY plane, then you will need a "TMXnnn" statement and a "TMYnnn" statement right after it.

4. When you reach a point in the profile where you need to move to the memorized position, issue the "RAa(a)(a)(a)" command. Since each "TMannn" command loads the Absolute Move register of the associated axis during profile execution, be careful not to change the contents of this register between the "TMannn" commands and their associated "RAa(a)(a)(a)" commands.

   Steps three and four create the "shell" profile which you will eventually execute. The next steps actually teach the controller the correct points.

5. When you are ready to memorize positions, issue the "TCn" command, where "n" is the number of the profile defined in steps three and four above.

6. Using the Auxiliary Position Equation and a joystick or pushbutton arrangement, jog the motor(s) to the correct position(s). When you reach a point you wish the controller to memorize, hit the pushbutton once. The controller will memorize the motor position(s) and flash the lamp attached to I/O line "C".

7. Repeat step six until all positions have been taught.

Note: The "nnn" portion of the "TMannn" command is optional. It allows you to "pre-load" the teach point with a position of your choice. However, this value is replaced by the actual teach point once the operator teaches the controller the desired positions.

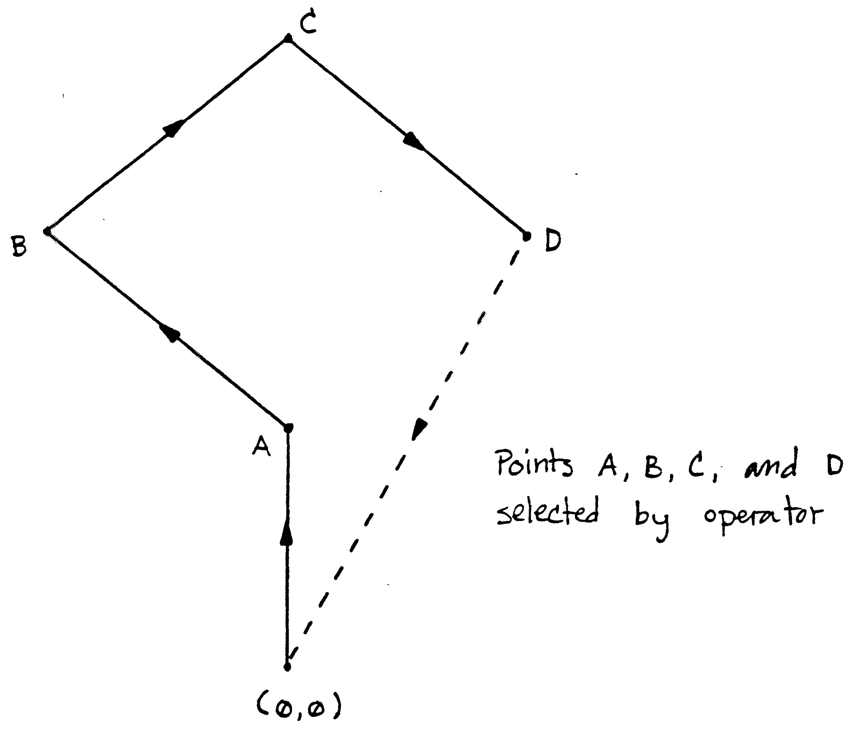
To execute the profile once all positions have been taught, simply issue the "GTn" command, where "n" is the profile number.

As the controller comes upon each "TMannn" statement, it loads the associated Absolute Move register with the memorized position. This position is then used with the next "RAa(a)(a)(a)" command to move the motor(s) to the correct point. This sequence continues until the end of the profile is reached.

An example may help to illustrate this feature. Suppose you have an XY system which is used to position a variety of steel blocks for drilling. Also, suppose that each block will have exactly four holes drilled in it, and that the positions of these four holes change frequently, i.e. each time a new part is designed and introduced into the manufacturing process.

You decide that, rather than have somebody spend time to program a new profile every time a new block is introduced, an operator will be given a template for each block so that he can place the template on the table, jog to the correct positions, and "teach" the CMC system a new drill pattern. A typical pattern is shown in Figure 5-11.

Figure 5-11 Example Teach Mode Contour



The profile must start from (X,Y) = (0,0), run to positions A, B, C, and D, and then return to (X,Y) = (0,0). In addition, the controller must wait for 5 seconds in between moves to allow time for the drilling operation. The first step is to define the joystick profile which allows us to move from one Teach Point to the next. This profile is shown on the next page.

Example Profile 19: Jog Profile for Teach Mode

```
Command              Description
 4DE19     <CR>      Define profile 19
*4FSX.5    <CR>      Load frequency source for X axis of .5 units/sec.
*4LIXF     <CR>      Specify frequency source as input for Aux. Equation
*4LMXA1    <CR>      Define Analog Channel 1 as X-axis multiplier
*4FSY.5    <CR>      Load frequency source for Y axis of .5 units/sec.
*4LIYF     <CR>      Specify frequency source as input for Aux. Equation
*4LMYA2    <CR>      Define Analog Channel 2 as Y-axis multiplier
*4AEX1     <CR>      Enable Aux. Equation for axis X
*4AEY1     <CR>      Enable Aux. Equation for axis Y
*4ED       <CR>      End profile definition
*
```

The next step is to define the profile which contains the four Teach Points. It assumes that I/O lines "A" and "C" have already been initialized for input and output, respectively.

Example Profile 20: Teach Mode Profile

```
Command              Description
 4DE20   <CR>        Define profile 20
*4ACX20  <CR         Load acceleration rate, X axis
*4ACY20  <CR>        Load acceleration rate, Y axis
*4VX3    <CR>        Load trajectory velocity, XY axis pair
*4TMX    <CR>        Load Teach Point A, X axis
*4TMY    <CR>        Load Teach Point A, Y axis
*4RAXY   <CR>        Run to position XY
*4WTMA   <CR>        Wait for all axes to be stopped
*4WTT5   <CR>        Wait for 5 seconds
*4TMX    <CR>        Load Teach Point B, X axis
*4TMY    <CR>        Load Teach Point B, Y axis
*4RAXY   <CR>        Run to position XY
*4WTMA   <CR>        Wait for all axes to be stopped
*4WTT5   <CR>        Wait for 5 seconds
*4TMX    <CR>        Load Teach Point C, X axis
*4TMY    <CR>        Load Teach Point C, Y axis
*4RAXY   <CR>        Run to position XY
*4WTMA   <CR>        Wait for all axes to be stopped
*4WTT5   <CR>        Wait for 5 seconds
*4TMX    <CR>        Load Teach Point D, X axis
*4TMY    <CR>        Load Teach Point D, Y axis
*4RAXY   <CR>        Run to position XY
*4WTMA   <CR>        Wait for all axes to be stopped
*4WTT5   <CR>        Wait for 5 seconds
*4AMX0   <CR>        Load absolute move of 0 for X axis
*4AMY0   <CR>        Load absolute move of 0 for Y axis
*4RAXY   <CR>        Return to (X,Y) = (0,0)
*4ED     <CR>        End profile definition
*
```

It is now simple to teach the controller the desired positions and execute the memorized move. The command sequence:

"4GT19 <CR>"
"4TC20 <CR>"

initializes the Auxiliary Position Equations for the X and Y axes, and waits for the operator to teach the controller the four desired drilling positions. Once these have been taught, issue the "GT20 <CR>" command to execute the memorized drill profile.

5.16 CONDITIONAL PROFILE EXECUTION

The CMC-0 allows the structures listed below for conditional profile execution. Note that these use the following notation:

A - AND function	I - User Input	> - Greater than
O - OR function	V - User Integer Variable	= - Equal to
X - XOR function	B - User Boolean Variable	<> - Not equal to
N - NOT function		

Mnemonic	Action
WTTnn	Wait until timer is elapsed
WTMa	Wait until axis is stopped
WTMA	Wait until all axes are stopped
WTI	Wait until index pulse
$\text{WTE} \left[\begin{array}{l} (\text{N}) \begin{bmatrix}\text{B}\\ \text{I}\end{bmatrix} \text{d} \left(\begin{bmatrix}\text{A}\\ \text{O}\\ \text{X}\end{bmatrix} (\text{N}) \begin{bmatrix}\text{B}\\ \text{I}\end{bmatrix} \text{d} \right) \\ \text{Vd} \begin{bmatrix} > \\ = \\ <> \end{bmatrix} \text{Vd} \end{array} \right]$	Wait until expression is true
$\text{IF} \left[\begin{array}{l} (\text{N}) \begin{bmatrix}\text{B}\\ \text{I}\end{bmatrix} \text{d} \left(\begin{bmatrix}\text{A}\\ \text{O}\\ \text{X}\end{bmatrix} (\text{N}) \begin{bmatrix}\text{B}\\ \text{I}\end{bmatrix} \text{d} \right) \\ \text{Vd} \begin{bmatrix} > \\ = \\ <> \end{bmatrix} \text{Vd} \end{array} \right] \text{GSn}$	If expression true, GOSUB profile n
$\text{IF} \left[\begin{array}{l} (\text{N}) \begin{bmatrix}\text{B}\\ \text{I}\end{bmatrix} \text{d} \left(\begin{bmatrix}\text{A}\\ \text{O}\\ \text{X}\end{bmatrix} (\text{N}) \begin{bmatrix}\text{B}\\ \text{I}\end{bmatrix} \text{d} \right) \\ \text{Vd} \begin{bmatrix} > \\ = \\ <> \end{bmatrix} \text{Vd} \end{array} \right] \text{GTn}$	If expression true, GOTO profile n

The GOTO structure forces the controller to abort the current profile if the condition is satisfied. Note that the "GTn" command is simply an unconditional GOTO statement which causes profile "n" to execute immediately.

The GOSUB structure forces the controller to temporarily leave the current profile, execute the subroutine profile in its entirety, and then return to the current profile. Note that the "GSn" command is simply an unconditional GOSUB statement.

The WAIT (WT) statements cause the controller to suspend profile execution until the wait condition has been met (i.e. timer elapsed, move complete, etc.)

Example:

"IFI9AB3GT5" means "If user input nine is true and user Boolean variable three is true, then abort current profile and execute profile five." This is a conditional GOTO statement.

Example:

"IFV1>V3GS6" means "If user integer variable one is greater than user integer variable three, then execute profile six before continuing with current profile." This is a conditional GOSUB statement.

Example:

"WTENI2ANI3" means "Wait until user input two is not true and user input three is not true, then continue execution of current profile."

An example profile which uses a conditional statement is given on the following page. It is exactly the same as Example Profile 13 from Section 5.11, but it causes the controller to wait for an inactive to active signal transition on user input 5 before executing the move.

Example Profile 21: Conditional Profile Execution

```
Command              Description
 4DE21      <CR>    Define profile 21
*4ACX20     <CR>    Load acceleration rate, X axis
*4ACY20     <CR>    Load acceleration rate, Y axis
*4V1=25000  <CR>    Assign variable V1 to fixed number 25,000
*4V2=2047   <CR>    Assign variable V2 to fixed number 2,047
*4V3=A1*V1  <CR>    Multiply A1 by 25,000
*4V3=V3/V2  <CR>    Divide result by 2,047 - A1 conversion finished
*4V4=A2*V1  <CR>    Multiply A2 by 25,000
*4V4=V4/V2  <CR>    Divide result by 2,047 - A2 conversion finished
*4RXV3      <CR>    Load XY radius with V3
*4VXV4      <CR>    Load XY velocity with V4
*4WTENI5    <CR>    Make sure input 5 is not true
*4WTEI5     <CR>    Wait for input 5 to be true
*4CRX       <CR>    Generate circle, XY axes
*4RM9       <CR>    Repeat profile 9 times
*4ED        <CR>    End profile definition
*4GT21      <CR>    Execute profile 21
*
```

Note: The stars (*) are the responses from the controller, and the controller is assumed to be addressed at "4".

Note that conditional statements can be placed anywhere in a profile that they are needed. Make sure, however, that all user inputs and integer variables have been defined correctly before using them in a conditional statement. This example assumes that user I/O line 5 has already been initialized as an input (I/O lines cannot be initialized inside of profiles).

Also note that the use of both integer variables and either Boolean variables or user inputs in the same conditional expression is not allowed. Also, if you wish to complement only one logical operand in an expression, then it must be the first.

5.17 PROFILE EDITING

The following three commands are used for editing profiles:

Mnemonic	Action
X	Scroll to next line in profile
DEL	Delete profile line
!	Abort edit mode

To get into an existing profile, simply send the "DEn" command, where "n" is the profile number. The controller will then return the next line in the profile. If the profile is "empty", the controller will immediately return "ED", for "End of Definition".

Once you are inside a profile definition, you can scroll from line to line by using the "X" command. Each time you use this command the controller returns the next line in the profile.

The general method for changing a profile is:

1. Get into the profile definition with the "DEn" command.

2. Use the "X" command to scroll to the first statement you wish to change, or to the first line that you wish to insert a statement in front of.

3. If you want to insert a statement at this point, simply type it in. If you want to delete the statement completely, type "DEL". If you wish to put another statement in its place, then type it in after deleting the old line.

4. Repeat 2 and 3 as required.

5. Use the "!" command to abort the edit mode.

Note that the "!" command can also be used to abort individual commands when communicating with the CMC-0. If a "!" appears as any character other than the first character after the unit address, the entire command line is ignored.

SECTION 6.0

MAINTENANCE

WARNING: Voltages up to 400V can be present on some of the components inside the CMC controllers. DO NOT attempt to repair these units with power applied.

6.1 TROUBLESHOOTING

This section is intended to help the user diagnose the majority of problems which might occur with the CMC controllers. The first part of the section is devoted to general problems which may occur during CMC system operation. The remainder of the section discusses those problems which cause the controller(s) to fault. If the problem is still unresolved after following the appropriate procedure, contact Whedco Customer Service for assistance.

6.1.1 System Problems.

CONTROLLERS DO NOTHING

Symptoms
When power is applied to the system, the "fault" LED does not light on the CMC-0 unit and/or the slave units, and the motors have no holding torque. There is no +5 or +12 VDC on the suspect controller(s).

Solution
Check the AC line fuse and power input wiring (see Section 2.2 and/or 3.2). If the fuse is good and the unit is getting its proper supply voltage, then contact Customer Service.

CMC-0 WILL NOT COMMUNICATE WITH HOST COMPUTER OR TERMINAL

Symptoms
Above problem does not exist. The "xmit" LED on the CMC-0 does not blink when the unit is addressed in "echo mode". The unit does not echo characters as they are typed in, and it does not return a response after a command is sent.

Solution
Check the serial wiring and serial format of your terminal or host computer to see that it conforms to the format which the CMC-0 unit expects (see Section 2.3). Check the settings of CMC-0 DIP switch SW2 for correct address, communication format (echo, non-echo), baud rate, and serial format (RS-232, etc.). (See Section 2.10 for settings.) Also, review the communication format given in Section 4.1 and note that a <CR> may be required to clear the serial channel if incorrect characters were inadvertently typed in.

CMC-0 LOSES MEMORY

Symptoms
On power-up, the Memory Lost flag is set in the status register, all previously saved profiles are lost, and the system's parameters are back at their default values.

Solution
The CMC-0 should not have this problem under normal operation unless a Cold Boot (CB) command is issued. If a Cold Boot command was not sent and the unit still loses memory, return the unit for repairs.

PARALLEL INPUTS DO NOT FUNCTION ON CMC-0 UNIT AND/OR SLAVE UNIT(S)

Symptoms
CMC-0 does not recognize user-defined parallel inputs. Slave units do not recognize parallel profile inputs and/or Home and Overtravel inputs.

Solution
Refer to Appendix D, Module Configuration, to see if the unit uses sourcing or sinking I/O. For sourcing units, the active state of an input line (except for Overtravel inputs) is a high voltage level. Make certain the inputs are switched from below 1.6 volts to at least 3.4 volts. For sinking inputs, the active state of an input line (except for Overtravel inputs) is a low voltage level. Make certain the inputs are switched from at least 3.4 volts to below 1.6 volts.

For systems using a Programmable Logic Controller (PLC) as the parallel interface, make certain the DC signal ground of the PLC output circuit is directly connected to the DC signal ground of the CMC controller(s).

Finally, make certain that the "V ext" inputs of the CMC controllers are connected to an appropriate external power supply (5 - 24 VDC).

MOTOR HAS INSUFFICIENT TORQUE

Symptoms
The motor cannot keep the load from moving when at rest and/or the motor cannot accelerate the load.

Solution
Check the setting of the current switches on DIP switch SW1 of the slave axis controller (see Section 3.8). The current should be set (at most) for the maximum continuous current rating of the motor. If the current is set to the maximum, check the motor sizing calculations. Also, if the motor is a stepping motor try disabling the Power Save feature if it is used.

MOTOR TURNS IN WRONG DIRECTION

Symptoms
"fwd" and "rev" LEDs on slave axis controller correspond to direction of command given, but motor turns in the opposite direction to that desired.

Solution (servos)

Reverse the connections to the motor armature, the tachometer (if used) and the encoder channels A and B. Note that the motor, tachometer, and encoder channels must all be reversed or the motor operation will become unstable.

Solution (steppers)

Reverse the connection of one of the motor windings, and reverse encoder channels A and B (if used).

STEPPING MOTOR LOSES STEPS

Symptoms
Although motor has sufficient holding torque, the motor is jerky during low speed operation.

Solution
This is almost always due to operating the motor at or near its resonant frequency. To minimize or eliminate the effects of mechanical resonance you should:

- Increase the load on the motor (add mechanical damping),

- Select a smaller microstep size, or

- Always operate the motor above its resonant frequency.

6.1.2 Fault Conditions.

If the "fault" LEDs of the CMC-0 unit and/or slave units are lit, then a fault condition exists. To determine the cause of the fault, use the "FC" command and/or the "FCa" command, where "a" is the axis letter of the faulted axis.

<u>CMC-0 Faults</u>

The FC command returns the "fault code" of the CMC-0 master unit. If the controller is in the "echo" communication mode, the response will be in English, i.e. "Power Failure". Responses to the FC command are listed below:

"Unit Functional"	Indicates that the system is operational. That is, the CMC-0 and all slave axes are not faulted.
"Power Failure"	Indicates that power has been cycled to the system. This is the normal power-up fault condition.
"Axis Initialize"	Indicates that an axis was initialized with the "ITa" command. This command always causes the CMC-0 to fault for safety reasons.
"Arithmetic Overflow"	Indicates that an illegal arithmetic operation was executed during an integer variable operation. For example, a divide by zero operation would cause this fault. See Section 5.11.2.
"X (Y, W, Z) Faulted"	Indicates that an axis is faulted. Use FCa command to determine cause of fault.
"X (Y, W, Z) Not Responding"	Indicates that the communication link between the slave axis controller and the CMC-0 has been lost or improperly wired.

If the CMC-0 unit is in the "non-echo" communication mode, the reponse to the FC command will be a number. If none of the axis controllers are faulted, the number returned will be one of the following:

0	- Unit Functional
256	- Power Failure
512	- Axis Initialized
1024	- Arithmetic Overflow

If there is a problem with one or more of the slave axes, the number returned by the FC command will be one of the numbers listed above plus the number formed by the following byte:

```
  Z   W   Y   X | Z   W   Y   X
---------------------------------
| a | a | a | a | b | b | b | b |
---------------------------------
```

where:

"b" = 0 indicates the axis is functional.
"b" = 1 indicates the axis is not functional.

"a" = 0 indicates the axis is not communicating with the CMC-0.
"a" = 1 indicates the axis is faulted.

Servo Axis Faults

If a servo axis is faulted you can determine the cause by sending the "FCa" command, where "a" is the axis letter. If the CMC-0 is in the "echo" communication mode, the response will be in English. If it is in the "non-echo" mode the response will be the appropriate fault code number. The English responses and their associated fault code numbers are given below:

Code 0 - "Power Failure"

This is the response if the most recent fault was loss of power to the unit. It is the normal power-up fault condition.

Code 2 - "Over-temperature"

If the unit persistently faults due to getting too hot, some means of cooling the unit will have to be provided. However, if the unit is cool and still faults due to over-temperature, contact Whedco Customer Service.

Code 3 - "Excessive Following Error"

This fault is by far the most common. It can be caused by improper control constants (see Section 4.2.2), improper encoder wiring (see Section 3.3), improper tachometer wiring (see Section 3.5), improper tachometer voltage DIP switch settings (see Section 3.8.2), improper current DIP switch settings (see Section 3.8.2), or an undersized motor.

Code 4 - "Encoder Lost"

This fault occurs when the controller is trying to move the motor but sees absolutely no change in actual position. It can be caused by a damaged or improperly wired encoder, or a broken encoder lead.

Code 6 - "Lost Enable"

The enable input went inactive.

Code 7 - "Position Register Overflow"

The motor was commanded to go to a position outside of the position register's range. Use the "WRab" command to enable position register wrapping if this is a problem.

Code 8 - "Unit Functional"

No fault exists.

Stepper Axis Faults

If a stepper axis is faulted you can determine the cause by sending the "FCa" command, where "a" is the axis letter. If the CMC-0 is in the "echo" communication mode, the response will be in English. If it is in the "non-echo" mode the response will be the appropriate fault code number. The English responses and their associated fault code numbers are given below:

Code 0 - "Power Failure"

This is the response if the most recent fault was loss of power to the unit. It is the normal power-up fault condition.

Code 5 - "Deadband Exceeded"

This occurs if the "Deadband Fault" toggle is set to "1" and the user-specified deadband is exceeded.

Code 6 - "Lost Enable"

The enable input went inactive.

Code 7 - "Position Register Overflow"

The motor was commanded to go to a position outside of the position register's range. You may need to set the axis' "WR" toggle to "1".

Code 8 - "Unit Functional"

No fault exists.

Code 9 - "Over-current"

This fault is usually caused by a short-circuit in the motor wiring. Check the motor wiring first, then try this test: (Disconnect AC power first.)

- Disconnect the motor from the unit.
- Power up the unit and send a Warm Boot to the CMC-0.

If the Warm Boot gets rid of the fault condition, the problem lies in the motor itself or in the wiring. If not, contact Whedco Customer Service.

Code A - "Motor Stalled"

This fault occurs when encoder feedback indicates that the motor is not following the pulse train. It can be caused by loss of encoder feedback, too high an acceleration rate or speed, or an improperly wired encoder.

6.2 FIELD REPLACEMENT

All CMC controllers are mechanically interchangeable with other CMC controllers of the same model number. In addition, the terminal blocks detach from the unit with the field wiring intact. For these reasons, field replacement is very simple. However, when replacing a unit, make sure all field adjustments (i.e. DIP switch settings) are the same as the unit being replaced. It is especially important that the current settings of the axis controllers not exceed the maximum rating for the motor in order to avoid motor damage.

6.3 SERVICE

If a Whedco product malfunctions, it may be returned using the following procedure:

a) Obtain a Return Merchandise Authorization (RMA) number by calling the Customer Service Department at 313/665-5473. When packing the unit for shipment, enclose a purchase order for the repair with a description of the problem and the RMA number.

b) Ship to (freight prepaid):

Service Department
WHEDCO INCORPORATED

4750 Venture Drive
Ann Arbor, Michigan 48108-9559

Note that products returned to Whedco without conforming to the procedure outlined above will not be accepted for repair.

All repairs are shipped F.O.B. Ann Arbor, Michigan. Products meeting the criteria for warranty service will be repaired or replaced free of charge. Products not covered by the warranty will be repaired for a time and materials charge against the customer's purchase order, not to exceed the price of a new unit. Repairs are warranted for thirty (30) days from date of shipment from Whedco. The repair warranty does not apply to new or different problems and/or failures which may occur.

Unless otherwise requested, repaired products will be shipped prepaid via United Parcel Service (UPS) ground service. Other methods of shipment such as UPS Blue Label, Federal Express, and Emery Air Freight can be arranged but are at the expense of the owner. On-site repair is available at a daily rate plus all travel expenses portal-to-portal from Whedco Incorporated.

6.4 WARRANTY

Whedco Incorporated warrants its products for one (1) year from the date of original shipment. During the warranty period, Whedco will repair or, at its option, replace without charge, any product returned to Whedco for repair. This warranty does not apply to any product which has been subject to misuse, accident, improper installation, and/or improper application, nor does it apply to any product which has been repaired or altered by anyone other than a Whedco authorized service representative. Whedco shall remain the sole arbitrator in all warranty claims.

There are no warranties which extend beyond those herein specifically given. In no event shall Whedco be liable to the purchaser for loss of use, profit, or consequential damages, or damages of any kind, including, but not limited to accidental loss or damage to other equipment, whether or not said equipment was properly used.

This warranty is in lieu of any other warranty, expressed, implied, or statutory, including, without limitation, any implied warranty of merchantibility or fitness for a particular purpose. No amendment of this warranty may be effected except in writing by a duly authorized representative of Whedco Incorporated.

APPENDIX A

SPECIFICATIONS

SPECIFICATIONS

A.1 PERFORMANCE

Table A-1 lists performance specifications for the motion profile parameters. Note that "u" stands for length units, which are implicitly defined by the Unit Ratio. Note also that in some cases the minimum, maximum, and resolution values depend on the Unit Ratio.

For example, if the Unit Ratio equals 1 pulse/unit, then the velocity range is 10 to 655,350 units/sec, and the resolution is 1 unit/sec. However, if the Unit Ratio equals 10,000 pulses/unit, then the velocity range is .01 to 65.53 units/sec, and the resolution is .01 units/sec.

Table A-1 Performance Specifications

PARAMETER	RANGE	UNITS
<u>UNIT RATIO</u>		
Minimum	1	pulse/unit
Maximum	10,000	pulses/unit
Resolution	1	pulse/unit
<u>ACCELERATION</u>		
Minimum	8 - 80,000	u/s/s
Maximum	650 - 6,500,000	u/s/s
Resolution	1 - 100	u/s/s
<u>VELOCITY</u>		
Minimum	.01 - 10	u/s
Maximum	65.53 - 655,350	u/s
Resolution	.01 - 10	u/s
<u>ABSOLUTE MOVE</u>		
Minimum	.0001 - 1	u
Maximum	838.8607 - 8,388,607	u
Resolution	.0001 - 1	u
<u>INCREMENTAL MOVE</u>		
Minimum	.0001 - 1	u
Maximum	6.5535 - 65,535	u
Resolution	.0001 - 1	u
<u>ABSOLUTE POSITION</u>		
Minimum	.0001 - 1	u
Maximum	838.8607 - 8,388,607	u
Resolution	.0001 - 1	u
<u>RADIUS</u>		
Minimum	0	u
Maximum	6.5535 - 65,535	u
Resolution	.0001 - 1	u

Table A-1 Performance Specifications (cont'd)

PARAMETER	RANGE	UNITS
<u>WAIT TIME</u>		
Minimum	0	s
Maximum	655.35	s
Resolution	.01	s
<u>FREQUENCY SOURCE</u>		
Minimum	.02 - 200	u/s
Maximum	5.12 - 51,200	u/s
Resolution	.02 - 200	u/s
<u>DEADBAND</u>		
Minimum	0	pulses
Maximum	32,767	pulses
Resolution	1	pulse
<u>POSITION PULSE</u>		
Minimum	50	pulses/rev
Maximum	32,000	pulses/rev
Resolution	1	pulse/rev

A.2 ELECTRICAL

TIMING

Timing specifications do not include serial transmission time.

Command Execution:

2.0 milliseconds average

Absolute Position Register read:

5.0 milliseconds minimum
10.0 milliseconds maximum

CMC-0 CONNECTOR PIN-OUTS

All signal directions are referenced to the CMC unit.

P1 CONNECTOR

	Pin	Mnemonic	Signal Flow	Description
	1	Ext. V+	in	External supply for outputs
User I/O	2	I/O 1	user defined	User defined I/O pin #1
	3	I/O 2	user defined	User defined I/O pin #2
	4	I/O 3	user defined	User defined I/O pin #3
	5	I/O 4	user defined	User defined I/O pin #4
	6	I/O 5	user defined	User defined I/O pin #5
	7	I/O 6	user defined	User defined I/O pin #6
	8	I/O 7	user defined	User defined I/O pin #7
	9	I/O 8	user defined	User defined I/O pin #8
	10	I/O 9	user defined	User defined I/O pin #9
	11	I/O 10	user defined	User defined I/O pin #10
	12	I/O 11	user defined	User defined I/O pin #11
	13	I/O 12	user defined	User defined I/O pin #12
	14	GND	return	Signal gnd. and DC common
Dedicated Outputs	15	Set Pt. A	out	Setpoint A output
	16	Set Pt. B	out	Setpoint B output
	17	Busy	out	Controller "busy" output
	18	Fault	out	Controller "faulted" output
	19	GND	return	Signal gnd. and DC common
Analog I/O	20	Analog 1	in	Analog input #1
	21	Analog 2	in	Analog input #2
	22	Analog GND	return	Analog signal ground
	23	Analog 3	in	Analog input #3
	24	Analog 4	in	Analog input #4

P2 CONNECTOR

	Pin	Mnemonic	Signal Flow	Description
Master/Slave	1	Link A	in/out	Master/Slave Communication
RS-485 Comm.	2	Link B	in/out	Master/Slave Communication
	3	GND	return	Signal gnd. and DC common
	4	Chnl. 1 A+	in	Encoder 1 Channel A+
Encoder 1	5	Chnl. 1 A-	in	Encoder 1 Channel A-
	6	Chnl. 1 B+	in	Encoder 1 Channel B+
	7	Chnl. 1 B-	in	Encoder 1 Channel B-
External	8	Index +	in	External Index input +
Index	9	Index -	in	External Index input -
	10	Chnl. 2 A+	in	Encoder 2 Channel A+
Encoder 2	11	Chnl. 2 A-	in	Encoder 2 Channel A-
	12	Chnl. 2 B+	in	Encoder 2 Channel B+
	13	Chnl. 2 B-	in	Encoder 2 Channel B-
	14	Chnl. 3 A+	in	Encoder 3 Channel A+
Encoder 3	15	Chnl. 3 A-	in	Encoder 3 Channel A-
	16	Chnl. 3 B+	in	Encoder 3 Channel B+
	17	Chnl. 3 B-	in	Encoder 4 Channel B-
	18	Encoder Tie	out	Encoder - input tie
DC Power	19	+5 VDC	out	5 volt supply, 1 amp max.
outputs	20	+12 VDC	out	12 volt supply, .5 amp max.
	21	GND	return	Signal gnd. and DC common
	22	AC GND	return	AC power ground
AC Power	23	AC -	in	AC power -
inputs	24	AC +	in	AC power +

P3 CONNECTOR
(9-pin RS-232/422/485 connector)

	Pin	Mnemonic	Signal Flow	Description
	1	TxA	out	Transmit A output line
	2	TxB	out	Transmit B output line
	3	RxA'	in	Receive A input line
User Serial	4	RxB'	in	Receive B input line
Comm.	5	Ground	return	Signal gnd. and DC common
	6	Sync	in/out	Sync. line
	7	-	-	No connection
	8	-	-	No connection
	9	-	-	No connection

CMC-1 AND CMC-2 CONNECTOR PIN-OUTS

All signal directions are referenced to the CMC units.

P1 CONNECTOR

	Pin	Mnemonic	Signal Flow	Description
Ext. VDC in	1	External V+	in	Ext. supply for outputs
	2	One	in	Profile line one
	3	Two	in	Profile line two
	4	Three	in	Profile line three
Parallel	5	Four	in	Profile line four
Profile	6	Five	in	Profile line five
Execution	7	Six	in	Profile line six
	8	GND	return	Signal gnd. and DC common
	9	t-one	in	Profile Line seven
	10	t-two	in	Profile Line eight
	11	Home	in	Home signal input
	12	+Overtravel	in	Positive O.T. input
Inputs	13	-Overtravel	in	Negative O.T. input
	14	Enable	in	Enable power amplifier
	15	GND	return	Signal gnd. and DC common
	16	Set Point A	out	Set Point A output
	17	Set Point B	out	Set Point B output
Outputs	18	Busy	out	Axis Busy output
	19	Fault	out	Axis Fault output
	20	GND	return	Signal gnd. and DC common
	21	GND (CMC-1)	return	Signal gnd. and DC common
Control Signals	21	Tach + (CMC-2 w/amp)	in	Tachometer signal input
	21	Motor Command (CMC-2 w/out amp)	out	Motor command output to servo amplifier
	22	GND	return	Signal gnd. and DC common

P2 CONNECTOR

	Pin	Mnemonic	Signal Flow	Description
	1	Tx A	out	Transmit A output line
	2	Tx B	out	Transmit B output line
Master/Slave	3	Rx A'	in	Receive A' input line
data link	4	Rx B'	in	Receive B' input line
	5	GND	return	Signal gnd. and DC common
	6	SYNC	in/out	Sync line of slaves
	7	Chnl. A+	in	Encoder channel A+ input
	8	Chnl. A-	in	Encoder channel A- input
	9	Chnl. B+	in	Encoder channel B+ input
	10	Chnl. B-	in	Encoder channel B- input
Encoder	11	Index+	in	Encoder index + input
Connections	12	Index-	in	Encoder index - input
	13	Encoder Tie	out	Encoder - input tie
	14	+5 VDC	out	+5 VDC encoder supply
	15	+12 VDC	out	+12 VDC encoder supply
	16	GND	return	Signal gnd. and DC common
	17	Coil 1 +	out	Motor coil 1 connection
Motor	18	Coil 1 -	out	Motor coil 1 connection
(CMC-1)	19	Coil 2 +	out	Motor coil 2 connection
	20	Coil 2 -	out	Motor coil 2 connection
	21	Motor GND	return	Motor ground connection
	17	Motor +	out	Motor positive connection
Motor	18	Motor +	out	Internally connected to 17
(CMC-2	19	Motor -	out	Motor negative connection
with amp)	20	Motor -	out	Internally connected to 19
	21	Motor GND	return	Motor ground connection
	17	not used		
Motor	18	not used		
(CMC-2	19	not used		
w/out amp)	20	not used		
	21	not used		
	22	AC GND	return	AC power ground
AC Power	23	AC -	in	AC power -
inputs	24	AC +	in	AC power +

Power Supply Requirements:

Input voltage: 90-130 VAC, 50-440 Hz for models so rated, or
180-260 VAC, 50-440 Hz for models so rated.

Input Current: (max)

CMC-0 Master Controllers	- 1.0 amp ;	or 0.5 amps
CMC-1 Stepping Motor Controllers	- 5.0 amps;	or 2.5 amps
CMC-2 Servo Motor Controllers	- 10.0 amps;	or 5.0 amps

Serial I/O Requirements (programming terminal to CMC-0 Master unit):

RS-232C, RS-422, and RS-485 compatible. Baud rate selectable to 1200, 4800, 9600, or 38400 baud.

Encoder Compatibility:

Two-channel incremental, sine or square wave output
Outputs single-ended or differential
Output voltage +/-5, +/-12, +/-15 VDC
Maximum encoder frequency: 500 kHz. Position pulse frequency may be up to 1 MHz. CMC-0 encoder inputs also configurable for pulse/direction inputs.

Parallel Inputs and Outputs, CMC-0, CMC-1, and CMC-2 units:

Outputs - 24 VDC (max), 100 mA current (maximum for CMC-0)
200 mA current (maximum for CMC-1 and CMC-2)

Inputs - 24 VDC (max), 2k resistive pull-up or pull-down depending on model.

Motor Drive Outputs:	CONTROLLER TYPE	
	STEPPING (CMC-1)	SERVO (CMC-2)
Continuous Motor Current		
Minimum	0.33 A	0.47 A
Maximum	5.00 A	7.00 A
Resolution	0.33 A	0.47 A
Motor Excitation Voltage: +/- 5%, factory set	+65 VDC	+65 VDC
Motor Power:		
Maximum	325 W	455 W
Power Save Condition	1/3 programmed current	N/A
Peak Motor Current	100% of continuous	2X cont. for 1s
Minimum Load Inductance:	Not Specified	0
Switching Frequency:	40 kHz	50 kHz

Tachometer Input for servo amplifier (optional):
Impedance: 27k
Max output voltage: 3-65 VDC

A.3 ENVIRONMENTAL (all units)

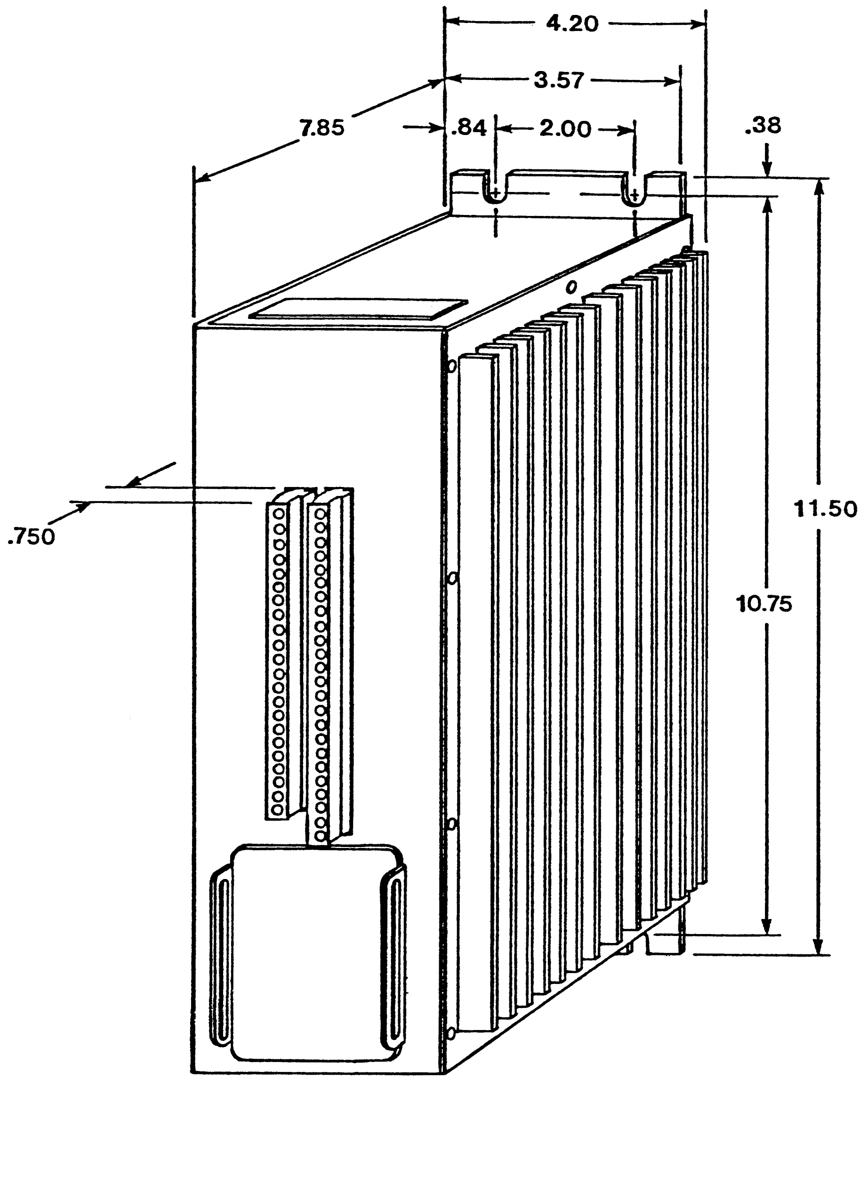
Operating Temperature: 0-55 degrees Celsius, free-air ambient
Storage Temperature: -40-80 degrees Celsius

A.4 MECHANICAL

Weight: Approx. 8 pounds for CMC-1 and CMC-2 with amplifier
Approx. 6 pounds for CMC-0 and CMC-2 without amplifier

Dimensions: See figure A-1
(CMC-0 units and CMC-2 units without amplifier do not have heatsink)

Figure A-1 Unit Dimensions



APPENDIX B

SUMMARY OF COMMANDS

SUMMARY OF COMMANDS

All programming of CMC systems is accomplished through the CMC-0 master unit. The command set can be broken down into the following categories:

1. System commands
2. Axis Initialization commands
3. Movement and Profile Definition commands
4. Conditional Profile Execution and Branching commands
5. Auxiliary Position Equation commands
6. Phase-Locked Loop commands
7. I/O and Variable commands
8. Status and Position Query commands
9. Editing commands

A brief description of all command types is given below. In the descriptions, the following notation applies:

a	- denotes axis W, X, Y, or Z
b	- denotes a binary digit, 0 or 1
d	- denotes a hexadecimal digit, 0-F
V	- denotes an integer variable
n	- denotes an 8-bit number
nn	- denotes a 16-bit number
nnn	- denotes a 24-bit number
nnnn	- denotes a 32-bit number

In addition, brackets indicate that the programmer must choose one of the items within the brackets. Parentheses indicate optional items.

System commands: (Not allowed in defined profiles.)

Numbers in parentheses after the command description indicate sections where more detailed information is available about the command.

Mnemonic	Action	
CB	Clear all profiles, reset parameters to default values	
WB	Clear faults, leave profiles and parameters intact	(4.2.6)
PEb	Enable/disable parallel profile execution	(4.6)
EB	Empty command buffer	
MS	Upload memory from one CMC-0 to another	(4.8)
ML	Download memory to CMC-0 from another	(4.8)
DGb	Enable/disable diagnostic routine	(4.5)
?	Returns parameter value	(4.4.1)

Axis initialization commands: (Not allowed in defined profiles.)

These commands initialize the individual axes with user-specified control constants. Each axis used must be initialized at least once prior to executing moves. These commands are described completely in Sections 4.2.1 and 4.2.2.

(1) - used with CMC-1 Stepping Motor Controllers only.
(2) - used with CMC-2 Servo Motor Controllers only.

Mnemonic		Action	
ITa		Initialize axis with current control constants	(4.2.3)
URann		Load unit ratio	
NOannn		Load negative software O.T.	
POannn		Load positive software O.T.	
WRab		Enable/disable position register wraparound	
LDann	(2)	Load encoder line density	
GNan	(2)	Load controller gain	
ZRan	(2)	Load controller zero	
PLan	(2)	Load controller pole	
FFan	(2)	Load feedforward constant	
DBann		Load deadband constant	
IBann	(2)	Load integration band	
PBann	(2)	Load position band	
ERann	(1)	Load encoder ratio	
DFab	(1)	Enable/disable fault on deadband error	
PCab	(1)	Enable/disable position correction	
PSab	(1)	Enable/disable power-save feature	
CTan	(1)	Load correction time constant	

Movement and profile definition commands:

These commands are used to load motion parameters and execute moves. In general, they can be executed interactively or stored in a profile for later execution. Numbers in parentheses after the command description indicate sections where more detailed information is available about the command.

Mnemonic	Action	
DEn	Begin profile definition	
ED	End profile definition	
UEb	Enable/disable profile 128 execution	(5.11.1)
RMnn	Repeat statements from start of profile (n times)	
GTn	Unconditional GOTO profile n	(5.15)
GSn	Unconditional GOSUB profile n	(5.15)
HT	Halt all axes	
STZa	Immediately set at zero	(5.5)
SHZa	Find home limit switch, then set at zero	(5.5)
SIZa	Find index, then set at zero	(5.5)
SBZa	Find both, then set at zero	(5.5)
STPannn	Immediately set at position nnn	(5.5)
SHPannn	Find home limit switch, then set at position nnn	(5.5)
SIPannn	Find index, then set at position nnn	(5.5)
SBPannn	Find both, then set at position nnn	(5.5)

Movement and profile definition commands: (cont'd)

Mnemonic	Action	
AMa$\begin{bmatrix}\text{nnn}\\\text{Vd}\end{bmatrix}$	Load absolute move register	(5.3)
IMa$\begin{bmatrix}\text{nn}\\\text{Vd}\end{bmatrix}$	Load incremental move register	(5.3)
ACa$\begin{bmatrix}\text{nn}\\\text{Vd}\end{bmatrix}$	Load accel/decel rate	(5.3)
VX$\begin{bmatrix}\text{nn}\\\text{Vd}\end{bmatrix}$	Load trajectory velocity for XY move	(5.3)
VW$\begin{bmatrix}\text{nn}\\\text{Vd}\end{bmatrix}$	Load trajectory velocity for WZ move	(5.3)
VOXnn	Load velocity offset for XY move	(5.9)
VOWnn	Load velocity offset for WZ move	(5.9)
RX$\begin{bmatrix}\text{nn}\\\text{Vd}\end{bmatrix}$	Load radius for XY move	(5.3)
RW$\begin{bmatrix}\text{nn}\\\text{Vd}\end{bmatrix}$	Load radius for WZ move	(5.3)
IXnn	Add increment to radius for XY move	
IWnn	Add increment to radius for WZ move	
RAa(a)(a)(a)	Run absolute, all axes given	(5.6)
RIa(a)(a)(a)	Run incremental, all axes given	(5.6)
RRX	Run radius XY	(5.7.3)
RRW	Run radius WZ	(5.7.3)
RRA	Run radius, all axes	(5.7.3)
CRX	Run circular, XY	(5.7.1)
CRW	Run circular, WZ	(5.7.1)
CRA	Run circular, XY and WZ	(5.7.1)
ELa$\begin{bmatrix}\text{nn}\\\text{Vd}\end{bmatrix}$	Load ellipse ratio	(5.7.2)
SNann	Step input command to axis	(4.2.2.7)
PTAxxBannn	Load axis set point A begin position, range xx	(5.14.1)
PTAxxEannn	Load axis set point A end position, range xx	(5.14.1)
PTAxxFa	Forget axis set point A range xx	(5.14.1)
PTBxxBannn	Load axis set point B begin position, range xx	(5.14.1)
PTBxxEannn	Load axis set point B end position, range xx	(5.14.1)
PTBxxFa	Forget axis set point B range xx	(5.14.1)
SPAa$\begin{bmatrix}\text{A}\\\text{B}\end{bmatrix}a\begin{bmatrix}\text{A}\\\text{B}\end{bmatrix}$	Define CMC-0 set point A as occurrence of two axis set points	(5.14.2)
SPBa$\begin{bmatrix}\text{A}\\\text{B}\end{bmatrix}a\begin{bmatrix}\text{A}\\\text{B}\end{bmatrix}$	Define CMC-0 set point B as occurrence of two axis set points	(5.14.2)
SPCa$\begin{bmatrix}\text{A}\\\text{B}\end{bmatrix}a\begin{bmatrix}\text{A}\\\text{B}\end{bmatrix}$	Define CMC-0 set point C as occurrence of two axis set points	(5.14.2)
SPDa$\begin{bmatrix}\text{A}\\\text{B}\end{bmatrix}a\begin{bmatrix}\text{A}\\\text{B}\end{bmatrix}$	Define CMC-0 set point D as occurrence of two axis set points	(5.14.2)
SPAF	Forget CMC-0 set point A	(5.14.2)
SPBF	Forget CMC-0 set point B	(5.14.2)
SPCF	Forget CMC-0 set point C	(5.14.2)
SPDF	Forget CMC-0 set point D	(5.14.2)
TCn	Initialize teach mode for profile n	(5.15)
TMa(nnn)	Load teach point	(5.15)

Conditional execution and profile branching commands:

These commands can also be executed interactively or stored within profiles. They allow you to control profile execution through logic functions and mathematical comparisons of integer variables. See Section 5.16 for details and examples.

Note: these commands use the following notation:

A	- AND function	I	- User Input	>	- Greater than
O	- OR function	V	- User Integer Variable	=	- Equal to
X	- XOR function	B	- User Boolean Variable	<>	- Not equal to
N	- NOT function				

In addition, brackets indicate that the programmer must choose one of the items within the brackets. Parentheses indicate optional items.

Mnemonic	Action
WTTnn	Wait until timer is elapsed
WTMa	Wait until axis is stopped
WTMA	Wait until all axes are stopped
WTI	Wait until index pulse
$\text{WTE}\left[\begin{array}{l}(\text{N})\begin{bmatrix}\text{B}\\\text{I}\end{bmatrix}\text{d}\left(\begin{bmatrix}\text{A}\\\text{O}\\\text{X}\end{bmatrix}(\text{N})\begin{bmatrix}\text{B}\\\text{I}\end{bmatrix}\text{d}\right)\\ \text{Vd}\begin{bmatrix}>\\=\\<>\end{bmatrix}\text{Vd}\end{array}\right]$	Wait until expression is true
$\text{IF}\left[\begin{array}{l}(\text{N})\begin{bmatrix}\text{B}\\\text{I}\end{bmatrix}\text{d}\left(\begin{bmatrix}\text{A}\\\text{O}\\\text{X}\end{bmatrix}(\text{N})\begin{bmatrix}\text{B}\\\text{I}\end{bmatrix}\text{d}\right)\\ \text{Vd}\begin{bmatrix}>\\=\\<>\end{bmatrix}\text{Vd}\end{array}\right]\text{GSn}$	If expression true, GOSUB profile n
$\text{IF}\left[\begin{array}{l}(\text{N})\begin{bmatrix}\text{B}\\\text{I}\end{bmatrix}\text{d}\left(\begin{bmatrix}\text{A}\\\text{O}\\\text{X}\end{bmatrix}(\text{N})\begin{bmatrix}\text{B}\\\text{I}\end{bmatrix}\text{d}\right)\\ \text{Vd}\begin{bmatrix}>\\=\\<>\end{bmatrix}\text{Vd}\end{array}\right]\text{GTn}$	If expression true, GOTO profile n

Note: Use of both integer variables and either Boolean variables or user inputs in the same conditional expression is not allowed. Also, if you wish to complement only one logical operand in an expression, then it must be the first.

Auxiliary Position Equation commands:

These commands allow you to control individual axes with the Auxiliary Position Equation. They are useful for jogging functions, encoder tracking, and joystick applications, to name a few. See Section 5.10 for details and examples.

Note: these commands use the following notation:

A - Analog input channel
N - Number
E - External encoder input
F - Internal Frequency Source
V - Integer Variable

Mnemonic	Action
$\text{FSa}\begin{bmatrix}\text{n}\\ \text{Vd}\end{bmatrix}$	Load axis frequency source value
AEab	Enable/disable axis Auxiliary Position Equation
$\text{LMa}\begin{bmatrix}\text{Ad}\\ \text{Nnn}\\ \text{Vd}\end{bmatrix}$	Load axis Auxiliary Position Equation multiplier, either analog input, constant number, or integer variable
$\text{LIa}\begin{bmatrix}\text{Ed}\\ \text{F}\end{bmatrix}$	Load axis Auxiliary Position Equation input, either external encoder or internal frequency source

Phase-Locked Loop commands:

These commands are used only for phase-locked loop (PLL) applications. See Section 5.13 for details and examples.

Mnemonic	Action
PHGn	Load PLL gain
PHZn	Load PLL zero
$\text{PHO}\begin{bmatrix}\text{nn}\\ \text{Vd}\end{bmatrix}$	Load PLL offset
$\text{PHL}\begin{bmatrix}\text{nn}\\ \text{Vd}\end{bmatrix}$	Load PLL cycle length
PHCnn	Load PLL correction limit
SPH	Set PLL counter to zero

I/O and Variable commands:

These commands are for use with the user I/O lines, integer variables, and analog inputs. Some are allowed in defined profiles, while others must be executed interactively. Those marked with a (#) are not allowed in defined profiles.

Numbers in parentheses after the command description indicate sections of the manual where more detailed information is available about the command.

Mnemonic		Action	
ST$\begin{bmatrix}B\\I\end{bmatrix}$d		Set Boolean variable or I/O pin	
RS$\begin{bmatrix}B\\I\end{bmatrix}$d		Reset Boolean variable or I/O pin	
DRdb	(#)	Set direction of I/O pin (output or input)	(4.2.4)
RAd	(#)	Report analog input value	(4.4.3)
RDB	(#)	Report Boolean variable values	(4.4.3)
RDI	(#)	Report user input values	(4.4.3)
ADdnn		Load analog channel deadband	(4.2.5)
AOd$\begin{bmatrix}nn\\Vd\end{bmatrix}$		Load analog channel offset	(4.2.5)
Vd=?	(#)	Report integer variable value	(5.11)
Vd=a$\left(\begin{bmatrix}*\\/\\+\\-\end{bmatrix}b\right)$		Arithmetic integer variable assignment	(5.11)
Vd=nnnn		Fixed number integer variable assignment	(5.11)

In the first integer variable assignment command, "a" and "b" can be:

PHE	- PLL error (in position pulses)
PHP	- PLL master position (in position pulses)
Ed	- External encoder input (represents velocity)
Vd	- Other integer variable
(N)Bd	- Boolean variable ("N" is optional, implies complement)
(N)Id	- User I/O pin ("N" is optional, implies complement)
Td	- Thumbwheel input
Ad	- Analog input
Pa	- Axis position

(#) - Not allowed in defined profiles.

Status and position query commands: (Not allowed in defined profiles.)

Mnemonic	Action	
RS	Report status of CMC-0 unit	(4.3.1)
RSa	Report status of axis	(4.3.1)
RCa	Report command position of axis	(4.4.2)
RPa	Report actual position of axis	(4.4.2)
FEa	Report following error of axis	(4.4.2)
FC	Report fault code of CMC-0 unit	(6.1.2)
FCa	Report fault code of axis	(6.1.2)

Editing commands:

These commands are used when you are "inside" a profile definition and need to scroll through it to verify its contents, delete a line, or insert a line. See Section 5.17.

Mnemonic	Action
X	Scroll to next line in profile
DEL	Delete profile line
!	Abort edit mode

APPENDIX C

DEFAULT VALUES

Table C-1 CMC-0 Parameter Default Values

CMC-0 PARAMETER	AXES	DEFAULT VALUE
Unit Ratio	All	1 pulse/unit
Absolute Move	All	0 units
Incremental Move	All	0 units
Acceleration Rate	All	80,000 p/sec/sec
Trajectory Velocity	XY pair	0 units/sec
Trajectory Velocity	WZ pair	0 units/sec
Velocity Offset	XY pair	0 units/sec
Velocity Offset	WZ pair	0 units/sec
Radius of Curvature	XY pair	0 units
Radius of Curvature	WZ pair	0 units
Ellipse Ratio	All	1
User I/O Line Direction		1 (input)
Analog Input Deadband		0 A/D counts
Analog Input Offset		0 A/D counts
Auxiliary Frequency Source	All	0 Hz
Auxiliary Multiplier	All	0
Auxiliary Input	All	frequency source
Auxiliary Equation Toggle	All	0 (off)
PLL Gain		1
PLL Zero		245
PLL Offset		0 pulses
PLL Cycle Length		2,500 pulses
PLL Correction Limit		65,535
Profile Line Enable Toggle		0 (off)
Diagnostic Toggle		0 (off)
Profile 128 Enable Toggle		0 (off)

Table C-2 CMC-1 Control Constant Default Values

CONTROL CONSTANT	DEFAULT VALUE
Negative Software Overtravel	- 8,388,608 pulses
Positive Software Overtravel	8,388,607 pulses
Encoder Ratio	4,096
Deadband	0 pulses
Deadband Fault	0 (off)
Correction Time	8 ms
Position Correction	0 (off)
Power Save	0 (off)
Register Wrap	0 (off)

Table C-3 CMC-2 Control Constant Default Values

CONTROL CONSTANT	DEFAULT VALUE
Negative Software Overtravel	- 8,388,608 pulses
Positive Software Overtravel	8,388,607 pulses
Encoder Line Density	1,000 pulses/rev
Deadband	0 pulses
Integration Band	10 pulses
Position Band	32,767 pulses
Pole	0
Gain	1
Zero	0
Feedforward	0
Register Wrap	0 (off)

APPENDIX D

HARDWARE CONFIGURATION

HARDWARE CONFIGURATION

The CMC Coordinated Motion Controllers offer flexible hardware configuration in a combination of permanent and field-configurable settings.

D.1 Permanent Settings

The catalog numbering system is used to specify permanent settings as described below:

Table D-1 Catalog Number Designation

```
CATALOG NUMBER FORMAT                              CMC - X X X X - X
                                                   ---   -------   -
                                                    |    | | | |   |
product series prefix: ------------------------------    | | | |   |
                                                         | | | |   |
     CMC = Coordinated Motion Controller                 | | | |   |
                                                         | | | |   |
type of controller: -------------------------------------- | | |   |
                                                           | | |   |
     0 = Master (CMC-0)                                    | | |   |
     1 = stepper slave axis (CMC-1)                        | | |   |
     2 = servo slave axis (CMC-2)                          | | |   |
                                                           | | |   |
                                                           | | |   |
                                                           | | |   |
microstepping capability: ---------------------------------- | |   |
                                                             | |   |
     1 = microstepping available (All CMC-1's)               | |   |
     0 for servo slaves and Masters (CMC-0's)                | |   |
                                                             | |   |
                                                             | |   |
maximum continuous output current to motor: ------------------ |   |
                                                               |   |
     0 = unit without amplifier                                |   |
     5 = 5 amps (CMC-1)                                        |   |
     7 = 7 amps (CMC-2)                                        |   |
                                                               |   |
parallel output current type: ----------------------------------   |
                                                                   |
     0 = sinking outputs                                           |
     1 = sourcing outputs                                          |
                                                                   |
input voltage range: -----------------------------------------------

     1 =  90-130 VAC
     2 = 180-260 VAC
```

```
EXAMPLE :                                                 CMC - 0 0 0 0 - 1
                                                          ---   - - - -   -
                                                            |   | | | |   |
product series prefix: -------------------------------------    | | | |   |
                                                                | | | |   |
      CMC = Coordinated Motion Controller                       | | | |   |
                                                                | | | |   |
type of controller: ---------------------------------------------- | | |   |
                                                                  | | |   |
      0 = Master (CMC-0)                                          | | |   |
                                                                  | | |   |
not used for CMC-0: ------------------------------------------------ |   |
                                                                    |   |
                                                                    |   |
parallel output current type: ----------------------------------------  |
                                                                        |
      0 = sinking                                                       |
                                                                        |
input voltage range: -----------------------------------------------------

      1 = 90-130 VAC


EXAMPLE :                                                 CMC - 2 0 7 1 - 2
                                                          ---   - - - -   -
                                                            |   | | | |   |
product series prefix: -------------------------------------    | | | |   |
                                                                | | | |   |
      CMC = Coordinated Motion Controller                       | | | |   |
                                                                | | | |   |
type of controller: ---------------------------------------------- | | |   |
                                                                  | | |   |
      2 = Servo slave (CMC-2)                                     | | |   |
                                                                  | | |   |
not used for CMC-2: ------------------------------------------------ | |   |
                                                                    | |   |
maximum continuous output current to motor: ------------------------- |   |
                                                                      |   |
      7 = 7.0 amps                                                    |   |
                                                                      |   |
parallel output current type: ----------------------------------------  |
                                                                        |
      1 = sourcing                                                      |
                                                                        |
input voltage range: -----------------------------------------------------

      2 = 180-260 VAC
```

D.2 Field-Configurable Hardware Settings

DIP switch adjustment of field-configurable hardware settings is discussed in detail in Sections 2.10 and 3.8 of this manual. The tables below summarize possible field adjustments and the factory settings for them.

Table D-2 CMC-0 Field-Configurable Hardware Settings

Setting Type	Switch Number	Switch Positions	Factory Setting
Encoder 1 Signal Type	1	1	quadrature
Encoder 1 Pulse Multiplier	1	2-3	1X multiplier
Encoder 2 Signal Type	1	4	quadrature
Encoder 2 Pulse Multiplier	1	5-6	1X multiplier
Encoder 3 Signal Type	1	7	quadrature
Encoder 3 Pulse Multiplier	1	8-9	1X multiplier
Controller Address	2	1-3	4
Communication Format	2	5	echo mode
Baud Rate	2	6-7	1200
Serial Standard	2	8	RS-232

Table D-3 CMC-1 Field-Configurable Hardware Settings

Setting Type	Switch Number	Switch Positions	Factory Setting
Step Size	1	1-3	full
Encoder Feedback	1	4	not used
Motor Current	1	5-8	5.0 amps
Axis Assignment	2	1-3	X axis
Profile Line Format	2	8	non-encoded
Position Pulse Multiplier	2	9-10	1X multiplier

Table D-4 CMC-2 Field-Configurable Hardware Settings

Setting Type	Switch Number	Switch Positions	Factory Setting
Tachometer Selection	1	1-6	no tach
Continuous Motor Current	1	7-10	7.0 amps
Axis Assignment	2	1-3	X axis
Profile Line Format	2	8	non-encoded
Position Pulse Multiplier	2	9-10	1X multiplier