Whedco CMC-31PX-2-B

Coordinated Motion Controller

$2295.00

In Stock

Qty Available: 1

Used and in Excellent Condition

Open Web Page

https://www.artisantg.com/87203-2

ARTISAN TECHNOLOGY GROUP

Your **definitive** source for quality pre-owned equipment.

Artisan Technology Group

(217) 352-9330 | sales@artisantg.com | artisantg.com

- Critical and expedited services
- In stock / Ready-to-ship
- We buy your excess, underutilized, and idle equipment
- Full-service, independent repair center

Whedco

CMC SERIES
COORDINATED MOTION CONTROLLERS
USER'S MANUAL (014/2)
MODELS
CMC-XXXX-X-X

Price $50.00

Pub. No. 014 ■ Revision 2 ■ Copyright 1987

WHEDCO INCORPORATED
4750 Venture Drive ■ Ann Arbor, MI 48108
Technical Support (313) 665-7540 ■ Facsimile (313) 665-6694

Since this equipment can be applied in many diverse situations and in conjunction with equipment from other vendors, the user and those responsible for specifying this equipment must determine for themselves its suitability for the use intended. In no event shall Whedco Incorporated be liable for loss of use, profit or consequential damages, or damage to other equipment, resulting from the use of this equipment.

The figures and examples in this manual are designed to demonstrate general concepts for the installation and maintenance of this equipment. The users should always verify interconnection requirements to and for other equipment as well as confirm installation and maintenance requirements for the specific application. In no case shall Whedco Incorporated be liable for actual use based on the guidelines mentioned herein.

CMC-000x Product Bulletin February 26, 1988

Revision 2.0 Operating System

The operating system of the CMC-000x product line has been upgraded to Revision 2.0 The purpose of this upgrade is to enhance the operation of the CMC product line by increasing the utility of some existing commands and add new capability by adding new commands. This Product Bulletin describes the differences between the previous CMC operating system and Revision 2.0. Users can identify CMC's with Rev 2 operating systems by the Rev 2 designation on the EPROM label.

I. Changes Which May Effect Compatibility

A primary goal of this revision is to increase the functionality while maintaining a level of compatibility with the previous operating system. Changes which may effect the compatibility of the new release with programs designed to execute on older CMC's are explained below.

Command Effected	Description of Change
CCS1 Memory Upload and Download	The memory image of the Rev 2.0 CMC is NOT compatible with the previous operating system. Memory images saved from previous systems will NOT execute in the Rev 2.0 system. To avoid incompatibility, we recommend downloading programs including initialization commands and profiles from ASCII source files using the Whedco CCS1 utilities.
Any command using Boolean variable 15	Boolean variable F (15) is set when an axis is stopped by a hardware or software overtravel.
Any command using Boolean variable 0	Boolean variable 0 is reset on power-up.
Any command using User Defined Outputs	All user defined outputs are reset on power-up.
WB	Profile 255 is automatically executed when the CMC is warm booted.
ELa	The elipse ratio is no longer supported. If the ELa command is entered, the message "Invalid Command" is sent to the terminal.
CRX CRW CRA	The run circular commands have been superceeded with CIX, CIW, CIA for circular commands with incremental parameters and with CAX, CAW and CAA for circular commands with absolute para-

	meters. See Section III below for a description of the new commands.
Any parameter load command	Parameter load commands executed without a value no longer load zero. If a parameter is expected and none is loaded, the message "Invalid Command" is sent to the terminal.
Diagnostic Mode	The diagnostic mode no longer automatically sends a line each time a profile is executed. The programmer must now include the DP command in a profile to cause the diagnostic line to be sent when the Diagnostic Mode is enabled.
Diagnostic Mode	The diagnostic line will not be sent if the serial interface port is busy.
Diagnostic Mode	Boolean variables are no longer reported in the Diagnostic Mode. Use the RDB command to send the Boolean variables to the terminal.

II. Command Enhancements

The commands listed in this section have been enhanced but are downwardly compatible with previous CMC Operating Systems.

Command Enhanced	Description of Enhancement
URann	When the unit ratio of an axis pair is loaded the unit ratio of the other axis in the pair is loaded to the same value unless it was previously loaded.
WTE[ ] IF[ ]GSn IF[ ]GTn	Conditional commands now allow tests for home inputs on the axis.
RRX[ ] RRW[ ] RRA[ ]	The maximum allowed radius has been increased from 65,535 to +/- 8,388,608 pulses.
TCn	The teach mode switch input has a longer delay to accomodate switch bounce.

III. New Commands

The following commands are new to the CMC Rev 2 Operating System. They are summarized here and described more fully in the Command Description and Format section.

Command	Description
STI0	This command has been added to allow the programmer to cause a CMC fault. The Fault Code will report a Software Fault.
FRa	Report the Fine Resolver reading.
CRa	Report the Coarse Resolver reading.
CDXd	Arcs and circles in the XY plane are run in the clockwise direction if d=0; counter-clockwise if d=1.
CDWd	Arcs and circles in the WZ plane are run in the clockwise direction if d=0; counter-clockwise if d=1.
ARX	Draw an arc in the XY plane with the radius specified by RX to the XY coordinates in the Absolute Move Register.
ARW	Draw an arc in the WZ plane with the radius specified by RW to the WZ coordinates in the Absolute Move Register.
ARA	Draw arcs simultaneously in the XY and WZ planes with the radii specified by RX and RW to the coordinates in the Absolute Move Register.

NOTE: The sign of the radius in the RX and RW registers select whether the major or minor arc segment is drawn by the RRX, RRW, RRA, ARX, ARW, and ARA commands. A positive radius selects the minor arc, a negative radius selects the major arc.

CIX	Draw a circle in the XY plane with the center specified by the Incremental Move Register in the direction specified by CDXd.
CIW	Draw a circle in the WZ plane with the center specified by the Incremental Move Register in the direction specified by CDWd.
CIA	Draw circles simultaneously in the XY and WZ planes with centers specified by the Incremental Move Registers in the directions specified by CXd and CWd.
CAX	Draw a circle in the XY plane with the center specified by the Absolute Move Register in the direction specified by CDXd.

CAW	Draw a circle in the WZ plane with the center specified by the Absolute Move Register in the direction specified by CDWd.
CAA	Draw circles simultaneously in the XY and WZ planes with centers specified by the Absolute Move Registers in the directions specified by CXd and CWd.

IV. COMMAND DESCRIPTIONS AND FORMATS (new commands)

NOTE: All of the descriptions of the circle and arc segment commands assume a right handed coordinate system.

ARA — Absolute Run Radius All

Operation: Execute arcs simultaneously in the XY and WZ planes from the present position to the coordinates in the Absolute Move Register. The arc radii are specified by RX and RW. The arc directions are specified by CDX and CDW; (see the CDX and CDW commands below) the choice of major or minor arc segment by the sign of the radius.

Example:

If Present position = 0,0,0,0; AMX=10, AMY=10, AMW=5, AMZ=-5, RX=20, RW=-15, CDX=0, CDW=1

Then executing the statement

ARA draws the minor arc segment of a circle with a radius of 20 from X,Y=0,0 to X,Y=10,10 in the clockwise direction and simultaneously draws the major arc segment of a circle with a radius of 15

from W,Z=0,0 to W,Z=5,-5 in the counterclockwise direction.

ARW — Absolute Run Radius WZ

Operation: Execute an arc in the WZ plane from the present position to the W,Z coordinates in the Absolute Move Register. The arc radius is specified by RW. The arc direction is specified by CDW; (see the CDX and CDW commands below) the choice of major or minor arc segment by the sign of the radius.

Example:

If Present position = 0,0; AMW=5, AMZ=-5
RW=-15, CDW=1

Then executing the statement

ARW draws the major arc segment of a circle with a radius of 15 from W,Z=0,0 to W,Z=5,-5 in the ccw direction.

ARX — Absolute Run Radius XY

Operation: Execute an arc in the XY plane from the present position to the X,Y coordinates in the Absolute Move Register. The arc radius is specified by RX. The arc direction is specified by CDX; (see the CDX and CDW commands below) the choice of major or minor arc segment by the sign of the radius.

Example:

If Present position = 0,0; AMX=10, AMY=10
RW=20, CDW=0

Then executing the statement

ARX draws the major arc segment of a circle with a radius of 20 from X,Y=0,0 to X,Y=10,10 in the clockwise direction.

CAA — Circle Absolute All

Operation Execute circles simultaneously in the XY and WZ planes. The circle centers are specified by the X,Y and W,Z coordinates in the Absolute Move Register. The circle directions are specified by CDX and CDW.

Example

If Present position = 0,0,0,0; AMX=10, AMY=10, AMW=5, AMZ=-5, CDX=0, CDW=1

Then executing the statement

CAA draws a circle in the clockwise direction with a center at 10,10 in the XY plane and simultaneously draws a circle in the counterclockwise direction with a center at 5,-5 in the WZ plane.

CAW Circle Absolute WZ

Operation: Execute a circle in the WZ plane. The circle center is specified by the W,Z coordinates in the Absolute Move Register. The circle direction is specified by CDW.

Example:

If Present position = 0,0; AMW=5, AMZ=-5, CDW=1

Then executing the statement

CAW draws a circle in the counterclockwise direction with a center at 5,-5 in the WZ plane.

CAX Circle Absolute XY

Operation: Execute a circle in the XY plane. The circle center is specified by the X,Y coordinates in the Absolute Move Register. The circle direction is specified by CDX.

Example:

If Present position = 0,0; AMX=10, AMY=10 CDX=0

Then executing the statement

CAX draws a circle in the clockwise direction with a center at 10,10 in the XY plane.

CDWb Circle Direction WZ

Operation Specify the direction of circles in the WZ plane.

Example CDW0 specifies that arcs and circles drawn in the WZ plane will be in the clockwise direction.

CDW1 specifies that arcs and circles drawn in the WZ plane will be in the counterclockwise direction.

CDXb — Circle Direction XY

Operation — Specify the direction of circles in the XY plane.

Example — CDX0 specifies that arcs and circles drawn in the XY plane will be in the clockwise direction.

CDX1 specifies that arcs and circles drawn in the XY plane will be in the counterclockwise direction.

CIA — Circle Incremental All

Operation: — Execute circles simultaneously in the XY and WZ planes. The circle centers are specified by the X,Y and W,Z coordinates in the Incremental Move Register. The circle directions are specified by CDX and CDW.

Example:

If the present position of X,Y,W,Z = 0,5,10,15 and IMX=10, IMY=10, IMW=5, IMZ=-5, CDX=0, CDW=1

Then executing the statement

CIA — draws a circle in the clockwise direction with a center at 10,15 in the XY plane and simultaneously draws a circle in the counterclockwise direction with a center at 15,10 in the WZ plane.

CIW — Circle Incremental WZ

Operation: — Execute a circle in the WZ plane. The circle center is specified by the W,Z coordinates in the Incremental Move Register. The circle direction is specified by CDW.

Example:

If the present position of W,Z = 10,15 and IMW=5, IMZ=-5, CDW=1

Then executing the statement

CIW — draws a circle in the counterclockwise direction with a center at 15,10 in the WZ plane.

CIX — Circle Incremental XY

Operation: Execute a circle in the XY plane. The circle center is specified by the X,Y coordinates in the Incremental Move Register. The circle direction is specified by CDX.

Example:

If the present position of X,Y = 0,5 and IMX=10, IMY=10, CDX=0

Then executing the statement

CIX draws a circle in the clockwise direction with a center at 10,15 in the XY plane.

CRa — Coarse Resolver

Operation: Report the axis a Coarse Resolver position to the terminal.

Example:

If the CMC controlling axis X is equipped with the two resolver option and the coarse resolver is at 1234,

Then executing the statement

CRX writes 1234 to the terminal.

FRa — Fine Resolver

Operation: Report the axis a Fine Resolver position to the terminal.

Example:

If the CMC controlling axis X is equipped with the resolver option and the fine resolver is at 2345,

Then executing the statement

FRX writes 2345 to the terminal.

STI0 — Set I/O zero

Operation: Cause a CMC fault condition.

Example:
Executing the statement

STIO causes the CMC-000x to immediately halt all of the axes and light the FAULT lamp on the CMC-000x. (No axes motion can occur until the controller is warm booted by issuing a WB command or by toggling the Enable Line on the X axis controller from the inactive state to the active state.)

TABLE OF CONTENTS

TABLE OF CONTENTS
(cont'd)

TABLE OF CONTENTS
(cont'd)

TABLE OF CONTENTS
(cont'd)

TABLE OF CONTENTS
(cont'd)

LIST OF FIGURES

LIST OF FIGURES
(cont'd)

LIST OF TABLES

LIST OF TABLES
(cont'd)

LIST OF EXAMPLE PROFILES

SECTION 1.0

GENERAL DESCRIPTION

1.1 INTRODUCTION

The Whedco CMC Series Coordinated Motion Controllers provide for coordinated control of up to four axes of stepping and/or servo motors. The CMC-0 acts as the coordinating (master) unit and the programming interface. CMC-1 Stepping Motor Controllers and CMC-2 Servo Motor Controllers perform the actual motor control functions.

Programming is accomplished through the CMC-0 unit via an ASCII command set. Once programmed, individual profiles can be executed via any serial device or by parallel inputs which reside on the individual axis controllers. Up to eight CMC-0 units can be daisy-chained off of one serial line for easy programming and control of large systems.

The CMC-0 incorporates many features to make profile definition and system operation as flexible as possible. Analog and encoder inputs can both be used for jog functions and other moves. The encoder inputs can be configured to accept either quadrature or pulse and direction signals. In addition, profiles can incorporate moves based on dwell times, motor position(s), and logical combinations of external inputs and internal Boolean variables. Profiles can also initiate the execution of other profiles through the use of GOTO and GOSUB statements.

1.2 FEATURES

1.2.1 CMC-0 Master Unit.

-Four axis controller (axes W, X, Y and Z) with linear and circular interpolation capabilities.

-Coordinates both stepping and servo motors in any combination.

-Allows circular interpolation between X and Y axes and between W and Z axes. Also allows coordination between XY circular motion and WZ circular motion.

-Allows linear interpolation between any combination of the four axes.

-Acts as programming interface for entire system.

-Programmable via RS-232, RS-422, or RS-485 at 1200, 4800, 9600, and 38,400 baud.

-Up to four CMC-0 units can be programmed off of one RS-232 line. As many as eight can be chained off of one RS-422 or RS-485 line.

-Edit capability allows for easy redefinition of profiles and parameter values.

-Allows parameter entry in user-specified units.

-Three incremental encoder inputs allow for moves based on quadrature or pulse/direction inputs. One of these allows for index pulse input as well.

-Four analog inputs accept signals of -10 VDC to +10 VDC. ADC resolution is 12 bits. Each input has programmable deadband and offset.

-Analog velocity offset allows trimming of programmed run speeds.

-Two dedicated parallel outputs indicate BUSY and FAULT conditions.

-Twelve user-definable parallel I/O lines allow user to completely specify input and output signals based on specific system requirements.

-Two parallel output setpoint lines provided. Each can be specified to be active when the setpoints of any two axes become active simultaneously, for example when a certain point in a plane has been reached.

-Parallel outputs can be either sourcing or sinking. Must be specified at time of order.

-Sixteen integer variables allow system parameters to be computed and changed "on the fly".

-Supports both GOSUB and GOTO profile execution based on logical combinations of user input lines and/or Boolean variables, and also on algebraic combinations of integer variables.

-Up to five thumbwheels can be interfaced to the unit using the user-definable parallel I/O.

-Special phase-locked loop equation allows the controller to phase-lock one axis to another.

-Teach Mode program construct provides additional flexibility by allowing users to "teach" the controller different positions for different machine operations without editing the profile.

-Battery-backed RAM eliminates the need to redefine profiles and parameters on power-up or system reset.

-Profiles can be executed either via serial link or parallel I/O.

-255 profile maximum capacity.

1.2.2 CMC-1 and CMC-2 Slave Units.

-Self-contained units for stepping motor control (CMC-1) and servo motor control (CMC-2). Each unit contains motor control software, encoder interface, and switching power supply.

-Tachometer feedback is not required for servo systems but may be used if desired.

-Output current to motor is DIP switch programmable.

-Microstepping capability available for stepping motor controllers.

-Inputs allow predefined profiles to be executed via parallel I/O.

SECTION 2.0

CMC-0 INSTALLATION AND CONFIGURATION

Installation and configuration of the CMC-0 Master Unit consists of the following steps:

1. Mounting.
2. Connecting AC power.
3. Connecting host terminal/computer for programming.
4. Connecting communication wiring between CMC-0 and slave units.
5. Connecting parallel I/O (if used).
6. Connecting external encoders (if used).
7. Connecting analog inputs (if used).
8. Configuring DIP switches.

The following sections describe each of these steps in detail.

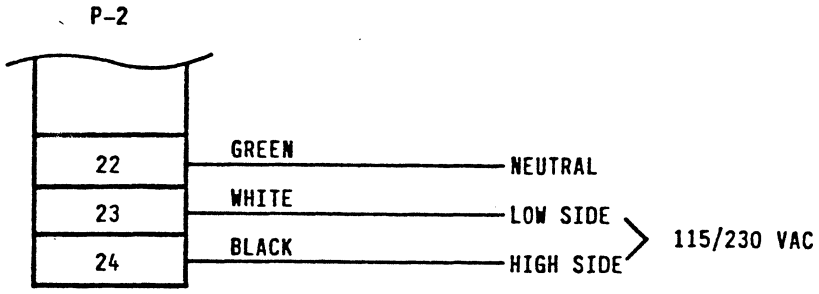
2.1 MOUNTING

The Whedco CMC-0 controller may be mounted vertically in an enclosure using the four slotted holes in the controller end plates. The mounting panel or surface should be tapped to accept #10-32 UNF machine screws, and locking hardware should be used. If the ambient temperature of the enclosure will exceed 55 degrees Centigrade, then adequate cooling should also be provided.

2.2 POWER INPUT WIRING

The CMC-0 controller operates at 115 VAC or 230 VAC depending on the model selected. Appendix D, Hardware Configuration, explains the model numbering system used to identify the line voltage input. Verify the proper input voltage by checking the model number; severe damage could result from applying improper line voltage. Figure 2-1 illustrates power line wiring for the CMC-0.

Figure 2-1 Wiring CMC-0 to Power Line



2.3 COMMUNICATION WIRING: HOST TO CMC-0

The CMC-0 controllers are compatible with the RS-232C and RS-422 serial communication standards. The controllers can also operate on a party line as outlined in the RS-485 standard. However, if the party line approach is used, the echo mode of operation is not allowed (see Section 4.1.2). If the Common Device Address will be used, the SYNC line of each CMC-0 controller connected to the same serial port must be tied together.

To connect the host computer or terminal to the CMC-0 unit(s), refer to the figure appropriate for the communication protocol chosen. Note that the CMC-0 units transmit seven character bits with one parity bit (odd), one "start" bit, and one "stop" bit for a total of 10 bits per character. The serial device which will communicate with the controller must conform to this format.

Figure 2-2 Serial Connections for RS-232C

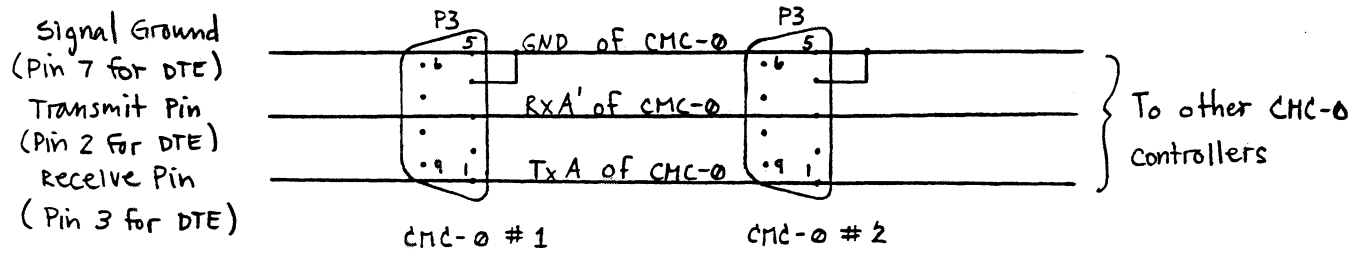


Figure 2-3 Serial Connections for RS-422

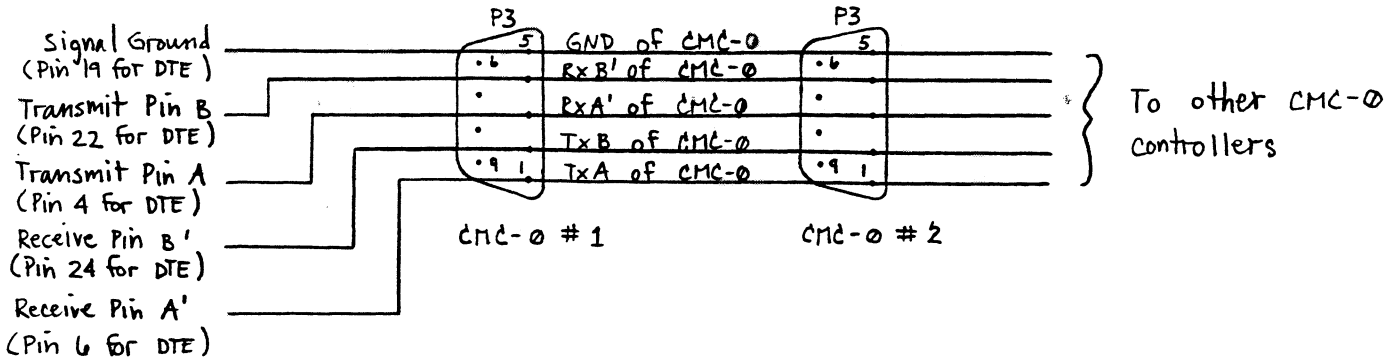
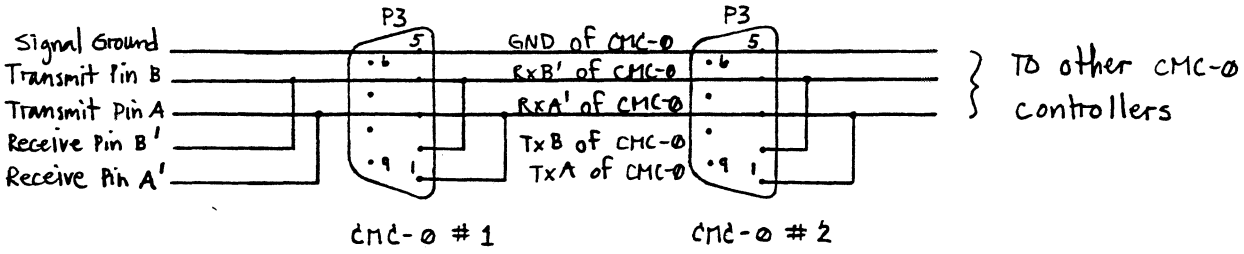


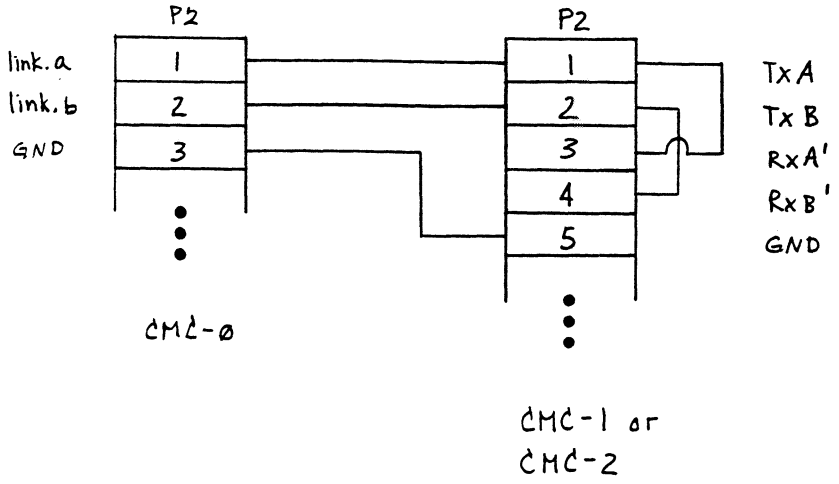
Figure 2-4 Serial Connections for RS-485 (party line)



2.4 COMMUNICATION WIRING: CMC-0 TO SLAVE UNITS

Communication between the CMC-0 master unit and the CMC-1 and CMC-2 slave units is accomplished via a high-speed serial link. Figure 2-5 illustrates the proper connection.

Figure 2-5 Master/Slave Communication Wiring



2.5 PARALLEL I/O WIRING

The CMC-0 controller has twelve user-defined parallel I/O lines and four dedicated parallel outputs. Each user-defined I/O line can be specified as either an input or an output. Certain user I/O lines may also be configured for thumbwheel inputs (see Section 2.6)

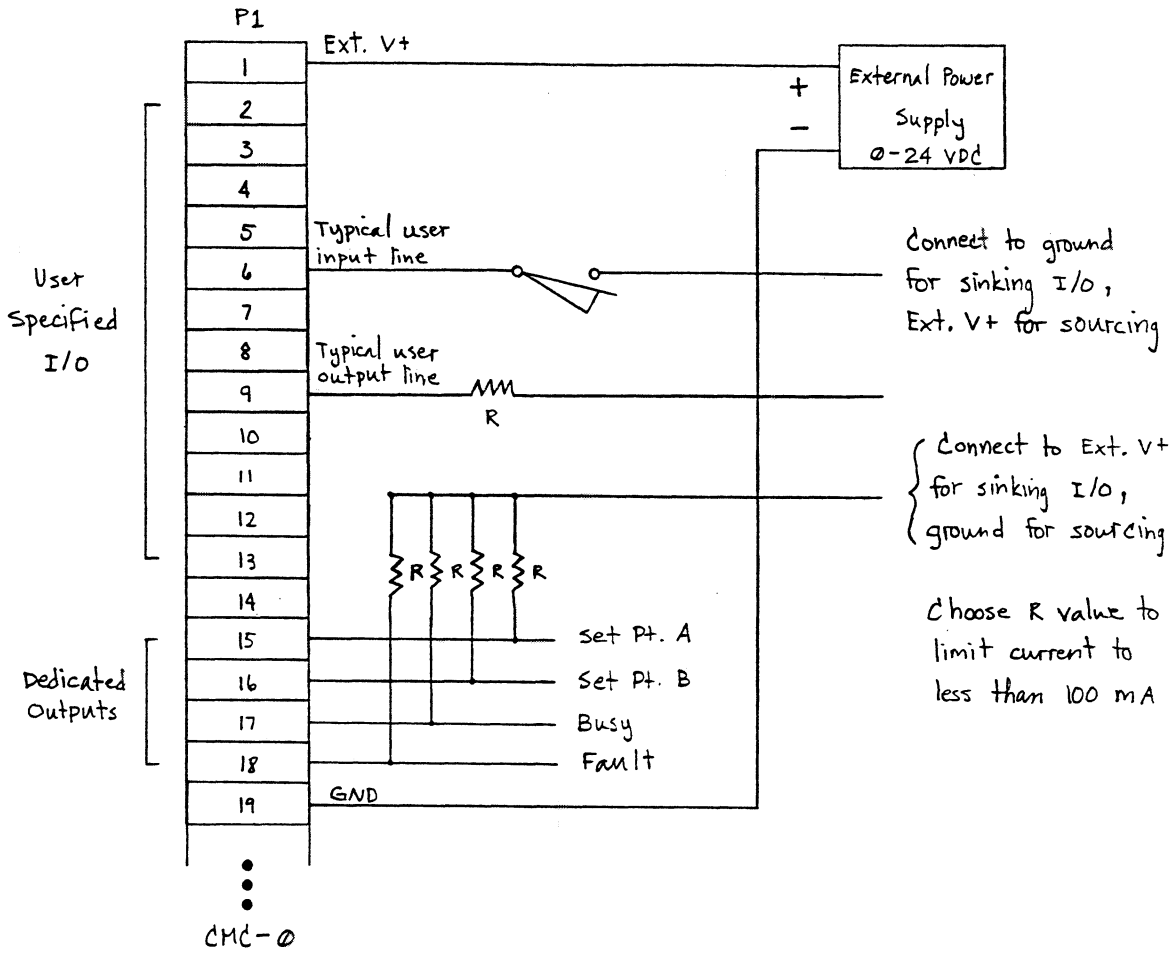
Prior to delivery, each CMC-0 unit is configured for either sinking or sourcing I/O per the customers order. The active state for the inputs and outputs on sourcing units corresponds to a high voltage level; for sinking units the active state corresponds to a low voltage level. The "Fault" output is active whenever the controller is faulted, while the "Busy" output is active whenever a profile is being defined or executed. Each set point output is active when its programmed "true" condition is satisfied.

The user I/O lines on sourcing units have a 2K resister to ground (pull-down resistor). User I/O lines on sinking units have a 2K resistor to V+ (pull-up resistor). All output lines can withstand currents up to 100 mA. Maximum voltage for inputs and outputs is 24 VDC.

Figure 2-6 on the next page illustrates typical parallel I/O wiring.

Note: The ext. V+ line must be connected to an external (user supplied) voltage source of not more than 24 VDC for the I/O lines to work properly.

Figure 2-6 CMC-0 Parallel I/O Wiring



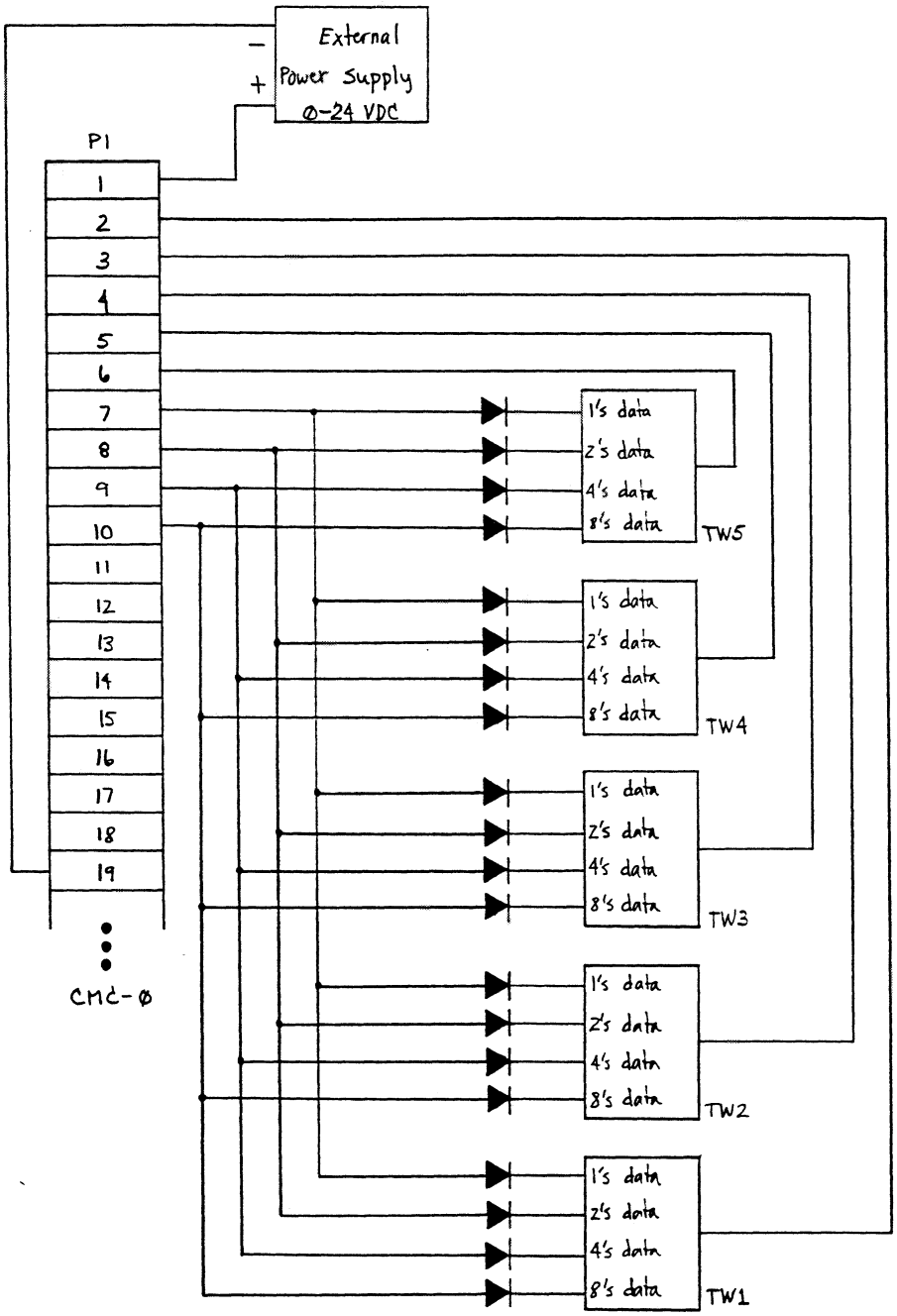
2.6 THUMBWHEEL INTERFACE

The CMC-0 controller allows you to interface up to five thumbwheel digits to the user parallel I/O lines. Figure 2-7 on the next page illustrates the proper connections between the thumbwheels and the I/O lines. See Section 5.12 for thumbwheel programming details.

Note that I/O lines 6 through 9 are used for the four data lines required. All thumbwheels used must have their data lines connected to these lines, and each of these lines must be defined as an input. I/O lines 1 through 5 are the strobe lines. Each of these must be defined as an output.

Of course, if a user I/O line is connected to a thumbwheel, it can not be used as a general purpose I/O line.

Figure 2-7 Thumbwheel Interface Connections



The isolation diodes shown in the figure are often included in the thumbwheel switch package. If they are not, then you will have to add them to the circuit. Note that the direction of the diodes in the figure is for a CMC-0 with sinking I/O. For units with sourcing I/O, reverse the direction of the diodes from that shown.

2.7 EXTERNAL ENCODER WIRING

The CMC-0 unit provides connections for three optical incremental encoders and one index (marker) pulse. Note that these encoders are not required for position feedback for the individual axes. They are only required when the external encoders are used with the Auxiliary Position Equation. (See Section 5.10.4.)

Each encoder input section can accept either standard sine or square wave quadrature encoder signals, or pulse and direction signals. (As configured by DIP switch SW1 on the CMC-0 unit.) In either case, both single-ended and differential modes are supported.

Figures 2-8 and 2-9 illustrate the possible configurations.

Figure 2-8 Wiring Single-Ended Encoders and Pulse/Direction Sources to CMC-0

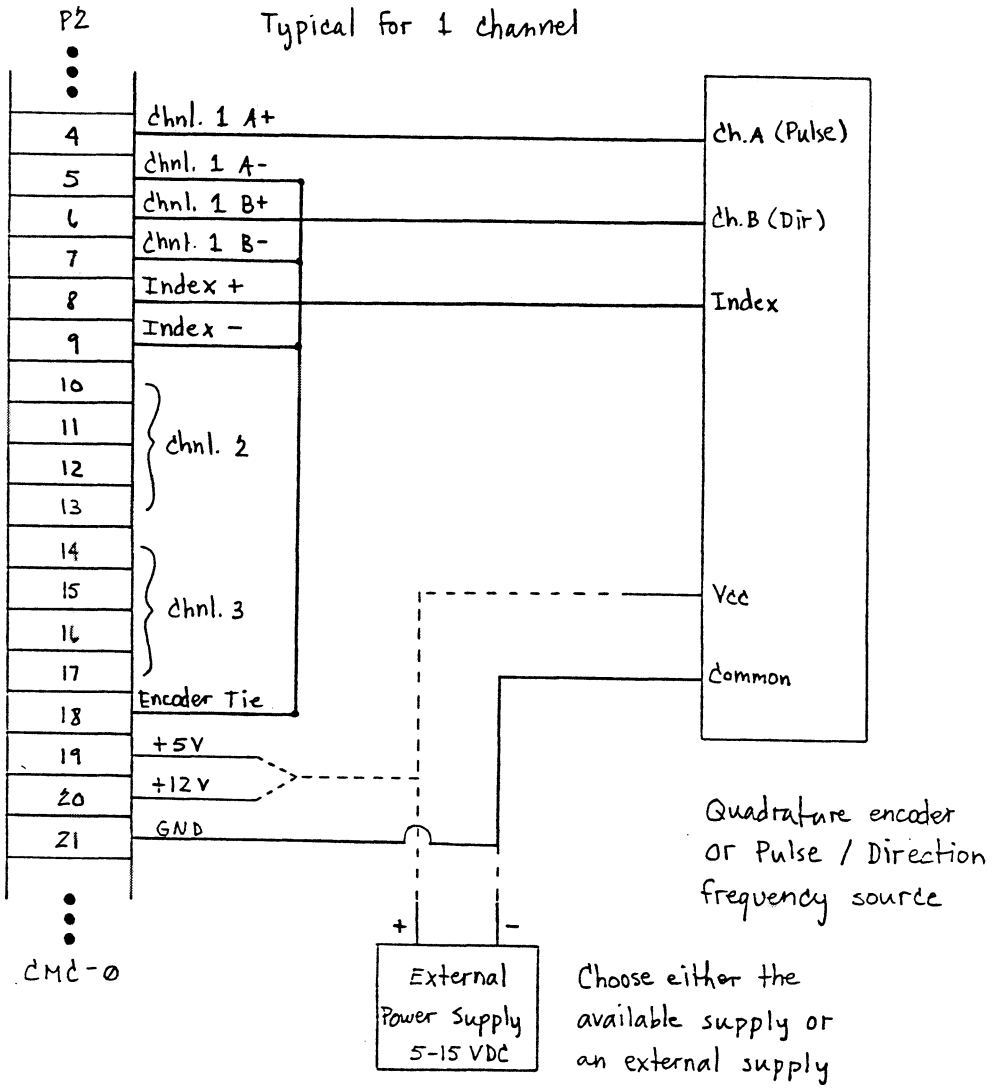
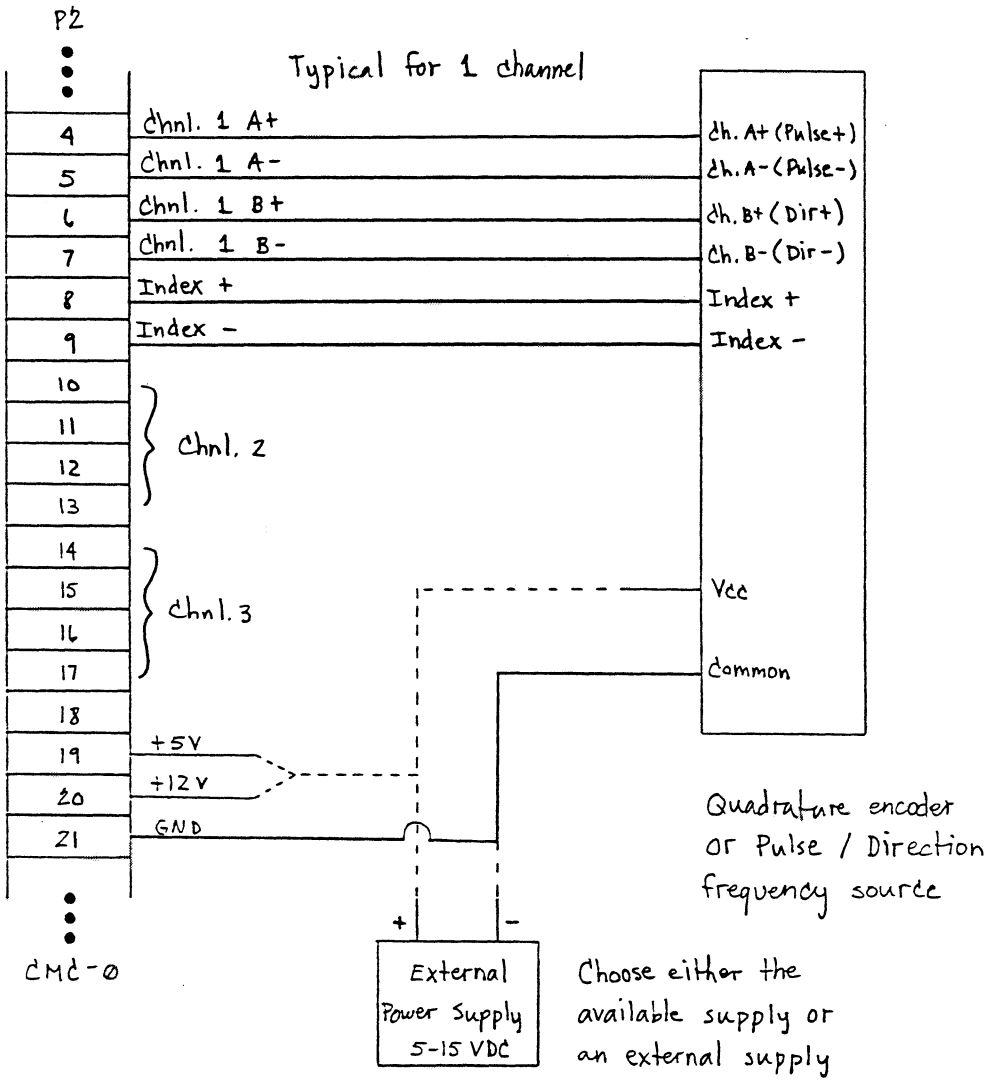


Figure 2-9 Wiring Differential Encoders and Pulse/Direction Sources to CMC-0



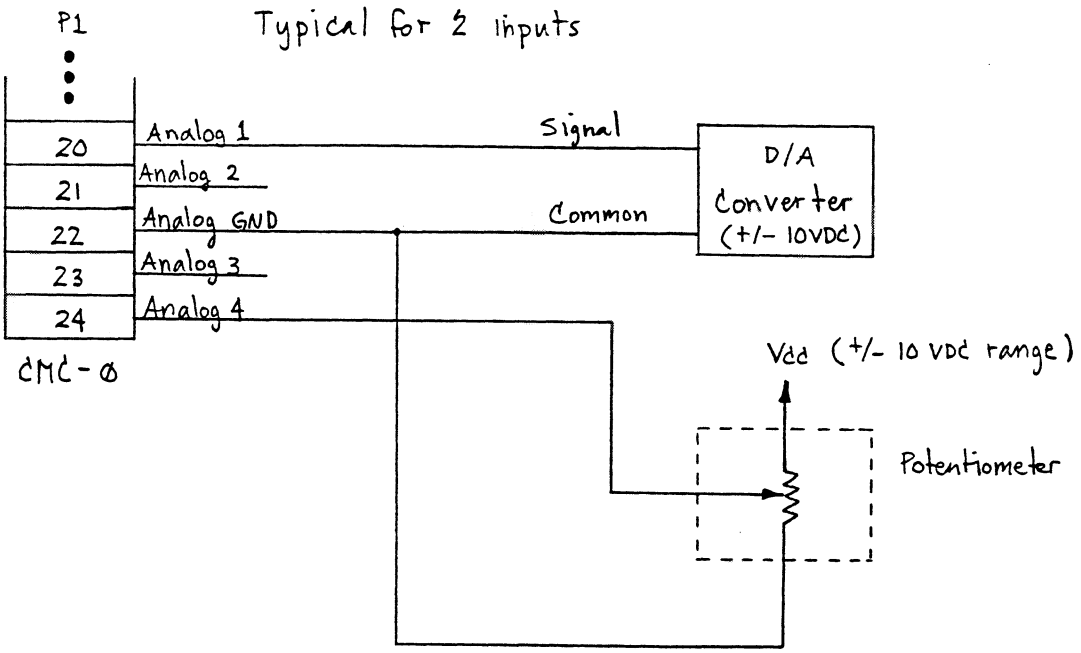
When using one of the CMC-0 external encoder inputs, the "fwd" (green) and "rev" (yellow) LEDs on the controller indicate the direction of encoder motion. If the encoder direction (as indicated by the LEDs) does not match the actual encoder direction, simply reverse the wiring between the A and B channel inputs.

2.8 ANALOG INPUT WIRING

The CMC-0 controller provides four analog inputs for use with the Auxiliary Position Equation, velocity offset commands (VOX and VOW), and integer variables. Each of these inputs can accept signals in the range of +/- 10 VDC. In typical applications these inputs would be used with potentiometers or potentiometer-based joysticks for manual speed trimming, jogging functions, or motion parameter adjustments.

Figure 2-10 illustrates typical wiring for these inputs.

Figure 2-10 Analog Input Wiring



2.9 CMC-0 DC SUPPLIES.

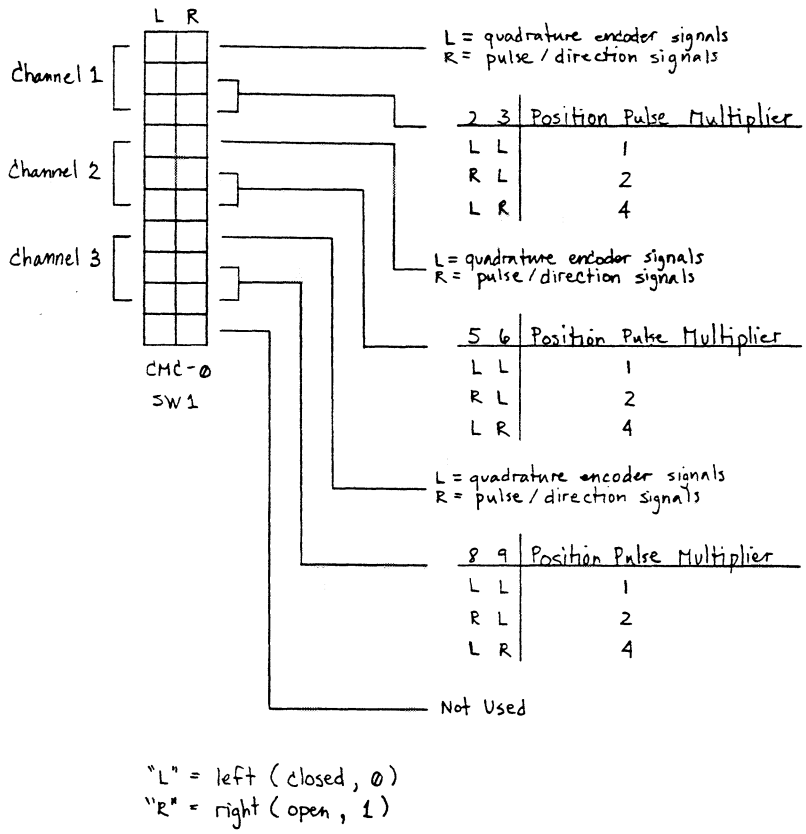
The CMC-0 unit provides +5 VDC and +12 VDC power supplies for general purpose use. They are typically used to power external encoders and/or joysticks. The five volt supply is rated for a maximum of 1.0 amp. The twelve volt supply will supply 0.5 amps.

2.10 DIP SWITCH CONFIGURATION

The CMC-0 unit has two DIP switches for configuring the encoder channels, unit address, and serial communication protocol. The upper switch bank configures the three external encoder inputs for either quadrature or pulse/direction signals. It also sets the Position Pulse Multiplier for each input channel.

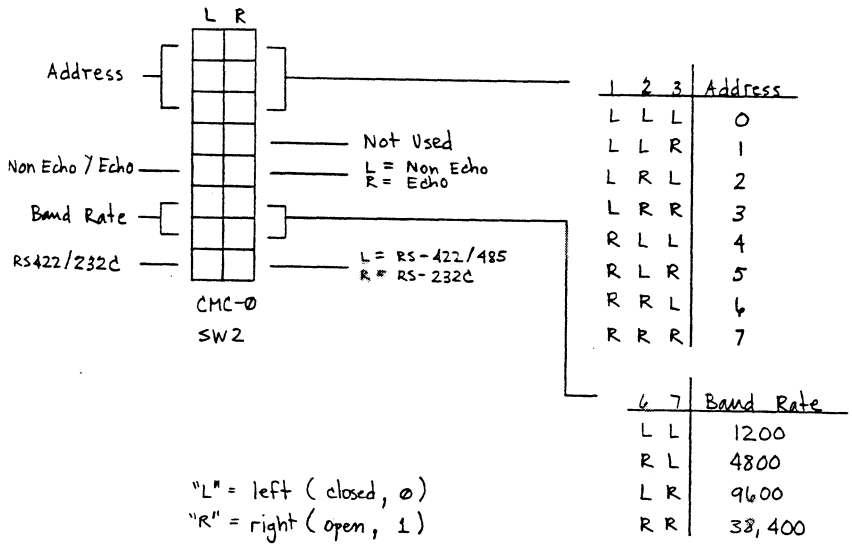
See Table 2-1 for the configuration of DIP switch SW1.

Table 2-1 External Encoder DIP Switch Settings (SW1)



DIP switch SW2 determines the address and the communication protocol used by the CMC-0. Table 2-2 illustrates the available options and the corresponding settings.

Table 2-2 CMC-0 Communication DIP Switch Settings (SW2)



SECTION 3.0

CMC-1 & CMC-2 INSTALLATION AND CONFIGURATION

Installation and configuration of CMC-1 Stepping Motor Controllers and CMC-2 Servo Motor Controllers consists of the following steps:

1. Mounting.
2. Connecting AC power.
3. Connecting Master/Slave communication link (Refer to Section 2.4).
4. Connecting encoder (optional for CMC-1, required for CMC-2).
5. Connecting motor to drive outputs.
6. Connecting parallel I/O.
7. Configuring DIP switches.

The following sections describe each of these steps in detail.

3.1 MOUNTING

3.1.1 Panel Mounting.

Whedco CMC-1 and CMC-2 controllers may be mounted vertically in enclosures using the four slotted holes in the controller end plates. The mounting panel or surface should be tapped to accept #10-32 UNF machine screws. Locking hardware should also be used. For optimum convection cooling, mount the unit with the heat sink fins in a vertical position.

Whedco does not recommend this mounting method in air-tight enclosures where the temperature of the inside air will exceed 55 degrees Centigrade. In this case, through-wall heat sink mounting is recommended as discussed in Section 3.1.2.

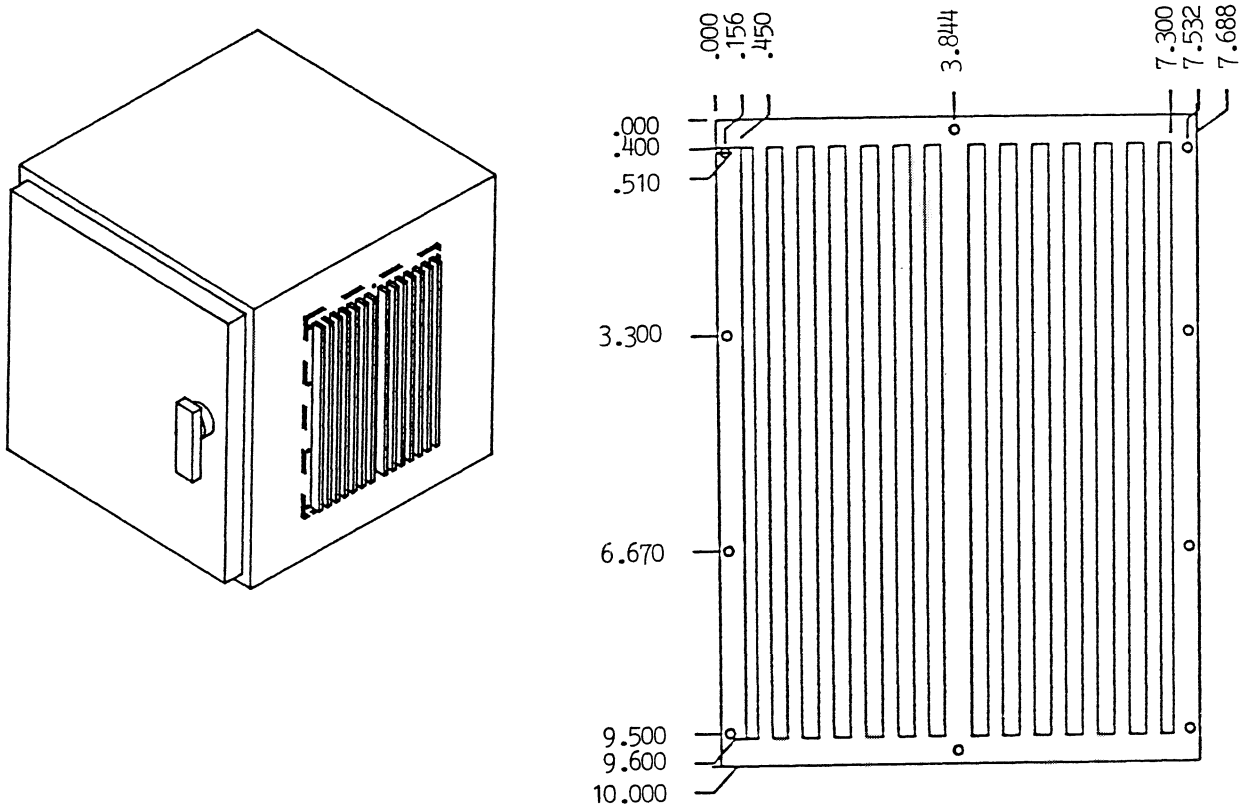
3.1.2 Through-Wall Heat Sink Mounting.

This mounting method is preferred when the temperature of the inside air may exceed 55 degrees Centigrade or where other heat-generating devices are in close proximity. Using Figure 3-1 as a guide, cut out the enclosure wall to the dimensions indicated. Drill ten 0.156" diameter holes in the locations indicated by the layout. Mount the unit in a vertical position to allow adequate air flow. If you desire to seal the controller heat sink to the enclosure wall, apply a thin bead of RTV sealant around the heat sink flange. Use #6-32 x 1/2 UNC screws to mount the controller using the tapped holes provided in the heat sink. Figure 3-2 illustrates this mounting configuration.

Figure 3-1 Layout for Through-Wall Mounting

(This figure intentionally left blank)

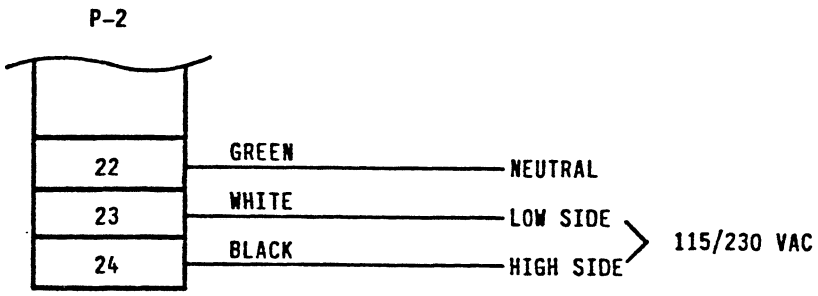
Figure 3-2 Through-Wall Heat Sink Mounting



3.2 POWER INPUT WIRING

The CMC-1 and CMC-2 controllers operate at 115 VAC or 230 VAC depending on the model selected. Appendix D, Hardware Configuration, explains the model numbering system used to identify the line voltage input. Verify the proper input voltage by checking the model number; severe damage could result from applying improper voltage. Figure 3-3 illustrates power line wiring.

Figure 3-3 Wiring CMC-1 and CMC-2 to Power Line



3.3 ENCODER INPUT WIRING

The CMC-1 and CMC-2 controllers can use position feedback from an incremental encoder with either sine or square wave quadrature output. Position feedback is required for the CMC-2 Servo Motor Controller, and optional for the CMC-1 Stepping Motor Controller. The controllers provide input lines for single-ended or differential output encoders as well as power output lines for supplying 5 and 12 volt encoders. 15 volt encoders will require an external power supply. The index line can be optionally connected provided the encoder is so equipped. To connect the encoder, follow the diagrams given in Figures 3-4 and 3-5.

When using an encoder with a CMC-1 Stepping Motor Controller or CMC-2 Servo Motor Controller, the "fwd" (green) and "rev" (amber) LEDs on the controller indicate the direction of encoder motion. If the encoder direction does not match the motor direction, the controller will fault when a motion command is initiated. To solve this problem, either:

1. Switch the A and B encoder channel connections, or

2. If the motor is a stepping motor, reverse the motor direction by reversing the wiring of one of the motor windings. If the motor is a servo motor, reverse the motor direction by reversing the two armature leads. If a tachometer is used, make sure to reverse the tach leads as well.

Figure 3-4 Single-Ended Encoder Connections for CMC-1 and CMC-2 Controllers

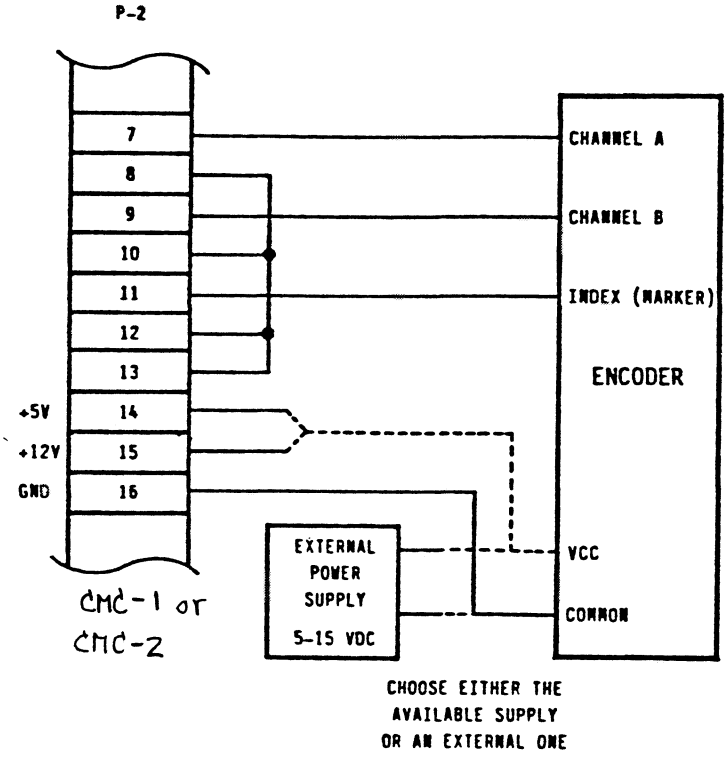
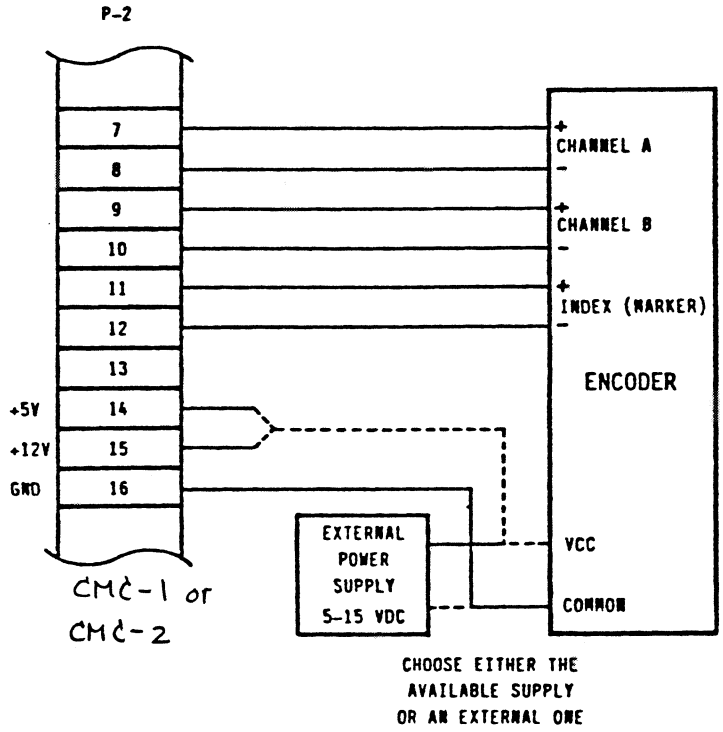


Figure 3-5 Differential Encoder Connections
for CMC-1 and CMC-2 Controllers



3.4 STEPPING MOTOR DRIVE OUTPUTS (CMC-1)

CMC-1 Stepping Motor Controllers drive four or six-lead motors from most manufacturers. Figure 3-6 illustrates connection to four-lead motors. See Figure 3-7 for connection to six-lead motors.

Figure 3-6 Wiring CMC-1 to Four-Lead Stepping Motors

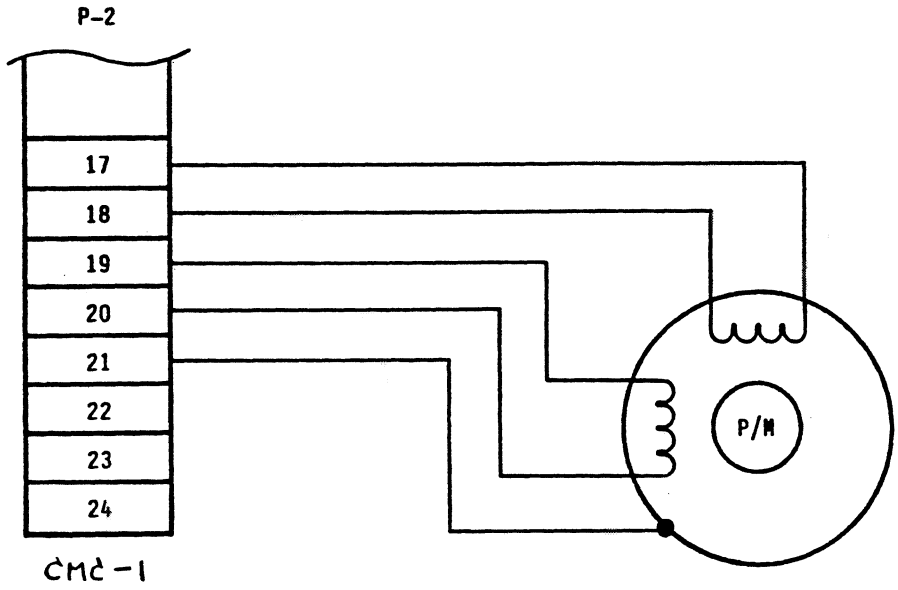
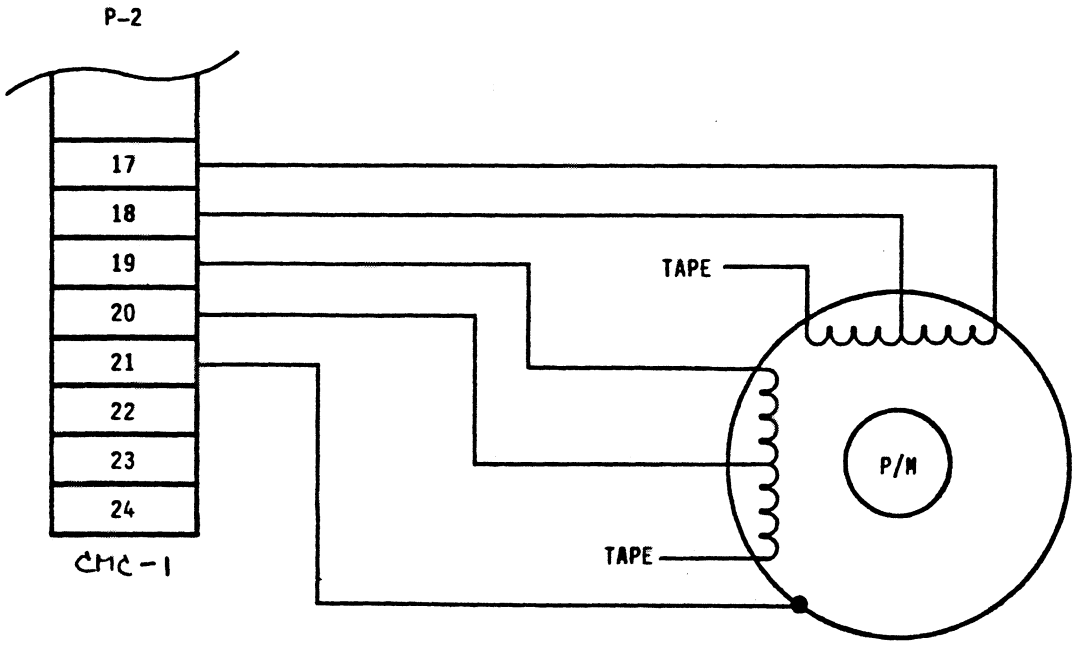


Figure 3-7 Wiring CMC-1 to Six-Lead Stepping Motors



3.5 SERVO MOTOR DRIVE OUTPUTS (CMC-2)

CMC-2 Servo Motor Controllers drive DC servo motors with or without tachometer feedback. Figure 3-8 illustrates connection for tachless systems. See Figure 3-9 for systems with tachometers.

Figure 3-8 Wiring CMC-2 to Tachless Servo Systems

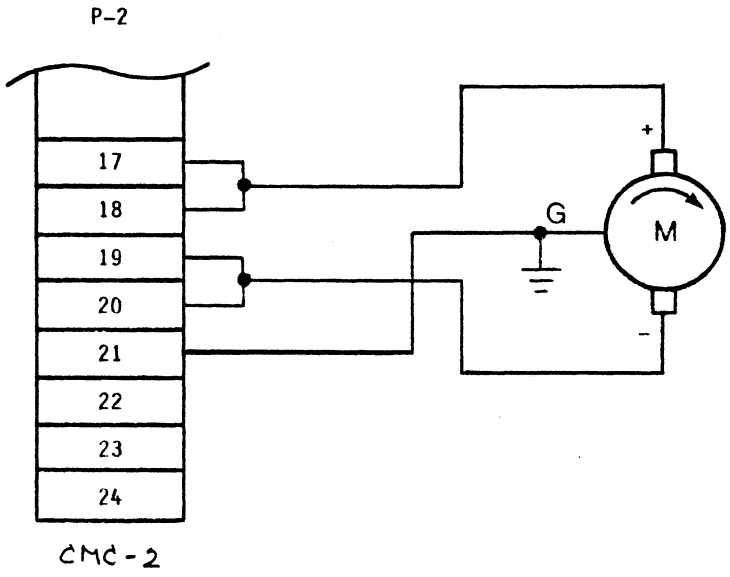
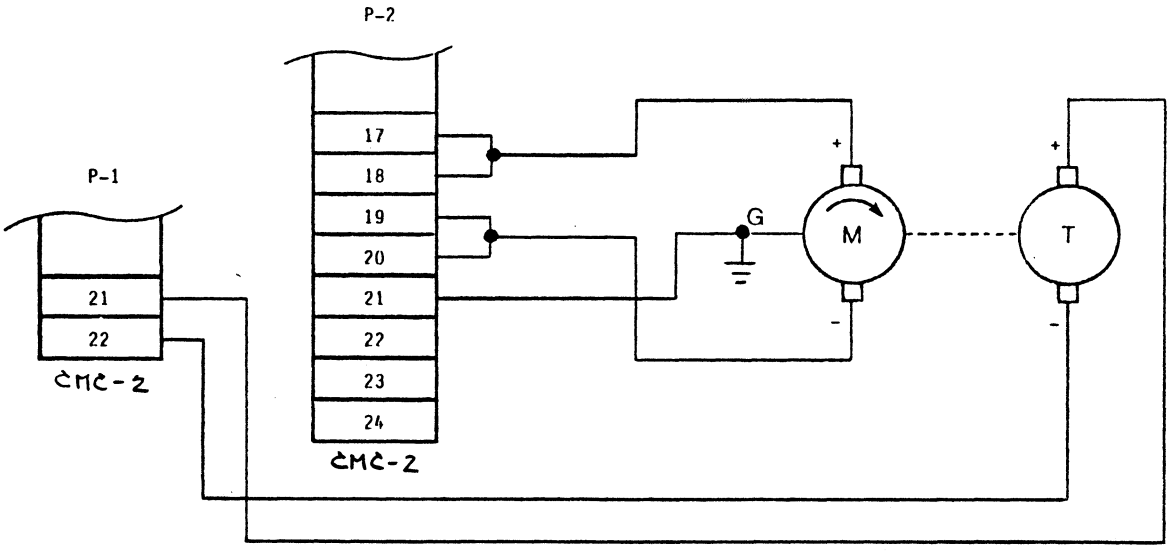


Figure 3-9 Wiring CMC-2 to Servo Systems with Tachometers



3.6 SERVO AMPLIFIER INTERFACE (CMC-2 without amplifier)

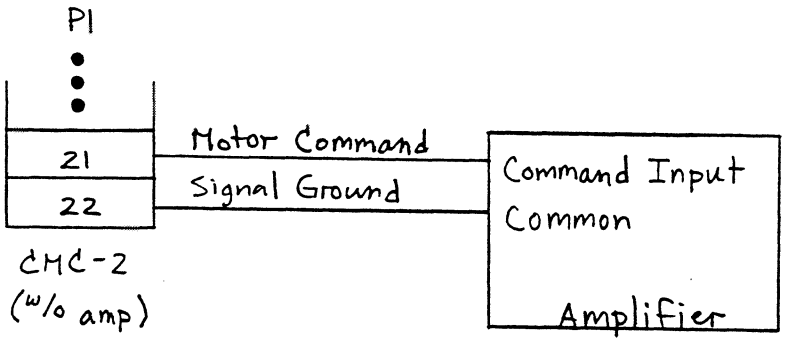
3.6.1 Introduction.

CMC-2 Servo Motor Controllers which are not equipped with amplifiers do not have motor drive outputs. Instead, they provide a +/- 10 VDC command signal for connection to the input stage of a servo amplifier.

Interfacing to the servo amplifier requires both connection of the controller to the amplifier as well as calibration of the other servo system components. Calibration procedures vary depending on the amplifier type used. Refer to the appropriate section below for the recommended calibration procedure for the amplifier in use. Once the amplifier is correctly calibrated, simply connect the motor command lead from the CMC-2 to the amplifier input and connect one of the controller DC grounds to the amplifier ground as shown in Figure 3-10.

Note: The output driver of a CMC-2 unit without amplifier is single-ended. If the amplifier is also single-ended, then never attempt to reverse the actual motor direction by reversing the command and ground leads from the controller. If you want to change the motor direction, you must reverse the encoder, tachometer (if used), and motor leads.

Figure 3-10 Wiring CMC-2 to a Servo Amplifier



3.6.2 Voltage Gain (voltage in - voltage out) Amplifier Without Tachometer Feedback.

1. Connect the amplifier to the motor.

2. Adjust the gain on the amplifier to make 9 volts in give an output voltage which is maximum for the motor.

3. Using a new "D" cell battery or other suitable voltage source connected with negative to the amplifier input and positive to ground, check that the motor runs in a direction which turns on the green LED of the CMC-2 controller. This will be the forward direction of travel.

4. If the motor direction is wrong, reverse the motor leads. If the direction is correct but the amber LED turns on, switch the A and B encoder channel leads at the controller.

3.6.3 Transconductance (voltage in - current out) Amplifier Without Tachometer Feedback.

1. Connect the amplifier to the motor.

2. Adjust the gain on the amplifier to make 9 volts in give an output current which is the maximum for the motor.

3. Using a new "D" cell battery or other suitable voltage source connected with negative to the amplifier input and positive to ground, check that the motor runs in a direction which turns on the green LED on the controller. This will be the positive direction of travel.

4. If the motor direction is wrong, reverse the motor leads. If the direction is correct but the amber LED turns on, switch the A and B encoder channel leads at the controller.

3.6.4 Servo Amplifier With Tachometer Feedback.

1. Connect the motor and tachometer to the amplifier, and follow steps 2-6 to adjust the gain of the amplifier to make 9 volts in run the motor at 10% more than the maximum speed desired. The speed is set at 10% more to provide an adequate error margin.

2. Connect a "D" cell battery or other suitable voltage source to the amplifier with negative to the amplifier input and positive to ground.

3. Verify that the voltage source is set for a steady 1.5 VDC.

4. Place a voltmeter across the tachometer leads to monitor actual speed.

5. Look up the tach constant, and calculate the correct tach voltage. At 1.5 volts in, the resultant speed should be 18.3% of the maximum speed desired. This resultant speed multiplied by the tach constant will give the desired tach voltage reading.

6. Adjust the gain of the amplifier until this speed (i.e tach voltage) is achieved.

   See the example in Table 3-1.

Table 3-1 Example Calculation of Voltage Reading for Servo Amplifier with Tachometer Feedback

Assume 3000 RPM is maximum desired speed
18.3% of 3000 RPM = 550 RPM
Ke of tachometer = 3 Volts/1000 RPM
Desired tach voltage = Ke * 550 = 1.65 volts

After the adjustments are made, check that the green LED on the controller is on when the motor is running. This will be the positive direction of travel. If the motor direction is wrong, reverse both the motor and the tachometer leads. If the direction is correct but the amber LED is on, switch the A and B encoder channel leads at the controller.

3.7 PARALLEL I/O CONNECTIONS

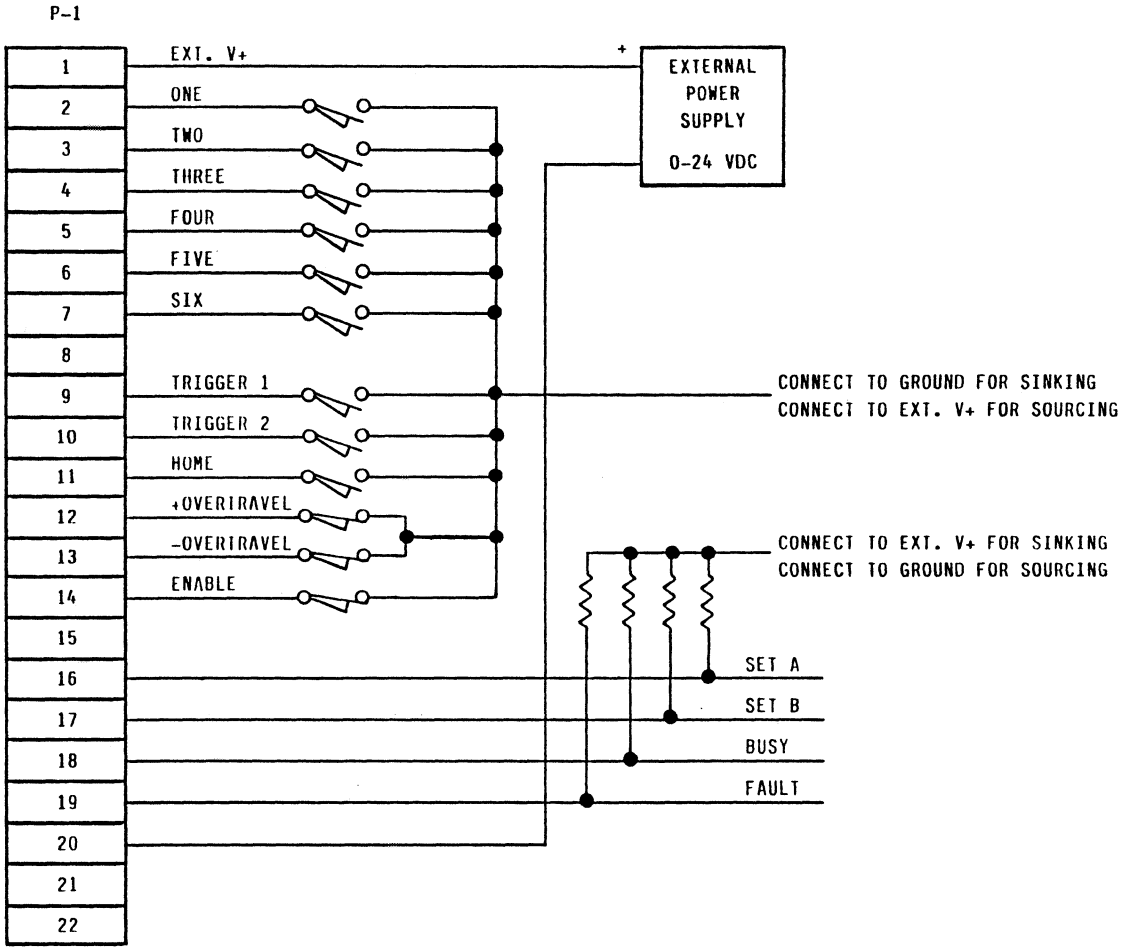
The CMC-1 and CMC-2 controllers can have either sourcing or sinking I/O (specified by customer prior to delivery). The active state for the inputs and outputs on sourcing units corresponds to a high voltage level except for the overtravel inputs. Overtravel inputs are active low on sourcing units.

The active state for the inputs and outputs on sinking units corresponds to a low voltage level except for the overtravel inputs. Overtravel inputs are active high on sinking units.

For either type of unit, the profile lines and the trigger inputs are sensitive to the inactive to active transition, while the Home, Overtravel, and Enable inputs are level sensitive. Note that the Overtravel inputs must be inactive for the unit to work. If Overtravel switches are not used, the inputs should be connected to V+ for sourcing units or DC ground for sinking units. DO NOT USE OVERTRAVEL INPUTS IN EMERGENCY STOP CIRCUITS (See Section 4.10).

See Figure 3-11 on the next page for typical connection details.

Figure 3-11 Wiring CMC-1 and CMC-2 I/O Lines



The output lines can withstand current up to 200 mA. The input lines have a 2K resistor to ground for sourcing units or V+ for sinking units. All the inputs and outputs can withstand voltages up to 24 VDC.

Note: The ext. V+ line must be connected to an external (user supplied) voltage source of not more than 24 VDC for the I/O lines to work properly.

3.8 DIP SWITCH CONFIGURATIONS

3.8.1 CMC-1 DIP Switch Settings.

The CMC-1 Stepping Motor Controller has two DIP switch banks for adjusting step size, motor current, position feedback, and axis assignment (X, Y, W, or Z). Switches for changing these settings are accessible via the front panel by removing the protective metal plate at the bottom front of the unit. Only make these adjustments when power is not applied to the unit.

3.8.1.1 Step Size.

All CMC-1 Stepping Motor Controllers operate in full or half-step mode. Controllers equipped with optional microstepping capability can operate at nominal fractional step sizes to 1/32 of a full step angle. Refer to Appendix D, Unit Configuration, to see if the unit is equipped with microstepping capability. To adjust the step size, locate the uppermost switch bank (SW1) behind the cover plate. Refer to Table 3-2 to set the desired step size. Note that only positions 1, 2, and 3 set step size.

Table 3-2 CMC-1 Step Size DIP Switch Settings (SW1)

Step Size	Models Available	Switch Positions 1 2 3	Nominal Step Angle*
Full Step	all	L L L	1.800 degrees
Half Step	all	R L L	0.900 degrees
1/4 Step	microstep	L R L	0.450 degrees
1/8 Step	microstep	R R L	0.225 degrees
1/16 Step	microstep	L L R	0.113 degrees
1/32 Step	microstep	R L R	0.057 degrees

"R" = right (open)
"L" = left (closed)

*Note: Step angle based on 200 step per revolution motor. Instantaneous step angle is motor and load dependent.

3.8.1.2 Motor Current.

For three Amp units, the output current to the stepping motor is adjustable in 0.20 Amp increments from 0.20 Amps to 3.0 Amps. Five Amp units are adjustable in 0.33 Amp increments from 0.33 Amps to 5.0 Amps. Refer to Appendix D, Unit Configuration, to confirm the maximum current output of the unit. The motor current is set with the last four positions of DIP switch SW1. Refer to Table 3-3 to select the switch setting for the required motor current.

Table 3-3 CMC-1 Motor Current DIP Switch Settings (SW1)

Switch Positions 5 6 7 8	Current (3A unit)	Current (5A unit)
R R R R	0.00 Amps*	0.00 Amps*
L R R R	0.20 Amps	0.33 Amps
R L R R	0.40 Amps	0.67 Amps
L L R R	0.60 Amps	1.00 Amps
R R L R	0.80 Amps	1.33 Amps
L R L R	1.00 Amps	1.67 Amps
R L L R	1.20 Amps	2.00 Amps
L L L R	1.40 Amps	2.33 Amps
R R R L	1.60 Amps	2.67 Amps
L R R L	1.80 Amps	3.00 Amps
R L R L	2.00 Amps	3.33 Amps
L L R L	2.20 Amps	3.67 Amps
R R L L	2.40 Amps	4.00 Amps
L R L L	2.60 Amps	4.33 Amps
R L L L	2.80 Amps	4.67 Amps
L L L L	3.00 Amps	5.00 Amps

"R" = right (open)
"L" = left (closed)

*Note: Whedco recommends the unit not be operated with zero motor current.

3.8.1.3 Encoder Feedback.

Position 4 on DIP switch SW1 selects encoder feedback. If an encoder is used, position four must be open (right); if not used it must be closed (left). The encoder line density can be multiplied by 1, 2, or 4 to produce the actual position pulses used for feedback information. Refer to Section 3.8.1.6 for position pulse multiplier switch settings.

3.8.1.4 Axis Assignment.

The first three switch positions of DIP switch SW2 are used to assign the CMC-1 unit to a specific axis. Table 3-4 shows the settings for each axis.

Table 3-4 CMC-1 Axis Assignment DIP Switch Settings (SW2)

Switch Positions 1 2 3	Axis Assignment
L L L	X
L L R	Y
L R L	W
L R R	Z

"R" = right (open)
"L" = left (closed)

3.8.1.5 Profile Line Configuration.

The parallel profile lines on the CMC-1 unit can be either encoded or non-encoded at the user's discretion (see Section 4.6). If you use them in the encoded state, then switch position 8 of DIP switch SW2 must be open (right). If you wish to use them in the non-encoded mode, then this switch must be closed (left).

3.8.1.6 Position Pulse Multiplier.

Switch positions 9 and 10 of DIP switch SW2 set the position pulse multiplier of the CMC-1. Refer to Table 3-5 for the proper setting.

Table 3-5 CMC-1 Position Pulse Multiplier DIP Switch Settings

Switch Positions 9 10	Position Pulse Multiplier
L L	1
R L	2
L R	4

"R" = right (open)
"L" = left (closed)

IMPORTANT - Switch positions 4 and 5 of DIP switch SW2 are unused on the CMC-1. However, positions 6 and 7 must be closed (left) for the unit to be operational.

3.8.2 CMC-2 DIP Switch Settings.

The CMC-2 Servo Motor Controller has two DIP switch banks for adjusting tachometer feedback, motor current, position feedback, and axis assignment (X, Y, W, or Z). Switches for changing these settings are accessible via the front panel by removing the protective metal plate at the bottom front of the unit. Only make these adjustments when power is not applied to the unit.

3.8.2.1 Tachometer Feedback.

The CMC-2 Servo Motor Controllers will operate with or without tachometer feedback. The first six positions of the uppermost DIP switch (SW1) configure the controller for tachometer feedback. Table 3-6 shows the appropriate settings.

Table 3-6 CMC-2 Tachometer Feedback DIP Switch Settings (SW1)

Tachometer Feedback	Maximum Tachometer Voltage	Switch Positions 1 2 3 4 5 6
NO	N/A	L L L L R L
YES	3.00 - 4.50 V	L R R R L R
	4.51 - 6.75 V	R L R R L R
	6.76 - 11.25 V	L L R R L R
	11.26 - 18.00 V	R R L R L R
	18.01 - 22.50 V	L R L R L R
	22.51 - 24.75 V	R L L R L R
	24.76 - 29.25 V	L L L R L R
	29.26 - 37.50 V	R R R L L R
	37.51 - 42.00 V	L R R L L R
	42.01 - 44.25 V	R L R L L R
	44.26 - 48.75 V	L L R L L R
	48.76 - 55.50 V	R R L L L R
	55.51 - 60.00 V	L R L L L R
	60.01 - 62.25 V	R L L L L R
	62.26 - 66.75 V	L L L L L R

"R" = right (open)
"L" = left (closed)

When using a tachometer, switch positions 1-6 must be set for maximum generated tachometer voltage using the following equation:

Max. Tach. Voltage = Tach EMF constant * Max. motor RPM required

The potentiometer located below SW1 adjusts tachometer offset. It can be ignored if a tachometer is not used. After the controller has been completely set up and is functioning properly, the potentiometer may need to be adjusted. To check its setting, follow these steps:

1. Temporarily turn the integrator off by sending the command IBa0, where "a" is the axis (W, X, Y, or Z).

2. With the motor stopped, read the following error by sending the FEa command. The number returned will represent the amount of offset error due to the tachometer. An offset less than five in magnitude is acceptable although zero is best.

3. If the offset is large, adjust the potentiometer and read the following error until it is within bounds. After completing the adjustment, be sure to set the integrator back to its previous value.

   Note: This adjustment is best made when the unit is at normal operating temperature.

3.8.2.2 Motor Current.

The maximum continuous output current to the motor is adjustable in 0.47 Amp increments from 0.47 Amps to 7.0 Amps. The maximum peak current is automatically set to two times the maximum continuous rating. Positions 7-10 of DIP switch SW1 set continuous output current to the motor. Refer to Table 3-7 on the next page to select the switch setting for the required motor current.

Table 3-7 CMC-2 Motor Current Switch Settings (SW1)

Continuous Current	Switch Positions 7 8 9 10
0.00 Amps*	R R R R
0.47 Amps	L R R R
0.93 Amps	R L R R
1.40 Amps	L L R R
1.87 Amps	R R L R
2.33 Amps	L R L R
2.80 Amps	R L L R
3.27 Amps	L L L R
3.73 Amps	R R R L
4.20 Amps	L R R L
4.67 Amps	R L R L
5.13 Amps	L L R L
5.60 Amps	R R L L
6.07 Amps	L R L L
6.53 Amps	R L L L
7.00 Amps	L L L L

"R" = right (open)
"L" = left (closed)

*Note: Whedco recommends the unit not be operated with zero motor current.

3.8.2.3 Axis Assignment.

The first three switch positions of DIP switch SW2 are used to assign the CMC-2 unit to a specific axis. Table 3-8 shows the settings for each axis.

Table 3-8 CMC-2 Axis Assignment DIP Switch Settings (SW2)

Switch Positions 1 2 3	Axis Assignment
L L L	X
L L R	Y
L R L	W
L R R	Z

"R" = right (open)
"L" = left (closed)

3.8.2.4 Profile Line Configuration.

The parallel profile lines on the CMC-2 unit can be either encoded or non-encoded at the user's discretion (see Section 4.6). If you use them in the encoded mode, then switch position 8 of DIP switch SW2 must be open (right). If you wish to use them in the non-encoded mode, then this switch must be closed (left).

3.8.2.5 Position Pulse Multiplier.

Switch positions 9 and 10 of DIP switch SW2 set the position pulse multiplier of the CMC-2. Refer to Table 3-9 for the proper setting.

Table 3-9 CMC-2 Position Pulse Multiplier DIP Switch Settings

Switch Positions 9 10	Position Pulse Multiplier
L L	1
R L	2
L R	4

"R" = right (open)
"L" = left (closed)

IMPORTANT - Switch positions 4 and 5 of DIP switch SW2 are unused on the CMC-2. However, positions 6 and 7 must be closed (left) for the unit to be operational.

SECTION 4.0

OPERATION

4.1 COMMUNICATION FORMAT

The CMC-0 controller communicates with the programming terminal or computer in one of two formats; non-echo and echo.

The non-echo mode is generally used when the CMC-0 must communicate with another computer. This mode minimizes transmission time because the controller does not echo any of the characters sent to it.

The echo mode is generally used when the CMC-0 must communicate with a user. In this mode the controller echoes each character that it receives so the user can see what he or she is sending.

4.1.1 Non-Echo Mode.

When the controller is set to the non-echo mode the format is as follows:

CaXXXnn<CR>(LF)

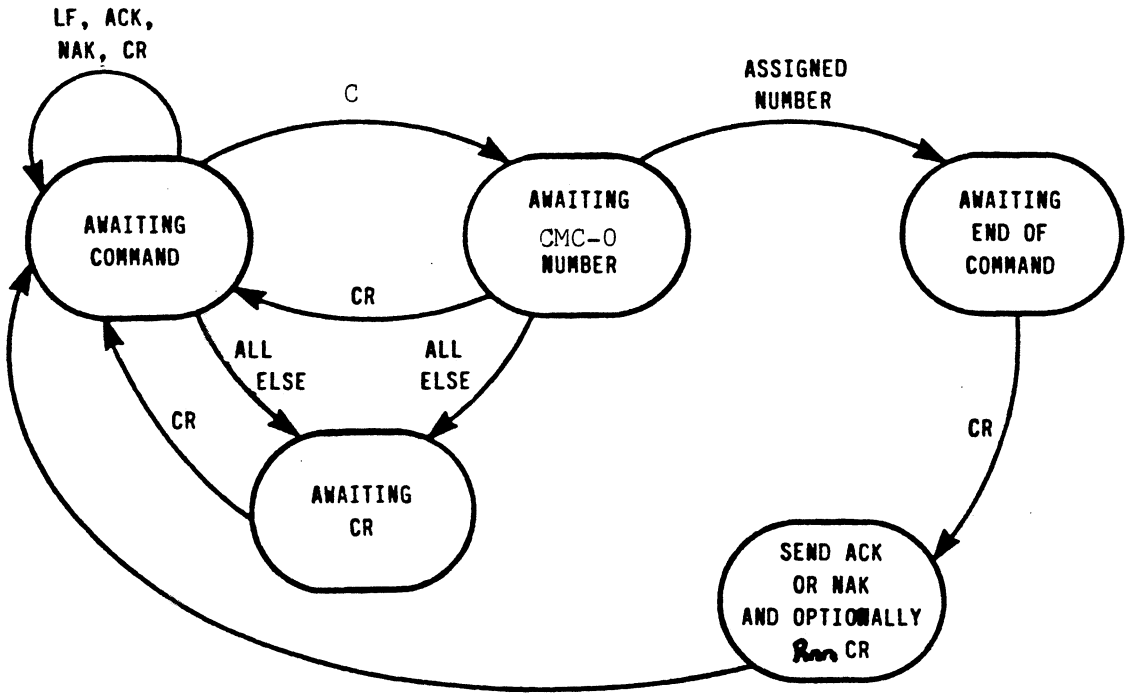
where: C = Controller.		Used to select CMC controllers from other devices on the serial line.
a = CMC-0 address.		Number from 0 to 8.
XXX = Command.		One of the available commands.
nn = Parameter.		Signed BCD parameter associated with the command.
<CR> = Carriage Return.		Terminating character.
(LF) = Line Feed.		Optional character.

Upon receiving the Carriage Return, the selected CMC-0 sends back the ASCII control code 06H (ACK) or 15H (NAK) based on whether or not it accepted the command. The device communicating with the CMC-0 must wait for the command acknowledgement before sending another command string. If the command requires a number to be sent back, the controller will send an "R" (for "response") followed by the signed BCD number. The controller will then send a Carriage Return to terminate the string.

When waiting for a new command string, the controller ignores all LF, ACK, NAK, and CR characters, and responds to all others. If the first character is other than a "C", the controller will wait for a Carriage Return before responding to characters again.

Figure 4-1 on the next page depicts the non-echo mode communication format.

Figure 4-1 Non-Echo Mode State Diagram



4.1.2 Echo Mode.

When the controller is set to the echo mode, the format is as follows:

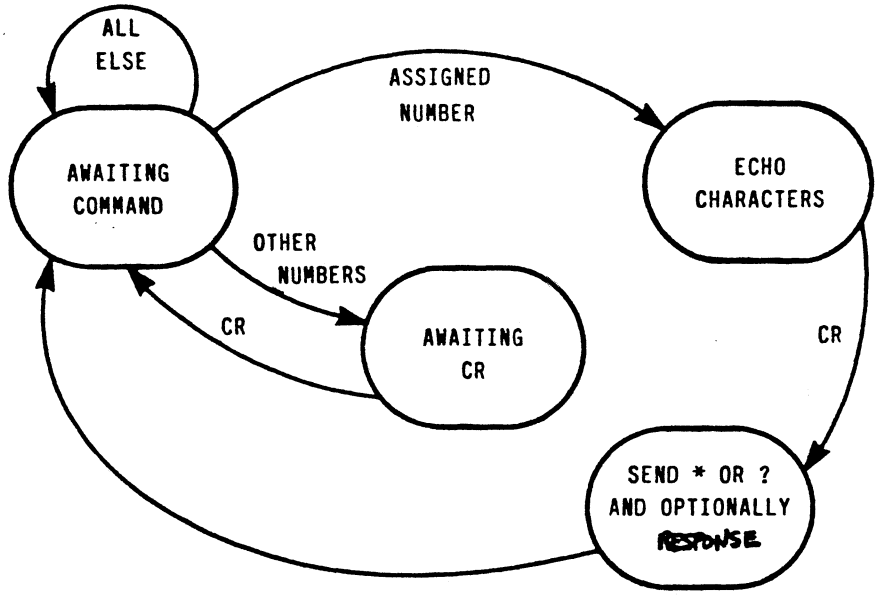
aXXXnn<CR>(LF)

where:	a	= CMC-0 address.	Number from 0 to 8.
	XXX	= Command.	One of the available commands.
	nn	= Parameter.	Signed BCD parameter associated with the command.
	<CR>	= Carriage Return.	Terminating character.
	(LF)	= Line Feed.	Optional character.

As each character is sent, the selected controller echoes the character, and after the <CR>, both a <CR> and (LF) are echoed even if an (LF) is not sent. The next character sent back from the controller is a "*" if the command is accepted and a "?" if it is not. The device communicating with the controller must wait for the command acknowledgement ("*" or "?") before sending another command string. If the command requires a response to be returned, the response is sent directly after the acknowledgement and is terminated with a <CR>(LF).

After receiving a <CR>, the controller will ignore all characters except numbers. If the number is not its assigned address or the Common Device Address (8), the controller will wait for a <CR> before responding to characters again. Figure 4-2 on the next page depicts the echo mode communication format.

Figure 4-2 Echo Mode State Diagram



In either mode, if a mistake is made when sending the command, type a "!" on the command line. If this character appears anywhere in the command line except as the first character, the controller will reject the command.

If you are using the echo mode and you make a mistake, the controller will allow you to backspace up to the controller's address number. Note that it does not allow you to erase the CMC-0 address character.

4.1.3 Common Device Address.

In either communication mode, the Common Device Address (8) can be used to communicate with all CMC-0 controllers connected to the same serial line.

To use this feature, simply connect the SYNC lines of the CMC-0 units together, and address one of the units at "0" with the address select DIP switches (see Section 2.10). The unit addressed at "0" will act as the "master" controller in this configuration, and will respond for all units when commands are sent.

Finally, use address 8 as the address for all commands which must go to all units simultaneously. For example, "8WB<CR>" would Warm Boot all CMC-0 units connected to the same serial line.

4.1.4 Controller Responses.

Upon receiving a command, the CMC-0 will always respond with either an acknowledgement (ACK in non-echo mode, "*" in echo mode) or a negative acknowledgement (NAK in non-echo mode, "?" in echo mode). In certain cases, the controller will send a number or response string in addition to the acknowledgement, e.g. after a position or status request, etc.

If the controller is in the echo communication mode, it will send one of the following messages after a negative acknowledgement ("?"):

NOT READY	Issued when the controller is busy executing a profile or other operation and cannot allow the command to execute. Typically only happens when using the "DE", "MS", and "ML" commands.
NOT OUTPUT	Issued when an "ST" or "RS" command is sent and the associated I/O line is not defined as an output.
BUFFER FULL	Issued when the command buffer is full and cannot accept any more commands. The buffer can hold up to 21 commands.
OUT OF RANGE	Issued when a parameter is entered which is too large or too small. Note that valid parameter ranges depend on the Unit Ratio and change as the Unit Ratio changes.
INVALID COMMAND	Issued when the controller does not recognize the command.
PROFILE TOO LONG	Issued during profile definition when the profile becomes too long. If this occurs, use the "GS" or "GT" command to direct the profile to another profile which picks up where the first one left off.
COMMAND NOT ALLOWED	Issued during profile definition when a command is entered which does not belong in a profile. For example, trying to load a control constant inside a profile definition will cause this reponse.
AXIS NOT AVAILABLE	Issued when there is no axis mapped at the requested address (W, X, Y, or Z).
AXIS IS NOT TALKING	Issued when the axis is not returning requested information to the host. This is generally due to problems with the link between the slave axis and the CMC-0.
AXIS IS NOT LISTENING	Issued when the axis is not responding to the CMC-0 during "ITa" initializations. This may be due to problems with the master-slave communication link, or an improperly mapped slave axis controller.

4.2 CONTROLLER INITIALIZATION

CMC system initialization consists of the following steps:

1. Loading all necessary control constants to each axis controller.
2. Initializing user I/O lines (if used).
3. Initializing analog channel deadband and offset (if used).
4. Clearing fault conditions.
5. Defining starting position for each axis.

These steps are described below in the following sections.

4.2.1 CMC-1 Control Constants (Stepper Axis)

Ten constants regulate operation of the CMC-1 Stepping Motor Controller. These are set up only once during the initialization procedure, but can be changed later if necessary. Table 4-1 lists the control constants and the commands which load them. These constants are discussed in detail in the following sections. Appendix C lists default values.

Note: In Table 4-1 below, the following notation applies:

a = axis (X, Y, W, or Z)
n = 8-bit value
nn = 16-bit value
nnn = 24-bit value
b = binary value (0 or 1)

Table 4-1 CMC-1 Control Constants

Control Constant	Range	Units	Load Command
Unit Ratio	1-10,000	pulses/unit length	URann
- Software O.T.	+/- 8,388,607	length units	NOannn
+ Software O.T.	+/- 8,388,607	length units	POannn
Encoder Ratio	256-65,535	N/A	ERann
Deadband	0-32,767	pulses	DBann
Deadband Fault	0 or 1	N/A	DFab
Correction Time	0-255	ms	CTan
Position Correction	0 or 1	N/A	PCab
Power Save	0 or 1	N/A	PSab
Register Wrap	0 or 1	N/A	WRab

4.2.1.1 Unit Ratio.

This constant allows you to specify the number of pulses per unit of distance. Internally, the controller represents distance in terms of pulses, but for programming simplicity the controller allows you to use other units such as inches or millimeters. Therefore, you must tell the controller how many pulses there are for each unit of distance by loading the Unit Ratio.

For example, suppose you have a 200 step/revolution motor attached to a linear slide with a lead screw of ten turns per inch. Also, assume that the controller is configured for half-stepping, so that there are effectively 400 pulses per revolution of the motor. If you were to use inches as units, then the correct unit ratio would be 4000 pulses/inch (400 pulse/rev * 10 revs/inch).

Note that this constant must be loaded before the software overtravel limits are loaded, or the limits (defined in length units) may be interpreted incorrectly.

4.2.1.2 Software Overtravel Limits.

Software overtravel limits allow you to define variable limits on motor motion using the position register. When one of these limits is encountered, the controller stops the motor, sets the Software Overtravel flag in the CMC-1 status register, and prohibits further motion in that direction. These are especially useful in linear slide applications where physical travel limitations exist.

4.2.1.3 Encoder Ratio.

The Encoder Ratio informs the controller of the ratio between the pulses which it generates and the pulses which it receives as position feedback. The number which should be entered can be determined by the following formula:

$$\frac{\text{Effective Motor Steps per Revolution}}{\text{Effective Position Pulses per Revolution}} * 4096$$

For example, a 200 step/revolution motor which is microstepped at 16 microsteps per motor step has (16 * 200) = 3200 effective motor steps per revolution. If it has a 1000 line per revolution encoder on it and the controller's position pulse multiplier is set to 2, then the effective number of position pulses per revolution is (2 * 1000) = 2000. Therefore, the correct Encoder Ratio is (3200/2000) * 4096 = 6554.

Note that the Encoder Ratio must always be 4096 when encoder feedback is not used.

When a ratio less than 4096 is used, a deadband value equal to or greater than the ratio between the Encoder Ratio used and 4096 will be necessary. The reason for this can be made clear by example. If a 200 step motor is used in half-step mode with a 512 line encoder and a position pulse multiplier of one, then the effective encoder pulses per motor pulse will be 1.28 (512/400), and the encoder ratio will be 3200.

Now suppose the encoder and motor line up precisely at position zero. The first step of the motor will make the encoder read one. The next step will make the encoder read two. The third step will make the encoder read three. However, the fourth motor step will make the encoder read five (4 * 1.28 = 5.12). It will be impossible for the controller to step the motor to a position which will make the encoder read four. Therefore, a deadband value of at least one will be required.

4.2.1.4 Disturbance Constants.

Four constants correct for system disturbances when using encoder feedback: Deadband, Deadband Fault, Position Correction, and Correction Time.

The Deadband constant defines the number of position pulses of error which the controller will allow before it sets the Deadband Flag in the Status Register. The Deadband Fault constant, if set to one, will cause the controller to fault when the Deadband Flag is set.

The Position Correction constant, if set to one, will cause the controller to automatically correct the position when the Deadband Flag in the Status Register is set. The Correction Time constant determines how quickly the controller reacts to final positioning error. For example, if the Correction Time constant is set to 5 ms and Position Correction is enabled, the controller will wait 5 ms after completing a move, read the position error, then correct for that error. It then waits another 5 ms before checking and correcting errors again. This sequence continues until no error exists.

Note that the Deadband Fault constant and the Position Correction constant should not both be set to one at the same time.

4.2.1.5 Power Save.

When the Power Save constant is set to one, the controller will reduce the current to the motor windings to 30% of its set value when the motor is at rest. This reduces the energy used by the motor and, therefore, the heating of the motor.

To use this constant, holding torque requirements should be about 30% or less of the maximum torque of the motor. When Power Save is enabled, the controller automatically returns the current to its set value before moving the motor and brings the current back down to 30% when the motor stops. If Power Save is not desired, the constant should be loaded with zero.

4.2.1.6 Register Wrap.

When the Register Wrap constant is set to one, the controller automatically wraps the absolute position register to prevent it from overflowing. This is useful in applications where the motor is continuously running in one direction for long periods of time.

4.2.1.7 Example Stepper Axis Initialization.

The following example should give you an idea of what a complete steppper axis initialization would look like (this one is for the X axis). Remember that control constants will vary from system to system, so the values given here may not always be appropriate. Also, note that none of the commands which load control constants are allowed in defined profiles.

Table 4-2 Sample Stepper Axis Initialization

Command		Description
4URX1000	<CR>	Load Unit Ratio = 1000 pulses/unit length
*4NOX-10	<CR>	Load negative software overtravel = -10 units
*4POX10	<CR>	Load positive software overtravel = +10 units
*4ERX4096	<CR>	Load Encoder Ratio = 4096 (no encoder)
*4DBX0	<CR>	Load deadband = 0
*4DFX0	<CR>	Disable deadband fault feature
*4CTX0	<CR>	Load correction time = 0
*4PCX0	<CR>	Disable position correction feature
*4PSX1	<CR>	Enable power save feature
*4WRX0	<CR>	Disable register wrap feature
*		

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

4.2.2 CMC-2 Control Constants.

Twelve constants regulate operation of the CMC-2 Servo Motor Controller. These are set up only once during the initialization procedure, but can be changed later if necessary. Table 4-3 lists the control constants and the commands which load them. These constants are discussed in detail in the following sections. Appendix C lists default values.

Note: In Table 4-3 below, the following notation applies:

a = axis (X, Y, W, or Z)
n = 8-bit value
nn = 16-bit value
nnn = 24-bit value
b = binary value (0 or 1)

Table 4-3 CMC-2 Control Constants

Control Constant	Range	Units	Load Command
Unit Ratio	1-10,000	pulses/unit length	URann
- Software O.T.	+/- 8,388,607	length units	NOannn
+ Software O.T.	+/- 8,388,607	length units	POannn
Encoder Line Density	50-40,000	lines/rev	LDann
Deadband	0-32,767	pulses	DBann
Integration Band	0-65,535	N/A	IBann
Position Band	0-32,767	length units	PBann
Pole	0-15	N/A	PLan
Gain	0-511	N/A	GNann
Zero	0-255	N/A	ZRan
Feedforward	0-255	N/A	FFan
Register Wrap	0 or 1	N/A	WRab

4.2.2.1 Unit Ratio.

This constant allows you to specify the number of encoder pulses per unit of distance. Internally, the controller represents distance in terms of encoder pulses, but for programming simplicity it allows you to use other units such as inches or millimeters. Therefore, you must tell the controller how many pulses there are for each unit of distance by loading the Unit Ratio.

For example, suppose you have a servo motor attached to a linear slide with a lead screw of five turns per inch. Also, assume that the encoder on the motor has a line density of 1000 lines per revolution, and the controller's position pulse multiplier is set at two, so that there are effectively 2000 pulses per motor revolution. If you were to use inches as units, then the correct unit ratio would be 10,000 pulses/inch (2000 pulses/rev * 5 revs/inch).

4.2.2.2 Software Overtravel Limits.

Software overtravel limits allow you to define variable limits on motor motion using the position register. When one of these limits is encountered, the controller stops the motor, sets the Software Overtravel flag in the CMC-2 status register, and prohibits further motion in that direction. These are especially useful in linear slide applications where physical travel limitations exist.

4.2.2.3 Encoder Line Density.

The encoder line density constant informs the controller of the number of position pulses the controller will receive for each revolution of the motor. The number to enter is the position pulse multiplier times the number of encoder lines per revolution of the motor. (The position pulse multiplier has a value of 1, 2, or 4 and is set with positions 9 and 10 of SW2 as described in section 3.8.)

4.2.2.4 Disturbance Constants.

The Deadband and Integration Band constants are used to control the controller's response to position disturbances. The deadband constant is the number of position pulses of error the controller will allow before it tries to correct the error. This constant is normally set to zero.

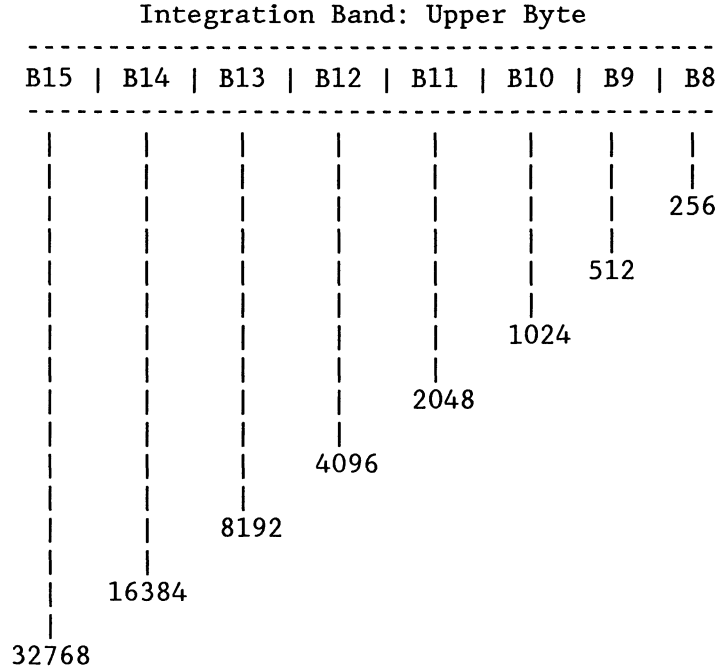
The Integration Band constant has two related uses:

1. To eliminate all final positioning error caused by mechanical friction and/or load torques.

2. To eliminate all final positioning error caused by offset drifts in the analog velocity loop.

The integrator constant is treated internally as two 8-bit bytes which have different meanings depending on how they are used.

In case 1 above, the lower byte is the number of position pulses from the final command position that the integrator turns on. It has an effective range of 0 to 255 pulses. The upper byte controls the speed of integration and ranges from zero (moderately fast) to fifteen (very fast).



For example, to set the integrator to a speed of one and a band of 20 pulses, load the Integration Band constant with (1 * 256) + 20 = 276. To set it to a speed of three and a band of 20 pulses, load the constant with (1 * 512) + (1 * 256) + 20 = 788.

This use of the Integration Band constant is most useful in systems that do not have a tachometer for velocity feedback, although it may also be useful in some systems which use tachometer feedback. When used in this mode, the controller turns on the integrator at the end of each move as soon as the position error is less than the integration band as defined by the contents of the lower byte. The integrator is turned off as soon as another move is initialized.

Note that when using the integrator in this way, bits B12 - B15 must always be zero since the speed range is from 0 to 15.

Case 2 described on the previous page is most useful in systems which use tachometer feedback and have drift (varying offsets) in the velocity loop.

When used this way, the integrator compensates for varying offsets over a long period of time, and does not reset from move to move. To use this mode, simply set bit B15 to 1 and load the appropriate value into the lower byte. The lower byte of the integrator constant defines how quickly the controller updates the integrator after each move. Its effective range is 0 to 255, with each unit representing 40 milliseconds of delay. In effect, the lower byte acts as a time constant for the integrator.

For example, to set the integrator to this mode with a delay of 480 milliseconds, load the constant with (1 * 32768) + 12 = 32780.

4.2.2.5 Position Band.

The Position Band constant controls when the Motor Stopped Flag is turned on after a move. The number loaded is the distance the motor position will be from the command position when the flag is set true. If the motor must be completely stopped before this flag is set, then a small number should be loaded. But if just knowing that the profile generator is finished is sufficient, then leave it at the default value.

4.2.2.6 Register Wrap.

When the Register Wrap constant is set to one, the controller automatically wraps the absolute position register to prevent it from overflowing. This is useful in applications where the motor is continuously running in one direction for long periods of time.

4.2.2.7 Compensation Constants.

The final four constants determine the pole, gain, zero, and velocity feedforward compensation of the motor control equation. The proper selection of these constants is very important to the overall system performance. The sections below describe how to select these constants for the two most common configurations.

Systems Without Tachometers

The first step in selecting the control constants is to start with the integrator off (zero), the pole at 0, the gain at 50, and the zero at 242. The feedforward constant must always be zero for tachless systems.

Observe the step response. If the motor is sluggish, increase the gain. If the motor tends to overshoot, increase the zero. If the motor is not stiff enough at zero velocity, increase the gain or decrease the zero. Note that the higher the gain, the higher the zero may have to be to compensate. A pole of up to 11 may also be useful. As the pole gets larger, the system response time should decrease but the motor may also start to overshoot.

When the dynamic response is satisfactory, set the Integration Band constant to 10 and see if the steady-state error goes away. If it does not, try larger values (up to 255). Increasing the integration speed by adding 256, 512, 768, or 1024 to the integrator band value will increase the speed at which the error is reduced and add stiffness to the motor response.

Systems With Tachometers

The first step in selecting the control constants is to start with the integrator off (zero), the pole at zero, the gain at one, the zero at zero, and the feedforward constant at zero.

Observe the step response. If the response is sluggish, increase the gain. If the motor is extremely nervous at standstill, then try decreasing the gain by adding 256 to the value used. This effectively divides the gain by 16. (For example, 257 = 1/16.) A pole of up to 11 may also be useful. As the pole gets larger, the system response time should decrease but the motor may also start to overshoot. The zero will almost always be zero when tachometer feedback is used. When the dynamic response is satisfactory, check the Following Error with the motor stopped. If it is more than one pulse, try using the integrator as described in section 4.2.2.4.

Velocity feedforward compensation can be used to compensate for the position following error at a constant velocity. This following error is a normal part of the controller's behavior and is not a problem unless the controller's motion is to be coordinated with another axis. (Note that the velocity error remains +/-0.01% whether or not feedforward compensation is used.) To compensate for the following error, slew the motor at 1000 RPM and check the error with the "FEa" command, where "a" is the axis letter. While the motor is slewing, load different values for the feedforward compensation until the error is within the desired bound.

Two commands aid in the selection of control constants. The Step Input command (mnemonic SN) offsets the command position of the motor by the amount specified. This gives a step input to the control equation. The response to this kind of input is the best indication of overall system behavior. Step inputs equal to about 1/4 motor revolution are usually adequate for this test. Never give a step input greater than one motor revolution as this will cause the controller to fault due to excessive following error.

See Table 4-4 for an example listing of a program which causes the motor to step back and forth every .5 seconds, and note that the step input value can be either positive or negative.

Table 4-4 Step Response Program

Command		Description
4DE1	<CR>	Define profile 1
*4SNX.2	<CR>	Issue step input command, .2 units
*4WTT.5	<CR>	Wait for .5 seconds
*4SNX-.2	<CR>	Issue step input command, -.2 units
*4WTT.5	<CR>	Wait for .5 seconds
*RM0	<CR>	Repeat profile (0 = infinite)
*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".

The other command useful in analyzing the performance of the system is the Following Error command (mnemonic FEa). It returns the difference between the command position and the actual motor position. This command will indicate if the integrator is doing its job and also provide numerical data on the step response.

4.2.2.8 Example Servo Axis Initialization.

The following example should give you an idea of what a complete servo axis initialization would look like (this one is for the X axis). Remember that control constants will vary from system to system, so the values given here may not always be appropriate. Also, note that none of the commands which load control constants are allowed in defined profiles.

Table 4-5 Sample Servo Axis Initialization

Command		Description
4URX5000	<CR>	Load Unit Ratio = 5000 pulses/unit length
*4NOX-10	<CR>	Load negative software overtravel = -10 units
*4POX10	<CR>	Load positive software overtravel = +10 units
*4LDX1000	<CR>	Load encoder line density = 1000 pulses/rev
*4DBX0	<CR>	Load Deadband = 0
*4IBX10	<CR>	Load Integration band = 10 pulses
*4PBX.1	<CR>	Load Position Band = 0.1 units
*4PLX0	<CR>	Load controller pole = 0
*4GNX1	<CR>	Load controller gain = 1
*4ZRX0	<CR>	Load controller zero = 0
*4FFX0	<CR>	Load controller feedforward = 0
*4WRX0	<CR>	Disable register wrap feature
*		

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

4.2.3 Automatic Control Constant Initialization.

The CMC-0 unit stores all axis control constants as they are entered, and automatically reissues them to the individual axes upon power up. Therefore, it is not necessary to reload the control constants every time power is cycled to the CMC system.

If at any time you wish to reinitialize an axis without cycling power to the system, simply issue the ITa command, where "a" is the axis you want to initialize. This causes the CMC-0 unit to reload all the control constants to that axis. If all you need to do is change one or two control constants, just reload those constants. The CMC-0 unit will load them to the axis as soon as the axis is ready (i.e. not executing profiles), and will store the new constants in place of the old ones for future power up initializations and ITa initializations.

4.2.4 Initializing User I/O Lines.

Initialization of the user I/O lines on the CMC-0 consists of defining the direction of the line (input or output). To do this, use the DRdb command, where "d" is the I/O line number, and "b" is a "1" for input, or "0" for output.

For example, DR10 defines I/O line 1 as an output. DRB1 defines I/O line 11 as an input. (There are twelve user I/O lines, each identified by a hex digit 1 through C. Therefore, A=10, B=11, and C=12.)

Note: To use the I/O lines for thumbwheel inputs, see Section 2.6.

4.2.5 Initializing Analog Channel Deadband and Offset.

Each analog channel has programmable deadband and offset values associated with it which allow you to compensate for imperfect inputs. The offset is useful for zeroing the channel when the input voltage is non-zero. Deadband is generally used to keep electrical noise from affecting the input. Each input has a +/- 10 VDC range and 12 bit resolution (+/- 2,047 A/D counts), so the effective resolution is 4.9 mV/count.

To initialize each channel used:

1. Apply voltage which corresponds to zero input to the channel's input. This is generally zero volts, but may be non-zero in certain applications.

2. Use the RAd command to read the analog input value.

3. Using the AOdnn command, load the negative of this value as the offset for the channel. This effectively zeroes the channel.

4. Use the RAd command to confirm that the channel is now reading zero.

5. Using the ADdnn command, set the channel's deadband to a value which will keep electrical noise from affecting the channel. A value of 5-10 will generally work well. Smaller values will be required if greater sensitivity is required.

4.2.6 Clearing Fault Conditions.

When power is first applied to the CMC system, each controller's Fault light will be on and current to the motors will be completely shut down. This condition must be cleared before the controller will move the motor(s). To do this, perform either of the following steps:

1. Warm Boot the system by issuing the WB command over serial I/O, or

2. Toggle the Enable Line on the X axis controller from the inactive state to the active state.

4.2.7 Defining Starting Position.

The CMC system is ready to execute movement commands and/or defined profiles as soon as it is warm booted. However, it is important for the motor to be at a known location after power-up, especially if physical travel limitations exist. For this reason, it is advisable to define the starting position with one of the "set position" commands each time the system is powered up. To do this, either:

1. Issue the appropriate "set position" command over serial I/O, or

2. Define a homing profile which uses one of the "set position" commands, and execute this profile after powering up the system (See Section 5.5).

4.3 OPERATING STATUS

4.3.1 Status Registers.

Each CMC controller has its own status register which the user can read. To read the status of the CMC-0 Master unit, simply issue the "RS" command. To read the status of one of the slave axis controllers (CMC-1 or CMC-2), issue the "RSa" command, where "a" is the axis (W, X, Y, or Z).

The controller's response to these commands will differ depending on the communication format used. If the controller is communicating in the "echo" mode, a bit string showing the state of each bit in the register is returned. For example, the bit string:

0000111100110111

would be the response from the CMC-0 controller if all four axes were stopped at their home positions (See Table 4-6). Note that the leftmost bit in the response is B15; the rightmost bit is B0.

When the controller is communicating in the non-echo mode, it returns a number which is the result of adding all the powers of two associated with each "true" flag in the register.

For example, the number corresponding to the bit stream response shown above is 1 + 2 + 4 + 16 + 32 + 256 + 512 + 1024 + 2048 = 3895.

The reason for using different responses for different communication modes is that the bit stream response is easier for operators to interpret, while the numerical response is easier for computers to decode.

Refer to the three tables which folllow for descriptions of each controller's status register contents.

Table 4-6 CMC-0 Status Register Flags

NUMBER	BIT	LABEL		DESCRIPTION
1	B0	AXES STOPPED	---	all axes stopped
2	B1	AXIS X HOME	---	X at home
4	B2	AXIS Y HOME	---	Y at home
8	B3	BUFFER FULL	---	command buffer full
16	B4	AXIS W HOME	---	W at home
32	B5	AXIS Z HOME	---	Z at home
64	B6	SET POINT A	---	set point A active
128	B7	SET POINT B	---	set point B active
256	B8	AXIS X STOPPED	---	X stopped
512	B9	AXIS Y STOPPED	---	Y stopped
1024	B10	AXIS W STOPPED	---	W stopped
2048	B11	AXIS Z STOPPED	---	Z stopped
4096	B12	SYSTEM FAULT	---	system fault detected ACTION: axes disabled
8192	B13	ENABLE LOST	---	enable line inactive ACTION: axes disabled
16384	B14	MEMORY LOST	---	memory check failed ACTION: Cold Boot executed
32768	B15	PROFILE	---	executing profile

The MEMORY LOST flag is set when power is first applied or when the Cold Boot command is issued. The Cold Boot command always causes this flag to be set. However, it is only set on power up if the memory does not checksum correctly. In any case, the Warm Boot command resets it.

Table 4-7 CMC-1 Status Register Flags

NUMBER	BIT	LABEL		DESCRIPTION
1	B0	MOTOR STOPPED	----	motor is not moving
2	B1	Reserved	----	system use only
4	B2	DIRECTION	----	direction is positive (reset when negative)
8	B3	Reserved	----	system use only
16	B4	HOME POSITION	----	motor is at user-defined home
32	B5	HOME INPUT	----	HOME input detected
64	B6	SET POINT A	----	CMC-1 set point A active
128	B7	SET POINT B	----	CMC-1 set point B active
256	B8	+ O. T.	----	positive overtravel detected ACTION: command position set equal to limit, motor stopped
512	B9	- O. T.	----	negative overtravel detected ACTION: command position set equal to limit, motor stopped
1024	B10	SOFT. O. T.	----	commanded position equals or exceeds user-defined OT limits ACTION: command position set equal to limit, motor stopped
2048	B11	DEADBAND	----	user-set deadband exceeded
4096	B12	SYSTEM FAULT	----	position register overflow, excessive current draw, enable line inactive, or other fault ACTION: motor disabled
8192	B13	ENABLE LOST	----	enable line inactive ACTION: motor disabled
16384	B14	INDEX PULSE	----	encoder index pulse detected
32768	B15	Reserved	----	system use only

Table 4-8 CMC-2 Status Register Flags

NUMBER	BIT	LABEL		DESCRIPTION
1	B0	MOTOR STOPPED	----	motor is not moving
2	B1	Reserved	----	system use only
4	B2	DIRECTION	----	direction is positive (reset when negative)
8	B3	Reserved	----	system use only
16	B4	HOME POSITION	----	motor is at user-defined home
32	B5	HOME INPUT	----	HOME input detected
64	B6	SET POINT A	----	CMC-1 set point A active
128	B7	SET POINT B	----	CMC-1 set point B active
256	B8	+ O. T.	----	positive overtravel detected ACTION: command position set equal to limit, motor stopped
512	B9	- O. T.	----	negative overtravel detected ACTION: command position set equal to limit, motor stopped
1024	B10	SOFT. O. T.	----	commanded position equals or exceeds user-defined OT limits ACTION: command position set equal to limit, motor stopped
2048	B11	Reserved	----	system use only
4096	B12	SYSTEM FAULT	----	position register overflow, excessive current draw, enable line inactive, or other fault ACTION: motor disabled
8192	B13	ENABLE LOST	----	enable line inactive ACTION: motor disabled
16384	B14	INDEX PULSE	----	encoder index pulse detected
32768	B15	Reserved	----	system use only

4.3.2 Status LEDs.

4.3.2.1 CMC-0 LEDs.

The CMC-0 controller has eight LEDs on its front panel which help facilitate system diagnosis. The "xmit" LED lights whenever the unit is communicating over its serial port. This helps to determine which unit is responding if more than one unit is attached to the same serial line. It also helps to determine if the serial interface is wired correctly.

The LED labeled "Fault" lights whenever the controller detects a problem. The problem can be a logic fault within the CMC-0 unit itself, cycling of power to the unit, loss of the Enable signal, or the presence of a fault condition in one of the controlled axes. If the problem is not major, the fault may be reset by issuing the Warm Boot (WB) command or by toggling the X axis Enable input from the inactive to the active state.

The remaining LEDs (three green, three yellow) are used with the external encoder inputs. For each input, the green LED indicates motion in the positive direction, while the yellow LED indicates motion in the negative direction.

4.3.2.2 CMC-1 and CMC-2 LEDs.

Each axis controller has five status LEDs on its front panel. The green LED labeled "xmit" should always be very faintly lit while the system is powered up. This indicates that the high-speed data link between it and the CMC-0 controller is intact.

The "fwd" and "rev" LEDs indicate motor direction and help determine the correct connection of the motor and encoder leads. For CMC-2 servo motor controllers and CMC-1 stepping motor controllers using encoder feedback, these LEDs indicate pulse activity at the encoder inputs. For CMC-1 stepping motor controllers operating without encoder feedback, these LEDs indicate outgoing motor pulses.

The bottom two LEDs on the CMC-1 and CMC-2 units indicate fault conditions. The bottom LED indicates that a general fault condition exists. This LED will light if the axis controller detects a problem such as a system fault, loss of its Enable input, logic malfunction, cycling of AC power, over-current condition (CMC-1), or over-temperature condition (CMC-2). If the fault is due to an over-current or over-temperature condition, then the other red LED will light as well. Short-circuit and over-temperature protections are discussed in Section 4.9.

4.4 READING PARAMETERS, POSITIONS, I/O, AND VARIABLES

4.4.1 Parameters and Integer Variables.

Controller parameters and integer variables may be queried by typing the parameter load command or variable name followed by a "?" and a carriage return. For instance, to determine the current value of the trajectory velocity for the XY axis pair, type the address number of the CMC-0 unit followed by "VX?<CR>", i.e "4VX?<CR>". To determine the current value of integer variable V1, type "4V1=?<CR>".

This method can be used to determine the present value of any parameter which does not have its own "read" command. For example, control constants, move times, move distances, radii, velocities, and Auxiliary Position Equation constants can all be read by using a "?" in conjunction with the load command.

4.4.2 Position.

The CMC-0 controller provides commands for reading command position, actual position, and following error (command - actual).

To read the command position, issue the "RCa" command, where "a" = W, X, Y, or Z. This will tell you where the controller currently wants the axis to be. To read the actual motor position, issue the "RPa" command. This will tell you where the motor actually is.

To read the following error, issue the "FEa" command. Obviously, if the command and actual positions are equal, the following error will be zero.

4.4.3 I/O, Boolean Variables, and Analog Inputs.

To read the current state of the twelve user I/O lines, issue the "RDI" command. To read the current state of the sixteen Boolean variables, issue the "RDB" command. In either case, the controller will return a bit pattern which shows the state of each line or variable, with the rightmost bit in the pattern being I/O line 1, or Boolean variable 0. If the controller is communicating in the non-echo mode, it will return a number equal to the sum of the "true" bits in the bit pattern.

To read the current value of one of the four analog inputs, issue the "RAd" command, where "d" is a digit 1-4. Keep in mind that the value reported is the sum of the "raw" value, offset, and deadband, not the raw value itself.

4.5 DIAGNOSTIC MODE

The CMC-0 controller has a diagnostic feature which allows you to view the state of the system at any time. To enable this feature, simply send the "DG1" command preceded by the proper unit address. After this feature is enabled, the controller will automatically send out a diagnostic response line whenever:

- The CMC-0 executes a new profile
- A CMC-0 set point changes state
- A WAIT command is satisfied

The controller will also send out diagnostic information in response to the "X" command. For example, to receive diagnostic information when a profile is not executing, simply send the "X" command along with the correct address, i.e. "4X<CR>".

Note: This feature should only be enabled for system verification and troubleshooting. Always disable this feature when returning to normal operation.

The diagnostic response line looks like this:

```
*PX      I/O        ABI      BOOLEAN        POSX  POSY  POSW  POSZ
  n  bbbbbbbbbbbb  bbb  bbbbbbbbbbbbbbb    nnn   nnn   nnn   nnn
```

The first field (PX) tells which profile caused the line to be printed. If the line was printed due to something other than a profile executing, then the number returned is zero.

The second field (I/O) shows the states of the user I/O lines. The numbers returned for each line will either be "0" for inactive, or "1" for active. The rightmost bit corresponds to I/O line 1, the leftmost bit corresponds to I/O line C (12).

The third field (ABI) shows the states of CMC-0 set points A and B, and also the state of the CMC-0 index pulse. The numbers returned will either be "0" for inactive, or "1" for active.

The fourth field (BOOLEAN) shows the states of the user Boolean variables. Again, the numbers returned will either be "0" for inactive, or "1" for active. The rightmost bit corresponds to B0, the leftmost bit corresponds to BF (15).

The last four fields give the absolute positions (in units) of the four axes.

Note that this feature only works when the controller is communicating in the echo mode. To disable it, send the "DG0" command along with the proper unit address, i.e. "4DG0<CR>".

4.6 STAND-ALONE MODE

The CMC Series Controllers are capable of stand-alone operation. In this mode, the controller executes a previously defined profile as selected by the profile inputs on the axis controllers. A description of these inputs is given below.

Table 4-9 Parallel Profile Allocation

AXIS	NON-ENCODED	ENCODED
X	1 - 8	1 - 127
Y	9 - 16	129 - 255
W	17 - 24	1 - 127
Z	25 - 32	129 - 255

Note: The eight profile input lines on each axis controller are the six numbered profile lines and the "t-one" and "t-two" inputs. These inputs are the seventh and eighth profile lines, respectively.

Notice that the profile input lines may be either non-encoded or encoded. When the inputs are non-encoded, the unit executes the defined profile corresponding to the profile line which becomes active. For example, if the line labeled "t-two" becomes active on the Y axis controller, the CMC-0 will execute defined profile 16.

A profile will not repeat due to the profile line being left in the active state. The input must be toggled from the inactive state to the active state in order for the profile to execute.

When the inputs are encoded, the defined profile corresponding to the binary representation on profile lines 1-6 and t-one is executed when t-two becomes active. In this 7-bit code, t-one acts as the MSB, and profile line one acts as the LSB.

For example, if the six profile lines and the t-one line of axis W are all active, the CMC-0 will execute profile number 127 when t-two of axis W becomes active. If this same sequence is performed on axis Z, then the CMC-0 will execute profile 255. In the encoded mode, the seven profile lines which make up the code are level sensitive, while the t-two line is sensitive to the inactive to active transition.

To enable stand-alone operation (either encoded or non-encoded mode), issue the Profile Enable (PE) command with a value of 1, i.e. PE1. To disable stand-alone operation, reset the toggle with the PE0 command.

See section 3.8 for instructions on how to configure an axis for encoded or non-encoded operation.

4.7 ENABLE LINES

Each axis controller in a CMC system has an Enable input on I/O connector P1. Each of these must be active for the system to be operational. If any of these become inactive during system operation, the CMC-0 controller will fault, and the outputs of the axis controllers with their Enable lines inactive will shut down.

To clear this condition, toggle the X axis Enable line from the inactive to the active state. This will Warm Boot the system, clearing all faults in the process.

Note that only the X axis Enable line needs to be toggled to Warm Boot the system, but the Enable lines on all the axis controllers must be active for the system to be operational.

4.8 MEMORY TRANSFER

The Memory Save (MS) and Memory Load (ML) commands allow the entire memory contents of one CMC-0 controller to be transferred to another CMC-0 controller or to a disk-based computer for storage. (Whedco has an IBM PC-compatible support program available which allows memory transfers to disk.) All transfers should be done with the X axis Enable Line inactive.

To do a direct transfer, follow the steps given below.

1. Send each CMC-0 unit which will be receiving the dump the "ML" command.

2. Send the unit which is transmitting its memory the "MS" command.

3. Disconnect the transmit lines of each receiving CMC-0 unit.

4. Connect the receive lines of each receiving unit to the transmit line of the sending unit.

5. At this point, send the ASCII control code 02H (STX) to the sending unit. The transfer will now proceed.

At the end of the transfer, each receiving unit will compare the sent checksum with its own checksum. If the checksum is correct, each unit's memory contents should be identical to that of the sending unit. If the checksum is incorrect, the unit will execute a Cold Boot command.

4.9 SHORT CIRCUIT AND OVER TEMPERATURE PROTECTION

4.9.1 CMC-1 Short Circuit Protection.

All CMC-1 Stepping Motor Controllers are protected against short circuiting. In the event that any motor output terminal is shorted to another or shorted to ground, the CMC-1 will fault, set the System Fault flag in the Status Register, light both the over-current and the Fault LEDs, and disable the drive. This will also cause the CMC-0 unit to fault. To clear this condition, issue the Warm Boot command (WB) or toggle the X axis Enable line from inactive to active. If the fault persists, disconnect power to the unit and recheck the motor wiring (see Section 3.4).

Once the short has been corrected, reapply power to the unit. If the fault still occurs, contact Whedco Customer Service for assistance.

4.9.2 CMC-2 Over-temperature Protection.

All CMC-2 Servo Motor Controllers are protected against operation at excessively high temperatures. In the event that the heat-sink temperature exceeds 80 degrees Celsius, the CMC-2 will fault, set the System Fault flag in the Status Register, light both the over-temperature and the Fault LEDs, and disable the servo amplifier (CMC-2 units without amplifiers do not have this function.) This will also cause the CMC-0 unit to fault.

To clear the fault, issue a Warm Boot (WB) command or toggle the X axis Enable line from inactive to active. If the fault persists, investigate air blockage or improper installation of the unit (see Section 3.1).

4.10 WARNINGS

1. The 5 volt supply on the CMC-1 and CMC-2 units provides 250 mA maximum and is internally limited.

2. The 12 volt supply on the CMC-1 and CMC-2 units provides 250 mA maximum and is internally limited.

3. Voltages of up to 400 VDC can be present on some of the components inside the CMC controllers. Do not attempt to troubleshoot or repair these units with power applied.

4. If encoder feedback or tachometer feedback is lost, the axis overtravel inputs may not stop the motor. However, when the Enable line is made inactive, the power output stage of units with amplifiers is always disabled, which removes power to the motor.

SECTION 5.0

PROGRAMMING

5.1 INTRODUCTION

The CMC-0 allows you to initiate motion profiles interactively or by defining them first and executing them later. When programming in the interactive mode, you must first load the appropriate parameters (i.e distance, trajectory velocity, etc.), then issue the appropriate movement command(s). The profile will begin as soon as you enter the first movement command.

Programming defined profiles is essentially the same, except that the parameter load commands and movement commands are stored under a specific profile number. They are not executed when you type them in. To execute a profile defined in this way, simply issue the "GTn" command, where "n" is the number of the profile which contains the stored commands.

Before you actually program a CMC system, you should understand how the controller prioritizes its tasks. The CMC-0 controller divides its duties between two control loops:

The outer loop, called the "foreground", takes care of general housekeeping functions like monitoring profile lines and user inputs, etc. It also interprets commands from the serial port. The amount of time it takes to complete this loop varies depending on how much the controller needs to do. The controller typically takes no more than 10 ms to complete the outer loop.

The inner loop, called the "background", interrupts the foreground loop every five milliseconds in order to perform important control functions on a real-time basis. From a system programmer's standpoint, it may be important to know what is and what is not executed/updated continuously in the background control loop.

The following are updated continuously in the background loop:

- External encoders
- Analog inputs
- Profile 128
- Axis command positions
- Set points
- Auxiliary Position Equations

Profile 128 is a special profile which can be used for Phase-Locked Loop (PLL) control and the Auxiliary Position Equation. It can only contain integer variable operations and load commands for the Auxiliary Position Equation multiplier. See the sections on integer variables and PLL control for more information about Profile 128.

Note that integer variable operations are not updated every five milliseconds. These are only updated when the profiles containing them are executed.