USER MANUAL

DMC-31xx Supplement

Manual Rev. 1.0

By Galil Motion Control, Inc.

Galil Motion Control, Inc.
270 Technology Way
Rocklin, California 95765
Phone: (916) 626-0101
Fax: (916) 626-0102
Internet Address: support@galilmc.com
URL: www.galilmc.com

Rev 2/04

Using This Manual

This user manual provides information for proper operation of the DMC-31x2 and DMC-31x3 controllers. A separate supplemental manual, the Command Reference, contains a description of the commands available for use with this controller.

Note: The DMC-31x2 and DMC-31x3 controllers are identical except the DMC-31x2 has 100 pin high-density connectors for breaking out the signals and the DMC-31x3 has 96 pin DIN connectors for breaking out the signals. The ICM/AMP-1900 and the ICM-2900 do not interface to the DMC-31x3. Look in the appendix of the complete users manual for the controller pinouts.

Your DMC-31x2/31x3 motion controller has been designed to work with both servo and stepper type motors. Installation and system setup will vary depending upon whether the controller will be used with stepper motors or servo motors. To make finding the appropriate instructions faster and easier, icons will be next to any information that applies exclusively to one type of system. Otherwise, assume that the instructions apply to all types of systems. The icon legend is shown below.

Attention: Pertains to servo motor use.

Attention: Pertains to stepper motor use.

The DMC-31x2 and 31x3 controllers use identical hardware to the DMC-21x2 and DMC-21x3 controllers. This supplement contains information for setting up the firmware features contained in the controller to allow distributed control. The examples contained in the DMC-21x2 and DMC-21x3 manual still pertain to the DMC-31xx controllers. Please refer to the DMC-21x2 and DMC-21x3 user manual for complete operation of the controller. This supplement only contains differences due to the distributed nature of the product.

WARNING: Machinery in motion can be dangerous! It is the responsibility of the user to design effective error handling and safety protection as part of the machine. Galil shall not be liable or responsible for any incidental or consequential damages.

Contents

Ethernet Configuration

Communication Protocols

The Ethernet is a local area network through which information is transferred in units known as packets. Communication protocols are necessary to dictate how these packets are sent and received. The DMC-31xx supports two industry standard protocols, TCP/IP and UDP/IP. The controller will automatically respond in the format in which it is contacted.

TCP/IP is a "connection" protocol. The master must be connected to the slave in order to begin communicating. Each packet sent is acknowledged when received. If no acknowledgement is received, the information is assumed lost and is resent.

Unlike TCP/IP, UDP does not require a "connection". This protocol is similar to communicating via RS232. If information is lost, the controller does not return a colon or question mark. Because the protocol does not provide for lost information, the sender must re-send the packet.

Ethernet communication transfers information in ‘packets’. The packets must be limited to 470 data bytes or less. Larger packets could cause the controller to lose communication.

NOTE: In order not to lose information in transit, Galil recommends that the user wait for an acknowledgement of receipt of a packet before sending the next packet.

Addressing

There are three levels of addresses that define Ethernet devices. The first is the Ethernet or hardware address. This is a unique and permanent 6 byte number. No other device will have the same Ethernet address. The DMC-31xx Ethernet address is set by the factory and the last two bytes of the address are the serial number of the controller.

The second level of addressing is the IP address. This is a 32-bit (or 4 byte) number. The IP address is constrained by each local network and must be assigned locally. Assigning an IP address to the controller can be done in a number of ways.

The first method is to use the BOOT-P utility via the Ethernet connection (the DMC-31xx must be connected to network and powered). For a brief explanation of BOOT-P, see the section: *Third Party Software*. Either a BOOT-P server on the internal network or the Galil terminal software may be used. To use the Galil BOOT-P utility, select the registry in the DMC Smart Terminal or the DMC Net Utility. If you open the registry, click the “Find Ethernet Controllers” button. After your controller has been found, click the button to assign an IP address. After the IP address has been successfully defined, highlight the controller and click the “Assign” button to add the controller to the registry. Close the window, and then select the controller in the registry. Click the properties button and then select the “Ethernet Parameters” tab. This tab will show you the various options of connection via Ethernet (TCP/IP or UDP/IP). It will also give various options regarding how you would like to receive unsolicited messages. Next enter the terminal and type in BN to save the IP address to the controller's non-volatile memory. A full description of addressing the card may be found in Chapter 2 Getting Started.

CAUTION: Be sure that there is only one BOOT-P server running. If your network has DHCP or BOOT-P running, it may automatically assign an IP address to the controller upon linking it to the network. In order to ensure that the IP address is correct, please contact your system administrator before connecting the controller to the Ethernet network.

The second method for setting an IP address is to send the IA command through the DMC-31xx main RS-232 port. The IP address you want to assign may be entered as a 4 byte number delimited by commas (industry standard uses periods) or a signed 32 bit number. (Ex. IA 124,51,29,31 or IA 2083724575) Type in BN to save the IP address to the controller's non-volatile memory.

NOTE: Galil strongly recommends that the IP address selected is not one that can be accessed across the Gateway. The Gateway is an application that controls communication between an internal network and the outside world.

The third level of Ethernet addressing is the UDP or TCP port number. The Galil controller does not require a specific port number. The port number is established by the client or master each time it connects to the controller.

Ethernet Handles

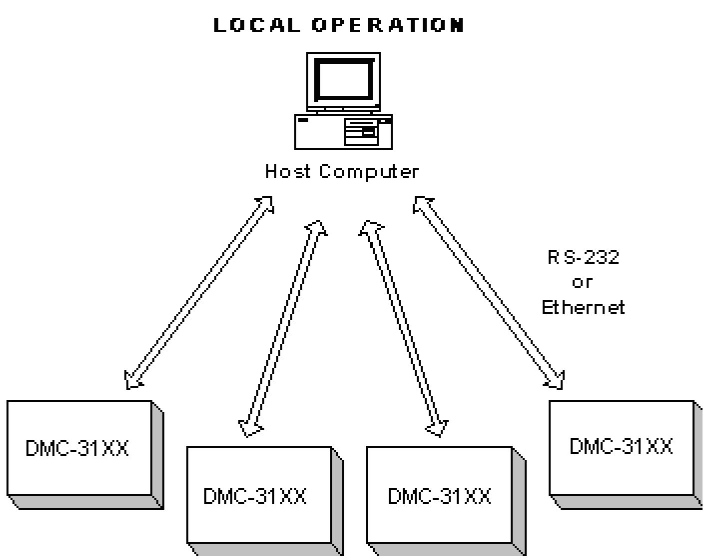
An Ethernet handle is a communication resource within a device. The DMC-31xx can have a maximum of 8 Ethernet handles open at any time. When using TCP/IP, each connection to a device, such as the host computer, requires an individual Ethernet handle. In UDP/IP, one handle may be used for all the masters, but each slave uses one. (Pings and ARP's do not occupy handles.) If all 8 handles are in use and a 9th master tries to connect, it will be sent a "reset packet" that generates the appropriate error in its windows application.

The TH command may be used to indicate which handles are currently connected to and which are currently free.

Global vs Local Operation

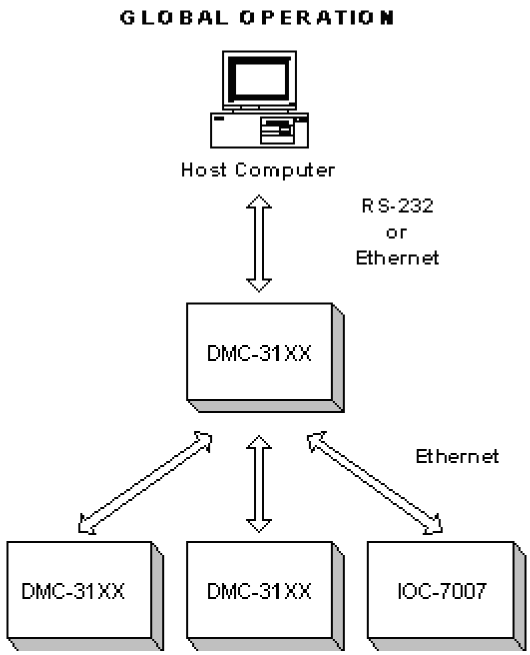
Each DMC-31xx controls one to seven axes of motion. The host computer can communicate directly with any DMC-31xx using an Ethernet or RS-232 connection. When the host computer is directly communicating with any slave DMC-31xx, all commands refer to the local axes beginning with A (X). Direct communication with the DMC-31xx is known as LOCAL OPERATION.

The concept of Local and Global Operation also applies to application programming.



The DMC-31xx supports Galil's Distributed Control System. This allows a combination of DMC-31xx's to be connected together as a single virtual 8-axis controller. In this system, one of the controllers is designated as the master. The master can receive commands from the host computer that apply to all of the axes in the system.

A simple way to view Local and Global Operation: When the host communicates with a slave controller, it considers the slave as a local master controller. When the host communicates with a master, it acts as a global multi-axis controller. Similarly, an application program residing in a slave controller deals only with local motors such as A & B. An application program in a master deals with all motors referenced as A through H.



The controllers may operate under both Local and/or Global Mode. In general, operating in Global Mode simplifies controlling the entire system. However, Local Mode operation is necessary in some situations; using Local Mode for setup and testing is useful since this isolates the controller. Specific modes of motion require operation in Local Mode. Also, each controller can have a program, including the slave controllers. When a slave controller has a program, this program would always operate in Local Mode. The distributed system works by getting periodic updates from the slave controllers. The update rate is set with the HC command. A complete listing of local and global commands can be found at the end of this chapter.

Operation of Distributed Control

For most commands it is not necessary to be conscious of whether an axis is local or remote. For instance to set the KP value for the A and C axes, the command to the master would be

```
KP 10,,20
```

Similarly, the interrogation commands can also be issued. For example, the position error for all axes would be TE. The position operand for the F axis would be_TPF.

Some commands inherently are sent to all controllers. These include commands such as AB (abort), CN and TM. In addition, the * may be used to send commands to all controllers. For example

```
SP*=1000
```

will send a speed of 1000 cts/sec to all axes. This syntax may be used with any configuration or parameter commands.

Certain commands need to be launched specifically. For this purpose there is the SA command. In its simplest form the SA command is

SAh= "command string"

Here "command string" will be sent to handle h. For example, the SA command is the means for sending an XQ command to a slave/server. A more flexible form of the command is

SAh= field1,field2,field3,field4 ... field8

where each field can be a string in quotes or a variable.

For example, to send the command KI,,5,10; Assume var1=5 and var2=10 and send the command:

SAF= "KI",var1,var2

When the Master/client sends an SA command to a Slave/server, it is possible for the master to determine the status of the command. The response _IHh4 will return the number 1 to 4. One means waiting for the acknowledgement from the slave. Two means a colon (command accepted) has been received. Three means a question mark (command rejected) has been received. Four means the command timed out.

If a command generates responses (such as the TE command), the values will be stored in _SAh0 thru _SAh7. If a field is unused its _SA value will be -2^31.

Configuring the Distributed Network

A multi-axis distributed control system may be composed of DMC-31xx motion controllers along with the IOC-7007 I/O controller. Before you configure the distributed system, you should choose which DMC-31xx motion controller you would like to designate as the master controller. This controller will handle the communication between the other controllers in the system to begin the appropriate motions, and set the proper I/O bits. The master controller may be connected to from the host either serially with RS-232 or Ethernet with UDP, or TCP. It is also possible for the master controller to operate in a standalone mode. An IP address must be assigned to the master before the master can configure the network either standalone or through the host. Up to 8 handles of communication may be connected to and from the master controller. If the connection is made over TCP with TCP mixed with UDP, 2 handles will be used for the connection to each slave.

Master Controller Configuration

The first step required to set up the master is to give the master controller an IP address. The IP address may be assigned with the IA command serially, or by using the “Find Ethernet Controllers” button in the Galil registry editor. The master controller may communicate with the host over serial or Ethernet. It is also possible for the master to operate in a stand alone configuration after the distributed network has been completely configured. After the master has been assigned an IP address, it is possible to proceed configuring the remaining slave controllers to operate in a distributed manner. The user must first know the serial numbers of the slave controllers, and the number of axes located on each of the master and slave axes. The number of axes of the master may be queried on the master controller with the ^R^V command before configuring any slaves. The ^R^V command will return the number of the controller DMC-31xx. The third number in the model will tell you the number of axes. For example if the response from ^R^V is DMC-3123, then you have a master with two axes of motion control.

Querying for Slave Controllers

If the serial numbers and corresponding available axes of each slave are not known, then the HQ command may be issued to search for motion controllers and I/O controllers without IP addresses. To read the results of the HQ command, issue HQ? to the master controller. This will return the controller type, number of motion axes available for the distributed network, and the serial number. The controller types are 1 for motion controllers and 255 for the IOC-7007. A motion controller may be found with the HQ command that can not be configured for distributed control. In this case, the number of axes available will display as 0. The IOC-7007 will also show that there are 0 axes available as it is purely an I/O controller. The serial numbers of the found controllers will also be returned.

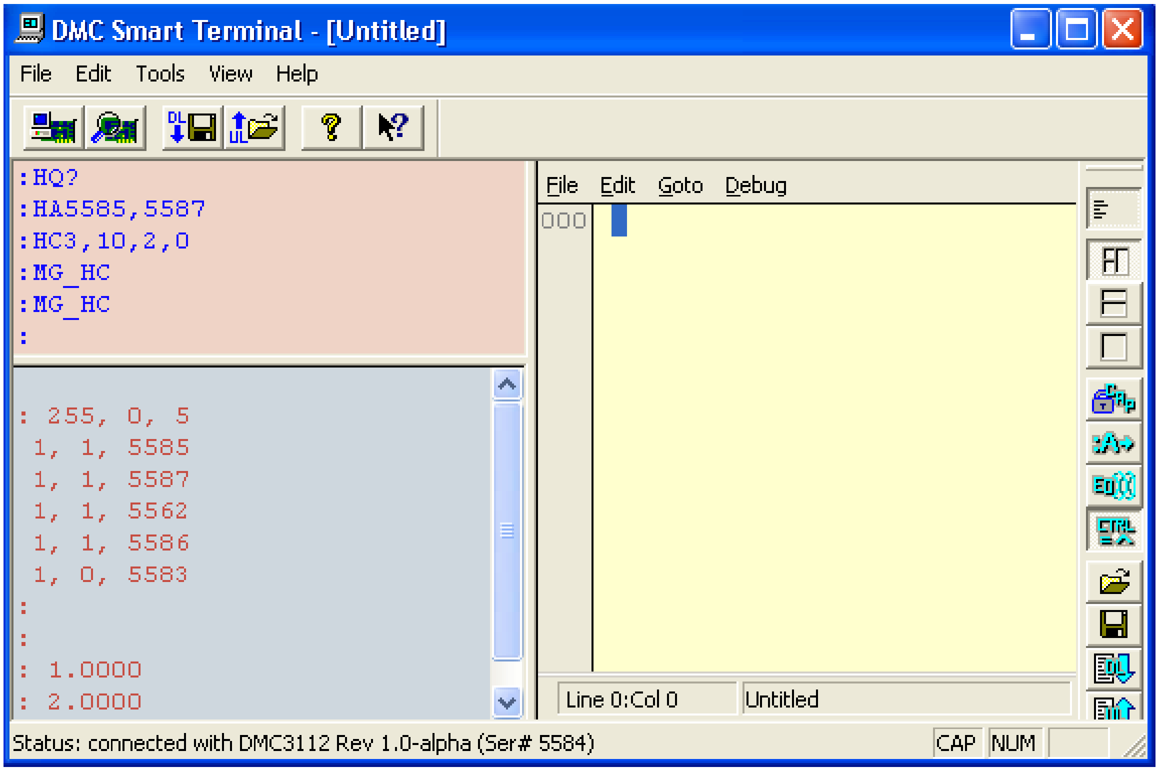


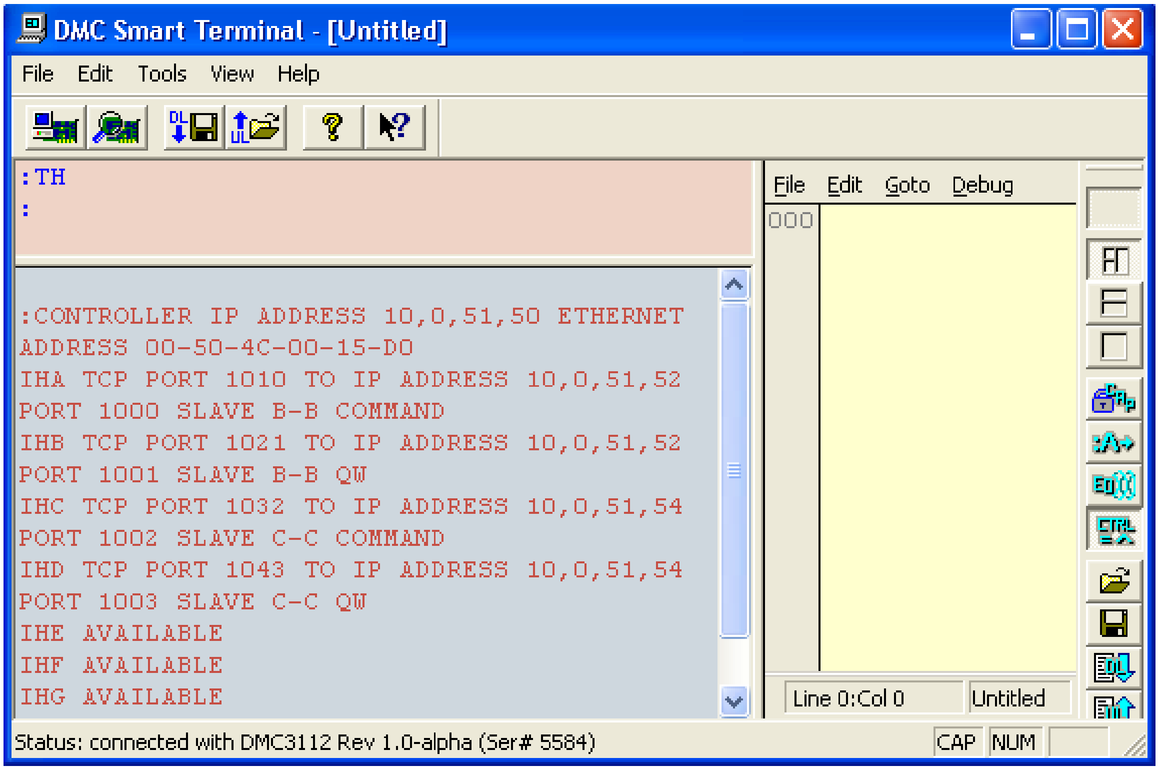
Figure 1- Examples of HA, HQ, and HC

Configuration order of the Slave Controllers

Before the slave controllers can be configured with the master, the master must know the order in which the axes will be addressed. If an IOC-7007 is located in the system, its order should be configured with jumpers located on the IOC-7007. It will not affect the axis selection of the motion axes, but its order of configuration will dictate which Ethernet handles it will use for sending updates to the master controller. For example, the master controller is a single-axis card. There are two single-axis slave controllers, serial no. 550 and serial no. 5501. HA5500, 5501 will set the virtual B-axis to controller 5500 and the virtual C-axis to controller 5501.

Assigning Slave Addresses

The final step in configuring the distributed network is setting up the total number of axes in the distributed system, the frequency of updates between the master and slave controllers, the type of connection (TCP or UDP), and the number of IOC-7007 controllers in the system. These operations are configured with the HC command. To continue the previous example, if there are to be three axes in the distributed system with the slave controllers sending updates to the master controller every 10 milliseconds over TCP/IP. The HC command setting would be HC 3,10,2,1. Since the communication is over TCP, each slave controller will have two handles of communication open to the master. The first handle is for sending commands and responses between the master and slave controllers. The second handle is used to send the data update between the slave and the master. In this example, the controller with serial number 5500 will communicate on handles A and B, 5501 on C and D, and 5503 on handle E. The three remaining handles could be used for communication with the host, or for connection to some other device. After the devices have been configured, the status of handles connected to the controller may be queried with the TH command.



Accessing the I/O of the Slaves

The I/O of the server/slaves is settable and readable from the master. The bit numbers are adjusted by the handle number of the slave controller. Each handle adds 100 to the bit number. Handle A is 100 and handle H is 800. In a TCP/IP control setup with two handles per slave; it is imperative that you send commands to the first handle designated as the "command" handle. In a UDP system, the single handle per slave is used to address the I/O. For the IOC-7007, each handle adds 1000 to the bit number. To set bit 61 if you are communicating on the C handle, the command would read SB3061.

The command TZ can be used to display all of the digital I/O contained in a distributed control system. Specific slave controllers may be queried by issuing TZn where n is the specific Ethernet handle. Any IOC-7007's configured using the HC command will not be displayed with the TZ command. See the Command Reference for more information on the TZ command.

Digital Outputs

For outputs, the SB and CB commands are used to command individual output ports, while the OP command is used for setting bytes of data. The SB and CB commands may be set globally through the master, while the OP command must be sent to the slave using the SA command.

Outputs may be set globally according to the following numbering scheme: Bitnum = (Slave Handle * 100) + Output Bit. For I/O located on motion controllers. For I/O located on the IOC-7007, Bitnum= (Slave Handle * 1000) + Output Bit.

Set Bit 2 on a UDP distributed slave using the E handle for communication. The E handle would have a numerical value of 500, plus the bit number of 2. The command would therefore become SB502.

Specific outputs in a distributed system may be read by using the @OUT[n] function, where n is the corresponding bit number as defined above.

Output bits on an IOC-7007 may also be set through the master controller in a distributed network. Please refer to the IOC-7007 Manual for information on setting and reading these I/O points.

Digital Inputs

Digital inputs may be addressed individually using the @IN[n] function, or in blocks using the TI command. Both of these commands may be sent globally to the controller. The ‘n’ in the @IN[n] function operates identically to the SB/CB syntax. This means that a specific input bit is referenced as the slave handle number * 100 plus the input bit. The IOC-7007 is referenced by slave handle number * 1000 plus the input bit.

Read input bit 4 on a TCP/IP distributed slave using the C handle for communication. The C handle in this case would give a value of 300. Therefore, to read bit 4, the command would be MG@IN[304]. The MG in this case simply displays this data to the terminal.

The TI command may be used to read all inputs on a slave in blocks of 8. This is helpful if the slave controller in question has a DB-28040 expanded I/O daughter card. The TI command uses the slave handle number * 100 plus the block number to be read. The block number is only used if the controller has the DB-28040 expansion option.

Inputs on an IOC-7007 may also be read through the master controller in a distributed network. Please refer to the IOC-7007 Manual for information on setting and reading these points.

Analog Inputs

Each DMC-31xx controller may have eight 12-bit analog inputs if the DB-28040 has been added. These inputs are read with the command @AN[n], where n is the input to be read n may be calculated by the handle number * 100 plus the bit number for motion controllers and handle number * 1000 plus the bit number for the IOC-7007.

Handling Communication Errors

A new automatic subroutine which is identified by the label #TCPERR, has been added. If a controller has an application program running and the TCP or UDP communication is lost, the #TCPERR routine will automatically execute. The #TCPERR routine should be ended with a RE command. In the UDP configuration, the QW commands must be active in order for the #TCPERR routine on the master to operate properly.

Multicasting

A multicast may only be used in UDP and is similar to a broadcast, (where everyone on the network gets the information) but specific to a group. In other words, all devices within a specified group will receive the information that is sent in a multicast. There can be many multicast groups on a network and are differentiated by their multicast IP address. To communicate with all the devices in a specific multicast group, the information can be sent to the multicast IP address rather than to each individual device IP address. All Galil controllers belong to a default multicast address of 239.255.19.56. The controller's multicast IP address can be changed by using the IA> u command.

The Galil Registry has an option to disable the opening of the multicast handle on the DMC-31xx. By default this multicast handle will be opened.

Unsolicited Message Handling

Anytime a controller generates an internal response from a program, generates an internal error or sends a message from a program using the MG command, this is termed an unsolicited message. There are two software commands that will configure how the controller handles these messages; the CW and the CF command.

The DMC-31xx has 8 Ethernet handles as well as 1 serial port where unsolicited messages may be sent. The CF command is used to configure the controller to send these messages to specific ports. In addition, the Galil Registry has various options for sending this CF command. For more information, see the CF command in the DMC-21x3 Command Reference. The MG can also send the message to a specific handle using the MG{Eh} syntax, where h is the handle. See the MG command in the Command Reference for more information.

The CW command has two data fields that affect unsolicited messages. The first field configures the most significant bit (MSB) of the message. A value of 1 will set the MSB of unsolicited messages, while a value of 2 suppresses the MSB. The majority of software programs use a setting of CW2, although the Galil Smart Terminal and WSDK will set this to CW1 for internal usage. If you have difficulty receiving characters from the controller, or receive garbage characters instead of messages, check the status of the CW command for a setting of CW2.

IOC-7007 Support

The IOC-7007 is an Intelligent Ethernet I/O controller that can be programmed in standard Galil language. This module allows various configurations of TTL inputs, opto-isolated inputs, high power outputs and relay switches to be used in the Galil distributed motion system. Each IOC-7007 may be populated by up to seven IOM I/O modules.

The IOC-7007 Ethernet I/O controller may be used in a distributed system and commanded by the master controller. The HC command is used to specify total number of IOC-7007 controllers within that distributed system. Once configured, the I/O of that IOC-7007 becomes incorporated in the distributed system, much the same as board level I/O of the DMC-31xx slaves.

Inputs of the IOC-7007 are read using the standard @IN[n] and TI commands as follows:

@IN[n] where n is the IOC-7007 input bit to be read. n is calculated with the equation n = (HandleNum * 1000) + BitNum. HandleNum is the numeric value of the IOC-7007 handle (1 – 8) while BitNum is the specific bit number on the IOC to be read.

TIn where n is the IOC-7007 input slot to be read. n is calculated with the equation n = (HandleNum * 1000) + SlotNum. Again, HandleNum is the numeric value of the IOC-7007 handle (1 – 8). SlotNum corresponds to the location of the IOM input module in the 7 slots of the IOC-7007 (0 – 6). This will return either an 8 bit or 16 bit decimal value depending on which IOM input module is being used.

Outputs of the IOC-7007 are set and cleared using the standard SB and CB commands, as well as with the OQ and OB commands. Outputs can be read with the @OUT[n] command. These commands operate as follows:

SBn or CBn where n is the IOC-7007 output to be set or cleared. n is calculated identically to the @IN[n] configuration, with n = (HandleNum * 1000) + BitNum.

@OUT[n] where n is the IOC-7007 output to be read. This uses the same n configuration as SB and CB.

OQn,m where n is the IOC-7007 output location and m is the data to be written. Specifically, n = (HandleNum * 1000) + SlotNum where HandleNum is the numeric value of the IOC-7007 handle

(1 – 8) and SlotNum is the slot number of the IOM output module to be written to (0 – 6). m is the decimal representation of the data written to the 4 (0 – 15) or 8 (0 – 255) output points of the IOM module.

Please refer to the IOC-7007 manual for complete information on how to configure, read and write information to the IOC-7007 Ethernet I/O module.

Modbus Support

The Modbus protocol supports communication between masters and slaves. The masters may be multiple PC's that send commands to the controller. The slaves are typically peripheral I/O devices that receive commands from the controller.

When the Galil controller acts as the master, the IH command is used to assign handles and connect to its slaves. The IP address may be entered as a 4 byte number separated with commas (industry standard uses periods) or as a signed 32 bit number. A port number may also be specified, and should be set to 502, which is the Modbus defined port number. The protocol must be TCP/IP for use with Modbus over Ethernet. Otherwise, the controller will not connect to the slave. (Ex. IHB=151,25,255,9<502>2 - This will open handle #2 and connect to the IP address 151.25.255.9, port 502, using TCP/IP)

An additional protocol layer is available for speaking to I/O devices. Modbus is an RS-485 protocol that packages information in binary packets that are sent as part of a TCP/IP packet. In this protocol, each slave has a 1 byte slave address. The DMC-31xx can use a specific slave address or default to the handle number.

The Modbus protocol has a set of commands called function codes. The DMC-31xx supports the 10 major function codes:

Function Code	**Definition**
01	Read Coil Status (Read Bits)
02	Read Input Status (Read Bits)
03	Read Holding Registers (Read Words)
04	Read Input Registers (Read Words)
05	Force Single Coil (Write One Bit)
06	Preset Single Register (Write One Word)
07	Read Exception Status (Read Error Code)
15	Force Multiple Coils (Write Multiple Bits)
16	Preset Multiple Registers (Write Words)
17	Report Slave ID

The DMC-31xx provides three levels of Modbus communication. The first level allows the user to create a raw packet and receive raw data. It uses the MBh command with a function code of –1. The format of the command is

MBh = -1,len,array[]   where   len is the number of bytes array[] is the array with the data

The second level incorporates the Modbus structure. This is necessary for sending configuration and special commands to an I/O device. The formats vary depending on the function code that is called. For more information refer to the Command Reference.

The third level of Modbus communication uses standard Galil commands. Once the slave has been configured, the commands that may be used are @IN[], @AN[], SB, CB, OB, and AO. For example, AO 2020,8.2 would tell I/O number 2020 to output 8.2 volts.

If a specific slave address is not necessary, the I/O number to be used can be calculated with the following:

I/O Number = (HandleNum*1000) +((Module-1)*4) + (BitNum-1)

Where HandleNum is the handle number from 1 (A) to 8 (H). Module is the position of the module in the rack from 1 to 16. BitNum is the I/O point in the module from 1 to 4.

If an explicit slave address is to be used, the equation becomes:

I/O Number = (SlaveAddress*10000) + (HandleNum*1000) +((Module-1)*4) + (Bitnum-1)

To view an example procedure for communicating with an OPTO-22 rack, refer to the appendix of the DMC-21x3 users manual.

Other Communication Options

User Defined Ethernet Variables

It may be necessary within a distributed system to share information that is not contained as position, torque, velocity or other control data. The DMC-31xx provides 2 user defined variables that are passed as part of the QW record shared among the distributed system. In this way, it is not necessary for a single controller to write variable data directly to all the other controllers in the system.

ZA and ZB are two user defined variables which are passed with the QW record at each update. Data that is written to these variables is then seen by the master DMC-31xx in the system.

Handle Switching

By default, when initiating a communication session with a DMC-31xx controller, the first available handle is used. If no handles have been assigned to the controller, the A handle is chosen. The command HS allows the user to switch this connection to another handle, freeing up the initial handle or trading with another currently used handle. Or, once handles have been defined, the HS command may be used to switch handles to prioritize slave locations and I/O locations.

Handle Restore on Communication Failure

There are instances within an Ethernet system, whether UDP or TCP/IP, when a handle may become disconnected without closing properly. An example of this would be a simple cable failure, where the Ethernet cable of a certain slave becomes detached.

The command HR is used to enable a mode in which the master controller, upon seeing a failure on a handle, will attempt to restore that handle. This is helpful when a distributed system is already fully configured and a slave is lost. The #TCPERR routine can be used to flag the error, while the handle restore will attempt to reconnect to the slave until the problem is fixed. This makes it unnecessary to re-run the setup for the entire distributed system.

Note: This function is only available if the system has been configured using the automatic handle configuration command, HC.

Waiting on Handle Responses

The operation of the distributed network has commands being sent to the master controller, which then distributes these commands to the slave axes in the system. For example, the command PR10,10,10,10,10,10,10,10 sent to the master becomes packets of PR10,10, PR10, or possibly PR10,10,10,10 sent by the master to each of the slaves in the system depending upon the number of axes on each slave. When the slave receives this command from the master, a colon or question mark is generated and sent back to the master to acknowledge the command.

The HW command allows the user to select whether or not the master will wait on this colon response from the slave. If the HW is set to 0, the master will not wait for these responses. This results in faster command execution but could cause problems if any slave errors are generated. The setting HW1, on the other hand, insures that the master knows of any slave errors but does result in a slightly increased command execution time as it waits for these responses.

Data Record

The DMC-31xx can provide a block of status information with the use of a single command, QR. This command, along with the QZ command can be very useful for accessing complete controller status. The QR command will return 4 bytes of header information and specific blocks of information as specified by the command arguments: QR ABCDEFGHS

Each argument corresponds to a block of information according to the Data Record Map below. If no argument is given, the entire data record map will be returned. Note that the data record size will depend on the number of axes.

NOTE: A, B, C, & D can be interchanged with X, Y, Z, & W respectively.

Data Record Map

DATA TYPE	ITEM	BLOCK
UB	1st byte of header	Header
UB	2nd byte of header	Header
UB	3rd byte of header	Header
UB	4rth byte of header	Header
UW	sample number	I block
UB	general input bank 0	I block
UB	general input bank 1	I block
UB	general input bank 2 (DB-28040)	I block
UB	general input bank 3 (DB-28040)	I block
UB	general input bank 4 (DB-28040)	I block
UB	general input bank 5 (DB-28040)	I block
UB	general input bank 6 (DB-28040)	I block
UB	general output bank 0	I block
UB	general output bank 1	I block
UB	general output bank 2 (DB-28040)	I block
UB	general output bank 3 (DB-28040)	I block
UB	general output bank 4 (DB-28040)	I block
UB	general output bank 5 (DB-28040)	I block
UB	general output bank 6 (DB-28040)	I block

UB	error code	I block
UB	general status	I block
UW	segment count of coordinated move for S plane	S block
UW	coordinated move status for S plane	S block
SL	distance traveled in coordinated move for S plane	S block
UW	segment count of coordinated move for T plane	T block
UW	coordinated move status for T plane	T block
SL	distance traveled in coordinated move for T plane	T block
UW	A axis status	A block
UB	A axis switches	A block
UB	A axis stopcode	A block
SL	A axis reference position	A block
SL	A axis motor position	A block
SL	A axis position error	A block
SL	A axis auxiliary position	A block
SL	A axis velocity	A block
SW	A axis torque	A block
SW	Analog Input 1	A block
UW	B axis status	B block
UB	B axis switches	B block
UB	B axis stopcode	B block
SL	B axis reference position	B block
SL	B axis motor position	B block
SL	B axis position error	B block
SL	B axis auxiliary position	B block
SL	B axis velocity	B block
SW	B axis torque	B block
SW	Analog Input 2	B block
UW	C axis status	C block
UB	C axis switches	C block
UB	C axis stopcode	C block
SL	C axis reference position	C block
SL	C axis motor position	C block
SL	C axis position error	C block
SL	C axis auxiliary position	C block
SL	C axis velocity	C block
SW	C axis torque	C block
SW	C axis analog input	C block
UW	D axis status	D block
UB	D axis switches	D block
UB	D axis stopcode	D block
SL	D axis reference position	D block
SL	D axis motor position	D block
SL	D axis position error	D block
SL	D axis auxiliary position	D block

SL	D axis velocity	D block
SW	D axis torque	D block
SW	D axis analog input	D block
UW	E axis status	E block
UB	E axis switches	E block
UB	E axis stopcode	E block
SL	E axis reference position	E block
SL	E axis motor position	E block
SL	E axis position error	E block
SL	E axis auxiliary position	E block
SL	E axis velocity	E block
SW	E axis torque	E block
SW	E axis analog input	E block
UW	F axis status	F block
UB	F axis switches	F block
UB	F axis stopcode	F block
SL	F axis reference position	F block
SL	F axis motor position	F block
SL	F axis position error	F block
SL	F axis auxiliary position	F block
SL	F axis velocity	F block
SW	F axis torque	F block
SW	F axis analog input	F block
UW	G axis status	G block
UB	G axis switches	G block
UB	G axis stopcode	G block
SL	G axis reference position	G block
SL	G axis motor position	G block
SL	G axis position error	G block
SL	G axis auxiliary position	G block
SL	G axis velocity	G block
SW	G axis torque	G block
SW	G axis analog input	G block
UW	H axis status	H block
UB	H axis switches	H block
UB	H axis stopcode	H block
SL	H axis reference position	H block
SL	H axis motor position	H block
SL	H axis position error	H block
SL	H axis auxiliary position	H block
SL	H axis velocity	H block
SW	H axis torque	H block
SW	H axis analog input	H block

NOTE: UB = Unsigned Byte, UW = Unsigned Word, SW = Signed Word, SL = Signed Long Word

Explanation of Status Information and Axis Switch Information

Header Information - Byte 0, 1 of Header:

BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8
1	N/A	N/A	N/A	N/A	I Block Present in Data Record	T Block Present in Data Record	S Block Present in Data Record
BIT 7	**BIT 6**	**BIT 5**	**BIT 4**	**BIT 3**	**BIT 2**	**BIT 1**	**BIT 0**
H Block Present in Data Record	G Block Present in Data Record	F Block Present in Data Record	E Block Present in Data Record	D Block Present in Data Record	C Block Present in Data Record	B Block Present in Data Record	A Block Present in Data Record

Bytes 2, 3 of Header:

Bytes 2 and 3 make a word that represents the Number of bytes in the data record, including the header. Byte 2 is the low byte and byte 3 is the high byte

NOTE: The header information of the data records is formatted in little endian.

General Status Information (1 Byte)

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
Program Running	N/A	N/A	N/A	N/A	Waiting for input from IN command	Trace On	Echo On

Axis Switch Information (1 Byte)

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
Latch Occurred	State of Latch Input	N/A	N/A	State of Forward Limit	State of Reverse Limit	State of Home Input	SM Jumper Installed

Axis Status Information (2 Byte)

BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8
Move in Progress	Mode of Motion PA or PR	Mode of Motion PA only	(FE) Find Edge in Progress	Home (HM) in Progress	1st Phase of HM complete	2nd Phase of HM complete <u>or</u> FI command issued	Mode of Motion Coord. Motion
BIT 7	**BIT 6**	**BIT 5**	**BIT 4**	**BIT 3**	**BIT 2**	**BIT 1**	**BIT 0**
Negative Direction Move	Mode of Motion Contour	Motion is slewing	Motion is stopping due to ST or Limit Switch	Motion is making final decel.	Latch is armed	Off-On-Error occurred	Motor Off

Coordinated Motion Status Information for plane (2 Byte)

BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8
Move in Progress	N/A	N/A	N/A	N/A	N/A	N/A	N/A
BIT 7	**BIT 6**	**BIT 5**	**BIT 4**	**BIT 3**	**BIT 2**	**BIT 1**	**BIT 0**
N/A	N/A	Motion is slewing	Motion is stopping due to ST or Limit Switch	Motion is making final decel.	N/A	N/A	N/A

Notes Regarding Velocity and Torque Information

The velocity information that is returned in the data record is 64 times larger than the value returned when using the command TV (Tell Velocity). See command reference for more information about TV.

The Torque information is represented as a number in the range of +/-32767. Maximum negative torque is -32767. Maximum positive torque is 32767. Zero torque is 0.

QZ Command

The QZ command can be very useful when using the QR command, since it provides information about the controller and the data record. The QZ command returns the following 4 bytes of information.

BYTE #	INFORMATION
0	Number of axes present
1	Number of bytes in general block of data record
2	Number of bytes in coordinate plane block of data record
3	Number of Bytes in each axis block of data record

Using Third Party Software

Galil supports ARP, BOOT-P, and Ping, which are utilities for establishing Ethernet connections. ARP is an application that determines the Ethernet (hardware) address of a device at a specific IP address. BOOT-P is an application that determines which devices on the network do not have an IP address and assigns the IP address you have chosen to it. Ping is used to check the communication between the device at a specific IP address and the host computer.

The DMC-31xx can communicate with a host computer through any application that can send TCP/IP or UDP/IP packets. A good example of this is Telnet, a utility that comes with most Windows systems. In the absence of the Galil Windows Terminal software, the Telnet terminal may be used for communication with the DMC-3425 Ethernet controller. The Windows Hyperterminal may also be used for communication.

Global vs. Local Command Listing

Command	Validity	Description
AB	Global	Stops motion and programs on all controllers in the distributed network
AC	Global	Sets accelerations on all axes specified
AD	Global	Trip point set for after distance on a specific axis in the network
AE	Local	Logic to monitor for amplifier errors should take place on the local controller
AF	Global	Configures axes to accept Analog Feedback (local controller requires DB-28040)
AG	Global	Sets the gain for the specified axis
AI	Local	Trip point to wait for a specific I/O to change states
AL	Global	Arms latch for the specified axis
AM	Global	Trip point to wait for the profiled motion to be completed
AO	Global	Sets the analog output voltage on an IOC-7007
AP	Global	Trip point to wait for an absolute position on a specific axis
AR	Global	Trip point to wait for a relative distance to be moved
AS	Global	Trip point set to wait for the specified axis to reach a speed
AT	Local	Trip point to wait for a specific amount of time
AU	Local	Setting for the bandwidth of the local amplifier
AV	Local	Trip point for after a vector distance has passed
AW	Local	Amplifier Bandwidth calculation
BA	Local	Brushless Axis setting used with commutation of a sinusoidal drive (do not use with AMP-20540)
BB	Global	Brushless Phase Beginning may be set globally, but setup of a sinusoidal axis is done locally
BC	Local	Brushless commutation may be used when configuring a sinusoidal axis
BD	Local	Brushless degrees may be used locally when configuring a sinusoidal axis
BG	Global	Begin motion on specified axes
BI	Global	May be used to configure inputs for hall inputs when configuring a sinusoidal axis (do not use with AMP-20540)
BK	Local	Sets a breakpoint at a specific line number for debug purposes
BL	Global	Reverse software limit set in counts
BM	Global	Configures the brushless modulus may be set globally, but sinusoidal axis configuration must be completed locally
BN	Local	Burns the local parameters into non volatile memory
BO	Global	Sets a voltage offset to an axis configured for sinusoidal operation
BP	Local	Burns the local program into non volatile memory
BR	Global	Brushed axis set for a specific axis that is also associated with an AMP-20540
BS	Local	Brushless axis used to configure a sinusoidal axis (do not use with AMP-20540)
BV	Local	Burns the local variables to non volatile memory
BZ	Local	Brushless zero is used for configuration of sinusoidal axes (do not use with AMP-20540)
CA	Local	Coordinate Axes selector used for vector or linear interpolation modes
CB	Global	Clears a specified bit
CD	Local	Contour Data points may be sent locally
CE	Global	Configure the encoder for quadrature/pulse and direction
CF	Local	Configure the handle to be used for unsolicited messages
CM	Local	Setup contour mode on a local axis

CN	Local	Configure the local setup of limit switch and home switch activity
CO	Local	Configure the setup of outputs on local extended I/O (Requires DB-28040 on the local controller)
CR	Local	Configures the parameters for a circle in the vector mode of local axes
CS	Local	Clear Sequence of vector/linear interpolation moves
CW	Local	Copyright information/Data Adjustment Bit
DA	Local	Deallocate local arrays
DC	Global	Set axis specific declarations
DE	Global	Define auxiliary encoder positions
DL	Local	Download program to local controller
DM	Local	Allocate space for arrays on the local controller
DP	Global	Define position of main encoder
DT	Local	Delta time for contour mode
DV	Local	Configure dual loop mode for a specific axis
EA	Local	Set up ecam mode for local axis
EB	Global	Enables the ecam mode for the specific axis (when commanded globally ecam parameters must be set up locally)
EC	Local	ECAM counter used when entering ECAM table information
ED	Local	Edit the local program space
EG	Global	ECAM go on a specific master position by axis (ECAM tables must be set up locally)
ELSE	Local	ELSE statement may be used in a local program
EM	Global	ECAM Modulus defines the change in position over one cycle of the master
EN	Local	Program or Subroutine end for a local program
ENDIF	Local	Endif statement for a local program
EO	Local	Sets the echo to off or on for communications
EP	Local	ECAM Interval for the local ECAM table
EQ	Global	ECAM quit for the specified axis
ER	Global	Error limit for the specified axis
ES	Local	Elliptical scale for local axes in vector/linear interpolation modes
ET	Local	Ecam Table point
FA	Global	Sets the feedforward acceleration for the specified axis.
FE	Global	Find the edge of the home switch for the specified axis
FI	Global	Find the index for the specified axis
FL	Global	Sets the forward software limit for the specified axis
FV	Global	Sets the feedforward velocity for the specified axis
GA	Local	Sets the master axis for the specified axis (gearing may only occur with axes on the same controller)
GM	Global	Sets the gantry mode for the specified axis
GR	Global	Sets the Gear ratio for the specified axis (gearing may only occur with axes on the same controller)
HM	Global	Home the specified axis
HS	Local	Switch ethernet handles

HX	Local	Halt the specified program thread
IA	Local	Set the IP address
IF	Local	If statement for a local program
IH	Local	Open/Close ethernet handle specified
II	Local	Designate input for an input interrupt
IL	Global	Sets the integrator Limit for the specified axes
IP	Global	Increment Position on specified axes
IT	Global	Motion smooting constant for specified axes
JG	Global	Jog for specified axes
JP	Local	Jump to specified program location
JS	Local	Jump subroutine to specified local subroutine
KD	Global	Sets the derivative constant for the specified axes
KI	Global	Sets the integrator for the specified axes
KP	Global	Sets the proportional constant for the specified axes
KS	Global	Sets the stepper smoothing constant for the specified axes
LA	Local	List the declared arrays on the local controller
LC	Global	Sets the stepper axes into a low current mode
LE	Local	Linear sequence end
_LF*	Global	Forward limit switch operand
LI	Local	Linear interpolation segment
LL	Local	List local program labels
LM	Local	Declare axes for linear interpolation mode
_LR*	Global	Reverse limit switch opearnd
LS	Local	List local program
LV	Local	List local variables
LZ	Local	Format the number of leading zeros returned
MB	Local	Modbus Command
MC	Global	Motion complete on the specified axes
MF	Global	Motion forward specified distance on the specified axis
MG	Local	Message command
MO	Global	Motor off for the specified axes
MR	Global	Motion reverse specified distance on the specified axis
MT	Global	Motor type for the specified axes
MW	Local	Modbus wait
NB	Global	Notch filter bandwidth
NF	Global	Notch filter frequency
NO	Local	No operation on program line
NZ	Global	Notch filter zero
OB	Global	Output specifed bits based on logic
OC	Local	Specify the output compare pulse and reoccurrence rate

OE	Global	Off on error function specified by axis
OF	Global	Offset command for the specified axes
OP	Local	Sets the states of multiple outputs
PA	Global	Position absolute for the specified axes
PF	Local	Sets the format for returned position information
PL	Local	Sets the constant of the pole filter
PR	Global	Position relative for the specified axes
QD	Local	Array upload
QH	Local	Returns hall states when connected with AMP-20540
QR	Local	Data record command for the local controller
QU	Local	Array upload
RA	Local	Record array function
RC	Local	Begins the array record
RD	Local	Sets what data to record
RE	Local	Return from error routine
RI	Local	Return from interrupt routine
RL	Global	Report latched position
RP	Global	Reports the reference position
RS	Local	Reset the local controller
^R^S	Local	Master reset the local controller
^R^V	Local	Returns the local controller model and firmware revision
SA	Local	Send ASCII command to the specified communication handle
SB	Global	Set the specifed output bit
SC	Global	Returns the stop code for the various axes
SH	Global	Servo here for the specified axes
SL	Local	Single line step through
SP	Global	Sets the speed for the specified axes
ST	Global	Stops the motion of the specified axes
TA	Local	Tells the status of any amplifier errors when used with AMP-20540
TB	Local	Tell the status byte of the local control
TC	Local	Tell the error code from the command in error
TD	Global	Tell auxiliary encoder position
TE	Global	Tell the following error
TH	Local	Returns the status information of local ethernet handles
TI	Local	Tells the status of the specified inputs
TIME	Local	Operand containing the free running clock on the controller
TK	Global	Sets the peak torque limit for the specified axes
TL	Global	Sets the average torque limit for the specified axes when used with AMP-20540 or AMP-20440
TM	Local	Sets the servo update rate

TN	Local	Sets the tangent axes for a vector move
TP	Global	Tells the position of the specified axes
TR	Local	Trace the program execution of the local program
TS	Global	Tells the status of the switches on the specified axes
TT	Global	Tells the voltage command output of the specified axes
TV	Global	Tells the velocity of the specified axes
TW	Local	Sets the timeout for the MC command
TZ	Global	Tells the status of the I/O for the distributed system
UL	Local	Upload the local program
VA	Local	Sets the vector acceleration
VD	Local	Sets the vector deceleration
VE	Local	End of vector sequence
VF	Local	Sets the displayed format of variables
VM	Local	Sets the specified axes in vector mode
VP	Local	Vector position command
VR	Local	Vector ratio
VS	Local	Sets the vector speed
VT	Local	Sets the constant for vector smoothing
WC	Local	Trip point for contour data
WH	Local	Tells what handle the executed command came from
WT	Local	Trip point telling the controller to wait n samples
XQ	Local	Execute program
ZS	Local	Zero program stack

Supplemental Commands

HA

FUNCTION: Handle Assignment

DESCRIPTION:

The HA command establishes the connection order for the slave controllers in a distributed system. This command must be executed in order for the HC command to configure and assign the slaves with the proper IP addresses within the distributed system. The arguments given with the command are the serial numbers of the slave controllers in the system. If you do not know the serial numbers of the controllers in your system, you may query them by issuing the HQ command to the master controller. The master controller must have a valid IP address before it can execute the HQ command.

ARGUMENTS: HA n,n,n,n,n,n,n where

n represents the serial numbers of the slave controllers in the system. The system may have a total of 8 axes. Each slave may have as few as 1 axis and as many as 7 axes.

USAGE: **DEFAULTS:**

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**DMC-31xx**

OPERAND USAGE:

_Han contains the serial number of the appropriate slave where n may range from 0 to 7.

RELATED COMMANDS:

IA	Internet Address
IH	Internet Handle
HQ	Handle Query
QW	Slave data records

EXAMPLES:

HA 5522,5533	Assigns the connection order of slaves in a distributed system. The controller with serial number 5522 will be slave 1 and the controller with serial number 5533 will be slave 2.
HC4,20,2,0	Configures a 4 axis system with two TCP/IP handles per slave. The data update interval is set to 20 milliseconds. For each slave, the TCP?IP handle will be used for the data update.
IA 151,12,53,89	Assigns the controller with the address 151.12.53.89
HQ	Queries the network for controllers without IP addresses issuing Boot-P packets.
HQ?	Returns the results of the HQ command. The results contain serial numbers along with the number of axes available on each controller. It may be required to wait 5-10 seconds for the HQ process to complete.

HC

FUNCTION: Handle Configuration

DESCRIPTION:

The HC command configures and establishes communications for a master/slave system. The command is executed in the master controller and addresses all slaves and IOC modules in the system. After the HC command is initiated, the master responds to the slave and IOC bootp requests and assigns corresponding IP addresses in the order assigned by the HA command. The master then opens handles and initiates the slave update packets (QW).

The IP address for the master controller must be established with the IA command or DMCNet software prior to the HC command being issued. The master will assign IP addresses to these controllers as it receives the bootp packets. The slave controllers must not be assigned IP addresses or they will not be sending out bootp packets.

ARGUMENTS: HCa,b,c,d where

a is the total number of axes in the system

b is the slave update interval (QW) in milliseconds.

c is the communication protocol for the slave communications

1 = UDP (1 handle used)

2 = TCP/IP (2 handles used)

3 = TCP/IP used for Command Handle, UDP used for QW update Handle

d is the total number of IOC-7007 modules in the system

HC? Returns the present setting of the HC command

USAGE: **DEFAULTS:**

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**DMC-3xxx**

OPERAND USAGE:

_HC contains a 1 if the handle configuration is in progress

contains a 2 if the handle configuration has completed successfully

contains a 0 if the handle configuration failed or has not been issued

RELATED COMMANDS:

IA	Internet Address
IH	Internet Handle
HA	Handle Assignment
HQ	Handle Query
QW	Slave data records

EXAMPLES:

IA 151, 12,53,89	Assigns the controller with the addresses 151.12.53.89
HQ	Queries the network for controllers without IP addresses issuing Boot-P packets.

HQ?	Returns the results of the HQ command. The results contain serial numbers along with the number of axes available on each controller. It may be required to wait 5-10 seconds for the HQ process to complete.
HA 5522,5533	Assigns the connection order of slaves in a distributed system. The controller with serial number 5522 will be slave 1 and the controller with serial number 5533 will be slave 2.
HC4,20,2,0	Configures a 4 axis system with two TCP/IP handles per slave. The data update interval is set to 20 milliseconds. For each slave, one TCP/IP handle will be used for sending commands while the other TCP/IP handle will be used for the data update.
HC6,30,1,0	Configures a 6 axis system with a single UDP handle per slave at updates of 30 msec. The single UDP handle is used for both sending commands and receiving data packets.
#AUTO HC 3,250, 2	Example program that will automatically run when controller is powered up (#AUTO). HC command configures a 3 axis system with a 250 msec update rate.
#LOOP;JP#LOOP,_HC<>2 MG"Connected"EN	#Loop routine causes controller to wait for successful connection before continuing execution of code.

Hint: Use a WT (Wait) or #LOOP; JP#LOOP,_HC<>2 when issuing the HC command in a program to allow enough time for slaves to be configured correctly before executing any other commands.

HQ

FUNCTION: Handle Query

DESCRIPTION:

The HC command queries the network for controllers that are issuing bootp packets. Only motion controllers without IP addresses will be issuing bootp packets. To see the results of the command, issue the HQ? after the command has completed executing. It may be necessary to wait 5-10 seconds for HQ to complete. This command must be issued to the master controller.

The IP address for the master controller must be established with the IA command or DMCNet software prior to the HQ command being issued.

ARGUMENTS: HQ

HQ? returns the controllers found without IP addresses in the format a,b,c where

a = controller type 1 for motion controllers and 255 for the IOC-7007

b = number of motion axes available

c = the serial number of the controller

If the HQ command has not completed execution, HQ? returns "1"

USAGE: **DEFAULTS:**

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**DMC-3xxx**

RELATED COMMANDS:

IA	Internet Address
IH	Internet Handle
HA	Handle Assignment
QW	Slave data records

EXAMPLES:

HQ	Queries the network for controllers without IP addresses issuing Boot-P packets.
HQ?	Returns the results of the HQ command. The results contain serial numbers along with the number of axes available on each controller. It may be required to wait 5-10 seconds for the HQ process to complete.
IA 151, 12,53,89	Assigns the controller with the addresses 151.12.53.89
HA 5522,5533	Assigns the connection order of slaves in a distributed system. The controller with serial number 5522 will be slave 1 and the controller with serial number 5533 will be slave 2.
HC4,20,2,0	Configures a 4 axis system with two TCP/IP handles per slave. The data update interval is set to 20 milliseconds. For each slave, one TCP/IP handle will be used for sending commands while the other TCP/IP handle will be used for the data update.
HC6,30,1,0	Configures a 6 axis system with a single UDP handle per slave at updates of 30 msec. The single UDP handle is used for both sending commands and receiving data packets.