USER MANUAL

IOC-7007

Manual Rev. 1.0c

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 Date: 12/08

Contents

Chapter 1 Overview

Introduction

Derived from the same fundamentals used in building the Galil motion controllers, the IOC-7007 is a programmable I/O board that conveniently interfaces with other Galil boards through its Ethernet port. The IOC-7007 is programmed exactly the same way as a DMC (Galil controller) with the exception of a few revised commands and the removal of all motion-related commands. Communication with the IOC-7007 even works the same way as with other Galil controllers, and it utilizes the same software programs, such as the DMC Terminal. New interrogation commands have been created to allow a user to instantly view the entire I/O status, thread execution, or Ethernet handle availability (see the TZ, TQ and TH commands in Chapter 5).

The purpose of an IOC-7007 board is to offer remote I/O in a system and the ability to synchronize complex events. To do this, the IOC-7007 accepts up to 7 plug-in I/O modules (starting with the abbreviation IOM for Input/Output Module) that can be "mixed and matched" according to the system requirements. There are 8 different IOM variations currently available, consisting of digital inputs, digital outputs, analog inputs, or analog outputs. See Chapter 4 for IOM specifications.

Packaging options for the IOC-7007 include a card level product (IOC board and IOM modules only), a box level product with a steel enclosure, and a DIN rail mountable version. The IOC can be powered with either a 90-260VAC 50/60Hz input or a 20-60V DC input. The AC option is only available on the IOC-7007-Box.

IOC-7007 Functional Elements

Microcomputer Section

The main processing unit of the IOC-7007 is a specialized 32-bit, 25MHz Motorola 68331 Series Microcomputer with 1 Mbit RAM and 2 Mbit Flash EEPROM. The RAM provides memory for variables, array elements and application programs. The flash EEPROM provides non-volatile storage of variables, programs, and arrays; it also contains the IOC-7007 firmware.

The IOC-7007 can processes individual Galil Commands in less than 100 microseconds. The Non-volatile memory includes 500 lines x 80 characters of program space, 126 symbolic variables, 126 labels, and 2000 array elements in up to 14 arrays.

Communication

The communication interface with the IOC-7007 consists of one RS-232 port (default is 19.2 kBaud/s) and one auto negotiating 10/100Base-T Ethernet port.

There are six status LEDs on the IOC that indicate operating and error conditions on the controller. Figure 1-1 shows a diagram of the LED bank followed by the description of the six lights.

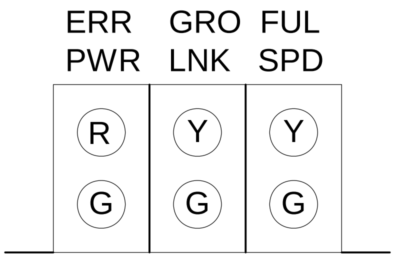


Figure 1-1 - Diagram of LED bank on the IOC-7007

Green Power LED (PWR) - The green status LED indicates that the power has been applied properly to the IOC.

Red Status/Error LED (ERR) - The red error LED will flash on initially at power up, and stay lit for approximately 1 – 8 seconds before turning off. After this initial power up condition, the LED will illuminate for the following reasons:

1. The reset line on the controller is held low or is being affected by noise.
2. There is a failure on the controller and the processor is resetting itself.
3. There is a failure with the output IC that drives the error signal.

Green Activity LED (LNK) – The yellow LED indicates traffic across the Ethernet connection. This LED will show both transmit and receive activity across the connection. It illuminates with a good link and then blinks when activity is detected. If the controller has not been assigned an IP address, it will send BOOTP packets at regular intervals as indicated by the rhythmic blinking of this LED. Once the IP address is assigned, the LED will only blink with Ethernet traffic. Also, if an IP address isn't burned in with the BN command, the next time power is cycled or a reset is issued, the IP address will be lost and the controller will begin to issue BOOTP packets again.

General Purpose Output (GRO) – This LED is not currently used by the IOC-7007.

Ethernet Speed LED (SPD) - This LED will light up if the Ethernet connection is 100 Base-T.

Full Duplex LED (FUL) – This LED turns on when the Ethernet connection is full duplex.

Chapter 2 Getting Started

The IOC-7007 I/O Controller

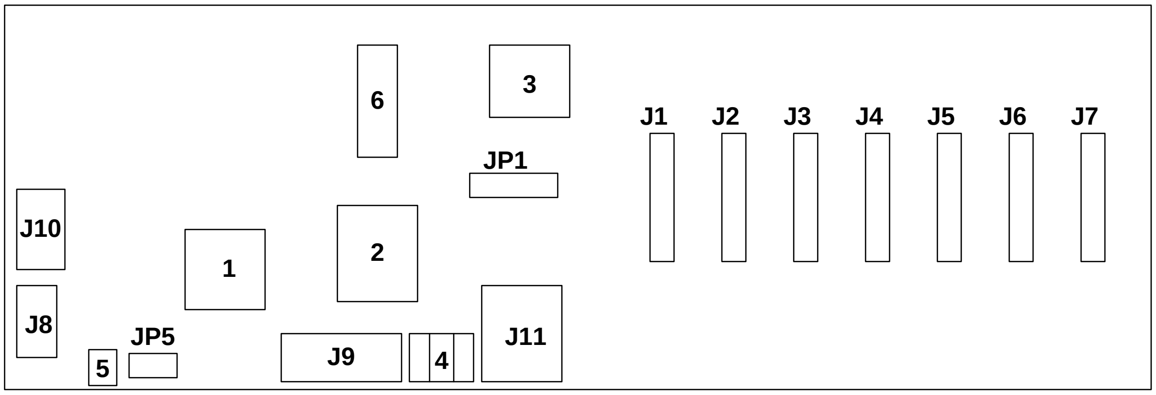


Figure 2-1: - Outline of IOC-7007

1	**Motorola 68331 microprocessor**	**J8**	**4 pin Molex for 20-60 VDC connection**
2	**ASIX Ethernet Chip**	**J9**	**RS232 Serial connection**
3	**Xilinx I/O chip**	**J10**	**250V-2A Fuse for 20-60 VDC input, or 6 pin Molex for AC power supply.**
4	**Error LEDs for active Ethernet connection, transmit/receive on Ethernet, error output and power**	**J11**	**10/100 BaseT Ethernet Port**
5	**Reset Button**	**JP1**	**Adress selection jumpers**
6	**RAM**	**JP5**	**Master reset , upgrade and baud rate selection jumpers.**
J1-J7	**26 pin connectors for I/O Slots 0 through 6**		

Installing the IOC-7007 Board

Installation of a complete, operational IOC-7007 system consists of 6 steps:

Step 1. Configure jumpers

Step 2. Plug in the desired IOM modules

Step 3. Connect power to the IOC-7007

Step 4. Install the communications software

Step 5. Establish communications between the IOC-7007 and the host PC

Step 6. Configure the IOC for Galil's Distributed Control System

Step 1. Configure Jumpers

Master Reset and Upgrade Jumper

Jumpers labeled as MRST and UPGD are located at JP5, adjacent to the reset button. The MRST jumper is for a master reset. When MRST is connected, the IOC-7007 will perform a master reset upon power recycle to the board or when the board reset button is pushed. Whenever the I/O board has a master reset, all programs, arrays, and variables stored in EEPROM will be **erased**.

The UPGD jumper enables the user to unconditionally update the board firmware. This jumper is not necessary for firmware updates when the IOC board is operating normally, but may be necessary in cases of a corrupted EEPROM. EEPROM corruption should never occur under normal operating circumstances; however, corruption is possible if there is a power fault during a firmware update. If EEPROM corruption occurs, your board may not operate properly. In this case, install the UPGD jumper and use the update firmware function on the Galil Terminal to re-load the system firmware.

Setting the Baud Rate on the IOC-7007

The jumpers labeled "1200" and "9600," also located at JP5, allow the user to select the serial communication baud rate. The baud rate can be set using the following table:

9600	1200	BAUD RATE
OFF	OFF	19200
ON	OFF	9600
OFF	ON	1200

The default baud rate for the IOC-7007 is 19.2K (no jumpers).

Distributed Control Address Jumpers

The IOC-7007 can be set up in a distributed control network with other IOCs and DMC-3425 controllers. In this configuration, the controllers and IOCs act as if they were one multi-axis controller. The host computer communicates to only one master controller, and the master coordinates data transfer between the other controllers and IOCs in the network. For more information on distributed control, see the DMC-3425 User Manual.

The HC command, available on the DMC-3425, is used to automatically configure the various controllers and IOCs in a Galil distributed network. If this command is used, then each IOC must have a jumper corresponding to its address setting in the distributed network.

The 16-pin jumper, found at location JP1, is used to select address settings for the IOC-7007. Jumpers at this location are labeled ADR1, ADR2, ADR4 and ADR8, which represent the binary value for each of the 8 addresses available for an IOC. The following chart shows the proper jumper settings for every possible address selection.

IOC #1	ADR1 On	ADR2 Off	ADR4 Off	ADR8 Off
IOC #2	ADR1 Off	ADR2 On	ADR4 Off	ADR8 Off
IOC #3	ADR1 On	ADR2 On	ADR4 Off	ADR8 Off
IOC #4	ADR1 Off	ADR2 Off	ADR4 On	ADR8 Off
IOC #5	ADR1 On	ADR2 Off	ADR4 On	ADR8 Off
IOC #6	ADR1 Off	ADR2 On	ADR4 On	ADR8 Off
IOC #7	ADR1 On	ADR2 On	ADR4 On	ADR8 Off
IOC #8	ADR1 Off	ADR2 Off	ADR4 Off	ADR8 On

If two IOCs are needed for a Galil distributed network then the "IOC #1" jumper setting is used on one of the IOCs and the "IOC #2" setting is used on the other. Likewise, if three IOC's are configured, then the #1, #2 and #3 settings are used.

For example, the jumper settings for the third IOC in a distributed network are shown below.

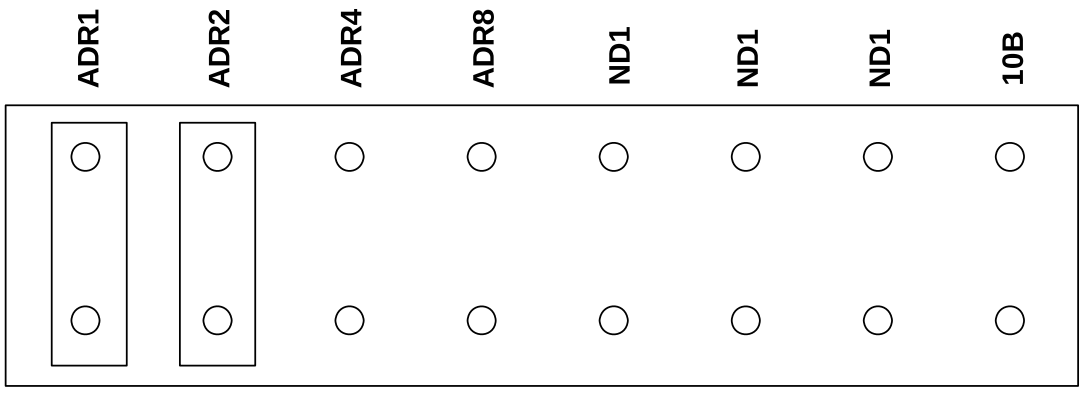


Fig. 2-2 – Example jumper settings for the 3rd IOC-7007 in a distributed network

10 Base-T jumper

At JP1 there is a set of pins labeled **10B** (shown in figure 2.3 above). If a jumper is placed at this location, the Ethernet port on the IOC-7007 will be restricted to 10 Base-T. Normally, the Ethernet port is auto-negotiating between 10 and 100 Base-T, but the user may want to force the port to 10 Base-T when 100 Base-T is not required.

Step2. Plug in the Desired IOM Modules

The IOC-7007 can accommodate up to 7 IOM modules. Plug each module into one of the 26 pin connectors labeled Slot-0 through Slot-6. The user can choose any slot to insert an IOM module. As soon as the IOC is powered, the location of these modules is detected and their configuration as either input or output modules is determined. Also, the IOC detects the number of I/O ports available on each IOM module.

Note: Be sure to connect the IOM modules in such a way that the chips on the modules face in the direction of the IOC power connectors. The edge of the IOM cards should lie in the narrow rectangular silk screening on the IOC.

Step 3. Connecting Power to the IOC-7007

The IOC-7007 has the option for either 20-60 VDC power or 90-260 VAC power. The 90-260 VAC power option is only available on the IOC-7007-Box.

For the 20-60 VDC option, apply the voltage to the +24 and G terminals of the 4 pin MOLEX connector at J8. The other two pins on J8, E and NC, are for earth ground and no connection. Earth ground can be used if desired.

Warning: *Damage can occur if a supply of 60VDC or larger is connected to the board.*

For the 90-260 VAC power option, connect the IOC-7007-Box directly to AC power from a wall socket.

Applying power will turn on the green LED power indicator.

Step 4. Install the Communications Software

After applying power to the computer, install the Galil software that enables communication between the I/O board and your PC.

Using DOS:

Using the Galil Software CD-ROM, go to the directory, DMCDOS. Type "INSTALL" at the DOS prompt and follow the directions.

Using Windows 3.x (16 bit versions):

Using the Galil Software CD-ROM, go to the directory, DMCWIN16. Run DMCWIN16.exe at the Command prompt and follow the directions.

Using Windows 95 or 98 First Edition (32 bit versions):

The Galil Software CD-ROM will automatically begin the installation procedure when the CD-ROM is inserted. After installing the Galil CD-ROM software on the computer, install other software components as desired. To install the basic communications software, run the Galil Software CD-ROM and choose "DMCWIN32 - Windows Utilities and Programming Libraries, WIN95, 98, NT". This will install the Galil Terminal, which can be used for communication.

Using Windows 98 SE, NT 4, 2000, ME or XP (32 bit versions):

The procedure for installing software for these operating systems is the same as for 95 and 98. However, there are different versions of the same software for these operating systems.

Step 5. Establish Communications between IOC-7007 and the Host PC

Communicating through the RS-232 Serial Communications Port

To use the serial communications, connect the CABLE-9-PIND (straight-through RS-232 cable) between the serial port of the IOC-7007 and the computer or terminal communications port. The IOC-7007 serial port is configured as DATASET. The computer or terminal must therefore be configured as a DATATERM for full duplex, no parity, 8-bit data, one start bit and one stop bit. It should also be configured as a "dumb" terminal, which sends ASCII characters as they are typed to the IOC-7007.

Using Galil Software for DOS

To communicate with the IOC-7007, type TALK2DMC at the prompt. It is necessary to provide information about the IOC, but the model to specify is DMC-1412. Provide the port number and baud rate for communication. Once communication has been established, the terminal display should show a colon, ":". If a colon is not received, press the carriage return. If a colon prompt is still not returned, there is most likely an incorrect setting of the serial communications port. The user must ensure that the correct communication port and baud rate are specified when attempting to communicate with the I/O board. The user must insure that the proper serial cable is being used (see page 15 for pin-out of serial cable).

Using Galil Software for Windows 95 and 98 First Edition

In order for the windows software to communicate with the IOC-7007, the board must be registered in the Windows Registry. To register this I/O board, specify the model (use DMC-1412), the communication parameters, and other information. The registry is accessed through Galil software, such as DMC Terminal and DTERM (DTERM is installed with DMCWIN and installed as the icon "Galil Terminal"). From DMC Terminal, the registry is accessed under the FILE menu. From the DTERM program, the registry is accessed from the REGISTRY menu.

The registry window is equipped with buttons to **Add, Change,** or **Delete** a 'controller.' Pressing any of these buttons will bring up the Set Registry Information window.

Use the **Add** button to add a new entry to the Registry.

Note: Since the new IOC series is not included in the list of models, register it as a DMC-1412 communicating via the RS-232 interface.

The registry information will show a default Comm Port of 2 and a default Comm Speed of 9600. This information should be changed as necessary to reflect the computers Comm Port and the baud rate set by the IOC-7007 baud rate jumpers. The registry entry also displays timeout and delay information. These are advanced parameters that should only be modified by advanced users (see software documentation for more information).

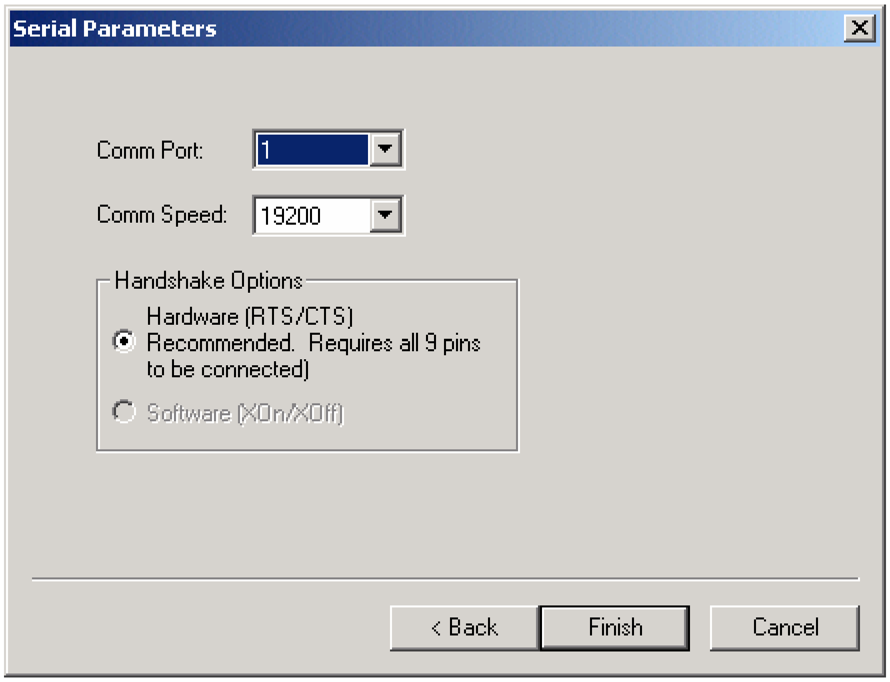
Once the appropriate Registry information has been set, in this case as a DMC-1412, select OK and close the registry window. The user is now able to communicate with the IOC-7007. Once the entry has been selected, click on the **OK** button. If the software has successfully established communications with the IOC board, the registry entry will be displayed at the top of the screen.

If the computer is not properly communicating with the board, the program will pause for 3-15 seconds. The top of the screen will display the message “Status: not connected with Galil motion controller” and the following error will appear: “STOP - Unable to establish communication with the Galil controller. A time-out occurred while waiting for a response from the Galil controller.” If this message appears, click OK. In this case, there is most likely an incorrect setting of the serial communications port. The user must ensure that the correct communication port and baud rate are specified when attempting to communicate with the board. The user must insure that the proper serial cable is being used (see page 15 for pin-out of serial cable).

Once communications have been established, click on Terminal found under the main menu, and a colon prompt should appear. Communicating with the IOC board is described in later sections.

Using Galil Software for Windows 98 SE, NT 4.0, 2000, ME, and XP

As in Windows 95 and 98 FE, the IOC-7007 is registered through the Galil Registry in these operating systems. The user can access the Galil registry by selecting “Register Controller…” in the File menu of WSDK or DMC Terminal or just click “Registry” in DMCWIN32. Select the button that says “New Controller” under the Non-PnP Tools and choose IOC-7007 from the pull down menu. Next, select the ‘Serial’ button under the Connection Type and click ‘Next’. The final step is to select the Comm Port being used on the PC and the Comm Speed for data transfer. Hardware handshaking will be selected by default. Select ‘Next’, and the controller will be entered into the registry. Connect to the controller by selecting the Terminal utility and choosing the controller from the registry list.


Serial Parameters

Comm Port: 1

Comm Speed: 19200

Handshake Options

[✓ Hardware (RTS/CTS) Recommended. Requires all 9 pins to be connected)]

Software (XOn/XOff)

Fig. 2-3 – Configuring Serial Parameters

Note: Be sure to configure the controller for the specified Comm Speed with the necessary jumper settings.

Using Non-Galil Communication Software

The IOC-7007 serial port is configured as DATASET. The computer or terminal must be configured as a DATATERM for full duplex, no parity, 8 data bits, one start bit and one stop bit.

Check to insure that the baud rate switches (jumpers) have been set to the desired baud rate as described above. Also, the hardware handshake lines (RTS/CTS) need to be in the correct state to communicate. See Chapter 3 for more information on 'Handshake Modes.'

The computer needs to be configured as a "dumb" terminal, which sends ASCII characters as they are typed to the IOC-7007.

Sending Test Commands to the Terminal

After connecting to the computer or terminal, press <carriage return> or the <enter> key on the keyboard. In response to carriage return (CR), the controller responds with a colon, :

Now type

```
TZ (CR)
```

This command directs the IOC to return the current I/O status. The controller should respond with something similar to the following:

```
Slot 0 (7-0) = Digital Output – value 0 (0000_0000)
Slot 1 = I/O module not detected
Slot 2 (79-64) = Digital Input – value 65535 (1111_1111_1111_1111)
Slot 3 = I/O module not detected
Sot 4  (135-128) = Digital Input – value 255 (1111_1111)
```

Slot 5 = I/O module not detected
Slot 6 (195-192) = Digital Output – value 0 (0000)

This response shows slots 2 and 4 occupied by input modules, and slots 0 and 6 with output modules.

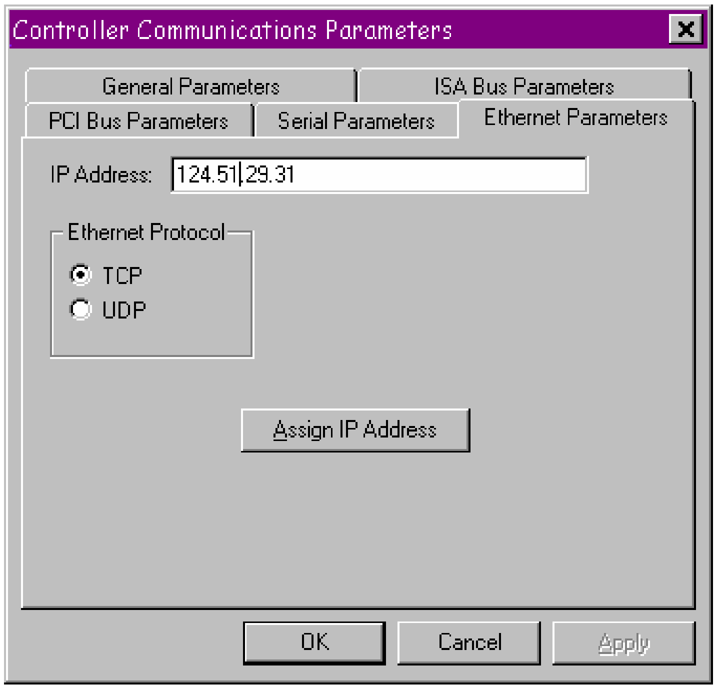
Communicating through the Ethernet

Connect the IOC-7007 Ethernet port to your computer via a crossover or null modem Ethernet cable, or to a network hub by a straight through Ethernet cable.

Using Galil Software for Windows 95 and 98 FE

The IOC-7007 must be registered in the Windows registry for the host computer to communicate with it. The registry may be accessed via Galil software, such as DTERM.

From DTERM, the registry is accessed under the REGISTRY menu. Use the ***Add*** button to add a new entry in the registry. Choose DMC-1415 as the model type and click on the Ethernet tab. Enter the IP address obtained from your system administrator. Select the button corresponding to the UDP or TCP protocol in which to communicate with the IOC-7007. If the IP address has not been already assigned to the board, click on **ASSIGN IP ADDRESS.**



IP Address: 124.51.29.31

Ethernet Protocol
(●) TCP
( ) UDP

Fig. 2-4 –Ethernet Parameters for the IOC-7007

ASSIGN IP ADDRESS will check the I/O boards that are linked to the network to see which ones do not have an IP address. The program will then prompt the user to assign the IP address, which has been entered, to the device with the specified serial number. Click on **YES** to assign it, **NO** to move to next device, or **CANCEL** to not save the changes. If there are no devices on the network that do not have an IP address assigned, the program will state this.

When done registering, click on **OK**. If you do not wish to save the changes, click on **CANCEL**.

Once the IOC-7007 has been registered, enter the Terminal and select the IOC board again from the list and click on **OK**. If the software successfully established communications with the board, the registry entry will be displayed at the top of the screen. Type in BN to save the IP address to the non-volatile memory of the board.

NOTE: The IOC board must be registered via an Ethernet connection, using DMC-1415 as the model type.

If the above method is unsuccessful in assigning an IP address to the IOC-7007, the second option is connecting to the IOC serially and using the IA command to assign the IP address. The user can then add an Ethernet IOC to the registry with the specified address, and connect to it by opening the terminal utility. It is possible to add an Ethernet IOC to the registry without assigning the address by simply clicking **FINISH** in the Ethernet parameters tab instead of clicking **ASSIGN IP ADDRESS**.

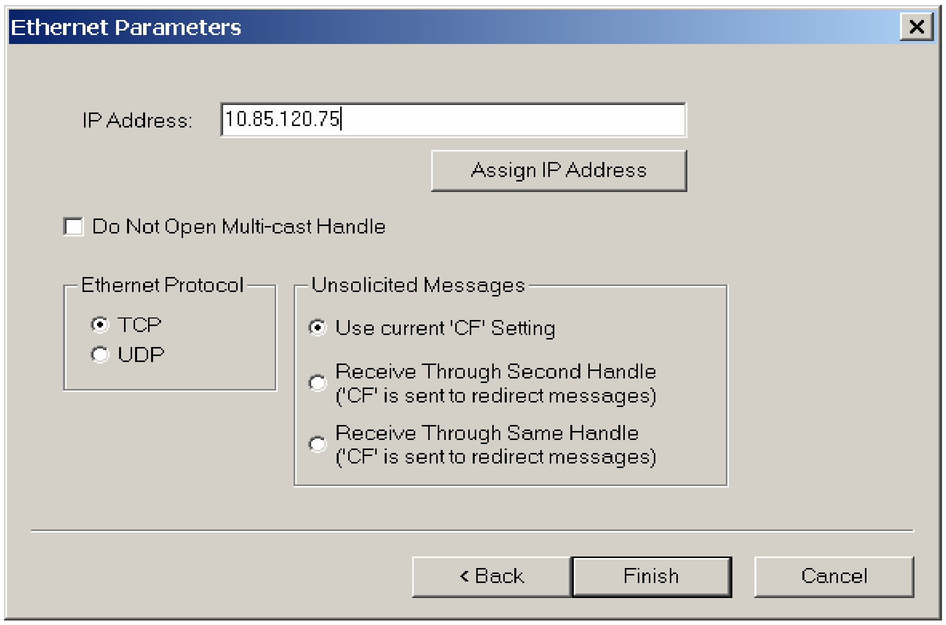
When connecting to an IOC-7007 via Ethernet, the user must be aware of the type of Ethernet cable being used, and the method of communication. To connect the IOC directly to the PC, use a crossover or null-modem Ethernet cable. This type of cable allows for the crossing of signals between the PC and the controller.   If instead the connection to the IOC is through a network hub, a straight through cable must be used. Hubs perform the signal crossing function of a null-modem cable. If the wrong cable is used, communication with the controller will not be possible. The LNK led must be illuminated for Ethernet communications to occur.

Note: If the IOC-7007 is connected in a LAN, make sure the assigned IP address is allowed. Also, Galil strongly recommends the IP address selected cannot be accessed across the Gateway. The Gateway is an application that controls communication between an internal network and the outside world. Ask your network administrator for acceptable IP addresses.

Using Galil Software in Windows 98 SE, NT 4.0, 2000, ME, and XP

The IOC-7007 is registered through the Galil Registry in these operating systems. The user can access the Galil registry by selecting “Register Controller…” in the File menu of WSDK or DMC Terminal, or by clicking “Registry” in DMCWIN32. Select the button that says “New Controller” under Non-PnP Tools and choose IOC-7007 from the pull down menu. Click “Ethernet” under “Connection Type”, and the next screen will allow the user to enter an IP address for the device. This is a 4-byte number, each byte separated by periods. The bytes are represented by their decimal equivalents, meaning that the numeric range is between 0 and 255.

Also, select the Ethernet Protocol as either TCP or UDP. Galil recommends TCP because if information is lost during communication, it will be resent using this protocol. UDP is a more efficient protocol, but does not resend lost information. See Chapter 3 for more detailed information on Ethernet and communication protocols.


Ethernet Parameters

IP Address: 10.85.120.75

Assign IP Address

[ ] Do Not Open Multi-cast Handle

Ethernet Protocol
- [✓ TCP]
- UDP

Unsolicited Messages
- [✓ Use current 'CF' Setting]
- Receive Through Second Handle ('CF' is sent to redirect messages)
- Receive Through Same Handle ('CF' is sent to redirect messages)

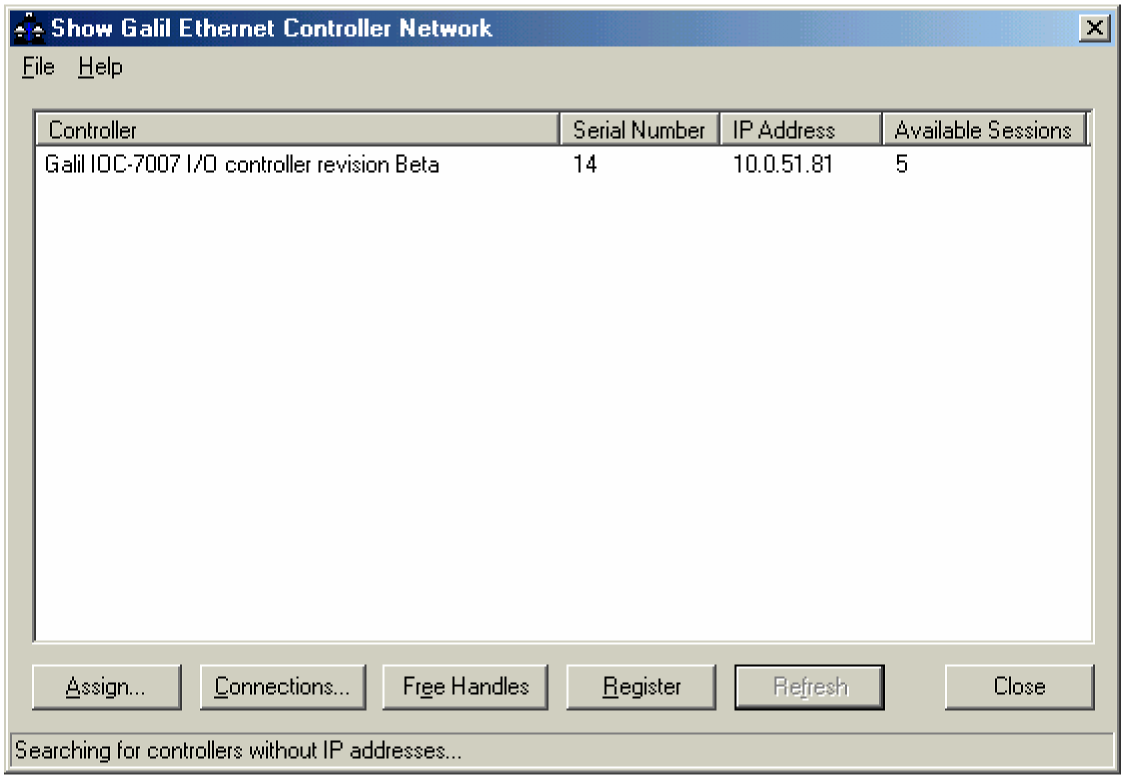
< Back | Finish | Cancel

Fig. 2-5 – Assigning IP Address

In the Ethernet Parameters window there are 3 options for sending Unsolicited Messages. These include "Use current 'CF' Setting", "Receive Through Second Handle", and "Receive Through Same Handle". If "Receive Through Second Handle" is selected, the IOC-7007 will open a second TCP/UDP handle between it and the computer over which unsolicited messages are sent. With DMCTERM, a second thread listens for messages, which provides a faster response when compared to receiving messages through the same handle. If "Receive Through Same Handle" is selected, unsolicited message are sent back through the same handle that the terminal is using. Now DMCTERM must poll to get these messages, which slows the response time. The first option "Use current 'CF' setting" is the default setting, and it allows the user to select the method of message transfer with the CF command. Once all the Ethernet parameters are entered, select 'Assign IP Address'. The software will search for Galil devices that do not have IP addresses. Once the IOC-7007 has been found and the IP address is assigned, select 'Finish', and the IOC will be entered in the Galil Registry. Connect to the IOC through the Terminal and be sure to burn in the IP address with the BN command.

Another method of connecting to the IOC-7007 is using the DMCNET utility in the Registry. Select 'Find Ethernet Controller' under "Non PnP Tools", and the DMCNET window will appear and search for all Galil Devices on the network. Once DMCNET is finished searching, the user can highlight the IOC and give it an IP address by selecting the 'Assign' button. From there, the user can add it to the Galil registry by selecting the 'Register' button.

The 'Connections…' button in DMCNET will provide a list of communication handles associated with the IOC-7007. Furthermore, the 'Free Handles…' button frees all handles. See Chapter 3 for more information on Ethernet handles.



Controller	Serial Number	IP Address	Available Sessions
Galil IOC-7007 I/O controller revision Beta	14	10.0.51.81	5

Searching for controllers without IP addresses...

Fig. 2-6 – DMCNET Utility

If the two methods above are unsuccessful in assigning an IP address to the IOC-7007, the third option is connecting to the IOC serially and using the IA command to assign the IP address. The user can then add an Ethernet IOC to the registry with the specified address, and connect to it by opening the terminal utility. It is possible to add an Ethernet IOC to the registry by simply clicking 'Finish' in the Ethernet parameters window (shown above) instead of clicking on 'Assign IP Address'.

When connecting to an IOC-7007 via Ethernet, the user must be aware of the type of Ethernet cable being used, and the method of communication. To connect the IOC directly to the PC, use a crossover or null-modem Ethernet cable. This type of cable allows for the crossing of signals between the PC and the controller. If instead the connection to the IOC is through a network hub, a straight through cable must be used. Hubs perform the signal crossing function of a null-modem cable. If the wrong cable is used, communication with the controller will not be possible.

Note: If the IOC-7007 is connected in a LAN, make sure the assigned IP address is allowed. Also, Galil strongly recommends the IP address selected cannot be accessed across the Gateway. The Gateway is an application that controls communication between an internal network and the outside world. Ask your network administrator for acceptable IP addresses.

Sending Test Commands to the Terminal:

After the IOC is connected to the computer or terminal, press <carriage return> or the <enter> key on your keyboard. In response to carriage return (CR), the controller responds with a colon, :

Now type

```
TZ (CR)
```

This command directs the IOC to return the current I/O status. The controller should respond with something similar to the following.

```
Slot 0 (7-0) = Digital Output – value 0 (0000_0000)
Slot 1 = I/O module not detected
Slot 2 (79-64) = Digital Input – value 65535 (1111_1111_1111_1111)
Slot 3 = I/O module not detected
Sot 4  (135-128) = Digital Input – value 255 (1111_1111)
Slot 5 = I/O module not detected
Slot 6 (195-192) = Digital Output – value 0 (0000)
```

This response shows slots 2 and 4 occupied by input modules, and slots 0 and 6 with output modules.

Step 6. Configure the Distributed Control System

The IOC-7007 can be set up in a distributed control network with other IOCs and DMC-3425 controllers. In this setup, all the DMC-3425s and IOCs act as if they were one multi-axis controller. The host computer communicates to only one master controller, and the master coordinates data transfer between the other controllers and IOCs in the network.

The HC command, available on the DMC-3425, is used to automatically configure the various controllers and IOCs in a distributed network. If this command is used, then each IOC must have a jumper corresponding to its address setting in the distributed network. See *Step 1. Configure Jumpers* for information on these jumper settings. Also, if the automatic configuration is used, then the IOC **should not** have an IP address assigned to it. When issued by the DMC-3425, the HC command assigns IP addresses automatically to all devices configured for distributed control.

All information pertaining to the HC command and distributed control is covered in detail in the DMC-3425 User Manual.

Chapter 3 Communication

Introduction

The IOC-7007 has one RS-232 port and one Ethernet port. The RS-232 port is the data set, and it is a standard serial link with communication baud rates up to 19.2kbaud. The Ethernet port is an auto-negotiating 10/100Base-T link.

RS232 Port

The IOC board has a single RS232 connection for sending and receiving commands from a PC or other terminal. The pin-outs for the RS232 connection are as follows.

RS232 - Port 1 DATATERM

1 CTS – output	6 CTS – output
2 Transmit Data - output	7 RTS – input
3 Receive Data - input	8 CTS – output
4 RTS – input	9 No connect (Can connect to +5V)
5 Ground	

RS-232 Configuration

Configure the PC for 8-bit data, one start-bit, one stop-bit, full duplex and no parity. The baud rate for the RS232 communication can be selected by using the proper jumper configuration on the IOC-7007 according to the table below:

Baud Rate Selection

JUMPER SETTINGS		BAUD RATE
9600	**1200**	**--**
OFF	OFF	19200
ON	OFF	9600
OFF	ON	1200

Handshaking Modes

The RS232 port is configured for hardware handshaking. In this mode, the RTS and CTS lines are used. The CTS line will go high whenever the IOC-7007 is not ready to receive additional characters. The RTS line will inhibit the IOC board from sending additional characters. **Note:** The RTS line goes high for inhibit. This handshake procedure ensures proper communication especially at higher baud rates.

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 IOC-7007 supports two industry standard protocols, TCP/IP and UDP/IP. The board 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/IP does not require a "connection". This protocol is similar to communicating via RS232. If information is lost, the IOC board does not return a colon or question mark. Because the protocol does not provide for lost information, the sender must re-send the packet.

Although UDP/IP is more efficient and simple, Galil recommends using the TCP/IP protocol. TCP/IP insures that if a packet is lost or destroyed while in transit, it will be resent.

Ethernet communication transfers information in ‘packets’. The packets must be limited to 470 data bytes or less. Larger packets could cause the IOC-7007 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 IOC-7007 Ethernet address is set by the factory and the last two bytes of the address are the serial number of the board.

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 IOC board can be done in a number of ways.

The first method is to use the BOOT-P utility via the Ethernet connection (the IOC-7007 must be connected to the network and powered). For a brief explanation of BOOT-P, see the section: *Third Party Software (page 20)*. 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 terminal emulator. **NOTE: For Windows 95 and 98 FE, Select the DMC-1415 controller.** Once the model type is selected, click on the Ethernet Parameters tab. Enter the IP address at the prompt and select either TCP/IP or UDP/IP as the protocol. When done, click on the ASSIGN IP ADDRESS. The Galil Terminal Software will respond with a list of all I/O boards and controllers on the network that do not currently have IP addresses. The user selects the board, and the software will assign the specified IP address to it. Then enter the terminal and type in BN to save the IP address to the non-volatile memory.

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 IOC board upon linking it to the network. In order to ensure that the IP address is correct, please contact your system administrator before connecting the I/O board to the Ethernet network.

The second method for setting an IP address is to send the IA command through the IOC-7007 main RS-232 port. The IP address may be entered as a 4 byte number delimited by commas (industry standard uses periods) or a signed 32 bit number (e.g. IA 124,51,29,31 or IA 2083724575). Type in BN to save the IP address to the IOC non-volatile memory. At this point, it is a good practice to do a manual reset on the IOC board before connecting it to the network.

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 board does not require a specific port number. The port number is established by the client or master each time it connects to the IOC board.

Communicating with Multiple Devices

The IOC-7007 is capable of supporting multiple masters and slaves. The masters may be multiple PCs and motion controllers that send commands to the I/O board. The slaves can be other IOC's, Galil Ethernet controllers such as the DMC-3425 or DMC-21x0, or I/O devices manufactured by other companies.

NOTE: The term "Master" is equivalent to the Internet "client". The term "Slave" is equivalent to the Internet "server".

An Ethernet handle is a communication resource within a device. The IOC-7007 can have a maximum of 6 Ethernet handles open at any time. When using TCP/IP, each master or slave uses 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 6 handles are in use and a 7th master tries to connect, it will be sent a "reset packet" that generates the appropriate error in its windows application.

NOTE: There are a number of ways to reset the board. Hardware reset (push reset button or power down IOC board) and software resets (through Ethernet or RS232 by entering the RS command). The only reset that will not cause the board to disconnect is a software reset via the Ethernet.

When the IOC-7007 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, but if it is not, it will default to 1000. The protocol (TCP/IP or UDP/IP) to use must also be designated at this time. Otherwise, the board will not connect to the slave. (Ex: IHB=151,25,255,9<179>2. This will open handle #2 and connect to the IP address 151.25.255.9, port 179, using TCP/IP)

Once the IH command is used to connect to slaves, the user can communicate to these slaves by sending commands to the master IOC-7007. The SA command is used for this purpose, and it has the following syntax.

```
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 to a DMC-3425 controller; 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. 1 indicates waiting for the acknowledgement from the slave. 2 indicates a colon (command accepted) has been received. 3 indicates a question mark (command rejected) has been received. 4 indicates the command timed out.

If a command generates responses (such as the TE command on a DMC-3425 controller), the values will be stored in _SAh0 thru _SAh7. If a field is unused, its _SA value will be -2^31.

See Chapter 5 for more information on the SA command.

Which devices receive what information from the IOC-7007 depends on various things. If a device queries the IOC board, it will receive the response unless it explicitly tells the IOC board to send it to another device. If the command that generates a response is part of a downloaded program, the response will route to whichever port is specified by the CF command (either a specific Ethernet handle or the RS232 port). If the user wants to send the message to a port other than what is specified by the CF command, add an {Eh} or {S1} to the end of the command (Ex. MG{EC}"Hello" will send the message "Hello" to handle #3 and MG{P1}”Hello” will send it to the serial port).

IOC-7007 Support in Galil’s Distributed network

There are two levels of distributed control supported by the IOC-7007. The first method, as described above, uses the IH command to assign Ethernet handles and connect to slave devices. The other method is a custom Galil architecture in which several DMC-3425 controllers and IOCs act as if they were one multi-axis controller with extended I/O. The host computer communicates to only one master controller, and the master coordinates data transfer between the other controllers and IOCs in the network.

The HC command, available on the DMC-3425, is used to automatically configure the various controllers and IOCs in this distributed network. If this command is used, then each IOC must have a jumper corresponding to its address setting in the distributed network. See Chapter 2, *Step 2. Configure Jumpers* for information on these jumper settings. Also, if the automatic configuration is used, then the IOC **should not** have an IP address assigned to it. When issued by the DMC-3425, the HC command assigns IP addresses automatically to all devices configured for Galil’s distributed control.

All information pertaining to the HC command and Galil's distributed control architecture is covered in detail in the DMC-3425 User Manual.

IOC-7007 as Modbus master

An additional protocol layer is available for speaking to I/O devices. Modbus is an RS-485 protocol that combines 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 IOC-7007 assumes a specific slave address is not necessary.

The Modbus protocol has a set of commands called function codes. The IOC-7007 as a Modbus master 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 IOC-7007 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, and array[] is the array with the data.

The second level incorporates the Modbus structure. This is necessary for sending configuration and special commands to another device. The formats vary depending on the function code that is called. For more information refer to the MB command in the Command Reference section.

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 6 (F). 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 8.

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.

IOC-7007 as Modbus Slave

Unlike Galil Ethernet controllers, the IOC-7007 can be a ModBus slave as well as a master. Being a slave means that another IOC or Galil Controller can send and receive information from an IOC using all three levels of Modbus communication described above. The port used to communicate to an IOC as a ModBus slave is 502 (use the IH command to set the port number), and the IOC supports function codes 1-7 and 16 as a slave. See the table above for a description of the function codes. If the third level of ModBus communication is used to communicate from a Galil master device to a slave IOC then the I/O number calculation is as follows

I/O Number = (HandleNum*1000) + (BitNum)

where the bit number is from 0 to 223.

The slave address is not needed. See the beginning of Chapter 4 for the bit number assignments on the IOC-7007.

Handling Communication Errors

A new automatic subroutine which is identified by the label #TCPERR, has been added. If an IOC-7007 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 the 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/IP 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 devices belong to a default multicast address of 239.255.19.56. This 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 IOC-7007. By default this multicast handle will be opened.

Unsolicited Message Handling

Unsolicited messages are internal responses from programs, responses from internal errors and messages from the MG command. There are two software commands that will configure how the controller handles these messages: CW and CF.

The IOC-7007 has 6 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 description in Chapter 5.

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 Terminal and WSDK will sometimes 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.

The second field of the CW command controls whether the product should pause while waiting for the hardware handshake to enable the transmission of characters over RS-232 (CW,0), or continue processing commands and lose characters until the hardware handshake allows characters to be sent (CW,1).

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 IOC-7007 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.

Data Record

QR Command

The IOC-7007 can provide a block of status information back to the host computer with the use of a single command, QR. This command can be very useful for accessing board status. No arguments are needed for this command.

The QR command returns all information in binary format; therefore the result of this command cannot be displayed in a Galil terminal. However, the Galil API function calls for use in a C/C++ programming environment can be used to extract information from the IOCs data record. Also, certain features of Galil's ActiveX Tool Kit utilize the data record as well. See the C/C++ Programmer and ActiveX Tool Kit manuals for more information.

The QR command will return 4 bytes of header information, followed by an entire data record. A data record map is provided below.

Data Record Map

DATA TYPE	**ITEM**
UB	1st byte of header
UB	2nd byte of header
UB	3rd byte of header
UB	4th byte of header
UW	Sample number
UB	Error Code
UB	General Status
UB	Module Detection in slot 0 (0=nothing in slot, 1=module detected)

UB	Module Type (0=digital else analog bit count)
UB	Number of I/O points
UB	Direction (0=output, 1=input)
UB	Range if Analog
UB	Range Type (0=unipolar, 1=bipolar)
UW	Analog Channel 0 Data or Digital Data (bits 16-31)
UW	Analog Channel 1 Data or Digital Data (bits 0-15)
UW	Analog Channel 2 Data or 0
UW	Analog Channel 3 Data or 0
UW	Analog Channel 4 Data or 0
UW	Analog Channel 5 Data or 0
UW	Analog Channel 6 Data or 0
UW	Analog Channel 7 Data or 0

Note: UB=Unsigned Byte, UW=Unsigned Word, SL=Signed Long Word

This data map can be broken up into sections. The **Data Record Map** includes the 4 bytes of header. The **General Data Block** consists of the sample number, the error code, and the general status. **The Slot Data Block** includes all the other items in the above table (module detection, module type etc.)

Note: This data record table only contains the **Slot Data Block** for slot 0 of the IOC-7007. The entire IOC data record contains a **Slot Data Block** for each slot. If a slot is not occupied by an IOM module then its **Slot Data Block** is all zeros.

Explanation of Status Information

Header Information - Bytes 0, 1 of Header:

The first two bytes of the data record provide the

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	N/A	N/A	N/A

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
N/A	Slot 6 Present in Data Record	Slot 5 Present in Data Record	Slot 4 Present in Data Record	Slot 3 Present in Data Record	Slot 2 Present in Data Record	Slot 1 Present in Data Record	Slot 0 Present in Data Record

Bytes 2, 3 of Header:

Bytes 2 and 3 make up a word, which 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

QZ Command

The QZ command can be very useful when using the QR command, since it provides information about the data record. The QZ command returns four fields of data, the first field being the number of slots in the IOC board (currently 7 for the IOC-7007). The second field is the number of bytes in the **General Data Block** of the data map, and the fourth field contains the number of bytes in all the **Slot Data Blocks**. The third field is zero and has no significance. See the paragraph under the **Data Record Map** above for more information on the **General Data Block** and the **Slot Data Blocks**.

LR Command

The LR command returns similar information as the QR, but works between the IOC-7007 and a DMC-3425 controller. The DMC-3425 supports Galil's Distributed Control System that allows several DMC-3425s and IOCs to be connected together as a single virtual axis controller. In this system, one of the DMC-3425s is designated as the master. The maser can receive commands from the host computer that apply to all the other controllers and IOCs in the system. Therefore, in order for this system to coordinate properly, data must be constantly sent from the master controller to the other devices and visa versa. The data, coming from the slaves to the master, is sent in large packets, or slave records. Typically, these slave records are sent every 'n' msec where 'n' is specified by the QW command. However, the LR command can be used to send the slave record at any time so that the master can be immediately notified of important slave events. On the IOC-7007, such an event could be a particular input being triggered. For more information on distributed control and the QW command, see the DMC-3425 User Manual and Command Reference.

Another important feature of the slave data record is that it contains two variables that can be set by the user. The ZC and ZD commands are responsible for these variables. Each variable can be a number, a mathematical equation, or a string. See the Chapter 5 Command Reference for more information on the ZC and ZD commands.

The example below shows the use of the LR command by an IOC-7007. The DMC-3425 is the IOCs master, and data being sent from the slave to the master is on TCP handle B. Notice that once inputs 3 and 2 go low, the ZC and ZD variables are set to "3L" and "2L" in the IOCs program. The slave record (containing these variables) is then sent to the DMC-3425 and the message, "IOC INS 3 & 2 ARE LOW", is sent to the host computer.

Program on the IOC-7007	**Program on the DMC-3425**
#A	#B
#C;JP#C,@IN[3]=1	JP#D, (ZCB="3L")&(ZDB="2L")
#E;JP#E,@IN[2]=1	JP#B
ZC "3L"; ZD "2L	#D
LR	MG"IOC INS 3 & 2 ARE LOW"
EN	EN

The IOC-7007 does not necessarily have to be in Galil's distributed control network to utilize the LR command. If the IOC is a slave to a DMC-3425 or another IOC, the ZA and ZB variables will still be recognized, even if the rest of the slave record is irrelevant. Due to the fact that the slave record is only meant to be read by a master controller, the details of its contents are not described in detail. Essentially it includes the I/O status of the 7 IOC slots.   Like the QR record, the LR data is sent in binary format and cannot be viewed from the Galil Terminal. Unlike the QR, the LR data cannot be accessed from the API function calls or the ActiveX toolkit.

Chapter 4 IOM Modules

Introduction

The base IOC-7007 controller can accept up to 7 plug in I/O IOM modules that can be "mixed and matched" according to the system requirements. The IOM modules currently available are listed below:

IOM-70016:	16 TTL inputs
IOM-70108:	8 optoislolated inputs
IOM-70208:	8 optoislolated outputs
IOM-70308:	8 high power outputs
IOM-70404:	4 dry contact relays
IOM-70808:	8 analog inputs
IOM-70908:	8 analog outputs
IOM-70904:	4 analog outputs

The IOM modules are plugged into slots 0 through 6 on the IOC-7007. The IOC automatically recognizes the type of modules that are plugged into these slots, and no configuration setup is required of the user.

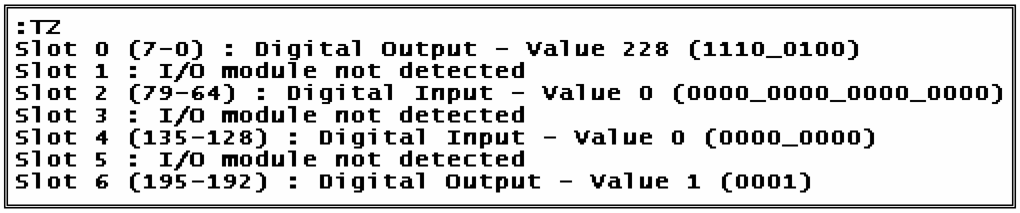
Note: If custom I/O modules are desired, contact Galil.

Each slot in the IOC-7007 is configured to provide up to 32 I/O channels. The I/O numbering assignments are shown in the following table.

SLOT	I/O RANGE
0	0-31
1	32-63
2	64-95
3	96-127
4	128-159
5	160-191
6	192-223

These I/O ranges remain the same despite the type of IOM module plugged into the slot. This means that if a relay output module, which has 4 relay channels, is plugged into slot 4, the outputs associated with that module are numbered OUT[128] through OUT[131]. Likewise, if the TTL input module, which has 16 input channels, is plugged into slot 4 instead, the inputs are numbered IN[128] through IN[143]. Currently, none of the IOM modules use all 32 I/O channels associated with a slot. Future IOM modules may use all the allocated I/O space, but for now, all additional numbers are disregarded.

A new interrogation command, **TZ**, has been created to allow the user to get a quick view of every slot configuration and their complete I/O bit status. The following is an example of the response for the TZ command, where only slots 0, 2, 4, and 6 are occupied. The I/O range available is specified in parenthesis after each slot number



```
:TZ
Slot 0 (7-0) : Digital Output - Value 228 (1110_0100)
Slot 1 : I/O module not detected
Slot 2 (79-64) : Digital Input - Value 0 (0000_0000_0000_0000)
Slot 3 : I/O module not detected
Slot 4 (135-128) : Digital Input - Value 0 (0000_0000)
Slot 5 : I/O module not detected
Slot 6 (195-192) : Digital Output - Value 1 (0001)
```

Slot 0 has an 8 channel digital output module with bits 2, 5, 6, and 7 on. The decimal value of these bits being set is 228. Also, slot 2 is occupied by a 16 channel input module. Since none of the inputs are high, the decimal value is zero.

Module Specifications

Access to the I/O channels is made through the 20 pin Molex connectors on the top of the IOM modules. Pin outs and module specifications are listed below.

20-Pin Molex Connector

Description	**Molex Part Number**
Molex KK Crimp Terminal/Housing, 0.100" Spacing, Series 2695	22-01-3207
Molex KK Crimp Terminal Pins	08-50-0032

IOM-70016: 16-TTL input module

The IOM-70016 has 16 TTL inputs. The input is guaranteed to be low at 0.8V or below and guaranteed to be hight at 2.2V or above. The inputs are TTL, which typically means a 0-5VDC range. However, the MC3486 is capable of accepting voltages up to 12V. The pin out table and electrical schematic for the IOM-70016 are shown below.

Pin 1	Vcc – 5 Vdc supply output (50 mA max)
Pin 2	GND reference for TTL inputs
Pin 3	TTL input 1
Pin 4	TTL input 2
Pin 5	TTL input 3
Pin 6	TTL input 4
Pin 7	TTL input 5
Pin 8	TTL input 6
Pin 9	TTL input 7
Pin 10	TTL input 8

Pin 11	TTL input 9
Pin 12	TTL input 10
Pin 13	TTL input 11
Pin 14	TTL input 12
Pin 15	TTL input 13
Pin 16	TTL input 14
Pin 17	TTL input 15
Pin 18	TTL input 16
Pin 19	NC
Pin 20	NC

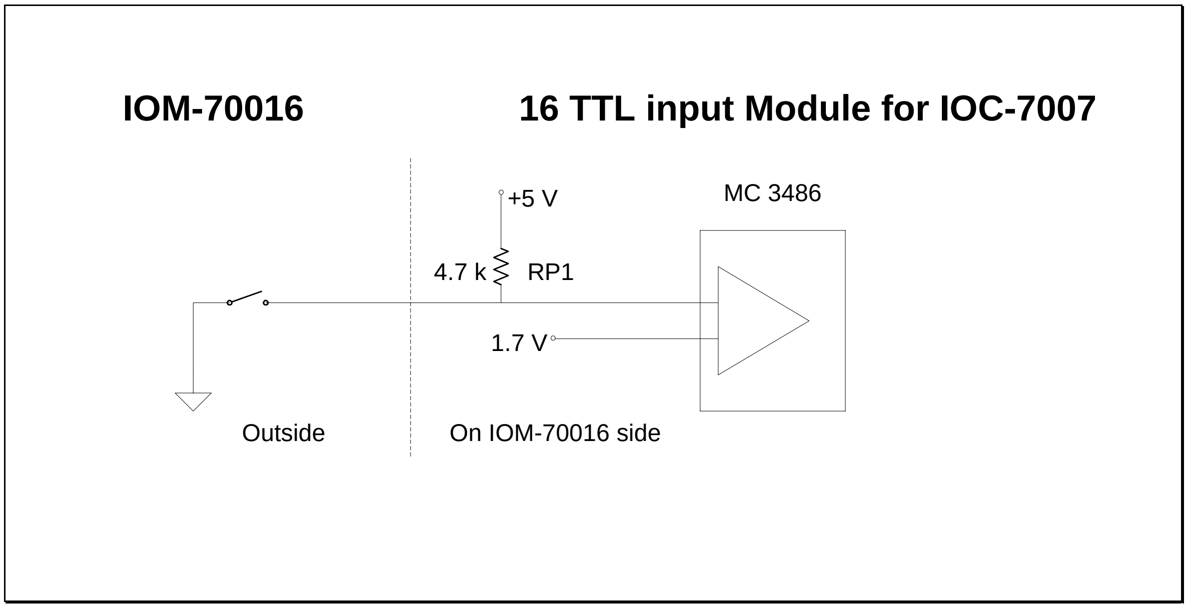


Fig. 4-1 – IOM-70016 Circuit Diagram

IOM-70108: 8 Opto-isolated input module

The IOM-70108 opto-isolated input module has 9 optically isolated inputs. The resistor pack, RP7 on Rev. A and RP2 on Rev. B can be changed to accommodate 5 or 24VDC. When using 5V inputs, the resistor pack should be left with the default 2.2K resistor pack the module shops with. When using 24V on the input, a

10K resistor pack should be used. The inputs are considered low when the current flow is greater than 1mA and high when the current is less than 0.5 mA.

Pin 1	NC
Pin 2	NC
Pin 3	Opto Input 1 (Anode)
Pin 4	Opto Input 1 (Cathode)
Pin 5	Opto Input 2 (Anode)
Pin 6	Opto Input 2 (Cathode)
Pin 7	Opto Input 3 (Anode)
Pin 8	Opto Input 3 (Cathode)
Pin 9	Opto Input 4 (Anode)
Pin 10	Opto Input 4 (Cathode)
Pin 11	Opto Input 5 (Anode)
Pin 12	Opto Input 5 (Cathode)
Pin 13	Opto Input 6 (Anode)
Pin 14	Opto Input 6 (Cathode)
Pin 15	Opto Input 7 (Anode)
Pin 16	Opto Input 7 (Cathode)
Pin 17	Opto Input 8 (Anode)
Pin 18	Opto Input 8 (Cathode)
Pin 19	NC
Pin 20	NC

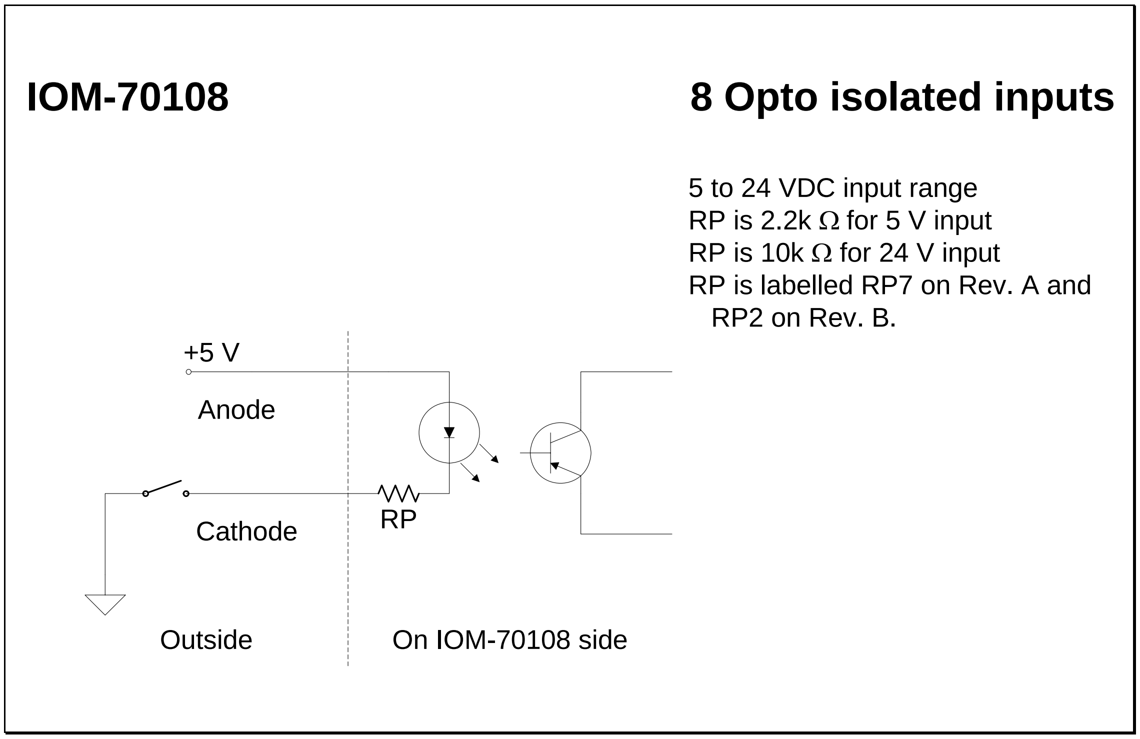

IOM-70108
8 Opto isolated inputs
5 to 24 VDC input range
RP is 2.2k Ω for 5 V input
RP is 10k Ω for 24 V input
RP is labelled RP7 on Rev. A and RP2 on Rev. B.
+5 V
Anode
Cathode
RP
Outside
On IOM-70108 side

Fig. 4-2 – IOM-70108 Circuit Diagram

Note: This diagram shows the positive side of the external power supply connected to the Anode and the negative or ground pin being connected to the Cathode. Since the internal diode allows current flow in only one direction, the Anode must be at a higher potential than the Cathode.

IOM-70208: 8 Opto-isolated output module

The IOM-70208 opto-isolated output module offers 8 optically isolated outputs that accept between 5 and 24VDC. Each output channel is capable of handling up to 25mA. An external power supply must be provided for powering the external load and connecting to the **Opto Output** pins on the IOM.

Pin 1	NC
Pin 2	NC
Pin 3	Opto Output 1 (Collector)
Pin 4	Opto Output 1 (Emitter)
Pin 5	Opto Output 2 (Collector)
Pin 6	Opto Output 2 (Emitter)
Pin 7	Opto Output 3 (Collector)
Pin 8	Opto Output 3 (Emitter)

Pin 9	Opto Output 4 (Collector)
Pin 10	Opto Output 4 (Emitter)
Pin 11	Opto Output 5 (Collector)
Pin 12	Opto Output 5 (Emitter)
Pin 13	Opto Output 6 (Collector)
Pin 14	Opto Output 6 (Emitter)
Pin 15	Opto Output 7 (Collector)
Pin 16	Opto Output 7 (Emitter)
Pin 17	Opto Output 8 (Collector)
Pin 18	Opto Output 8 (Emitter)
Pin 19	NC
Pin 20	NC

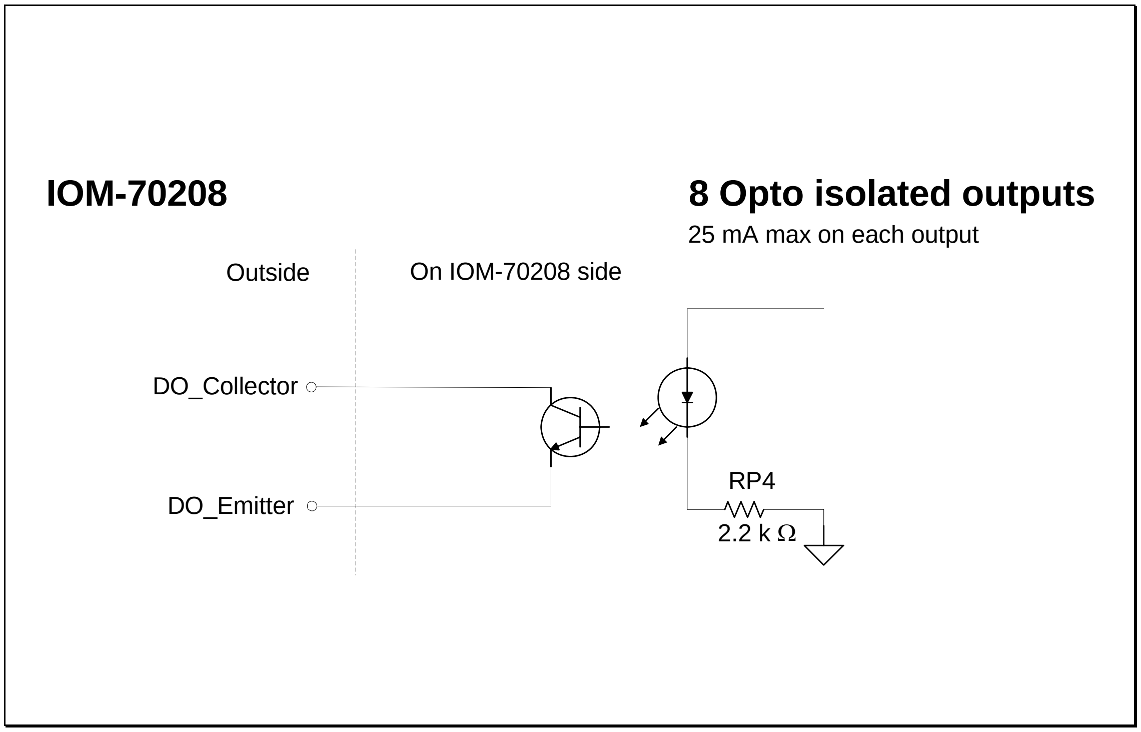


Fig. 4-3 – IOM-70108 Circuit Diagram

IOM-70308: 8 Opto-isolated power output module

The IOM-70308 provides 8 optically isolated power output channels that source up 100mA each. As with the IOM-70208, the voltage range of the 70308 is between 5 and 24VDC. The user will connect an isolated power supply to **VIN_ISO** and **GND_ISO**, and the load will be connected to one of the **Power Output** pins and GND_ISO. Current will flow from the Power Out pin to GND_ISO. To reverse the logic of the output, RPACK RP301, may be reversed (Rev B1 and higher)

Pin 1	NC
Pin 2	NC
Pin 3	Power Output 1
Pin 4	Power Output 2
Pin 5	Power Output 3
Pin 6	Power Output 4
Pin 7	Power Output 5
Pin 8	Power Output 6
Pin 9	Power Output 7
Pin 10	Power Output 8
Pin 11	NC
Pin 12	NC
Pin 13	NC
Pin 14	NC
Pin 15	NC
Pin 16	NC
Pin 17	NC
Pin 18	NC
Pin 19	VIN_ISO, Input for Power supply (+)
Pin 20	GND_ISO, Input for Return supply (-)

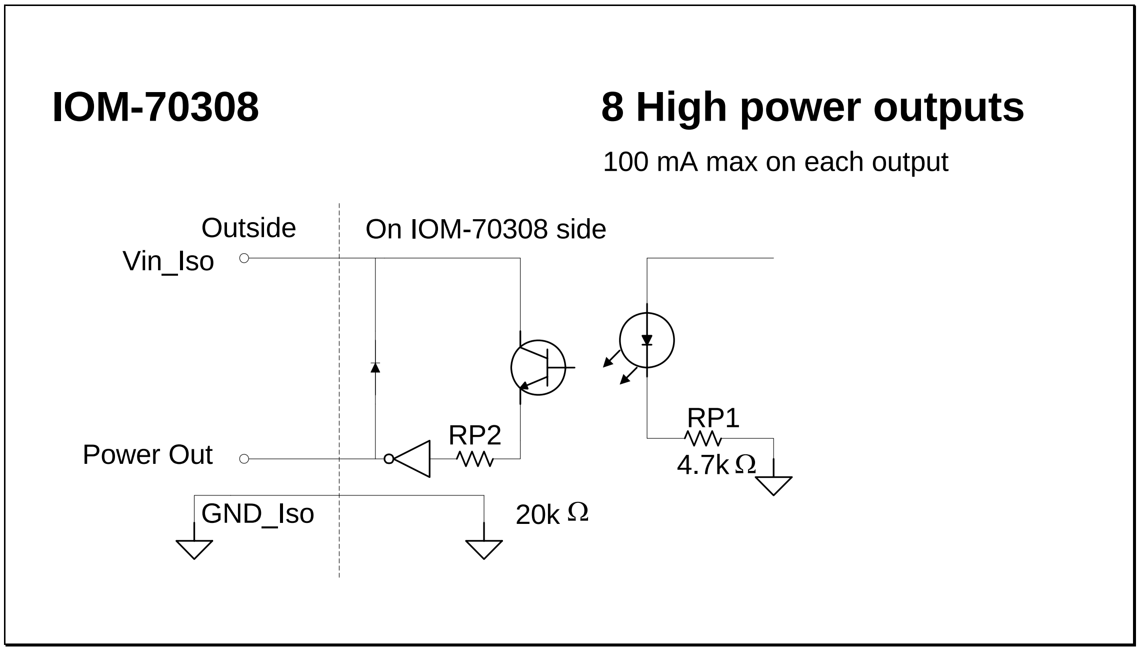


Fig. 4-4 – IOM-70308 Circuit Diagram

IOM-70404: 4 Dry contact relay output module

The IOM-70404 features 4 dry contact relay outputs that allow up to 150V and 250mA per channel. The voltage can be either AC or DC, and the user can wire the output to be either normally open or normally closed. If he wants to have a normally open output, one lead of his external circuit will be connected to the **Relay Out Common**, and the other lead will be connected to **Relay Output Normally Open**. Likewise, if a normally closed output is desired, connect to **Relay Out Common** and **Relay Output Normally Close**. Current can flow in either direction through these output channels.

Pin 1	NC
Pin 2	NC
Pin 3	Relay Output Common 1
Pin 4	Relay Output Normally Open 1
Pin 5	Relay Output Normally Close 1
Pin 6	Relay Output Common 2
Pin 7	Relay Output Normally Open 2
Pin 8	Relay Output Normally Close 2
Pin 9	Relay Output Common 3
Pin 10	Relay Output Normally Open 3
Pin 11	Relay Output Normally Close 3
Pin 12	Relay Output Common 4
Pin 13	Relay Output Normally Open 4
Pin 14	Relay Output Normally Close 4
Pin 15	NC
Pin 16	NC
Pin 17	NC
Pin 18	NC
Pin 19	NC
Pin 20	NC

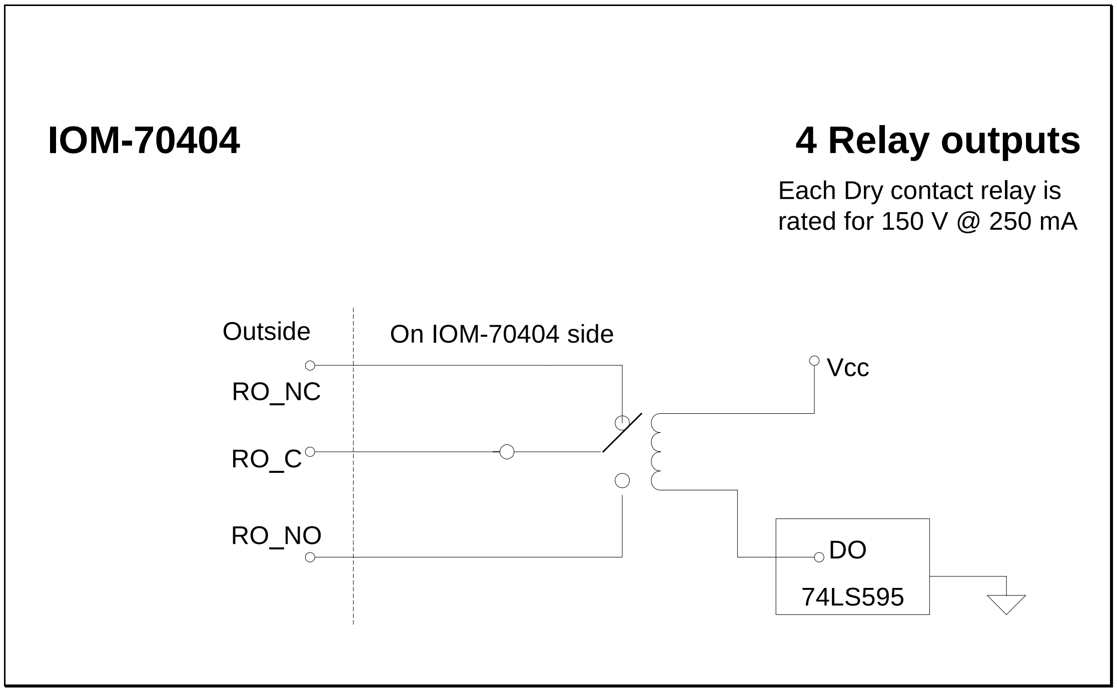


Fig. 4-5 – IOM-70404 Circuit Diagram

IOM-70508: 8 Opto-isolated high power outputs

The IOM-70508 provides 8 optically isolated high power outputs that are capable of sourcing 500mA of current. The voltage range is 12VDC-24VDC. The user will connect an external power supply to VIN_ISO and GND_ISO. The load will be connected to one of the Power Output pins and GND_ISO. Current will flow from the Power Output pin to GND_ISO. To reverse the logic of the output RP502 may be rotated 180 degrees.

Pin 1	NC
Pin 2	NC
Pin 3	Power Output 1 (+)
Pin 4	Power Output 2 (+)
Pin 5	Power Output 3 (+)
Pin 6	Power Output 4 (+)
Pin 7	Power Output 5 (+)
Pin 8	Power Output 6 (+)
Pin 9	Power Output 7 (+)

Pin 10	Power Output 8 (+)
Pin 11	NC
Pin 12	NC
Pin 13	NC
Pin 14	NC
Pin 15	VIN_ISO, Input for power supply (+)
Pin 16	VIN_ISO, Input for power supply (+)
Pin 17	VIN_ISO, Input for power supply (+)
Pin 18	VIN_ISO, Input for power supply (+)
Pin 19	VIN_ISO, Input for power supply (+)
Pin 20	Ground_ISO, Input for return supply (-)

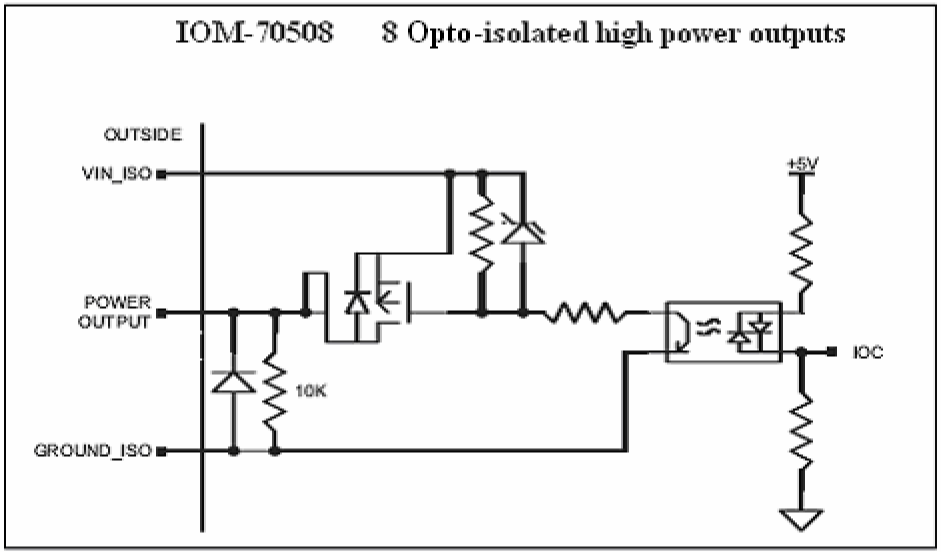


Fig 4-6 - IOM-70508 Circuit Diagram

IOM-70808: 8 Analog input module

The IOM-70808 features 8 analog inputs with12 bit resolution. The standard DC voltage range for each input is –10V to 10V. The other ranges available on request are as follows:

0V to 10V

0V to 5V

-5V to 5V

The input impedance for each input point is 10 MOhms despite the voltage range listed above.

The IOM-70808P offers 8 analog inputs with 16 bit resolution. Currently, only the –10V to 10V range is available. However, 0V to 5V will be available in the future. The input impedance for each +/-10V input on an IOM-70808P is 45.7 kOhms. The 0V to 5V inputs will have 20 kOhms of impedance per channel.

Pin 1	NC
Pin 2	NC
Pin 3	Analog Input 1
Pin 4	Ground
Pin 5	Analog Input 2
Pin 6	Ground
Pin 7	Analog Input 3
Pin 8	Ground
Pin 9	Analog Input 4
Pin 10	Ground
Pin 11	Analog Input 5
Pin 12	Ground
Pin 13	Analog Input 6
Pin 14	Ground
Pin 15	Analog Input 7
Pin 16	Ground
Pin 17	Analog Input 8
Pin 18	Ground
Pin 19	NC
Pin 20	NC

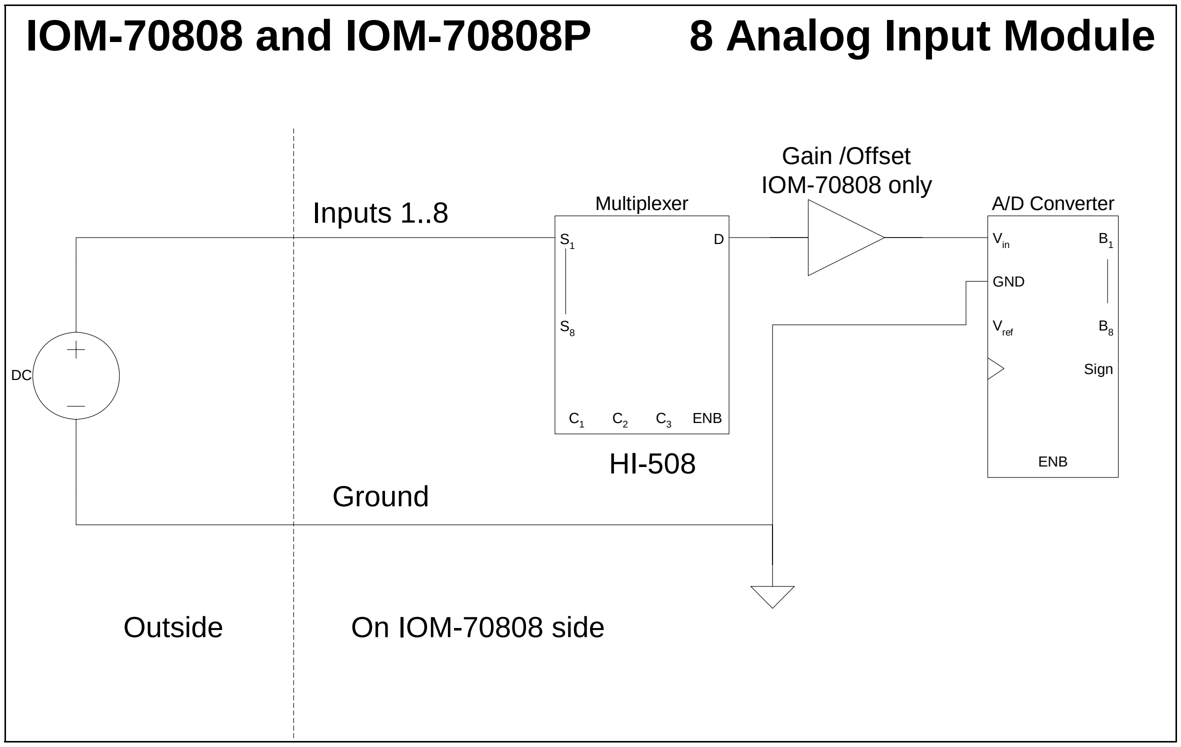


Fig. 4-7 – IOM-70808 Circuit Diagram

IOM-70908: 8 Analog output module

The IOM-70908 features 8, 12 bit analog outputs with a standard DC voltage range of –10V to 10V. The other ranges available on request are as follows.

0V to 10V

0V to 5V

-5V to 5V

The AO command is used to set the output voltage on the individual pins. Each of the output pins can source up to 3mA.

Pin 1	NC
Pin 2	NC
Pin 3	Analog Output 1
Pin 4	Ground
Pin 5	Analog Output 2
Pin 6	Ground
Pin 7	Analog Output 3

Pin 8	Ground
Pin 9	Analog Output 4
Pin 10	Ground
Pin 11	Analog Output 5
Pin 12	Ground
Pin 13	Analog Output 6
Pin 14	Ground
Pin 15	Analog Output 7
Pin 16	Ground
Pin 17	Analog Output 8
Pin 18	Ground
Pin 19	NC
Pin 20	NC

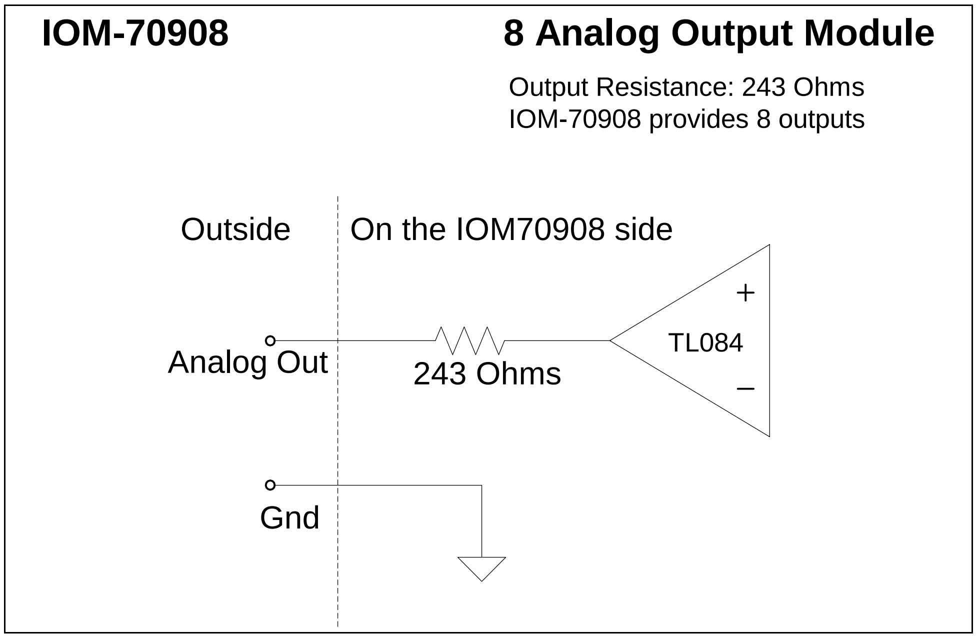


Fig. 4-8 – IOM-70908 Circuit Diagram

IOM-70904: 4 Analog output module

The IOM-70904 features 4, 12 bit analog outputs with a range from –10V DC to +10V DC (No other voltage ranges are available). The AO command is used to set the output voltage on the individual pins. Each of the output pins can source up to 3mA.

Pin 1	NC
Pin 2	NC
Pin 3	Analog Output 1
Pin 4	Ground
Pin 5	Analog Output 2
Pin 6	Ground
Pin 7	Analog Output 3
Pin 8	Ground
Pin 9	Analog Output 4
Pin 10	Ground
Pin 11	NC
Pin 12	NC
Pin 13	NC
Pin 14	NC
Pin 15	NC
Pin 16	NC
Pin 17	NC
Pin 18	NC
Pin 19	NC
Pin 20	NC

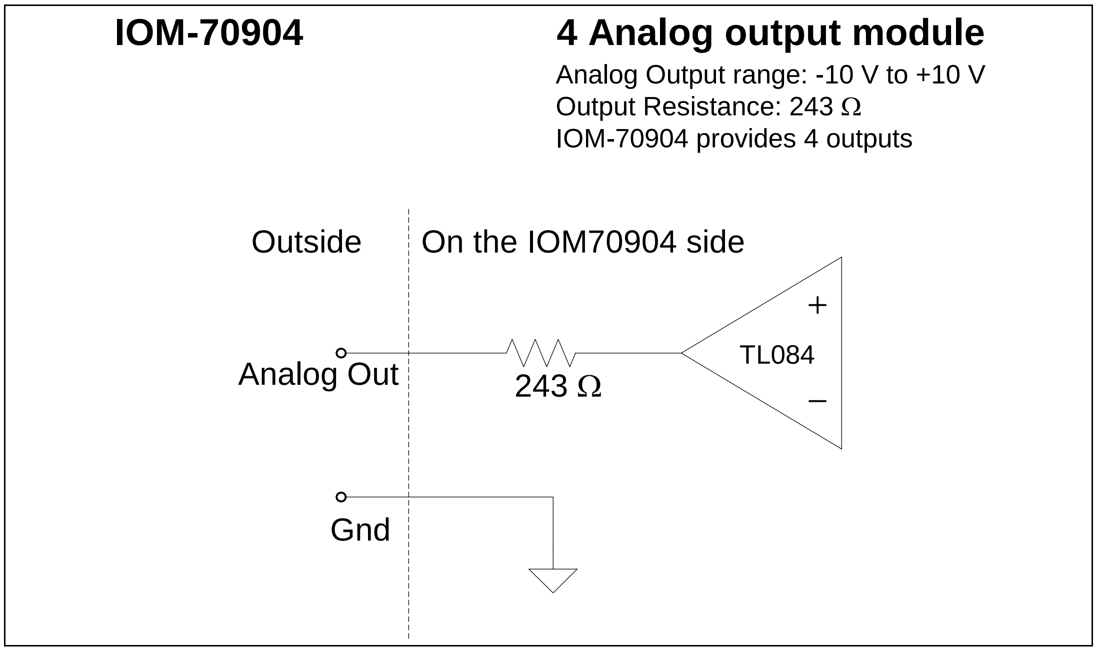


Fig. 4-9 – IOM-70904 Circuit Diagram

THIS PAGE LEFT BLANK INTENTIONALLY

Chapter 5 Command Reference

The Instruction Set section provided in the beginning of this chapter helps to identify commands belonging to the same functional group. To find out more details of each command, the Commands section in this chapter lists each executable instruction in alphabetical order.

Instruction Set

Note: Commands in bold are newly added or reformatted. Commands with a '*' preceding their description are available for the PLC Mode.

Ethernet Commands

AO	*Analog output voltage
IA	Set IP address
IH	Internet handle
MB	ModBus
HS	Handle Assignment Switch

I/O Commands

CB	*Clear bit
II	Input interrupt
OB	*Define output bit
OQ	*Output port
SB	*Set bit

Interrogation Commands

LA	List arrays
LL	List labels
LR	Launch slave record
LS	List program
LV	List variables
MG	Message command
QR	Data record
QZ	Return data record information
^R^V	Firmware revision information
TB	Tell status
TC	Tell error code
TH	Tell Ethernet handle

TI	Tell input
TIME	*Time operand, internal clock
TR	Trace program
TQ	Tell thread execution
TZ	Tell I/O configuration
WH	Which Handle

Math/Special Functions

@SIN[x]	Sine of x
@COS[x]	Cosine of x
@COM[x]	1's compliment of x
@ASIN[x]	Arc sine of x
@ACOS[x]	Arc cosine of x
@ATAN[x]	Arc tangent of x
@ABS[x]	Absolute value of x
@FRAC[x]	Fraction portion of x
@INT[x]	Integer portion of x
@RND[x]	Round of x
@SQR[x]	Square root of x
@IN[x]	*State of digital input x
@OUT[x]	*State of digital output x
@AN[x]	*Value of analog input x
@AO[x]	*Value of analog output x
+,-,*,/	*Arithmetic commands
>,<,==,>=,<=,<>	*Logical operators
&	*Logical AND
|	*Logical OR

Programming Commands

AB	Abort
DA	Deallocate variables/arrays
DL	Download program
DM	Dimension arrays
ED	Edit program
ELSE	* Conditional statement
EN	* End program
ENDIF	* End of conditional statement
HX	* Halt execution
IF	* If statement
IN	Input variable
JP	* Jump
JS	Jump to subroutine
NO	* No-operation—for remarks
RA	Record array, automatic data capture
RC	Record interval for RA
RD	Record data for RA
RI	Return from interrupt routine
SA	Send command
UL	Upload program
XQ	* Execute program
ZC	User variable
ZD	User variable

ZS	Zero stack

System Configuration

BN	Burn parameters
BP	Burn program
BV	Burn variables and arrays
CF	Configure default port
CW	Data adjustment bit
EO	Echo off
LZ	Leading zeros format
QD	Download array
QU	Upload array
RS	Reset
^R^S	Master reset
^R^V	Revision information
VF	Variable format

Trippoint Commands

AA	After analog input
AI	After digital input
AT	At time
WT	Wait for time

PLC Mode Commands

CP	* Compile PLC thread
HP	* Halt PLC thread
TX	* Tell PLC execution time
XP	* Execute PLC thread

Commands

AA

FUNCTION: After Analog Trippoint

DESCRIPTION:

AA is a trippoint that halts program execution until a voltage on a particular analog input has been reached

ARGUMENTS: AA m,n where

m is the I/O number assigned to a particular analog input pin on one of the IOM modules

n = the voltage which ranges from 9.99 to –9.99

USAGE:

In a Program	Yes
Command Line	Yes
Can be Interrogated	No
Used as an Operand	No

RELATED COMMANDS:

@AN[x]	Function to query the voltage on an analog input

AB

FUNCTION: Abort

DESCRIPTION:

AB (Abort) aborts the application programs including any of the 8 simultaneous threads running and the #PLCSCAN thread

ARGUMENTS: AB

USAGE:

In a Program	Yes
Command Line	Yes

RELATED COMMANDS:

“HX”	Halt Execution
“HP”	Halt #PLCSCAN thread

AI

FUNCTION: After Input

DESCRIPTION:

The AI command is used in programs to wait for a specified input pattern. If the parameter entered is positive, it waits for that input to go high. If negative, it waits for that input to go low.

ARGUMENTS: AI $\pm m_1 \& \pm m_{2\&} \pm m_3 \&$ … where

m_n is an integer in the range 0 to 223 decimal. When m_n is positive, the IOC will wait for the input to go high. When m_n is negative, the IOC will wait for the input to go low.

USAGE:

In a Program	Yes
Command Line	Yes
Can be Interrogated	No
Used as an Operand	No

RELATED COMMANDS:

@IN[n]	Function to read inputs
"II"	Input interrupt

EXAMPLES:

#A	Begin Program
AI 7&15&-0&-30	Wait until inputs 7 & 15 are high, and inputs 0 & 30 are low
MG "DONE"	Send message 'DONE' when conditions are satisfied
EN	End Program

HINT: *The AI command actually halts execution until specified input(s) is at desired logic level. Use the conditional Jump command (JP) or input interrupt (II) if you do not want the program sequence to halt.*

AO

FUNCTION: Analog Out

DESCRIPTION:

The AO command sets the analog output voltage on an analog output pin. It is also used to set the analog output voltage on ModBus devices over Ethernet

ARGUMENTS: AO m, n where

m is the I/O number assigned to a particular analog output pin on one of the IOM modules

m can also be the I/O number of a ModBus device that is calculated using the following equations:

m = (SlaveAddress*10000) + (HandleNum*1000) + ((Module-1)*4) + (Bitnum-1)

Slave Address is used when the ModBus device has slave devices connected to it and specified as Addresses 0 to 255. Please note that the use of slave devices for Modbus are very rare and this number will usually be 0.

HandleNum is the handle specifier from A to F (1 to 6).

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.

n = the voltage which ranges from 9.99 to –9.99

USAGE:

In a Program	Yes
Command Line	Yes
Can be Interrogated	No
Used as an Operand	No

RELATED COMMANDS:

"SB" on page 102	Set Bit
“CB" on page 53	Clear Bit

AT

FUNCTION: At Time

DESCRIPTION:

The AT command is a trippoint which is used to hold up execution of the next command until after the specified time has elapsed. The time is measured with respect to a defined reference time. AT 0 establishes the initial reference. AT n specifies n msec from the reference. AT -n specifies n msec from the reference and establishes a new reference after the elapsed time period.

ARGUMENTS: AT n where

n is a signed integer in the range 0 to 2 Billion

n = 0 defines a reference time at current time

positive n waits n msec from reference

negative n waits n msec from reference and sets new reference after elapsed time period

(AT -n is equivalent to AT n; AT 0)

USAGE:

In a Program	Yes
Command Line	Yes

EXAMPLES:

The following commands are sent sequentially

AT 0	Establishes reference time 0 as current time
AT 50	Waits 50 msec from reference 0
AT 100	Waits 100 msec from reference 0
AT –150	Waits 150 msec from reference 0 and sets new reference at 150
AT 80	Waits 80 msec from new reference (total elapsed time is 230 msec from previous command)

BN

FUNCTION: Burn

DESCRIPTION:

The BN command saves certain board parameters in non-volatile EEPROM memory. This command typically takes 1 second to execute and must not be interrupted. The IOC board returns a: when the Burn is complete.

USAGE:

In a Program	Yes
Command Line	Yes
Can be Interrogated	No
Used as an Operand	Yes

PARAMETERS SAVED DURING BURN:

CB	OQ
CW	SB
EO	TR
IA	VF
LZ	

OPERAND USAGE:

_BN contains the serial number of the IOC

RELATED COMMANDS:

"BP"	Burn Program
"BV"	Burn Variables

BP

FUNCTION: Burn Program

DESCRIPTION:

The BP command saves the application program in non-volatile EEPROM memory. This command typically takes up to 10 seconds to execute and must not be interrupted. The IOC-7007 returns a : when the Burn is complete.

USAGE:

In a Program	No
Command Line	Yes
Can be Interrogated	No
Used as an Operand	No

RELATED COMMANDS:

"BN"	Burn Parameters
"BV"	Burn Variables

Note: This command may cause the Galil software to issue the following warning "A time-out occurred while waiting for a response from the controller". This warning is normal and is designed to warn the user when the 'controller' (in this case, the IOC board) does not respond to a command within the timeout period. This occurs because this command takes more time than the default timeout of 1 sec. The timeout can be changed in the Galil software, and this warning does not affect the operation of the IOC board or software.

BV

FUNCTION: Burn Variables and Array

DESCRIPTION:

The BV command saves the defined variables and arrays in non-volatile EEPROM memory. This command typically takes up to 2 seconds to execute and must not be interrupted. The IOC-7007 returns a : when the Burn variables is complete.

ARGUMENTS: None

USAGE:

In a Program	Yes
Command Line	Yes
Can be Interrogated	No
Used as an Operand	No

RELATED COMMANDS:

"BN"	Burn Parameters
"BP"	Burn Program

Note: This command may cause the Galil software to issue the following warning "A time-out occurred while waiting for a response from the controller". This warning is normal and is designed to warn the user when the ‘controller’ (in this case, the IOC board) does not respond to a command within the timeout period. This occurs because this command takes more time than the default timeout of 1 sec. The timeout can be changed in the Galil software, and this warning does not affect the operation of the IOC board or software.

CB

FUNCTION: Clear Bit

DESCRIPTION:

The CB command clears a particular bit on the output module, setting the output to logic 0. The CB and SB (Set Bit) instructions can be used to control the state of output lines. The CB command can be used in the #PLCSCAN thread, but only for local outputs in this case.

ARGUMENTS: CB n where

n is an integer corresponding to a specific output on the IOC-7007 to be cleared (set to 0). See Chapter 4 for the numbering scheme for the different IOC slots.

Note: When using Modbus devices, the I/O points of the Modbus devices are calculated using the following formula:

n = (SlaveAddress*10000) + (HandleNum*1000) + ((Module-1)*4) + (Bitnum-1)

Slave Address is used when the ModBus device has slave devices connected to it and specified as Addresses 0 to 255. Please note that the use of slave devices for Modbus is very rare and this number will usually be 0.

HandleNum is the handle specifier, labeled with letters from A to F (1 to 6).

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.

USAGE:

In a Program	Yes
Command Line	Yes
Can be Interrogated	No
Used as an Operand	No

RELATED COMMANDS:

"SB"	Set Bit

EXAMPLES:

CB 0	Clear output bit 0
CB 1	Clear output bit 1
CB 2	Clear output bit 2

Note: When communicating between two IOC modules using SB/CB commands – Port 502 must be used when opening the handle (IH).

CF

FUNCTION: Configure

DESCRIPTION:

Sets the default port for unsolicited messages. By default, the IOC-7007 will send unsolicited responses to whichever communication channel is currently being used (the WH command provides the communication channel). The CF command allows the user to override this setting and send unsolicited responses to the Serial port or Handles A – F.

ARGUMENTS: CF n where

n is A thru F for Ethernet handles 1 thru 6 or S for the serial port.

USAGE:

In a Program	Yes
Command Line	No
Can be Interrogated	No
Used as an Operand	Yes

OPERAND USAGE:

_CF will return the current port selected for unsolicited responses from the IOC board. The _CF operand will return a decimal value.

EXAMPLES:

CFA	Select Ethernet handle A to return unsolicited responses.
MG_CF	Interrogate configuration
:65.000	Response from _CF showing handle A as default port.

CP

FUNCTION: Compile PLC Thread

DESCRIPTION:

The CP command compiles the #PLCSCAN thread prior to execution. The CP command can be issued in one of the eight simultaneous threads or from the command line. If the CP is not issued before issuing the XP, then the XP will compile the #PLCSCAN thread prior to executing it. However, the XP will take much more time to process if the program is not compiled first.

ARGUMENTS: CP

USAGE:

In a Program	Yes
Command Line	Yes
Can be Interrogated	No
Used as an Operand	Yes

OPERAND USAGE:

_CP will return a 0 if the program is not compiled or a 1 if it has compiled successfully.

RELATED COMMANDS:

"XP"	Execute PLC thread
“CP”	Compile PLC thread
“TX”	Tell PLC execution time

EXAMPLES:

CP	Compiles the PLC thread

CW

FUNCTION: Copyright information / Data Adjustment bit on/off

DESCRIPTION:

The CW command has a dual usage. The CW command will return the copyright information when the argument, n is 0. Otherwise, the CW command is used as a communications enhancement for use by the Galil terminal software programs. When turned on, the communication enhancement from the command sets the MSB of unsolicited, returned ASCII characters to 1. Unsolicited ASCII characters are those characters that are returned from the IOC board without being directly queried from the terminal. This is the case when a program has a command that requires the IOC-7007 to return a value or string.

ARGUMENTS: CW n,m where

n is a number, either 0,1 or 2:

0 or ?	Causes the IOC to return the copyright information
1	Causes the IOC to set the MSB of unsolicited returned characters to 1
2	Causes the IOC to not set the MSB of unsolicited characters.

m is 0 or 1 (optional)

0	Causes the IOC to pause program execution when hardware handshaking disables character transmissions.
1	Causes the IOC to continue program execution when hardware handshake disables character transmissions - output characters will be lost.

USAGE:

In a Program	Yes
Command Line	Yes
Can be Interrogated	Yes
Used as an Operand	Yes

OPERAND USAGE:

_CW contains the value of the data adjustment bit. 1 =on, 2 = off

Note: *The CW command can cause garbled characters to be returned by the IOC board. The default state of the IOC board is to disable the CW command. However, the terminal software may sometimes enable the CW command for internal usage. If the IOC board is reset while the Galil software is running, the CW command could be reset to the default value, which would create difficulty for the software. It may be necessary to re-enable the CW command. The CW command status can be stored in EEPROM.*

DA

FUNCTION: Deallocate the Variables & Arrays

DESCRIPTION:

The DA command frees the array and/or variable memory space. In this command, more than one array or variable can be specified for memory de-allocation. Different arrays and variables are separated by comma when specified in one command. The * argument deallocates all the variables, and *[0] deallocates all the arrays.

ARGUMENTS: DA c[0],d,etc.   where

c[0] - Defined array name

d - Defined variable name

* - Deallocates all the variables

*[0] - Deallocates all the arrays

DA? Returns the number of arrays available on the IOC-7007.

USAGE:

In a Program	Yes
Command Line	Yes
Can be Interrogated	No
Used as an Operand	Yes

OPERAND USAGE:

_DA contains the total number of arrays available. For example, on an IOC-7007, before any arrays have been defined, the operand _DA is 14. If one array is defined, the operand _DA will return 13.

RELATED COMMANDS:

"DM"	Dimension Array

EXAMPLES:

'Cars' and 'Salesmen' are arrays, and 'Total' is a variable.

DM Cars[400],Salesmen[50]	Dimension 2 arrays
Total=70	Assign 70 to the variable Total
DA Cars[0],Salesmen[0],Total	Deallocate the 2 arrays & variables
DA*[0]	Deallocate all arrays
DA *,*[0]	Deallocate all variables and all arrays

NOTE: *Since this command deallocates the spaces and compacts the array spaces in the memory, it is possible that execution of this command may take longer time than 2 ms.*

DL

FUNCTION: Download

DESCRIPTION:

The DL command transfers a data file from the host computer to the IOC-7007. If this command is typed on the command input line of the Galil Terminal, another window will come up asking the user to specify a file to download. If the DL command is used in a terminal utility such as Hyper Terminal, a file will be accepted as a datastream without line numbers. In this case, the file is terminated using <control> Z, <control> Q, <control> D, or \. DO NOT insert spaces before each command.

If no parameter is specified, downloading a data file will clear any programs in the IOC board's RAM. The data is entered beginning at line 0. If there are too many lines or too many characters per line, the IOC-7007 will return a "?". To download a program after a label, specify the label name following DL. The # argument may be used with DL to append a file at the end of the IOC-7007 program in RAM.

ARGUMENTS: DL n

n = no argument	Downloads program beginning at line 0. Erases programs in RAM.
n = #Label	Begins download at line following #Label where label may be any valid program label.
n = #	Begins download at end of program in RAM.

USAGE:

In a Program	No
Command Line	Yes
Can be Interrogated	No
Used as an Operand	Yes

OPERAND USAGE:

When used as an operand, _DL gives the number of available labels. The total number of labels is 126.

RELATED COMMANDS:

"UL"	Upload

EXAMPLES:

```
DL                  Begin Download
SB0                 Data
CB2                 Data
EN                  Data
<control>Z          End download
```

DM

FUNCTION: Dimension

DESCRIPTION:

The DM command defines a single dimensional array with a name and n total elements. The first element of the defined array starts with element number 0 and the last element is at n-1.

ARGUMENTS: DM c[n] where

c is a array name of up to eight alphanumeric characters, starting with an uppercase alphabetic character. n is the number of entries or array elements from 1 to 2000.

DM? Returns the number of array elements available.

USAGE:

In a Program	Yes
Command Line	Yes
Can be Interrogated	Yes
Used as an Operand	Yes

OPERAND USAGE:

_DM contains the available array space. On the IOC-7007, before any arrays have been defined, the operand _DM will return 2000. If an array of 100 elements is defined, the operand _DM will return 1900.

RELATED COMMANDS:

"DA"	Deallocate Array

EXAMPLES:

DM Pets[5],Dogs[2],Cats[3]	Define dimension of arrays, pets with 5 elements; Dogs with 2 elements; Cats with 3 elements
	DM Tests [1000] Define dimension of array Tests with 1000 elements.

DR

FUNCTION: Configures Axes and I/O Data Record Update Rate

DESCRIPTION:

The controller creates a QR record and sends it periodically to a UDP Ethernet Handle

ARGUMENTS: DR n, m

n specifies the data update rate in samples between updates. When TM is set to the default of 1000, n specifies the data update rate in milliseconds. n=0 to turn it off, or n must be an integer of at least 8.

m specifies the Ethernet handle on which to periodically send the Data Record. 0 is handle A, 1 is B… 7 is H. The handle must be UDP (not TCP).

USAGE: **DEFAULTS:**

While Moving	Yes	Default Value	DR0 (off)
In a Program	Yes	Default Format	--
Command Line	Yes		
Controller Usage	**EXCEPT FOR DMC-2000**		

OPERAND USAGE:

_DR contains the data record update rate.

RELATED COMMANDS:

QZ	Sets format of data
QR	Query a single data record

EXAMPLES:

```
:DR1000,0

:G   x          ~      P

_   `          @~      P

_   H           `~      P

_   0          ~      P

DR0
```

Note: *The data record is in a binary, non-printable format (the output above is normal)*

ED

FUNCTION: Edit

DESCRIPTION:

Using Galil DOS Teminal Software: The ED command puts the IOC board into the Edit subsystem. In the Edit subsystem, programs can be created, changed or destroyed. The commands in the Edit subsystem are:

<cntrl>D	Deletes a line
<cntrl>I	Inserts a line before the current one
<cntrl>P	Displays the previous line
<cntrl>Q	Exits the Edit subsystem
<return>	Saves a line

Using Galil Windows Terminal Software: The ED command causes the Windows terminal software to open the terminal editor.

ARGUMENTS: ED n where

n specifies the line number to begin editing. The default line number is the last line of program space with commands.

USAGE:

In a Program	No
Command Line	Yes
Can be Interrogated	No
Used as an Operand	Yes

OPERAND USAGE:

_ED contains the line number of the last line to have an error

EXAMPLES:

```
000 #BEGIN
001 OQ0,1
002 SB3
003 SLKJ                          Bad line
004 EN
005 #CMDERR                       Routine which occurs upon a command error
006 V=_ED
007 MG "An error has occurred" {n}
008 MG "In line", V{F3.0}
009 AB
010 ZS0
011 EN
```

HINT: *Remember to quit the Edit Mode prior to executing or listing a program.*

ELSE

FUNCTION: Else function for use with IF conditional statement

DESCRIPTION:

The ELSE command is an optional part of an IF conditional statement. The ELSE command must occur after an IF command and it has no arguments. It allows for the execution of a command only when the argument of the IF command evaluates False. If the argument of the IF command evaluates false, the IOC-7007 will skip commands until the ELSE command. If the argument for the IF command evaluates true, the IOC board will execute the commands between the IF and ELSE command.

The ELSE command can be used in the #PLCSCAN thread.

ARGUMENTS: ELSE

USAGE:

In a Program	Yes
Command Line	No
Can be Interrogated	No
Used as an Operand	No

RELATED COMMANDS:

"ENDIF"	End of IF conditional Statement

EXAMPLES:

IF (@IN[1]=0)	IF conditional statement based on input 1
IF (@IN[2]=0)	2nd IF conditional statement executed if 1st IF conditional true
MG "INPUT 1 AND INPUT 2 ARE ACTIVE"	Message to be executed if 2nd IF conditional is true
ELSE	ELSE command for 2nd IF conditional statement
MG "ONLY INPUT 1 IS ACTIVE	Message to be executed if 2nd IF conditional is false
ENDIF	End of 2nd conditional statement
ELSE	ELSE command for 1st IF conditional statement
MG"ONLY INPUT 2 IS ACTIVE"	Message to be executed if 1st IF conditional statement
ENDIF	End of 1st conditional statement

EN

FUNCTION: End

DESCRIPTION:

The EN command is used to designate the end of a program or subroutine. If a subroutine was called by the JS command, the EN command ends the subroutine and returns program flow to the point just after the JS command. No argument is required.

The EN command is also used to end the automatic subroutine, #CMDERR, and an argument is required to handle trippoints in the main thread upon returning from the subroutine.

The EN command can be used in the #PLCSCAN thread.

ARGUMENTS: EN m

m=0	Return from #CMDERR without restoring trippoint in main thread
m=1	Return from #CMDERR and restore trippoint in main thread

Note 1: The default value for the argument is 0. For example EN0 and EN have the same effect.

Note 2: Trippoints cause a program to wait for a particular event. The AI command, for example, waits for specific inputs to go high or low. If the #CMDERR subroutine is executed due to an invalid command while the program is waiting for a trippoint, the #CMDERR can end by continuing to wait for the trippoint as if nothing happened, or clear the trippoint and continue executing the program at the command just after the trippoint. The EN arguments will specify how the #CMDERR routine handles trippoints.

Note 3: Use the RI command to return from the #ININT subroutine.

Note 4: Trippoints occurring in other threads other than the main thread will not be affected by the #CMDERR subroutine.

USAGE:

In a Program	Yes
Command Line	No
Can be Interrogated	No
Used as an Operand	No

RELATED COMMANDS:

"RI"	Return from interrupt subroutine

EXAMPLES:

```
#A              Program A
SB1             Set output 1 high
WT500           Wait for 500 msec
CB1             Set output 1 low
MG "DONE"       Print message
EN              End of Program
```

***Note:** Use the RI command to end the input interrupt (ININT) subroutine.*

ENDIF

FUNCTION: End of IF conditional statement

DESCRIPTION:

The ENDIF command is used to designate the end of an IF conditional statement. An IF conditional statement is formed by the combination of an IF and ENDIF command. An ENDIF command must always be executed for every IF command that has been executed. It is recommended that the user not include jump commands inside IF conditional statements, since this causes re-direction of command execution. In this case, the command interpreter may not execute an ENDIF command.

The ENDIF command can be used in the #PLCSCAN thread.

ARGUMENTS: ENDIF

USAGE:

In a Program	Yes
Command Line	No
Can be Interrogated	No
Used as an Operand	No

RELATED COMMANDS:

"ELSE"	Optional command to be used only after IF command
"JP”	Jump command
"JS"	Jump to subroutine command

EXAMPLES:

IF (@IN[1]=0)	IF conditional statement based on input 1
MG “INPUT 1 IS ACTIVE”	Message to be executed if “IF” conditional is true
ENDIF	End of conditional statement

EO

FUNCTION: Echo

DESCRIPTION:

The EO command turns the echo on or off. If the echo is off, characters input over the bus will not be echoed back.

ARGUMENTS: EO n where

n=0 or 1. 0 turns echo off, 1 turns echo on.

USAGE:

In a Program	Yes
Command Line	Yes
Can be Interrogated	No
Used as an Operand	Yes

OPERAND USAGE:

_EO contains the state of the echo; 0 is off, 1 is on

EXAMPLES:

EO 0	Turns echo off
EO 1	Turns echo on

HP

FUNCTION: Halts the #PLCSCAN thread

DESCRIPTION:

The HP command halts execution of the #PLCSCAN thread. This command can be issued from one of the eight simultaneous threads or the command line.

ARGUMENTS: HP

USAGE:

In a Program	Yes
Command Line	Yes
Can be Interrogated	No
Used as an Operand	No

RELATED COMMANDS:

“CP”	Compile PLC thread
“XP”	Execute PLC thread
“TX”	Tell PLC execution time

EXAMPLES:

HP	Halts the #PLCSCAN thread

HS

FUNCTION: Handle Assignment Switch

DESCRIPTION:

The HS command is used to switch the handle assignments between two handles. Handles are assigned by the IOC when the handles are opened with the HC command, or are assigned explicitly with the IH command. Should those assignments need modifications, the HS command allows the handles to be reassigned.

ARGUMENTS: HSh=i where

h is the first handle of the switch (A through F, S)

i is the second handle of the switch (A through F, S)

S is used to represent the current handle executing the command

USAGE:

In a Program	Yes
Command Line	Yes
Can be Interrogated	No
Used as an Operand	No

RELATED COMMANDS:

"IH"	Internet Handle

EXAMPLES:

HSC=D	Connection for handle C is assigned to handle D. Connection for handle D is assigned to handle C.
HSS=E	Executing handle connection is assigned to handle E. Connection for handle E is assigned to executing handle.

HX

FUNCTION: Halt Execution

DESCRIPTION:

The HX command halts the execution of any of the 8 programs that may be running independently in multitasking. The parameter n specifies the program to be halted.

HX can be used in the #PLCSCAN thread.

ARGUMENTS: HX n where

n is 0 to 7 to indicate the eight threads

USAGE:

In a Program	Yes
Command Line	Yes
Can be Interrogated	No
Used as an Operand	Yes

OPERAND USAGE:

When used as an operand, _HX n contains the running status of thread n with:

0	Thread not running
1	Thread is running
2	Thread has paused at trippoint

RELATED COMMANDS:

"XQ"	Execute program

EXAMPLES:

XQ #A	Execute program #A, thread zero
XQ #B,1	Execute program #B, thread one
HX0	Halt thread zero
HX1	Halt thread one

IA

FUNCTION: IP Address

DESCRIPTION:

The IA command assigns the IOC-7007 with an IP address.

The IA command may also be used to specify the time out value. This is only applicable when using the TCP/IP protocol.

The IA command can only be used via RS-232. Since it assigns an IP address to the IOC board, communication with the IOC via Ethernet cannot be accomplished until after the address has been assigned.

ARGUMENTS: IA ip0,ip1,ip2, ip3 **or** IA n **or** IA<t where

ip0, ip1, ip2, ip3 are 1 byte numbers separated by commas and represent the individual fields of the IP address.

n is the IP address for the IOC-7007, which is specified as an integer representing the signed 32 bit number (two's complement).

<t specifies the time in update samples between TCP retries.

>u specifies the multicast IP address where u is an integer between 0 and 63.

IA? will return the IP address of the IOC board.

USAGE:

In a Program	No
Command Line	Yes
Can be Interrogated	Yes
Used as an Operand	Yes

OPERAND USAGE:

_IA0	contains the IP address representing a 32 bit signed number (Two's complement)
_IA1	contains the value for t (retry time)
_IA2	contains the number of available handles
_IA3	contains the number of the handle using this operand where the number is 0 to 5. 0 represents handle A, 1 for handle B, etc.
_IA4	contains the handle that lost communication last. Contains –1 on reset to indicate no handles lost.
_IA5	contains the Ethernet speed, 10 or 100 Mbits/sec

RELATED COMMANDS:

"IH"	Internet Handle

EXAMPLES:

IA 151, 12, 53, 89	Assigns the IOC-7007 with the address 151.12.53.89
IA 2534159705	Assigns the IOC-7007 with the address 151.12.53.89
IA < 500	Sets the timeout value to 500msec

IF

FUNCTION: IF conditional statement

DESCRIPTION:

The IF command is used in conjunction with an ENDIF command to form an IF conditional statement. The arguments are one or more conditional statements. If the conditional statement(s) evaluates true, the command interpreter will continue executing commands which follow the IF command. If the conditional statement evaluates false, the IOC-7007 will ignore commands until the associated ENDIF command OR an ELSE command occurs in the program.

The IF command can be used in the #PLCSCAN thread.

ARGUMENTS: IF condition where

Conditions are tested with the following logical operators:

< less than or equal to

> greater than

= equal to

<= less than or equal to

>= greater than or equal to

<> not equal

USAGE:

In a Program	Yes
Command Line	No
Can be Interrogated	No
Used as an Operand	No

RELATED COMMANDS:

"ELSE"	Optional command to be used only after IF command
"ENDIF"	End of IF conditional Statement

EXAMPLES:

IF (@IN[1]=0)	IF conditional statement based on input 1
MG "Input 1 is Low"	Message to be executed if “IF” statement is true
ENDIF	End of IF conditional statement
IF(V1>25)&(_OQ1=3)	Conditions based on V1 variable and Slot 1 I/O status
MG “Conditions met”	Message to be executed if “IF” statement is true
ENDIF	End of IF statement

IH

FUNCTION: Open Internet Handle

DESCRIPTION:

The IH command is used when the IOC-7007 is operated as a master (also known as a client). This command opens a handle and connects to a slave (server).

Each IOC board may have 6 handles open at any given time. They are designated by the letters A through F. To open a handle, the user must specify:

1. The IP address of the slave
2. The type of session: TCP/IP or UDP/IP
3. The port number of the slave. This number is not necessary if the slave device does not require a specific port value. If not specified, the board will specify the port value as 1000.

ARGUMENTS: IHh= ip0,ip1,ip2,ip3 <p >q **or** IHh=n <p >q **or** IHh= >r where

h is the handle, specified as A,B,C,D,E or F

ip0,ip1,ip2,ip3 are integers between 0 and 255 and represent the individual fields of the IP address. These values must be separated by commas.

n is a signed integer between - 2147483648 and 2147483647. This value is the 32 bit IP address and can be used instead of specifying the 4 address fields.

IHS => C closes the handle that sent the command; where C=-1 for UDP/IP, or C=-2 for TCP/IP.

IHN => C closes all handles except for the one sending the command; where C=-1 UDP, or C=-2 TCP.

<p specifies the port number of the slave where p is an integer between 0 and 65535. This value is not required for opening a handle.

>q specifies the connection type where q is 0 for no connection, 1 for UDP and 2 for TCP

>r specifies that the connection be terminated and the handle be freed, where r is -1 for UDP, -2 for TCP/IP or –3 for TCP/IP reset

"?" returns the IP address as 4 1-byte numbers

OPERAND USAGE:

_IHh0	contains the IP address as a 32 bit number
_IHh1	contains the slave port number
_IHh2	contains a 0 if the handle is free
	contains a 1 if it is for a UDP slave
	contains a 2 if it is for a TCP slave
	contains a -1 if it is for a UDP master
	contains a -2 if it is for a TCP master
	contains a –5 while establishing a UDP handle
	contains a –6 while establishing a TCP handle

_IHh3	contains a 0 if the ARP was successful contains a 1 if it has failed or is still in progress.

USAGE:

In a Program	Yes
Command Line	Yes
Can be Interrogated	Yes
Used as an Operand	Yes

RELATED COMMANDS:

"IA"	Internet Address

EXAMPLES:

IHA=251,29,51,1	Open handle A at IP address 251.29.51.1
IHA= -2095238399	Open handle A at IP address 251.29.51.1

Note: When the IH command is given, the IOC-7007 initializes an ARP on the slave device before opening a handle. This operation can cause a small time delay before the IOC board responds.

II

FUNCTION: Input Interrupt

DESCRIPTION:

This newly formatted II command enables the interrupt function for the specified inputs. This function can trigger one of eight input interrupt subroutines (#ININT) when the IOC-7007 sees that the conditional statement is satisfied.

When the condition is satisfied, the program will jump to the subroutine with label #ININTn, where n ranges from 0 to 7. This subroutine will be executed in thread m, where m is between 0 (main) and 7, causing any trippoint set in that thread to be cleared. But this trippoint can be re-enabled by the proper termination of the interrupt subroutine using RI. The RI command is used to return from the #ININTn routines.

To avoid returning to the program on an interrupt, use the command ZS to zero the subroutine stack and use the II command to reset the interrupt.

***Note**: An application program must be running on the controller for the interrupt function to work.*

ARGUMENTS: II n,m,condition where

n is an integer between 0 and 7 decimal. This number specifies the #ININTn subroutine to be executed when the interrupt occurs.

m is an integer between 0 and 7. This argument indicates the thread number in which the #ININTn subroutine is going to be executed. The specified thread needs to be running when the interrupt occurs, otherwise the #ININTn subroutine will not be executed. Upon interrupt, the existing thread m will be interrupted to allow the execution of the interrupt subroutine. Upon completion of the interrupt, the main program in thread m will once again be enabled from the point at which the interrupt occurred.

Condition can consist of any number of inputs, using the “&” operator between each input. A positive input number means the condition is for that input to go high, and a negative input number for that input to go low.

USAGE:

In a Program	Yes
Command Line	No
Can be Interrogated	No
Used as an Operand	No

RELATED COMMANDS:

"RI"	Return from Interrupt
#ININTn	Interrupt Subroutine
"AI"	Trippoint for input

EXAMPLES:

#A	Program A
II 2,0,3&5&-10	Specify interrupt #2 on main thread when inputs 3 and 5 go high, and 10 goes low
#LOOP;JP #LOOP	Loop
EN	End Program
#ININT2	Interrupt subroutine number 2
MG "INTERRUPT"	Print Message
#CLEAR;JP#CLEAR,@IN[1]=0	Check for ‘reset’ input 1 to clear interrupt
RI	Return to main program

IN

FUNCTION: Input Variable

DESCRIPTION:

The IN command allows a variable to be input from a keyboard. When the IN command is executed in a program, the prompt message is displayed. The operator then enters the variable value followed by a carriage return. The entered value is assigned to the specified variable name.

The IN command holds up execution of following commands in a program until a carriage return or semicolon is detected. If no value is given prior to a semicolon or carriage return, the previous variable value is kept. Input Interrupts will still be active.

ARGUMENTS: IN ”m” , n {So} where

"m" is the prompt message. May be letters, numbers, or symbols up to maximum line length and must be placed in quotations. Make sure that maximum line length of 80 characters is not exceeded.

n is the name of variable to hold value returned from input.

{So} specifies string data where o is the number of characters from 1 to 6. Not required if the data is numerical.

Note 1: Do not leave a space between the comma and n.

Note2: The IN command cannot be used over Ethernet, only RS-232.

Note 3: IN command can only be used in thread 0.

USAGE:

In a Program	Yes
Command Line	No
Can be Interrogated	No
Used as an Operand	No

EXAMPLES: Operator specifies a bit that is to be turned on.

```
#A                                           Program A
 IN "Enter output number  (0-7) to turn on",N1   Prompt operator for bit to set high
SBN1                                         Set the specified bit high
MG "DONE"                                    Print Message
EN                                           End Program
```

JP

FUNCTION: Jump to Program Location

DESCRIPTION:

The JP command causes a jump to a program location on a specified condition. The program location may be any program line number or label. The condition is a conditional statement which uses a logical operator such as equal to or less than. A jump is taken if the specified condition is true.

Multiple conditions can be used in a single jump statement. The conditional statements are combined in pairs using the operands "&" and "|". The "&" operand between any two conditions requires that both statements must be true for the combined statement to be true. The "|" operand between any two conditions requires that only one statement be true for the combined statement to be true.

Note 1: Each condition must be placed in parenthesis for proper evaluation by the IOC-7007.

Note 2: The JP command can be used in the #PLCSCAN thread.

ARGUMENTS: JP location,condition where

location is a program line number or label

condition is an optional conditional statement using a logical operator

The logical operators are:

< less than

> greater than

= equal to

<= less than or equal to

>= greater than or equal to

<> not equal to

USAGE:

In a Program	Yes
Command Line	No
Can be Interrogated	No
Used as an Operand	No

EXAMPLES:

JP #POS1,V1<5	Jump to label #POS1 if variable V1 is less than 5
JP #A,@IN[1]=0	Jump to #A if input 1 is low
JP #B	Jump to #B (no condition)

HINT: *JP is similar to an IF, THEN command. Text to the right of the comma is the condition that must be met for a jump to occur. The destination is the specified label before the comma.*

JS

FUNCTION: Jump to Subroutine

DESCRIPTION:

The JS command will change the sequential order of execution of commands in a program. If the jump is taken, program execution will continue at the line specified by the destination parameter, which can be either a line number or label. The line number of the JS command is saved and after the next EN command is encountered (End of subroutine), program execution will continue with the instruction following the JS command. There can be a JS command within a subroutine.

Multiple conditions can be used in a single jump subroutine statement. The conditional statements are combined in pairs using the operands “&” and “|”. The “&” operand between any two conditions requires that both statements must be true for the combined statement to be true. The ”|” operand between any two conditions requires that only one statement be true for the combined statement to be true.

Note: Each condition must be placed in parenthesis for proper evaluation by the IOC-7007.

Note: Subroutines can be nested 16 deep in the IOC-7007.

ARGUMENTS: JS destination,condition   where

destination is a line number or label

condition is an optional conditional statement using a logical operator

The logical operators are:

< less than or equal to

> greater than

= equal to

<= less than or equal to

>= greater than or equal to

<> not equal

USAGE:

In a Program	Yes
Command Line	No
Can be Interrogated	No
Used as an Operand	No

RELATED COMMANDS:

"EN"	End

EXAMPLES:

JS #B,N1<5	Jump to subroutine #B if N1 is less than 5
JS #LOOP,N1<>0	Jump to #LOOP if N1 is not equal to 0
JS #A	Jump to subroutine #A (no condition)

LA

FUNCTION: List Arrays

DESCRIPTION:

The LA command returns a list of all arrays in memory. The listing will be in alphabetical order. The size of each array will be included next to each array name in square brackets.

ARGUMENTS: None

USAGE:

In a Program	Yes
Command Line	Yes
Can be Interrogated	Yes
Used as an Operand	No

RELATED COMMANDS:

"LL"	List Labels
"LS"	List Program
"LV"	List Variable

EXAMPLES:

: LA	Interrogation Command
CA [10]	IOC board returns a list of 4 Arrays
LA [5]	
NY [25]	
VA [17]	

LL

FUNCTION: List Labels

DESCRIPTION:

The LL command returns a listing of all of the program labels in memory. The listing will be in alphabetical order.

ARGUMENTS: None

USAGE:

In a Program	Yes
Command Line	Yes
Can be Interrogated	Yes
Used as an Operand	No

RELATED COMMANDS:

"LV"	List Variables

EXAMPLES:

: LL	Interrogation Command
# FIVE	IOC board returns a list of 5 labels
# FOUR	
# ONE	
# THREE	
# TWO	

LR

FUNCTION: Launch Slave Data Record

DESCRIPTION:

The LR command causes the IOC-7007 to launch a slave data record to the DMC-3425 master controller in a distributed network. This record includes a 2-byte header information and blocks of general information, such as the board's I/O status and error code. The QW command needs to be set in the master controller for the LR command to function.

ARGUMENTS: LR

USAGE:

In a Program	Yes
Command Line	Yes
Can be Interrogated	No
Used as an Operand	No

RELATED COMMANDS:

"QR"	I/O Data Record
"ZC"	Set first user variable
"ZD"	Set second user variable

Note: The data from the LR command is sent to a master controller. This information can't be viewed by the user with the Galil Terminal.

LS

FUNCTION: List

DESCRIPTION:

The LS command sends a listing of the program memory. The listing will start with the line pointed to by the first parameter, which can be either a line number or a label. If no parameter is specified, it will start with line 0. The listing will end with the line pointed to by the second parameter--again either a line number or label. If no parameter is specified, the listing will go to the last line of the program.

ARGUMENTS: LS n,m where

n,m are valid numbers from 0 to 499, or labels. n is the first line to be listed, m is the last.

USAGE:

In a Program	No
Command Line	Yes
Can be Interrogated	No
Used as an Operand	No

EXAMPLES:

```
:LS #A,6                List program starting at #A through line 6
002 #A
003 MG “Program A”
004 COUNT=1
005 OQ2,15
006 WT 2000
```

HINT: Remember to quit the Edit Mode <cntrl> Q prior to giving the LS command.

LV

FUNCTION: List Variables

DESCRIPTION:

The LV command returns a listing of all of the program variables in memory. The listing will be in alphabetical order.

ARGUMENTS: None

USAGE:

In a Program	Yes
Command Line	Yes
Can be Interrogated	Yes
Used as an Operand	No

RELATED COMMANDS:

"LL"	List Labels

EXAMPLES:

```
: LV                 Interrogation Command
APPLE  = 60.0000     IOC board returns a list of 3 variables
BOY    = 25.0000
ZEBRA = 37.0000
```

LZ

FUNCTION: Inhibit leading zeros

DESCRIPTION:

The LZ command is used for formatting the values returned from interrogation commands or interrogation of variables and arrays. By enabling the LZ function, all leading zeros of returned values will be removed.

ARGUMENTS: LZ n where n is

1 to remove leading zeros

0 to disabled the leading zero removal

LZ? Returns the state of the LZ function.

Note: Default value is 1.

USAGE:

In a Program	Yes
Command Line	Yes
Can be Interrogated	Yes
Used as an Operand	Yes

OPERAND USAGE:

_LZ contains the state of the LZ command

EXAMPLES:

```
TB                      Tell status
001
LZ 1                    Inhibit leading zeros
TB                      Tell status
1
```

MB

FUNCTION: Modbus

DESCRIPTION:

The MB command is used to communicate with I/O devices using the first two levels of the Modbus protocol.

The format of the command varies depending on each function code. The function code, -1, designates that the first level of Modbus is used (creates raw packets and receives raw data). The other codes are the 10 major function codes of the second level that the IOC board supports.

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

Note: For those command formats that have “addr”, this is the slave address. The slave address is usually zero.

Note: All the formats contain an h parameter. This designates the connection handle number (A thru F).

ARGUMENTS:

MBh = -1, len, array[]   where

len is the number of the bytes

Array[] is the name of array containing data

MBh = addr, 1, m, n, array[]   where

m is the starting bit number

n is the number of bits

array[] of which the first element will hold result

MBh = addr, 2, m, n, array[]   where

m is the starting bit number

n is the number of bits

array[] of which the first element will hold result

MBh = addr, 3, m, n, array[]   where

m is the starting register number

n is the number of registers

array[] will hold the response

MBh = addr, 4, m, n, array[]   where m is the starting register number

n is the number of registers

array[] will hold the response

MBh = addr, 5, m, n where

m is the starting bit number

n is 0 or 1 and represents the coil set to off or on.

MBh = addr, 6, m, n where

m is the register number

n is the 16 bit value

MBh = addr, 7, array[] where

array[] is where the returned data is stored (one byte per element)

MBh = addr, 15, m, n, array[] where

m is the starting bit number

n is the number of bits

array[] contains the data (one byte per element)

MBh = addr, 16, m, n, array[] where

m is the starting register number

n is the number of registers

array[] contains the data (one 16 bit word per element)

MBh = addr, 17, array[] where

array[] is where the returned data is stored

USAGE:

In a Program	Yes
Command Line	Yes

MG

FUNCTION: Message

DESCRIPTION:

The MG command transmits data from the IOC board. This can be used to alert an operator, send instructions or return a variable value.

ARGUMENTS: MG {Ex}, "m", {^n}, V {Fm.n or $m,n} {N} {Sn}

{Ex}for ethernet and ‘x’ specifies the ethernet handle (A,B,C,D,E or F) for sending the message through. {Ex} can be replaced by {P1} to send the message through the serial port. These settings are only needed when the user wants to specify another return port than the one set by the CF command.

"m" is a text message including letters, numbers, symbols or <ctrl>G. Make sure that maximum line length of 80 characters is not exceeded.

{^n} is an ASCII character specified by the value n

V is a variable name or array element, where the following specifiers can be used for formatting:

{Fm.n} Display variable in decimal format with m digits to left of decimal, and n to the right.

{$m,n} Display variable in hexadecimal format with m digits to left of decimal, and n to the right.

{N} Suppress carriage return line feed.

{Sn} Display variable as a string of length n where n is 1 thru 6

Note: Multiple text, variables, and ASCII characters may be used, each must be separated by a comma.

Note: The order of arguments is not important.

USAGE:

In a Program	Yes
Command Line	Yes
Can be Interrogated	No
Used as an Operand	No

EXAMPLES:

Case 1: Message command displays ASCII strings

MG "Good Morning" Displays the string

Case 2: Message command displays variables or arrays

MG "The Answer is", Total {F4.2} Displays the string with the content of variable TOTAL in local format of 4 digits before and 2 digits after the decimal point.

Case 3: Message command sends any ASCII characters to the port.

MG {^13}, {^30}, {^37}, {N} Sends carriage return, characters 0 and 7 followed by no carriage return line feed command to the port.

MW

FUNCTION: Modbus Wait

DESCRIPTION:

Enabling the MW command causes the controller to hold up execution of the program after sending a Modbus command until a response from the Modbus device has been received. If the response is never received, then the #TCPERR subroutine will be triggered and an error code of 123 will occur on _TC.

ARGUMENTS: MWn where

n = 0	Disables the Modbus Wait function
n = 1	Enables the Modbus Wait function

USAGE: **DEFAULTS:**

While Moving	Yes	Default Value	0
In a Program	Yes	Default Format	1.0
Command Line	Yes		
Controller Usage	**ALL CONTROLLERS**		

OPERAND USAGE:

MW? contains the state of the Modbus Wait.

RELATED COMMANDS:

"MB"	Modbus

EXAMPLES:

MW1	Enables Modbus Wait
SB1001	Set Bit 1 on Modbus Handle A
CB1001	Clear Bit 1 on Modbus Handle A

Hint: The MW command ensures that the command that was sent to the Modbus device was successfully received before continuing program execution. This prevents the controller from sending multiple commands to the same Modbus device before it has a chance to execute them

NO (‘ apostrophe also accepted)

FUNCTION: No Operation

DESCRIPTION:

The NO command performs no action in a sequence, but can be used as a comment in a program. After the NO, characters can be given to form a program comment up to the maximum line length of the IOC-7007, which is 80. This helps to document a program.

An apostrophe ‘ may also be used instead of the NO to document a program.

The NO command can be used in the #PLCSCAN thread.

ARGUMENTS: NO m where

m is any group of letters and numbers

USAGE:

In a Program	Yes
Command Line	Yes
Can be Interrogated	No
Used as an Operand	No

EXAMPLES:

```
#A                   Program A
NO                   No Operation
NO This Program      No Operation
‘ Does Absolutely    No Operation
‘ Nothing            No Operation
EN                   End of Program
```

OB

FUNCTION: Output Bit

DESCRIPTION:

The OB n, logical expression command defines output bit n = 0 through 223 as either 0 or 1 depending on the result from the logical expression. Any non-zero value of the expression results in a one on the output. The OB command can be used in the #PLCSCAN thread.

ARGUMENTS: OB n, expression where

n is 0 through 223, denoting output bit

expression is any valid logical expression, variable or array element.

USAGE:

In a Program	Yes
Command Line	Yes
Can be Interrogated	No
Used as an Operand	No

RELATED COMMAND:

"CB"	Clear Bit
"SB"	Set Bit

EXAMPLES:

OB 1, POS=1	If POS 1 is non-zero, Bit 1 is high. If POS 1 is zero, Bit 1 is low
OB 2, @IN[5]&@IN[6]	If Input 5 and Input 6 are both high, then Output 2 is set high
OB 3, COUNT[1]	If the element 1 in the array is zero, clear bit 3, otherwise set bit 3
OB N, COUNT[1]	If element 1 in the array is zero, clear bit N

OQ

FUNCTION: Output Module

DESCRIPTION:

The OQ command sends data to the output modules of the IOC-7007. You can use the output module to control external switches and relays. The OQ command can be used in the #PLCSCAN thread.

ARGUMENTS: OQ m,n where

m is the slot number from 0 thru 6

n is the decimal representation of the output modules' bit value. If an eight-bit output module is used, the decimal representation will be from 0 to 255. Likewise, if a 16 bit module is used, the decimal representation will be from 0 and 65,535.

USAGE:

In a Program	Yes
Command Line	Yes
Can be Interrogated	No
Used as an Operand	Yes

OPERAND USAGE:

_OQm, contains the output bit status of slot m.

EXAMPLES:

OQ0,3	Set the first 2 outputs (0 and 1) on slot 0 high
OQ1,255	Set all 8 outputs on slot 1 high
:MG_OQ1	Print the decimal value for slot 1's output status
255	IOC-7007 returns value

QD

FUNCTION: Download Array

DESCRIPTION:

The QD command transfers array data from the host computer to the IOC-7007.

QD array[],start,end requires that the array name be specified along with the first element of the array and last element of the array. The array elements can be separated by a comma (,) or by <CR><LF>. The downloaded array is terminated by a \.

ARGUMENTS: QD array[],start,end where

“array[]” is a valid array name

“start” is the first element of the array (default=0)

“end” is the last element of the array (default=last element)

USAGE:

In a Program	Yes
Command Line	Yes
Can be Interrogated	No
Used as an Operand	No

RELATED COMMANDS:

"QU"	Upload array

HINT:

Using Galil terminal software, the command can be used in the following manner:

1. Define the array with the DM command
2. Set the timeout to 0 (this disengages the timeout)
3. Send the command QD

3a. Use the send file command to send the data file.

OR

3b. Enter data manually from the terminal. End the data entry with the character '\'

4. Set the timeout back to a positive number

QR

FUNCTION: I/O Data Record

DESCRIPTION:

The QR command causes the IOC-7007 to return a record of information regarding the IOC board I/O status back to the host PC. This status information includes 4 bytes of header information and specific blocks of I/O information. The details of the status information are described in Chapter 3 of the user's manual.

ARGUMENTS: QR xxxxxxxx where

x is ABCDEFG or any combination to specify the slot data.

QR without any arguments returns all the slots' data.

Chapter 3 of the users manual provides the definition of the data record information.

USAGE:

In a Program	Yes
Command Line	Yes
Can be Interrogated	No
Used as an Operand	No

RELATED COMMANDS:

"QZ"	Return Data Record information

Note: The Galil windows terminal will not display the results of the QR command since the results are in binary format.

QU

FUNCTION: Upload Array

DESCRIPTION:

The QU command transfers array data from the IOC-7007 to a host computer. QU requires that the array name be specified along with the first element of the array and last element of the array. The uploaded array will be followed by a <control>Z as an end of text marker.

ARGUMENTS: QU array[],start,end,delimwhere

"array[]" is a valid array name

"start" is the first element of the array (default=0)

"end" is the last element of the array (default=last element)

"delim" specifies the character used to delimit the array elements. If delim is 1, then the array elements will be separated by a comma. Otherwise, the elements will be separated by a carriage return.

USAGE:

In a Program	Yes
Command Line	Yes
Can be Interrogated	No
Used as an Operand	No

RELATED COMMANDS:

"QD"	Download array

QZ

FUNCTION: Return Data Record information

DESCRIPTION:

The QZ command is an interrogation command that returns information regarding the Data Record. The IOC board's response to this command will be the return of 4 integers separated by commas.

The first field returns the number of slots in the IOC board (for the IOC-7007 this number will always be seven. Future IOC generations may have more or less than 7 slots.)

The second field returns the number of bytes in the general data block of the QR record. This is always 4 for the IOC-7007.

Third field returns 0, representing nothing

Fourth field returns the number of bytes in each of the slot blocks of the QR record

Note: See chapter 3 for more details on the Data Record and the QZ command

ARGUMENTS: QZ

USAGE:

In a Program	Yes
Command Line	Yes
Can be Interrogated	No
Used as an Operand	No

RELATED COMMANDS:

"QR"	Data Record

EXAMPLES:

```
:QZ                    Enter Command
7,4,0,22               There are 22 total bytes in the slot blocks of the IOC-7007
```

RA

FUNCTION: Record Array

DESCRIPTION:

The RA command selects one or two arrays for automatic data capture. The selected arrays must have been dimensioned by the DM command. The data to be captured is specified by the RD command and time interval by the RC command.

ARGUMENTS: RA n [],m [] where

n,m are dimensioned arrays as defined by DM command. The [] contain nothing.

USAGE:

In a Program	Yes
Command Line	Yes
Can be Interrogated	No
Used as an Operand	No

RELATED COMMANDS:

"DM"	Dimension Array
"RD"	Record Data
"RC"	Record Interval

EXAMPLES:

#Record	Label
DM BANK5[100]	Define array for bank 5’s I/O status
RA BANK5[]	Specify Record Mode
RD _TI5	Specify data type for record
RC 1	Begin recording at 2 msec intervals
EN	End

RC

FUNCTION: Record

DESCRIPTION:

The RC command begins recording for the Automatic Record Array Mode (RA). RC 0 stops recording.

ARGUMENTS: RC n,m where

n is an integer 1 thru 8 and specifies 2^n samples between records. RC 0 stops recording.

m is optional and specifies the number of records to be recorded. If m is not specified, the DM number will be used. A negative number for m causes circular recording over array addresses 0 to m-1. The address for the array element for the next recording can be interrogated with _RD.

RC? returns status of recording. ‘1’ if recording, ‘0’ if not recording.

USAGE:

In a Program	Yes
Command Line	Yes
Can be Interrogated	Yes
Used as an Operand	Yes

OPERAND USAGE:

_RC contains status of recording, '1' if recording, '0' if not recording.

RELATED COMMANDS:

"DM"	Dimension Array
"RA"	Record Array Mode
"RD"	Record Data

EXAMPLES:

See example for RA command.

RD

FUNCTION: Record Data

DESCRIPTION:

The RD command specifies the data type to be captured for the Record Array (RA) mode. The command type includes:

DATA TYPE	MEANING
_TIn	Input Slot n Status
_OQn	Output Slot n Status

ARGUMENTS: RD m1, m2 where

the arguments are the data type to be captured using the record array feature. The order is important. Each of the two data types corresponds with the array specified in the RA command.

USAGE:

In a Program	Yes
Command Line	Yes
Can be Interrogated	No
Used as an Operand	Yes

OPERAND USAGE:

_RD contains the address for the next array element for recording.

RELATED COMMANDS:

"DM"	Dimension Array
"RA"	Record Array
"RC"	Record Interval

EXAMPLES:

See example for RA command.

RI

FUNCTION: Return from Interrupt Routine

DESCRIPTION:

The RI command is used to end the interrupt subroutine beginning with the label #ININTn. If the program sequencer was interrupted while waiting for a trippoint (such as WT), RI1 restores the trippoint on the return to the program. RI0 clears the trippoint. The second field of the RI command either restores or disables the input interrupt feature. This field has been added as a special feature of the IOC-7007. To avoid returning to the main program on an interrupt, use the command ZS to zero the subroutine stack. This turns the jump subroutine into a jump only.

ARGUMENTS: RI m,n where

m = 0 or 1

0 clears trippoint (e.g. AI trippoint)

1 restores trippoint

n = 0 (or no field) or 1

0 restores interrupt

1 disables interrupt

USAGE:

In a Program	Yes
Command Line	Yes
Can be Interrogated	No
Used as an Operand	No

RELATED COMMANDS:

#ININTn	Input interrupt subroutine
“II”	Enable input interrupts

EXAMPLES:

#A;II1,0,3	Program label; enable interrupt on input 3
AI5;MG”DONE”;EN	AI trippoint on input 5; end of program
#ININT1	Begin interrupt subroutine
MG "IN[3] INTERRUPTED"	Print Message
CB 3	Set output line 1
RI 1,1	Return to the main program, restore AI trippoint and disable interrupt on input 3

HINT: *An applications program must be executing for the #ININTn subroutine to function.*

RS

FUNCTION: Reset

DESCRIPTION:

The RS command resets the state of the processor to its power-on condition. The previously saved state of the IOC-7007, along with parameter values and saved sequences, are restored.

USAGE:

In a Program	No
Command Line	Yes
Can be Interrogated	No
Used as an Operand	No

EXAMPLES:

RS	Reset the IOC board

<control>R<control>S

FUNCTION: Master Reset

DESCRIPTION:

The Master Reset command resets the IOC-7007 to factory default settings and erases EEPROM.

A master reset can also be performed by installing a jumper on the IOC-7007 at the location labeled MRST and resetting the board (power cycle or pressing the reset button). Remove the jumper after this procedure.

USAGE:

In a Program	No
Command Line	Yes
Can be Interrogated	No
Used as an Operand	No

<control>R<control>V

FUNCTION: Revision Information

DESCRIPTION:

The Revision Information command causes the IOC board to return the firmware revision information.

USAGE:

In a Program	No
Command Line	Yes
Can be Interrogated	No
Used as an Operand	No

SA

FUNCTION: Send Command

DESCRIPTION:

SA sends a command from the IOC to another Galil Ethernet controller. Any command can be sent to a slave device and will be interpreted by the controller as a "local" command.

Note*: A wait statement (e.g. WT5) must be inserted between successive calls to SA.*

ARGUMENTS: SAh= arg or SAh=arg,arg,arg,arg,arg,arg,arg,arg where

h is the handle being used to send commands to another Galil Ethernet controller.

arg is a number, an IOC board or DMC controller operand, variable, mathematical function, or string; The range for numeric values is 4 bytes of integer (2^{31})followed by two bytes of fraction (+/-2,147,483,647.9999). The maximum number of characters for a string is 6 characters. Strings are identified by quotations.

Typical usage would have the first argument as a string such as "OQ" and the subsequent arguments as the arguments to the command: Example SAF= "OQ",0,255 would send the command "OQ 0,255".

USAGE:

In a Program	Yes
Command Line	Yes
Can be Interrogated	No
Used as an Operand	Yes

OPERAND USAGE:

_SAhn gives the value of the response to the command sent with an SA command. The h value represents the handle A thru F and the n value represents the specific field returned from typically a multi-axis controller (1-8). The n value gets defaulted to 1 if unspecified in the operand. If a specific field is not used by the controller (e.g. fifth field on a four axes controller), the operand will be –2^31. Response from another IOC board has usually just one field; therefore the n value does not need to be included in the operand.

RELATED COMMANDS:

"IH"	Set Internet Handles

EXAMPLES:

SAA="KI",1,2	Sends the command to handle A (e.g. a controller): KI 1,2
WT5	
SAA="TE"	Sends the command to handle A (controller): TE
MG _SAA	Display the content of the operand _SAA (first response to TE command)
: 132	
MG _SAA2	Display the content of the operand _SAA (2nd response to TE command)
: 12	

Note*: The SA command does not wait for a response from the slave controller before continuing code execution. Therefore, a WTxx is required between* *two SA commands or between an SA command and querying the response using _SAhn. There is a 38 character maximum string length for the SA command. It is helpful for timing to keep the SA command query as short as possible.*

SB

FUNCTION: Set Bit

DESCRIPTION:

The SB command sets one of the output bits on slots 0 to 6.

Note 1: When using Modbus devices, the I/O points of the modbus devices are calculated using the following formula:

n = (SlaveAddress*10000) + (HandleNum*1000) + ((Module-1)*4) + (Bitnum-1)

Slave Address is used when the ModBus device has slave devices connected to it and specified as Addresses 0 to 255. Please note that the use of slave devices for modbus are very rare and this number will usually be 0.

HandleNum is the handle specifier from A to F (1 to 6).

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.

Note 2: The SB command can be used in the #PLCSCAN thread. In this case, it can only be used for local outputs, not ModBus.

ARGUMENTS: SB n where

n is an integer in the range 0 to 223 decimal.

USAGE:

In a Program	Yes
Command Line	Yes
Can be Interrogated	No
Used as an Operand	No

RELATED COMMAND:

"CB"	Clear Bit
“OB”	Output Bit
“OQ”	Output Port

EXAMPLES:

SB 3	Set output line 3
SB 1	Set output line 1

Note: When communicating between two IOC modules using SB/CB commands – Port 502 must be used when opening the handle (IH).

TB

FUNCTION: Tell Status Byte

DESCRIPTION:

The TB command returns status information from the IOC-7007 as a decimal number. Each bit of the status byte denotes the following condition when the bit is set (high):

BIT	STATUS
Bit 7	Executing program
Bit 6	N/A
Bit 5	N/A
Bit 4	N/A
Bit 3	Input interrupt enabled in thread 0
Bit 2	Executing input interrupt routine in thread 0
Bit 1	0 (Reserved)
Bit 0	Echo on

ARGUMENTS: None

USAGE:

In a Program	Yes
Command Line	Yes
Can be Interrogated	Yes
Used as an Operand	Yes

OPERAND USAGE:

_TB contains the status byte.

EXAMPLES:

TB	Tell status information from the IOC board
129	Executing program and echo on ($2^7 + 2^0 = 128 + 1 = 129$)

TC

FUNCTION: Tell Error Code

DESCRIPTION:

The TC command returns a number between 1 and 255. This number is a code that reflects why a command was not accepted by the IOC-7007. This command is useful when the IOC board halts execution of a program at a command or when the response to a command is a question mark. Entering the TC command will provide the user with a code as to the reason. After TC has been read, it is set to zero. TC 1 returns the text message as well as the numeric code.

ARGUMENTS: TC n

n=0 returns code only

n=1 returns code and message

CODE	**EXPLANATION**	**CODE**	**EXPLANATION**
1	Unrecognized command	65	IN command must have a comma
2	Command only valid from program	66	Array space full
3	Command not valid in program	67	Too many arrays or variables
4	Operand error	71	IN only valid in task #0
5	Input buffer full	80	Record mode already running
6	Number out of range	81	No array or source specified
9	Variable error	82	Undefined Array
10	Empty program line or undefined label	83	Not a valid number
11	Invalid label or line number	84	Too many elements
12	Subroutine more than 16 deep	97	Bad Binary Command Format
14	EEPROM check sum error	98	Binary Commands not valid in application program
15	EEPROM write error	99	Bad binary command number
19	Application strand already executing	120	Bad Ethernet transmit
25	Variable not given in IN command	121	Bad Ethernet packet received
50	Not enough fields	122	Ethernet input buffer overrun
51	Question mark not valid	123	TCP lost sync
52	Missing " or string too long	124	Ethernet handle already in use
53	Error in {}	125	No ARP response from IP address
54	Question mark part of string	126	Closed Ethernet handle
55	Missing [or []	127	Illegal Modbus function code
56	Array index invalid or out of range	131	Serial Port Timeout
57	Bad function or array	132	PLCSCAN Program not defined
58	Not a valid Command Operand (i.e._GNX)	133	Compile error in PLCSCAN
59	Mismatched parentheses	134	PLCSCAN program executing
60	Download error - line too long or too many lines	150	Not defined as digital input
61	Duplicate or bad label	151	Not defined as digital output

62	Too many labels	152	Not defined as analog input
63	IF statement without ENDIF	153	Not defined as analog output

USAGE:

In a Program	Yes
Command Line	Yes
Can be Interrogated	Yes
Used as an Operand	Yes

OPERAND USAGE:

_TC contains the value of the error code.

EXAMPLES:

```
:GF32          Bad command
?TC            Tell error code
1              Unrecognized command
```

TH

FUNCTION: Tell Ethernet Handle

DESCRIPTION:

This command returns a list of data pertaining to the IOC-7007's Ethernet connection. This list begins with the IOC-7007 IP address and Ethernet address (physical address), followed by the availability of each of the 6 Ethernet handles.

ARGUMENTS: None

USAGE:

In a Program	Yes
Command Line	Yes
Can be Interrogated	Yes
Used as an Operand	No

RELATED COMMAND:

"IH"	Internet Handle
"HR"	Handle Restore
"WH"	Which Handle

EXAMPLES:

```
:TH
CONTROLLER IP ADDRESS 10,0,51,82 ETHERNET ADDRESS 10-80-3C-10-01-2F
IHA TCP PORT 1010 TO IP ADDRESS 10,0,51,87 PORT 1030
IHB TCP PORT 1020 TO IP ADDRESS 10,0,51,87 PORT 1070
IHC AVAILABLE
IHD AVAILABLE
IHE AVAILABLE
IHF AVAILABLE
```

TI

FUNCTION: Tell Inputs

DESCRIPTION:

This command returns the state of all inputs on a specified IOC slot. Response is a decimal number which when converted to binary represents the status of all digital inputs on that IOM module. Therefore, if the TI is used to return the state of the 16 port IOM-70016, the returned number will be from 0 to 65,535. Likewise if the TI is used to return the state of an 8 port IOM-70108, the returned number will be from 0 to 256.

ARGUMENTS: TI n where

n is the IOC slot number whose input state is to be interrogated, and n ranges from 0 thru 6.

Note: If slot n does not contain an input module, an error message will be displayed when TI n command is entered.

USAGE:

In a Program	Yes
Command Line	Yes
Can be Interrogated	Yes
Used as an Operand	Yes

OPERAND USAGE:

_TIn contains the status of the input module in slot n.

Note: The operand can be masked to return only specified bit information - see section on Bitwise operations on page 139.

EXAMPLES:

TI1	Tell input state on slot 1
8	Bit 3 on slot 1 is high, others low
TI0	
0	All inputs on slot 0 low
Input =_TI1	Sets the variable, Input, with the TI1 value
:Input=?	
8.0000	IOC-7007 returns value of _TI1

*TIME**

FUNCTION: Time Operand (Keyword)

DESCRIPTION:

*The TIME operand contains the value of the intenal free running, real time clock. The operand TIME will increase by 1 count every update of approximately 1000usec. Note that a value of 1000 for IOC's update rate will actually set an update rate of 976 micoseconds. Thus the value returned by the TIME operand will be off by 2.4% of the actual time.

The clock is reset to 0 with a standard reset or a master reset.

The keyword, TIME, does not require an underscore (_) as with the other operands.

The TIME operand can be used in the #PLCSCAN thread.

USAGE:

In a Program	Yes
Command Line	Yes
Can be Interrogated	No
Used as an Operand	Yes

EXAMPLES:

MG TIME	Display the value of the internal clock
T1=TIME	Sets the variable, T1, with the TIME value

TQ

FUNCTION: Tell Thread Execution

DESCRIPTION:

This command returns a list of all threads and their execution status. The thread number starts at 0 (main thread) and ends at 7.

ARGUMENTS: TQ

USAGE:

In a Program	Yes
Command Line	Yes
Can be Interrogated	Yes
Used as an Operand	No

RELATED COMMAND:

"HX"	Halt Execution
"XQ"	Execute Program

TR

FUNCTION: Trace

DESCRIPTION:

The TR command causes each instruction in a program to be sent out the communications port prior to execution. TR1 enables this function and TR0 disables it. The trace command is useful in debugging programs.

ARGUMENTS: TR n where

n = 0 or 1

0 disables function

1 enables function

USAGE:

In a Program	Yes
Command Line	Yes
Can be Interrogated	No
Used as an Operand	No

TX

FUNCTION: Tell PLC execution Time

DESCRIPTION:

The TX command gives the #PLCSCAN thread execution time. This time begins as soon as the XP command is issued. The TX command can be issued in one of the eight simultaneous threads or from the command line.

ARGUMENTS: TX

USAGE:

In a Program	Yes
Command Line	Yes
Can be Interrogated	No
Used as an Operand	Yes

OPERAND USAGE:

_TX contains the #PLCSCAN execution time

RELATED COMMANDS:

"CP"	Compiles the PLC thread
“HP”	Halt the PLC thread
“XP”	Executes the PLC thread

EXAMPLES:

TX	Tells PLC execution Time

TZ

FUNCTION: Tell I/O Configuration

DESCRIPTION:

This command returns a list of information pertaining to the status of the IOC-7007's seven I/O slots. On each line, the information starts with the slot number (0 to 6), the corresponding I/O number range, the configuration (as inputs or outputs) and the bit value (255 for all 8 output bits high etc.)

ARGUMENTS: TZ

USAGE:

In a Program	Yes
Command Line	Yes
Can be Interrogated	Yes
Used as an Operand	No

RELATED COMMAND:

"TH"	Tell Ethernet Handle
"TQ"	Tell Thread Execution

EXAMPLES:

```
:TZ
Slot 0 (7-0) = Digital Output – value 228 (1110_0100)
Slot 1 = I/O module not detected
Slot 2 (79-64) = Digital Input – value 0 (0000_0000_0000_0000)
Slot 3 = I/O module not detected
Sot 4  (135-128) = Digital Input – value 0 (0000_0000)
Slot 5 = I/O module not detected
Slot 6 (195-192) = Digital Output – value 1 (0001)
```

UL

FUNCTION: Upload

DESCRIPTION:

The UL command transfers data from the IOC-7007 to a host computer. Programs are sent without line numbers. In the terminal utility within DMCTerminal or DMCWIN32, the UL command will bring a Save As window to screen and allow the user to save the uploaded text to file. In a terminal utility such as Hyper Terminal, the UL command will just bring the uploaded program to screen. From there, the user can copy it and save it to a file. In Hyper Terminal, the UL command will be followed by a <control>Z or a \ as an end of Text marker.

ARGUMENTS: None

USAGE:

In a Program	No
Command Line	Yes
Can be Interrogated	No
Used as an Operand	Yes

OPERAND USAGE:

When used as an operand, _UL gives the number of available variables. The total number of variables is 126.

RELATED COMMAND:

"DL"	Download

EXAMPLES:

UL;	Begin upload
#A	Line 0
NO This is an Example	Line 1
NO Program	Line 2
EN	Line 3
<cntrl>Z	Terminator

VF

FUNCTION: Variable Format

DESCRIPTION:

The VF command formats the number of digits to be displayed when interrogating the IOC board.

If a number exceeds the format, the number will be displayed as the maximum possible positive or negative number (i.e. 999.99, -999, $8000 or $7FF).

ARGUMENTS: VF m.n where

m and n are unsigned numbers in the range 0<m<10 and 0<n<4.

m represents the number of digits before the decimal point. A negative m specifies hexadecimal format. When in hexadecimal, the string will be preceded by a $ and Hex numbers are displayed as 2's complement with the first bit used to signify the sign.

n represents the number of digits after the decimal point.

m = ?	Returns the value of the format for variables and arrays.

USAGE:

In a Program	Yes
Command Line	Yes
Can be Interrogated	Yes
Used as an Operand	Yes

OPERAND USAGE:

_VF contains the value of the format for variables and arrays.

EXAMPLES:

VF 5.3	Sets 5 digits of integers and 3 digits after the decimal point
VF 8.0	Sets 8 digits of integers and no fractions
VF -4.0	Specify hexadecimal format with 4 bytes to the left of the decimal

WH

FUNCTION: Which Handle

DESCRIPTION:

The WH command is used to identify the handle in which the command is executed on. The command returns IHA through IHF to indicate on which handle the command was executed on. The command returns RS232 if using serial communication.

ARGUMENTS: None

USAGE:

In a Program	Yes
Command Line	Yes
Can be Interrogated	Yes
Used as an Operand	Yes

RELATED COMMAND:

"TH"	Tell Handle
"IH"	Internet Handle

OPERAND USAGE:

_WH contains the numeric representation of the handle in which a command is executed. Handles A through F are indicated by the value 0 – 5, while a –1 indicates the serial port.

EXAMPLES:

:WH	Request handle identification
IHC	Command executed in handle C
:WH	Request handle identification
RS232	Command executed in RS232 port

WT

FUNCTION: Wait

DESCRIPTION:

The WT command is a trippoint used to time events. After this command is executed, the IOC board will wait for the number of samples specified before executing the next command. The units of the Wait command are in milliseconds.

ARGUMENTS: WT n where

n is an integer in the range 0 to 2 Billion decimal

USAGE:

In a Program	Yes
Command Line	Yes
Can be Interrogated	No
Used as an Operand	No

EXAMPLES:

Assume that 10 seconds after an input goes high, message to user.

```
#A                  Program A
AI3                 After input 3 goes high
WT 10000            Wait 10 seconds
MG “RELAY ON”       Print message
EN                  End Program
```

HINT: *To achieve longer wait intervals, just stack multiple WT commands.*

XP

FUNCTION: Execute PLC Thread

DESCRIPTION:

The XP command executes the #PLCSCAN thread. The XP command can be issued in one of the eight simultaneous threads or from the command line. If the CP is not issued before issuing the XP, then the XP will compile the #PLCSCAN thread prior to executing it. However, the XP will take much more time to process if the program is not compiled first.

ARGUMENTS: XP

USAGE:

In a Program	Yes
Command Line	Yes
Can be Interrogated	No
Used as an Operand	Yes

OPERAND USAGE:

_XP contains a 1 if the #PLCSCAN thread is executing and a 0 if it isn't

RELATED COMMANDS:

"CP"	Compiles the PLC thread
"HP"	Halt the PLC thread
"TX"	Tell execution time of #PLCSCAN

EXAMPLES:

XP	Executes the PLC thread

XQ

FUNCTION: Execute Program

DESCRIPTION:

The XQ command begins execution of a program residing in the program memory of the IOC board. Execution will start at the label or line number specified. Up to 8 programs may be executed simultaneously to perform multitasking. The #PLCSCAN thread is executed with the XP command.

The XQ command can be used within the #PLCSCAN thread as well.

ARGUMENTS: XQ #A,n   XQm,n   where

A is a program name of up to seven characters

m is a line number

n is the thread number (0 thru 7) for multitasking

NOTE: The arguments for the command, XQ, are optional. If no arguments are given, the first program in memory will be executed as thread 0.

USAGE:

In a Program	Yes
Command Line	Yes
Can be Interrogated	No
Used as an Operand	Yes

OPERAND USAGE:

_XQn contains the current line number of execution for thread n, and -1 if thread n is not running.

RELATED COMMANDS:

"HX"	Halt execution

EXAMPLES:

XQ #Apple,0	Start execution at label Apple, thread zero
XQ #data,1	Start execution at label data, thread one
XQ 0	Start execution at line 0

HINT: *Don't forget to quit the edit mode first before executing a program!*

ZC

FUNCTION: User Variable, ZC

DESCRIPTION:

ZC sets the first user variable, which is sent back to a master controller from a slave IOC. This variable is part of the LR record and provides a method for specific board information to be passed to the master controller during data record.

ARGUMENTS: ZCn where

n can be a number, IOC board operand, variable, mathematical function, or string; The range for numeric values is from –2,147,483,648 to +2,147,483,647. The maximum number of characters for a string is 4 characters. Strings are identified by quotations.

NOTE: If n is a '?,' the decimal value of ZC will be returned.

USAGE:

In a Program	Yes
Command Line	Yes
Can be Interrogated	Yes
Used as an Operand	Yes

OPERAND USAGE:

_ZC contains the decimal value of the user variable.

RELATED COMMANDS:

"ZD"	Set second user variable
"LR"	Launch slave data record

EXAMPLES:

ZC 2343	Sets the first user variable to a number (2343)

ZD

FUNCTION: User Variable, ZD

DESCRIPTION:

ZD sets the second user variable, which is sent back to a master controller from a slave IOC. This variable is a part of the LR record and provides a method for specific board information to be passed to the master controller during data record.

ARGUMENTS: ZDn where

n can be a number, IOC board operand, variable, mathematical function, or string; The range for numeric values is from –2,147,483,648 to +2,147,483,647. The maximum number of characters for a string is 4 characters. Strings are identified by quotations.

NOTE: If n is a '?,' the decimal value of ZD will be returned.

USAGE:

In a Program	Yes
Command Line	Yes
Can be Interrogated	Yes
Used as an Operand	Yes

OPERAND USAGE:

_ZD contains the decimal value of the user variable.

RELATED COMMANDS:

"ZC"	Set first user variable
"LR"	Launch slave data record

EXAMPLES:

ZD "INPT"	Sets the second user variable to the string "INPT"

ZS

FUNCTION: Zero Subroutine Stack

DESCRIPTION:

The ZS command is only valid in an application program and is used to avoid returning from an input interrupt. ZS alone returns the stack to its original condition. ZS1 adjusts the stack to eliminate one return. This turns the ‘jump to subroutine, JS,’ into a jump, JP. Do not use RI (Return from Interrupt) when using ZS. To re-enable interrupts, you must use the II command again.

ARGUMENTS: ZS n where

0 returns stack to original condition

1 eliminates one return on stack

USAGE:

In a Program	Yes
Command Line	No
Can be Interrogated	Yes
Used as an Operand	Yes

OPERAND USAGE:

_ZSn contains the stack level for the specified thread where n = 0 thru 7. The response, an integer between zero and sixteen, indicates zero for beginning condition and sixteen for the deepest value.

EXAMPLES:

```
II0,1,7            Input Interrupt on 7
#A;JP #A;EN        Main program
#ININT0            Input Interrupt
MG "INTERRUPT"     Print message
S=_ZS              Interrogate stack before ZS
S=                 Print stack
ZS                 Zero stack
S=_ZS              Interrogate stack after ZS
S=                 Print stack
EN
```

THIS PAGE LEFT BLANK INTENTIONALLY

Chapter 6 Application Programming

Overview

The IOC-7007 provides a versatile programming language that allows users to customize the IOC board for their particular application. Programs can be downloaded into the IOC-7007 memory, freeing up the host computer for other tasks. However, the host computer can send commands to the IOC-7007 at any time, even while a program is being executed. Only ASCII commands can be used for application programming.

In addition to commands that handle I/Os, the IOC-7007 provides commands that allow it to make decisions. These commands include conditional jumps, event triggers, and subroutines. For example, the command JP#LOOP, n<10 causes a jump to the label #LOOP if the variable n is less than 10.

For greater programming flexibility, the IOC-7007 provides user-defined variables, arrays, and arithmetic functions. The following sections in this chapter discuss all aspects of creating applications programs. **The program memory size is 500 lines x 80 characters.**

Editing Programs

The IOC-7007 has an internal editor that may be used to create and edit programs in the IOCs memory. The internal editor is a rudimentary editor and is only recommended when operating with Galil's DOS utilities or through a simple RS-232 communication interface such as the Windows Utility Hyperterminal.

The internal editor is opened by the command ED. **Note that the command ED will not open the internal editor if issued from Galil's Window based software - in this case, a Windows based editor will be automatically opened**. The Windows based editor provides much more functionality and ease-of-use, therefore, the internal editor is most useful when using a simple terminal with the controller and a Windows based editor is not available.

Once the ED command has been given, each program line is automatically numbered sequentially starting with 000. If no parameter follows the ED command, the editor prompter will default to the last line of the last program in memory. If desired, the user can edit a specific line number or label by specifying a line number or label following ED.

ED	Puts Editor at end of last program
ED 5	Puts Editor at line 5
ED #BEGIN	Puts Editor at label #BEGIN

Line numbers appear as 000,001,002, and so on. Program commands are entered following the line numbers. Multiple commands may be given on a single line as long as the total number of characters does not exceed 80 characters per line.

While in the Edit Mode, the programmer has access to special instructions for saving, inserting, and deleting program lines. These special instructions are listed below:

Edit Mode Commands

<RETURN>

Typing the return key causes the current line of entered instructions to be saved. The editor will automatically advance to the next line. Thus, hitting a series of <RETURN> will cause the editor to advance a series of lines.

Note: Changes on a program line will not be saved unless a <return> is given.

<**cntrl**>**P**

The <cntrl>P command moves the editor to the previous line.

<cntrl>I

The <cntrl>I command inserts a line above the current line. For example, if the editor is at line number 2 and <cntrl>I is applied, a new line will be inserted between lines 1 and 2. This new line will be labeled line 2. The old line number 2 is renumbered as line 3.

<cntrl>D

The <cntrl>D command deletes the line currently being edited. For example, if the editor is at line number 2 and <cntrl>D is applied, line 2 will be deleted. The previous line number 3 is now renumbered as line number 2.

<cntrl>Q

The <cntrl>Q quits the editor mode. In response, the IOC-7007 will return a colon.

After the Edit session is over, the user may list the entered program using the LS command. If no operand follows the LS command, the entire program will be listed. The user can start listing at a specific line or label using the operand n. A command and new line number or label following the start listing operand specifies the location at which listing is to stop.

Example:

Instruction	Interpretation
LS	List entire program
LS 5	Begin listing at line 5
LS 5,9	List lines 5 thru 9
LS #A,9	List line label #A thru line 9
LS #A, #A +5	List line label #A and additional 5 lines

Program Format

An IOC-7007 program consists of IOC-7007 instructions combined to solve a programmable logic application. Action instructions, such as setting and clearing I/O bits, are combined with Program Flow instructions to form the complete program. Program Flow instructions evaluate real-time conditions, such as elapsed time or input interrupts, and alter program flow accordingly.

A delimiter must separate each IOC-7007 instruction. Valid delimiters are the semicolon (;) or carriage return. The semicolon is used to separate multiple instructions on a single program line where the maximum number of characters on a line is 80 (including semicolons). A carriage return enters the final command on a program line.

Using Labels in Programs

All IOC-7007 programs must begin with a label and end with an End (EN) statement. Labels start with the pound (#) sign followed by a maximum of seven characters. The first character must be a letter; after that, numbers are permitted. Spaces are not allowed.

The maximum number of labels that can be defined is 126.

Valid labels

```
#BASICIO

#SQUARE

#X1

#input1
```

Invalid labels

```
#1Square

#123

#PROGRAMMING                (longer than 7 characters)
```

Special Labels

The IOC-7007 also has some special labels, which are used to define input interrupt subroutines and command error subroutines. The following is a list of the automatic subroutines supported by the IOC-7007. Sample programs for these subroutines can be found in the section *Automatic Subroutines for Monitoring Conditions.*

#ININTn	Label for Input Interrupt subroutine
#CMDERR	Label for incorrect command subroutine
#TCPERR	Ethernet communication error

The IOC-7007 also has a special label for automatic program execution. A program which has been saved into the controller non-volatile memory can be automatically executed upon power up or reset by beginning the program with the label #AUTO. The program must be saved into non-volatile memory using the command, BP.

Commenting Programs

Using the command, NO

The IOC-7007 provides a command, NO, for commenting programs. This command allows the user to include up to 78 characters on a single line after the NO command and can be used to include comments from the programmer as in the following example:

```
#OUTPUT
NO PROGRAM LABEL
OQ1,7
NO SET THE FIRST 3 BITS ON SLOT 1
EN
NO END OF PROGRAM
```

Note: The NO command is an actual IOC-7007 command. Therefore, inclusion of the NO commands will require process time by the IOC board.

Using REM Statements with the Galil Terminal Software

When using Galil software to communicate with the IOC-7007, REM, as in remark, statements may also be included. 'REM' statements begin with the word 'REM' and may be followed by any comments that are on the same line. The Galil terminal software will remove these statements (or exclude during the download) when the program is downloaded to the IOC board. For example:

```
#OUTPUT
REM PROGRAM LABEL
OQ1,7
REM SET THE FIRST 3 BITS ON SLOT 1
EN
REM END OF PROGRAM
```

The REM statements will be removed when the program is downloaded to IOC-7007.

Executing Programs - Multitasking

The IOC-7007 can run up to 9 independent programs or threads simultaneously. The first eight of these are numbered 0 thru 7, where 0 is the main thread. The ninth thread is labeled #PLCSCAN and allows for the execution of several commands in a deterministic time. The numbered threads are useful for executing independent operations, and #PLCSCAN is useful for real-time PLC applications.

The main thread differs from the others in the following ways:

1. Only the main thread, thread 0, may use the input command, IN.

2. When interrupts are implemented for command errors, the subroutines are executed in thread 0. A new feature on the IOC-7007, something not on other Galil motion controllers, is the ability to execute multiple input interrupts (#ININTn) on designated threads, not limited to the main thread. For more information, refer to the II command in Chapter 5.

To begin execution of the various programs, use the following instruction:

```
XQ #A,n
```

Where A represents the label and n indicates the thread number. To halt the execution of any thread, use the instruction

HX n

where n is the thread number.

Note that both the XQ and HX commands can be performed from within an executing program.

For example:

Instruction	**Interpretation**
#TASK1	Task1 label
AT0	Initialize reference time
CB1	Clear Output 1
#LOOP1	Loop1 label
AT 10	Wait 10 msec from reference time
SB1	Set Output 1
AT -40	Wait 40 msec from reference time, then initialize reference
CB1	Clear Output 1
JP #LOOP1	Repeat Loop1
#TASK2	Task2 label
XQ #TASK1,1	Execute Task1
#LOOP2	Loop2 label
WT20000	Wait for 20 seconds
HX1	Stop thread 1
MG"DONE"	Print Message
EN	End of Program

The program above is executed with the instruction XQ #TASK2,0 which designates TASK2 as the main thread (i.e. Thread 0). #TASK1 is executed within TASK2.

#PLCSCAN thread

The PLC Mode is a special mode of operation that allows fast execution of an application program. The program is compiled into optimized code for faster execution with deterministic timing. This feature provides quick and accurate I/O scans.

The special PLC application program is designated with the label #PLCSCAN. All commands following the #PLCSCAN label are part of the program. A subset of Galil commands that are available for use in the PLC mode are designated with an * in the command list in Chapter 5. Variables and arrays are also available in the PLC mode. The CP command compiles the PLC program and the PLC program is executed with the XP command. Exactly the same number of PLC commands are executed per update period which allows for deterministic timing.

Example:

Instruction	**Interpretation**
#PLCSCAN	PLC special label
IF (@IN[5]=1)	If Input 5 equals one
CB1	Clear Output 1
ELSE	If Input 5 equals 0
SB1	Set Output 1
ENDIF	Terminate IF statement
EN	End PLC program

The are several programming limitations associated with the #PLCSCAN thread. They are listed below.

1. The #PLCSCAN thread executes at the beginning of each 1ms sample period. However, it may take more than one sample period for the thread to complete one cycle. If any digital output commands are issued in the other threads while the #PLCSCAN is within an iteration, these output bits will be changed when the #PLCSCAN finishes that iteration.
2. The #PLCSCAN thread executes repeatedly until the HP command is issued. The EN command does not indicate a true end of program execution, only that there are no more commands in the thread.
3. Commands or calculations within the #PLCSCAN thread are limited to two levels of nested parentheses. Ex: V1=(5*(14/2))
4. Commands or calculations are also limited to 4 operands (numbers or variables) with 3 operators (+, -, *, /, ==, <, >, <=, >=, <>, &, and |). Ex: V5=100*2*4
5. Square brackets, [], are not allowed to be nested.
6. No operators are allowed within square brackets.
7. Jumps within the #PLCSCAN thread cannot be made to locations outside the thread.
8. IF/ELSE statements can be nested up to 16 levels deep.
9. All numbers used in mathematical and comparison operations can only be integers.
10. Analog references are in counts, not Volts.
11. Operands (commands starting with the underscore character "_") are not supported.

Debugging Programs

The IOC-7007 provides commands and operands that are useful in debugging application programs. These commands include interrogation commands to monitor program execution, determine the state of the IOC board and the contents of the IOC-7007 program, array, and variable space. Operands also contain important status information, which can help to debug a program.

Trace Commands

The trace command causes the IOC to send each line in a program to the host computer immediately prior to execution. Tracing is enabled with the command, TR1. TR0 turns the trace function off. Note: When the trace function is enabled, the line numbers as well as the command line will be displayed as each command line is executed.

Error Code Command

When a program error occurs, the IOC-7007 halts the program execution at the point of the error. To display the last line number of program execution, issue the command, MG _ED.

The user can obtain information about the type of error condition that occurred by using the command TC1. This command returns a number and text message, which describe the error condition. The command TC0 (or TC) will return the error code without the text message. For more information about the command TC, see Chapter 5.

RAM Memory Interrogation Commands

For debugging the status of the program memory, array memory, or variable memory, the IOC-7007 has several useful commands. The command DM ? will return the number of array elements currently available. The command DA ? will return the number of arrays that can be currently defined. For example, the IOC-

7007 has a maximum of 2000 array elements in up to 14 arrays. If a single array of 100 elements is defined, the command DM ? will return the value 1900, and the command DA ? will return 13.

To list the contents of the variable space, use the interrogation command LV (List Variables). To list the contents of array space, use the interrogation command LA (List Arrays). To list the contents of the program space, use the interrogation command LS (List Program). To list the application program labels only, use the interrogation command LL (List Labels).

Operands

In general, all operands provide information that may be useful in debugging an application program. Below is a list of operands that are particularly valuable for program debugging. To display the value of an operand, the message command may be used. For example, since the operand, _ED, contains the last line of program execution, the command MG _ED will display this line number.

_ED contains the last line of program execution (useful to determine where program stopped)

_DL contains the number of available labels (126 max.)

_UL contains the number of available variables (126 max.)

_DA contains the number of available arrays (14 max.)

_DM contains the number of available array elements (2000 max.)

Debugging Example:

The following program has an error. It attempts to set bit 34 high, but "SD" is used as the command instead of "SB". When the program is executed, the IOC-7007 stops at line 001. The user can then query the IOC board using the command, TC1. The IOC-7007 responds with the corresponding explanation:

Instruction	**Interpretation**
:LS	List Program
000 #A	Program Label
001 SD34	Set bit 34 high
002 SB35	Set bit 35 high
003 MG"DONE"	Print message
004 EN	End
:XQ #A	Execute #A
?001 SD34	Error on Line 1
:TC1	Tell Error Code
130 Unrecognized Command	This command doesn't
:MG_ED	Print line number where problem occurred
1.00	The error occurred on line 1 of the program

Program Flow Commands

The IOC-7007 provides instructions to control program flow. The IOC program sequencer normally executes program instructions sequentially. The program flow can be altered with the use of interrupts and conditional jump statements.

Interrupts

To function independently from the host computer, the IOC-7007 can be programmed to make decisions based on the occurrence of an input interrupt, causing the IOC board to wait for multiple inputs to change their logic levels before jumping into a corresponding subroutine. Normally, in the case of a Galil controller, when an interrupt occurs, the main thread will be halted. However, in the IOC-7007, the user can indicate in which thread (the thread must be already running when the interrupt occurs) the interrupt subroutine is to be run. When the interrupt occurs, the specified thread's main program will be paused to allow the interrupt subroutine to be executed. Therefore, the user has the choice of interrupting a particular thread execution upon an input interrupt (see II command on page 73). In this way, the IOC-7007 can make decisions based on its own I/O status without intervention from a host computer.

Examples:

Interrupt

Instruction	**Interpretation**
#A	Program Label
XQ#B,1	Execute #B in thread 1
II1,0,-1&3	#ININT1 in thread 0 when input 1 low and input 3 high
II2,1,-5&10	#ININT2 in thread 1 when input 5 low and input 10 high
AI 13&14	Trippoint on inputs 13 and 14
#LOOP;JP#LOOP	Pseudo program – Loop indefinitely
EN	End program
#B	Program Label
AI 7&-8	Trippoint on inputs 7 and 8
#LOOP2	
SB20	Set bit 20 high
WT500	Wait for half a second
CB20	Set bit 20 low
WT500	Wait for 500msec
JP#LOOP2	Create a 'light-blinker' effect
EN	End program
#ININT1	Input interrupt program label
MG"Loop stops"	Print message, saying loop program in main thread halted
RI0	Return to main program without restoring trippoint, but keeping the interrupt enabled
#ININT2	
MG"Blinker stops"	Print message, saying blinker effect in thread 1 halted, since #ININT2 runs in thread 1
WT10000	Wait 10 seconds for user to reset inputs 5 and 10
RI1,1	Return to thread 1's main program (blinker continues) while restoring trippoint on inputs 5 and 10; interrupt disabled

Note: This multitasking program can be executed with the instruction XQ #A,0 designating A as the main thread (i.e. Thread 0). #B is executed within A.

Event Trigger

This example waits for input 1 to go low and input 3 to go high, and then execute the TZ interrogation command. **Note:** The AI command actually halts execution of the program until the input occurs. If you do not want to halt the program sequences, use the Input Interrupt function (II) or a conditional jump on an input, such as:

```
JP #GO,(@IN[1] = 0) | (@IN[3] = 1).
```

Instruction	Interpretation
#INPUT	Program Label
AI-1&3	Wait for input 1 low and input 3 high
TZ	List the entire I/O configuration
EN	End program

Conditional Jumps

The IOC-7007 provides Conditional Jump (JP) and Conditional Jump to Subroutine (JS) instructions for branching to a new program location based on a specified condition. The conditional jump determines if a condition is satisfied and then branches to a new location or subroutine. Unlike event triggers such as the AI command, the conditional jump instruction does not halt the program sequence. Conditional jumps are useful for testing events in real-time. They allow the IOC-7007 to make decisions without a host computer.

Command Format - JP and JS

Format	Description
JS destination, logical condition	Jump to subroutine if logical condition is satisfied
JP destination, logical condition	Jump to location if logical condition is satisfied

The destination is a program line number or label where the program sequencer will jump if the specified condition is satisfied. Note that the line number of the first line of program memory is 0. The comma designates "IF". The logical condition tests two operands with logical operators.

Logical operators:

Operator	Description
<	less than
>	greater than
=	equal to
<=	less than or equal to
>=	greater than or equal to
<>	not equal

Conditional Statements

The conditional statement is satisfied if it evaluates to any value other than zero. The conditional statement can be any valid IOC-7007 numeric operand, including variables, array elements, numeric values, functions, keywords, and arithmetic expressions. If no conditional statement is given, the jump will always occur.

Examples:

Number	V1=6
Numeric Expression	V1=V7*6
	@ABS[V1]>10
Array Element	V1<Count[2]
Variable	V1<V2
Internal Variable	_TI1=255
	_OQ2<>15
I/O	V1>@IN[2]
	@IN[1]=0

Multiple Conditional Statements

The IOC-7007 will accept multiple conditions in a single jump statement. The conditional statements are combined in pairs using the operands "&" and "|". The "&" operand between any two conditions, requires that both statements be true for the combined statement to be true. The "|" operand between any two conditions requires that only one statement be true for the combined statement to be true.

***Note:** Each condition must be placed in parentheses for proper evaluation by the IOC board. In addition, the IOC-7007 executes operations from left to right.*

For example, using variables named V1, V2, V3 and V4:

JP #TEST, (V1<V2) & (V3<V4)

In this example, this statement will cause the program to jump to the label #TEST if V1 is less than V2 and V3 is less than V4. To illustrate this further, consider this same example with an additional condition:

JP #TEST, ((V1<V2) & (V3<V4)) | (V5<V6)

This statement will cause the program to jump to the label #TEST under two conditions: 1) If V1 is less than V2 AND V3 is less than V4.   2) If V5 is less than V6.

Using the JP Command:

If the condition for the JP command is satisfied, the IOC-7007 branches to the specified label or line number and continues executing commands from this point. If the condition is not satisfied, the IOC board continues to execute the next commands in sequence.

Instruction	Interpretation
JP #Loop,COUNT<10	Jump to #Loop if the variable, COUNT, is less than 10
JS #MOVE2,@IN[1]=1	Jump to subroutine #MOVE2 if input 1 is logic level high. After the subroutine MOVE2 is executed, the program sequencer returns to the main program location where the subroutine was called.
JP #BLUE,@ABS[V2]>2	Jump to #BLUE if the absolute value of variable, V2, is greater than 2
JP #C,V1*V7<=V8*V2	Jump to #C if the value of V1 times V7 is less than or equal to the value of V8*V2
JP#A	Jump to #A

Using If, Else, and Endif Commands

The IOC-7007 provides a structured approach to conditional statements using IF, ELSE and ENDIF commands.

Using the IF and ENDIF Commands

An IF conditional statement is formed by the combination of an IF and ENDIF command. The IF command has arguments of one or more conditional statements. If the conditional statement(s) evaluates true, the command interpreter will continue executing commands which follow the IF command. If the conditional statement evaluates false, the IOC-7007 will ignore commands until the associated ENDIF command is executed <u>OR</u> an ELSE command occurs in the program (see discussion of ELSE command below).

Note: An ENDIF command must always be executed for every IF command that has been executed. It is recommended that the user not include jump commands inside IF conditional statements, since this causes re-direction of command execution. In this case, the command interpreter may not execute an ENDIF command.

Using the ELSE Command

The ELSE command is an optional part of an IF conditional statement and allows for the execution of commands only when the argument of the IF command evaluates False. The ELSE command must occur after an IF command and has no arguments. If the argument of the IF command evaluates false, the IOC-7007 will skip commands until the ELSE command. If the argument for the IF command evaluates true, the IOC board will execute the commands between the IF and ELSE commands.

Nesting IF Conditional Statements

The IOC-7007 allows for IF conditional statements to be included within other IF conditional statements. This technique is known as 'nesting' and the IOC-7007 allows up to 255 IF conditional statements to be nested. This is a very powerful technique allowing the user to specify a variety of different cases for branching.

Command Format - IF, ELSE and ENDIF

Function	Condition
IF conditional statement(s)	Execute commands proceeding IF command (up to ELSE command) if conditional statement(s) is true, otherwise continue executing at ENDIF command or optional ELSE command.
ELSE	Optional command. Allows for commands to be executed when argument of IF command evaluates not true. Can only be used with IF command.
ENDIF	Command to end IF conditional statement. Program must have an ENDIF command for every IF command.

Example using IF, ELSE and ENDIF:

Instruction	Interpretation
#TEST	Begin Main Program "TEST"
#LOOP	Begin loop inside main program
TEMP=@IN[1]|@IN[2]	TEMP is equal to 1 if either Input 1 or Input 2 is high
JS#COND, TEMP=1	Jump to subroutine if TEMP equals 1
JP#LOOP	Loop back if TEMP doesn't equal 1
EN	End of main program
#COND	Begin subroutine "COND"
IF (@IN[1]=0)	IF conditional statement based on input 1
IF (@IN[2]=0)	2nd IF conditional statement executed if 1st IF conditional true
MG "INPUT 1 AND INPUT 2 ARE ACTIVE"	Message to be executed if 2nd IF conditional is true
ELSE	ELSE command for 2nd IF conditional statement
MG "ONLY INPUT 1 IS ACTIVE	Message to be executed if 2nd IF conditional is false
ENDIF	End of 2nd conditional statement
ELSE	ELSE command for 1st IF conditional statement
MG"ONLY INPUT 2 IS ACTIVE"	Message to be executed if 1st IF conditional statement
ENDIF	End of 1st conditional statement
#WAIT	Label to be used for a loop
JP#WAIT,(@IN[1]=0) & (@IN[2]=0)	Loop until both input 1 and input 2 are not active
EN	End of subroutine

Stack Manipulation

It is possible to manipulate the subroutine stack by using the ZS command. Every time a JS instruction, interrupt or automatic routine (such as #ININTn or #CMDERR) is executed, the subroutine stack is incremented by 1. Normally the stack is restored with an EN instruction. Occasionally it is desirable not to return back to the program line where the subroutine or interrupt was called. The ZS1 command clears 1 level of the stack. This allows the program sequencer to continue to the next line. The ZS0 command resets the stack to its initial value. For example, if an interrupt occurs and the #ININT1 routine is executed, it may be desirable to restart the program sequence instead of returning to the location where the interrupt occurred. To do this, give a ZS (ZS0) command at the end of the #ININT1 routine.

Auto-Start Routine

The IOC-7007 has a special label for automatic program execution. A program that has been saved into the IOC non-volatile memory can be automatically executed upon power up or reset, simply by beginning the program with the label #AUTO.

Note: The program must be saved into non-volatile memory using the command, BP.

Automatic Subroutines for Monitoring Conditions

Often it is desirable to monitor certain conditions continuously without tying up the host or IOC-7007 program sequences. The IOC-7007 can monitor several important conditions in the background. These conditions include checking for the occurrence of a defined input, position error, a command error, or an Ethernet communication error. Automatic monitoring is enabled by inserting a special, predefined label in the applications program. The pre-defined labels are:

SUBROUTINE	DESCRIPTION
#ININTn	Input specified by II goes low (n from 0 to 7)
#CMDERR	Bad command given
#TCPERR	Ethernet communication error

For example, the #ININT label could be used to designate an input interrupt subroutine. When the specified input occurs, the program will be executed automatically.

NOTE: An application program must be running for automatic monitoring to function.

Example - Input Interrupt

Instruction	Interpretation
#A	Label
II1	Input Interrupt on 1
#LOOP;JP#LOOP;EN	Loop
#ININT	Input Interrupt
MG “INPUT 1 IS HIGH”	Send Message to screen
RI0	Return from interrupt routine to Main Program and do not re-enable trippoints

Example - Command Error

Instruction	Interpretation
#BEGIN	Begin main program
IN "ENTER THE OUTPUT (0-7)", OUT	Prompt for output number
OQ 0,0	Clear all the outputs on slot 0
SB OUT	Set the specified bit
JP #BEGIN	Repeat
EN	End main program
#CMDERR	Command error utility
JP#DONE,_ED<>3	Check if error on line 3
JP#DONE,_TC<>6	Check if out of range
MG "VALUE OUT OF RANGE"	Send message
MG "TRY AGAIN"	Send message
ZS1	Adjust stack
JP #BEGIN	Return to main program
#DONE	End program if other error
ZS0	Zero stack
EN	End program

The above program prompts the operator to enter the output port to set. The example assumes that only slot 0 is occupied by an IOM output module. If the operator enters a number out of range (greater than 7), the #CMDERR routine will be executed prompting the operator to enter a new number.

In multitasking applications, there is an alternate method for handling command errors from different threads. Using the XQ command along with the special operands described below allows the controller to either skip or retry invalid commands.

OPERAND	FUNCTION
_ED1	Returns the number of the thread that generated an error
_ED2	Retry failed command (operand contains the location of the failed command)
_ED3	Skip failed command (operand contains the location of the command after the failed command)

The operands are used with the XQ command in the following format:

```
XQ _ED2 (or _ED3),_ED1,1
```

Where the ",1" at the end of the command line indicates a restart; therefore, the existing program stack will not be removed when the above format executes.

The following example shows an error correction routine that uses the operands. **This example assumes that only slot 0 or the IOC controller is occupied by an 8 port IOM module.**

Example - Command Error w/Multitasking

Instruction	Interpretation
#A	Begin thread 0 (continuous loop)
JP#A	
EN	End of thread 0
#B	Begin thread 1
N=8	Create new variable
SB N	Set the eighth bit, an invalid value
TY	Issue invalid command
EN	End of thread 1
#CMDERR	Begin command error subroutine
IF _TC=6	If error is out of range (SB 8)
N=1	Set N to a valid number
XQ _ED2,_ED1,1	Retry SB N command
ENDIF	
IF _TC=1	If error is invalid command (TY)
XQ _ED3,_ED1,1	Skip invalid command
ENDIF	
EN	End of command error routine

Example – Ethernet Communication Error

This simple program executes in the IOC-7007 and indicates (via the serial port) when a communication handle fails. By monitoring the serial port, the user can re-establish communication if needed.

Instruction	Interpretation
#LOOP	Simple program loop
JP#LOOP	
EN	
#TCPERR	Ethernet communication error auto routine
MG {P1}_IA4	Send message to serial port indicating which handle did not receive proper acknowledgment.
RE	Return to main program

Note: The #TCPERR routine only detects the loss of TCP/IP Ethernet handles, not UDP.

Mathematical and Functional Expressions

Mathematical Operators

For manipulation of data, the IOC-7007 provides the use of the following mathematical operators:

Operator	Function
+	Addition
-	Subtraction
*	Multiplication
/	Division
&	Logical And (Bit-wise)
|	Logical Or (On some computers, a solid vertical line appears as a broken line)
( )	Parenthesis

The numeric range for addition, subtraction and multiplication operations is +/-2,147,483,647.9999. The precision for division is 1/65,000.

Mathematical operations are executed from left to right. Calculations within parentheses have precedence.

Examples:

SPEED=7.5*V1/2	The variable, SPEED, is equal to 7.5 multiplied by V1 and divided by 2
COUNT=COUNT+2	The variable, COUNT, is equal to the current value plus 2.
RESULT=Val1 - (@COS[45]*40)	Puts the value of Val1 - 28.28 in RESULT. 40 * cosine of 45° is 28.28
K=@IN[1]&@IN[2]	K is equal to 1 only if Input 1 and Input 2 are high

Note: Mathematical operations can be done in hexadecimal as well as decimal. Just precede hexadecimal numbers with a $ sign so that the IOC recognizes them as such.

Bit-Wise Operators

The mathematical operators & and | are bit-wise operators. The operator, &, is a Logical And. The operator, |, is a Logical Or. These operators allow for bit-wise operations on any valid IOC-7007 numeric operand, including variables, array elements, numeric values, functions, keywords, and arithmetic expressions. The bit-wise operators may also be used with strings. This is useful for separating characters from an input string. When using the input command for string input, the input variable will hold up to 6 characters. These characters are combined into a single value, which is represented as 32 bits of integer and 16 bits of fraction. Each ASCII character is represented as one byte (8 bits), therefore the input variable can hold up to six characters. The first character of the string will be placed in the top byte of the variable and the last character will be placed in the lowest significant byte of the fraction. The characters can be individually separated, by using bit-wise operations as illustrated in the following example:

Instruction	Interpretation
#TEST	Begin main program
IN "ENTER",LEN{S6}	Input character string of up to 6 characters into variable 'LEN'
FLEN=@FRAC[LEN]	Define variable 'FLEN' as fractional part of variable 'LEN'
FLEN=$10000*FLEN	Shift FLEN by 32 bits (IE - convert fraction, FLEN, to integer)
LEN1=(FLEN&$00FF)	Mask top byte of FLEN and set this value to variable 'LEN1'
LEN2=(FLEN&$FF00)/$100	Let variable, 'LEN2' = top byte of FLEN
LEN3=LEN&$000000FF	Let variable, 'LEN3' = bottom byte of LEN
LEN4=(LEN&$0000FF00)/$100	Let variable, 'LEN4' = second byte of LEN
LEN5=(LEN&$00FF0000)/$10000	Let variable, 'LEN5' = third byte of LEN
LEN6=(LEN&$FF000000)/$1000000	Let variable, 'LEN6' = fourth byte of LEN
MG LEN6 {S4}	Display 'LEN6' as string message of up to 4 chars
MG LEN5 {S4}	Display 'LEN5' as string message of up to 4 chars
MG LEN4 {S4}	Display 'LEN4' as string message of up to 4 chars
MG LEN3 {S4}	Display 'LEN3' as string message of up to 4 chars
MG LEN2 {S4}	Display 'LEN2' as string message of up to 4 chars
MG LEN1 {S4}	Display 'LEN1' as string message of up to 4 chars
EN	

This program will accept a string input of up to 6 characters, parse each character, and then display each character. Notice also that the values used for masking are represented in hexadecimal (as denoted by the preceding '$'). For more information, see the section on *Sending Messages* (page 146).

To illustrate further, if the user types in the string "TESTME" at the input prompt, the IOC-7007 will respond with the following:

T	Response from command MG LEN6 {S4}
E	Response from command MG LEN5 {S4}
S	Response from command MG LEN4 {S4}
T	Response from command MG LEN3 {S4}
M	Response from command MG LEN2 {S4}
E	Response from command MG LEN1 {S4}

Functions

Function	Description
@SIN[n]	Sine of n (n in degrees, with range of -32768 to 32767 and 16-bit fractional resolution)
@COS[n]	Cosine of n (n in degrees, with range of -32768 to 32767 and 16-bit fractional resolution)
@TAN[n]	Tangent of n (n in degrees, with range of -32768 to 32767 and 16-bit fractional resolution)
@ASIN*[n]	Arc Sine of n, between -90° and +90°. Angle resolution in 1/64000 degrees.
@ACOS* [n]	Arc Cosine of n, between 0 and 180°. Angle resolution in 1/64000 degrees.
@ATAN* [n]	Arc Tangent of n, between -90° and +90°. Angle resolution in 1/64000 degrees
@COM[n]	1's Complement of n
@ABS[n]	Absolute value of n
@FRAC[n]	Fraction portion of n
@INT[n]	Integer portion of n
@RND[n]	Round of n (Rounds up if the fractional part of n is .5 or greater)

@SQR[n]	Square root of n (Accuracy is +/-.004)
@IN[n]	Return digital input at general input n (where n starts at 1)
@OUT[n]	Return digital output at general output n (where n starts at 1)

Note: These functions are multi-valued. An application program may be used to find the correct band.

Functions may be combined with mathematical expressions. The order of execution of mathematical expressions is from left to right and can be over-ridden by using parentheses.

Examples:

V1=@ABS[V7]	The variable, V1, is equal to the absolute value of variable V7.
V2=5*@SIN[POS]	The variable, V2, is equal to five times the sine of the variable, POS.
V3=@IN[1]	The variable, V3, is equal to the digital value of input 1.

Variables

For applications that require a parameter that is variable, the IOC board provides 126 variables. These variables can be numbers or strings. A program can be written in which certain parameters, such as I/O status or particular I/O bit, are defined as variables. The variables can later be assigned by the operator or determined by program calculations. Example:

SB RED	Assigns variable RED to SB command
T1=_OQ1	Assigns value of bank 1's I/O status to T1.

Programmable Variables

The IOC-7007 allows the user to create up to 126 variables. Each variable is defined by a name, which can be up to eight characters. The name must start with an alphabetic character, however, and numbers are permitted in the rest of the name. *Spaces are not permitted.* Variable names should not be the same as IOC-7007 instructions. For example, RS is not a good choice for a variable name.

Examples of valid and invalid variable names are:

Valid Variable Names

STATUS1

TEMP1

POINT

Invalid Variable Names

REALLONGNAME	; Cannot have more than 8 characters
123	; Cannot begin variable name with a number
STAT Z	; Cannot have spaces in the name

Assigning Values to Variables:

Assigned values can be numbers, internal variables and keywords, functions, IOC board parameters and strings; the range for numeric variable values is 4 bytes of integer (2^{31}) followed by two bytes of fraction (+/-2,147,483,647.9999).

Numeric values can be assigned to programmable variables using the equal sign.

Any valid IOC-7007 functions can be used to assign a value to a variable. For example, S1=@ABS[V2] or S2=@IN[1]. Arithmetic operations are also permitted.

To assign a string value, the string must be in quotations. String variables can contain up to six characters that must be in quotation.

Examples:

INTWO=_TI2	Assigns returned value from TI2 command to variable INTWO.
INPUT=@IN[1]	Assigns logical value of input 1 to variable INPUT
V2=V1+V3*V4	Assigns the value of V1 plus V3 times V4 to the variable V2.
VAR="CAT"	Assign the string, CAT, to VAR

Displaying the value of variables at the terminal

Variables may be sent to the screen using the format, variable=. For example, V1= , returns the value of the variable V1.

Operands

Operands allow status parameters of the IOC-7007 to be incorporated into programmable variables and expressions. Most IOC-7007 commands have an equivalent operand - which are designated by adding an underscore (_) prior to the command (see command reference section). Operands are not supported in #PLCSCAN.

Examples of Internal Variables:

IN1=@IN[1]	Assigns value of input 1 to the variable IN1.
JP #LOOP,_OQ1=1	Jump to #LOOP if only bit 0 of bank 1 is set high
JP #ERROR,_TC=1	Jump to #ERROR if the error code equals 1.

Operands can be used in an expression and assigned to a programmable variable, but they cannot be assigned a value. For example: _OQ1=1 is invalid.

Special Operands (Keywords)

The IOC-7007 provides a few additional operands that give access to internal variables that are not accessible by standard IOC-7007 commands.

Operand	Function
_BN	*Returns serial # of the board.

_DA	*Returns the number of arrays available
_DL	*Returns the number of available labels for programming
_DM	*Returns the available array memory
_UL	*Returns the number of available variables
TIME	Free-Running Real Time Clock (off by 2.4% - Resets with power-on). Note: TIME does not use an underscore character (_) as other keywords.

Note: All these keywords have corresponding commands except for TIME.

Examples of Keywords:

V1=_DA	Assign V1 the number of available array names
V3=TIME	Assign V3 the current value of the time clock

Arrays

For storing and collecting numerical data, the IOC-7007 provides array space for 2000 elements. The arrays are one-dimensional, and up to 16 different arrays may be defined. Each array element has a numeric range of 4 bytes of integer (2^{31}) followed by two bytes of fraction (+/-2,147,483,647.9999). Arrays can be used to capture real-time data, such as the bit status of a particular I/O bank.

Defining Arrays

An array is defined with the command DM. The user must specify a name and the number of entries to be held in the array. An array name can contain up to eight characters, starting with an uppercase alphabetic character. The number of entries in the defined array is enclosed in [ ].

Example:

DM IOSTAT[100]	Defines an array names IOSTAT with 100 entries
DM TEMP [0]	Frees array space

Assignment of Array Entries

Like variables, each array element can be assigned a value. Assigned values can be numbers or returned values from instructions, functions and keywords.

Array elements are addressed starting at count 0. For example, the first element in the OUTPUT array (defined with the DM command, DM OUTPUT[7]) would be specified as OUTPUT[0].

Values are assigned to array entries using the equal sign. Assignments are made one element at a time by specifying the element number with the associated array name.

NOTE: Arrays must be defined using the command, DM, before assigning entry values.

Examples:

DM OUTPUT[10]	Dimension Output Array
OUTPUT[1]=3	Assigns the second element of the array, OUTPUT, the value of 3.
OUTPUT[1]=	Returns array element value
OUTPUT[9]=_OQ0	Assigns the 10th element of the array, OUTPUT, the value for slot 0's output bit status.

CON[2]=@COS[POS]*2	Assigns the third element of the array CON the cosine of the variable POS multiplied by 2.
TIMER[1]=TIME	Assigns the second element of the array timer the returned value of the TIME keyword.

Using a Variable to Address Array Elements

An array element number can also be a variable. This allows array entries to be assigned sequentially using a counter.

For example:

Instruction	Interpretation
#A	Begin Program
COUNT=0;DM POS[10]	Initialize counter and define array
#LOOP	Begin loop
WT 10	Wait 10 msec
INPUT[COUNT]=_TI0	Record bank 0's input bit value into array element
INPUT[COUNT]=	Report input bit value
COUNT=COUNT+1	Increment counter
JP #LOOP,COUNT<10	Loop until 10 elements have been stored
EN	End Program

The above example records 10 input bit values for bank 0 at a rate of one value per 10 msec. The values are stored in an array named INPUT. The variable, COUNT, is used to increment the array element counter. The above example can also be executed with the automatic data capture feature described below.

Uploading and Downloading Arrays to On Board Memory

Arrays may be uploaded and downloaded using the QU and QD commands.

```
QU array[],start,end,delim

QD array[],start,end
```

where array is an array name such as A[].

Start is the first element of array (default=0)

End is the last element of array (default=last element)

Delim specifies whether the array data is separated by a comma (delim=1) or a carriage return (delim=0).

The file is terminated using <control>Z, <control>Q, <control>D or \.

Automatic Data Capture into Arrays

The IOC-7007 provides a special feature for automatic capture of data such as inputs or outputs. Up to four types of data can be captured and stored in four arrays. The capture rate or time interval may be specified. Recording can be done as a one-time event or as a circular continuous recording.

Command Summary - Automatic Data Capture

Command	Description
RA n[],m[],o[],p[]	Selects up to four arrays for data capture. The arrays must be defined with the DM command.

RD type1,type2,type3,type4	Selects the type of data to be recorded, where type1, type2, type3, and type 4 represent the various types of data (see table below). The order of data type is important and corresponds with the order of n,m,o,p arrays in the RA command.
RC n,m	The RC command begins data collection. Sets data capture time interval where n is an integer between 1 and 8 and designates 2^n msec between data. m is optional and specifies the number of elements to be captured. If m is not defined, the number of elements defaults to the smallest array defined by DM. When m is a negative number, the recording is done continuously in a circular manner. _RD is the recording pointer and indicates the address of the next array element. n=0 stops recording.
RC?	Returns a 0 or 1 where, 0 denotes not recording, 1 specifies recording in progress

Data Types for Recording:

Data type	Description
_TIn	Inputs at bank n
_OQn	Outputs at bank n

Operand Summary - Automatic Data Capture

_RC	Returns a 0 or 1 where, 0 denotes not recording, 1 specifies recording in progress
_RD	Returns address of next array element.

Deallocating Array Space

Array space may be deallocated using the DA command followed by the array name. DA*[0] deallocates all the arrays.

Input of Data (Numeric and String)

Input of Data

The command, IN, is used to prompt the user to input numeric or string data. Using the IN command, the user may specify a message prompt by placing a message in quotations. When the IOC-7007 executes an IN command, it will wait for the input of data. The input data is assigned to the specified variable or array element.

Note: The IN command is only valid when communicating through RS232. This command will not work through the Ethernet.

An Example for Inputting Numeric Data

```
#A
IN "Enter output number", OUT
```

EN

In this example, the message “Enter output number” is displayed on the computer screen. The IOC board waits for the operator to enter a value. The operator enters the numeric value that is then assigned to the variable, OUT.

Inputting String Variables

String variables with up to six characters may input using the specifier, {Sn} where n represents the number of string characters to be input. If n is not specified, six characters will be accepted. For example, IN "Enter X,Y or Z", V{S} specifies a string variable of up to six characters to be input.

Output of Data (Numeric and String)

Numerical and string data can be output from the IOC board using several methods. The message command, MG, can output string and numerical data. Also, the IOC-7007 can be commanded to return the values of variables and arrays, as well as other information using the interrogation commands, such as V1=? and TZ.

Sending Messages

Messages may be sent using the message command, MG. This command sends specified text and numerical or string data from variables or arrays to the screen.

Text strings are specified in quotes and variable or array data is designated by the name of the variable or array. For example:

MG "The Final Value is", RESULT

In addition to variables, functions and commands, responses can be used in the message command. For example:

```
MG "The input is", @IN[1]
```

Formatting Messages

String variables can be formatted using the specifier, {Sn} where n is the number of characters, 1 thru 6. For example:

```
MG STR {S3}
```

This statement returns 3 characters of the string variable named STR.

Numeric data may be formatted using the {Fn.m} expression following the completed MG statement. {$n.m} formats data in HEX instead of decimal. The actual numerical value will be formatted with n characters to the left of the decimal and m characters to the right of the decimal. Leading zeros will be used to display specified format.

For example:

```
MG "The Final Value is", RESULT {F5.2}
```

If the value of the variable RESULT is equal to 4.1, this statement returns the following:

The Final Value is 00004.10

If the value of the variable RESULT is equal to 999999.999, the above message statement returns the following:

```
The Final Value is 99999.99
```

The message command normally sends a carriage return and line feed following the statement. The carriage return and the line feed may be suppressed by sending {N} at the end of the statement. This is useful when a text string needs to surround a numeric value.

Example:

```
#A
NAME="Jon"
LNAME="Jonson"
MG "The name is ", NAME{S3} {N}
MG " ",LNAME{S6}
EN
```

When #A is executed, the above example will appear on the screen as:

```
The name is Jon Jonson
```

Using the MG Command to Configure Terminals

The MG command can be used to configure a terminal. Any ASCII character can be sent by using the format {^n} where n is any integer between 1 and 255.

Example:

```
MG {^07} {^255}
```

sends the ASCII characters represented by 7 and 255 to the bus.

Summary of Message Functions:

Function	Description
" "	Surrounds text string
{Fn.m}	Formats numeric values in decimal n digits to the right of the decimal point and m digits to the left
{$n.m}	Formats numeric values in hexadecimal
{^n}	Sends ASCII character specified by integer n
{N}	Suppresses carriage return/line feed
{Sn}	Sends the first n characters of a string variable, where n is 1 thru 6.

Displaying Variables and Arrays

Variables and arrays may be sent to the screen using the format, variable= **or** array[x]=. For example, V1= , returns the value of V1.

Removing Leading Zeros from Response

The leading zeros on data returned as a response to interrogation commands or variables and arrays can be removed by the use of the command, LZ. The default value for LZ is 0, meaning that the leading zeroes do not get printed out unless LZ1 command is entered.

Example - Using the LZ command

```
LZ0                         Disables the LZ function
MG_TI1                      Print input status of bank 1
```

```
0000000255.0000         Response from Interrogation Command
                        (With Leading Zeros)

LZ1                     Enables the LZ function
MG_TI1                  Print input status of bank 1
 255.0000               Response from Interrogation Command
                        (Without Leading Zeros)
```

Formatting Variables and Array Elements

The Variable Format (VF) command is used to format variables and array elements. The VF command is specified by:

VF m.n

where m is the number of digits to the left of the decimal point (0 thru 10), and n is the number of digits to the right of the decimal point (0 thru 4).

A negative sign for m specifies hexadecimal format. The default format for VF is VF 10.4

Hex values are returned preceded by a $ and in 2's complement.

```
:V1=10                Assign V1
:V1=                  Return V1
0000000010.0000       Default format
:VF2.2                Change format
:V1=                  Return V1
10.00                 New format
:VF-2.2               Specify hex format
:V1=                  Return V1
$0A.00                Hex value
:VF1                  Change format
:V1=                  Return V1
9                     Overflow
```

Local Formatting of Variables

VF command is a global format command that effect the format of all relevant returned values and variables. Variables may also be formatted locally. To format locally, use the command, {Fn.m} or {$n.m} following the variable name and the '=' symbol. F specifies decimal and $ specifies hexadecimal. n is the number of digits to the left of the decimal, and m is the number of digits to the right of the decimal. For example:

Examples:

```
:V1=10                Assign V1
:V1=                  Return V1
0000000010.0000       Default Format
:V1={F4.2}            Specify local format
0010.00               New format
:V1={$4.2}            Specify hex format
$000A.00              Hex value
```

:V1="ALPHA"	Assign string "ALPHA" to V1
:V1={S4}	Specify string format first 4 characters
ALPH	

The local format is also used with the MG command (see page 147).

Programmable I/O

As described earlier in chapter 4, the IOC-7007 has 7 I/O slots that can be occupied by a “mix and match” of digital input and output modules. The paragraphs below describe the commands that are used for I/O manipulation and interrogation. For information on the IOM modules and the numbering scheme for the I/O slots, see Chapter 4.

Digital Outputs

The most common method of changing the state of digital outputs is by using the set bit ‘SB’ and clear bit ‘CB’ commands. The following table shows an example of the SB and CB commands.

Instruction	Interpretation
SB2	Sets bit 2
CB1	Clears bit 1

The Output Bit (OB) instruction is useful for setting or clearing outputs depending on the value of a variable, array, input or expression. Any non-zero value results in a set bit.

Instruction	Interpretation
OB1,POS	Set Output 1 if the variable POS is non-zero. Clear Output 1 if POS equals 0.
OB2,@IN [1]	Set Output 2 if Input 1 is high. If Input 1 is low, clear Output 2.
OB3,@IN [1]&@IN [2]	Set Output 3 only if Input 1 and Input 2 are high.
OB2,COUNT [1]	Set Output 2 if element 1 in array COUNT is non-zero.

The output port can be set by specifying the OQ (Output Port) command. This instruction allows a single command to define the state of the entire IOM output module, where 2^0 is bit 0, 2^1 is bit 1 and so on. A 1 designates that the output is on.

For example:

Instruction	Interpretation
OQ1,6	Sets bits 1 and 2 (output 33 and 34) of slot 1 to high. All other bits on slot 1 are 0. ($2^1 + 2^2 = 6$)
OQ0,0	Clears all bits of slot 0 to zero
OQ0,7	Sets output bits 0, 1 and 2 to one ($2^0 + 2^1 + 2^2$); this is the same as giving the commands: SB0;SB1;SB2;CB3;CB4;CB5;CB6;CB7

The state of the digital outputs can be accessed with the @OUT[n] where n is the output number (Ex: MG@OUT[1] displays the state of output number 1).

Digital Inputs

The digital inputs are accessed by using the @IN[n] function or the TI n command. The @IN[n] function returns the logic level of a specified input, n, where 'n' is the input bit number. The TI n command gives the input status of an IOC slot, where 'n' is the slot number, 0 through 6. Also, the AI is a trippoint the pauses program execution until the specified combination of inputs is high or low.

Example – Using Inputs to control program flow

Instruction	Instruction
JP #A,@IN[1]=0	Jump to A if input 1 is low
MG@IN[2]	Display the state of input 2
AI 7&-6	Wait until input 7 is high and input 6 is low

Input Interrupt Function

The IOC-7007 provides an input interrupt function which causes the program to automatically execute the instructions following the #ININTn label, where n ranges from 0 to 7. This function is enabled using the II n,m,condition command, where n specifies the #ININTn subroutine to be executed when interrupt occurs. The m argument specifies the thread number in which the interrupt subroutine #ININTn is going to be executed. Note that this thread needs to be executing at the time of the interrupt, otherwise the #ININTn subroutine will not have any thread to run in and will be ignored (for more on the II command, refer to the command reference section on page 73). Condition is any number of inputs separated by the "&" operator. A *positive* input number means the IOC-7007 looks for that input to go *high* to satisfy the interrupt condition, and a *negative* number means *low*.

For example, II1,0,3&-5 sets up the conditions of input 3 going high and input 5 going low, for the interrupt to occur at #ININT1 in thread 0 (main).

The Return from Interrupt (RI) command is used to return from this subroutine to the place in the program where the interrupt had occurred. If it is desired to return to somewhere else in the program after the execution of the #ININTn subroutine, the Zero Stack (ZS) command is used, followed by unconditional jump statements.

IMPORTANT: Use the RI instruction (not EN) to return from the #ININTn subroutine.

For an example on input interrupt, look at the trippoint example on page 133.

Analog Inputs

Analog inputs are accessed with the @AN[n] function where n is the number assigned to the analog input channel. The returned value will be a voltage reading with 12 or 16 bit resolution. The standard voltage range is –10VDC to +10VDC, but other ranges are available on request.

Note: When @AN[n] is used in the #PLCSCAN thread, the returned value will be a decimal number that represents the analog voltage. For a 12 bit module, the equation used to determine the decimal equivalent of the analog voltage is as follows:

N= ((V-Vlo)*4095)/(Vhi-Vlo)

Where N is the decimal equivalent of the analog voltage, V is the expected analog voltage, Vlo is the lowest voltage in the total range (-10V for the standard analog input module) and Vhi is the highest voltage in the total range (10V for the standard module).

Likewise, for a 16 bit module, the equation is:

N = ((V-Vlo)*65535)/(Vhi-Vlo)

These decimal values will also be returned when accessing the analog inputs by the API calls in C/C++ or Visual Basic.

The AA command is a trippoint that halts program execution until the specified voltage on an analog input is reached. If the specified voltage is exceeded prior to arrival at the AA command, the program will continue to execute without a pause. Analog inputs are useful for reading special sensors such as temperature, tension or pressure.

Instruction	Instruction
JP #C,@AN[1]>2	Jump to A if analog input number 1 is greater than 2 volts
MG@AN[2]	Display the analog voltage reading on input 2
AA 1,4.5	Wait until the voltage on input 1 reaches 4.5

Analog Outputs

Analog output voltage is set with the AO command. The AO command has the format AO m,n where m is the output pin and n is the voltage assigned to it. The analog output voltage is accessed with the @AO[n] function where n is the analog output channel. Analog output modules are available in 12 bit and 16 bit versions. The standard voltage range is –10VDC to +10VDC, but other ranges are available on request.

Note: When @AO[n] is used in the #PLCSCAN thread, the returned value will be a decimal number that represents the analog voltage. Likewise, when AO is used in the #PLCSCAN thread, the number used for the m field will also be a decimal number. For a 12 bit module, the equation used to determine the decimal equivalent of the analog voltage is as follows:

N= ((V-Vlo)*4095)/(Vhi-Vlo)

Where N is the decimal equivalent of the analog voltage, V is the expected analog voltage, Vlo is the lowest voltage in the total range (-10V for the standard analog input module) and Vhi is the highest voltage in the total range (10V for the standard module).

Likewise, for a 16 bit module, the equation is:

N = ((V-Vlo)*65535)/(Vhi-Vlo)

These decimal values will also be returned when accessing the analog inputs by the API calls in C/C++ or Visual Basic.

The AO command can also be used to set the analog voltage on ModBus devices over Ethernet. See Chapter 5 for more details on the AO command.

Instruction	Instruction
AO 7,1.5	Set the output voltage on output 7 to 1.5V
MG@AO[2]	Display the analog voltage reading on output 2

THIS PAGE LEFT BLANK INTENTIONALLY

Appendix

Electrical Specifications

Input/Output

See Chapter 4 for details on the IOM modules

Power Requirements

20-60 VDC (IOC-7007 Card, Din or Box)	480 mA

DC requirements from 90-260 VAC 50/60 Hz

(IOC-7007-Box only)

+5 V	400 mA
+12 V	40 mA
-12 V	40 mA

Performance Specifications

Variable Range:	+/-2 billion
Variable Resolution:	$1 \cdot 10^{-4}$
Variable Size	126 variables
Array Size:	2000 elements, 14 array names
Program Size:	500 lines x 80 characters

Connectors on the IOC-7007

J9 RS-232 Port: DB-9 Pin Male

Standard connector and cable, 9Pin

Pin	Signal
1	CTS – OUTPUT
2	Transmit data-output
3	Receive data-input
4	RTS – input
5	Gnd
6	CTS – output
7	RTS – input
8	CTS – output
9	Nc

J11 Ethernet Port: 10/100 Base-T (RJ-45)

10/100 BASE- T - Kycon GS-NS-88-3.5

Pin	Signal
1	TXP
2	TXN
3	RXP
4	NC
5	NC
6	RXN
7	NC
8	NC

J8 Power: 4 pin Molex for 20-60VDC

Pin	Signal
1	E (Earth Ground)
2	NC (No Connection)
3	+24
4	G (Ground)

J10 Power: 6 pin Molex for 90-260VAC 50/60Hz Power supply (IOC-7007 Box)

Pin	Signal
1	+12
2	+5
3	+5
4	G (Ground)

5	G
6	-12

Note: If J8 is used to power the IOC, J10 will be occupied by a 250V, 0.8A fuse. If it's a IOC-7007-Box, then the power supply outputs connect to J10.

Cable Connections for IOC-7007

The IOC-7007 requires the transmit, receive, and ground for slow communication rates. (ie 1200 baud) For faster rates, the handshake lines are required. The connection tables below contain the handshake lines. These descriptions and tables are for RS-232 only.

Standard RS-232 Specifications

25 pin Serial Connector (Male, D-type)

This table describes the pin out for standard serial ports found on most computers.

Pin Number	Function
1	NC
2	Transmitted Data
3	Received Data
4	Request to Send
5	Clear to Send
6	Data Set Ready
7	Signal Ground
8	Carrier Detect, CTS
9	+Transmit Current Loop Return
10	NC
11	-Transmit Current Loop Data
12	NC
13	NC
14	NC
15	NC
16	NC
17	NC
18	+Receive Current Loop Data
19	NC
20	Data Terminal Ready, RTS
21	NC
22	Ring Indicator
23	NC
24	NC
25	-Receive Current Loop Return

9 Pin Serial Connector (Male, D-type)

Standard serial port connections found on most computers.

PIN NUMBER	FUNCTION
1	Carrier Detect, CTS
2	Receive Data
3	Transmit Data
4	Data Terminal Ready, RTS
5	Signal Ground
6	Data Set Ready
7	Request to Send
8	Clear to Send
9	Ring Indicator

IOC-7007 Serial Cable Specifications

Cable to Connect Computer 25 pin to Serial Port

25 Pin (Male - computer)	9 Pin (Female - controller)
8 (Carrier Detect, CTS)	1
3 (Receive Data)	2
2 (Transmit Data)	3
20 (Data Terminal Ready, RTS)	4
7 (Signal Ground)	5
Controller Ground	9

Cable to Connect Computer 9 pin to Serial Port Cable (9 pin)

9 Pin (Female - Computer)	9 Pin (Female - Controller)
1 (Carrier Detect, CTS)	1
2 (Receive Data)	2
3 (Transmit Data)	3
4 (Data Terminal Ready, RTS)	4
5 (Signal Ground)	5
Controller Ground	9

Jumper Description for IOC-7007

Jumper	Label	Function (If jumpered)
JP5	MRST	Master Reset enable. Returns IOC to factory default settings and erases EEPROM. Requires power-on or RESET to be activated.
	UPGD	Used to upgrade controller firmware when resident firmware is corrupt.
	1200	Baud rate jumper (1200 bits/sec)
	9600	Baud rate jumper (9600 bits/sec)
JP1	ADR1-ADR8	Address jumpers for configuration in Galil's Distributed Network
	10B	Jumper to restrict the Ethernet port to 10Base-T

Accessories and Options

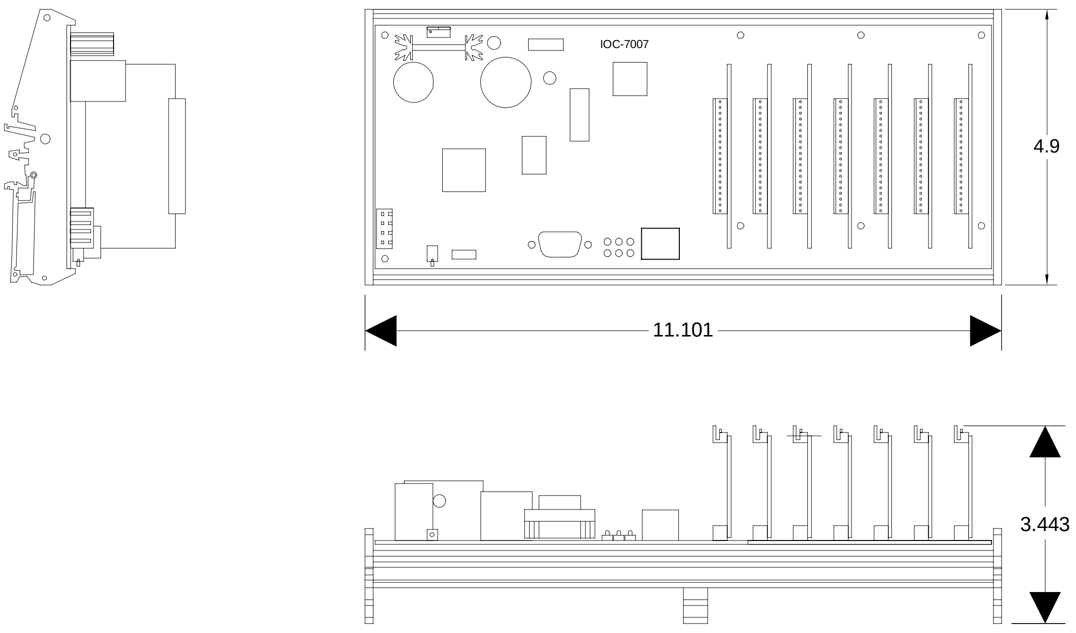
IOC-7007-card	Card level product available with 20-60 VDC power only
IOC-7007-box	IOC with metal enclosure. Available with 20-60 VDC power or 90-260 VAC 50/60Hz.
IOC-7007-DIN	DIN rail mountable IOC. Available with 20-60 VDC power only
Cable 9-Pin D	Straight through RS-232 cable for communication
IOM-70016	16 TTL input module
IOM-70108	8 opto-isolated input module
IOM-70208	8 opto-isolated output module
IOM-70308	8 high powered output module
IOM-70404	4 dry contact relay module
IOM-70808	8 analog input module with 12 bit resolution
IOM-70908	8 analog output module with 12 bit resolution
IOM-70904	4 analog output module with 12 bit resolution
Galil CD-ROM / Utilities. Includes the following: DMCWIN16 DMCWIN32 and DMCTERM	Windows 3.x Terminal

SETUP16	Windows 95/98/NT/2000/ME/XP Terminal
SETUP32	Setup Utility for Window 3.x
C KIT	Setup Utility for Windows 95/98/NT/2000/ME/XP
	C-Programmers Kit
ActiveX Tool Kit	Visual Basic^{TM} Tool Kit (includes VBXs and OCXs)

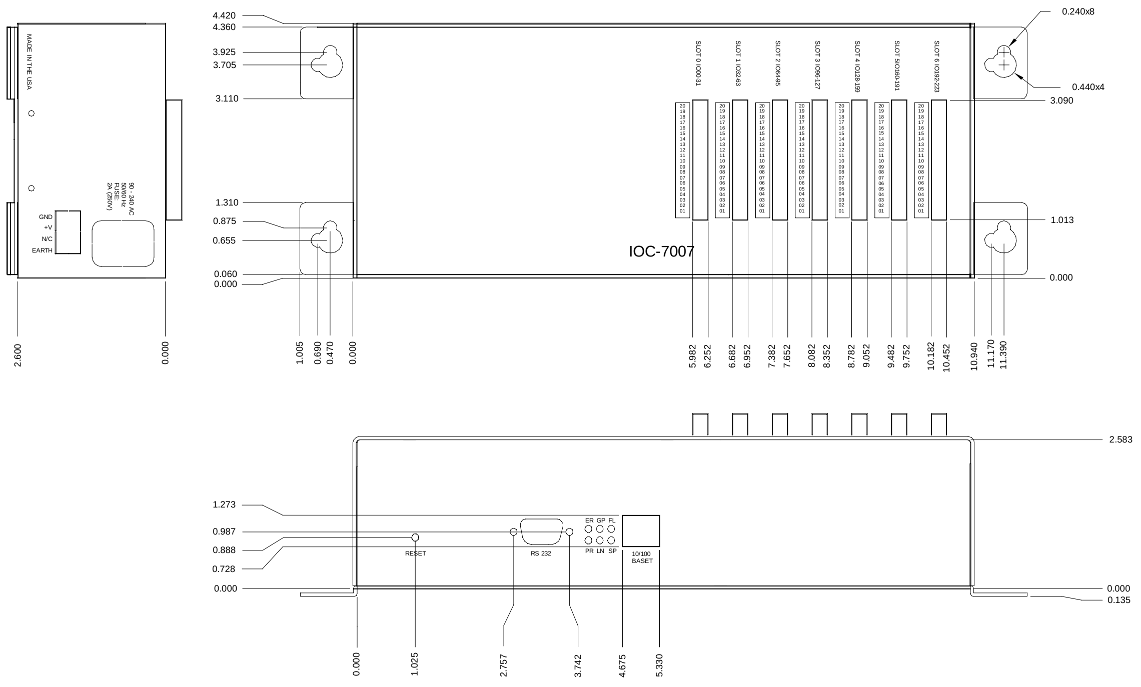
Note: The 16 bit software packages will only communicate to the IOC through the serial port. You will need to register it as a DMC-1412 for it to work.

IOC-7007 Dimensions

IOC-7007-Din Mounting Dimensions
Overall size: 11.1" x 4.9" x 3.4"



IOC-7007-Box Mounting Dimensions
Overall size: 11.8" x 4.4" x 2.6"



Example- Communicating with OPTO-22 SNAP-B3000-ENET

Assume that the IOC-7007 is connected to OPTO-22 via handle F. The OPTO-22's IP address is 131.29.50.30. The Rack has the following configuration:

Digital Inputs	Module 1
Digital Outputs	Module 2
Analog Outputs (+/-10V)	Module 3
Analog Inputs (+/-10V)	Module 4

Instruction	**Interpretation**
#CONFIG	Label
IHF=131,29,50,30<502>2	Establish connection
WT10	Wait 10 milliseconds
JP #CFGERR,_IHF2=0	Jump to subroutine
JS #CFGDOUT	Configure digital outputs
JS #CFGAOUT	Configure analog outputs
JS #CFGAIN	Configure analog inputs
MBF = 6,6,1025,1	Save configuration to OPTO-22
EN	End

```
#CFGDOUT                            Label
MODULE=2                            Set variable
CFGVALUE=$180                       Set variable
NUMOFIO=4                           Set variable
JP #CFGJOIN                         Jump to subroutine

#CFGAOUT                            Label
MODULE=3                            Set variable
CFGVALUE=$A7                        Set variable
NUMOFIO=2                           Set variable
JP #CFGJOIN                         Jump to subroutine

#CFGAIN                             Label
MODULE=4                            Set variable
CFGVALUE=12                         Set variable
NUMOFIO=2                           Set variable
JP#CFGJOIN                          Jump to subroutine

#CFGJOIN                            Label
 DA*[]                              Deallocate all arrays
 DM A[(2*NUMOFIO)]                  Dimension array
I=0                                 Set variable
#CFGLOOP                            Loop subroutine
A[I]=0                              Set array element
I=I+1                               Increment
A[I]=CFGVALUE                       Set array element
I=I+1                               Increment
JP #CFGLOOP,I<(2*NUMOFIO)           Conditional statement
MBF=6,16,632+(MODULE*8),NUMOFIO*    Configure I/O using Modbus function code 16 where
 2,A[]                               the starting register is 632+(MODULE*8), number of
                                     registers is NUMOFIO*2 and A[] contains the data.
EN                                  End

#CFERR                              Label
MG”UNABLE TO ESTABLISH              Message
 CONNECTION”
EN                                  End
```

Using the equation

I/O number = (Handlenum*1000) + ((Module-1)*4) + (Bitnum-1),

this means that the command, MG @IN[6001], will display the level of input at handle 6, module 1, bit 2. Similarly:

SB 6006 or OB 6006,1 → set bit of output at handle 6, module 2, bit 3 to one

AO 6008,3.6 → set analog output at handle 6, module 3, bit 1 to 3.6 volts

MG @AN[6013] → display voltage value of analog input at handle6, module 4, bit 2

List of Other Publications

"Step by Step Design of Motion Control Systems"

by Dr. Jacob Tal

"Motion Control Applications"

by Dr. Jacob Tal

"Motion Control by Microprocessors"

by Dr. Jacob Tal

Training Seminars

Galil, a leader in motion control with over 200,000 controllers working worldwide, has a proud reputation for anticipating and setting the trends in motion control. Galil understands your need to keep abreast with these trends in order to remain resourceful and competitive. Through a series of seminars and workshops held over the past 15 years, Galil has actively shared their market insights in a no-nonsense way for a world of engineers on the move. In fact, over 10,000 engineers have attended Galil seminars. The tradition continues with three different seminar, each designed for your particular skill set--from beginner to the most advanced.

MOTION CONTROL MADE EASY

WHO SHOULD ATTEND

Those who need a basic introduction or refresher on how to successfully implement servo motion control systems.

TIME: 4 hours (8:30 am-12:30 pm)

ADVANCED MOTION CONTROL

WHO SHOULD ATTEND

Those who consider themselves a "servo specialist" and require an in-depth knowledge of motion control systems to ensure outstanding controller performance. Also, prior completion of "Motion Control Made Easy" or equivalent is required. Analysis and design tools as well as several design examples will be provided.

TIME: 8 hours (8:00 am-5:00 pm)

PRODUCT WORKSHOP

WHO SHOULD ATTEND

Current users of Galil motion controllers. Conducted at Galil's headquarters in Rocklin, CA, students will gain detailed understanding about connecting systems elements, system tuning and motion programming. This is a "hands-on" seminar and students can test their application on actual hardware and review it with Galil specialists.

TIME: Two days (8:30 am-5:00 pm)

Contacting Us

Galil Motion Control

3750 Atherton Road

Rocklin, CA 95765

Phone: 916-626-0101

Fax: 916-626-0102

E-Mail Address: support@galilmc.com

URL: www.galilmc.com

FTP: www.galilmc.com/ftp

WARRANTY

All products manufactured by Galil Motion Control are warranted against defects in materials and workmanship. The warranty period for controller boards is 1 year. The warranty period for all other products is 180 days.

In the event of any defects in materials or workmanship, Galil Motion Control will, at its sole option, repair or replace the defective product covered by this warranty without charge. To obtain warranty service, the defective product must be returned within 30 days of the expiration of the applicable warranty period to Galil Motion Control, properly packaged and with transportation and insurance prepaid. We will reship at our expense only to destinations in the United States.

Any defect in materials or workmanship determined by Galil Motion Control to be attributable to customer alteration, modification, negligence or misuse is not covered by this warranty.

EXCEPT AS SET FORTH ABOVE, GALIL MOTION CONTROL WILL MAKE NO WARRANTIES EITHER EXPRESSED OR IMPLIED, WITH RESPECT TO SUCH PRODUCTS, AND SHALL NOT BE LIABLE OR RESPONSIBLE FOR ANY INCIDENTAL OR CONSEQUENTIAL DAMAGES.

THIS PAGE LEFT BLANK INTENTIONALLY

Index