MISTICWARE MANUAL

Form 522-100823 — August 2010

OPTO 22

43044 Business Park Drive, Temecula, CA 92590-3614
Phone: 800-321-OPTO (6786) or 951-695-3000
Fax: 800-832-OPTO (6786) or 951-695-2712
www.opto22.com

Product Support Services:
800-TEK-OPTO (835-6786) or 951-695-3080
Fax: 951-695-3017
E-mail: support@opto22.com
Web: support.opto22.com

MisticWare Manual
Form 522-100823—August 2010

The information in this manual has been checked carefully and is believed to be accurate; however, Opto 22 assumes no responsibility for possible inaccuracies or omissions. Specifications are subject to change without notice.

Opto 22 warrants all of its products to be free from defects in material or workmanship for 30 months from the manufacturing date code. This warranty is limited to the original cost of the unit only and does not cover installation, labor, or any other contingent costs. Opto 22 I/O modules and solid-state relays with date codes of 1/96 or later are guaranteed for life. This lifetime warranty excludes reed relay, SNAP serial communication modules, SNAP PID modules, and modules that contain mechanical contacts or switches. Opto 22 does not warrant any product, components, or parts not manufactured by Opto 22; for these items, the warranty from the original manufacturer applies. These products include, but are not limited to, the OptoTerminal-G70, OptoTerminal-G75, and Sony Ericsson GT-48; see the product data sheet for specific warranty information. Refer to Opto 22 form number 1042 for complete warranty information.

Opto 22 FactoryFloor, Cyrano, Optomux, and Pamux are registered trademarks of Opto 22. Generation 4, ioControl, ioDisplay, ioManager, ioProject, ioUtilities, mistic, Nvio, Nvio.net Web Portal, OptoConnect, OptoControl, OptoDisplay, OptoENETSniff, OptoOPCServer, OptoScript, OptoServer, OptoTerminal, OptoUtilities, SNAP Ethernet I/O, SNAP I/O, SNAP OEM I/O, SNAP Simple I/O, SNAP Ultimate I/O, and SNAP Wireless LAN I/O are trademarks of Opto 22.

ActiveX, JScript, Microsoft, MS-DOS, VBScript, Visual Basic, Visual C++, and Windows are either registered trademarks or trademarks of Microsoft Corporation in the United States and other countries. Linux is a registered trademark of Linus Torvalds. Unicenter is a registered trademark of Computer Associates International, Inc. ARCNET is a registered trademark of Datapoint Corporation. Modbus is a registered trademark of Schneider Electric. Wiegand is a registered trademark of Sensor Engineering Corporation. Nokia, Nokia M2M Platform, Nokia M2M Gateway Software, and Nokia 31 GSM Connectivity Terminal are trademarks or registered trademarks of Nokia Corporation. Sony is a trademark of Sony Corporation. Ericsson is a trademark of Telefonaktiebolaget LM Ericsson.

All other brand or product names are trademarks or registered trademarks of their respective companies or organizations.

Table of Contents

Welcome

Overview

The MisticWare I/O driver is a software package designed to simplify communications with Mistic protocol I/O units.

The MisticWare driver provides the software interface between the Mistic I/O units and an application program written in a high-level language. The driver is a function call. Examples are provided to demonstrate how to use the driver from various languages.

The MisticWare driver was written for the IBM-PC family computers. Complete source code is included for your convenience.

The MisticWare I/O driver performs the following functions:

- Builds and transmits Mistic protocol I/O command messages
- Carries out all necessary handshaking and communications
- Converts the data returned by a Mistic I/O unit into a form that is easily manipulated by the application program
- Performs extensive error checking and returns diagnostic error codes.

This manual outlines the use and command structure of the driver.

For a complete description of each low-level Mistic protocol I/O command, please refer to Opto 22's *Mistic Protocol Guide* (Form 270).

This manual assumes you have read and understand Form 270.

To use the MisticWare I/O driver in an application program, you must know the following:

- How to call a "C" function from an application program
- How to use a linker to link a program
- How to tell the driver what command to send by assigning values to the proper parameters
- How to interpret the data passed back by the driver.

What's In This Manual?

This manual includes the following sections:

- **Chapter 1: "Getting Started"**—installing and using the driver.
- **Chapter 2: "MisticWare Commands"**—index listing all of the MisticWare commands and their page members.
- **Chapter 3: "Analog and Digital I/O Commands"**—commands common to both analog and digital I/O units or "brain boards".
- **Chapter 4: "Multichannel Digital Commands"**—commands used to access banks or groups of digital channels.
- **Chapter 5: "Single Channel Digital Commands"**—commands used to access individual digital channels.
- **Chapter 6: "Multichannel Analog Commands"**—commands used to access groups of analog channels.
- **Chapter 7: "Single Channel Analog Commands"**—commands used to access individual analog channels.
- **Chapter 8: "Analog PID Commands"**—commands used to access PIDs on analog units.
- **Chapter 9: "Digital Event/Reaction Commands"**—commands for configuring event reactions on digital units.
- **Chapter 10: "Analog Event/Reaction Commands"**—commands for configuring event reactions on analog units.
- **Chapter 11: "Driver Commands"**—driver configuration commands.
- **Appendix A: "Channel Data Dump Structures"**—supplemental information for command 452.
- **Appendix B: "MisticWare Driver for Windows"**—information for use with 16-bit Windows.
- **Appendix C: "Using the Driver with Microsoft Basic/Quick Basic"**—information for use with Basic on DOS.
- **Appendix D: "Using Opto in 32-Bit Windows"**—describes how to access Opto I/O in Win 32.

Document Conventions

- **Bold** typeface indicates text to be typed. Unless otherwise noted, such text may be entered in upper or lower case. (Example: “At the DOS prompt, type **cd\windows**.”)
- *Italic* typeface indicates emphasis and is used for book titles. (Example: “See the *OptoControl User’s Guide* for details.”)
- File names appear in all capital letters. (Example: “Open the file TEST1.TXT.”)
- Key names appear in small capital letters. (Example: “Press SHIFT.”)
- Key press combinations are indicated by hyphens between two or more key names. For example, SHIFT F1 is the result of holding down the SHIFT key, then pressing and releasing the F1 key. Similarly, CTRL-ALT-DELETE is the result of pressing and holding the CTRL and ALT keys, then pressing and releasing the DELETE key.
- “Press” (or “click”) means press and release when used in reference to a mouse button.
- Menu commands are sometimes referred to with the Menu→Command convention. For example, “Select File→Run” means to select the Run command from the File menu.
- Numbered lists indicate procedures to be followed sequentially. Bulleted lists (such as this one) provide general information.

CHAPTER 1

Getting Started

Before proceeding with this section, please make backup copies of the driver diskettes. Refer to your DOS manual for instructions on copying diskettes.

Using The Driver

The MisticWare I/O driver is a "C" function. Two structures are used for passing parameter information. The SEND structure passes data to the driver, and the RESP structure passes data back. An application program must declare variables of these types, then pass a pointer to those variables when calling the driver. The function then returns an integer which corresponds to an error value. The value is zero if no error has occurred, otherwise it is a negative number representing an error code.

```
Function: int g4driver(struct SEND_OBJ *send, struct RESP_OBJ name
*response)

Inputs:          *send points to a structure of parameters to send

                 struct SEND_OBJ
                 {
                        unsigned int address;     /*unit address */
                        unsigned int command;     /*command number */
                        unsigned int position[2]; /*position masks */
                        long         data[16];    /*data array */
                 };

Outputs: *response points to a structure of where to put the data

                 struct RESP_OBJ
                 {
                        int          error;       /*return error */
                        long         data[16];    /*return data */
                 };

Return:          0 returned if no error, else error is returned.
```

Driver Parameters

Send Structure

Address

This field contains the address of the I/O unit. The field is an integer value ranging from 0 to 255 decimal.

Command

This parameter is an integer variable that contains the number of the desired driver command.

Positions[0],Positions[1]

The POSITIONS array contains two integer parameters. For many of the multichannel commands, a positions mask (0000 to FFFF Hex) is placed in the POSITIONS array element specified by the command. For the single channel commands, a channel number (0 to 15 Decimal) is used. The majority of multichannel commands only use POSITIONS[0], although some commands use both POSITIONS[0] and POSITIONS[1].

Data[0]...Data[15]

The DATA array parameter is made up of 16 long integer (4-byte) values. These are used to hold the data values necessary to execute the command.

Response Structure

Error

The ERROR parameter is an integer field which will contain a value on return from the driver. If no error has occurred, a value of zero (0) will be returned, otherwise a specific negative error code is returned. For a list of error codes, refer to the section, “Mistic Driver Error Codes.” An application program should always inspect this parameter after calling the driver to verify that communications were successful and that the data was received correctly.

Data[0]...Data[15]

The DATA array parameter is made up of 16 long integer (4-byte) values. These are used to hold the data values that are returned by the I/O unit.

Examples

Several examples are provided on the disk to illustrate the use of the driver. The examples attempt to cover the wide range of features of both the digital and the analog I/O units. The examples start with simple initialization and error handling and progress to more advanced examples of using the event-reaction and PID commands.

Error Handling

See the DOS example EXAMPLE2.C, which shows one method of processing errors that are detected by the I/O unit or the driver. This example uses the routine for driver initialization shown in the previous example, and also shows a typical method of initializing the I/O units.

The important concept to remember in error handling is to re-configure the I/O units after a power-up or watchdog condition. The extent of the re-configuration depends on what features were used and whether the configurations were stored in EEPROM.

Using The Driver With Microsoft C on DOS

To use the Microsoft C version of the driver, you will need to change the define statement MSC to a true (1) in the file G4DRIVER.H before compilation. This statement changes certain function names to those recognized by the Microsoft C compiler. A simple example called G4MSC.C is provided on the disk to show how to interface to the driver. The make file for the project is also provided and is called GMSC.MAK.

Using The Driver With Other Languages on DOS

Previous versions of the MisticWare driver provided a method for using the driver with the Microsoft BASIC compiler. This method, however, required the existence of the Microsoft “C” libraries during the linking process. In order to simplify the interface to the BASIC language (as well as other languages such as Turbo PASCAL), an alternate method is now used.

A program named LOADMD.EXE is provided and allows the MisticWare driver to function as a Terminate-Stay-Resident (TSR) program. The program is written in “C” and the source code is contained in the file LOADMD.C. Running this program prior to running the application loads the driver into memory where it becomes accessible using software interrupt 61 hex. Prior to an application issuing the interrupt, the CPU AX, BX, CX, and DX registers are setup with the addresses to the send and receive structures used by the application.

The language used must be able to support the following; structures (records) that contain the driver parameters, a method of deriving segment and offset addresses of those structures, a method of setting the CPU registers and issuing a software interrupt.

Sample programs are provided on the disk which have been tested with Microsoft QuickBASIC 4.0/4.5, the Microsoft BASIC Compiler 7.1, and Turbo PASCAL 4.0-6.0. Appendices in the back of this manual explain the details specific to each language.

If changes are made to the driver source code, LOADMD.EXE will need to be re-created by linking LOADMD.OBJ, G4DRIVER.OBJ, and G4COMM.OBJ together.

Mistic Driver Error Codes

Mistic I/O unit Generated Errors

-1 Undefined Command Error

This error code indicates that the Mistic I/O unit received a command letter that it did not understand. This may indicate that a new command that was added to the driver is not supported by an older I/O unit or that a command for one type of I/O unit was actually sent to an I/O unit of a different type (analog command sent to a digital board).

-2 Mistic Protocol I/O Unit Detected A Checksum Error

This error code indicates a problem with the communications link that caused a checksum error when a message from the host was sent to the I/O unit. Checksum errors often occur when the Local Bus or RS-485 network is not wired, terminated, or biased properly. Make sure of the following:

1. Twisted-pair cable with at least two twists per inch is used for remote I/O units.
2. The link is routed in a daisy-chain fashion, NOT a "star" type distribution.
3. The link is terminated at the end points and NOT in the middle.
4. The length of the cable does not exceed the length defined in the I/O unit specifications.

-3 Mistic I/O Unit's Buffer Was Overrun

This error indicates that the I/O unit received more characters than were expected. This may occur because of improper biasing of remote links combined with an electrically noisy environment. See recommendations described for a -2 error.

-4 Device Lost Power Since Last Message

This error indicates that a power failure occurred and the I/O unit went through a reset cycle. Configuration parameters may have been lost unless they were stored in the I/O units' EEPROM. This error can only be cleared by issuing a "Power Up Clear" (0) command. Remember to re-configure the unit and download PID parameters and Event/Reaction commands where necessary.

-5 Error In A Data Field

This error occurs when the I/O unit receives a message that does not contain enough data characters for that command. This may be due to sending a digital command to an analog I/O unit or vice-versa. It also may indicate a driver version that does not match the firmware of an I/O unit because of a change to the command structure.

-6 A Serial Watchdog Occurred Since Last Message

This is a flag that indicates that the I/O unit went to a watchdog state because of inactivity on the communications link (local or remote) for a specified length of time. The command that was sent which received this response is not executed. Since the watchdog state may be different than what the host had previously setup, it may be necessary to rewrite those outputs. If you receive this error often, you may wish to change the timeout period using the watchdog command. Under normal operations, this error should only occur after a recovery from a disconnection to the host or the communications link.

-7 Invalid Data-Limits Sent Are Out Of Range

This error indicates that at least one of the data fields in the command message contains an illegal value. Make sure that the values sent are within the range allowed by the configuration for that channel or loop.

-8 Reserved

-9 Invalid Module Type Was Specified

This error occurs when a command is sent that requires a channel to be configured differently than it is presently. For example; a command to turn on an output on a channel that is configured as an input.

-10 Invalid Event Type Was Specified

This error occurs when an attempt is made to enable an event entry or define a reaction before the event has been defined (NULL entry).

-11 Invalid Delay Specified

This error occurs with digital I/O units when an attempt is made to start a square wave or generate N pulses, or a TPO, with a delay time less than 10 mS on more than eight output positions.

I/O Driver Generated Errors

-20 Invalid Command Error

This error occurs when the command variable contains a value that is not supported by the driver.

-22 Data Range Error

This error indicates that an invalid data value was used with a 900 series driver command. Examples include invalid port numbers, baud rates, and timeout ranges.

-25 Invalid Address Error

This error occurs when the address variable contains a value that is less than 0 or greater than 255.

-29 Driver Timeout Error

This error indicates that a command was sent to an I/O unit and a response was NOT received within the delay period used by the driver. A -29 error may occur because of the following:

1. The address variable contains an address of an I/O unit that doesn't exist. Check variable and verify address jumpers on I/O unit.
2. No power or bad power to the I/O unit.
3. A problem with the communications cables. Check continuity and polarity. Make sure you use correct cable types with the proper termination and bias.
4. If both the transmit and receive LEDs flash on the I/O unit, then check for the following: Incorrect communications interrupt line selected or another adapter card using the same address/interrupt line.
5. Turn-Around Delay may be set too short for the baud rate and command selected. A Reset command (1) and a "Store Configuration To EEPROM" command (3) may take up to two seconds to execute.

-31 Driver Detected A Checksum Error In The Response

This error code indicates that a problem with the communications link caused a checksum error when a response from the I/O unit was received. See recommendations described for a -2 error.

-33 Driver Send Error

The driver had a problem transmitting a message. This may occur when there is a problem with the serial or local adpater card. Check the address and interrupt line for the adapter card and make sure it does not conflict with any other adapter in the PC.

-36 General Mistic Controller Error

This error code indicates that the Controller received a message it did not understand. Make sure that you are in pass-thru mode when sending commands to the I/O units through the Mistic Controller.

Mistic Controller PASS-THRU Errors

-41 Bad Port Number

The Mistic Controller reports that the specified port number is illegal in the PASS-THRU preamble. This error should never occur, because driver command 904 checks for valid numbers.

-42 Mistic Controller Detected A CRC Error

This error code indicates that a problem with the communications link caused a CRC error when a message from the host was sent to the Controller. Checksum errors often occur when the RS-485 network is not wired, terminated, or biased properly. Make sure of the following:

1. Twisted-pair cable with at least two twists per inch is used for Mistic Controller RS-485 serial ports.
2. The link is routed in a daisy-chain fashion, NOT a "star" type distribution.
3. The link is terminated at the endpoints and NOT in the middle.
4. The length of the cable does not exceed the length defined in the G4LC specifications.

-43 Mistic Controller Buffer Overrun-From I/O unit

This error indicates that the Mistic Controller received more characters from the I/O unit than were expected. This may occur because of improper biasing of remote links combined with an electrically noisy environment. See recommendations described for a -42 error.

-44 Mistic Controller Lost Power Since Last Message

This error indicates that a power failure occurred and the Mistic Controller went through a reset cycle. This error can only be cleared by issuing a "Power Up Clear" (0) command to the Mistic Controller. **Remember to send this command in the binary mode, NOT in the PASS-THRU mode.**

-45 Bad CRC Length Error

This error indicates that the Mistic Controller received more or less characters than were expected when doing the CRC calculations. This may occur because of improper jumper settings for mode.

CDR/checksum on the Mistic I/O units. Verify that mode 2 (Command 904) is used with multifunction I/O units and mode 3 is used with local, simple I/O units.

-46 Mistic Controller Buffer Overrun - From Host Computer

This error indicates that the Mistic Controller received more characters from the host than were expected. This may occur because of improper biasing of remote links combined with an electrically noisy environment. See recommendations described for a -42 error.

-47 Reserved

-48 Bus Error Occurred in Mistic Controller

This error indicates that a BUS ERROR condition occurred in the Mistic controller. It is usually caused by a failure in the Mistic controller's hardware. The failure may be due to a bad memory or port device. The error flag can only be cleared by issuing a "Bus Error Clear" (7) command to the Mistic Controller. **Remember to send this command in the binary mode, NOT in the PASS-THRU mode.** This error is a warning that there is something wrong with the Mistic Controller, and clearing the bus error flag will not solve the problem. If it occurs, make sure that all the cards in the controller are properly seated.

Additional Error Codes

-100 Invalid Port Error

This error occurs when a handle number for a port is specified and that port has not been assigned a driver or adapter type.

-102 Initialize Error

This error occurs if the proper AC37 or AC39 adapter could not be found at the location specified in the ConfigureAC37 or ConfigureAC39 functions.

-104 Port Lock Error

This error occurs when a call to the SendMIO() function is made while another command is being processed. The DLL allows multiple applications to share a port. If a port is in use and is accessed by a second application, the second application will wait a limited amount of time for the port to become available. If, after waiting, the port is still unavailable this error is returned.

-105 Driver Configuration Error

This error occurs when the SendMIO() function is called prior to configuring a port with the AssignPortDriver() and Configure functions.

For other errors, see Error.H in Win 16 or OptoErr.RH in Win 32

MisticWare Commands

Mistic Driver Commands Sample

SAMPLE COMMAND (LETTER REFERS TO MISTIC PROTOCOL GUIDE) COMMAND

DESCRIPTION:

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	#
POSITION[0]:	Bitmask or Channel Number
POSITION[1]:	Bitmask or Channel Number
DATA[0]...DATA[15]:	Send Data Fields

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error
DATA[0]...DATA[15]:	Return Data Fields

CHAPTER 3

Analog And Digital I/O Commands

POWER UP CLEAR (A) 0

DESCRIPTION:

Prevents I/O Unit from returning a”Power Up Clear expected“ error message in response to instructions following application of power or a RESET command.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	0

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

RESET (B) 1

DESCRIPTION:

This command forces a hardware reset. The I/O unit is restored to the configuration stored in EEPROM. The factory default is used if nothing was previously saved
to EEPROM.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	1

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

SET SYSTEM OPTIONS (C) 2

DESCRIPTION:

This command is used to set or clear the bits in the Option Control Byte (OCB) of the I/O unit.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	2
DATA[0]:	Byte indicating bits to set (0-255)
DATA[1]:	Byte indicating bits to clear (0-255)

The system options and controlling bits are:

bit 0 : Frequency Range; 0 = 1 sec., 1 = .1 sec (Digital)
bit 0 : 0 = Degrees C, 1 = Degrees F (Analog)
bit 1 : Not Used
bit 2 : Not Used
bit 3 : Not Used
bit 4 : CRC init value; 0 = 0000, 1 = FFFF
bit 5 : CRC Method; 0 = reverse, 1 = classical
bit 6 : CRC Polynomial; 0 = CRC16, 1 = CCITT
bit 7 : Global Event Interrupt Enable; 0 = disabled, 1 = enabled

Factory default is 00

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error
DATA[0]:	OCB Value (0-255)

STORE CONFIGURATION TO EEPROM (E) 3

DESCRIPTION:

This command saves the current system parameters to EEPROM.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	3

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

IDENTIFY I/O UNIT TYPE (F) 4

DESCRIPTION:

This command causes the I/O unit to send back a response that identifies the type of I/O Unit.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	4

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error
DATA[0]:	16 = G4D16R — 16-channel remote multifunction digital I/O unit 17 = G4D32RS — 16-channel remote simple digital I/O unit 18 = G4A8R w/G4RAX — B200 16-channel remote analog I/O unit 19 = G4A8R — 8-channel remote analog I/O unit 22 = G4HDAR (no interface card) 23 = G4HDAR w/G4AITM 24 = G4HDAR w/G4AIVA 25 = G4HDAR w/G4AIRTD 26 = G4HDAR w/G4AOV 27 = G4HDAR w/G4AOA 32 = G4D16L — 16-channel local multifunction digital I/O unit 33 = G4D16LS — 16-channel local simple digital I/O unit 34 = G4A8L w/G4LAX — 16-channel local analog I/O unit 35 = G4A8L — 8-channel local analog I/O unit 38 = G4HDAL (no interface card) 39 = G4HDAL w/G4AITM 40 = G4HDAL w/G4AIVA 41 = G4HDAL w/G4AIRTD 42 = G4HDAL w/G4AOV 43 = G4HDAL w/G4AOA 72 = B3000 (digital address) multifunction I/O unit 80 = B3000 (analog address) multifunction I/O unit 288 = G4D16L (M4RTU local I/O unit) 291 = G4A8L (M4RTU local I/O unit)

REPEAT LAST RESPONSE (^) 5

DESCRIPTION:

This command causes the I/O Unit to repeat the response to the previous command.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	5

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

ROM REVISION/DATE (') 6

DESCRIPTION:

This command returns the revision level and date of the I/O Unit's EPROM Firmware. The values returned must be converted to hex to have meaning.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	6

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error
DATA[0]:	Low Byte of Checksum (Hex)
DATA[1]:	High Byte of Checksum (Hex)
DATA[2]:	Year (Hex)
DATA[3]:	Day (Hex)
DATA[4]:	Month (Hex)
DATA[5]:	Revision Level Minor (Hex)
DATA[6]:	Revision Level Major (Hex)

CLEAR BUS ERROR FLAG MISTIC CONTROLLER (C) 7

DESCRIPTION:

Clears an existing BUS ERROR flag which will stop the Mistic controller from sending a BUS ERROR CLEAR EXPECTED ERROR (-48) message in response to instructions following a bus error occurrence. Make sure this instruction is sent directly to the Mistic Controller (in BINARY mode) and NOT while in PASS-THRU mode.

SEND PARAMETERS:

ADDRESS:	Address of Mistic Controller
COMMAND:	7

RECEIVE PARAMETERS:

ERROR:	Driver or Mistic Controller Error

SET RESPONSE DELAY (~) 8

DESCRIPTION:

This command is used to set a delay time for command responses.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	8
DATA[0]:	Delay Value in units of 10 mSec. Range 8-bit value (0-255).

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

CLEAR INTERRUPT REQUEST (ZB) 9

DESCRIPTION:

This command clears the interrupt line if the IRQ line is active.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	9

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

CHAPTER 4

Multichannel Digital Commands

SET I/O CONFIGURATION (G) 100

DESCRIPTION:

This command configures the channels on an I/O unit.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	100
POSITION[0]:	Group Mask (0000 to FFFF Hex)
DATA[0]...DATA[15]:	Configure Type. If bit n is set in *POSITION[0]:*, type must be in Data[n]. eg. *POSITION[0]:* = 32, Data[6] = Type.
Types	Counter (default input) = 0 Positive Pulse Measurement = 1 Negative Pulse Measurement = 2 Period Measurement = 3 Frequency Measurement = 4 Quadrature Counter Input = 5 On Time Totalizer Input = 6 Off Time Totalizer Input = 7 Output = 128

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

READ I/O CONFIGURATION (Y) 101

DESCRIPTION:

This command reads the I/O configuration of the I/O unit.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	101

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

DATA[0]...DATA[15]:	Configuration type.
	Data[0] = type for channel 0.
	Data[15] = type for channel 15.

Types	0 = Counter (default) 1 = Positive Pulse Measurement 2 = Negative Pulse Measurement 3 = Period Measurement 4 = Frequency Measurement 5 = Quadrature Counter Input 6 = On Time Totalizer Input 7 = Off Time Totalizer Input 128 = Output

READ I/O STATUS (R) 102

DESCRIPTION:

This command reads the status of all the digital channels (input or output).

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	102

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

DATA[0]...DATA[15]:	0 if channel is in OFF state.
	1 if channel is in ON state.
	Data[0] = status of channel 0.
	Data[15] = status of channel 15.

SET OUTPUTS (J) 103

DESCRIPTION:

This command sets a group of output channels on an I/O Unit to an ON or an OFF state.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	103
POSITION[0]:	Bitmask of Modules to turn ON.
POSITION[1]:	Bitmask of Modules to turn OFF. Bitmask range is 0000 to FFFF Hex.

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Unit Error

READ LATCHES WITHOUT CLEARING (S) 104

DESCRIPTION:

This command returns the state of the positive and negative latches but does not clear the latches.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	104

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Unit Error
DATA[0]:	Bitmask of positive latches; bit set indicates latch set
DATA[1]:	Bitmask of negative latches; bit set indicates latch set Bitmask range is 0000 to FFFF Hex.

ENABLE COUNTERS (H) 105

DESCRIPTION:

This command enables a group of counters (standard or quadrature) on an I/O Unit.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	105
POSITION[0]:	Bitmask of counters to enable. A bit set enables that channel's counter. Bitmask range is 0000 to FFFF Hex.

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Unit Error

DISABLE COUNTERS (H) 106

DESCRIPTION:

This command disables a group of counters (standard or quadrature) on an I/O Unit.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	106
POSITION[0]:	Bitmask of counters to disable. A bit set disables that channel's counter. Bitmask range is 0000 to FFFF Hex.

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Unit Error

READ COUNTER ENABLE/DISABLE STATUS (u) 107

DESCRIPTION:

This command reads the enable/disable status of counters (standard or quadrature) on an I/O Unit.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	107

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Unit Error

DATA[0]...DATA[15]:	0 if counter is disabled, 1 if counter is enabled.
	Data[0] = status of counter 0.
	Data[15] = status of counter 15.

READ COUNTERS WITHOUT CLEARING (T) 108

DESCRIPTION:

This command reads a group of counters (standard or quadrature) on an I/O Unit. The counter values are not cleared.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	108
POSITION[0]:	Bitmask of counters to read. A bit set reads that channel's counter. Bitmask range is 0000 to FFFF Hex.

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Unit Error

DATA[0]...DATA[15]:	Counter Values. Range is 0 to 4,294,967,295 counts.
	Data[0] = Value of counter 0.
	Data[15] = Value of counter 15.

READ AND CLEAR COUNTERS (U) 109

DESCRIPTION:

This command reads a group of counters (standard or quadrature) on an I/O Unit. The counter values are cleared after they are read.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	109
POSITION[0]:	Bitmask of counters to read. A bit set reads and clears that channel's counter. Bitmask range is 0000 to FFFF Hex.

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Unit Error

DATA[0]...DATA[15]:	Counter Values.
	Range is 0 to 4,294,967,295 counts.
	Data[0] = Value of counter 0.
	Data[15] = Value of counter 15.

READ PULSE/PERIOD/TOTALIZER COMPLETE STATUS (V) 110

DESCRIPTION:

This command reads the status of channels to indicate the channels which have measured one complete pulse, period, or totalizer.

SEND PARAMETERS:

COMMAND:	110

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Unit Error
DATA[0]:	Status Mask. Range*:* 0000 to FFFF Hex. A bit set in the status mask indicates that channel's measurement is complete.

READ PULSE/PERIOD/TOTALIZER (W) 111

DESCRIPTION:

This command reads the pulse or period measurement values from channels that have been configured as pulse, period, or totalizer measurement inputs.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	111
POSITION[0]:	Bitmask of pulse/period channels to read. A bit set reads that channel's measurement. Bitmask range is 0000 to FFFF Hex.

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Unit Error
DATA[0]...DATA[15]:	Pulse/Period Measurement Values. Range is 0 to 4,294,967,295 units. Units are 100 microseconds each. Data[0] = Value of channel 0. Data[15] = Value of channel 15.

READ AND RESET PULSE/PERIOD/TOTALIZER (X) 112

DESCRIPTION:

This command reads the pulse or period measurement values from channels that have been configured as pulse, period, or totalizer measurement inputs. The channels are then reset to 0 on the I/O Unit.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	112
POSITION[0]: :	Bitmask of pulse/period channels to read and reset. A bit set reads that channel's measurement. Bitmask range is 0000 to FFFF Hex.

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Unit Error
DATA[0]...DATA[15]:	Pulse/Period Measurement Values. Range is 0 to 4,294,967,295 units. Units are 100 microseconds each. Data[0] = Value of channel 0. Data[15] = Value of channel 15.

READ FREQUENCY (Z) 113

DESCRIPTION:

This command reads the frequency values from channels that have been configured for frequency measurement.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	113
POSITION[0]:	Bitmask of frequency channels to read. A bit set reads that channel's measurement. Bitmask range is 0000 to FFFF Hex.

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Unit Error
DATA[0]...DATA[15]:	Frequency Measurement Values. Range is 0 to 65535 units, 1 or 10 Hz units depending on the SystemOptions Byte. Maximum frequency input is 25KHz. Data[0] = Value of channel 0. Data[15] = Value of channel 15.

SET WATCHDOG (D) 114

DESCRIPTION:

This command sets a group of output channels on an I/O Unit to go to an ON or an OFF state when a watchdog condition occurs. A watchdog condition occurs when a message is not received within the specified delay time.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	114
POSITION[0]:	Bitmask of Modules to turn ON.
POSITION[1]:	Bitmask of Modules to turn OFF. Bitmask range is 0000 to FFFF Hex.
DATA[0]:	Delay time in 10ms units. Range is 0 to 65535 units.

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Unit Error

READ EVENT LATCHES (P) 115

DESCRIPTION:

This command reads the event latches of a group of 16 events out of a total of 256. Latches are set every time an event changes from a non-matching condition to a matching condition.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	115
POSITION[0]:	A group number ranging from 0 to 240 (F0 Hex) in multiples of 16.

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Unit Error
DATA[0]:	Status Mask. Range*:* 0000 to FFFF Hex. A bit set in the bitmask indicates that event latch is set.

READ AND CLEAR EVENT LATCHES (Q) 116

DESCRIPTION:

This command reads and then clears the event latches of a group of 16 events out of a total of 256. Latches are set every time an event changes from a non-matching condition to a matching condition.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	116
POSITION[0]:	A group number in the range of 0 to 240 (F0 Hex) in multiples of 16.

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Unit Error
DATA[0]:	Status Mask. Range*:* 0000 to FFFF Hex. A bit set in the bitmask indicates that event latch was set.

READ AND CLEAR POSITIVE LATCHES (S) 117

DESCRIPTION:

This command returns the state of the positive and negative latches, then clears the positive latches in the I/O Unit.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	117

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Unit Error
DATA[0]:	Bitmask of positive latches; bit set indicates latch set.
DATA[1]:	Bitmask of negative latches; bit set indicates latch set. Bitmask range is 0000 to FFFF Hex.

READ AND CLEAR NEGATIVE LATCHES (S) 118

DESCRIPTION:

This command returns the state of the positive and negative latches, then clears the negative latches in the I/O Unit.

SEND PARAMETERS:

COMMAND:	118

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Unit Error
DATA[0]:	Bitmask of positive latches; bit set indicates latch set.
DATA[1]:	Bitmask of negative latches; bit set indicates latch set. Bitmask range is 0000 to FFFF Hex.

READ AND CLEAR POSITIVE AND NEGATIVE LATCHES (S) 119

DESCRIPTION:

This command returns the state of the positive and negative latches, then clears all the latches in the I/O Unit.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	119

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Unit Error
DATA[0]:	Bitmask of positive latches; bit set indicates latch set.
DATA[1]:	Bitmask of negative latches; bit set indicates latch set. Bitmask range is 0000 to FFFF Hex.

READ I/O STATUS MASK (R) 120

DESCRIPTION:

This command reads the status of all the digital channels (input or output).
The value is returned as a bit mask.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	120

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error
DATA[0]:	Bitmask of status; bit set indicates channel is in ON state. Bitmask range is 0000 to FFFF Hex.

CHAPTER 5

Single Channel Digital Commands

SET I/O CONFIGURATION (a) 200

DESCRIPTION:

This command configures a channel on an I/O unit.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	200
POSITION[0]:	Channel Number 0-15 (00-0F Hex).
DATA[0]:	Configure
Types	Counter (default) = 0 Positive Pulse Measurement = 1 Negative Pulse Measurement = 2 Period Measurement = 3 Frequency Measurement = 4 Quadrature Counter Input = 5 On Time Totalizer Input = 6 Off Time Totalizer Input = 7 Output = 128 0522_MisticWare_Manual.ps

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

CLEAR (DEACTIVATE) OUTPUT (e) 201

DESCRIPTION:

This command turns OFF a channel on an I/O unit.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	201
POSITION[0]:	Channel Number 0-15 (00-0F Hex).

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

SET (ACTIVATE) OUTPUT (d) 202

DESCRIPTION:

This command turns ON a channel on an I/O unit.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	202
POSITION[0]:	Channel Number 0-15 (00-0F Hex).

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

ENABLE COUNTER INPUT (b) 203

DESCRIPTION:

This command enables a channel that has been configured as a counter.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	203
POSITION[0]:	Channel Number 0-15 (00-0F Hex).

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

DISABLE COUNTER INPUT (b) 204

DESCRIPTION:

This command disables a channel that has been configured as a standard counter or quadrature counter.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	204
POSITION[0]:	Channel Number 0-15 (00-0F Hex).

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

CLEAR COUNTER INPUT (c) 205

DESCRIPTION:

This command clears the counter value of a channel that has been configured as a standard counter or a quadrature counter to 0.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	205
POSITION[0]:	Channel Number 0-15 (00-0F Hex).

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

READ COUNTER INPUT (32-BIT) (I) 206

DESCRIPTION:

This command reads the counter (standard or quadrature) value of a channel and returns a 32-bit value.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	206
POSITION[0]:	Channel Number 0-15 (00-0F Hex).

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error
DATA[N]:	Counter Value; n is the value specified in *POSITION[0]:* for channel number. Range is 0 to 4,294,967,295 counts.

READ AND CLEAR COUNTER INPUT (32 BIT) (n) 207

DESCRIPTION:

This command reads the counter (standard or quadrature) value of a channel, returns a 32-bit value, then clears the counter to 0.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	207
POSITION[0]:	Channel Number 0-15 (00-0F Hex).

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error
DATA[N]:	Counter Value; n is the value specified in *POSITION[0]:* or channel number. Range is 0 to 4,294,967,295 counts.

READ COUNTER INPUT (16 BIT) (m) 208

DESCRIPTION:

This command reads the counter (standard or quadrature) value of a channel and returns a 16-bit value.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	208
POSITION[0]:	Channel Number 0-15 (00-0F Hex).

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error
DATA[N]:	Counter Value; n is the value specified in *POSITION[0]:* for channel number. Range is 0 to 65,535 counts.

READ AND CLEAR COUNTER INPUT (16 BIT) (0) 209

DESCRIPTION:

This command reads the counter (standard or quadrature) value of a channel, returns a 16-bit value, then clears the counter to 0.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	209
POSITION[0]:	Channel Number 0-15 (00-0F Hex).

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error
DATA[N]:	Counter Value; n is the value specified in *POSITION[0]:* for channel number. Range is 0 to 65,535 counts.

READ PULSE/PERIOD/TOTALIZER MEASUREMENT INPUT (32-BIT) (p) 210

DESCRIPTION:

This command reads the pulse, period, or totalizer measurement value of a channel and returns a 32-bit value.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	210
POSITION[0]:	Channel Number 0-15 (00-0F Hex).

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error
DATA[N]:	Pulse/Period Measurement Value; n is the value specified in *POSITION[0]:* for channel number. Range is 0 to 4,294,967,295 units, where one unit is 100 microseconds.

READ AND RESET PULSE/PERIOD/TOTALIZER MEASUREMENT (32 BIT) (r) 211

DESCRIPTION:

This command reads the pulse, period, or totalizer measurement value, returns a 32-bit value, then resets the I/O Unit's value to 0.

SEND PARAMETERS;

ADDRESS:	Address of I/O Unit
COMMAND:	211
POSITION[0]:	Channel Number 0-15 (00-0F Hex).

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error
DATA[N]:	Pulse/Period Measurement Value; n is the value specified in *POSITION[0]:* for channel number. Range is 0 to 4,294,967,295 units, where one unit is 100 microseconds.

READ PULSE/PERIOD/TOTALIZER MEASUREMENT INPUT (16-BIT) (q) 212

DESCRIPTION:

This command reads the pulse, period, or totalizer measurement value of a channel and returns a 16-bit value.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	212
POSITION[0]:	Channel Number 0-15 (00-0F Hex).

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error
DATA[N]:	Pulse/Period Measurement Value; n is the value specified in *POSITION[0]:* for channel number. Range is 0 to 65,535 units, where one unit is 100 microseconds.

READ AND RESET PULSE/PERIOD/TOTALIZER MEASUREMENT (16-BIT) (s) 213

DESCRIPTION:

This command reads the pulse, period, or totalizer measurement value, returns a 16-bit value, then resets the I/O Unit's value to 0.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	213
POSITION[0]:	Channel Number 0-15 (00-0F Hex).

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error
DATA[N]:	Pulse/Period Measurement Value; n is the value specified in *POSITION[0]:* for channel number. Range is 0 to 65,535 units, where one unit is 100 microseconds.

READ FREQUENCY (t) 214

DESCRIPTION:

This command reads the frequency value of a channel that has been configured for frequency measurement.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	214
POSITION[0]:	Channel Number 0-15 (00-0F Hex).

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Unit Error
DATA[N]:	Frequency Measurement Value; n is the value specified in *POSITION[0]:* for channel number. Range is 0 to 65,535 units. One unit represents 1 Hz or 10 Hz depending on System Options Byte. Maximum frequency input is 25KHz.

START ON PULSE (f) 215

DESCRIPTION:

This command starts a retriggerable ON pulse with a specified duration for the specified output channel.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	215
POSITION[0]:	Channel Number 0-15 (00-0F Hex).
DATA[0]:	Pulse duration in units of 100 microseconds. Range is 5 to 4,294,967,295 units.

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

START OFF PULSE (g) 216

DESCRIPTION:

This command starts a retriggerable OFF pulse with a specified duration for the specified output channel.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	216
POSITION[0]:	Channel Number 0-15 (00-0F Hex).
DATA[0]:	Pulse duration in units of 100 microseconds. Range is 5 to 4,294,967,295 units.

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

START CONTINUOUS SQUARE WAVE (h) 217

DESCRIPTION:

This command is used to start a continuous square wave with specified ON and OFF times for the specified output channel.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	217
POSITION[0]:	Channel Number 0-15 (00-0F Hex).
DATA[0]:	ON Pulse Duration in units of 100 microseconds. Range is 10 to 4,294,967,295 units.
DATA[1]:	OFF Pulse Duration in units of 100 microseconds. Range is 10 to 4,294,967,295 units.

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

GENERATE N PULSES (i) 218

DESCRIPTION:

This command is used to generate an output pulse stream with specified ON• and OFF pulse duration and a specified number of pulses for the specified output channel.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	218
POSITION[0]:	Channel Number 0-15 (00-0F Hex).
DATA[0]:	ON Pulse Duration in units of 100 microseconds. Range is 10 to 4,294,967,295 units.
DATA[1]:	OFF Pulse Duration in units of 100 microseconds. Range is 10 to 4,294,967,295 units.
DATA[2]:	Number of pulses. Range is 10 to 4,294,967,295 pulses.

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

SET TPO PERIOD ( ]) 219

DESCRIPTION:

This command is used to set the period of a time proportional output. This command must be executed before the “SET TPO PERCENT” command (220).

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	219
POSITION[0]:	Channel Number 0-15 (00-0F Hex).
DATA[0]:	Period Value in units of 100 microseconds. Range is 1000 to 4,294,967,295 units.

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

SET TPO PERCENT (j) 220

DESCRIPTION:

This command is used to set the percent ON time of a time proportional output. The "SET TPO PERIOD" (219) command must have been previously sent at least once before this command can be executed.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	220
POSITION[0]:	Channel Number 0-15 (00-0F Hex).
DATA[0]:	Percent Value in units of 1/65,536 of a percent. Range is 0 to 4,294,967,295 units. A value of 0 (0%) will turn the channel OFF, and a value of 6,553,600 (100%) or greater will turn the channel ON continuously.

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

READ LATCHES (w) 221

DESCRIPTION:

This command reads the latch values for a channel. It does not clear the latches.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	221
POSITION[0]:	Channel Number 0-15 (00-0F Hex).

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Unit Error
DATA[N]:	Latch Values; n is the value specified in *POSITION[0]:* for channel number. Range is 0 to 255 (FF Hex), with the following bits set: bit 0: State of Positive Latch bit 1: State of Negative Latch bit 2: State of Module bits 3-7 are reserved. If bit is set, latch is set or module is On.

READ LATCHES AND CLEAR POSITIVE LATCH (w) 222

DESCRIPTION:

This command reads the latch values for a channel, and clears the positive latch value.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	222
POSITION[0]:	Channel Number 0-15 (00-0F Hex).

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Unit Error
DATA[N]:	Latch Values; n is the value specified in *POSITION[0]:* for channel number. Range is 0 to 255 (FF Hex), with the following bits set*:* bit 0*:* State of Positive Latch bit 1*:* State of Negative Latch bit 2*:* State of Module bits 3-7 are reserved. If bit is set, latch is set or module is On

READ LATCHES AND CLEAR NEGATIVE LATCH (w) 223

DESCRIPTION:

This command reads the latch values for a channel, and clears the negative latch value.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	223
POSITION[0]:	Channel Number 0-15 (00-0F Hex).

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Unit Error
DATA[N]:	Latch Values; n is the value specified in *POSITION[0]:* for channel number. Rabit 0*:* State of Positive Latch bit 1*:* State of Negative Latch bit 2*:* State of Module bits 3-7 are reserved. If bit is set, latch is set or module is On.

READ AND CLEAR LATCHES (w) 224

DESCRIPTION:

This command reads the latch values for a channel, and clears both the negative and positive latch values.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	224
POSITION[0]:	Channel Number 0-15 (00-0F Hex).

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Unit Error
DATA[N]:	Latch Values; n is the value specified in *POSITION[0]:* for channel number. Range is 0 to 255 (FF Hex), with the following bits set*:* bit 0*:* State Of Positive Latch bit 1*:* State Of Negative Latch bit 2*:* State Of Module bits 3-7 are reserved If bit is set, latch is set or module is On.

READ OUTPUT TIMER COUNTER (k) 225

DESCRIPTION:

This command reads the current time delay value of an output channel which may have a delay in progress. The value returned is the time remaining for the delay.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	225
POSITION[0]:	Channel Number 0-15 (00-0F Hex).

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error
DATA[N]:	Time Delay Value in units of 100 microseconds; n is the value specified in *POSITION[0]:* for channel number. Range is 0 to 4,294,967,295 units.

CHAPTER 6

Multichannel Analog Commands

SET I/O CONFIGURATION (G) 300

DESCRIPTION:

This command configures the channels on an analog I/O unit.

SEND PARAMETERS:

ADDRESS: Address of I/O Unit

COMMAND: 300

POSITION[0]: Group Mask (0000 to FFFF Hex)

DATA[0]...
DATA[15]: Configure Type. If bit n is set in *POSITION[0]:*, type must be in Data[n]. eg. *POSITION[0]:* = 32 Data[6] = Type

Generic Input (default) = 0
G4AD3 4 - 20 mA = 3
G4AD4 ICTD = 4
G4AD5 Type J Therm. = 5
G4AD6 0 - 5 VDC = 6
G4AD7 0 - 10 VDC = 7
G4AD8 Type K Therm. = 8
G4AD9 0 - 50 mV = 9
G4AD10 100 Ohm RTD = 10
G4AD11 - 5 to + 5 VDC = 11
G4AD12 - 10 to + 10 VDC = 12
G4AD13 0 - 100 mV = 13
G4AD16 0 - 5 Amps, AC/DC = 16
G4AD17 Type R Therm. = 17
G4AD18 Type T Therm. = 18
G4AD19 Type E Therm. = 19
G4AD20 Rate = 20
G4AD22 0 - 1 VDC = 21
G4AD23 Type S Therm. = 23
G4AD24 Type B Therm. = 24

G4AD25 0 - 100 VAC/VDC = 25
Generic Output = 28
G4DA3 4 - 20 mA = 131
G4DA4 0 - 5 VDC = 132
G4DA5 0 - 10 VDC = 133
G4DA6 - 5 to + 5 VDC = 134
G4DA7 - 10 to + 10 VDC = 135
G4DA8 0 - 20 mA = 136
G4DA9 TPO = 137

0-20mA Input = 02
4 - 20 mA Input = 03
-20mA to +20mA Input = 40
0 to 10 Amps RMS Input = 47
-25 mV to +25mV Input = 43
-50 mV to +50 mV Input = 09
-75 mV to +75mV Input = 44
-150 mV to +150mV Input = 42
0 - 5 VDC Input = 06
0 - 10 VDC Input = 07
5 to + 5 VDC Input = 0B

-10 to + 10 VDC Input = 0C
0 to 250 V RMS Input = 46
ICTD Input = 04
100 Ohm (-200° to 850° C) RTD = 0A
100 Ohm (-60° to 250° C) RTD = 2E
120 Ohm (-80° to 260° C) RTD = 30
Type E Therm. = 13
Type J Therm. = 05
Type K Therm. = 08
Type B Therm. = 18
Type C Therm. = 20
Type D Therm. = 21
Type G Therm. = 1F
Type N Therm. = 1E
Type R Therm. = 11
Type S Therm. = 17
Type T. Therm. = 12
0 to 25, 000 Hz Input = 45
4 - 20 mA Output = A3
0 - 10 VDC Output = A5
-10 to + 10 VDC Output = A7

RECEIVE PARAMETERS:

ERROR: Driver or I/O Error

READ I/O CONFIGURATION (Y) 301

DESCRIPTION:

This command reads the I/O configuration of an analog I/O unit.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	301

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

DATA[0]...
DATA[15]:
	Configuration type.
	Data[0] = type for channel 0.
	Data[15] = type for channel 15.
	See command 300 for type descriptions.

READ I/O MODULE MAGNITUDES-COUNTS (R) 302-326

DESCRIPTION:

This command reads the magnitudes of analog channels and returns the values in accordance with the command used.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	302 = Raw Counts 303 = Average Count 304 = Peak Counts 305 = Low Counts 306 = Totalized Counts 322 = Square Root Of Raw Counts 323 = Square Root Of Average Counts 324 = Square Root Of Peak Counts 325 = Square Root Of Low Counts 326 = Square Root Of Total Counts
Positions[0]:	Bitmask of channels to read. A bit set reads that channel's value. Bitmask range is 0000 to FFFF Hex.

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error
DATA[0]... *DATA[15]:*	Channel Values from the channels specified in the Positions array. If *POSITION[0]:* = 8001 Hex (Chan. 0 and 15) Then Data[0] = Value of channel 0 and Data[15] = Value of channel 15.

READ I/O MODULE MAGNITUDES-ENGINEERING UNITS (R) 307-331

DESCRIPTION:

This command reads the magnitude of analog channels and returns values in engineering units in accordance with the command used and each channel's scaling parameters.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	307 = Scaled Value 308 = Average Value 309 = Peak Value 310 = Low Value 311 = Totalized Value 327 = Square Root Of Scaled Value 328 = Square Root Of Average Value 329 = Square Root Of Peak Value 330 = Square Root Of Low Value 331 = Square Root Of Totalized Value
Positions[0]:	Bitmask of channels to read. A bit set reads that channel's value. Bitmask range is 0000 to FFFF Hex.

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error
DATA[0]... *DATA[15]:*	Channel Values from the channels specified in the Positions array. If *POSITION[0]:* = 8001 Hex (Chan. 0 and 15) Then Data[0] = Value of channel 0, and Data[15] = Value of channel 15. 32-bit values in increments of 1/65,536 of the engineering units.

READ AND CLEAR I/O MAGNITUDES-COUNTS (S) 312-336

DESCRIPTION:

This command reads the magnitudes of analog channels, returns the count values in accordance with the command used, then clears the channels.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	312 = Raw Counts 313 = Average Count 314 = Peak Count 315 = Low Count 316 = Totalized Count 332 = Square Root Of Raw Count 333 = Square Root Of Average Count 334 = Square Root Of Peak Count 335 = Square Root Of Low Counts 336 = Square Root Of Total Counts
POSITION[0]:	Bitmask of channels to read. A bit set reads that channel's value. Bitmask range is 0000 to FFFF Hex.

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error
DATA[0]... *DATA[15]:*	Channel Values from the channels specified in the Positions Array. If *POSITION[0]:* = 8001 Hex (Chan. 0 & 15) Then Data[0] = Value of channel 0, and Data[15] = Value of channel 15.

READ AND CLEAR MAGNITUDES-ENGINEERING UNITS (S) 317-341

DESCRIPTION:

This command reads the magnitudes of analog channels, returns values in engineering units in accordance with the command used and the channel's scaling parameters, then clears the channels.

SEND PARAMETERS:

ADDRESS: Address of I/O Unit

COMMAND:
317 = Scaled Value
318 = Average Value
319 = Peak Value
320 = Low Value
321 = Totalized Value
337 = Square Root Of Scaled Value
338 = Square Root Of Average Value
339 = Square Root Of Peak Value
340 = Square Root Of Low Value
341 = Square Root Of Totalized Value

Positions[0]:
Bitmask of channels to read.
A bit set reads that channel's value.
Bitmask range is 0000 to FFFF Hex.

RECEIVE PARAMETERS:

ERROR: Driver or I/O Error

DATA[0]...
DATA[15]: Channel Values from the channels specified in the Positions array.
If *POSITION[0]:* = 8001 Hex (Chan. 0 & 15)
Then Data[0] = Value of channel 0, and
Data[15] = Value of channel 15.
32-bit values in increments of 1/65,536 of the engineering units.

SET I/O MODULE MAGNITUDES-COUNTS (X) 342

DESCRIPTION:

This command sets the magnitudes of analog channels configured as outputs.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	342
POSITION[0]:	Bitmask of channels to set. A bit set sets that channel's value. Bitmask range is 0000 to FFFF Hex.
DATA[0]... *DATA[15]:*	Channel Values to set for the channels specified in the Positions Array. Range: 0 to 4095. If *POSITION[0]:* = 8001 Hex (Chan. 0 and 15) Then Data[0] = Value for channel 0, and Data[15] = Value for channel 15.

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

SET I/O MODULE MAGNITUDES-ENGINEERING UNITS (W) 343

DESCRIPTION:

This command sets the magnitudes of analog channels configured as outputs using scaled engineering units.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	343
POSITION[0]:	Bitmask of channels to set. A bit set sets that channel's value. Bitmask range is 0000 to FFFF Hex.
DATA[0]... *DATA[15]:*	Channel values to set for the channels specifiedin the Positions Array. Range*:* 32-bit values (-2,147,483,648 to +2,147,483,647) in increments of 1/65,536 of the engineering units. If *POSITION[0]:* = 8001 Hex (Chan. 0 and 15) Then Data[0] = Value for channel 0, and Data[15] = Value for channel 15.

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

SET WATCHDOG TIME (D) 344

DESCRIPTION:

This command sets the communications line watchdog timeout period for the addressed I/O unit.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	344
DATA[0]:	Delay Value in units of 10 mS. Range*:* 20 to 4095 (200mS-40.95 Sec) Value of 0 Disables Watchdog

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

SET WATCHDOG DATA-ENGINEERING UNITS (H) 345

DESCRIPTION:

This command sets the desired output values for the specified analog modules upon a communications link watchdog timeout condition.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	345
POSITION[0]:	Bitmask of channels to set. A bit set sets that channel's value. Bitmask range is 0000 to FFFF Hex.
DATA[0]... *DATA[15]:*	Channel Values to set for the channels specified in the Positions Array. Range*:* 32-bit values (-2,147,483,648 to +2,147,483,647) in increments of 1/65,536 of the engineering units. If *POSITION[0]:* = 8001 Hex (Chan. 0 and 15) Then Data[0] = Value for channel 0, and Data[15] = Value for channel 15.

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

CHAPTER 7

Single Channel Analog Commands

SET I/O CHANNEL CONFIGURATION (a) 400

DESCRIPTION:

This command configures a channel on an analog I/O unit.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	400
POSITION[0]:	Channel Number 0-15 (00-0F Hex).
DATA[0]:	Configure Type.

[Decimal]
Generic Input (default or scalable) = 0
G4AD3 4 - 20 mA = 3
G4AD4 ICTD = 4
G4AD5 Type J Therm. = 5
G4AD6 0 - 5 VDC = 6
G4AD7 0 - 10 VDC = 7
G4AD8 Type K Therm. = 8
G4AD9 0 - 50 mV = 9
G4AD10 100 Ohm RTD = 10
G4AD11 - 5 to + 5 VDC = 11
G4AD12 - 10 to + 10 VDC = 12
G4AD13 0 - 100 mV = 13
G4AD16 0 - 5 Amps, AC/DC = 16
G4AD17 Type R Therm. = 17
G4AD18 Type T Therm. = 18
G4AD19 Type E Therm. = 19
G4AD20 Rate = 20
G4AD22 0 - 1 VDC = 21
G4AD23 Type S Therm. = 23
G4AD24 Type B Therm. = 24
G4AD25 0 - 100 VAC/VDC = 25
Generic Output = (default or scalable) 128
G4DA3 4 - 20 mA = 131
G4DA4 0 - 5 VDC = 132
G4DA5 0 - 10 VDC = 133
G4DA6 - 5 to + 5 VDC = 134
G4DA7 - 10 to + 10 VDC = 135
G4DA8 0 - 20 mA = 136
G4DA9 TPO = 137

[Hex]
0-20mA Input = 02
4 - 20 mA Input = 03
-20mA to +20mA Input = 40
0 to 10 Amps RMS Input = 47
-25 mV to +25mV Input = 43
-50 mV to +50 mV Input = 09
-75 mV to +75mV Input = 44
-150 mV to +150mV Input = 42
0 - 5 VDC Input = 06
0 - 10 VDC Input = 07
5 to + 5 VDC Input = 0B
-10 to + 10 VDC Input = 0C
0 to 250 V RMS Input = 46
ICTD Input = 04
100 Ohm (-200° to 850° C) RTD = 0A
100 Ohm (-60° to 250° C) RTD = 2E
120 Ohm (-80° to 260° C) RTD = 30
Type E Therm. = 13
Type J Therm. = 05
Type K Therm. = 08
Type B Therm. = 18
Type C Therm. = 20
Type D Therm. = 21
Type G Therm. = 1F
Type N Therm. = 1E
Type R Therm. = 11
Type S Therm. = 17
Type T. Therm. = 12
0 to 25, 000 Hz Input = 45
4 - 20 mA Output = A3
0 - 10 VDC Output = A5
-10 to + 10 VDC Output = A7
SNAP-AOD-29 = A9

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

SET ANALOG TPO PERIOD ([ ]) 401

DESCRIPTION:

This command sets the period prescale value for the G4DA9 or the SNAP-AOD-29 analog TPO module.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	401
POSITION[0]:	Channel Number 0-15 (00-0F Hex)
Data [0]:	Resolution Setting. Range: 8-bit value (1-255).

For the G4DA9, it equals the resolution in 500us units. The period of the TPO module is calculated as follows:

TPO Period = Resolution Setting * 500 microseconds * 4,096.
The default Resolution Setting is 2, yielding a resolution of 1 millisecond and a TPO period of 4.096 second.

For the SNAP-AOD-29, calculate the period as follows:

TPO Period = (Resolution Setting + 1) * 251 ms

For example, a setting of zero yields 251 ms; a setting of one yields 502 ms, etc.

IMPORTANT: After using this command, set the TPO percentage using the command Set I/O Mode Magnitude - Engineering Units (443). For example, set a percentage such as 50%, where the actual value for Data[0] is 50 (65536) = 3,276,800

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

READ I/O MODULE MAGNITUDE-COUNTS (r) 402-426

DESCRIPTION:

This command reads the magnitude of an analog channel and returns a value in accordance with the command used.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	402 = Raw Counts 403 = Average Counts 404 = Peak Counts 405 = Low Counts 406 = Totalized Counts 422 = Square Root Of Raw Counts 423 = Square Root Of Average Counts 424 = Square Root Of Peak Counts 425 = Square Root Of Low Counts 426 = Square Root Of Total Counts
POSITION[0]:	Channel Number 0-15 (00-0F Hex).

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error
DATA[0]:	Channel Value

READ I/O MODULE MAGNITUDE-ENGINEERING UNITS (r) 407-431

DESCRIPTION:

This command reads the magnitude of an analog channel and returns a value in engineering units in accordance with the command used and the channel's scaling parameters.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	407 = Scaled Value 408 = Average Value 409 = Peak Value 410 = Low Value 411 = Totalized Value 427 = Square Root Of Scaled Value 428 = Square Root Of Average Value 429 = Square Root Of Peak Value 430 = Square Root Of Low Value 431 = Square Root Of Totalized Value
POSITION[0]:	Channel Number 0-15 (00-0F Hex).

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error
DATA[0]:	Channel Value. 32-bit value in increments of 1/65,536 of the engineering units.

READ AND CLEAR I/O MAGNITUDE-COUNTS (s) 412-436

DESCRIPTION:

This command reads the magnitudes of an analog channel, returns the counts value in accordance with the command used, then clears the channel.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	412 = Raw Counts 413 = Average Counts 414 = Peak Counts 415 = Low Counts 416 = Totalized Counts 432 = Square Root Of Raw Counts 433 = Square Root Of Average Counts 434 = Square Root Of Peak Counts 435 = Square Root Of Low Counts 436 = Square Root Of Total Counts
POSITION[0]:	Channel Number 0-15 (00-0F Hex). Receive Parameters:
ERROR:	Driver or I/O Error
DATA[0]:	Channel Value

READ AND CLEAR I/O MAGNITUDE-ENGINEERING UNITS (s) 417-441

DESCRIPTION:

This command reads the magnitude of an analog channel, returns the value in engineering units in accordance with the command used and the channel's scaling parameters, then clears the channel.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	417 = Scaled Value 418 = Average Value 419 = Peak Value 420 = Low Value 421 = Totalized Value 437 = Square Root Of Scaled Value 438 = Square Root Of Average Value 439 = Square Root Of Peak Value 440 = Square Root Of Low Value 441 = Square Root Of Totalized Value
POSITION[0]:	Channel Number 0-15 (00-0F Hex).

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error
DATA[0]:	Channel Value. 32-bit value in increments of 1/65,536 of the engineering units.

SET I/O MODULE MAGNITUDE-COUNTS (x) 442

DESCRIPTION:

This command sets the magnitude of an analog channel configured as an output.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	442
POSITION[0]:	Channel Number 0-15 (00-0F Hex).
DATA[0]:	Channel Value. Range: 0 to 4,095.

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

SET I/O MODULE MAGNITUDE-ENGINEERING UNITS (w) 443

DESCRIPTION:

This command sets the magnitude of an analog channel configured as an output using scaled engineering units.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	443
POSITION[0]:	Channel Number 0-15 (00-0F Hex).
DATA[0]:	Channel Value. Range: 32-bit value (-2,147,483,648 to +2,147,483,647) in increments of 1/65,536 of the engineering units.

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

SET SCALING PARAMETERS (f) 444

DESCRIPTION:

This command sets the zero scale and full scale parameters of an analog channel (input or output).

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	444
POSITION[0]:	Channel Number 0-15 (00-0F Hex).
DATA [0]:	Full Scale Value. Range: 32-bit value (-2,147,483,648 to +2,147,483,647) in increments of 1/65,536 of the engineering units.
DATA [1]:	Zero Scale Value. Range: 32-bit value (-2,147,483,648 to +2,147,483,647) in increments of 1/65,536 of the engineering units.

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

SET TOTALIZATION SAMPLE RATE (g) 445

DESCRIPTION:

This command initiates totalization on an input or output channel by setting the sample rate in 100 mSec units.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	445
POSITION[0]:	Channel Number 0-15 (00-0F Hex).
DATA[0]:	Sample Rate. Range: 0 to 65,535 in units of 100 mSec (0 - 6,553.5 Sec.)

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

SET AVERAGING SAMPLE WEIGHT (h) 446

DESCRIPTION:

This command initiates averaging (digital filtering) on an input channel by setting the sample weight using the following formulas:

New Average = (Current-Old)/Sample Weight)+Old
Sample Weight = (100mSec + TC)/100mSec

Where TC is the desired filter time constant.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	446
POSITION[0]:	Channel Number 0-15 (00-0F-Hex)
DATA[0]:	Sample Weight. Range: 0 to 65,535.

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

CALCULATE AND SET OFFSET (d) 447

DESCRIPTION:

This command instructs the addressed I/O unit to use the current value of the specified input channel as the zero scale value. This reading is multiplied by -1 and is used as an offset. The offset value is returned.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	447
POSITION[0]:	Channel Number 0-15 (00-OF-Hex)

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error
DATA[0]:	Offset Counts. Range: 0 to 65,535.

Note: *The input must be at a zero or minimum value when this command is issued.*

CALCULATE AND SET GAIN (e) 448

DESCRIPTION:

This command instructs the addressed I/O unit to use the current value of the specified input channel as the full scale value. This reading is divided by 4,096 and is used as a gain coefficient. The input reading value is returned.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	448
POSITION[0]:	Channel Number 0-15 (00-OF-Hex)

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error
DATA[0]:	Gain Counts. Range: 0 to 65,535.

Note: *This command adjusts the gain so that the value applied to the input when the command is issued becomes a full-scale reading.*

SET OFFSET (b) 449

DESCRIPTION:

This command sets an offset for an analog input channel. The offset is added to the raw count data before any engineering unit scaling is done.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	449
POSITION[0]:	Channel Number 0-15 (00-OF-Hex)
DATA[0]:	Offset Counts. Range: 0 to 65,535.

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

SET GAIN (c) 450

DESCRIPTION:

This command is used to set the gain coefficient for the an analog input channel. The gain coefficient must be scaled by 4,096.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	450
POSITION[0]:	Channel Number 0-15 (00-OF-Hex)
DATA[0]:	Gain Counts. Range: 0 to 65,535. Example: For a gain of .96, Gain Counts = .96 * 4,096 = 3,932.

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

RAMP OUTPUT TO ENDPOINT (Z) 451

DESCRIPTION:

This command is used to ramp an analog output from its current setting to a specified endpoint at a specified ramp rate.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	451
POSITION[0]:	Channel Number 0-15 (00-OF-Hex)
DATA[0]:	Ramp Endpoint. Range: 32-bit value (-2,147,483,648 to +2,147,483,647) in increments of 1/65,536 of an engineering unit.
DATA[1]:	Ramp Slope (Engineering Units per Sec). Range: 32-bit value (-2,147,483,648 to +2,147,483,647) in increments of 1/65,536 of an engineering unit. Slope of 0 is not allowed.

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

DUMP CHANNEL DATA ([ ) 452

DESCRIPTION:

This command is a reserved command used to obtain a raw dump of the I/O unit's channel database for a specified channel. The structure of this data is included in appendix B.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	452
POSITION[0]:	Channel Number 0-15 (00-OF-Hex)

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error
DATA[0]... *DATA[15]:*	64 bytes of data.

CHAPTER 8

Analog PID Commands

INITIALIZE PID LOOP (i) 500

DESCRIPTION:

This command initializes a PID loop by defining the input and output channels, and the scan rate.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	500
POSITION[0]:	Loop Number 0-7.
DATA[0]:	Input Channel Number 0-15.
DATA[1]:	Setpoint Channel Number 0-15.
DATA[2]:	Output Channel Number 0-15.
DATA[3]:	Scan Rate. Range: 0 to 65,535 in units of 100 mSec (0 - 6,553.5 Sec.)

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

SET PID LOOP CONTROL OPTIONS (j) 501

DESCRIPTION:

This command is used to set the operating options in the loop control word. The Loop Control Word is structured as follows:

Least Significant Byte (LSB) - Control Byte

bit 0	:	0	=	Current Reading
		1	=	Average Reading
bit 1	:	0	=	Setpoint From Host
		1	=	Setpoint From Channel
bit 2	:	0	=	Process Variable From Channel
		1	=	Process Variable From Host
bit 3	:	0	=	Setpoint Track Input In Manual Disabled
		1	=	Setpoint Track Input In Manual Enabled
bit 4	:	0	=	Output Track Input In Manual Disabled
		1	=	Output Track Input In Manual Enabled
bit 5	:	0	=	Output Disabled
		1	=	Output Enabled
bit 6	:	0	=	PID In Manual Mode
		1	=	PID In Auto Mode
bit 7	:	0	=	PID In Reset Mode
		1	=	PID In Active Mode

Most Significant Byte (MSB) - Flags Byte

bit 0	:	0	=	No Error
		1	=	Input Under Range Error
bit 1	:	0	=	No Error
		1	=	Minimum Setpoint Exceeded Error
bit 2	:	0	=	No Error
		1	=	Maximum Setpoint Exceeded Error
bits 3-7	:			Reserved.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	501
POSITION[0]:	Loop Number 0-7.
DATA[0]:	Mask Of Control Bits To Set. 0000 to FFFF Hex (0-65,535)
DATA[1]:	Mask Of Control Bits To Clear. 0000 to FFFF Hex (0-65,535)

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

SET PID LOOP SETPOINT (k) 502

DESCRIPTION:

This command sets the magnitude of a PID loop's setpoint using scaled engineering units.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	502
POSITION[0]:	Loop Number 0-7.
DATA[0]:	Setpoint Value. Range: 32-bit value (-2,147,483,648 to +2,147,483,647) in increments of 1/65,536 of the engineering units. The range of values must be the same as the range for the input channel.

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

SET PID LOOP GAIN (I) 503

DESCRIPTION:

This command sets the PID loop's gain value. A negative loop gain is used to effect a reverse control action.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	503
POSITION[0]:	Loop Number 0-7.
DATA[0]:	Gain Value. Range: 32-bit value (-2,147,483,648 to +2,147,483,647) scaled in increments of 1/65,536. Example: For a gain of 2, Gain Value = 2 * 65,536 = 131,072.

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

SET PID LOOP INTEGRAL RESET RATE (m) 504

DESCRIPTION:

This command sets the PID loop's integral value in repeats per minute.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	504
POSITION[0]:	Loop Number 0-7.
DATA[0]:	Integral Value. Range: 32-bit value (-2,147,483,648 to +2,147,483,647) scaled in increments of 1/65,536. Example: For an integral of 7.5, Integral Value = 7.5 * 65,536 = 491,520.

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

SET PID LOOP DERIVATIVE RATE (n) 505

DESCRIPTION:

This command sets the PID loop's derivative value in minutes.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	505
POSITION[0]:	Loop Number 0-7.
DATA[0]:	Deriv. Value. Range: 32-bit value (-2,147,483,648 to +2,147,483,647) scaled in increments of 1/65,536. Example: For a derivative of .03, Derivative Value = .03 * 65,536 = 1966.

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

SET PID LOOP SETPOINT MIN/MAX LIMITS (o) 506

DESCRIPTION:

This command is used to set the minimum and maximum allowable setpoints for the PID Loop. The units must be the same as those specified for the input.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	506
POSITION[0]:	Loop Number 0-7
DATA[0]:	Maximum Setpoint. Range: 32-bit value (-2,147,483,648 to +2,147,483,647) in increments of 1/65,536 of an engineering unit.
DATA[1]:	Minimum Setpoint. Range: 32-bit value (-2,147,483,648 to +2,147,483,647) in increments of 1/65,536 of an engineering unit.

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

READ ALL PID LOOP PARAMETERS (T) 507

DESCRIPTION:

This command reads all PID loop parameters.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	507
POSITION[0]:	Loop Number 0-7

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error
DATA[0]:	Output Value. Range: 32-bit value (-2,147,483,648 to +2,147,483,647) in increments of 1/65,536 of an engineering unit.
DATA[1]:	Output Value Counts. Range: 16-bit value (0-4095 counts).
DATA[2]:	Min. Setpoint Limit. Range: 32-bit value (-2,147,483,648 to +2,147,483,647) in increments of 1/65,536 of an engineering unit.
DATA[3]:	Max. Setpoint Limit. Range: 32-bit value (-2,147,483,648 to +2,147,483,647) in increments of 1/65,536 of a engineering unit.
DATA[4]:	Loop Derivative Rate (Minutes). Range: 32-bit value (-2,147,483,648 to +2,147,483,647) in increments of 1/65,536 of an engineering unit.
DATA[5]:	Loop Integral Reset Rate (Repeats per Minutes). Range: 32-bit value (-2,147,483,648 to +2,147,483,647) in increments of 1/65,536 of an engineering unit.
DATA[6]:	Loop Gain Value. Range: 32-bit value (-2,147,483,648 to +2,147,483,647) in increments of 1/65,536 of an engineering unit.
DATA[7]:	Setpoint Value. Range: 32-bit value (-2,147,483,648 to +2,147,483,647) in increments of 1/65,536 of an engineering unit.
DATA[8]:	Input Value. Range: 32-bit value (-2,147,483,648 to +2,147,483,647) in increments of 1/65,536 of an engineering unit.
DATA[9]:	Input, Output, Setpoint Channel Numbers.
DATA[10]:	Loop Control Word and Scan Rate.

READ LOOP CONTROL WORD (t) 508

DESCRIPTION:

This command reads the PID loop Control Word. See command 501 for a definition of the Control Word and each bit it contains.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	508
POSITION[0]:	Loop Number 0-7

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error
DATA[0]:	Loop Control Word. Range: 16-bit value, (0-65,535 or 0000-FFFF Hex)

READ SCAN RATE WORD (t) 509

DESCRIPTION:

This command reads the PID loop scan rate value. The value is in units of 100 mSec.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	509
POSITION[0]:	Loop Number 0-7

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error
DATA[0]:	Scan Rate Word. Range: 16-bit value (0-65,535), in units of 100 mSec.

READ OUTPUT COUNTS (t) 510

DESCRIPTION:

This command reads the value of the PID loop output in counts.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	510
POSITION[0]:	Loop Number 0-7

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error
DATA[0]:	Output Counts. Range: 16-bit value (0-4,095).

READ LOOP CHANNELS (t) 511

DESCRIPTION:

This command reads the channel numbers for the PID loop's input, setpoint, and output.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	511
POSITION[0]:	Loop Number 0-7

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error
DATA[0]:	Output Channel (0-15).
DATA[1]:	Reserved.
DATA[2]:	Setpoint Channel (0-15).
DATA[3]:	Input Channel (0-15).

READ INPUT VALUE (PROCESS VARIABLE) (t) 512

DESCRIPTION:

This command reads the value for the PID loop's input channel.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	512
POSITION[0]:	Loop Number 0-7

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error
DATA[0]:	Input Value. Range: 32-bit value (-2,147,483,648 to +2,147,483,647) in increments of 1/65,536 of an engineering unit.

READ SETPOINT VALUE (t) 513

DESCRIPTION:

This command reads the PID loop's setpoint value.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	513
POSITION[0]:	Loop Number 0-7

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error
DATA[0]:	Setpoint Value. Range: 32-bit value (-2,147,483,648 to +2,147,483,647) in increments of 1/65,536 of an engineering unit.

READ OUTPUT VALUE (t) 514

DESCRIPTION:

This command reads the PID loop's output value.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	514
POSITION[0]:	Loop Number 0-7

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error
DATA[0]:	Output Value. Range: 32-bit value (-2,147,483,648 to +2,147,483,647) in increments of 1/65,536 of an engineering unit.

READ GAIN TERM (t) 515

DESCRIPTION:

This command reads the PID loop's Gain Term.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	515
POSITION[0]:	Loop Number 0-7

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error
DATA[0]:	Gain Term. Range: 32-bit value (-2,147,483,648 to +2,147,483,647) in increments of 1/65,536 of an engineering unit.

READ INTEGRAL TERM (t) 516

DESCRIPTION:

This command reads the PID loop's Integral Term. This value is in repeats per minute.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	516
POSITION[0]:	Loop Number 0-7

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error
DATA[0]:	Integral Term (Repeats per Min.). Range: 32-bit value (-2,147,483,648 to +2,147,483,647) in increments of 1/65,536 of an engineering unit.

READ DERIVATIVE TERM (t) 517

DESCRIPTION:

This command reads the PID loop's Derivative Term. This value is in minutes.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	517
POSITION[0]:	Loop Number 0-7

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error
DATA[0]:	Derivative Term (Minutes). Range: 32-bit value (-2,147,483,648 to +2,147,483,647) in increments of 1/65,536 of an engineering unit.

READ MAXIMUM SETPOINT LIMIT (t) 518

DESCRIPTION:

This command reads the Maximum Setpoint Limit of the PID loop.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	518
POSITION[0]:	Loop Number 0-7

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error
DATA[0]:	Max. Setpoint Limit. Range: 32-bit value (-2,147,483,648 to +2,147,483,647) in increments of 1/65,536 of an engineering unit.

READ MINIMUM SETPOINT LIMIT (t) 519

DESCRIPTION:

This command reads the Minimum Setpoint Limit of the PID loop.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	519
POSITION[0]:	Loop Number 0-7

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error
DATA[0]:	Minute Setpoint Limit. Range: 32-bit value (-2,147,483,648 to +2,147,483,647) in increments of 1/65,536 of an engineering unit.

SET PID LOOP OUTPUT MIN/MAX LIMITS (p) 520

DESCRIPTION:

This command is used to set the minimum and maximum allowable output of the PID Loop.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	520
POSITION[0]:	Loop Number 0-7
DATA[0]:	Maximum Output. Range: 32-bit value (-2,147,483,648 to +2,147,483,647) in increments of 1/65,536 of an engineering unit.
DATA[1]:	Minimum Output. Range: 32-bit value (-2,147,483,648 to +2,147,483,647) in increments of 1/65,536 of an engineering unit.

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

SET PID LOOP PROCESS VARIABLE (q) 521

DESCRIPTION:

This command sets the PID loop's process variable.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	521
POSITION[0]:	Loop Number 0-7
DATA[0]:	Process variable. Range: 32-bit value (-2,147,483,648 to +2,147,483,647) scaled in increments of 1/65,536. Example: For an integral of 7.5, Integral Value = 7.5* 65,536 = 491,520.

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

CHAPTER 9

Digital Event/Reaction Commands

CLEAR EVENT/REACTION TABLE (_) 600

DESCRIPTION:

This command is used to clear the entire Event/Reaction table. All event latches and interrupts are also cleared.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	600

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

CLEAR EVENT TABLE ENTRY (\ ) 601

DESCRIPTION:

This command is used to clear a single entry in the Event/Reaction table.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	601
POSITION[0]:	Event Table Entry Number (00-FF-Hex)

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

READ EVENT ENTRY ENABLE/DISABLE STATUS (v) 602

DESCRIPTION:

This command returns 16 status bits representing the status of 16 events starting with the specified event number. Each bit corresponds to the status of one event. If the bit is 1, the event is enabled; if the bit is 0, the event is disabled.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	602
POSITION[0]:	Event Table Entry Number (00-FF-Hex).
DATA[0]:	Status of 16 events. Range: 16-bit (0-65,535 or 0000-FFFF Hex).

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

READ ALL EVENT ENTRIES ENABLE/DISABLE STATUS (v) 603

DESCRIPTION:

This command returns 255 status bits representing the status of all events in the Event/Reaction Table. Each bit corresponds to the status of one event. If the bit is 1, the event is enabled; if the bit is 0, the event is disabled.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	603

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

DATA[0]...
DATA[15]:
	Status of 16 events.
	Range: 16-bit (0-65,535 or 0000-FFFF-Hex).
	Data[0] = Events 0-15, and
	Data[15] = Events 240-255.

ENABLE EVENT TABLE ENTRY (N) 604

DESCRIPTION:

This command is used to enable an event table entry.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	604
POSITION[0]:	Event Table Entry Number (00-FF-Hex).

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

DISABLE EVENT TABLE ENTRY (N) 605

DESCRIPTION:

This command is used to disable an event table entry.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	605
POSITION[0]:	Event Table Entry Number (00-FF-Hex).

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

ENABLE/DISABLE EVENT TABLE GROUP ({) 606

DESCRIPTION:

This command is used to set the enable/disable status of an entire event table group of 16 entries.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	606
POSITION[0]:	Event Table Group Number (00-FF-Hex).
DATA[0]:	16-bit Mask of events within group to enable. If bit is 1, event is enabled.
DATA[1]:	16-bit Mask of events within group to disable. If bit is 1, event is disabled.

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

READ EVENT TABLE ENTRY (O) 608

DESCRIPTION:

This command is used to read the contents of an event table entry. The Reaction Command can have the following value:

NULL REACTION (DO NOTHING)	00
SET OUTPUT MODULE STATE	01
START ON PULSE	02
START OFF PULSE	03
ENABLE/DISABLE COUNTER	04
CLEAR COUNTER	05
ENABLE/DISABLE EVENT TABLE ENTRY	06
ENABLE/DISABLE EVENT ENTRY GROUP	07
READ AND HOLD COUNTER VALUE	08

The control byte contains the following information:

Control Byte

bit 0	:	0	=	Event Latch-No Match
		1	=	Event Latch-Match
bit 1	:	0	=	Match Latch-No Match
		1	=	Match Latch-Match
bit 2	:	0	=	Null Entry-Not Scanned
		1	=	Valid Entry-Scan Active
bit 3	:	0	=	Watchdog Monitoring Disabled
		1	=	Watchdog Monitoring Enabled
bit 4	:	0	=	Counter/MOMO Compare-No Match
		1	=	Counter/MOMO Compare-Match
bit 5	:	0	=	Not The Last Entry
		1	=	Last Entry
bit 6	:	0	=	Interrupt Disabled
		1	=	Interrupt Enabled
bit 7	:	0	=	Event Scanning Disabled
		1	=	Event Scanning Enabled

SEND PARAMETERS:

ADDRESS: Address of I/O Unit

COMMAND: 608

POSITION[0]: Event Table Entry Number (0-FF-Hex).

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error
DATA[0]:	Low Word Reaction MOMO/Delay
DATA[1]:	High Word Reaction MOMO/Delay
DATA[2]:	Low Word Compare MOMO/Counter
DATA[3]:	High Word Compare MOMO/Counter
DATA[4]:	High Byte Bit 0 = Enable/Disable Flag Low Byte = Reserved
DATA[5]:	High Byte = Reaction Command Low Byte = Reaction Channel
DATA[6]:	High Byte = Control Byte Low Byte = Compare Channel

READ EVENT LATCHES (P) 609

DESCRIPTION:

This command returns 16 status bits representing the status of 16 event latches starting with the specified event number. Each bit corresponds to the status of one event latch. If the bit is 1, the event latch is set; if the bit is 0, the event latch is cleared.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	609
Positions[0]:	Event Table Entry Number (00-FF-Hex).

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error
DATA[0]:	Status of 16 event latches. Range: 16-bit (0-65,535 or 0000-FFFF-Hex).

READ ALL EVENT LATCHES (P) 610

DESCRIPTION:

This command returns 255 status bits representing the status of all event latches in the Event/Reaction Table. Each bit corresponds to the status of one event latch. If the bit is 1, the latch is set; if the bit is 0, the latch is cleared.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	610

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

DATA[0]...
DATA[15]:
	Status of 16 latches.
	Range: 16-bit (0-65,535 or 0000-FFFF-Hex).
	Data[0] = Event Latches 0-15, and
	Data[15] = Event Latches 240-255.

READ AND CLEAR EVENT LATCHES (Q) 611

DESCRIPTION:

This command returns 16 status bits representing the status of 16 event latches starting with the specified event number. Each bit corresponds to the status of one event latch. If the bit is 1, the event latch is set; if the bit is 0, the latch is cleared. The latch is cleared after it is read.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	611
Positions[0]:	Event Table Entry Number (00-FF-Hex).

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error
DATA[0]:	Status of 16 event latches. Range: 16-bit (0-65,535 or 0000-FFFF Hex).

READ AND CLEAR ALL EVENT LATCHES (Q) 612

DESCRIPTION:

This command returns 255 status bits representing the status of all event latches in the Event/Reaction Table. Each bit corresponds to the status of one event latch. If the bit is 1, the latch is set; if the bit is 0, the latch is cleared. The latches are all cleared after they are read.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	612

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

DATA[0]...
DATA[15]:
	Status of 16 latches.
	Range: 16-bit (0-65,535 or 0000-FFFF-Hex).
	Data[0] = Event Latches 0-15, and
	Data[15] = Event Latches 240-255.

ENABLE EVENT INTERRUPT (I) 613

DESCRIPTION:

This command is used to enable the interrupt output function of an event table entry.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	613
POSITIONS[0]:	Event Table Entry Number (00-FF-Hex).

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

DISABLE EVENT INTERRUPT (I) 614

DESCRIPTION:

This command is used to disable the interrupt output function of an event table entry.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	614
POSITION[0]:	Event Table Entry Number (00-FF-Hex).

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

SET EVENT ON COMM LINK WATCHDOG TIMEOUT (y) 615

DESCRIPTION:

This command is used to add an event entry to the table that monitors a watchdog timeout condition.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	615
POSITION[0]:	Event Table Entry Number (00-FF-Hex).

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

SET EVENT ON MOMO MATCH (K) 616

DESCRIPTION:

This command is used to add an event entry to the table that looks for a match of specified patterns of modules which must be on and which must be off.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	616
POSITION[0]:	Event Table Entry Number (00-FF-Hex).
DATA[0]:	Bitmask of Modules which must be ON. Bitmask range is 0000-FFFF-Hex.
DATA[1]:	Bitmask of Modules which must be OFF. Bitmask range is 0000-FFFF-Hex.

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

SET EVENT ON COUNTER GREATER OR EQUAL (L) 617

DESCRIPTION:

This command is used to add an event entry to the table that looks for a counter value to be greater than or equal to the specified value.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	617
POSITION[0]:	Event Table Entry Number (00-FF-Hex).
POSITION[1]:	Counter Channel 0-15 (00-0F-Hex).
DATA[0]:	Counter Compare Value; Range is 0 to 4,294,967,295 counts.

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

CLEAR REACTION (M) 618

DESCRIPTION:

This command is used to clear an existing reaction from an event/reaction table entry.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	618
POSITION[0]:	Event Table Entry Number (00-FF-Hex).

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

SET OUTPUT STATE REACTION (M) 619

DESCRIPTION:

This command is used to add a reaction entry to the event/reaction table which sets the outputs to a specified condition when the event occurs.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	619
POSITION[0]:	Event Table Entry Number (00-FF-Hex).
DATA[0]:	Bitmask of Modules to turn ON. Bitmask range is 0000-FFFF-Hex.
DATA[1]:	Bitmask of Modules to turn OFF. Bitmask range is 0000-FFFF-Hex.

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

SET ON PULSE REACTION (M) 620

DESCRIPTION:

This command is used to add a reaction entry to the event/reaction table which generates an ON-Pulse when the event occurs.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	620
POSITION[0]:	Event Table Entry Number (00-FF-Hex).
POSITION[1]:	Pulse Channel 0-15 (00-0F-Hex).
DATA[0]:	Pulse Duration in units of 100 microseconds. Range is 5 to 4,294,967,295 units.

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

SET OFF PULSE REACTION (M) 621

DESCRIPTION:

This command is used to add a reaction entry to the event/reaction table which generates an OFF-Pulse when the event occurs.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	621
POSITION[0]:	Event Table Entry Number (00-FF-Hex).
POSITION[1]:	Pulse Channel 0-15 (00-0F-Hex).
DATA[0]:	Pulse Duration in units of 100 microseconds. Range is 5 to 4,294,967,295 units.

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

SET ENABLE/DISABLE COUNTER REACTION (M) 622

DESCRIPTION:

This command is used to add a reaction entry to the event/reaction table which will enable or disable a counter when the event occurs.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	622
POSITION[0]:	Event Table Entry Number (00-FF-Hex).
POSITION[1]:	Counter Channel 0-15 (00-0F-Hex).
DATA[0]:	0 to disable counter, 1 to enable counter.

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

CLEAR COUNTER REACTION (M) 623

DESCRIPTION:

This command is used to add a reaction entry to the event/reaction table which will clear a counter when the event occurs.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	623
POSITION[0]:	Event Table Entry Number (00-FF-Hex).
POSITION[1]:	Counter Channel 0-15 (00-0F-Hex).

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

SET ENABLE/DISABLE EVENT ENTRY REACTION (M) 624

DESCRIPTION:

This command is used to add a reaction entry to the event/reaction table which will enable or disable an event entry when the event occurs.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	624
POSITION[0]:	Event Table Entry Number (00-FF-Hex).
POSITION[1]:	Event Table Entry Number to be enabled/disabled 0-255 (00-FF-Hex).
DATA[0]:	0 to disable the event entry, 1 to enable the event entry.

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

SET ENABLE/DISABLE EVENT GROUP REACTION (M) 625

DESCRIPTION:

This command is used to add a reaction entry to the event/reaction table which will enable or disable a group or all event entries when the event occurs. A group consists of sixteen events beginning with the event number in *POSITION[1]:*.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	625
POSITION[0]:	Event Table Entry Number (00-FF-Hex).
POSITION[1]:	Event Table Entry Number 0-255 (00-FF-Hex). An entry of 255 will enable/disable an entire group.
DATA[0]:	16-bit Mask of events within group to enable. If bit is 1, event is enabled. Bitmask range is 0000-FFFF-Hex.
DATA[1]:	16-bit Mask of events within group to disable. If bit is 1, event is disabled. Bitmask range is 0000-FFFF-Hex.

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

READ AND HOLD COUNTER VALUE REACTION (M) 626

DESCRIPTION:

This command is used to add a reaction entry to the event/reaction table which will read and store a counter value when the event occurs.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	626
POSITION[0]:	Event Table Entry Number (00-FF-Hex).
POSITION[1]:	Counter Channel 0-15 (00-0F-Hex).

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

READ EVENT HOLD BUFFER (|) 627

DESCRIPTION:

This command is used to read the event hold buffer. It is used in conjunction with Command 626 which sets up a reaction that will place a counter value in an event's hold buffer when an event occurs.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	627
POSITION[0]:	Event Table Entry Number (00-FF-Hex).

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error
DATA[0]:	Hold Buffer Value. Range is 0 to 4,294,967,295.

SET EVENT ON COUNTER LESS OR EQUAL (}) 628

DESCRIPTION:

This command is used to add an event entry to the table that looks for a counter value to be less than or equal to the specified value.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	628
POSITION[0]:	Event Table Entry Number (00-FF-Hex).
POSITION[1]:	Counter Channel 0-15 (00-0F-Hex).
Data [0]:	Counter Compare Value; Range is 0 to 4,294,967,295 counts.

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

READ AND CLEAR EVENT LATCH – SINGLE CHANNEL (zA) 629

DESCRIPTION:

This command returns a control byte which contains the state of the event latch and then optionally clears the latch.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	629
POSITION[0]:	Event Table Entry Number (00-FF-Hex)
DATA[0]:	Any 8-bit value (1-255) other than 0, clears the latch. A value of 0 leaves the latch in its current state.

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error
DATA[0]:	8-bit control byte. If bit 0 is set, the event occurred.

CHAPTER 10

Analog Event/Reaction Commands

CLEAR EVENT/REACTION TABLE (-) 700

DESCRIPTION:

This command is used to clear the entire Event/Reaction table. All event latches and interrupts are also cleared.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	700

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

CLEAR EVENT TABLE ENTRY (\) 701

DESCRIPTION:

This command is used to clear a single entry in the Event/Reaction table.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	701
POSITION[0]:	Event Table Entry Number (00-FF-Hex).

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

READ EVENT ENTRY ENABLE/DISABLE STATUS (v) 702

DESCRIPTION:

This command returns 16 status bits representing the status of 16 events starting with the specified event number. Each bit corresponds to the status of one event. If the bit is 1, the event is enabled; if the bit is 0, the event is disabled.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	702
POSITION[0]:	Event Table Entry Number (00-FF-Hex).

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error
DATA[0]:	Status of 16 events. Range: 16-bit (0-65,535 or 0000-FFFF-Hex).

READ ALL EVENT ENTRIES ENABLE/DISABLE STATUS (v) 703

DESCRIPTION:

This command returns 255 status bits representing the status of all events in the Event/Reaction Table. Each bit corresponds to the status of one event. If the bit is 1, the event is enabled; if the bit is 0, the event is disabled.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	703

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

DATA[0]...
DATA[15]:
	Status of 16 events.
	Range: 16-bit (0-65,535 or 0000-FFFF-Hex).
	Data[0] = Events 0-15, and
	Data[15] = Events 240-255

ENABLE EVENT TABLE ENTRY (N) 704

DESCRIPTION:

This command is used to enable an event table entry.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	704
POSITION[0]:	Event Table Entry Number (00-FF-Hex).

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

DISABLE EVENT TABLE ENTRY (N) 705

DESCRIPTION:

This command is used to disable an event table entry.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	705
POSITION[0]:	Event Table Entry Number (00-FF-Hex).

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

ENABLE/DISABLE EVENT TABLE GROUP ({) 706

DESCRIPTION:

This command is used to set the enable/disable status of an entire event table group of 16 entries.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	706
POSITION[0]:	Event Table Group Number (00-FF-Hex).
DATA[0]:	16-bit Mask of events within group to enable. If bit is 1, event is enabled.
DATA[1]:	16-bit Mask of events within group to disable. If bit is 1, event is disabled.

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

READ EVENT TABLE ENTRY (O) 708

DESCRIPTION:

This command is used to read the contents of an event table entry.
The Reaction Command can have the following value:

NULL REACTION (DO NOTHING)	00
SET DAC MODULE MAGNITUDE, COUNTS	01
SET DAC MODULE MAGNITUDE, ENG. UNITS	02
RAMP DAC OUTPUT TO ENDPOINT	03
ENABLE/DISABLE PID LOOP	04
SET PID LOOP SETPOINT	05
ENABLE/DISABLE EVENT TABLE ENTRY	06
ENABLE/DISABLE EVENT ENTRY GROUP	07
READ AND HOLD I/O DATA	08
SET PID LOOP MIN-MAX OUTPUT LIMITS	09

The control byte contains the following information:

Control Byte

bit 0	:	0	=	Event Latch - No Match
		1	=	Event Latch - Match
bit 1	:	0	=	Match Latch - No Match
		1	=	Match Latch - Match
bit 2	:	0	=	Null Entry - Not Scanned
		1	=	Valid Entry - Scan Active
bit 3	:	0	=	Watchdog Monitoring Disabled
		1	=	Watchdog Monitoring Enabled
bit 4	:	0	=	Compare <=
		1	=	Compare >=
bit 5	:	0	=	Not The Last Entry
		1	=	Last Entry
bit 6	:	0	=	Interrupt Disabled
		1	=	Interrupt Enabled
bit 7	:	0	=	Event Scanning Disabled
		1	=	Event Scanning Enabled

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	708
POSITION[0]:	Event Table Entry Number (00-FF-Hex).

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error
DATA[0]:	Low Word Reaction Ramp Slope
DATA[1]:	High Word Reaction Ramp Slope
DATA[2]:	Low Word Compare Value
DATA[3]:	High Word Compare Value
DATA[4]:	Low Word Reaction DAC Output Or Ramp Endpoint In Counts
DATA[5]:	High Byte = Reaction Command Low Byte = Reaction Channel
DATA[6]:	High Byte = Control Byte Low Byte = Compare Channel

READ EVENT LATCHES (P) 709

DESCRIPTION:

This command returns 16 status bits representing the status of 16 event latches starting with the specified event number. Each bit corresponds to the status of one event latch. If the bit is 1, the event latch is set; if the bit is 0, the event latch is cleared.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	709
POSITION[0]:	Event Table Entry Number (00-FF-Hex).

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error
DATA[0]:	Status of 16 event latches. Range: 16-bit (0-65,535 or 0000-FFFF-Hex).

READ ALL EVENT LATCHES (P) 710

DESCRIPTION:

This command returns 255 status bits representing the status of all event latches in the Event/Reaction Table. Each bit corresponds to the status of one event latch. If the bit is 1, the latch is set; if the bit is 0, the latch is cleared.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	710

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

DATA[0]...
DATA[15]:
	Status of 16 latches.
	Range: 16-bit (0-65,535 or 0000-FFFF-Hex).
	Data[0] = Event Latches 0-15, and
	Data[15] = Event Latches 240-255.

READ AND CLEAR EVENT LATCHES (Q) 711

DESCRIPTION:

This command returns 16 status bits representing the status of 16 event latches starting with the specified event number. Each bit corresponds to the status of one event latch. If the bit is 1, the event latch is set; if the bit is 0, the latch is cleared. The latch is cleared after it is read.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	711
POSITION[0]:	Event Table Entry Number (00-FF-Hex).

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error
DATA[0]:	Status of 16 event latches. Range: 16-bit (0-65,535 or 0000-FFFF-Hex).

READ AND CLEAR ALL EVENT LATCHES (Q) 712

DESCRIPTION:

This command returns 255 status bits representing the status of all event latches in the Event/Reaction Table. Each bit corresponds to the status of one event latch. If the bit is 1, the latch is set; if the bit is 0, the latch is cleared. The latches are all cleared after they are read.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	712

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

DATA[0]...
DATA[15]:
	Status of 16 latches.
	Range: 16-bit (0-65,535 or 0000-FFFF-Hex).
	Data[0] = Event Latches 0-15, and
	Data[15] = Event Latches 240-255.

ENABLE EVENT INTERRUPT (I) 713

DESCRIPTION:

This command is used to enable the interrupt output function of an event table entry.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	713
POSITION[0]:	Event Table Entry Number (00-FF-Hex).

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

DISABLE EVENT INTERRUPT (I) 714

DESCRIPTION:

This command is used to disable the interrupt output function of an event table entry.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	714
POSITION[0]:	Event Table Entry Number (00-FF-Hex).

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

SET EVENT ON COMM LINK WATCHDOG TIMEOUT (y) 715

DESCRIPTION:

This command is used to add an event entry to the table that monitors a watchdog timeout condition.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	715
POSITION[0]:	Event Table Entry Number (00-FF-Hex).

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

SET EVENT ON INPUT/OUTPUT > = SETPOINT (K) 716-725

DESCRIPTION:

This command is used to add an event entry to the table that monitors an input or an output channel, and compares to a specified value for a greater-than or equal to condition.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	716 = Raw Counts 717 = Average Counts 718 = Peak Counts 719 = Low Counts 720 = Totalized Counts 721 = Scaled Value (EUI) 722 = Average Value (EUI) 723 = Peak Value (EUI) 724 = Low Value (EUI) 725 = Totalized Value (EUI)
POSITION[0]:	Event Table Entry Number (00-FF-Hex).
POSITION[1]:	I/O Channel Number 0-15 (00-0F-Hex).
DATA[0]:	Compare Value. Range: For commands 716-720, Data[0] is 16-bit (-32768 to 32767). For commands 721-725, Data[0] is 32-bit (-2,147,483,648 to 2,147,483,647). 32-bit values are in increments of 1/65,536 of the engineering units.

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

SET EVENT ON INPUT/OUTPUT <= SETPOINT (L) 726-735

DESCRIPTION:

This command is used to add an event entry to the table that monitors an input or an output channel and compares to a specified value for a less-than or equal to condition.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	726 = Raw Counts 727 = Average Counts 728 = Peak Counts 729 = Low Counts 730 = Totalized Counts 731 = Scaled Value (EUI) 732 = Average Value (EUI) 733 = Peak Value (EUI) 734 = Low Value (EUI) 735 = Totalized Value (EUI)
POSITION[0]:	Event Table Entry Number (00-FF-Hex).
POSITION[1]:	I/O Channel 0-15 (00-0F-Hex).
DATA[0]:	Compare Value. Range: For commands 716-720, Data[0] is 16-bit (-32768 to 32767). For commands 721-725, Data[0] is 32-bit (-2,147,483,648 to 2,147,483,647). 32-bit values are in increments of 1/65,536 of the engineering units.

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

CLEAR REACTION (M) 736

DESCRIPTION:

This command is used to clear an existing reaction from an event/reaction table entry.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	736
POSITION[0]:	Event Table Entry Number (00-FF-Hex).

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

SET OUTPUT VALUE (COUNTS) REACTION (M) 737

DESCRIPTION:

This command is used to add a reaction entry to the event/reaction table which sets an output to a specified count value when the event occurs.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	737
POSITION[0]:	Event Table Entry Number (00-FF-Hex).
POSITION[1]:	I/O Channel Number 0-15 (00-0F-Hex).
DATA[0]:	Channel Value. Range: 0 to 4095.

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

SET OUTPUT VALUE (ENGINEERING UNITS) REACTION (M) 738

DESCRIPTION:

This command is used to add a reaction entry to the event/reaction table which sets an output to a specified value when the event occurs.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	738
POSITION[0]:	Event Table Entry Number (00-FF-Hex).
POSITION[1]:	I/O Channel Number 0-15 (00-0F-Hex).
DATA[0]:	Channel Value. Range: 32-bit (-2,147,483,648 to 2,147,483,647). 32-bit values are in increments of 1/65,536 of the engineering units.

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

SET OUTPUT RAMP TO SETPOINT REACTION (M) 739

DESCRIPTION:

This command is used to add a reaction entry to the event/reaction table which causes an output to ramp to a specified setpoint value when the event occurs.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	739
POSITION[0]:	Event Table Entry Number (00-FF-Hex).
POSITION[1]:	I/O Channel Number 0-15 (00-0F-Hex).
DATA[0]:	Ramp Endpoint. Range: 32-bit value (-2,147,483,648 to +2,147,483,647) in increments of 1/65,536 of an engineering unit.
DATA[1]:	Ramp Slope (Engineering Units per Sec). Range: 32-bit value (-2,147,483,648 to +2,147,483,647) in increments of 1/65,536 of an engineering unit. Slope of 0 is not allowed.

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

SET ENABLE/DISABLE EVENT ENTRY REACTION (M) 740

DESCRIPTION:

This command is used to add a reaction entry to the event/reaction table which will enable or disable an event entry when the event occurs.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	740
POSITION[0]:	Event Table Entry Number (00-FF-Hex).
POSITION[1]:	Event Entry Number 0-254 (00-FE-Hex).
DATA[0]:	0 to disable the event entry, 1 to enable the event entry.

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

SET ENABLE/DISABLE EVENT GROUP REACTION (M) 741

DESCRIPTION:

This command is used to add a reaction entry to the event/reaction table which will enable or disable a group or all event entries when the event occurs.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	741
POSITION[0]:	Event Table Entry Number (00-FF-Hex).
POSITION[1]:	Event Entry Number 0-255 (00-FF-Hex). An entry of 255 will enable/disable an entire group.
DATA[0]:	16-bit Mask of events within group to enable. If bit is 1, event is enabled. Bitmask range is 0000-FFFF-Hex.
DATA[1]:	16-bit Mask of events within group to disable. If bit is 1, event is disabled. Bitmask range is 0000-FFFF-Hex.

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

SET PID SETPOINT VALUE REACTION (M) 742

DESCRIPTION:

This command is used to add a reaction entry to the event/reaction table which sets a PID's setpoint to a specified value when the event occurs.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	742
POSITION[0]:	Event Table Entry Number (00-FF-Hex).
POSITION[1]:	Loop Number 0-7.
DATA[0]:	Setpoint Value. Range: 32-bit (-2,147,483,648 to 2,147,483,647). 32-bit values are in increments of 1/65,536 of the engineering units.

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

SET PID MIN/MAX OUTPUT LIMITS REACTION (M) 743

DESCRIPTION:

This command is used to add a reaction entry to the event/reaction table which sets the min/max limits of a PID's output to specified values when the event occurs.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	743
POSITION[0]:	Event Table Entry Number (00-FF-Hex).
POSITION[1]:	Loop Number 0-7.
DATA[0]:	Maximum Output. Range: 32-bit value (-2,147,483,648 to +2,147,483,647) in increments of 1/65,536 of an engineering unit.
DATA[1]:	Minimum Output. Range: 32-bit value (-2,147,483,648 to +2,147,483,647) in increments of 1/65,536 of an engineering unit.

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

SET PID ENABLE/DISABLE REACTION (M) 744

DESCRIPTION:

This command is used to add a reaction entry to the event/reaction table which will enable or disable a PID loop when the event occurs.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	744
POSITION[0]:	Event Table Entry Number (00-FF-Hex).
POSITION[1]:	Loop Number 0-7.
DATA[0]:	Enable/Disable Value. Range: 8-bit 0= PID Loop Disabled, 1 = PID Loop Enabled.

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

READ AND HOLD I/O VALUE REACTION (M) 745

DESCRIPTION:

This command is used to add a reaction entry to the event/reaction table which will read and store an I/O value when the event occurs.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	745
POSITION[0]:	Event Table Entry Number (00-FF-Hex).
POSITION[1]:	I/O Channel Number 0-15 (00-0F-Hex).

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error

READ EVENT HOLD BUFFER (|) 746

DESCRIPTION:

This command is used to read the event hold buffer. It is used in conjunction with Command 745 which sets up a reaction that will place an I/O value in an event's hold buffer when an event occurs.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	746
POSITION[0]:	Event Table Entry Number (00-FF-Hex).

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error
DATA[0]:	Hold Buffer Value. Range is 32-bit (-2,147,483,648 to 2,147,483,647). 32-bit values are in increments of 1/65,536 of the engineering units.

READ AND CLEAR EVENT LATCH – SINGLE CHANNEL (zA) 747

DESCRIPTION:

This command returns a control byte which contains the state of the event latch and then optionally clears the latch.

SEND PARAMETERS:

ADDRESS:	Address of I/O Unit
COMMAND:	747
POSITION[0]:	Event Table Entry Number (00-FF-Hex)
DATA[0]:	Any 8-bit value (1-255) other than 0, clears the latch. A value of 0 leaves the latch in its current state.

RECEIVE PARAMETERS:

ERROR:	Driver or I/O Error
DATA[0]:	8-bit control byte. If bit 0 is set, the event occurred.

CHAPTER 11

Driver Commands

SET TURNAROUND DELAY TIME 901

DESCRIPTION:

This command sets the delay time the driver will wait before retrying or reporting a timeout error during communications.

SEND PARAMETERS:

COMMAND:	901
DATA[0]:	Delay Time in 10 mSec Units. Range: 16-bit, 1 - 65,535 (10 mSec - 655.35 Sec.) Default is 300 (3 Sec.).

RECEIVE PARAMETERS:

ERROR:	Driver Error

SET PORT AND BAUD RATE 902

DESCRIPTION:

This command sets the port and baud rate the driver will use for communications.

SEND PARAMETERS:

COMMAND:	902
DATA[0]:	Port Number. Range: 1-4 1 = COM1 @ 3F8 Hex, IRQ 4 2 = COM2 @ 2F8 Hex, IRQ 3 3 = COM3 @ 348 Hex, IRQ 2 4 = COM4 @ 340 Hex, IRQ 5
DATA[1]:	Baud Rate. Range: 1-14 1 = 300 Baud 2 = 600 Baud 3 = 1,200 Baud 4 = 2,400 Baud 5 = 4,800 Baud 6 = 9,600 Baud 7 = 19,200 Baud 8 = 38,400 Baud 9 = 57,600 Baud 10 = 76,800 Baud 11 = 115,200 Baud 12 = 138,240 Baud 13 = 172,800 Baud 14 = 230,400 Baud

RECEIVE PARAMETERS:

ERROR:	Driver Error

SET NUMBER OF RETRIES 903

DESCRIPTION:

This command sets the number of retries that the driver will use before reporting an error during communications.

SEND PARAMETERS:

COMMAND:	903
DATA[0]:	Number Of Retries. Range: 16-bit, 0 - 32,767. If entered value is out of range, a value of 0 will be used.

RECEIVE PARAMETERS:

ERROR:	Driver Error

SET COMMUNICATIONS MODE 904

DESCRIPTION:

This command sets the mode the driver will use for communications. Three modes are supported; Binary, ASCII, and PASS-THRU. The PASS-THRU mode sets up the driver to work in a pass-thru mode with a Mistic Controller between the host computer and the Mistic I/O Units.
Any command sent after setting mode 2 will contain a preamble which causes the Mistic Controller to pass the command to a multifunction I/O unit, receive the response and pass it back to the host. Mode 3 is used when using Local Simple I/O units connected to the Mistic Controller .

SEND PARAMETERS:

COMMAND:	904
DATA[0]:	Mode Value. Range: 0 - 3. 0 = BINARY 1 = ASCII 2 = PASS-THRU (16-bit CRC) Multifunction I/O Units 3 = PASS-THRU (8-bit Checksum) Local, Simple I/O Units. The following entries apply only when Data[0] = 2 or 3.
DATA[1]:	Port on the Mistic Controller which is connected to the Mistic I/O units. Range: 0 - 3 (remotes), 6 (local).
DATA[2]:	Address of Mistic Controller. Range: 1-255.

RECEIVE PARAMETERS:

ERROR:	Driver Error

SET USER DEFINED PORT PARAMETERS 905

DESCRIPTION:

This command sets the port, interrupt level, and baud rate the driver will use for communications. The port address and interrupt level can be specified.

SEND PARAMETERS:

COMMAND: 905

DATA[0]: Port Address. Range: 16-bit (0000 - FFFF Hex).

Warning: *Be very careful when specifying this address because the driver will not verify this value. A wrong value may cause the PC to crash or perform erratically.*

DATA[1]: Interrupt Line.
Range: 1-15 Make sure that your communications card uses an interrupt line that is not used by another adapter in the system.

DATA[2]: Baud Rate.
Range: 1-14
1 = 300 Baud
2 = 600 Baud
3 = 1,200 Baud
4 = 2,400 Baud
5 = 4,800 Baud
6 = 9,600 Baud
7 = 19,200 Baud
8 = 38,400 Baud
9 = 57,600 Baud
10 = 76,800 Baud
11 = 115,200 Baud
12 = 138,240 Baud
13 = 172,800 Baud
14 = 230,400 Baud

RECEIVE PARAMETERS:

ERROR: Driver Error

READ DRIVER VERSION 906

DESCRIPTION:

This command returns the driver version and release date.

SEND PARAMETERS:

COMMAND:	906

RECEIVE PARAMETERS:

ERROR:	Driver Error
DATA[0]:	Revision Level
DATA[1]:	Revision Level Fraction 10ths
DATA[2]:	Revision Level Fraction 100ths
DATA[3]:	Month
DATA[4]:	Day
DATA[5]:	Year

Appendix A

Channel Data Dump Structures

The Mistic I/O driver provides command 452 to do a raw channel dump of an analog input, output, or PID database stored in the I/O unit. Since this raw channel dump is a series of bytes that does not match the RESP_OBJ structure for returning values to the calling program, an alternate method must be used. This method uses the receive buffer (rbuf[]) directly to transfer the bytes to the appropriate structure variables described below. Several more steps are required to swap the order of the bytes representing long (4-byte) numbers. Examples for transferring the bytes and swapping the orders are also described below.

Command 452 will do a raw channel dump on analog inputs and outputs if you specify channels 0 thru 15. Specifying channels 32 thru 39 will return PID loop data corresponding to loops 0 thru 7.

To access the receive buffer, you must have the following declaration in your program:

```
extern unsigned char rbuf[256];
```

Following are the structure definitions for analog inputs, outputs, and the PID structure:

```
typedef struct ain_data
{
        char            config;         /*MODULE TYPE */
        unsigned char   flags;          /*FLAGS */
        int             ceuv;           /*COUNTS TO ENG. UNITS VECTOR */
        int             ceuv2;          /*VECTOR 2 */
        int             cir;            /*CURRENT INPUT READING */
        int             cia;            /*CURRENT INPUT AVERAGE */
        long            cir_eu;         /*CURRENT INPUT READING, ENG. UNITS
*/
        int             caf;            /*CURRENT AVERAGE FRACTION */
        int             asc;            /*AVERAGE SAMPLE COUNT */
        int             cacv;           /*CURRENT ASIC COUNTER VALUE */
        int             civr;           /*CURRENT INPUT RAW

READING (NO OFFS/GAIN) */
        int             offset;         /*OFFSET */
        int             gain;           /*GAIN COEFFICIENT */
        int             peak_hi;        /*PEAK COUNTS */
        int             peak_lo;        /*LOWEST COUNTS */
        int             reserved1;      /*RESERVED */
        long            scale_hi;       /*SCALING HI LIMIT */
```

```
       long         scale_offs;    /*SCALING OFFSETT */
       long         ceu_gain;      /*COUNTS TO ENG. UNITS GAIN */
       int          reserved2;     /*RESERVED */
       int          reserved3;     /*RESERVED */
       int          reserved4;     /*RESERVED */
       int          total_sr;      /*TOTALIZER SAMPLE RATE */
       int          total_dc;      /*TOTALIZER DOWN COUNTER */
       int          total_cms;     /*TOTALIZER COUNTS MS WORD */
       int          total_cls;     /*TOTALIZER COUNTS LS WORD */
       long         total_eui;     /*TOTALIZER ENG. UNITS */
       int          reserved5;     /*RESERVED */

}      AIN_OBJ,     *AIN_OBJ_PTR;

typedef struct aout_data
{
       char           config;      /*MODULE TYPE */
       unsigned char  flags;       /*FLAGS */
       int            ceuv;        /*COUNTS TO ENG. UNITS VECTOR */
       int            ceuv2;       /*VECTOR 2 */
       int            cor;         /*CURRENT OUTPUT READING */
       int            por;         /*PENDING OUTPUT READING */
       long           cor_eu;      /*CURRENT OUTPUT READING ENG. UNITS
*/
       long           por_eu;      /*PENDING OUTPUT READING ENG. UNITS
*/
       int            rampf;       /*RAMPING OR PID OUT FRACTIONAL
REMAINDER */
       int            ramp_end;    /*RAMP ENDPOINT COUNTS */
       long           ramp_eu;     /*RAMP ENG. UNITS */
       int            peak_hi;     /*PEAK COUNTS */
       int            peak_lo;     /*LOWEST COUNTS */
       int            tpo_scale;   /*TPO PRESCALE FOR G4DA9 */
       long           scale_hi;    /*SCALING HI LIMIT */
       long           scale_offs;  /*SCALING OFFSETT */
       long           ceu_gain;    /*COUNTS TO ENG. UNITS GAIN */
       long           euc_gain;    /*ENG. UNITS TO COUNTS GAIN */
       int            reserved2;   /*RESERVED */
       int            total_sr;    /*TOTALIZER SAMPLE RATE */
       int            total_dc;    /*TOTALIZER DOWN COUNTER */
       int            total_cms;   /*TOTALIZER COUNTS MS WORD */
       int            total_cls;   /*TOTALIZER COUNTS LS WORD */
       long           total_eui;   /*TOTALIZER ENG. UNITS */
       int            wdog;        /*WATCHDOG VALUE IN COUNTS */
}      AOUT_OBJ,  *AOUT_OBJ_PTR;
typedef struct pid_data_raw
{
       unsigned char  control;     /*PID CONTROL BYTE */
       unsigned char  flags;       /*FLAGS */
       int            reserved1;   /*RESERVED */
       int            in_ptr;      /*INPUT CHANNEL POINTER */
       int            out_ptr;     /*OUTPUT CHANNEL POINTER */
       int            setp_ptr;    /*SETPOINT CHANNEL POINTER */
       long           gain;        /*GAIN COEFFICIENT */
       unsigned long  deriv;       /*DERIVATIVE TIME */
       unsigned long  integral;    /*INTEGRAL - RATES/MIN. */
       long           setpoint;    /*SETPOINT */
       long           p_error;     /*PREVIOUS ERROR */
       long           p2_error;    /*2ND PREVIOUS ERROR */
       long           range_calc;  /*256 / INPUT RANGE ENG. UNITS */
       long           setp_max;    /*MAX SETPOINT LIMIT */
       long            setp_min;    /*MIN SETPOINT LIMIT */
       unsigned long   intg_scan;   /*INTEGRAL / SCAN */
       unsigned long   derv_scan;   /*DERIVATIVE / SCAN */
       int             scan_time;   /*SCAN TIME IN .1 SEC UNITS */
       int             scan_dcnt;   /*SCAN DOWNCOUNTER */
       long            output;      /*OUTPUT COUNTS */
       int             output_hi;   /*OUTPUT COUNTS HIGH LIMIT */
       int             output_lo;   /*OUTPUT COUNTS LOW LIMIT */
       int             reserved2;   / *RESERVED */
       long            reserved3;   /*RESERVED */
       long            reserved4;   /*RESERVED */
       long            reserved5;   /*RESERVED */
}      PID_RAW_OBJ, *PID_RAW_OBJ_PTR;
```

The following are typical variable declarations:

```
AIN_OBJ       ain;
AOUT_OBJ      aout;
PID_OBJ       pid;
PID_RAW_OBJ   pid_raw;
```

The following are the routines to transfer data from the receive buffer to the structured variables:

```
/
**********************************************************************
*      Function      void swap_long(unsigned long *value)
*      Description   swaps the byte order of a 4 byte long at the
*                    location
*                    pointed at by the vaiable value.
**********************************************************************/
void swap_long(unsigned long *value)
{
       unsigned long hi_value;
       unsigned long lo_value;

       hi_value = (*value & 0xFFFFl)<< 16;
       lo_value = (*value >>16);

       *value = hi_value | lo_value;
}

/
**********************************************************************
*      Function      int read_ain_data(unsigned char address,char
*                    channel)
*      Description   reads the raw channel data for an input channel
*                    at the address specified and updates the ain
*                    structure variable.
**********************************************************************/

*/ int read_ain_data(unsigned char address, char channel)
{

struct SEND_OBJ driver_go;
struct RESP_OBJ driver_arrive;
char *data_ptr;
unsigned char *rbuf_ptr;
int i;

       driver_arrive.error = 0;
       driver_go.command = 452;
```

```
        driver_go.address = address;
        driver_go.position[0] = channel;

        data_ptr = (char *)&ain;             /*point to data struc */

        if ((driver_err = g4driver(&driver_go,&driver_arrive)) == NULL)
        {
                /*get the ain stuff */

                rbuf_ptr = rbuf;             /*point to raw receive
buffer */
                if (msg.type == 0)           /*skip past the length and
error bytes */
                        rbuf_ptr += 2;
                else
                        rbuf_ptr++;

                /*convert byte for byte straight from receive buffer */

                for (i=0;i<64;i++,data_ptr++)
                        rbuf_ptr = convert(rbuf_ptr,(unsigned char
*)data_ptr,0x10,1);

                /* fix all the longs by swapping low and high order
words
*/

                swap_long((unsigned long *)&ain.cir_eu);
                swap_long((unsigned long *)&ain.scale_hi);
                swap_long((unsigned long *)&ain.scale_offs);
                swap_long((unsigned long *)&ain.ceu_gain);
                swap_long((unsigned long *)&ain.total_eui);

        }
        return(driver_arrive.error);

}

/
******************************************************************************
*       Function        int read_aout_data(unsigned char address,char
*                       channel)
*
*       Description     reads the raw channel data for an output channel
*                       at the address specified and updates the aout
*                       structure variable.
*****************************************************************************/
int read_aout_data(unsigned char address, char channel)
{

struct SEND_OBJ driver_go;
struct RESP_OBJ driver_arrive;
char *data_ptr;
unsigned char *rbuf_ptr;
int i;
```

```
        driver_arrive.error = 0;
        driver_go.command = 452;
        driver_go.address = address;
        driver_go.position[0] = channel;

        data_ptr = (char *)&aout;           /*point to data struc */

        if ((driver_err = g4driver(&driver_go,&driver_arrive)) == NULL)
        {
                /*get the aout stuff */

                rbuf_ptr = rbuf;            /*point to raw receive
buffer */
                if (msg.type == 0)          /*skip past the length and
error bytes */
                        rbuf_ptr += 2;
                else
                        rbuf_ptr++;

                /*convert byte for byte straight from receive buffer */

                for (i=0;i<64;i++,data_ptr++)
                        rbuf_ptr = convert(rbuf_ptr,(unsigned char
*)data_ptr,0x10,1);

                /*fix all the longs by swapping low and high order words
*/

                swap_long((unsigned long *)&aout.cor_eu);
                swap_long((unsigned long *)&aout.por_eu);
                swap_long((unsigned long *)&aout.ramp_eu);
                swap_long((unsigned long *)&aout.scale_hi);
                swap_long((unsigned long *)&aout.scale_offs);
                swap_long((unsigned long *)&aout.ceu_gain);
                swap_long((unsigned long *)&aout.euc_gain);
                swap_long((unsigned long *)&aout.total_eui);

        }
        return(driver_arrive.error);

}

/
*********************************************************************
*       Function        int read_pid_raw_data(unsigned char
*                       address,char loop)
*
*       Description     reads the raw pid data for a loop at the address
*                       specified and updates the pid_raw structure
*                       variable.
********************************************************************/
int read_pid_raw_data(unsigned char address, char loop)
{

struct SEND_OBJ driver_go;
struct RESP_OBJ driver_arrive;
char *data_ptr;
unsigned char *rbuf_ptr;
int i;
```

```
        driver_arrive.error = 0;
        driver_go.command = 452;
        driver_go.address = address;
        driver_go.position[0] = 0x20 | loop;

        data_ptr = (char *)&pid_raw;           /*point to data struc */

        if ((driver_err = g4driver(&driver_go,&driver_arrive)) == NULL)
        {
                /*get the ain stuff */

                rbuf_ptr = rbuf;               /*point to raw receive
buffer */
                if (msg.type == 0)             /*skip past the length and
error bytes */
                        rbuf_ptr += 2;
                else
                        rbuf_ptr++;

                /*convert byte for byte straight from receive buffer */

                for (i=0;i<80;i++,data_ptr++)
                        rbuf_ptr = convert(rbuf_ptr,(unsigned char
*)data_ptr,0x10,1);

                /*fix all the longs by swapping low and high order words
*/

                swap_long((unsigned long *)&pid_raw.gain);
                swap_long((unsigned long *)&pid_raw.deriv);
                swap_long((unsigned long *)&pid_raw.integral);
                swap_long((unsigned long *)&pid_raw.setpoint);
                swap_long((unsigned long *)&pid_raw.p_error);
                swap_long((unsigned long *)&pid_raw.p2_error);
                swap_long((unsigned long *)&pid_raw.range_calc);
                swap_long((unsigned long *)&pid_raw.setp_max);
                swap_long((unsigned long *)&pid_raw.setp_min);
                swap_long((unsigned long *)&pid_raw.intg_scan);
                swap_long((unsigned long *)&pid_raw.derv_scan);
                swap_long((unsigned long *)&pid_raw.output);

        }
        return(driver_arrive.error);
}
```

The following is an example function for using the data in the PID structure variable as declared earlier:

```
/
****************************************************************************
*
* Function          void display_pid_raw(void)
*
* Description       displays the pid values using the raw structure
*
***************************************************************************/
void display_pid_raw(void)
{
   printf (" ------ LOOP %1d EXTENDED PARAMETERS ----- \n",LOOP);
   printf(" CONTROL BYTE (HEX): %-02X \n",pid_raw.control);
```

```
  printf(“ FLAGS BYTE (HEX): %-02X \n”,pid_raw.flags);
  printf(“ SCAN RATE (SEC.) : %-6d \n”,pid_raw.scan_time/10);
  printf(“ SCAN DOWNCOUNTER : %-6d \n”,pid_raw.scan_dcnt);
  printf(“ INPUT CH. PTR : %-4.4X \n”,pid_raw.in_ptr);
  printf(“ SETPOINT CH. PTR : %-4.4X \n”,pid_raw.setp_ptr);
  printf(“ OUTPUT CH. PTR : %-4.4X \n”,pid_raw.out_ptr);
  printf(“ SETPOINT : %-8.4f \n”,(float)pid_raw.setpoint/65536.0);
  printf(“ OUTPUT : %-8.4f \n”,(float)pid_raw.output/65536.0);
  printf(“ OUTPUT HI COUNTS : %-6d \n”,pid_raw.output_hi);
  printf(“ OUTPUT LO COUNTS : %-6d \n”,pid_raw.output_lo);
  printf(“ GAIN : %-8.4f \n”,(float)pid_raw.gain/65536.0);
  printf(“ INTEGRAL : %-8.4f \n”,(float)pid_raw.integral/65536.0);
  printf(“ DERIVATIVE : %-8.4f \n”,(float)pid_raw.deriv/65536.0);
  printf(“ 1ST ERROR TERM : %-8.4f \n”,(float)pid_raw.p_error/
65536.0);
  printf(“ 2ND ERROR TERM: %-8.4f \n”,(float)pid_raw.p2_error/
65536.0);
  printf(“ INTEGRAL/SCAN: %-8.4f \n”,(float)pid_raw.intg_scan/
65536.0);
  printf(“ DERIVATIVE/SCAN: %-8.4f \n”,(float)pid_raw.derv_scan/
65536.0);
  printf(“ RANGE CALC : %-8.4f \n”,(float)pid_raw.range_calc/65536.0);
  printf(“ SETPOINT MAX : %-8.4f \n”,(float)pid_raw.setp_max/65536.0);
  printf(“ SETPOINT MIN : %-8.4f \n”,(float)pid_raw.setp_min/65536.0);
  printf(“\n”);

}
```

Appendix B

MisticWare Driver for Windows

The MwDriver.DLL allows a Windows application to communicate with:

- Mistic controller
- Mistic I/O or SNAP I/O™ using the SendMIO API.

The driver is implemented as a Windows DLL providing a set of "C" APIs and may be used with any Windows language that supports DLLs such as Visual Basic or C/ C++ compilers.

Port Locking and Multiple Applications

Port locking is implemented so that multiple applications can share the same port. If an application needs exclusive access to a port to perform multiple commands without interruption, an API is provided to lock and unlock a port.

Port Types Supported

This driver supports the following Opto 22 adapter cards:

- **AC37** RS-485 buffered coprocessor card capable of either 2-wire or 4-wire RS-485 at up to 115.2 kbaud.
- **AC39** Local bus buffered coprocessor card.
- **AC42** Fiber optic RS-485 buffered coprocessor card.

This driver also supports the use of Standard Windows COM ports (via the Windows API's). This includes the use of the following Opto 22 adapter cards:

- **AC7A** external RS-232 to RS-485 converter (capable of either 2-wire or 4-wire RS-485).
- **AC24AT or AC422AT** RS-485 cards capable of 4-wire RS-485 at up to 38.4 kbaud.

Handles

MwDriver uses handles to access ports. A handle is provided when a port is opened and thereafter the handle is used to access a port.

Installation

The file MwDriver.Dll should be copied to either the application directory or the Windows System directory. Windows searches for a DLL in the following sequence: [1] in memory, [2] the current working directory, and [3] Windows System directory.

Header files (MwDriver.H for C and MwDriver.Bas for Basic) should be copied to the directory where the application source code is located.

The file MwDriver.Lib is provided as an import library. This tells the linker that the MwDriver APIs are part of a DLL. If this LIB file is used, the MwDriver APIs do not need to be included in the “imports” section of the application’s DEF file. The LIB file is not required for Visual Basic.

API List

For most applications, only 3 APIs in the library are required as follows:

- Open a port and get a port handle
- Read or write data
- Close the port and release the port handle
- See the code fragment below shows a simple example without error handling.

Simple Code Fragment Using These APIs

```
// Step [1] - Open a port and get a port handle.
// There’s a separate ‘open’ API for ARCNET, AC37 and a COM port.
iErrorCode = opto22MwdPortOpenXXX( &MwdHandle, ...);

// Step [2a] - For Mistic/SNAP, call SendMIO to interact with I/O.
iErrorCode = SendMIO( MwdHandle, ...);

// Step [2b] - For Mistic controllers, send Commands and get
responses.
iErrorCode = opto22MwdSendReceNoId( MwdHandle, ...);
printf( “\″%s\″, %d chars, Error:%d\n″, ReceStr,iActualLen,iErrorCode
);

// Step [3] Close the port when the application ends.
opto22MwdPortClose( MwdHandle );
```

Notes about APIs:

- The APIs are listed below with the function prototypes and a description.
- Refer to MwDriver.H for actual function prototypes or MwDriver.BAS for function declarations.

- In general, an API that returns an “int” returns an error number where 0 indicates “no error”.

Each API name starts with “opto22Mwd**” except for** SendMIO.

opto22MwdGetVersion, opto22MwdGetVersion2

```
char* FAR PASCAL opto22MwdGetVersion( void );

int opto22MwdGetVersion2(
   char* versionArg,
   UINT  maxLenArg );
```

The “get version” APIs get a version string of the form “1.2.3”. The version string is also in the MwDriver.H file (or MwDriver.BAS for Basic). The purpose of these functions is to allow an application to check the version of the DLL file. **opto22MwdGetVersion** returns a pointer to a string and **opto22MwdGetVersion2** copies the version string to a buffer provided by the caller.

opto22MwdSendRece, opto22MwdSendReceNoId

```
int opto22MwdSendRece(
   int    HandleArg,
   UINT   addressArg,
   BYTE*  sendStringArg,
   UINT   sendLength,
   BYTE*  receStringArg,
   UINT   receStringLenMaxArg,
   UINT*  receStringLenActualArg );

int opto22MwdSendReceNoId(
   int    HandleArg,
   UINT   addressArg,
   BYTE*  sendStringArg,
   UINT   sendLength,
   BYTE*  receStringArg,
   UINT   receStringLenMaxArg,
   UINT*  receStringLenActualArg );
```

Does both a send and a receive. The “NoId” does not do sequence ID checking which if faster. Usually, sequence ID checking is required on noisy ARCNET installations. Sequence ID checking cannot be used with mistic commands that have an imbedded “\r” (carriage return). Typically, the “NoId” API is sufficient.

SendMIO

```
int SendMIO(
  int                 iHandle,          // Handle provided by ‘port open’
  int                 iAddress,         // Address of brick
  unsigned int        iCommand,         // Command number i.e. 202=set
output
   unsigned int far* PositionArray,    // Position array - 2 elements
   long far*          SendDataArray,    // Send data array - 16 elements
   long far*          ReceDataArray);  // Received data array - 16
elements
```

This API is used to access Mistic/SNAP I/O. The command numbers are documented in the *MisticWare User's Guide*.

opto22MwdPortLock

```
int  opto22MwdPortLock(
   int    handle,
   BOOL   bLockState );
```

This API locks and unlocks a port for exclusive access by the specified handle. This is required when multiple commands need to be executed by one application before allowing another application to execute a command. Under normal conditions, this API is not required.

opto22MwdPortOpen for ARCNET, AC37 and WinApi

```
int  opto22MwdPortOpenARCNET(
   int*   handle,
   UINT   ioPort,
   DWORD  memAddr,
   float  timeOut,
   UINT   retry );

int  opto22MwdPortOpenAC37(
   int*   handle,
   UINT   ioPort,  // I/O address such as 3F8 (IRQ not required)
   DWORD  Baud,    // Baud rate such as 115200
   float  timeOut,
   UINT   retry,
   int    protocolType,
   int    dataCheckType );

int  opto22MwdPortOpenWinApi(
   int*   handle,
   UINT   comPort,          // COM port number. 1 for COM1.
   DWORD  Baud,             // Baud rate up to 38400
   float  timeOut,
   UINT   retry,
   int    protocolType,      // Protocol type: ASCII or Binary
   int    dataCheckType );  // Data check type (should be CRC)
```

The “PortOpen” APIs open a port and assign a handle number to be used by the application to access the port.

Parameters common to all “PortOpen” APIs:

> **timeOut** is the amount of time to wait, in seconds, for a response. A typical number to use for ARCNET or high speed serial is 0.75 seconds. Larger values are needed for lower baud rates or if several applications are accessing the same port.
>
> **retry** is the quantity of times the driver will retry a command if an error occurs. The driver always tries once but the number of reties depends on this value. A typical number is 1.
>
> **protocolType** should be either **ProtocolTypeBinary** or **ProtocolTypeAscii** which are defined in MwDriver.H. Windows 3.1 on an RS232 port will not support Binary mode. Typically,

Binary is better because twice as many characters need to be sent for ASCII. ASCII may be required if communicating via modem.

dataCheckType should be **DataCheckTypeCrc16** which is defined in MwDriver.H.

opto22MwdPortClose

```
void   FAR PASCAL opto22MwdPortClose( int HandleArg );
```

This API closes a port and releases the handle. If this API is not called by an application before exiting, the port handle will not be released until the DLL is released from memory. Visual Basic should call this API when the main form unloads.

opto22MwdPortRetrySet , opto22MwdPortDelaySet

```
int  opto22MwdPortRetrySet(
   int    HandleArg,
   UINT   retryArg );

float  FAR  PASCAL  opto22MwdPortDelaySet(
   int    HandleArg,
   float  DelayArg );
```

These two APIs set modify the retry and delay parameters that are set in the "PortOpen" APIs. The return value can be used by the application to restore the setting to its previous value. These APIs are normally not required except for Opto22 utilities.

Visual Basic Functions

StringAsLong, StringAsFloat

```
void FAR  PASCAL  StringAsLong(  long*  numArg,  char*  StringArg  );
void FAR  PASCAL  StringAsFloat(  float*  numArg,  char*  Stringarg  );
```

These two APIs are used by Visual Basic to extract Longs or Floats from a response string that contains binary data. A response from mistic can be ASCII (such as "1.234E12") or binary (which would appear as 4 bytes of garbage characters) depending upon the command. HostWords allows ASCII or Binary mode to be selected. "C" doesn't need these two functions because C can use type casts to convert the data.

Status or Error Codes

The status codes or error codes returned by the driver will be zero if there are no errors. Error codes are listed in the file Errors.H. Some of the more common errors are listed in the table below.

Additional Error Codes

-100 Invalid Port Error

This error occurs when a handle number for a port is specified and that port has not been assigned a driver or adpater type.

-102 Initialize Error

This error occurs if the proper AC37 or AC39 adapter could not be found at the location specified in the ConfigureAC37 or ConfigureAC39 functions.

-104 Port Lock Error

This error occurs when a call to the SendMIO() function is made while another command is being processed. Windows allows multiple applications to call the same DLL function. Since the DLL only has one set of global data he must protect himself by using a lock mechanism to ensure that commands get processed one application at a time. If you encounter this error, you might consider having all scanning controlled by one application, and all other applications communicate to the scanner.

-105 Driver Configuration Error

This error occurs when the SendMIO() is called prior to configuring a port with the AssignPortDriver() and Configure functions.

Appendix C

Using The Driver with Microsoft Basic/Quick Basic

The MisticWare driver can be called from applications written using Microsoft's Basic Compiler 7.0/7.1 and Microsoft Quick Basic 4.0/4.5.

To call the driver from Basic, the driver MUST first be loaded into memory using the program LOADMD.EXE provided on the MisticWare disks. To load the driver, run the program by typing LOADMD followed by the Enter key at the DOS prompt.

Example (assumes LOADMD.EXE is in the current directory):

```
C>LOADMD
```

To unload or remove the driver from memory, type LOADMD quit followed by the Enter key at the DOS prompt.

Example (assumes LOADMD.EXE is in the current directory):

```
C>LOADMD  quit
```

The LOADMD program loads the driver into memory where it becomes accessible using software interrupt 61 hex. Prior to the Basic program issuing the interrupt, the CPU AX, BX, CX, and DX registers are set up with the addresses to the send and receive structures used by the application.

Since the MisticWare driver uses a send and a receive structure to pass parameters, these structures must first be declared and initialized prior to any driver call. The Microsoft Basic compilers allow definition of structures using the keyword TYPE. However, a minor difference exists between the Quick Basic 4.0/4.5 compilers and the Basic compiler 7.0/7.1. Quick Basic does not allow declarations of arrays inside of user- defined types, therefore each array element required by the driver must be a standalone variable (integer for POSITIONS and long for INFO). A special function (CALL INTERRUPT) for issuing a software interrupt is also used. This function resides in the external library QB.LIB (QB.QLB) provided with Quick Basic 4.0/4.5 and QBX.LIB (QBX.QLB) provided with the Basic Compiler 7.0/7.1. Standard Basic functions VARPTR and VARSEG are used to setup the CPU registers before the interrupt call.

NOTE: The version of Basic called QBasic (which is included with MS-DOS 5.0) cannot be used to call the driver because it does not support the INTERRUPT function and type definitions needed to call the driver.

Basic Compiler 7.0/7.1

The following are the necessary declarations and functions needed for calling the driver using the Microsoft Basic Compiler Versions 7.0 and 7.1. The advantage of using this compiler over Quick Basic 4.0/4.5 is that the newer version of the Basic Compiler allows arrays to be defined inside of structure definitions, reducing the amount of code necessary. The following lines must be placed at the beginning of your application source code:

```
`
'**************************************************************************
'
'              DECLARE VARIABLES & SUBROUTINES
'
`**************************************************************************

`
REM $STATIC

TYPE SendRec
       Address AS INTEGER
       Command AS INTEGER
       Position(0 TO 1) AS INTEGER
       Info(0 TO 15) AS LONG
END TYPE



TYPE ReceiveRec
       Error AS INTEGER
       Info(0 TO 15) AS LONG
END TYPE



TYPE RegType
       AX AS INTEGER
       BX AS INTEGER
       CX AS INTEGER
       DX AS INTEGER
       BP AS INTEGER
       SI AS INTEGER
       DI AS INTEGER
       FLAGS AS INTEGER
END TYPE

DIM POSITIONS%(1)              ` DIMENSION TO 2 ELEMENTS
DIM INFO&(15)                  ` DIMENSION TO 16 ELEMENTS
DIM Send AS SendRec
DIM Rcv AS ReceiveRec
DIM Regs AS RegType

COMMON SHARED ERRORS%,ADDRESS%,COMMAND%,POSITIONS%(),INFO&()

`
```

```
'*********************************************************************
'*                                                           *
'*            CALL THE Mistic DRIVER TSR                           *
'*                                                           *
`*********************************************************************
'
mdriver:

ERRORS% = 0

Send.Address           = ADDRESS%
Send.Command           = COMMAND%
Send.Position(0)       = POSITIONS%(0)
Send.Position(1)       = POSITIONS%(1)
FOR I% = 0 TO 15
              Send.Info(I%) = INFO&(I%)
              Rcv.Info(I%) = 0
NEXT

Regs.AX = VARPTR(Send)
Regs.BX = VARSEG(Send)
Regs.CX = VARSEG(Rcv)
Regs.DX = VARPTR(Rcv)

CALL Interrupt(&H61,Regs,Regs)

ERRORS% = Rcv.Error

FOR I% = 0 TO 15
       INFO&(I%) = Rcv.Info(I%)
NEXT

RETURN
```

Before calling the mdriver subroutine, the global parameters ADDRESS%, COMMAND%, POSITIONS%(), and INFO&() must be set with the appropriate values dependent on the command to be issued.

After compiling an application, it is important to link it using the QBX.LIB library that is included with the Basic Compiler. This library contains the code for Basic's INTERRUPT function. The following is an example for the command line used when linking:

```
LINK filename QBX.LIB;
```

where filename is the name of the application program.

Two sample programs are provided on the MisticWare diskettes to illustrate the calling process. They are named MDTSR.BAS and MDTSR1.BAS. The batch files DOTSR.BAT and DOTSR1.BAT are also provided. Those files contain the commands for compiling and linking the respective sample programs.

Quick Basic 4.0/4.5

The method for calling the driver using Quick Basic 4.0/4.5 is the same as that shown previously for the Basic Compiler, with a minor difference. Quick Basic 4.0/4.5 does not allow arrays to be defined within a user-defined type structure, therefore, the array variables are replaced with individual elements of type INTEGER for Position and LONG for Info.

The following are the necessary declarations and functions needed for calling the driver using the Microsoft Quick Basic Compiler Versions 4.0/4.5:

```
’**************************************************************************
’
’              DECLARE  VARIABLES  &  SUBROUTINES
’
’**************************************************************************
’
’  $INCLUDE:  ‘QB.BI’

REM  $STATIC

TYPE  SendRec
       Address  AS  INTEGER
       Command  AS  INTEGER
       Position0  AS  INTEGER
       Position1  AS  INTEGER
       Info0   AS  LONG
       Info1   AS  LONG
       Info2   AS  LONG
       Info3   AS  LONG
       Info4   AS  LONG
       Info5   AS  LONG
       Info6   AS  LONG
       Info7   AS  LONG
       Info8   AS  LONG
       Info9   AS  LONG
       Info10  AS  LONG
       Info11  AS  LONG
       Info12  AS  LONG
       Info13  AS  LONG
       Info14  AS  LONG
       Info15  AS  LONG
END  TYPE

TYPE  ReceiveRec
       Errors  AS  INTEGER
       Info0   AS  LONG
       Info1   AS  LONG
       Info2   AS  LONG
       Info3   AS  LONG
       Info4   AS  LONG
       Info5   AS  LONG
       Info6   AS  LONG
       Info7   AS  LONG
       Info8   AS  LONG

       Info9   AS  LONG
```

```
      Info10 AS LONG
      Info11 AS LONG
      Info12 AS LONG
      Info13 AS LONG
      Info14 AS LONG
      Info15 AS LONG
END TYPE

DIM POSITIONS%(1)                  ` DIMENSION TO 2 ELEMENTS
DIM INFO&(15)                      ` DIMENSION TO 16 ELEMENTS
DIM Send AS SendRec
DIM Rcv AS ReceiveRec
DIM Regs AS RegType

COMMON SHARED ERRORS%,ADDRESS%,COMMAND%,POSITIONS%(),INFO&()

`
'**********************************************************************
'*                                                    *
'*                  CALL THE Mistic DRIVER                        *
'*                                                    *
'**********************************************************************
'
mdriver:

ERRORS%  = 0

Send.Address     = ADDRESS%
Send.Command     = COMMAND%
Send.Position0   = POSITIONS%(0)
Send.Position1   = POSITIONS%(1)
Send.Info0       = INFO&(0)
Send.Info1       = INFO&(1)
Send.Info2       = INFO&(2)
Send.Info3       = INFO&(3)
Send.Info4       = INFO&(4)
Send.Info5       = INFO&(5)
Send.Info6       = INFO&(6)
Send.Info7       = INFO&(7)
Send.Info8       = INFO&(8)
Send.Info9       = INFO&(9)
Send.Info10      = INFO&(10)
Send.Info11      = INFO&(11)
Send.Info12      = INFO&(12)
Send.Info13      = INFO&(13)
Send.Info14      = INFO&(14)
Send.Info15      = INFO&(15)

Regs.ax   = VARPTR(Send)
Regs.bx   = VARSEG(Send)
Regs.cx   = VARSEG(Rcv)
Regs.dx   = VARPTR(Rcv)

CALL INTERRUPT(&H61,Regs,Regs)
```

```
ERRORS%  =  Rcv.Errors

INFO&(0)    =  Rcv.Info0
INFO&(1)    =  Rcv.Info1
INFO&(2)    =  Rcv.Info2
INFO&(3)    =  Rcv.Info3
INFO&(4)    =  Rcv.Info4
INFO&(5)    =  Rcv.Info5
INFO&(6)    =  Rcv.Info6
INFO&(7)    =  Rcv.Info7
INFO&(8)    =  Rcv.Info8
INFO&(9)    =  Rcv.Info9
INFO&(10)   =  Rcv.Info10
INFO&(11)   =  Rcv.Info11
INFO&(12)   =  Rcv.Info12
INFO&(13)   =  Rcv.Info13
INFO&(14)   =  Rcv.Info14
INFO&(15)   =  Rcv.Info15

RETURN
```

If compiling the application using the integrated Quick Basic environment, it is necessary to start Quick Basic with the /l command line switch. This switch causes the extended library QB.LIB (which contains the INTERRUPT function) to be loaded. The following is an example for the command line used:

```
QB  /l
```

If the command line Basic compiler (versions prior to 7.0/7.1) included with Quick Basic 4.0/4.5 is used instead, then it is necessary to link the application using the QB.LIB library that is included with Quick Basic. The following is an example for the command line used when linking:

```
LINK filename QB.LIB; where filename is the name of the application
program.
```

Two sample programs are provided on the MisticWARE diskettes to illustrate the calling process. They are named QBMDTSR.BAS and QBMDTSR1.BAS. The batch files DOQBTSR.BAT and DOQBTSR1.BAT are also provided. They contain the commands for compiling and linking the respective sample programs.

Appendix D

Using Opto 22 I/O in 32 Bit Windows

Both Optomux and Mistic I/O may be accessed in Win32 using OptoMwd.DLL. This DLL allows multiple applications to access I/O simultaneously. This DLL provides a set of "C" APIs and may be used with any 32 bit Windows language that supports DLLs such as Visual Basic or C/ C++ compilers.

Port Locking and Multiple Applications

Port locking is implemented so that multiple applications can share the same port. If an application needs exclusive access to a port to perform multiple commands without interruption, an API is provided to lock and unlock a port.

Port Types Supported

This driver supports the following Opto 22 adapter cards:

- **AC37** RS-485 buffered coprocessor card capable of either 2-wire or 4-wire RS-485 at up to 115.2 kbaud.
- **AC39** Local bus buffered coprocessor card.
- **AC42** Fiber optic RS-485 buffered coprocessor card.

This driver also supports the use of Standard Windows COM ports (via the Windows API's). This includes the use of the following Opto 22 adapter cards:

- **AC7A** external RS-232 to RS-485 converter (capable of either 2-wire or 4-wire RS-485).
- **AC24AT or AC422AT** RS-485 cards capable of 4-wire RS-485 at up to 38.4 kbaud.

Handles

OptoMwd.DLL uses handles to access ports. A handle is provided when a port is opened and thereafter the handle is used to access a port.

Installation

Run the OptoDriver Toolkit installation to install I/O drivers, examples and on-line documentation.

Examples

Examples are provided in the OptoDriver Toolkit for several languages. These examples are installed when the OptoDriver Toolkit is installed.

API List

For most applications, only 3 APIs in the library are required as follows:

- Open a port and get a port handle
- Read or write data
- Close the port and release the port handle

See the code fragment below shows a simple example without error handling.

Simple Code Fragment Using These APIs

```
// Step [1] - Open a port and get a port handle.
// There's a separate 'open' API for Arcnet, AC37 and a COM port.
ErrorCode = opto22MwdPortOpenXXX( &MwdHandle, ...);

// Step [2a] - For Mistic/Snap, call SendMIO to interact with I/O.
ErrorCode = SendMIO( MwdHandle, ...);

// Step [2b] - For Optomux I/O, call SendOptoMux to interact with I/
O.
ErrorCode = SendOptoMux( MwdHandle, ...);
printf( "\"%s\", %d chars, Error:%d\n", ReceStr,iActualLen,iErrorCode
);

// Step [3] Close the port when the application ends.
opto22MwdPortClose( MwdHandle );
```

Notes about APIs:

- The APIs are listed below with the function prototypes and a description.
- Refer to OptoMwd.H for actual function prototypes or OptoMwd.BAS for function declarations.
- In general, an API that returns an "int" returns an error number where 0 indicates "no error".
- Each API name starts with "opto22Mwd" except for SendMIO.

opto22MwdGetVersion, opto22MwdGetVersion2

```
char* FAR PASCAL opto22MwdGetVersion( void );

int opto22MwdGetVersion2(
   char* versionArg,
   UINT  maxLenArg );
```

The “get version” APIs get a version string of the form “R1.9z”. The purpose of these functions is to allow an application to check the version of the DLL file. **opto22MwdGetVersion** returns a pointer to a string and **opto22MwdGetVersion2** copies the version string to a buffer provided by the caller.

SendMIO

```
O22_ERROR_CODE  SendMIO(
   int                   iHandle,   // Handle provided by ‘port open’
   int                   iAddress, // Address of brick
   unsigned int          iCommand, // Command number i.e., 202=set output
   unsigned int far* PositionArray,   // Position array - 2 elements
   long far*             SendDataArray,   // Send data array - 16 elements
   long far*             ReceDataArray); // Recv data array - 16 elements
```

Sends a command to Mistic I/O and gets the corresponding response. Returns an Opto 22 error code where a value of zero indicates no error.

SendOptoMux

```
O22_ERROR_CODE  SendOptoMux(
   int                   iHandle,      // Handle provided by ‘port open’
   int                   iAddress,     // Address of brick
   unsigned int          iCommand,     // Command number i.e., Max is 79
   long far*             cPosition,    // 16 element array
   unsigned int far* ModifierArray,   // Modifier array - 2 elements
   long far*             DataArray);   // Send Data Array, 16 elements
```

Sends a command to Optomux I/O and gets the corresponding response. Returns an Opto 22 error code where a value of zero indicates no error.

opto22MwdPortOpen for AC37 and WinApi

```
int  opto22MwdPortOpenAC37(
   int*   handle,
   UINT   ioPort,  // I/O address such as 3F8 (IRQ not required)
   DWORD  Baud,    // Baud rate such as 115200
   float  timeOut,
   UINT   retry,
   int     protocolType,
   int     dataCheckType );

int  opto22MwdPortOpenWinApi(
   int*   handle,
   UINT   comPort,          // COM port number. 1 for COM1.
   DWORD  Baud,             // Baud rate up to 38400
   float  timeOut,
   UINT   retry,
   int     protocolType,      // Protocol type: ASCII or Binary
   int     dataCheckType );  // Data check type (should be CRC)
```

The “PortOpen” APIs open a port and assign a handle number to be used by the application to access the port.

Parameters common to all “PortOpen” APIs:

timeOut is the amount of time to wait, in seconds, for a response. A typical number to use for Arcnet or high speed serial is 0.75 seconds. Larger values are needed for lower baud rates or if several applications are accessing the same port.

retry is the quantity of times the driver will retry a command if an error occurs. The driver always tries once but the number of reties depends on this value. A typical number is 1.

protocolType should be either **ProtocolTypeBinary** or **ProtocolTypeAscii** which are defined in OptoMwd.H. Windows 3.1 on an RS232 port will not support Binary mode. Typically, Binary is better because twice as many characters need to be sent for ASCII. ASCII may be required if communicating via modem.

dataCheckType should be **DataCheckTypeCrc16** which is defined in OptoMwd.H.

opto22MwdPortClose

```
void opto22MwdPortClose( int HandleArg );
```

This API closes a port and releases the handle. If this API is not called by an application before exiting, the port handle will not be released until the DLL is released from memory. Visual Basic should call this API when the main form unloads.

Visual Basic Functions

StringAsLong, StringAsFloat

```
void FAR PASCAL StringAsLong( long* numArg, char* StringArg );
void FAR PASCAL StringAsFloat( float* numArg, char* Stringarg );
```

These two APIs are used by Visual Basic to extract Longs or Floats from a response string that contains binary data. A response from mistic can be ASCII (such as “1.234E12”) or binary (which would appear as 4 bytes of garbage characters) depending upon the command. HostWords allows ASCII or Binary mode to be selected. “C” doesn’t need these two functions because C can use type casts to convert the data.

Status or Error Codes

Refer to OptoErr.RH for a list of error codes. OptoErr.H and OptoErr.BAS list some APIs that provide an error message given an error code

Index