Agilent 75000 Series C

Agilent E1458A
96-Channel Digital I/O Module

User's Manual and SCPI Programming Guide

Where to Find It - Online and Printed Information:

If your instrument has an Agilent VXI*plug&play* driver, product information is organized as indicated below.

System installation (hardware/software)	VXIbus Configuration Guide* Agilent VIC (VXI installation software)*
Module configuration and wiring	This Manual
SCPI programming	This Manual
SCPI example programs	This Manual
SCPI command reference	This Manual
Register-based programming	This Manual
VXI*plug&play* programming	VXI*plug&play* Online Help
VXI*plug&play* example programs	VXI*plug&play* Online Help
VXI*plug&play* function reference	VXI*plug&play* Online Help
Soft Front Panel information	VXI*plug&play* Online Help
VISA language information	Agilent VISA User's Guide
Agilent VEE programming information	Agilent VEE User's Manual

**Supplied with Agilent Command Modules,Embedded Controllers, and VXLink.*

Agilent Technologies

Manual Part Number: E1458-90002
Printed September 2012
Printed in Malaysia E0912

Contents

Certification

Agilent Technologies certifies that this product met its published specifications at the time of shipment from the factory. Agilent Technologies further certifies that its calibration measurements are traceable to the United States National Institute of Standards and Technology (formerly National Bureau of Standards), to the extent allowed by that organization's calibration facility, and to the calibration facilities of other International Standards Organization members.

Warranty

This Agilent Technologies product is warranted against defects in materials and workmanship for a period of one (1) year from date of shipment. Duration and conditions of warranty for this product may be superseded when the product is integrated into (becomes a part of) other Agilent products. During the warranty period, Agilent Technologies will, at its option, either repair or replace products which prove to be defective.

For warranty service or repair, this product must be returned to a service facility designated by Agilent Technologies. Buyer shall prepay shipping charges to Agilent and Agilent shall pay shipping charges to return the product to Buyer. However, Buyer shall pay all shipping charges, duties, and taxes for products returned to Agilent from another country.

Agilent warrants that its software and firmware designated by Agilent for use with a product will execute its programming instructions when properly installed on that product. Agilent does not warrant that the operation of the product, or software, or firmware will be uninterrupted or error free.

Limitation Of Warranty

The foregoing warranty shall not apply to defects resulting from improper or inadequate maintenance by Buyer, Buyer-supplied products or interfacing, unauthorized modification or misuse, operation outside of the environmental specifications for the product, or improper site preparation or maintenance.

The design and implementation of any circuit on this product is the sole responsibility of the Buyer. Agilent does not warrant the Buyer's circuitry or malfunctions of Agilent products that result from the Buyer's circuitry. In addition, Agilent does not warrant any damage that occurs as a result of the Buyer's circuit or any defects that result from Buyer-supplied products.

NO OTHER WARRANTY IS EXPRESSED OR IMPLIED. Agilent SPECIFICALLY DISCLAIMS THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Exclusive Remedies

THE REMEDIES PROVIDED HEREIN ARE BUYER'S SOLE AND EXCLUSIVE REMEDIES. Agilent SHALL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, WHETHER BASED ON CONTRACT, TORT, OR ANY OTHER LEGAL THEORY.

Notice

The information contained in this document is subject to change without notice. Agilent Technologies MAKES NO WARRANTY OF ANY KIND WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Agilent shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance or use of this material. This document contains proprietary information which is protected by copyright. All rights are reserved. No part of this document may be photocopied, reproduced, or translated to another language without the prior written consent of Agilent Technologies, Inc. Agilent assumes no responsibility for the use or reliability of its software on equipment that is not furnished by Agilent.

U.S. Government Restricted Rights

The Software and Documentation have been developed entirely at private expense. They are delivered and licensed as "commercial computer software" as defined in DFARS 252.227- 7013 (Oct 1988), DFARS 252.211-7015 (May 1991) or DFARS 252.227-7014 (Jun 1995), as a "commercial item" as defined in FAR 2.101(a), or as "Restricted computer software" as defined in FAR 52.227-19 (Jun 1987)(or any equivalent agency regulation or contract clause), whichever is applicable. You have only those rights provided for such Software and Documentation by the applicable FAR or DFARS clause or the Agilent standard software agreement for the product involved.

Agilent E1458A 96-Channel Digital I/O Module User's Manual
Edition 2 Rev 3

Printing History

The Printing History shown below lists all Editions and Updates of this manual and the printing date(s). The first printing of the manual is Edition 1. The Edition number increments by 1 whenever the manual is revised. Updates, which are issued between Editions, contain replacement pages to correct the current Edition of the manual. Updates are numbered sequentially starting with Update 1. When a new Edition is created, it contains all the Update information for the previous Edition. Each new Edition or Update also includes a revised copy of this printing history page. Many product updates or revisions do not require manual changes and, conversely, manual corrections may be done without accompanying product changes. Therefore, do not expect a one-to-one correspondence between product updates and manual updates.

Edition 1 (Part Number E1458-90001). . . . . October 1993

Edition 2 (Part Number E1458-90002). . . . . August 1996

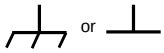
Edition 2 Rev 2 (Part Number E1458-90002) . . . . . April 2006

Edition 2 Rev 3 (Part Number E1458-90002) . . . . . September 2012

Safety Symbols

Instruction manual symbol affixed to product. Indicates that the user must refer to the manual for specific WARNING or CAUTION information to avoid personal injury or damage to the product.

Indicates the field wiring terminal that must be connected to earth ground before operating the equipment—protects against electrical shock in case of fault.



Frame or chassis ground terminal—typically connects to the equipment's metal frame.

Alternating current (AC).

Direct current (DC).

Indicates hazardous voltages.

WARNING Calls attention to a procedure, practice, or condition that could cause bodily injury or death.

CAUTION Calls attention to a procedure, practice, or condition that could possibly cause damage to equipment or permanent loss of data.

WARNINGS

The following general safety precautions must be observed during all phases of operation, service, and repair of this product. Failure to comply with these precautions or with specific warnings elsewhere in this manual violates safety standards of design, manufacture, and intended use of the product. Agilent Technologies assumes no liability for the customer's failure to comply with these requirements.

Ground the equipment: For Safety Class 1 equipment (equipment having a protective earth terminal), an uninterruptible safety earth ground must be provided from the mains power source to the product input wiring terminals or supplied power cable.

DO NOT operate the product in an explosive atmosphere or in the presence of flammable gases or fumes.

For continued protection against fire, replace the line fuse(s) only with fuse(s) of the same voltage and current rating and type. DO NOT use repaired fuses or short-circuited fuse holders.

Keep away from live circuits: Operating personnel must not remove equipment covers or shields. Procedures involving the removal of covers or shields are for use by service-trained personnel only. Under certain conditions, dangerous voltages may exist even with the equipment switched off. To avoid dangerous electrical shock, DO NOT perform procedures involving cover or shield removal unless you are qualified to do so.

DO NOT operate damaged equipment: Whenever it is possible that the safety protection features built into this product have been impaired, either through physical damage, excessive moisture, or any other reason, REMOVE POWER and do not use the product until safe operation can be verified by service-trained personnel. If necessary, return the product to an Agilent Technologies Sales and Service Office for service and repair to ensure that safety features are maintained.

DO NOT service or adjust alone: Do not attempt internal service or adjustment unless another person, capable of rendering first aid and resuscitation, is present.

DO NOT substitute parts or modify equipment: Because of the danger of introducing additional hazards, do not install substitute parts or perform any unauthorized modification to the product. Return the product to an Agilent Technologies Sales and Service Office for service and repair to ensure that safety features are maintained.

Declaration of Conformity

Declarations of Conformity for this product and for other Agilent products may be downloaded from the Internet. There are two methods to obtain the Declaration of Conformity:

- Go to **http://regulations.corporate.agilent.com/DoC/search.htm**. You can then search by product number to find the latest Declaration of Conformity.
- Alternately, you can go to the product web page **(www.agilent.com/find/E1458A),** click on the Document Library tab then scroll down until you find the Declaration of Conformity link.

Notes

Notes

Notes

Chapter 1
Getting Started

Using This Chapter

This chapter describes the Agilent E1458A 96-Channel Digital I/O Module and how to program the module using SCPI (Standard Commands for Programmable Instruments) commands. This chapter contains the following sections:

Digital I/O Module Description

The Agilent E1458A 96-Channel Digital I/O Module (referred to as the "digital I/O module") is a 12-port digital input/output module intended for data communication and digital control in electronic environments. The digital I/O module is compatible with TTL levels (0-5V) and CMOS levels (using an external pull-up). The digital I/O module complies with VXIbus (VMEbus Extensions for Instrumentation) definitions for the P1 and P2 bus connectors on C-sized modules. A jumper on the module sets the VXIbus interrupt level.

Each port is identical and consists of seven control lines and eight data lines. There are eight registers for control and status on each port. In addition, the module has Manufacturer ID, Device ID, Module Status/Control and Interrupt Status registers. Figure 1-1 shows the locations of the ports and a simplified diagram of a single port. Of the seven control lines, three (I/$\overline{O}$, CTL, and FLG) are used with SCPI commands, three ($\overline{RES}$, STS, and PIR) are controlled through register access and one, UTS, can be used to force all data lines into a three-state mode. Chapter 3, "Using the Agilent E1458A Digital I/O Module," contains detailed descriptions of these lines.

Each port has two hardware configuration switches. One switch allows you to connect the flag lines together for multiport data transmission. Another switch selects either active pull-up or passive pull-up to TTL compatible levels on the data lines. Chapter 2, "Configuring the Agilent E1458A Digital I/O Module," describes how to set these switches.

SCPI commands provided for the digital I/O module allow operation in 8-bit “BYTE” format, 16-bit “WORD” format (using two ports), 32-bit “LWORd” format (using four ports), 64-bit “LW64” format (using eight ports), and 96-bit “LW96” format (using all ports).

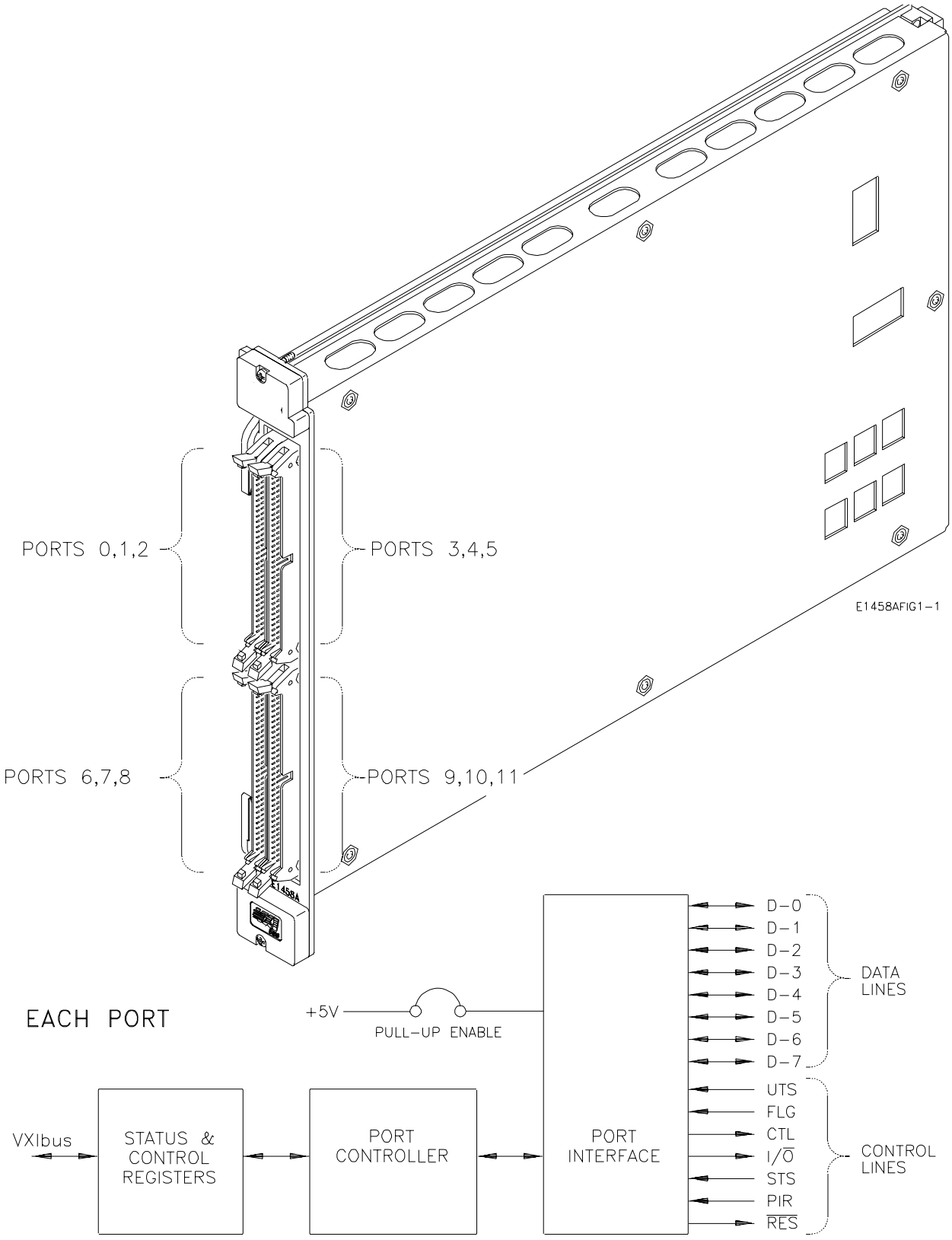


Figure 1-1. Agilent E1458A Digital I/O Module

Table 1-1. Data Lines

8-bit (BYTE) Operations												
Port	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
Bit Numbers	7 — 0	7 — 0	7 — 0	7 — 0	7 — 0	7 — 0	7 — 0	7 — 0	7 — 0	7 — 0	7 — 0	7 — 0
16-bit (WORD) Operations												
Port	**0**		**2**		**4**		**6**		**8**		**10**	
Bit Numbers	15 — 8	7 — 0	15 — 8	7 — 0	15 — 8	7 — 0	15 — 8	7 — 0	15 — 8	7 — 0	15 — 8	7 — 0
32-bit (LWORd) Operations												
Port	**0**				**4**				**8**			
Bit Numbers	31—24	23 —16	15— 8	7—0	31—24	23—16	15—8	7—0	31—24	23—16	15—8	7— 0
64-bit (LW64) Operations												
Port	**0**								**8**	**9**	**10**	**11**
Bit Numbers	63—56	55—48	47—40	39—32	31—24	23—16	15—8	7—0	7 — 0	7 — 0	7 — 0	7 — 0
96-bit (LW96) Operations												
Port	**0**											
Bit Numbers	95—88	87—80	79—72	71—64	63—56	55—48	47—40	39—32	31—24	23—16	15—8	7—0

Table 1-1 shows the mapping of bit numbers from the 8-bit ports to the 16-, 32-, 64-, and 96-bit ports. Chapter 4, "Agilent E1458A Command Reference," describes each command in detail and Chapter 3, "Using the Agilent E1458A Digital I/O Module," gives some examples of the use of SCPI commands.

Connections are made through 64-pin header connectors. Each connector contains connections for three ports as shown in Figure 1-1. Four cables with the 64-pin header installed are included with the Agilent E1458A module.

Additional cable sets and special Opto22® cables can be ordered from Agilent Technologies. (Contact your Agilent Technologies Sales office.) The cable part numbers are shown below. Additional details about these cables are given in Chapter 2, "Configuring the Agilent E1458A Digital I/O Module."

Agilent Part Number	**Description**
E1458-61601	Four 2-meter cables, 64-pin headers to unterminated flat ribbon cable
E1458-61604 (Agilent E1458A Option 022)	Two 1.5-meter cables, two 64-pin headers to three 50-pin headers for Opto22® connections.

Instrument Definition

Each Agilent E1458A Digital I/O Module installed in an Agilent mainframe is treated as an independent instrument, having a unique secondary GPIB address. Each instrument is also assigned a dedicated error queue, input and output buffers, status registers and, if applicable, dedicated mainframe memory space for readings or data. Multiple digital I/O modules cannot be combined into a single instrument.

Downloading the SCPI Drivers

The digital I/O driver allows the Agilent E1458A module to operate with an Agilent E1405B or E1406 Command Module using the Standard Commands for Programmable Instruments (SCPI) command language. The driver name for the digital I/O module is “DIG_IO”.

The procedure for downloading the drivers is contained in the *Downloading SCPI Device Drivers* Installation Note supplied with the module. The *C-Size VXIbus Systems Configuration Guide* also contain the installation procedure.

Programming the Digital I/O Module

To program the digital I/O module using SCPI, you must select the controller language, interface address, and SCPI commands. Guidelines for SCPI command selection for the digital I/O module are covered in this manual. See the *C-Size VXIbus Systems Configuration Guide* for detailed interface addressing and controller language information.

Note This discussion applies only to SCPI (Standard Commands for Programmable Instruments) programming. See "Register Descriptions" in Appendix B for details on register addressing. Do not mix SCPI programming and direct register access.

SCPI Command Format Used in this Manual

SCPI commands can be used in either long or short form. A long form example is:

```
DISPlay:MONitor <ON>
```

The same command, without the lowercase letters, is the short form. For example:

```
DISP:MON <ON>
```

Either the long form or the short form commands can be used to perform the same result. The long and short forms can also be mixed within the same program code. The commands are case insensitive; either upper- or lowercase letters are accepted.

In the command examples shown above, the item enclosed in angle brackets (**< >**) is a parameter required to use the command. In this example, the parameter input can be replaced with any one of the following: 0, 1, OFF, or ON. The allowable values of the parameters are given in Chapter 4, "Agilent E1458A Command Reference." You must be sure to include a space between the SCPI keyword and any parameters.

Some commands are shown with items enclosed in square brackets (**[ ]**). These are implied or optional items that do not have to be included. For example, the complete command syntax listing for the first example is:

```
DISPlay:MONitor[:STATe]  <0|1 or OFF|ON>
```

The item enclosed in brackets, [:STATe], does not have to be included for the command to work.

Complete descriptions of the SCPI command language, syntax, parameter types, and usage are in Chapter 4 of this manual.

Specifying SCPI Commands

SCPI commands related to the Agilent E1458A Digital I/O Module use three types of parameters to specify a port number, a bit number, or a multiple- port combining operation. Each type is briefly described here; descriptions and examples of usage can be found in Chapter 3 of this manual.

Specifying the Port

The digital I/O module has twelve identical ports numbered from 0 to 11. SCPI commands that relate to a specific port use a special parameter to indicate the port number. For example:

```
[SOURce:]DIGital:DATAn  <value>
```

This command writes <*value*> to the port specified by the *n* portion of the DATA parameter. Replace the *n* with the port number, making the number the last characters of the DATA keyword without spaces. For example, to set all port 2 data lines to logical zero, use the following command:

```
[SOURce:]DIGital:DATA2  0
```

The value of *n* may vary for multiple-port commands and operations. A description of multiple-port commands follows later in this chapter.

Specifying a Bit

Each of the twelve ports on the digital I/O module has eight bi-directional data bits. Some SCPI commands allow you to manipulate or read these bits individually. For example:

```
MEASure:DIGital:DATAn:BITm?
```

This command reads the state of a bit, specified by *m*, on port *n*. The result of the command will be either `0` or `1`, indicating the current state of the bit. Replace *m* with the desired bit number, and *n* with the desired port number, making each number the last characters of the DATA and BIT keywords without spaces. For example, to read the state of bit 7 on port 0, use the following command:

```
MEASure:DIGital:DATA0:BIT7?
```

For single ports, the value of *m* can range from 0 to 7. Some multiple port operations and commands may allow bit numbers to range from 0 to 95.

Specifying Multiple-Port Operations

The digital I/O module allows you to set or read multiple ports or bits with a single command. For example:

```
MEASure:DIGital:DATAn[:type]?
```

This command uses an optional keyword [:type] to specify how many ports are combined in a single returned value. The lowercase keyword [:type] is replaced with one of a fixed set of keywords. For example, to read all 12 ports (all 96 bits) as three returned values, use the command:

```
MEASure:DIGital:DATA0:LW96?
```

Keywords are provided to allow port combinations of 16, 32, 64, or 96 bits. Using multiple ports is described in more detail in Chapter 3 of this manual.

Initial Operation

Use the following example to verify initial operation. The example sets and then queries the polarity of a logical true condition on the port 0 FLG line.

The example uses an HP Series 200/300 Computer with BASIC as the programming language. The computer is connected to an Agilent 1406A Command Module in the C-Size VXIbus cardcage using the General Purpose Interface Bus (GPIB). The GPIB interface select code is 7, the GPIB primary address is 09, and the GPIB secondary address (used to specify the digital I/O module) is 18. Refer to the *C-Size VXIbus Systems Configuration Guide* for more details about programming languages and interfaces.

```
10  ASSIGN @Dio TO 70918                ! Sets an I/O path to the module.
20  DIM Polarity$[3]                    !
30  OUTPUT @Dio;"*RST"                  ! Reset the module.
40  OUTPUT @Dio;"*OPC?"                 ! Wait for the module to finish.
50  ENTER @Dio;Ready                    ! Hold here until command is
                                        finished.
60  OUTPUT @Dio;"SOUR:DIG               ! Set POSitive polarity.
    :FLAG0:POL POS;*OPC?"
70  ENTER @Dio;Ready                    ! Wait for finish.
```

Continued on next page.

```
80  OUTPUT @Dio;"SOUR:DIG                ! Query the polarity state.
    :FLAG0:POL?"
90  ENTER @Dio;Polarity$                 ! Get the result.
100 IF Polarity$ <> "POS" THEN           ! Check the result.
110     DISP "Polarity Check ERROR"      ! Error discovered.
120     PAUSE                            ! Pause on error.
130 ELSE
140     DISP"Polarity set to "&Polarity$
150 END IF
160 OUTPUT @Dio;"SOUR:DIG                ! Set NEGative polarity.
    :FLAG0:POL NEG;*OPC?"
170 ENTER @Dio;Ready                     ! Wait for finish.
180 OUTPUT @Dio;"SOUR:DIG                ! Query the polarity state.
    :FLAG0:POL?"
190 ENTER @Dio;Polarity$                 ! Get the result.
200 IF Polarity$ <> "NEG" THEN           ! Check the result.
210     DISP "Polarity Check ERROR"      ! Error discovered.
220     PAUSE                            ! Pause on error.
230 ELSE
240     DISP"Polarity set to "&Polarity$
250 END IF
260 OUTPUT @Dio;"*RST"                   ! Restore the card.
270 OUTPUT @Dio;"*OPC?"                  ! Wait for the module to finish.
280 ENTER @Dio;Ready                     !
290 END
```

Chapter 2
Configuring the Agilent E1458A Digital I/O Module

Using This Chapter

This chapter shows how to configure the Agilent E1458A Digital I/O Module for use in a VXIbus mainframe, install it in a mainframe, connect peripheral devices, and configure the module for operation. This chapter contains the following sections:

WARNINGS and CAUTIONS

WARNING **SHOCK HAZARD. Only service-trained personnel who are aware of the hazards involved should install, remove, or configure the digital I/O module. Before you remove any installed module, disconnect AC power from the mainframe. Disconnect power connected to the inputs/outputs of this module and other modules in the mainframe.**

CAUTION **MAXIMUM INPUTS. The maximum voltage that can be applied to any pin on any port is 5 Vdc. The maximum current allowed through any data line is 3.2 mA and through any control line is 1.75 mA.**

STATIC ELECTRICITY Static electricity is a major cause of component failure. To prevent damage to the electrical components in the module, observe anti-static techniques whenever removing a module from the mainframe or whenever working on a module.

Setting the Address Switch

Figure 2-1 shows the location and factory default setting of the logical address switch (LADDR). The factory setting is decimal 144; switch positions 4 and 7 are closed, all others are open. The factory setting of decimal 144 results in a GPIB secondary address of 18 (144 divided by 8). You can set the address of the digital I/O module to any number in the range of 0 to 255 (decimal).

Note For the digital I/O module to be recognized as an instrument when you are using it with an Agilent E1405/E1406 Command Module and using SCPI commands, the logical address should be set to an integer multiple of 8.

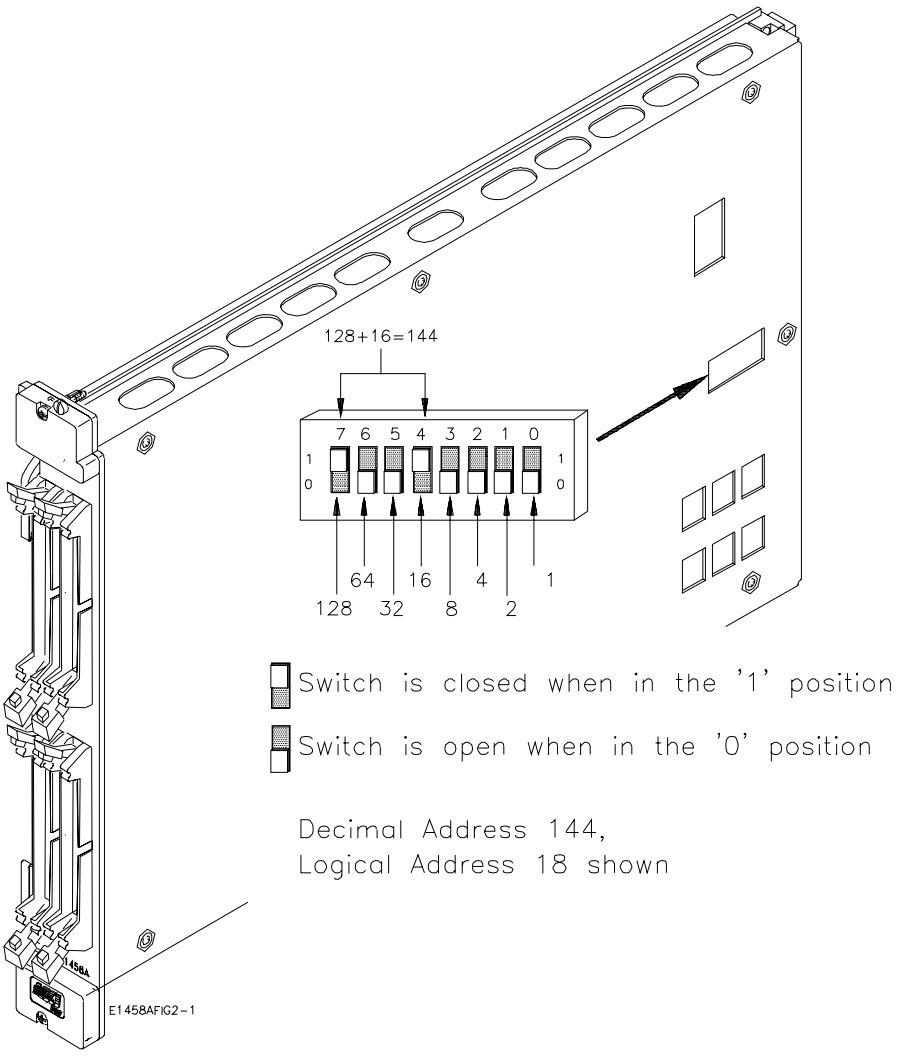


Figure 2-1. Logical Address Switch

Setting the Interrupt Priority

The VXIbus peripheral interrupt uses seven lines to carry the interrupt signal to the commander. The most common line to be used is line one, as this is the usual default interrupt line. Many VXIbus commanders have a way to change the interrupt line they manage (for example, the Agilent E1405/E1406 has an interrupt line allocation table). When doing direct register-based programming (instead of using the register-based SCPI driver), set the interrupt line to a line that is not used by the register-based SCPI driver. Module interrupt priority can be established with these lines; in general, the higher the line number, the higher the priority. Figure 2-2 shows the location and factory default setting of the interrupt priority jumper. Setting Level X disables the interrupt capability.

Note The interrupt priority MUST be set to a line handled by the command module when using and Agilent E1405 or Agilent E1406 Command Module. Level X priority should not be used under normal operating conditions. Changing the interrupt priority level is not recommended. Do not change the default setting (level 1) unless specifically instructed to do so.

The interrupt circuitry for the Agilent E1458A Digital I/O Module is implemented as release on interrupt acknowledge (ROAK). The digital I/O module will de-assert (or release) the interrupt request line during an interrupt acknowledge cycle.

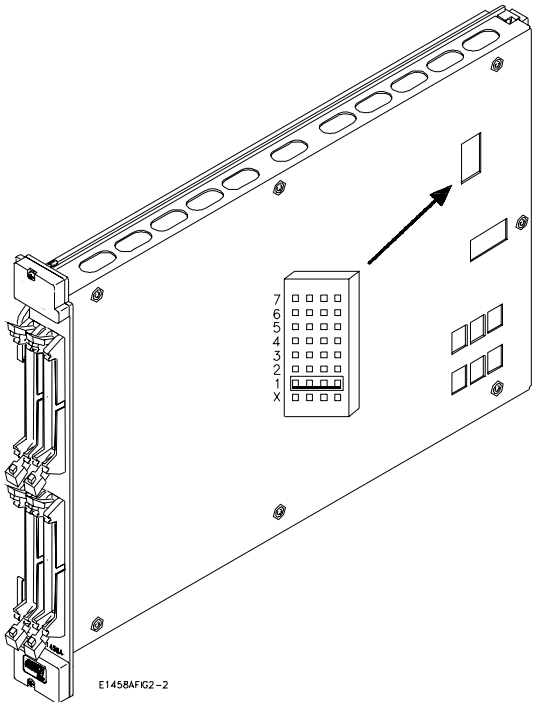


Figure 2-2. Interrupt Priority Jumper

Enabling the Data Line Pull-up Network

The characteristics of the data lines on each port are determined by the SN75ALS160 bus transceiver chip (see Figure 3-3). This chip was designed to drive the IEEE 488 bus, which is a proven high reliability communication bus. The data lines of each port contain a terminating network that is always present. This terminating network has the characteristics of being high impedance when the power to the module is off. This ensures that when power is removed from the module, the data lines do not load what is connected to them. With power applied, the terminating network is equivalent to the 3k/6.2k resistive network that is shown in Figure 2-3.

The resistive terminating network provides a pull-up to a logic high level. In addition to the resistive termination, an active pull-up to TTL levels for outputs may be enabled by the pull-up enable switches.

Figure 2-3 shows the location of the pull-up enable switches and a detail of a single pull-up enable switch setting. The factory-shipped default is pull-up disabled for all ports.

The data lines may be either outputs or inputs. When the data lines are outputs and the switch is in the enabled postition, the outputs will be forced high by both the resistive termination and the active pull-up. When the data lines are outputs and the switch is in the disabled position, the outputs will be forced high by only the 3k/6.2k resistive termination.

When the data lines are inputs, the resistive termination network is the dominating load that must be driven. This resistive termination network, if not driven, will go to a logic high. This characteristic may be used to sense a contact closure to ground. When the data lines are inputs, the position of the pull-up enable switches makes no difference, as the pull-up enable is only for outputs.

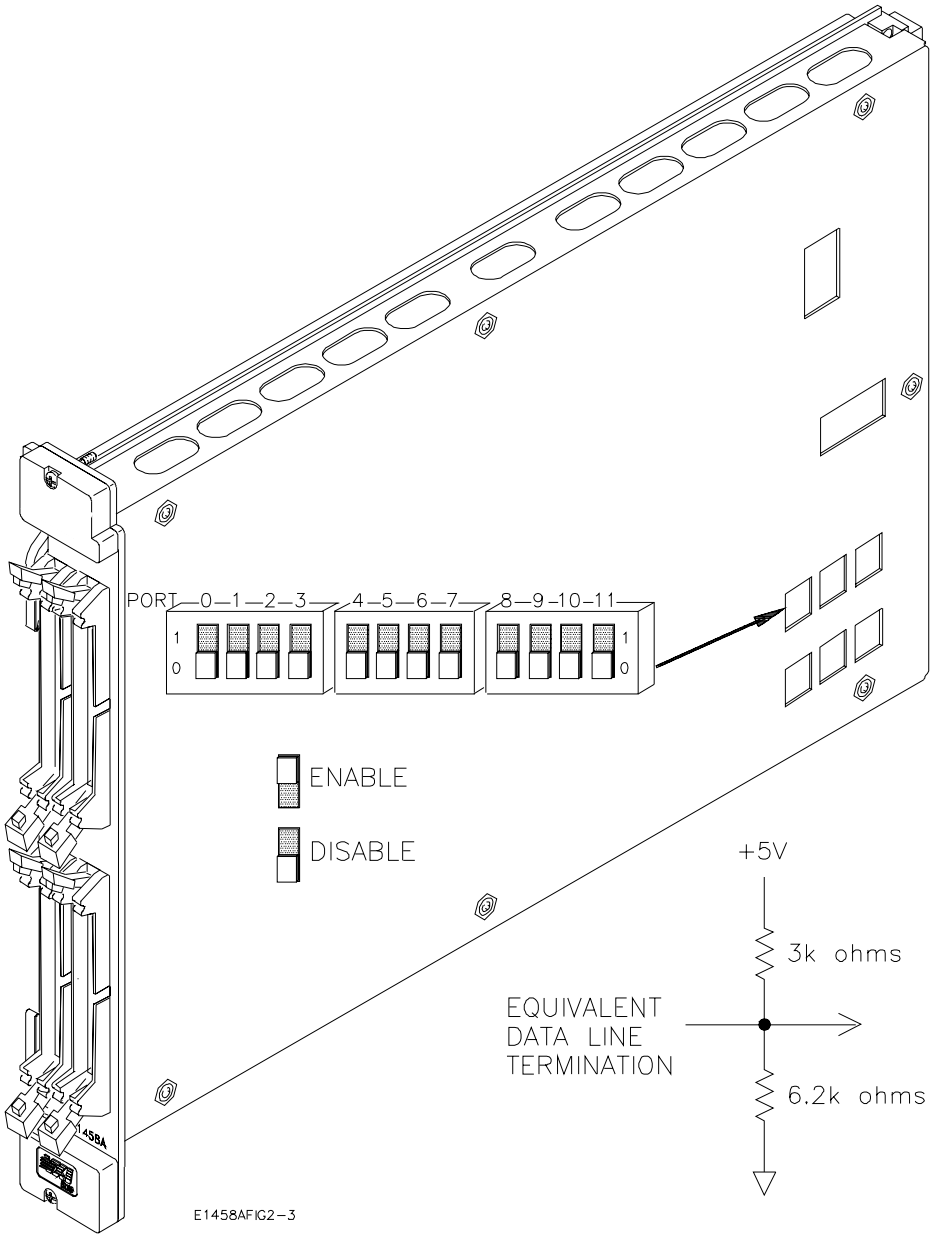


Figure 2-3. Pull-up Enable Switches

Combining the Flag Lines

Each port contains a Flag Line, labeled FLG, that can be used to implement a handshake scheme with a peripheral. For single-port operations, the FLG lines can be used in the factory default setting (no flag lines combined) to handshake with a peripheral. For multi-port operations with a single handshake line, you can combine the flag line from multiple ports. The combined flag lines are physically tied together. An action on any of the combined flag lines performs that action for all combined flag lines.

Figure 2-4 shows the locations of the flag combining switches and how to set them. Before setting any flag combine switches, you may wish to read the discussion regarding allowable port combinations and handshaking in Chapter 3 of this manual.

Note When using the FLG and CTL for handshaking on multiple-port operations, the CTL line is set for each port sequentially beginning at the lowest-numbered port.

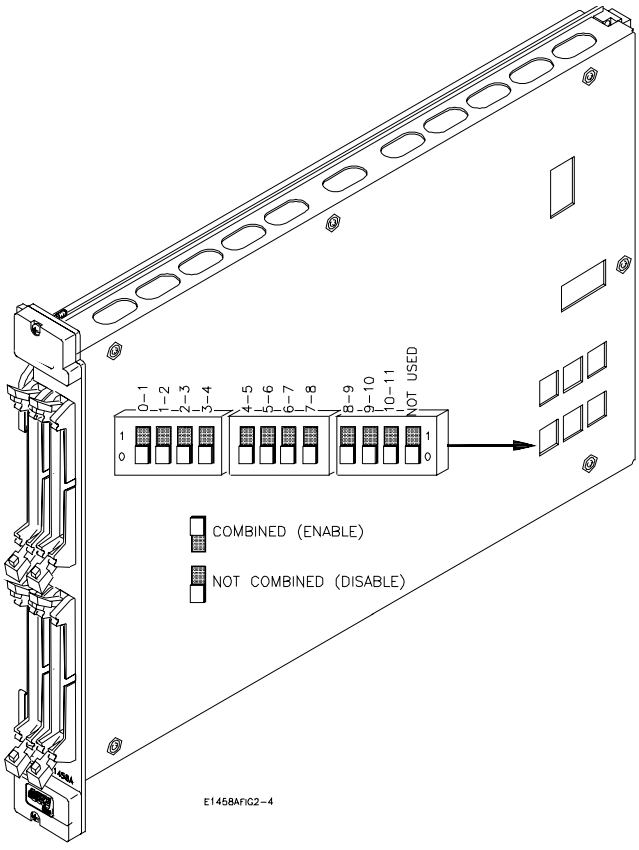
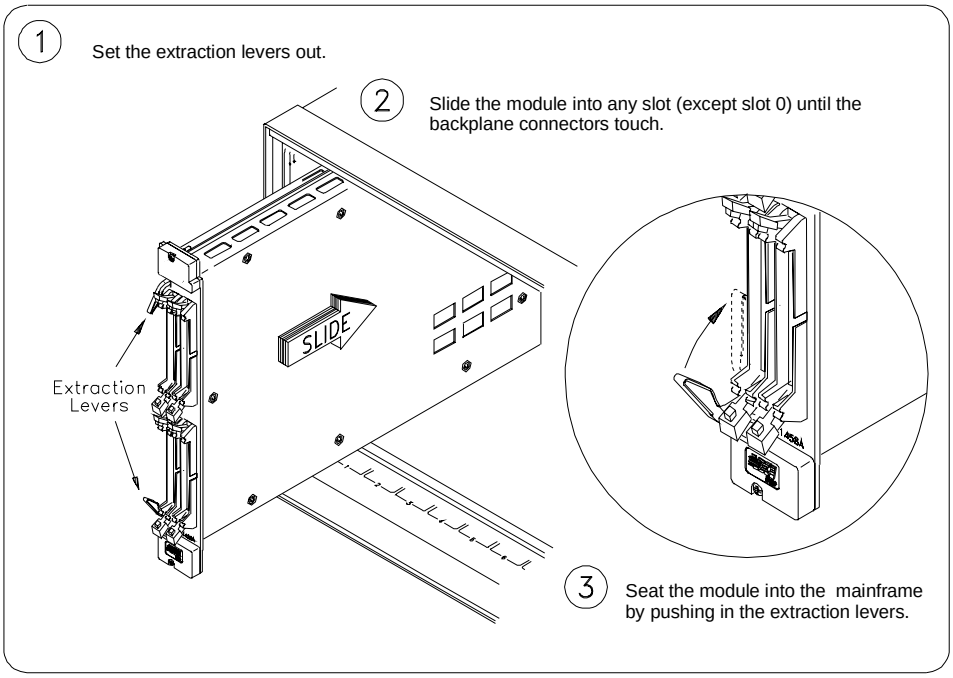
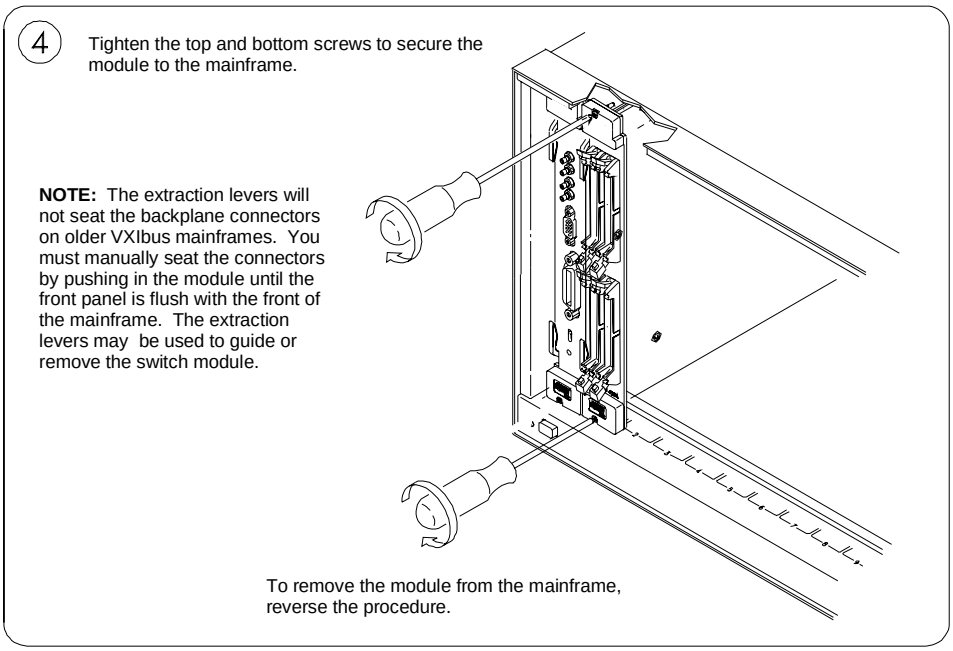


Figure 2-4. Flag Combining Switches

Installing the Agilent E1458A Digital I/O Module in a Mainframe

The Agilent E1458A Digital I/O Module may be installed in any slot (except slot 0) in a C-size mainframe. Refer to Figure 2-5 to install the module in a mainframe.





2-5. Installing the Module in a Mainframe

Connecting to Peripheral Devices

Each of the four Agilent E1458A Digital I/O Module peripheral connectors has 64 pins. The connectors are compatible with industry standard 64-pin .100" X .100" wiremount sockets. A compatible socket can be obtained by ordering Agilent part number 1252-1581 (3M Part Number 7964-6500EC) for compatibility with 0.050" ribbon cable. For discrete wire compatibility use 3M part number CHG-2064-J01010-KCP.

Figure 2-5 shows the peripheral connectors, their pin numbers, and the corresponding line names. Figure 2-6 shows the digital I/O ribbon cables (Agilent Part Number E1458-61601) and the line locations within the ribbon connector.

Note Not all control lines are required for every application. The use of each control line is described in Chapter 3.

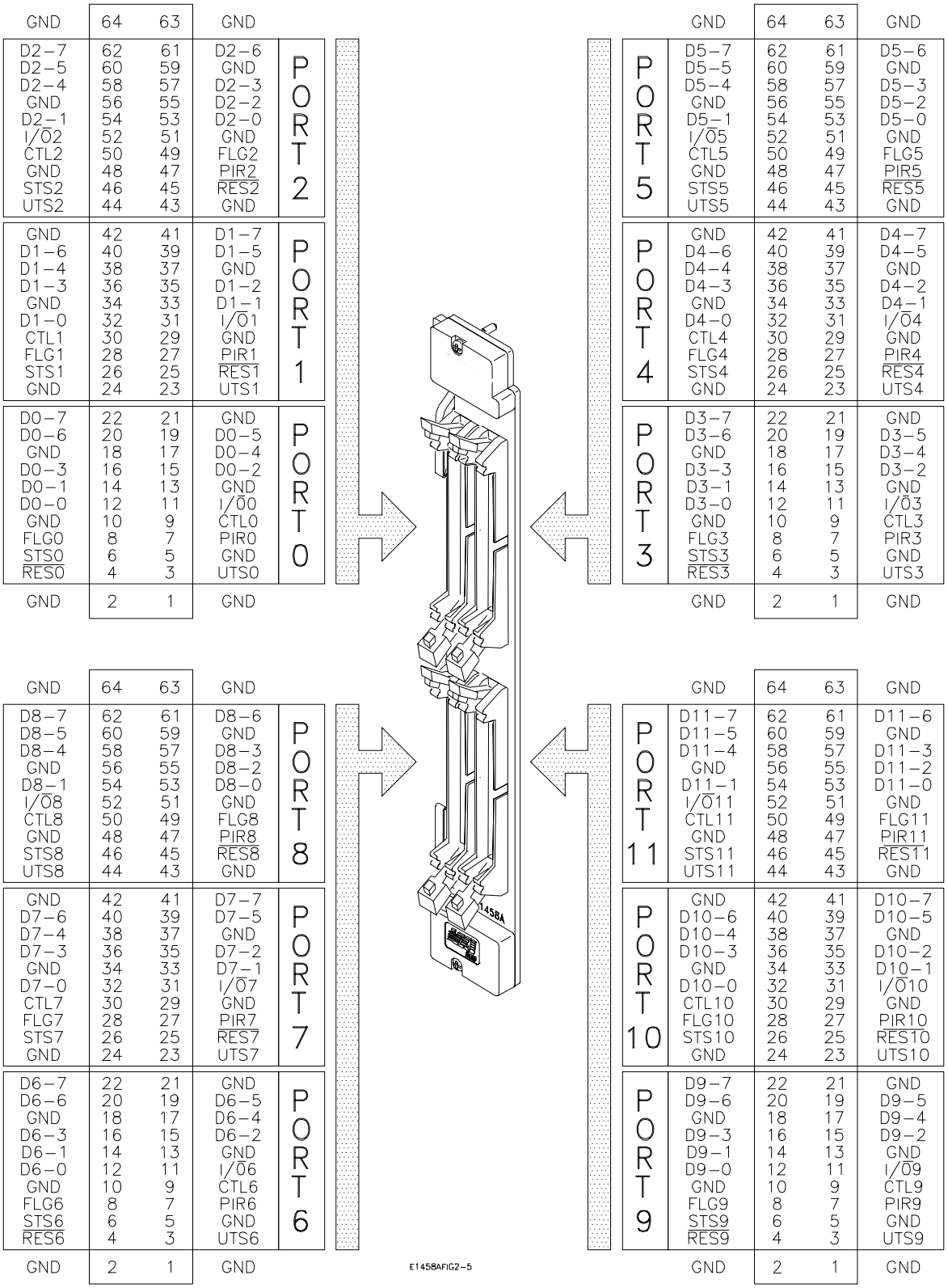
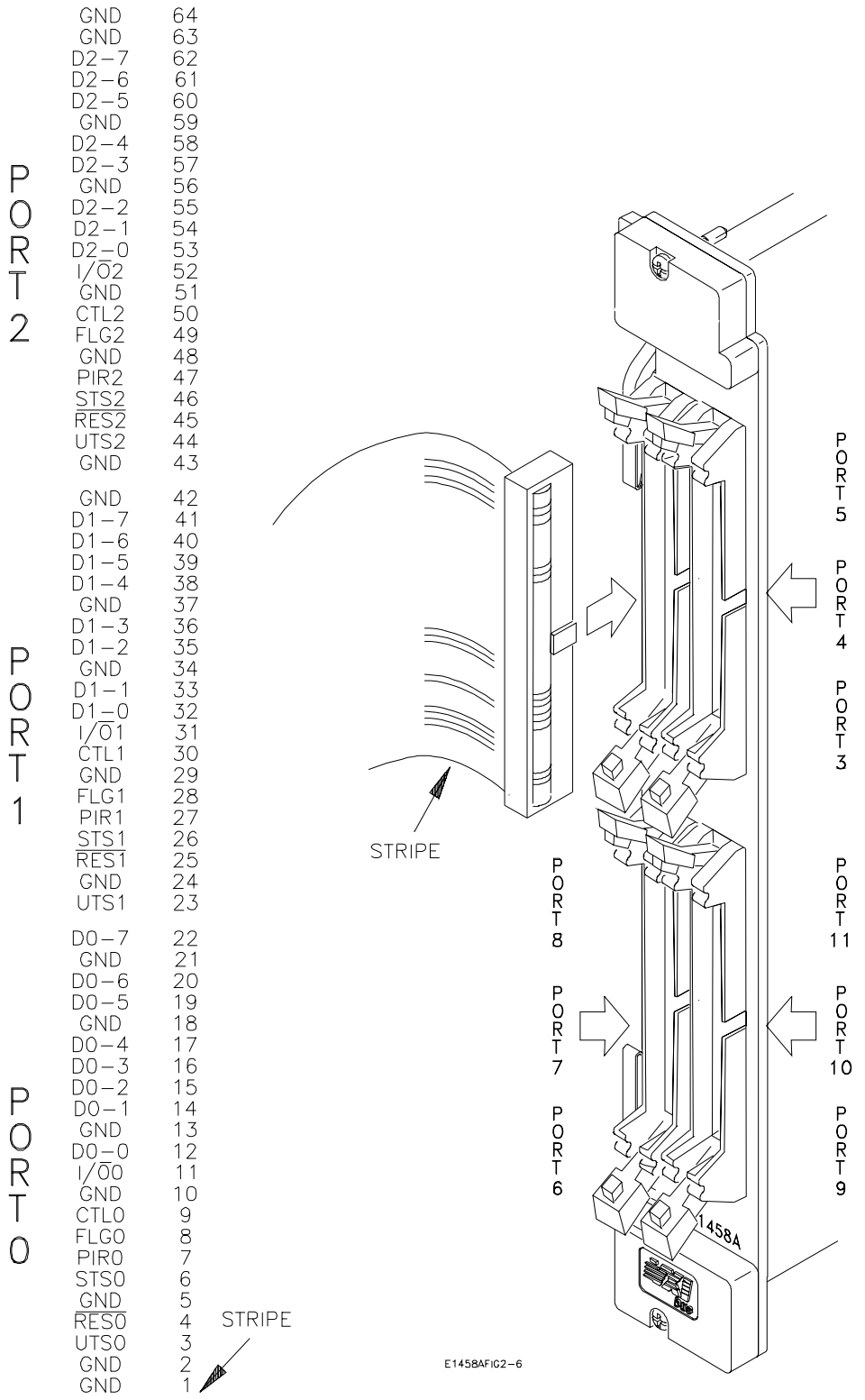


Figure 2-6. Connector Pin Assignments



2-7. Connections Within the Ribbon Cable

Configuring for Isolated Digital I/O

An industry-standard isolated digital I/O peripheral, like the **Opto 22**® 16-Position Single-Channel Mounting Rack, is a 50-pin connection. The connector is either a card edge or a header connector. Do not connect pins 1 and 49 on the Opto 22® rack connector.

For example, the Opto 22® rack, PB16C, uses a card edge connector; PB16H uses a header connector. They both have the same pinout for a ribbon cable. Both can accommodate up to 16 single-channel I/O lines.

A cable designed to connect up to six ports to three Opto 22® racks is available from Agilent Technologies, either as Option 022 when ordering the Agilent E1458A, or as Agilent part number E1458-61604. Figure 2-7 shows the connections from six ports to three racks.

Opto 22® is a registered trademark of Opto 22, Huntington Beach, CA 92649

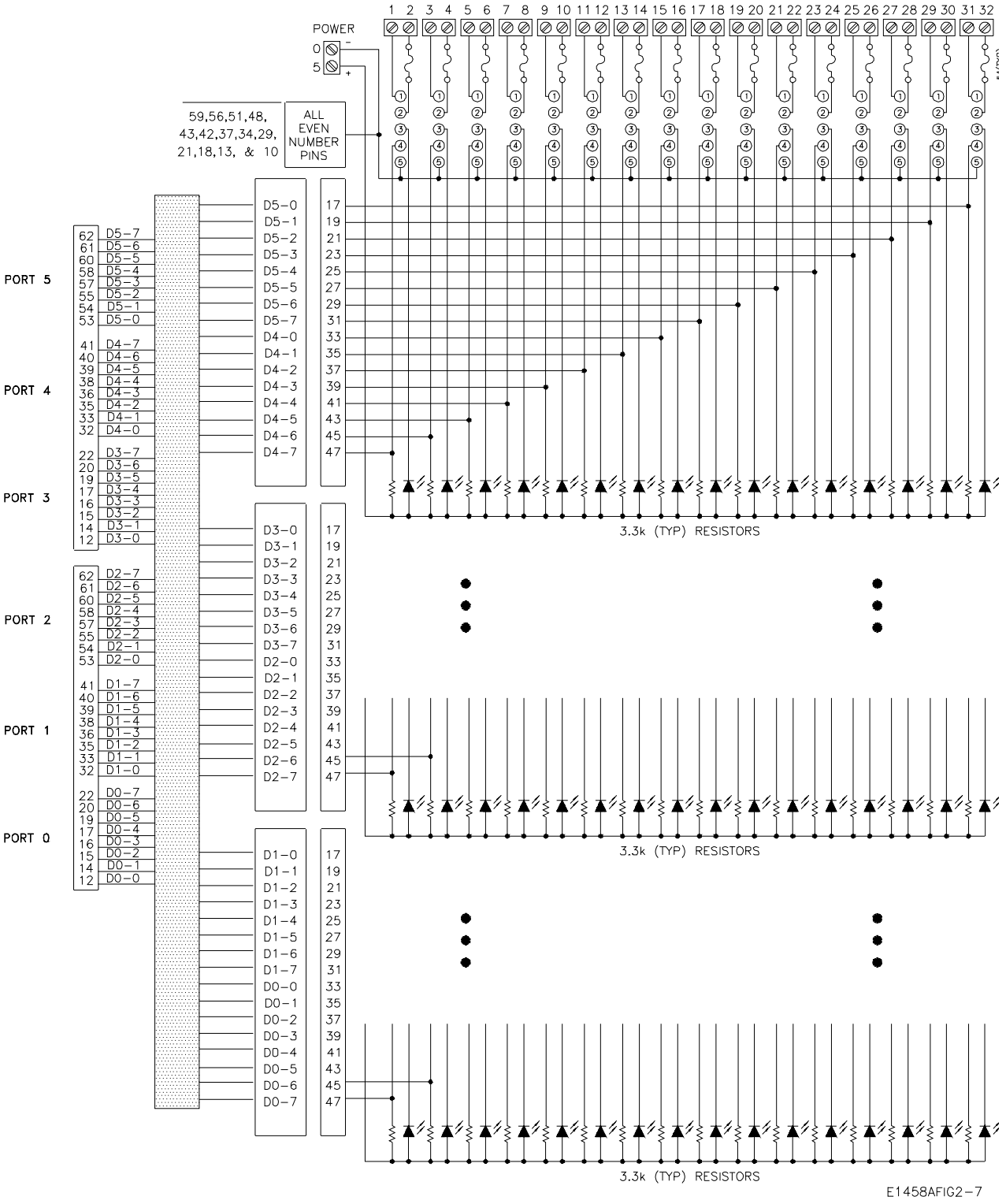


Figure 2-8. Opto 22 Connections

Chapter 3
Using the Agilent E1458A Digital I/O Module

Using This Chapter

This chapter is divided into twelve sections and details how to use the Agilent E1458A Digital I/O Module.

Port Description

Each of the digital I/O module ports has eight data lines and seven control lines. Not all these lines are required for every application. Figure 1-1 earlier in this manual shows the data and control lines. The following subsections describe the use of these lines.

Data Lines

Each port has eight data lines, numbered from 0 to 7. The data lines can be set as an 8-bit group, as part of a larger group, or individually using SCPI commands.

The logical TRUE condition of the data lines can be controlled with SCPI commands. Positive polarity is the default. The following table shows the effect of changing the Polarity with Input and Output operations for each data line.

	Input Operations	Output Operations
POSitive Polarity	TTL High = 1	1 = TTL High
	TTL Low = 0	0 = TTL Low
NEGative Polarity	TTL High = 0	0 = TTL High
	TTL Low = 1	1 = TTL Low

The data lines in each port, as a group, can be set to use an internal +5 Vdc pull-up during output operations. Additionally, the UTS control line can force all the data lines in a port to the three-state condition (non-sourcing).

The FLG Line (Input)

Each port has a flag (**FLG**) line. A flag line is an input line from a peripheral and has two states, READY and BUSY. A flag line is normally used in conjunction with the corresponding control line (CTL) to establish a handshake between a peripheral and the digital I/O module. SCPI commands that define handshake modes typically use the FLG and CTL lines. The state of the FLG line can also be read with a SCPI command to implement custom handshakes. Positive polarity is the default. The following shows the effect of changing the polarity of the FLG line:

POSitive Polarity	**TTL High = BUSY = 1** **TTL Low = READY = 0**
NEGative Polarity	**TTL High = READY = 0** **TTL Low = BUSY = 1**

The CTL Line (Output)

Each port has a control line (**CTL**). A control line is an output line from the digital I/O module to the peripheral and has two states, TRUE and FALSE. A control line is normally used in conjunction with the corresponding flag line on the same port to establish a handshake between a peripheral and the digital I/O module. SCPI commands that define handshake modes typically use the FLG and CTL lines. The state of the CTL line can be read and set with SCPI commands to implement custom handshakes. Positive polarity is the default. The following shows the effect of changing the polarity of the CTL line:

POSitive Polarity	**TTL High = TRUE = ON = 1** **TTL Low = FALSE = OFF = 0**
NEGative Polarity	**TTL High = FALSE = OFF = 0** **TTL Low = TRUE = ON = 1**

The I/$\overline{O}$ Line (Output)

Each port has an **I/$\overline{O}$** line. An I/$\overline{O}$ line is an output from the Agilent E1458A Digital I/O Module to the peripheral and has two states, TRUE or FALSE. The state of the I/$\overline{O}$ line is not directly programmable. The I/$\overline{O}$ line polarity cannot be changed.

When the I/$\overline{O}$ line is: TTL High =TRUE = 1 = Input

the data transceiver of that port is enabled for input. The peripheral should respond to the signal by enabling itself to send data.

When the I/$\overline{O}$ line is: TTL Low = FALSE = 0 = Output

the data transceiver of that port is enabled for output. The peripheral should respond to the signal by enabling itself to receive data.

CAUTION **To prevent damage to the digital I/O module, when the I/$\overline{O}$ line is set for Output (TTL Low), the peripheral MUST NOT attempt to source on any data lines.**

Note The I/$\overline{O}$, CTL and $\overline{RES}$ lines are true open collector output lines (see Figure 3-3). An external pull-up is required for proper operation. The FLG, PIR and STS input lines have a 3k/6.2k pull-up termination. If the PIR and STS lines are not used, then connecting them to the I/$\overline{O}$ and CTL line will provide the pull-up that the I/$\overline{O}$ and CTL lines need.

The STS Line

Each port has a status line labeled **STS**. The STS line is an input line to the digital I/O module. The use of the STS line is only at the register level and is not supported by SCPI commands. Refer to Appendix B, "Agilent E1458A Register Information," for more information about this line.

The PIR Line

Each port has a peripheral interrupt request line labeled **PIR**. The PIR line is an input line to the digital I/O module. The use of the PIR line is only at the register level and is not supported by SCPI commands. Refer to Appendix B for more information about this line.

The $\overline{RES}$ Line

Each port has a reset line labeled **$\overline{RES}$**. The $\overline{RES}$ line is an output line to the peripheral. Control of the $\overline{RES}$ line is only at the register level and is not supported by SCPI commands. Refer to Appendix B for more information about this line.

The UTS Line

Each port has a user three-state line labeled **UTS**. The UTS line is an input line to the digital I/O module. A TTL Low level on the UTS line will force the data line transceiver into a three-state mode. The data lines will be held in the three-state mode independent of the programmed state of the port. A TTL High on the UTS line allows normal digital I/O module operation. The UTS line can also be left unconnected to allow normal digital I/O module operation. See the section "Using the UTS Control Line" later in this chapter for a description of its use.

Addressing the Module

The examples shown in this chapter use the default addresses for the interface, command module, and digital I/O module. The address uses both GPIB primary and secondary addresses. The default address is:

7	0 9	1 8
GPIB Primary Address		GPIB Secondary Address
Interface Select Code	Command Module GPIB Address	Digital I/O Module Address Instrument (LADDR÷8)

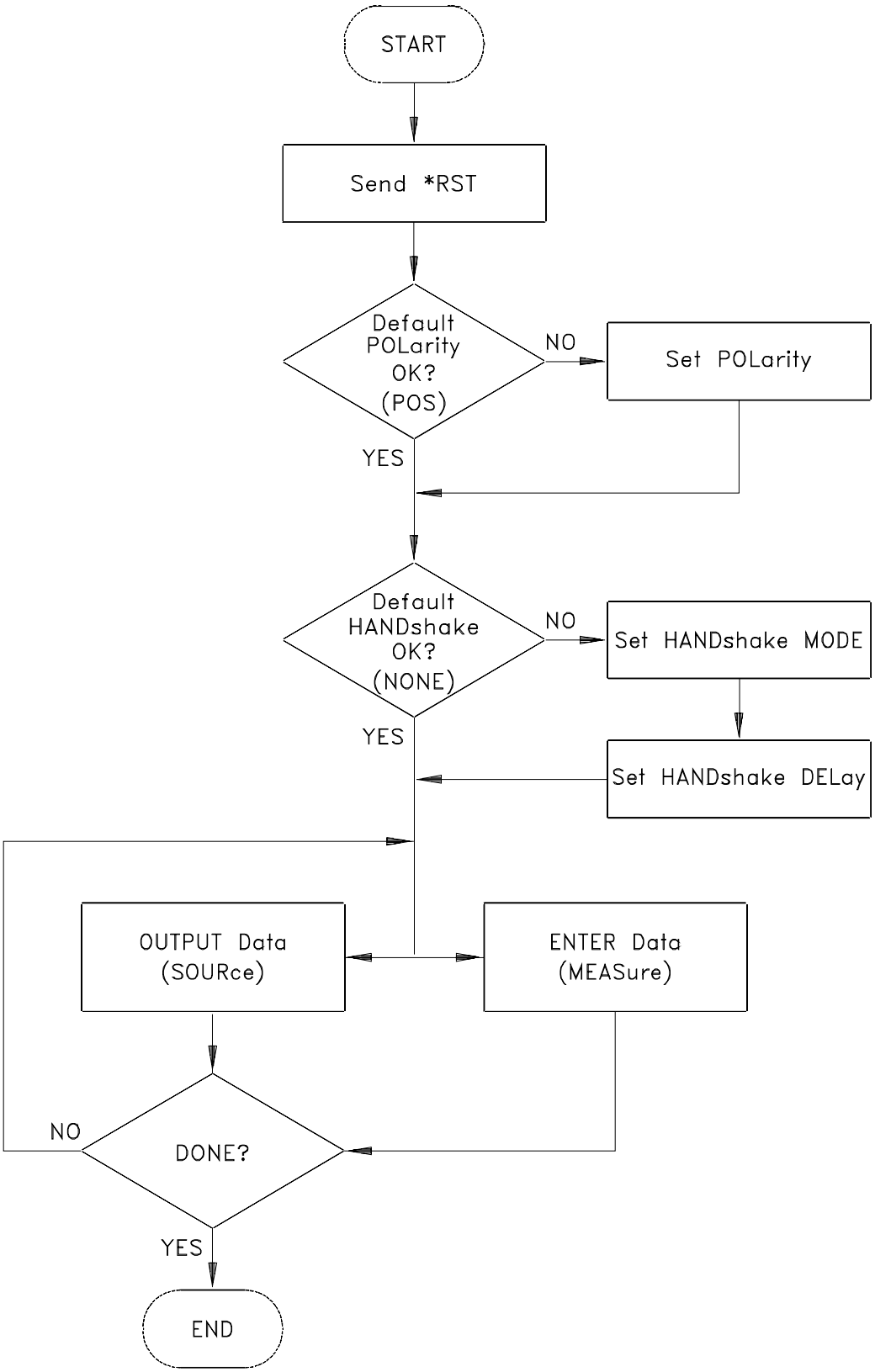
To establish these defaults as an I/O path in BASIC, the program examples use this code:

```
10  ASSIGN @Dio TO 70918
```

Each digital I/O module in a system must have a different logical address. Additionally, no two instruments in the same system can have the same logical address. Setting the logical address is described in Chapter 2, "Configuring the Agilent E1458A Digital I/O Module".

Operation Overview

The following steps illustrate general operation of the Agilent E1458A Digital I/O Module.



Default & Reset States

At initial power-on and following the *RST command, the Agilent E1458A Digital I/O Module is set to the following states:

CTL line	0 = TTL Low
I/$\overline{O}$ line	TRUE = input = TTL Low
Data, FLG, and CTL line Polarity	POSitive
Handshake mode	NONE

NOTE Following power-on or *RST, the digital I/O module is set to the input mode (I/O = TRUE). If no peripheral is driving the data lines, they will float to a TTL High state.

Setting the Polarity

The logical true level of the control (CTL) line, the flag (FLG) line, and the data lines of each port can be set to either TTL High (> 2.5V) or TTL Low (< 1.4V) levels. SCPI commands use the POLarity keyword as:

[SOURce:]DIGital:CONTrol*n*:POLarity <POSitive or NEGative> to set the control line's (CTL) polarity on port *n*.

[SOURce:]DIGital:FLAG*n*:POLarity <POSitive or NEGative> to set the flag line's (FLG) polarity on port *n*.

[SOURce:]DIGital:DATA*n*:POLarity <POSitive or NEGative> to set the data lines polarity on port *n*.

Example DIG:DATA1:POL POS

This command sets the polarity to positive on port 1 data lines, a TTL High will be input as a 1 or a bit set to 1 will output a TTL High level.

The *RST (reset) condition is positive polarity for control (CTL), flag (FLG), and data lines on all ports.

Using the Handshake Modes

Handshaking ensures correct transfer of data between devices. You must set both the mode and the timing to establish correct handshaking. SCPI commands support the following modes of handshaking:

- LEADing Edge
- TRAiling Edge
- PULSe
- PARTial
- STRobe
- NONE

These SCPI commands set the type of handshake mode used:

[SOURce:]DIGital:DATA*n*[:type]:HANDshake[:MODE] <*mode*>

[SOURce:]DIGital:HANDshake*n*[:MODE] <*mode*>

These SCPI commands set the timing of the handshake (where timing applies):

[SOURce:]DIGital:DATA*n*[:type]:HANDshake:DELay <*time*>

[SOURce:]DIGital:HANDshake*n*DELay <*time*>

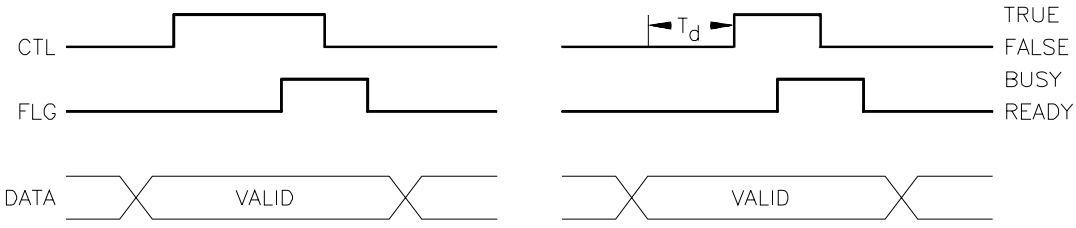
Handshake Modes

The operation of each handshake mode for input or output operations is described in the following subsections. In these discussions, only the FLG, CTL, and DATA lines are included. Other port control lines, controlled only through register access, are described in Appendix B of this manual.

LEADing Edge

The LEADing Edge handshake makes use of both the CTL and FLG lines. The input and output operations are described below.

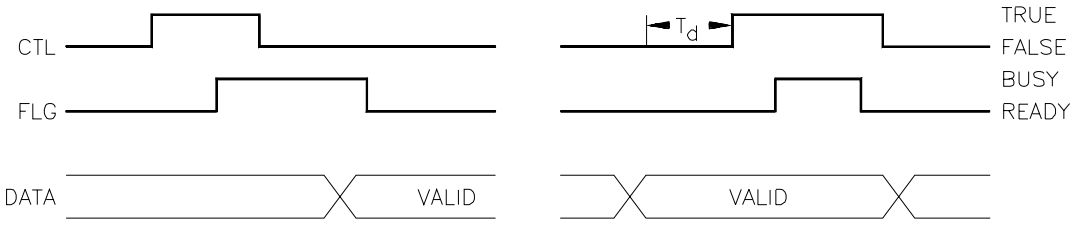
	INPUT		OUTPUT
1	The digital I/O module senses the FLG line and waits for READY.	1	The digital I/O module checks the state of the FLG line (must be READY).
2	The digital I/O module sets the I/O line HIGH.	2	The digital I/O module sets the I/O line LOW.
3	The digital I/O module sets CTL TRUE.	3	The digital I/O module places the data on the data lines.
4	The peripheral senses the CTL line and places data on the data lines.	4	After waiting the programmed delay time, T_d, the digital I/O module sets CTL to TRUE.
5	The peripheral sets the FLG line to BUSY indicating data is valid.	5	The peripheral senses the CTL line and sets the FLG line to BUSY while it latches the data.
6	The digital I/O module senses the FLG line and latches the data.	6	When the digital I/O module senses the FLG line in the BUSY state, it sets the CTL line to FALSE and monitors the FLG line.
7	The digital I/O module returns CTL to FALSE.	7	When the peripheral returns the FLG line to READY (indicating it has latched the data), the next handshake can begin.
8	The peripheral senses the CTL line and returns the FLG line to READY.		



TRAiling Edge

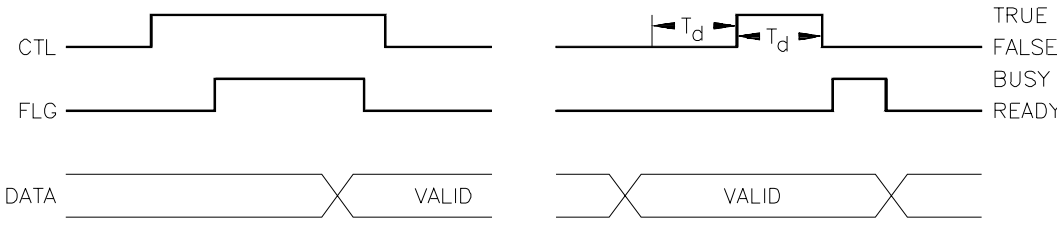
The TRAiling Edge handshake makes use of both the CTL and FLG lines. The input and output operations are described below.

	INPUT		OUTPUT
1	The digital I/O module senses the FLG line and waits for READY.	1	The digital I/O module checks the state of the FLG line (must be READY).
2	The digital I/O module sets the I/O line HIGH.	2	The digital I/O module sets the I/O line LOW.
3	The digital I/O module sets CTL TRUE.	3	The digital I/O module places the data on the data lines.
4	The peripheral senses the CTL line and sets the FLG line to BUSY.	4	After waiting the programmed delay time, T_d, the digital I/O module sets CTL to TRUE.
5	The digital I/O module senses the FLG BUSY and sets the CTL line FALSE.	5	The peripheral senses the CTL line and sets the FLG line to BUSY while it latches the data.
6	The peripheral senses the CTL line change and places data on the data lines.	6	The peripheral returns the FLG line to READY indicating the end of data transfer.
7	The peripheral indicates the data is valid by returning the FLG line to READY.	7	The digital I/O module senses the FLG line in the READY state and returns CTL to FALSE.
8	The digital I/O module senses the FLG READY and latches the data.		



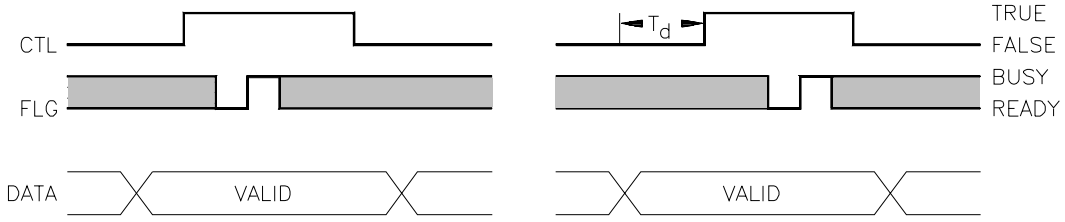
PULSe The PULSe handshake makes use of both the CTL and FLG lines. The input and output operations are described below.

	INPUT		OUTPUT
1	The digital I/O module senses the FLG line and waits for READY.	1	The digital I/O module checks the state of the FLG line (must be READY).
2	The digital I/O module sets the I/O line HIGH.	2	The digital I/O module sets the I/O line LOW.
3	The digital I/O module sets CTL TRUE.	3	The digital I/O module places the data on the data lines.
4	The peripheral senses the CTL line and sets the FLG line to BUSY.	4	After waiting the programmed delay time, T_d, the digital I/O module sets CTL to TRUE.
5	The peripheral places the data on the data lines and indicates valid data by setting the FLG line to READY.	5	The digital I/O module then waits another delay time, T_d, and sets the CTL line to FALSE.
6	The digital I/O module senses the FLG READY, returns CTL to FALSE, and latches the input data.	6	The peripheral senses the CTL line change, sets the FLG line to BUSY and latches the data.
		7	When the data is entered, the peripheral returns the FLG line to READY.



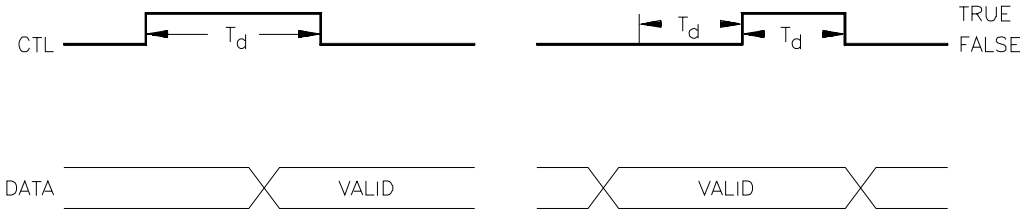
PARTial The PARTial handshake makes use of both the CTL and FLG lines. The input and output operations are described below.

	INPUT		OUTPUT
1	The digital I/O module sets the I/O line HIGH.	1	The digital I/O module sets the I/O line LOW.
2	The digital I/O module sets CTL TRUE.	2	The digital I/O module places the data on the data lines.
3	The peripheral senses the CTL line and sets the data lines.	3	After waiting the programmed delay time, T_d, the digital I/O module sets CTL to TRUE.
4	The peripheral holds the FLG line READY for at least 250 nsecs and then sets the FLG line BUSY to indicate the data is valid.	4	The peripheral senses the CTL line change, sets the FLG line to READY for a minimum of 250 nsecs, latches the data, and sets the FLG line to BUSY.
5	The digital I/O module senses the FLG line change to BUSY and latches the data.	5	The digital I/O module senses the change of the FLG line and sets CTL to FALSE.
6	The digital I/O module then sets the CTL line FALSE.		



STRobe The STRobe handshake makes use of the CTL line, but not the FLG line. The input and output operations are described below.

	INPUT		OUTPUT
1	The digital I/O module sets the I/O line HIGH.	1	The digital I/O module sets the I/O line LOW.
2	The digital I/O module sets CTL TRUE.	2	The digital I/O module places the data on the data lines.
3	The peripheral senses the CTL line and sets the data lines.	3	After waiting the programmed delay time, T_d, the digital I/O module sets CTL to TRUE.
4	The digital I/O module waits the programmed time delay , T_d, after setting CTL TRUE and then latches the data.	4	The peripheral senses the CTL line and latches the data.
5	The digital I/O module then returns CTL to FALSE.	5	After waiting the programmed delay time, T_d, the digital I/O module sets CTL to FALSE.



NONE When handshake is set to NONE, no control or flag lines are used. The Agilent E1458A Digital I/O Module will input data or output data when programmed. The I/O line is set for output (LOW) before data is output. Data lines programmed for output will remain as output until another command is received.

Handshake NONE can be combined with the SCPI commands MEASure:DIGital:FLAG*n* and [SOURce:]DIGital:CONTrol*n* to create custom handshakes.

Handshake Timing

Handshake timing is set through the SCPI commands [SOURce:]DIGital:DATA*n*[:type]:HANDshake:DELay <*time*> or [SOURce:]DIGital:HANDshake*n*DELay <*time*>. Handshake timing is generally used for data output operations. Timing for data input affects only STRobe handshake mode.

Input/Output of Data Bytes and Bits

Data input is performed using commands in the SCPI MEASure:DIGital subsystem. Data output is performed using the commands in SCPI [SOURce:]DIGital subsystem.

The returned value of an input, or the TTL levels of an output, will depend upon the POLarity programmed for the port.

Both input and output operations will attempt to complete the handshake mode set for the port and may "hang" if required handshake operations are not completed.

Input

Input operations can involve single bits, 8-bit bytes, or multiple bytes. Single-bit input operations always return a decimal value of `0` or `1`. Byte or multiple-byte input operations always return numbers in decimal format.

Bit Input

The SCPI command for inputting the state of a single bit on a data port is:

MEASure:DIGital:DATA*n*[:type]:BIT*m*?

This command instructs the Agilent E1458A Digital I/O Module to return a value of either `0` or `1`, indicating the condition of bit *m* on port *n*, following completion of the input handshake. The value returned depends upon the programmed state of the port POLarity. In the default state (POSitive polarity) a TTL High on the data line specified by *m* will return a `1`. For example, the following BASIC program code will request and display the state of data line 3 (bit-3) on port 4.

```
120  OUTPUT @Dio;"MEAS:DIG:DATA4:BIT3?"
130  ENTER @Dio;Bits
140  DISP "State of bit 3 on port 4" ;Bits
```

Bit numbers range from 0 to 7 for single-port operations. For multiple-port operations, bit numbers can range from 0 to 95. The section "Multiple-Port Operations" later in this chapter describes bit numbering for multiple-port operations. For a single port, the data lines number and bit numbers correspond:

D*n*_7	D*n*_6	D*n*_5	D*n*_4	D*n*_3	D*n*_2	D*n*_1	D*n*_0
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0

In this manual the physical data lines are indicated as D*n*_1. The *n* should be replaced with the port number for the input operation. For example, bit 3 of port 6 affects the state of data line D6_3.

Byte Input

The SCPI command requesting an 8-bit byte from a data port is:

MEASure:DIGital:DATA*n*[:BYTE][:VALue]?

This command instructs the Agilent E1458A Digital I/O Module to return a decimal value between 0 and 255, indicating the condition of the data lines on port *n*, following completion of the input handshake. The value returned depends upon the programmed state of the port POLarity. In the default state (POSitive polarity) if all data lines are at a TTL Low level, the returned value will be `0`; if all lines are at a TTL High level, the returned value will be `255`. For example, the following BASIC program code will request and display the decimal value of the data lines on port 2.

```
120  OUTPUT @Dio;"MEAS:DIG:DATA2?"
130  ENTER @Dio;Result
140  DISP "Decimal value of port 2 data lines ";Result
```

Port numbers range from 0 to 11 for single-port operations. The section "Multiple-Port Operations" later in this chapter describes port numbering for multiple-port operations. For a single port, the returned decimal value will have the following correspondence to the port data lines:

D*n*_7	D*n*_6	D*n*_5	D*n*_4	D*n*_3	D*n*_2	D*n*_1	D*n*_0
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
MSB							LSB

Output

Output operations can involve single bits, 8-bit bytes, or multiple bytes. Single-bit output operations always expect a value of 0 or 1. Byte or multiple-byte output operations can accept numbers in decimal, hexadecimal, octal, or binary formats.

Bit Output

The SCPI command for setting the state of a single bit on a data port is:

[SOURce:]DIGital:DATA*n*[:type]:BIT*m* <*value*>

This command instructs the digital I/O module to set bit *m* on port *n* to <*value*>, using the output handshake. The actual TTL level set on the corresponding data line depends upon the programmed state of the port polarity. If <*value*> is 1 and the default polarity (POSitive polarity) is used, the data line corresponding to bit *m* will be set to a TTL High level. For example, the following BASIC program code will set the state of data line 2 (bit-2) on port 6 to a value of 1.

```
120  OUTPUT @Dio;"DIG:DATA6:BIT2 1"
```

Bit numbers range from 0 to 7 for single-port operations. For multiple-port operations, bit numbers can range from 0 to 95. The section "Multiple-Port Operations" later in this chapter describes bit numbering for multiple-port operations. For a single port, the data lines number and bit numbers correspond:

D*n*_7	D*n*_6	D*n*_5	D*n*_4	D*n*_3	D*n*_2	D*n*_1	D*n*_0
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0

In this manual the physical data lines are indicated as D*n*_1. The *n* should be replaced with the port number for the input operation. For example, bit 3 of port 6 affects the state of data line D6_3.

Byte Output

The SCPI command syntax to send an 8-bit byte to a data port is:

[SOURce:]DIGital:DATA*n*[:BYTE][:VALue] [<*base*>]<*value*>

This command instructs the Agilent E1458A Digital I/O Module to set the port *n* data lines to <*value*> using the output handshake. The optional parameter <*base*> defines the numbering system to use to implement <*value*> on the data lines. There are four values allowed for <*base*>:

no parameter decimal format
#H hexadecimal format
#Q octal format
#B binary format

The TTL levels set on the data lines depends upon the programmed port polarity. In the default state (POSitive polarity) a TTL High level will be set for any bit set to 1. For example, the following four BASIC program lines all perform the same function and set the same data lines on port 3:

```
120  OUTPUT @Dio;"DIG:DATA3 170"
120  OUTPUT @Dio;"DIG:DATA3  #HAA"
120  OUTPUT @Dio;"DIG:DATA3  #Q252"
120  OUTPUT @Dio;"DIG:DATA3  #B10101010"
```

If port 3 is in the default POSitive polarity mode, the TTL levels set on the data lines by any of the program lines above will be:

TTL level	High	Low	High	Low	High	Low	High	Low
Data line	D3_7	D3_6	D3_5	D3_4	D3_3	D3_2	D3_1	D3_0

Port numbers range from 0 to 11 for single-port operations. The section "Multiple-Port Operations" later in this chapter describes port numbering and byte order for multiple-port operations. For single-port operations, the most significant bit is bit 7. The table below shows the bit numbers and the corresponding data lines.

D*n*_7	D*n*_6	D*n*_5	D*n*_4	D*n*_3	D*n*_2	D*n*_1	D*n*_0
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
MSB							LSB

Multiple-Port Operations

The Agilent E1458A Digital I/O Module supports multiple-port operations. You can combine operations using 2, 4, 8, or all 12 ports with a single SCPI command. Multiple-port operations are shown in the SCPI command syntax as the optional keyword [:type]. For example, this SCPI command syntax initiates a handshake and returns one or more values:

MEAS:DIG:DATA*n*[:type]?

The optional keyword [:type] is replaced by one of the following keywords:

:BYTE	This keyword, or no keyword (default), is used for 8-bit port operations.
:WORD	This keyword is used to combine 2 adjacent ports for 16-bit port operations.
:LWORd or :LW32	Either keyword is used to combine 4 adjacent ports for 32-bit operations.
:LW64	This keyword is used to combine 8 adjacent ports for 64-bit operations.
:LW96	This keyword is used to combine all 12 ports for 96-bit operations.

The SCPI keyword :DATA*n* specifies the port to be used for operations by replacing *n* with the port number. Multiple-port operations have fixed values allowed for *n*. For all operations, if *n* is omitted, port 0 is assumed. The values allowed for *n* are:

Operation	Values of ***n***
:BYTE	0, 1, 2, 3, 4, 5, 6 ,7, 8, 9, 10 , or 11
:WORD	0, 2, 4, 6, 8, or 10
:LWORd	0, 4, or 8
:LW64	0
:LW96	0

For example, the following BASIC program code will obtain a decimal value of the state of the 32 data lines contained in ports 4, 5, 6, and 7.

```
120 OUTPUT @Dio;"MEAS:DIG:DATA4:LWORD?"
130 ENTER @Dio;Result
140 DISP "32 bit longword at port 4 "&Result
```

Multiple-Port Handshaking

The SCPI command syntax to establish a multiple-port handshake and set handshake timing is:

```
[SOURce:]DIGital:DATAn[:type]:HANDshake[:MODE] <mode>
[SOURce:]DIGital:DATAn[:type]:HANDshake:DELay <time>
```

The optional keyword [:type], parameter DATA*n*, handshake <*mode*>, and handshake delay <*time*> are all described earlier in this chapter. See the sections "Handshake Modes", "Handshake Timing", and the introduction to "Multiple-Port Operations" for explanations of these keywords and parameters.

Multiple-port handshaking has the following two anomalies regarding the CTL and FLG control lines:

Input or Output handshaking using the CTL line. The CTL line is set TRUE or FALSE sequentially on all ports involved in the operation, from the lowest-numbered port to the highest-numbered port. A slight time delay exists between each port setting the CTL line TRUE or FALSE. When using handshaking on multiple-port operations, use the highest-numbered port CTL line to ensure correct data transfer.

Input or Output handshaking using the FLG line. A change in the state of any FLG line on any combined port continues the handshake operation for all the combined ports. FLG lines can also be electrically combined through a switch setting (see Chapter 2).

Multiple-Port Input/Output

Data input is performed using commands in the SCPI MEASure:DIGigital:DATA*n*[:type] subsystem. Data output is performed using the commands in SCPI [SOURce:]DIGital:DATA*n*[:type] subsystem.

The returned value of an input, or the TTL levels of an output, will depend upon the POLarity programmed.

Both Input and Output operations will attempt to complete the handshake mode set and may "hang" if required handshake operations are not completed.

The sections "Byte Input" and "Byte Output" earlier in this chapter describe operations that also apply to multiple-port commands. The values used for input and output operations depend upon the [:type] used in the command. Values for multiple-port input or output operations are given in the following table:

	Input Operations		Output Operations	
	format	**range**	**format**	**range**
BYTE	Decimal	0 to 255	Decimal #H #Q #B	-128 to 255 00 to FF 000 to 377 8-bits
WORD	Decimal	-32768 to 32767	Decimal #H #Q #B	-32768 to 32767 0000 to FFFF 00000 to 177777 16-bits
LWORd	Decimal	-2147483648 to 2147483647	Decimal #H #Q #B	-2147483648 to 2147483647 00000000 to FFFFFFFF 0 to 37777777777 32-bits
LW64 *	Decimal	-2147483648 to 2147483647	Decimal #H #Q #B	-2147483648 to 2147483647 00000000 to FFFFFFFF 0 to 37777777777 32-bits
LW96 *	Decimal	-2147483648 to 2147483647	Decimal #H #Q #B	-2147483648 to 2147483647 00000000 to FFFFFFFF 0 to 37777777777 32-bits

* LW64 and LW96 operations use more than one <*value*>, each separated by a comma. The range for each of the values is indicated in the table.

For example, to set all 96 bits to 0, use the following command syntax:

```
SOUR:DIG:DATA0:LW96  0,0,0
```

Table 3-1 shows allowable port combinations for each value of [:type] and the order of the values for LW64 and LW96 operations.

You can combine multiple-port operations on the same Agilent E1458A Digital I/O Module. For example, you could define two multiple-data ports as a 64-bit port 0 and a 32-bit port 8.

Table 3-1. Port Combinations

8-bit (BYTE) Operations												
Port	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
Bit Numbers	7 — 0	7 — 0	7 — 0	7 — 0	7 — 0	7 — 0	7 — 0	7 — 0	7 — 0	7 — 0	7 — 0	7 — 0
Data Lines	D0_7 — D0_0	D1_7 — D1_0	D2_7 — D_0	D3_7 — D3_0	D4_7 — D4_0	D5_7 — D5_0	D6_7 — D6_0	D7_7 — D7_0	D8_7 — D8_0	D9_7 — D9_0	D10_7 — D10_0	D11_7 — D11_0
Parameters	<*value*>	<*value*>	<*value*>	<*value*>	<*value*>	<*value*>	<*value*>	<*value*>	<*value*>	<*value*>	<*value*>	<*value*>

16-bit (WORD) Operations												
Port	**0**		**2**		**4**		**6**		**8**		**10**	
Bit Numbers	15 — 8	7 — 0	15 — 8	7 — 0	15 — 8	7 — 0	15 — 8	7 — 0	15 — 8	7 — 0	15 — 8	7 — 0
Data Lines	D0_7 — D0_0	D1_7 — D1_0	D2_7 — D_0	D3_7 — D3_0	D4_7 — D4_0	D5_7 — D5_0	D6_7 — D6_0	D7_7 — D7_0	D8_7 — D8_0	D9_7 — D9_0	D10_7 — D10_0	D11_7 — D11_0
Parameters	<*value*>		<*value*>		<*value*>		<*value*>		<*value*>		<*value*>	

32-bit (LWORd) Operations												
Port	**0**				**4**				**8**			
Bit Numbers	31—24	23 —16	15— 8	7—0	31—24	23—16	15—8	7—0	31—24	23—16	15—8	7— 0
Data Lines	D0_7 — D0_0	D1_7 — D1_0	D2_7 — D_0	D3_7 — D3_0	D4_7 — D4_0	D5_7 — D5_0	D6_7 — D6_0	D7_7 — D7_0	D8_7 — D8_0	D9_7 — D9_0	D10_7 — D10_0	D11_7 — D11_0
Parameters	<*value*>				<*value*>				<*value*>			

64-bit (LW64) Operations												
Port	**0**											
Bit Numbers	63—56	55—48	47—40	39—32	31—24	23—16	15—8	7—0				
Data Lines	D0_7 — D0_0	D1_7 — D1_0	D2_7 — D_0	D3_7 — D3_0	D4_7 — D4_0	D5_7 — D5_0	D6_7 — D6_0	D7_7 — D7_0				
Parameters	1st <*value*>				2nd <*value*>							

96-bit (LW96) Operations												
Port	**0**											
Bit Numbers	95—88	87—80	79—72	71—64	63—56	55—48	47—40	39—32	31—24	23—16	15—8	7—0
Data Lines	D0_7 — D0_0	D1_7 — D1_0	D2_7 — D_0	D3_7 — D3_0	D4_7 — D4_0	D5_7 — D5_0	D6_7 — D6_0	D7_7 — D7_0	D8_7 — D8_0	D9_7 — D9_0	D10_7 — D10_0	D11_7 — D11_0
Parameters	1st <*value*>				2nd <*value*>				3rd <*value*>			

Using the UTS Control Line

The UTS line can be used to prevent the Agilent E1458A Digital I/O Module from sourcing data on the data lines, independent of the programmed state of the port. A high or open on the UTS line allows the port to operate normally. When the UTS line is set LOW, the data line transmitter is disabled. The UTS line can be connected and used in either a static or active configuration.

A port is set to transmit on the data lines when a SOUR:DIG:DATA command is received and the programmed handshake is completed. Use the SOUR:DIG:DATA:VAL? command to verify the programmed state of the port. The SOUR:DIG:DATA:MON? command returns the actual state of the data lines. These two commands may return different values if the UTS line is LOW.

Static Configuration

The UTS line can disable the port transmitter during program development/debugging. Tie the UTS line for each port to ground. This configuration ensures that the digital I/O module cannot transmit on the data lines, independent of the programmed state of the port. The port data receivers are not affected.

Active Configuration

The UTS line can be connected to an active signal from the peripheral to ensure that digital I/O module does not attempt to transmit, independent of the programmed state of the port. The peripheral should hold the UTS line LOW when it is transmitting, and release the line when it is ready to receive. For example, two Agilent E1458A Digital I/O Modules could be connected as shown in Figure 3-1. The I/$\overline{O}$ line is set to a TTL Low when the port is programmed to transmit. By connecting the I/$\overline{O}$ lines from each port to the UTS lines, the two modules cannot transmit at the same time.

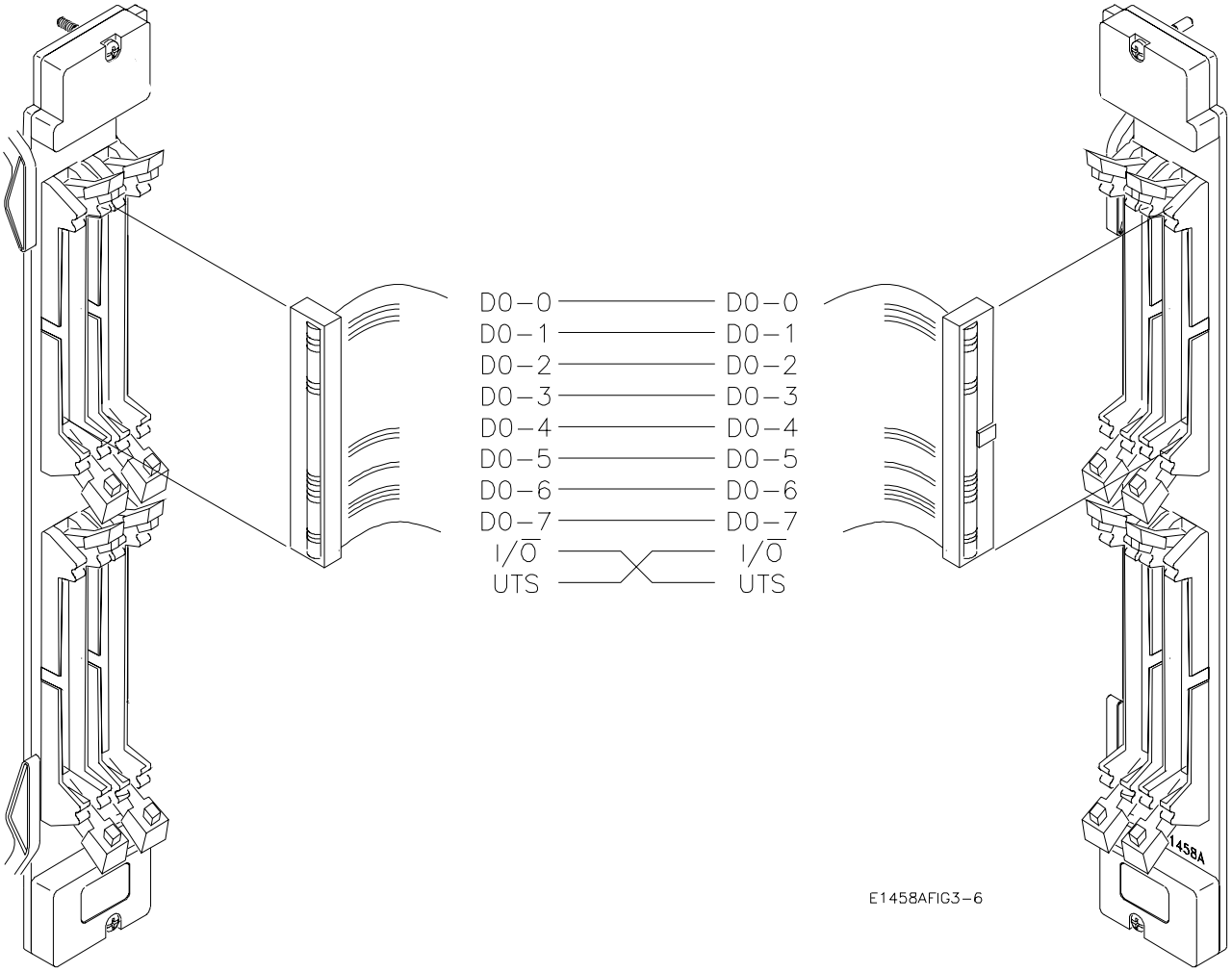


Figure 3-1. UTS Line Active Configuration

Using as an Open Collector Output

The Agilent E1458A Digital I/O Module data lines can be used in an open collector configuration. Connections for open collector require the use of external power supplies and pull-up resistors. The internal pull-up mode of the digital I/O module MUST be disabled for open collector output. Figure 3-2 shows a single data line connection.

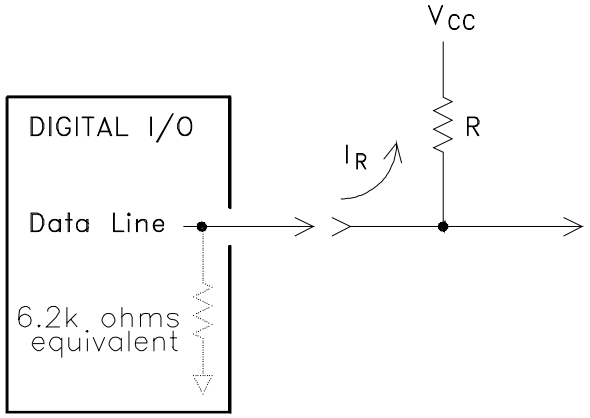


Figure 3-2. Typical Open Collector Data Line

The value of the pull-up resistor is calculated as follows:

$$Vcc = 5.0\ Vdc$$
$$Imax = Iout_{Low} * safety\_factor = 48\ mA * 0.52 = 25\ mA$$
$$R = \frac{Vcc}{Imax} = \frac{5}{0.025} = 200\ \Omega$$

The value of TTL High with the 200 **Ω** pull-up resistor is calculated as follows:

$$V_{High} = V_{cc} * \frac{6200}{6200 + 200} = 4.84\ V_{dc}$$

Typical Connection

Figure 3-3 shows a typical driver/receiver connection for data transfer. The FLG, PIR, and STS lines have a resistive pull-up network. The data lines do not have a resistive pull-up, but can use an internal pull-up in the SN75ALS160. The internal pull-up requires that the data lines sink 3.2 mA to pull the line to less than 0.4 V. The I/$\overline{O}$, CTL, and $\overline{RES}$ lines are open collector and require external pull-up to logic high.

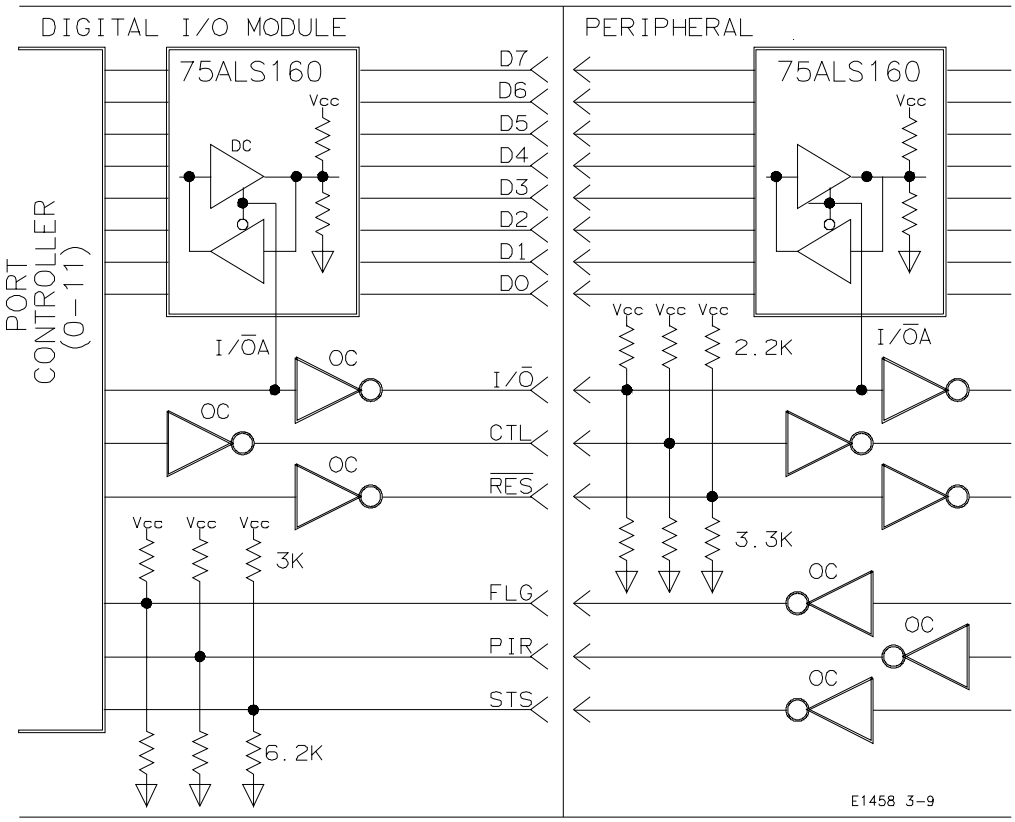


Figure 3-3. Typical Driver/Receiver Connections

Program Examples

The following BASIC program examples demonstrate a few of the uses of the Agilent E1458A Digital I/O Module. The hardware setup needed to support the example is described.

Checking Data Lines

This example checks for stuck data lines on all ports. It demonstrates sending data, setting bits, and checking the result. If the programmed state of the port is not correct, the program prints a message and stops. If a fast check of the data lines reveals a stuck line, then a bit by bit check of the data lines is performed and any errors found are printed.

Hardware setup: This example assumes that all 96 data lines are connected to a peripheral device. Further, all 96 data lines must be provided with some kind of pull-up (either internal or external).

```
10   RE-SAVE "Chk_line"                          !
20   ASSIGN @Dio TO 70918                        ! I/O path to digital I/O module.
30   DIM Pattern(1:3,0:1),Value(1:3),            !
     Mon(1:3)
40   INTEGER Errflg,Ready ,Bits                  !
50   Errflg = 0                                  ! Set to no error.
60   DATA   -1431655766,1431655765,              ! Alternating 0 and 1, 32-Bit .
            -1431655766,1431655765,
            -1431655766, 1431655765
70   READ Pattern(*)                             !
80   OUTPUT @Dio;"*RST;*OPC?"                    ! Establish defaults.
90   ENTER @Dio;Ready                            ! Wait for completion.
100  FOR I = 0 TO 1                              ! Fast test of the data lines.
110      OUTPUT @Dio;"SOUR:DIG                   ! Alternate 0 and 1.
         :DATA0:LW96 ";Pattern(1,I);",";
         Pattern(2,I);      ",";Pattern(3,I)
120      OUTPUT @Dio;"*OPC?"                     !
130      ENTER @Dio;Ready                        ! Wait for completion .
140      OUTPUT @Dio;"SOUR:DIG:                  ! Readback of port
         DATA0:LW96?"                            programming.
150      ENTER @DIO;Value(1),Value(2),           !
         Value(3)
160      FOR J=1 TO 3                            !
170        IF Val(J)  <> Pattern(J,I) THEN       ! Port not programmed.
180          PRINT "Port not programmed"         !
190          STOP                                !
200        END IF                                !
210      NEXT J                                  !
220      OUTPUT @Dio;"SOUR:DIG                   ! Read data lines.
         :DATA0:LW96:MON?"
230      ENTER @Dio;Mon(1),Mon(2),Mon(3)!
```

Continued on next page.

```
240       FOR J=1 TO 3                        !
250         IF Mon(J) <> Pattern(J,I) THEN    ! Data line error.
260           PRINT "One or more              !
               data lines stuck"
270           Errflg = 1                      ! Set error flag.
280         END IF                            !
290       NEXT J                              !
300  NEXT I                                   !
310  IF Errflg = 1 THEN                       ! Find the stuck line(s).
320     OUTPUT @Dio;"*RST;*OPC?"              ! Restore defaults.
330     ENTER @Dio;Ready                      ! Wait for completion.
340     FOR I = 0 TO 1                        ! I is logic counter.
350       FOR J = 0 TO 95                     ! J is bit counter.
360           OUTPUT @Dio;"SOUR:DIG           ! Set a bit.
              :DATA0:LW96:BIT"&VAL$(J)&
              " " &VAL$(I)$";*OPC?"
370           ENTER @Dio;Ready                ! Wait for completion.
380           OUTPUT @Dio;"SOUR               ! Get the state of the bit.
              :DIG:DATA0:LW96:BIT"
              &VAL$(J)&":MON?"
390           ENTER @Dio;Bits                 !
400           IF Bits  <> I THEN              !
410             PRINT "Logical ";I;" at bit ";J;!
                " error found"
420           END IF                          !
430           NEXT J                          !
440       NEXT I                              !
450  ELSE                                     !
460       PRINT " Data Line test passed"      !
470  END IF                                   !
480  END                                      !
```

Setting Polarity and Handshake

This example demonstrates setting the port polarity and handshake mode for a 16-bit port. Handshake mode LEADing will be set and the polarity of the FLG and CTL lines will be inverted.

Hardware setup: Port 0 and 1 data lines are connected to corresponding perpheral data lines. Port 1 CTL and FLG lines are used and are connected to the handshake lines of the perpherial. Port 0 CTL and FLG lines are not used or connected.

```
10  RE-SAVE "Pol_hnd                          !
20  ASSIGN @Dio TO 70918                      !
30  INTEGER Ready, Words,Bits                 !
40  OUTPUT @Dio;"*RST;*OPC?"                  !
50  ENTER @Dio;Ready                          ! Wait for completion.
60  OUTPUT @Dio;"SOUR:DIG:CONT1               ! Set CTL polarity.
      :POL NEG;*OPC?"
70  ENTER @Dio;Ready                          ! Wait for completion.
80  OUTPUT @Dio;"SOUR:DIG:FLAG1               !Set FLG polarity.
      :POL NEG;*OPC?"
90  ENTER @Dio;Ready                          ! Wait for completion.
100 OUTPUT @Dio;"SOUR:DIG:DATA0               !Set output handshake mode.
      :WORD:HAND LEAD;*OPC?"!
110 ENTER @Dio;Ready                          !Wait for completion.
120 OUTPUT @Dio;"SOUR:DIG:DATA0               ! Send 6565 hex using
      :WORD #H6565;*OPC?"                       hardware handshake.
130 ENTER @Dio;Ready                          !
140 OUTPUT @Dio;"SOUR:DIG:DATA:               ! Send AAAA hex using
      :WORD #HAAAA;"*OPC?"                      hardware handshake.
150 ENTER @Dio;Ready
160 OUTPUT @Dio;"MEAS:DIG:DATA0               ! Read the 16-bit port using
      :WORD?"                                   hardware handshake.
170 ENTER@Dio;Words                           !
180 PRINT "Integer value of port 0 ";Words    !
190 OUTPUT @Dio;"MEAS:DIG:DATA0               ! Read the state of a bit using
      :WORD:BIT9?"                              hardware handshake.
200 ENTER @Dio;Bits                           !
210 PRINT "Bit 9, data line D1_1, value ";Bits !
220 END                                       !
```

Using Trace Memory

These examples demonstrate using trace memory in various configurations.

Hardware setup: Example 2 uses an external VME memory card. You can also use shared memory on the command module for the Agilent E1405B/E1406. The memory starting address is usually at #H200000. (Use the VXI:CONF:DLIS? query to locate the address).

NOTE Byte swapping may occur when using the :TRACE commands with Agilent SCPI. If you are using a Motorola processor, the bytes are written or read to memory with the lowest port receiving the least significant byte (the case when directly addressing the port through SCPI commands). An Intel processor, however, when used with the :TRACE commands will swap the order the of the bytes. The bytes are written or read from memory with the lowest port receiving the most significant byte and the highest port the least significant byte.

Trace Memory Example 1

This example writes 20 bytes as 10 WORDS at ports 0 and 1.

```
10  RE-SAVE "Trace_1"                          !
20  ASSIGN @ Dio TO 70918                      !
30  INTEGER A(1:10) ,Ready                     !
40  DATA 65,66,67,68,69,70,71,                 ! A, B, C, D, E, F, G, H, I, J
    72,73,74
50  READ A(*)                                  !
60  OUTPUT @Dio;"*RST;*OPC?"                   !
70  ENTER @Dio;Ready                           ! Wait for completion.
80  OUTPUT @Dio;"SOUR:DIG                      ! Define memory name alpha.
    :TRAC:DEF alpha,100;*OPC?"
90  ENTER @Dio;Ready                           ! Wait for completion.
100 OUTPUT @Dio USING "K,10(W)                 ! Fill memory alpha with 20
    ";"SOUR:DIG:TRACE alpha,                   bytes.
    #220";A(*)
110 OUTPUT @Dio;"SOUR:DIG                      ! Output the 20 bytes.
    :DATA0:WORD:TRAC alpha;
    *OPC?"
120 ENTER @Dio;Ready                           ! Wait for completion.
130 OUTPUT @Dio;"SOUR:DIG                      ! Remove memory block.
    :TRAC:DEL alpha;
    *OPC?"
140 ENTER @Dio;Ready                           ! Wait for completion.
150 END
```

Trace Memory Example 2

This example writes 20 bytes as 10 WORDS at ports 0 and 1 as in the first example, it uses an external VME memory board.

```
10  RE-SAVE "Trace"_2                          !
20  ASSIGN @ Dio TO 70918                      !
30  INTEGER A(1:10) ,Ready                     !
40  DATA 65,66,67,68,69,70,71,                 ! A, B, C, D, E, F, G, H, I, J
    72,73,74
50  READ A(*)                                  !
60  OUTPUT @Dio;"*RST;*OPC?"                   !
70  ENTER @Dio;Ready                           ! Wait for completion.
80  OUTPUT @Dio;"MEM:VME                       ! Define memory location.
    :ADDR #H200000;*OPC?"
90  ENTER @Dio;Ready                           ! Wait for completion.
100 OUTPUT @Dio;"MEM:VME                       ! Reserve 100 bytes.
    :SIZE  100;*OPC?"
110 ENTER @Dio;Ready                           ! Wait for completion.
120 OUTPUT @Dio;"MEM:VME                       ! Enable memory.
    :STAT  ON;*OPC?"
130 ENTER @Dio;Ready                           ! Wait for completion.
140 OUTPUT @Dio;"SOUR:DIG                      ! Define memory name alpha.
    :TRAC:DEF alpha,100;*OPC?"
150 ENTER @Dio;Ready                           ! Wait for completion.
```

Continued on next page.

```
160 OUTPUT @Dio USING                    ! Fill memory alpha with 20
     "K,10(W)";"SOUR:DIG:TRACE           bytes.
    alpha,#220";A(*)
170 OUTPUT @Dio;"SOUR:DIG                ! Output the 20 bytes.
    :DATA0:WORD:TRAC alpha;
    *OPC?"
180 ENTER @Dio;Ready                     ! Wait for completion.
190 OUTPUT @Dio;"SOUR:DIG:TRAC           ! Remove memory block.
    :DEL alpha;*&OPC?"
200 ENTER @Dio;Ready                     ! Wait for completion.
210 END
```

Trace Memory Example 3

This example reads 40 WORDS from ports 0 and 1.

```
10  RE-SAVE "Trace_3"                    !
20  ASSIGN @ Dio TO 70918                !
30  DIM Head$[4]                         !
40  INTEGER A(1:20) ,Ready               !
50  OUTPUT @Dio;"*RST;*OPC?"             !
60  ENTER @Dio;Ready                     ! Wait for completion.
70  OUTPUT @Dio;"SOUR:DIG                ! Define memory name alpha.
    :TRAC:DEF alpha,80;*OPC?"
80  ENTER @Dio;Ready                     ! Wait for completion.
90  OUTPUT @Dio;"MEAS:DIG                ! Output 80 bytes.
    :DATA0:WORD:TRACE alpha;
    *OPC?"
100 ENTER @Dio;Ready                     ! Wait for completion.
110 OUTPUT @Dio;"SOUR:DIG                ! Request the data.
    :TRAC:DATA? alpha"
120 ENTER @Dio USING "4A,40(W)";         !
    Head$;A(*)
130 OUTPUT @Dio;"SOUR:DIG                ! Remove memory block.
    :TRACE:DEL alpha;*OPC?"
140 ENTER @Dio;Ready                     ! Wait for completion.
150  END
```

Chapter 4
Agilent E1458A Command Reference

Using This Chapter

This chapter describes Standard Commands for Programmable Instruments (SCPI) and summarizes IEEE 488.2 Common (*) commands applicable to the Agilent E1458A Digital I/O Module. This chapter contains the following sections:

Command Fundamentals

Commands are separated into two types: IEEE 488.2 Common commands and SCPI commands.

Common Command Format

The IEEE 488.2 standard defines the Common commands that perform functions like reset, self-test, status byte query, etc. Common commands are four or five characters in length, always begin with the asterisk character (*), and may include one or more parameters. The command keyword is separated from the first parameter by a space character. Common commands are not documented, except in a general manner, in this manual. Some examples of Common commands are:

```
*RST
*ESR 32
*STB?
```

SCPI Command Format

SCPI commands perform functions like making measurements, querying instrument states, or retrieving data. The command structure is a hierarchical structure that usually consists of a top-level (or root) command, one or more lower-level commands, and their parameters. The following example shows part of a typical subsystem:

```
[SOURce:]
    DIGital
        :DATAn
            [:VALue]?
            :BITm?
```

[SOURce:] is the root command, DIGital is a second-level command, :DATA*n* is a third-level command (where *n* is the port number 0 - 11), and :VALue and :BIT*m* are fourth-level commands (where *m* is the queried bit location).

Command Separator

A colon (**:**) always separates one command from the next lower-level command. This is illustrated as follows:

MEASure**:**DIGital**:**DATA*n***:**VALue?

Colons separate the root command from the second level (MEASure:DIGital) and the second from the third level (DIGital:DATA*n*), and so forth.

Abbreviated and Short Commands

The command syntax shows most commands as a mix of upper- and lowercase letters. The uppercase letters indicate an abbreviated spelling for the command. For shorter program lines, send only the abbreviated form. For better program readability use the entire command. The instrument will accept either the abbreviated form or the entire command.

For example, if the command reference syntax shows the command MEASure, then MEAS and MEASURE are both acceptable forms. Other forms of MEASure, such as MEASU or MEASUR will generate an error.

The instrument does not distinguish between uppercase and lowercase characters. Therefore MEASURE, measure, and MeAsUrE are all acceptable.

Command keywords can be entered in their full form as shown above, or can be entered in their short form. In this manual, the entry required in short form commands is always capitalized. The short form is generally used for examples in this manual.

Implied Commands

Implied commands appear in square brackets (**[ ]**) in the command syntax. (Note that the brackets are *not* part of the command and are not sent to the instrument.) Suppose you send a second-level command but do not send the preceding implied command. In this case, the instrument assumes you intend to use the implied command and it responds as if you had sent it. Examine this excerpt from the [SOURce:] subsystem shown below:

```
[SOURce:]
    DIGital
        :DATAn
            [:VALue] <parameter>
            :BITm <parameter>
```

The root command [SOURce:] is implied. To set the instrument to output a logical 1 to bit 0 of port 3, you may send either:

```
[SOURce:]DIGital:DATA3:BIT0 1
```

or

```
DIGital:DATA3:BIT0 1
```

Parameters

Parameter Types. The following table contains explanations and examples of parameter types you might see later in this chapter. Parameters *must* always be separated from the keywords by a space.

Parameter Type	Explanations and Examples
Numeric	Accepts all commonly used decimal representations of numbers including optional signs, decimal points, and scientific notation. 123, 123E2, -123, -1.23E2, .123, 1.23E-2, 1.23000E-01. Special cases include MIN, MAX, and DEF.
Boolean	Represents a single binary condition that is either true or false. ON, OFF, 1, 0.
Discrete	Selects from a finite number of values. These parameters use mnemonics to represent each valid setting. An example is the DIGital:CONTrol*n*:POLarity <*polarity*> command where the parameter <*polarity*> can be either POS or NEG.

Optional Parameters. Parameters shown within square brackets (**[ ]**) are optional parameters. (Note that the brackets are not part of the command and are *not* sent to the instrument.) If you do not specify a value for an optional parameter, the instrument chooses a default value. For example, consider the DISPlay:MONitor:PORT? [<MIN | MAX | DEF>] command. If you send the command without specifying a parameter, the command returns the number of the port last addressed. If you send the MIN parameter or the DEF parameter, the command returns `0`. If you send the MAX parameter, the command returns `11`. Be sure to place a space between the command and the parameter.

Keyword Substitutions

Some commands indicate a keyword substitution by showing the substitution in square brackets and lowercase letters. (It is important to note that this modifies the keyword and should not be confused with a parameter name.) For example, in the SCPI command MEASure:DIGital:DATA*n*[:type]**,** the keyword [:type] should be replaced by one these parameters:

- :BYTE
- :WORD
- :LWORd or :LW32
- :LW64
- :LW96

Linking Commands

Linking IEEE 488.2 Common Commands with SCPI Commands. Use a semicolon (**;**) between the commands. For example:

```
*RST;DIG:CONT2 1  or  DIG:CONT2:POL POS;*IDN?
```

Linking Multiple SCPI Commands. Use both a semicolon (**;**)and a colon (**:**) between the commands. For example:

```
DIG:DATA2:POL NEG;:DIG:DATA2:BIT1 1
```

SCPI Command Reference

The following sections describe the Standard Commands for Programmable Instruments (SCPI) commands for the Agilent E1458A Digital I/O Module. Commands are listed alphabetically by subsystem and within each subsystem.

The DISPlay subsystem turns on the monitor mode. Parameters related to the state of the data and control lines are shown on the external terminal. Refer to the *C-Size VXIbus Systems Configuration Guide* for supported terminal types. The parameters displayed are:

- port number
- polarity
- handshake mode
- state of the control line
- state of the flag line
- values on the data lines

Subsystem Syntax

```
DISPlay
    :MONitor
        :PORT <port | AUTO>
        :PORT? [<MAX | MIN | DEF>]
        [:STATe] <OFF | ON>
        [:STATe]?
```

DISPlay:MONitor:PORT

DISPlay:MONitor:PORT [<*port*>] sets the displayed port number.

Parameters

Parameter Name	Parameter Type	Range of Values	Default
<*port*>	Numeric or Discrete	none, 0 through 11 MIN | MAX | AUTO | DEF	AUTO

Comments

- In the AUTO mode of operation, the display shows the state of the port last programmed. MIN sets port 0. MAX sets port 11. No parameter or DEF sets AUTO mode of operation.
- **Related Commands:**
  DISPlay:MONitor[:STATe]
  DISPlay:MONitor:PORT?
- ***RST Condition:** AUTO

Example

DISP:MON:PORT 3 sets the port to be monitored to 3.

DISPlay:MONitor:PORT?

DISPlay:MONitor:PORT? [<MAX | MIN | DEF>], with no parameter, returns a decimal number indicating the port being monitored. If AUTO was selected as the port parameter in the DISP:MON:PORT AUTO command, the query returns a `-1`. If DEF is specified, the query always returns `-1`. If MAX is specified, the query returns the maximum port (always `11`). If MIN is specified, the query returns the minimum port (always `0`).

Parameters

Parameter Name	Parameter Type	Range of Values	Default
[<MAX | MIN | DEF>]	Optional or Discrete	None MAX, MIN, or DEF	None

Comments

- **Related Commands:**
  DISPlay:MONitor:PORT
  DISPlay:MONitor[:STATe]
- ***RST Condition:** Not applicable.

Example

DISP:MON:PORT? identifies the port being monitored.

DISPlay:MONitor[:STATe]

DISPlay:MONitor[:STATe] <OFF | ON> turns the monitor mode on and off.

Parameters

Parameter Name	Parameter Type	Range of Values	Default
<OFF | ON>	Boolean	0 | 1, OFF | ON	None

Comments

- **DISP:MON ON** enables the terminal display of port parameters. The parameters are updated to the terminal following each new command accessing a port.
- A keyboard entry at the terminal will set DISP:MON OFF.
- This command differs from the SOUR:DIG:DATA*n*[:type]:MON? command; this command does not perform an actual readback of the port data lines. It returns the last programmed state of the data lines.

- **Related Commands:**
  DISPlay:MONitor:PORT
  DISPlay:MONitor:PORT?

- ***RST Condition:** OFF

Example **DISP:MON ON** displays the state of the last port programmed.

DISPlay:MONitor[:STATe]?

DISPlay:MONitor[:STATe]? returns the state of the monitor mode.

Parameters None

Comments

- DISPlay:MONitor[:STATe]? returns a `1` if the monitor mode is on or a `0` if the monitor mode is off.

The MEASure subsystem defines the command set for the Agilent E1458A Digital I/O Module input statements

Subsystem Syntax

```
MEASure
    :DIGital
        :DATAn
            [:BYTE]
                :BITm?
                :TRACe <name>
                [:VALue]?
            :WORD
                :BITm?
                :TRACe <name>
                [:VALue]?
            :LWORd or :LW32
                :BITm?
                :TRACe <name>
                [:VALue]?
            :LW64
                :BITm?
                :TRACe <name>
                [:VALue]?
            :LW96
                :BITm?
                :TRACe <name>
                [:VALue]?
        :FLAGn?
```

MEASure:DIGital:DATA*n*[:type][:VALue]?

MEASure:DIGital:DATA*n*[:BYTE][:VALue]? reads one byte from 8-bit port *n* after the completion of the handshake and returns a decimal number between 0 and 255.

MEASure:DIGital:DATA*n*:WORD[:VALue]? reads two bytes (one word) from 16-bit port *n* after the completion of the handshake and returns a decimal number between -32768 and 32767.

MEASure:DIGital:DATA*n*:LWORd[:VALue]? reads four bytes (one longword) from the 32-bit port *n* after the completion of the handshake and returns a decimal number between -2^{31} and $(2^{31}-1)$.

MEASure:DIGital:DATA*n*:LW64[:VALue]? reads eight bytes from the 64-bit port after the completion of the handshake and returns two decimal numbers, each between -2^{31} and $(2^{31}-1)$.

MEASure:DIGital:DATA*n*:LW96[:VALue]? reads 12 bytes from the 96-bit port after the completion of the handshake and returns three decimal numbers, each between -2^{31} and $(2^{31}-1)$.

Parameters

Parameter Name	Parameter Type	Range of Values	Default
DATA*n*	Numeric	BYTE none, 0 through 11 WORD none, 0, 2, 4, 6, 8, or 10 LWORd none, 0, 4, or 8 LW64 none or 0 LW96 none or 0	0

Comments

- Input data from the digital I/O module is returned in decimal format. Other formats are not supported for input. However, data output to the digital I/O module may be in binary, octal, decimal, or hexadecimal.
- If *n* is omitted, the port number is set to 0.
- The keyword LW32 may be used instead of LWORd.
- Chapter 3, "Using the Agilent E1458A Digital I/O Module," describes the byte order of mutltiple-byte reads.
- **:**DATA*n* is the keyword used for commands relating to the data at port *n*. The port number *n* must be the last character of the keyword without spaces.

- **Related Commands:**
  [SOURce:]DIGital:DATA*n*[:type][:VALue]
  MEASure:DIGital:DATA*n*[:type]:BIT*m*?

- ***RST Condition:** Set to input positive true on all ports.

Examples **MEAS:DIG:DATA1?** reads 8-bit port 1 data. If all data lines are set to 1, this command returns the value `255`.

MEAS:DIG:DATA0:LW64? reads 64-bit port 0 data. If all data lines are set to 1, this command returns the values `-1` and `-1`.

MEAS:DIG:DATA0:LW96? reads 96-bit port 0 data. If all data lines are set to 1, this command returns the values `-1`, `-1` and `-1`.

MEASure:DIGital:DATA*n*[:type]:BIT*m*?

MEASure:DIGital:DATA*n*:BYTE:BIT*m*? reads the state of bit *m* of 8-bit port *n* after the completion of the handshake.

MEASure:DIGital:DATA*n*:WORD:BIT*m*? reads the state of bit *m* of 16-bit port *n* after the completion of the handshake.

MEASure:DIGital:DATA*n*:LWORd:BIT*m*? reads the state of bit *m* of the 32-bit port *n* after the completion of the handshake.

MEASure:DIGital:DATA*n*:LW64:BIT*m*? reads the state of bit *m* of the 64-bit port after the completion of the handshake.

MEASure:DIGital:DATA*n*:LW96:BIT*m*? reads the state of bit *m* of the 96-bit port after the completion of the handshake.

Parameters

Parameter Name	Parameter Type	Range of Values	Default
DATA*n*	Numeric	BYTE none, 0 through 11 WORD none, 0, 2, 4, 6, 8, or 10 LWORd none, 0, 4, or 8 LW64 none or 0 LW96 none or 0	0
BIT*m*	Numeric	BYTE 0-7 WORD 0-15 LWORd 0-31 LW64 0-63 LW96 0-95	0

Comments

- Input data is always assumed to be in binary format since only a single bit of data is being read. The command returns either a `0` or `1`.
- The keyword LW32 may be used instead of LWORd.
- :DATAn is the keyword used for commands relating to the data at port *n*. The port number *n* must be the last character of the keyword without spaces.
- :BIT*m* is the keyword that specifies the bit read by this command. Like the DATA*n* keyword, no space can be between the keyword BIT and the *m* parameter.
- If *n* is omitted, port 0 is used. If *m* is omitted, bit 0 is used.
- **Related Commands:** SOURce:DIGital:DATA*n*:POLarity
- ***RST Condition:** Set to input on all ports.

Example

MEAS:DIG:DATA2:BIT4? reads port 2, bit 4 (data line D2-4).

MEAS:DIG:DATA0:LW96:BIT95? reads port 0, bit 95 (data line D0-7).

MEASure:DIGital:DATA*n*[:type]:TRACe

MEASure:DIGital:DATA*n*[:BYTE]:TRACe <*name*> reads 8-bit port *n* after the completion of the handshake and stores the data block in <*name*>.

MEASure:DIGital:DATA*n*:WORD:TRACe <*name*> reads 16-bit port *n* after the completion of the handshake and stores the data block in <*name*>.

MEASure:DIGital:DATA*n*:LWORd:TRACe <*name*> reads the 32-bit port *n* after the completion of the handshake and stores the data block in <*name*>.

MEASure:DIGital:DATA*n*:LW64:TRACe <*name*> reads the 64-bit port after the completion of the handshake and stores the data block in <*name*>.

MEASure:DIGital:DATA*n*:LW96:TRACe <*name*> reads the 96-bit port after the completion of the handshake and stores the data block in <*name*>.

Parameters

Parameter Name	Parameter Type	Range of Values	Default
DATA*n*	Numeric	BYTE none, 0 through 11 WORD none, 0, 2, 4, 6, 8, or 10 LWORd none, 0, 4, or 8 LW64 none or 0 LW96 none or 0	0
<*name*>	String	previously defined block name (max 12 characters)	None

Comments

- :TRACe <*name*> is the keyword (maximum 12 characters) that specifies the block where the data should be stored. This block must have been previously defined by the [SOURce]:DIGital:TRACe:DEFine command.
- This command will completely fill the named block. The defined block size sets the amount of data read. The block size must by an integer multiple of the [:type] keyword used in this command. For example, valid block sizes for LWORd are 4, 8, 12, 16 etc.
- Input data is returned in decimal format. Other formats are not supported for input, however data output may be in binary, octal, decimal or hexadecimal.
- The keyword LW32 may be used instead of LWORd.
- :DATA*n* is the keyword used for commands relating to the data at port *n*. The port number *n* must be the last character of the keyword without spaces.

- **Related Commands:**
  MEASure:DIGital:DATA*n*[:VALue]?
  SOURce:DIGital:TRACe:DEFINE <*name*>

- ***RST Condition:** Set to input on all ports.

Example MEAS:DIG:DATA0:WORD:TRACe *first_block* reads 16-bit data from port 0 and stores it in the predefined user memory location *first_block*.

MEASure:DIGital:FLAG*n*?

MEASure:DIGital:FLAG*n*? reads the status of the flag line on port *n* and returns a `0` or `1` to show whether a peripheral has set the flag line to READY or BUSY.

Parameters

Parameter Name	Parameter Type	Range of Values	Default
FLAG*n*	Numeric	none, 0 through 11	0

Comments

- MEASure:DIGital:FLAG*n*? is used to implement custom handshakes. The handshake mode must be set to NONE to use this command.

- :FLAG*n* is the keyword used for commands relating to the flag line at port *n*. The port number *n* must be the last character of the keyword without spaces.

- If *n* is omitted FLAG0 is used.

- MEASure:DIGital:FLAG*n*? may be affected by the condition of the flag combining jumpers. Refer to Chapter 2 for additional information.

- **Related Commands:**
  [SOURce:]DIGital:CONTrol*n*:POLarity?
  [SOURce:]DIGital:CONTrol*n*[:VALue]
  [SOURce:]DIGital:FLAG*n*:POLarity
  [SOURce:]DIGital:FLAG*n*:POLarity?

Example MEAS:DIG:FLAG1? reads the port1 flag line.

The MEMory subsystem defines the command set for enabling the use of external VME memory or memory on the command module for storing traces and macros. The addressable range is #H200000 through #HDFFFF8 in A24 space.

Subsystem Syntax

```
MEMory
    DELete:MACRo <name>
    VME:
        ADDRess [<base>]<address>
        ADDRess? [MIN | MAX]
        SIZE [<base>]<size>
        SIZE? [MIN | MAX]
        STATe <state>
        STATe?
```

MEMory:DELete:MACRo

MEMory:DELete:MACRo <*name*> deletes a single macro previously recorded using the *DMC Common command.

Parameters

Parameter Name	Parameter Type	Range of Values	Default
<*name*>	String	Previously defined block name (maximum 12 characters)	None

Comments

- <*name*> must have been previously defined by a *DMC (Define Macro) Common command.
- The maximum length for <*name*> is 12 characters.
- This command purges a single, specific macro; the *PMC Common command purges all macros.

Example

MEM:DEL:MACR test_macro deletes macro named *test_macro* previously defined using the *DMC Common command.

MEMory:VME:ADDRess

MEMory:VME:ADDRess [<*base*>]<*address*> establishes the address of add-on VME memory in the system which can then be used to store block data in commands with the :TRACE keyword.

Parameters

Parameter Name	Parameter Type	Range of Values	Default
<*base*>	Discrete	none, #H, #Q, or #B	Decimal
<*address*>	Numeric or Discrete	200000_{16} - $DFFFF8_{16}$ MIN or MAX	None

Comments

- <*base*> specifies the numeric format as decimal, hexadecimal, octal, or binary. IEEE-488.2 specifies the following values for this parameter:

  Decimal = no parameter
  Hexadecimal = #H
  Octal = #Q
  Binary = #B

- Valid values for <*base*><*address*> are #H200000 (2,097,152 decimal) through #HDFFFF8 (14,680,056 decimal).

- For this memory to actually be used it must also have a defined length and have been turned ON using the MEMory:VME:STATe command.

- **Related Commands:**
  SOUR:DIG:TRAC:DEF <*name*>,<*size*>
  MEMory:VME:ADDress?
  MEMory:VME:SIZE <*size*>
  MEMory:VME:STATe <ON or OFF>

- ***RST Condition:** #H200000.

Example

MEM:VME:ADDR #H200000 sets the starting VME address to 200000_{16}.

MEMory:VME:ADDRess?

MEMory:VME:ADDRess? [<MIN | MAX>] queries for the current VME memory address. The optional parameter lets you query for the fixed minimum or maximum address.

Parameters

Parameter Name	Parameter Type	Range of Values	Default
<MIN | MAX>	Discrete	none, MIN, or MAX	None

Comments

- This command always returns the address in decimal format**.**
- The address returned using MIN is always `2,097,152`.
- The address returned using MAX is always `14,680,056`.
- **Related Commands:**
  MEMory:VME:ADDRess <*address*>
  MEMory:VME:STATe?
  MEMory:VME:SIZE?[<MIN or MAX>]

MEMory:VME:SIZE

MEMory:VME:SIZE [<*base*>]<*size*> sets the size in bytes of the external or shared command module memory.

Parameters

Parameter Name	Parameter Type	Range of Values	Default
<*base*>	Discrete	None, #H, #Q, or #B	Decimal
<*size*>	Numeric or Discrete	000000_{16} - $C00000_{16}$ or MIN or MAX	None None

Comments

- Address plus size must not exceed #HE00000.
- <*base*> specifies the numeric format as decimal, hexadecimal, octal, or binary. IEEE-488.2 specifies the following values for this parameter:

  Decimal = no parameter
  Hexadecimal = #H
  Octal = #Q
  Binary = #B

- **Related Commands:**
  MEMory:VME:ADDRess?
  MEMory:VME:STATe?
  MEMory:VME:SIZE?

- ***RST Condition:** #H000000.

MEMory:VME:SIZE?

MEMory:VME:SIZE? [<MIN | MAX>] queries for the current VME memory size. The optional parameter lets you query for the fixed maximum or minimum VME memory size.

Parameters

Parameter Name	Parameter Type	Range of Values	Default
<MIN | MAX>	Discrete	none, MIN or MAX	None

Comments

This command always returns the memory size in decimal format.

The size returned using MIN is always `0`.

The size returned using MAX is always `12582912`

Related Commands:
MEMory:VME:ADDRess? [<MIN or MAX>]
MEMory:VME:STATe?
MEMory:VME:SIZE <*size*>

MEMory:VME:STATe

MEMory:VME:STATe <*state*> enables/disables the use of VME memory for storage.

Parameters

Parameter Name	Parameter Type	Range of Values	Default
<*state*>	Boolean	0 or 1, OFF or ON	None

Comments **Related Commands:**
SOUR:DIG:TRAC:DEF <*name*><*size*>
SOUR:DIG:TRAC[:DATA] <*name*><*block_data*>
MEMory:VME:ADDress <*address*>
MEMory:VME:SIZE <*size*>

***RST Condition:** Set to OFF.

Example MEM:VME:STAT ON enables access to the VME memory.

MEMory:VME:STATe?

MEMory:VME:STATe? queries the state of the external memory flag.

Parameters None

Comments This command returns a `0` or a `1`, indicating external memory is OFF or ON.

- **Related Commands:**
  MEMory:VME:ADDRess?[<MIN or MAX>]
  MEMory:VME:SIZE?[<MIN or MAX>]

The [SOURce:] subsystem defines the command set for the Agilent E1458A Digital I/O Module output statements. It also defines the state and polarity of the control line (CTL), the polarity of the flag line (FLG), the handshaking mode, and handshake delay for both data input and output. The root command [SOURce:] is optional.

Subsystem Syntax

```
[SOURce:]
    DIGital
        :CONTrol n
            :POLarity <POS | NEG>
            :POLarity?
            [:VALue] <0 | 1 or ON | OFF>
            [:VALue]?
        :DATA n
            [:BYTE]
                :BIT m <0 | 1>
                :BIT m?
                :BIT m
                    :MONitor?
                :HANDshake
                    :DELay <time>
                    :DELay?
                    [:MODE] <NONE | LEADing | TRAiling
                                | PULSe | PARTial | STRobe>
                    [:MODE]?
                :MONitor?
                :POLarity <POS | NEG>
                :POLarity?
                :TRACe <name>
                [:VALue] [<base>]<value>
                [:VALue]?
            :WORD
                :BIT m <0 | 1>
                :BIT m?
                :BIT m
                    :MONitor?
                :HANDshake
                    :DELay <time>
                    :DELay?
                    [:MODE] <NONE | LEADing | TRAiling
                                | PULSe | PARTial | STRobe>
                    [:MODE]?
                :MONitor?
```

```
                    :POLarity <POS | NEG>
                    :POLarity?
                    :TRACe <name>
                    [:VALue] [<base>]<value>
                    [:VALue]?

[SOURCE:]
        DIGital
            DATAn
                :LWORd or LW32
                    :BITm <0 | 1>
                    :BITm?
                    :BITm
                        :MONitor?
                    :HANDshake
                        :DELay <time>
                        :DELay?
                        [:MODE] <NONE | LEADing | TRAiling
                                | PULSe | PARTial |STRobe>
                        [:MODE]?
                    :MONitor?
                    :POLarity <POS | NEG>
                    :POLarity?
                    :TRACe <name>
                    [:VALue] [<base>]<value>
                    [:VALue]?
                :LW64
                    :BITm <0 | 1>
                    :BITm?
                    :BITm
                        :MONitor?
                    :HANDshake
                        :DELay <time>
                        :DELay?
                        [:MODE] <NONE | LEADing | TRAiling
                                | PULSe | PARTial | STRobe>
                        [:MODE]?
                    :MONitor?
                    :POLarity <POS | NEG>
                    :POLarity?
                    :TRACe <name>
                    [:VALue] [<base>]<value>
                    [:VALue]?
```

```
[SOURCE:]
        DIGital
            :DATAn
                :LW96
                    :BITm <0 | 1>
                    :BITm?
                    :BITm
                        :MONitor?
                    :HANDshake
                        :DELay <time>
                        :DELay?
                        [:MODE] <NONE | LEADing | TRAiling
                                    | PULSe | PARTial | STRobe>
                        [:MODE]?
                    :MONitor?
                    :POLarity <POS | NEG>
                    :POLarity?
                    :TRACe <name>
                    [:VALue] [<base>]<value>
                    [:VALue]?
            :FLAGn
                :POLarity <POS | NEG>
                :POLarity?
            :HANDshaken
                :DELay <time>
                :DELay?
                [:MODE] <NONE | LEADing | TRAiling
                            | PULSe | PARTial | STRobe>
                [:MODE]?
            :IOn?
            :TRACe
                :CATalog?
                [:DATA] <name><block_data>
                [:DATA]? <name>
                :DEFine <name><size>[<fill>]
                :DEFine? <name>
                :DELete
                    [:NAME]<name>
                    :ALL
```

[SOURce:]DIGital:CONTrol*n*:POLarity

[SOURce:]DIGital:CONTrol*n*:POLarity <*polarity*> sets the CTL line voltage level for logical true in port *n* to either TTL High for POSitive polarity or TTL Low for NEGative polarity.

Parameters

Parameter Name	Parameter Type	Range of Values	Default
CONTrol*n*	Numeric	none, 0 through 11	0
<*polarity*>	Discrete	POSitive or NEGative	None

Comments

- Control lines are always accessed by their 8-bit port number
- :CONTrol*n* is the keyword used for commands relating to the control (CTL) line at port *n*. The port number *n* must be the last character of the keyword without spaces.
- If *n* is omitted, port 0 is used.
- The control line is used with the flag line to handshake data to and from peripherals.
- **Related Commands:**
  [SOURce:]DIGital:CONTrol*n*:POLarity?
  [SOURce:]DIGital:CONTrol*n*[:VALue]
  [SOURce:]DIGital:FLAG*n*:POLarity
  [SOURce:]DIGital:FLAG*n*:POLarity?
- ***RST Condition:** POLarity = POSitive

Example

DIG:CONT0:POL POS sets logical true to TTL High on port 0 control line.

[SOURce:]DIGital:CONTrol*n*:POLarity?

[SOURce:]DIGital:CONTrol*n*:POLarity? returns a three-character string, either POS or NEG, indicating the logical true condition of the control (CTL) line at port *n*.

Parameters

Parameter Name	Parameter Type	Range of Values	Default
CONTrol*n*	Numeric	none, 0 through 11	0

Example DIG:CONT0:POL? queries the state of the logical true condition on port 0.

- :CONTrol*n* is the keyword used for commands relating to the control (CTL) line at port *n*. The port number *n* must be the last character of the keyword without spaces.
- If *n* is omitted, port 0 is used.

[SOURce:]DIGital:CONTrol*n*[:VALue]

[SOURce:]DIGital:CONTrol*n*[:VALue] <***value***> sets or clears the control line on the selected port *n*.

Parameters

Parameter Name	Parameter Type	Range of Values	Default
CONTrol*n*	Numeric	none, 0 through 11	0
<*value*>	Boolean	0 or 1, OFF or ON	None

Comments

- This command is used to create custom handshakes when the HANDshake is set to NONE.
- :CONTrol*n* is the keyword used for commands relating to the control (CTL) line at port *n*. The port number *n* must be the last character of the keyword without spaces.
- The control line is used with the flag line to handshake data to and from peripherals.

- **Related Commands:**
  [SOURce:]DIGital:CONTrol*n*:POLarity
  [SOURce:]DIGital:CONTrol*n*:POLarity?
  [SOURce:]DIGital:CONTrol*n*:[:VALue]?
  [SOURce:]DIGital:FLAG*n*:POLarity
  [SOURce:]DIGital:FLAG*n*:POLarity?

- ***RST Condition:** Clears the control line; i.e., sets the control line to logical 0.

Example DIG:CONT2 1 sets the 8-bit port 2 control line true.

[SOURce:]DIGital:CONTrol*n*[:VALue]?

[SOURce:]DIGital:CONTrol*n*[:VALue]? reads the state of the control line on port *n* and returns a `0` or `1`, indicating the logical condition of the CTL line.

Parameters

Parameter Name	Parameter Type	Range of Values	Default
CONTrol*n*	Numeric	none, 0 through 11	0

Comments

- This command is used to create custom handshakes when the HANDshake is set to NONE.
- The condition of the CTL line returned by this command is the logical true value set by the DIG:CONT*n*:[:VALue]? command.

Example DIG:CONT2? returns the current state of the 8-bit port 2 control line.

[SOURce:]DIGital:DATA*n*[:type]:BIT*m*

[SOURce:]DIGital:DATA*n*[:BYTE]:BIT*m* <***value***> sets bit *m* on 8-bit port *n*.

[SOURce:]DIGital:DATA*n*:WORD:BIT*m* <***value***> sets bit *m* on 16-bit port *n*.

[SOURce:]DIGital:DATA*n*:LWORd:BIT*m* <***value***> sets bit *m* on 32-bit port *n*.

[SOURce:]DIGital:DATA*n*:LW64:BIT*m* <***value***> sets bit *m* on 64-bit port *n*.

[SOURce:]DIGital:DATA*n*:LW96:BIT*m* <***value***> sets bit *m* on 96-bit port *n*.

Parameters

Parameter Name	Parameter Type	Range of Values	Default
DATA*n*	Numeric	BYTE none, 0 through 11 WORD none, 0, 2, 4, 6, 8, or 10 LWORd none, 0, 4, or 8 LW64 none or 0 LW96 none or 0	0
BIT*m*	Numeric	BYTE 0 - 7 WORD 0 - 15 LWORd 0 - 31 LW64 0 - 63 LW96 0 - 95	None
<*value*>	Numeric	0 or 1	None

Comments

- :DATA*n* and :BIT*m* are the keywords used to write data to port *n* and bit *m*. The port number *n* and bit number *m* must be the last character of the keyword without spaces.
- For 16-bit operations using :WORD, *n* must be 0, 2, 4, 6, 8, or 10.
- For 32-bit operations using :LWORd, *n* must be 0, 4, or 8.
- For 64-bit or 96-bit operations using :LW64 or :LW96, *n* must be 0.
- The keyword LW32 may be used instead of LWORd.
- **Related Commands:**
  [SOURce:]DIGital:DATA*n*[:VALue]
  [SOURce:]DIGital:DATA*n*:POLarity
- ***RST Condition:** All ports are set for data input.

Example

DIG:DATA3:BIT4 1 sets bit 4 (the 5th bit) of port 3 to logical 1.

[SOURce:]DIGital:DATA*n*[:type]:BIT*m*?

[SOURce:]DIGital:DATA*n*[:BYTE]:BIT*m*? returns a `0` or `1` indicating the current programmed state of bit *m* on 8-bit port *n*.

[SOURce:]DIGital:DATA*n*:WORD:BIT*m*? returns a `0` or `1` indicating the current programmed state of bit *m* on 16-bit port *n*.

[SOURce:]DIGital:DATA*n*:LWORd:BIT*m*? returns a `0` or `1` indicating the current programmed state of bit *m* on 32-bit port *n*.

[SOURce:]DIGital:DATA*n*:LW64:BIT*m*? returns a `0` or `1` indicating the current programmed state of bit *m* on 64-bit port *n*.

[SOURce:]DIGital:DATA*n*:LW96:BIT*m*? returns a `0` or `1` indicating the current programmed state of bit *m* on 96-bit port *n*.

Parameters

Parameter Name	Parameter Type	Range of Values	Default
DATA*n*	Numeric	BYTE none, 0 through 11 WORD none, 0, 2, 4, 6, 8, or 10 LWORd none, 0, 4, or 8 LW64 none or 0 LW96 none or 0	0
BIT*m*	Numeric	BYTE 0 - 7 WORD 0 - 15 LWORd 0 - 31 LW64 0 - 63 LW96 0 - 95	None

Comments

- This command performs a readback of the data line register. the DIG:DATA*n*[:type]:BIT*m*:MON? command returns the actual condition of the data lines.
- The keyword LW32 may be used instead of LWORd.
- :DATA*n* and :BIT*m* are the keywords used to write data to port *n* and bit *m*. The port number *n* and bit number *m* must be the last character of the keyword without spaces.
- For 16-bit operations using :WORD, *n* must be 0, 2, 4, 6, 8, or 10.
- For 32-bit operations using :LWORd, *n* must be 0, 4, or 8.
- For 64-bit or 96-bit operations using :LW64 or :LW96, *n* must be 0.

- **Related Commands:**
  [SOURce:]DIGital:DATA*n*[:VALue]
  [SOURce:]DIGital:DATA*n*:POLarity

Example DIG:DATA3:BIT4:VAL? returns a `0` or `1` indicating the last programmed state of bit 4 on port 3.

[SOURce:]DIGital:DATA*n*[:type]:BIT*m*:MONitor?

[SOURce:]DIGital:DATA*n*[:BYTE]:BIT*m*:MONitor? returns a `0` or `1`, indicating the current state of the data line specified by bit *m* on 8-bit port *n*.

[SOURce:]DIGital:DATA*n*:WORD:BIT*m*:MONitor? returns a `0` or `1`, indicating the current state of the data line specified by bit *m* on 16-bit port *n*.

[SOURce:]DIGital:DATA*n*:LWORd:BIT*m*:MONitor? returns a `0` or `1`, indicating the current state of the data line specified by bit *m* on 32-bit port *n*.

[SOURce:]DIGital:DATA*n*:LW64:BIT*m*:MONitor? returns a `0` or `1`, indicating the current state of the data line specified by bit *m* on 64-bit port *n*.

[SOURce:]DIGital:DATA*n*:LW96:BIT*m*:MONitor? returns a `0` or `1`, indicating the current state of the data line specified by bit *m* on 96-bit port *n*.

Parameters

Parameter Name	Parameter Type	Range of Values	Default
DATA*n*	Numeric	BYTE none, 0 through 11 WORD none, 0, 2, 4, 6, 8, or 10 LWORd none, 0, 4, or 8 LW64 none or 0 LW96 none or 0	0
BIT*m*	Numeric	BYTE 0 - 7 WORD 0 - 15 LWORd 0 - 31 LW64 0-63 LW96 0-95	None

Comments

- This command performs a readback of the actual condition of the data line. The DIG:DATA*n*[:type]BIT*m*? command returns the programmed state of the data line.
- The value returned is not affected by the programmed data line polarity. A `1` returned always indicates a TTL High level on the data line.
- For 16-bit operations using :WORD, *n* must be 0, 2, 4, 6, 8, or 10.

- For 32-bit operations using :LWORd, *n* must be 0, 4, or 8.
- For 64-bit or 96-bit operations using :LW64 or :LW96, *n* must be 0.
- The keyword LW32 maybe used instead of LWORd.
- :DATA*n* and :BIT*m* are the keywords used to write data to port *n* and bit *m*. The port number *n* and bit number *m* must be the last character of the keyword without spaces.
- **Related Commands:**
  [SOURce:]DIGital:DATA*n*:BIT*n*[:VALue]?
  [SOURce:]DIGital:DATA*n*:POLarity

Example DIG:DATA3:BIT4:MON? returns either a `0` or a `1` indicating the current condition of bit 4 on port 3.

[SOURce:]DIGital:DATA*n*[:type]:HANDshake:DELay

[SOURce:]DIGital:DATA*n*[:BYTE]:HANDshake:DELay <*time*> sets the delay between data output and the control line for data output at 8-bit port *n*.

[SOURce:]DIGital:DATA*n*:WORD:HANDshake:DELay <*time*> sets the delay between data output and the control line for data output at 16-bit port *n*

[SOURce:]DIGital:DATA*n*:LWORd:HANDshake:DELay <*time*> sets the delay between data output and the control line for data output at the 32-bit port *n*.

[SOURce:]DIGital:DATA*n*:LW64:HANDshake:DELay <*time*> sets the delay between data output and the control line for data output at the 64-bit port.

[SOURce:]DIGital:DATA*n*:LW96:HANDshake:DELay <*time*> sets the delay between data output and the control line for data output at the 96-bit port.

Parameters

Parameter Name	Parameter Type	Range of Values	Default
DATA*n*	Numeric	BYTE none, 0 through 11 WORD none, 0, 2, 4, 6, 8, or 10 LWORd none, 0, 4, or 8 LW64 none or 0 LW96 none or 0	0
<*time*>	Numeric	2 µs to 15 µs 20 µs to 150 µs 200 µs to 1.5 ms 2ms to 15ms	None
	Discrete	MIN | MAX | DEF	

Comments

- This command is related to the handshake mode in use. Chapter 3 describes the handshake modes and timing.
- The delay time must be set to the same value on all ports used in a multiple-port operation.
- MAX sets a 15-ms delay. DEF sets 2-µs delay. MIN sets a delay of 0. Because of hardware ambiguity at this delay setting, the actual delay is not guaranteed. Note that MIN is illegal for PULSe or STRobe handshakes.
- This command sets the strobe pulse width for both input and output STRobe handshakes.
- DIGital:DATA*n*:HANDshake[:MODE] NONE command ignores any programmed delay time. For all other modes of handshaking, 2 µs is the minimum.
- Specific bands of delay settings are not allowed. These are:
  0 µs > <*time*> < *2* µs
  15 µs > <*time*> < 20 µs
  150 µs > <*time*> < 200 µs
  1.5 ms > <*time*> < 2.0 ms
  The controller uses a rounded-up value for <*time*> if these values are specified.
- The keyword LW32 may be used instead of LWORd.
- DIG:DATA*n*[:type]:HANDshake is the sequence used for commands relating to data handshaking at ports defined by *n*. The port number *n* must be the last character of the keyword without spaces.
- **Related Commands:**
  [SOURce:]DIGital:HANDshake*n*[:MODE]
  [SOURce:]DIGital:CONTrol*n*:POLarity
  [SOURce:]DIGital:CONTrol*n*[:VALue]
  [SOURce:]DIGital:FLAG*n*:POLarity

- ***RST Condition:** Delay is set to 2 µs.

Example DIG:HAND3:DEL .005 sets the delay between the data output and the assertion of the control line to true on 8-bit port 3 to 5 ms.

[SOURce:]DIGital:DATA*n*[:type]:HANDshake:DELay?

[SOURce:]DIGital:DATA*n*[:BYTE]:HANDshake:DELay? queries for the delay time between data output and the control line for data output at 8-bit port *n* and returns a decimal number between 0 and .015.

[SOURce:]DIGital:DATA*n*:WORD:HANDshake:DELay? queries for the delay time between data output and the control line for data output at 16-bit port *n* and returns a decimal number between 0 and .015.

[SOURce:]DIGital:DATA*n*:LWORd:HANDshake:DELay? queries for the delay between data output and the control line for data output at the 32-bit port *n* and returns a decimal number between 0 and .015.

[SOURce:]DIGital:DATA*n*:LW64:HANDshake:DELay? queries for the delay between data output and the control line for data output at the 64-bit port and returns a decimal number between 0 and .015.

[SOURce:]DIGital:DATA*n*:LW96:HANDshake:DELay? queries for the delay between data output and the control line for data output at the 96-bit port and returns a decimal number between 0 and .015.

Parameters

Parameter Name	Parameter Type	Range of Values	Default
DATA*n*	Numeric	BYTE none, 0 through 11 WORD none, 0, 2, 4, 6, 8, or 10 LWORd none, 0, 4, or 8 LW64 none or 0 LW96 none or 0	0
<MIN|MAX|DEF>	Discrete	None or MIN | MAX | DEF	None

Comments

- The keyword LW32 may be used instead of LWORd.
- MIN or DEF returns `0.000002`. MAX returns `0.015`.
- DIG:DATA*n*[:type]:HANDshake is the sequence used for commands relating to data handshaking at ports defined by *n*. The port number *n* must be the last character of the keyword without spaces.

[SOURce:]DIGital:DATA*n*[:type]:HANDshake[:MODE]

[SOURce:]DIGital:DATA*n*[:BYTE]:HANDshake[:MODE] <***mode***> selects the type of handshake and defines the timing relationship between the control (CTL) line, the flag (FLG) line, and when data is transferred in either direction between the Agilent E1458A Digital I/O Module and a peripheral on 8-bit port *n*.

[SOURce:]DIGital:DATA*n*:WORD:HANDshake[:MODE] <***mode***> selects the handshake mode used on the 16-bit port *n*.

[SOURce:]DIGital:DATA*n*:LWORd:HANDshake[:MODE] <***mode***> selects the handshake mode used on the 32-bit port *n*.

[SOURce:]DIGital:DATA*n*:LW64:HANDshake[:MODE] <***mode***> selects the handshake mode used on the 64-bit port.

[SOURce:]DIGital:DATA*n*:LW96:HANDshake[:MODE] <***mode***> selects the handshake mode used on the 96-bit port.

Parameters

Parameter Name	Parameter Type	Range of Values	Default
DATA*n*	Numeric	BYTE none, 0 through 11 WORD none, 0, 2, 4, 6, 8, or 10 LWORd none, 0, 4, or 8 LW64 none or 0 LW96 none or 0	0
<*mode*>	Discrete	NONE, LEADing, TRAiling, PULSe PARTial, or STRobe	NONE

Comments

- Handshake modes are described in Chapter 3.
- The handshake <*mode*> must be the same on all ports used in multiple-port operations.
- The keyword LW32 may be used instead of LWORd.
- DIG:DATA*n*[:type]HANDshake is the sequence used for commands relating to data handshaking at port *n*. The port number *n* must be the last character of the keyword without spaces.
- NONE deletes all automatic data handshaking between the Agilent E1458A Digital I/O Module and the peripheral. For custom handshaking, the control and the flag lines are controlled by the DIGital:CONTrol*n* and MEAS:DIGital:FLAG*n* commands.

- **Related Commands:**
  [SOURce:]DIGital:HANDshake*n*:[DELay]
  [SOURce:]DIGital:CONTrol*n*:POLarity
  [SOURce:]DIGital:CONTrol*n*[:VALue]
  [SOURce:]DIGital:FLAG*n*:POLarity

- ***RST Condition:** Mode is NONE on all ports.

Example DIG:DATA3:HAND LEAD sets the handshake mode to LEADing on 8-bit port 3.

[SOURce:]DIGital:DATA*n*[:type]:HANDshake[:MODE]?

[SOURce:]DIGital:DATA*n*[:BYTE]:HANDshake[:MODE]? returns a string indicating the type of handshake set on 8-bit port *n*.

[SOURce:]DIGital:DATA*n*:WORD:HANDshake[:MODE]? returns a string indicating the type of handshake set on the 16-bit port *n*.

[SOURce:]DIGital:DATA*n*:LWORd:HANDshake[:MODE]? returns a string indicating the type of handshake set on the 32-bit port *n*.

[SOURce:]DIGital:DATA*n*:LW64:HANDshake[:MODE]? returns a string indicating the type of handshake set on the 64-bit port.

[SOURce:]DIGital:DATA*n*:LW96:HANDshake[:MODE]? returns a string indicating the type of handshake set on the 96-bit port.

Parameters

Parameter Name	Parameter Type	Range of Values	Default
DATA*n*	Numeric	BYTE none, 0 through 11 WORD none, 0, 2, 4, 6, 8, or 10 LWORd none, 0, 4, or 8 LW64 none or 0 LW96 none or 0	0

Comments

- The handshake <*mode*> must be the same on all ports used in multiple-port operations.

- The keyword LW32 may be used instead of LWORd.

- This command will return one of the following strings:
  NONE
  LEAD
  TRA
  PULS
  PART
  STR

- :DATA*n*[:type]HANDshake? is the sequence used for commands relating to data handshaking at port *n*. The port number *n* must be the last character of the keyword without spaces.

- **Related Commands:**
  [SOURce:]DIGital:HANDshake*n*:[DELay]
  [SOURce:]DIGital:CONTrol*n*:POLarity
  [SOURce:]DIGital:CONTrol*n*[:VALue]
  [SOURce:]DIGital:FLAG*n*:POLarity

- ***RST Condition:** Mode is NONE on all ports.

Example DIG:DATA3:HAND? returns the handshake mode set on port 3.

[SOURce:]DIGital:DATA*n*[:type]:MONitor?

[SOURce:]DIGital:DATA*n*[:BYTE]:MONitor? returns a decimal value indicating the current state of the data lines on 8-bit port *n*. The value returned is in the range of 0 to 255.

[SOURce:]DIGital:DATA*n*:WORD:MONitor? returns a decimal value indicating the current state of the data lines on 16-bit port *n*. The value returned is in the range of -32768 to 32767.

[SOURce:]DIGital:DATA*n*:LWORd:MONitor? returns a decimal value indicating the current state of the data lines on 16-bit port *n*. The value returned is in the range of -2^{31} to $(2^{31} - 1)$.

[SOURce:]DIGital:DATA*n*:LW64:MONitor? returns two decimal values indicating the current state of the data lines on 64-bit port *n*. The values returned are each in the range of -2^{31} to $(2^{31} - 1)$.

[SOURce:]DIGital:DATA*n*:LW96:MONitor? returns three decimal values indicating the current state of the data lines on 96-bit port *n*. The values returned are each in the range of -2^{31} to $(2^{31} - 1)$.

Parameters

Parameter Name	Parameter Type	Range of Values	Default
DATA*n*	Numeric	BYTE none, 0 through 11 WORD none, 0, 2, 4, 6, 8, or 10 LWORd none, 0, 4, or 8 LW64 none or 0 LW96 none or 0	0

Comments

- This command performs a readback of the actual condition of the selected data port. The DIG:DATA*n*[:type][:VALue]? command returns the programmed state of the data port.
- The value returned is not affected by the programmed data line polarity. The returned value indicates the actual TTL state of the data lines.
- DIG:DATA*n*[:type]:MON? is the sequence used for commands relating to data lines at port *n*. The port number *n* must be the last character of the keyword without spaces.
- The keyword LW32 may be used instead of LWORd**.**
- **Related Commands:**
  [SOURce:]DIGital:DATA*n*[:type]:POLarity
- ***RST Condition:** Mode is NONE on all ports.

Example

DIG:DATA3:MON? returns a number representing the state of the data lines on 8-bit port 3.

[SOURce:]DIGital:DATA*n*[:type]:POLarity

[SOURce:]DIGital:DATA*n*[:BYTE]:POLarity <*polarity*> sets the data line voltage level for logical true in port *n* to either TTL High for POSitive polarity or TTL Low for NEGative polarity.

[SOURce:]DIGital:DATA*n*:WORD:POLarity <*polarity*> sets the data line voltage level for logical true in the 16-bit port *n* to either TTL High for POSitive polarity or TTL Low for NEGative polarity.

[SOURce:]DIGital:DATA*n*:LWORd:POLarity <*polarity*> sets the data line voltage level for logical true in the 32-bit port *n* to either TTL High for POSitive polarity or TTL Low for NEGative polarity.

[SOURce:]DIGital:DATA*n*:LW64:POLarity <*polarity*> sets the data line voltage level for logical true in the 64-bit port *n* to either TTL High for POSitive polarity or TTL Low for NEGative polarity.

[SOURce:]DIGital:DATA*n*:LW96:POLarity <*polarity*> sets the data line voltage level for logical true for all ports to either TTL High for POSitive polarity or TTL Low for NEGative polarity.

Parameters

Parameter Name	Parameter Type	Range of Values	Default
DATA*n*	Numeric	BYTE none, 0 through 11 WORD none, 0, 2, 4, 6, 8, or 10 LWORd none, 0, 4, or 8 LW64 none or 0 LW96 none or 0	0
<*polarity*>	Discrete	POSitive or NEGative	None

Comments

- :DATA*n* is the keyword used for commands relating to the data lines at port *n*. The port number *n* must be the last character of the keyword without spaces.
- **Related Commands:**
  [SOURce:]DIGital:DATA*n*:POLarity?
  [SOURce:]DIGital:DATA*n*[:VALue]
  [SOURce:]DIGital:DATA*n*:BIT*m*
- ***RST Condition:** POLarity = POSitive

Example

DIG:DATA0:POL POS sets logical true to TTL High on 8-bit port 0 data lines.

[SOURce:]DIGital:DATA*n*[:type]:POLarity?

[SOURce:]DIGital:DATA*n*[:BYTE]:POLarity? returns a string, either POS or NEG, indicating the logical true condition of the data lines of 8-bit port *n*.

[SOURce:]DIGital:DATA*n*:WORD:POLarity? returns a string, either POS or NEG, indicating the logical true condition of the data lines of 16-bit port *n*.

[SOURce:]DIGital:DATA*n*:LWORd:POLarity? returns a string, either POS or NEG, indicating the logical true condition of the data lines of the 32-bit port *n*.

[SOURce:]DIGital:DATA*n*:LW64:POLarity? returns a string, either POS or NEG, indicating the logical true condition of the data lines of the 64-bit port.

[SOURce:]DIGital:DATA*n*:LW96:POLarity? returns a string, either POS or NEG, indicating the logical true condition of the data lines of the 96-bit port.

Parameters

Parameter Name	Parameter Type	Range of Values	Default
DATA*n*	Numeric	BYTE none, 0 through 11 WORD none, 0, 2, 4, 6, 8, or 10 LWORd none, 0, 4, or 8 LW64 none or 0 LW96 none or 0	0

Example DIG:DATA0:POL? returns the state of the logical true condition on port 0 as either POS or NEG.

[SOURce:]DIGital:DATA*n*[:type]:TRACe

[SOURce:]DIGital:DATA*n*[:BYTE]:TRACe <*name*> writes the named block of data to 8-bit port *n* whenever the port is ready to start a new handshake.

[SOURce:]DIGital:DATA*n*:WORD:TRACe <*name*> writes the named block of data to 16-bit port *n* whenever the port is ready start a new handshake.

[SOURce:]DIGital:DATA*n*:LWORd:TRACe <*name*> writes the named block of data to the 32-bit port *n* whenever the port is ready to start a new handshake.

[SOURce:]DIGital:DATA*n*:LW64:TRACe <*name*> writes the named block of data to the 64-bit port whenever the port is ready to start a new handshake.

[SOURce:]DIGital:DATA*n*:LW96:TRACe <*name*> writes the named block of data to the 96-bit port whenever the port is ready to start a new handshake.

Parameters

Parameter Name	Parameter Type	Range of Values	Default
DATA*n*	Numeric	BYTE none, 0 through 11 WORD none, 0, 2, 4, 6, 8, or 10 LWORd none, 0, 4, or 8 LW64 none or 0 LW96 none or 0	0
<*name*>	String	Name of user memory block (maximum 12 characters)	None

Comments

- The keyword LW32 may be used instead of LWORd.
- :DATA*n* and :TRACe are the keywords used to write data to port *n* from block <*name*>. The port number *n* must be the last character of the keyword without spaces.
- **Related Commands:**
  [SOURce:]DIGital:DATA*n*[:VALue]
  [SOURce:]DIGital:DATA*n*:POLarity
- ***RST Condition:** All ports are set for data input.

Example

DIG:DATA2:TRAC:WORD first_block writes data from the user memory block *first_block* to 16-bit port 2.

[SOURce:]DIGital:DATA*n*[:type][:VALue]

[SOURce:]DIGital:DATA*n*[:BYTE][:VALue] [<*base*>]<*value*> writes data to 8-bit port *n*. Values can be binary, octal, decimal, or hexadecimal.

[SOURce:]DIGital:DATA*n*:WORD[:VALue] [<*base*>]<*value*> writes data to 16-bit port *n*. Values can be binary, octal, decimal, or hexadecimal.

[SOURce:]DIGital:DATA*n*:LWORd[:VALue] [<base>]<*value*> writes data to the 32-bit port *n*. Values can be binary, octal, decimal, or hexadecimal.

[SOURce:]DIGital:DATA*n*:LW64[:VALue] [<*base*>]<*value*> writes data to the 64-bit port. Values can be binary, octal, decimal, or hexadecimal.

[SOURce:]DIGital:DATA*n*:LW96[:VALue] [<*base*>]<*value*> writes data to the 96-bit port. Values can be binary, octal, decimal, or hexadecimal.

Parameters

Parameter Name	Parameter Type	Range of Values	Default
DATA*n*	Numeric	BYTE none, 0 through 11 WORD none, 0, 2, 4, 6, 8, or 10 LWORd none, 0, 4, or 8 LW64 none or 0 LW96 none or 0	0
<*base*>	Discrete	None, #H, #Q, or #B	Decimal
<*value*>	Numeric	BYTE -2^7 to (2^8-1) WORD -2^{15} to $(2^{16}-1)$ LWORd -2^{31} to $(2^{31}-1)$ LW64 -2^{31} to $(2^{31}-1)$ LW96 -2^{31} to $(2^{31}-1)$	None

Comments

- The keyword LW32 may be used instead of LWORd**.**
- <*base*> specifies the numeric format as decimal, hexadecimal, octal, or binary. IEEE-488.2 specifies the following values for this parameter:

  Decimal = no parameter
  Hexadecimal = #H
  Octal = #Q
  Binary = #B

- :DATA*n* is the keyword used for commands relating to data output at port *n*. The port number *n* must be the last character of the keyword without spaces.

- **Related Commands:**
  [SOURce:]DIGital:DATA*n*:BIT*m*
  [SOURce:]DIGital:DATA*n*:POLarity

- ***RST Condition:** All ports are set for data input.

Examples DIG:DATA3 27 writes the binary equivalent of the decimal number 27 (00011011) to 8-bit port 3.

DIG:DATA3 #B00011011 writes the same byte of data as in the example above to port 3, but uses binary format.

[SOURce:]DIGital:DATA*n*[:type][:VALue]?

[SOURce:]DIGital:DATA*n*[:BYTE][:VALue]? returns the programmed state of 8-bit port *n* as a decimal number between 0 and 255.

[SOURce:]DIGital:DATA*n*:WORD[:VALue]? returns the programmed state of 16-bit port *n* as a decimal number between -32768 and 32767.

[SOURce:]DIGital:DATA*n*:LWORd[:VALue]? returns the programmed state of 32-bit port *n* as a decimal number between -2^{31} and $(2^{31} - 1)$.

[SOURce:]DIGital:DATA*n*:LW64[:VALue]? returns the programmed state of the 64-bit port as two decimal values, each in the range of -2^{31} to $(2^{31} - 1)$.

[SOURce:]DIGital:DATA*n*:LW96[:VALue]? returns the programmed state the 96-bit port as three decimal numbers, each in the range of -2^{31} to $(2^{31} - 1)$.

Parameters

Parameter Name	Parameter Type	Range of Values	Default
DATA*n*	Numeric	BYTE none, 0 through 11 WORD none, 0, 2, 4, 6, 8, or 10 LWORd none, 0, 4, or 8 LW64 none or 0 LW96 none or 0	0

Comments

- This command returns the programmed state of the data lines. The DIG:DATA*n*[:type]:MON? returns the actual state of the data lines.
- :DATA*n* is the keyword used for commands relating to data output at port *n*. The port number *n* must be the last character of the keyword without spaces.
- **Related Commands:**
  [SOURce:]DIGital:DATA*n*:BIT*m*
  [SOURce:]DIGital:DATA*n*:POLarity
- ***RST Condition:** All ports are set for data input.

Example

DIG:DATA3? returns the decimal equivalent of the data lines on 8-bit port 3.

[SOURce:]DIGital:FLAG*n*:POLarity

[SOURce:]DIGital:FLAG*n*:POLarity <POS or **NEG>** sets the voltage level for logical true to either TTL High, POSitive, or TTL Low, NEGative, on the FLG handshake line.

Parameters

Parameter Name	Parameter Type	Range of Values	Default
FLAG*n*	Numeric	none, 0 through 11	0
<*polarity*>	Discrete	POSitive or NEGative	None

Comments

- :FLAG*n* is the keyword used for commands relating to the flag line at port *n*. The port number *n* must be the last character of the keyword without spaces.
- **Related Commands:**
  [SOURce:]DIGital:FLAG*n*:POLarity?
  [SOURce:]DIGital:CONTrol*n*:POLarity
  [SOURce:]DIGital:CONTrol*n*:POLarity?
- ***RST Condition:** POLarity = POSitive

Example

DIG:FLAG0:POL POS sets logical true to TTL High on the port 0 flag line.

[SOURce:]DIGital:FLAG*n*:POLarity?

[SOURce:]DIGital:FLAG*n*:POLarity? returns a string, either POS or NEG, indicating the logical true condition of the flag (FLG) line.

Parameters

Parameter Name	Parameter Type	Range of Values	Default
FLAG*n*	Numeric	none, 0 through 11	0

Example

SOURCE:DIGITAL:FLAG0:POLarity? uses long commands to query the state of the logical true condition on port 0.

DIG:FLAG0:POL? performs the same function as the example above with short commands.

[SOURce:]DIGital:HANDshake*n*:DELay

[SOURce:]DIGital:HANDshake*n*:DELay <*time*> sets the time between data valid and the assertion of the control line to TRUE for port *n*. This form of the command operates on 8-bit ports only.

Parameters

Parameter Name	Parameter Type	Range of Values	Default
HANDshake*n*	Numeric	0 through 11	None
<*time*>	Numeric	2 µs to 15 µs 20 µs to 150 µs 200 µs to 1.5 ms 2ms to 15ms	None
	Discrete	MIN | MAX | DEF	

- :HANDshake*n* is the keyword used for commands relating to data handshaking at port *n*. The port number *n* must be the last character of the keyword without spaces.
- The delay time must be set to the same value on all ports used in a multiple-port operation.
- MAX sets a 15-ms delay. DEF sets 2-µs delay. MIN sets a delay of 0. Because of hardware ambiguity at this delay setting, the actual delay is not guaranteed. Note that MIN is illegal for PULSe or STRobe handshakes.
- This command sets the strobe pulse width for both input and output STRobe handshakes.
- DIGital:HANDshake*n* NONE command ignores any programmed delay time. For all other modes of handshaking, 2 µs is the minimum.
- Specific bands of delay settings are not allowed. These are:
  0 µs > <*time*> < 2 µs
  15 µs > <*time*> < 20 µs
  150 µs > <*time*> < 200 µs
  1.5 ms > <*time*> < 2.0 ms
  The controller uses a rounded-up value for <*time*> if these values are specified.
- **Related Commands:**
  [SOURce:]DIGital:HANDshake*n*[:MODE]
  [SOURce:]DIGital:CONTrol*n*:POLarity
  [SOURce:]DIGital:CONTrol*n*[:VALue]
  [SOURce:]DIGital:FLAG*n*:POLarity

- ***RST Condition:** Delay is set to 2 µs.

Example DIG:HAND3:DEL .005 sets the delay between the data output and the assertion of the control line to true on 8-bit port 3 to 5 ms.

[SOURce:]DIGital:HANDshake*n*:DELay?

[SOURce:]DIGital:HANDshake*n*:DELay? queries for the time between data valid and the assertion of the control line to TRUE. This form of the command operates only on 8-bit ports and returns a decimal value between 0 and 0.015.

Parameters

Parameter Name	Parameter Type	Range of Values	Default
HANDshake*n*	Numeric	None, 0 through 11	0
	Discrete	MIN | MAX | DEF	

Comments MIN or DEF returns `0.000002`. MAX returns `0.015`.

The delay time must be set to the same value on all ports used in a multiple-port operation.

:HANDshake*n* is the keyword used for commands relating to data handshaking at 8-bit port *n*. The port number *n* must be the last character of the keyword without spaces.

Example DIG:HAND0:DEL? queries the delay time between data valid and the assertion of the control line to TRUE on 8-bit port 0.

[SOURce:]DIGital:HANDshake*n*[:MODE]

[SOURce:]DIGital:HANDshake*n*[:MODE] <*mode*> selects the type of handshake mode and defines the timing relationship between the control (CTL) line, the flag (FLG) line, and when data is transferred in either direction between the Agilent E1458A Digital I/O Module and a peripheral on 8-bit port *n*. This form of the HANDshake command operates only on 8-bit ports.

Parameters

Parameter Name	Parameter Type	Range of Values	Default
HANDshake*n*	Numeric	None, 0 through 11	0
<*mode*>	Discrete	NONE, LEADing, TRAiling, PULSe PARTial, or STRobe	NONE

Comments

:HANDshake*n* is the keyword used for commands relating to data handshaking at port *n*. The 8-bit port number *n* must be the last character of the keyword without spaces.

NONE deletes all automatic data handshaking between the digital I/O module and the peripheral. For custom handshaking, the control and the flag lines are controlled by the DIGital:CONTrol*n* and DIGital:FLAG*n* commands.

Related Commands:
[SOURce:]DIGital:HANDshake*n*:DELay
[SOURce:]DIGital:CONTrol*n*:POLarity
[SOURce:]DIGital:CONTrol*n*[:VALue]
[SOURce:]DIGital:FLAG*n*:POLarity

***RST Condition:** Mode is NONE on all ports.

Example

DIG:HAND3 LEAD sets the handshake mode to LEADing on 8-bit port 3.

[SOURce:]DIGital:HANDshake*n*[:MODE]?

[SOURce:]DIGital:HANDshake*n*[:MODE]? returns a string indicating the current handshake mode of 8-bit port *n*. This form of the HANDshake command operates only on 8-bit ports.

Parameters

Parameter Name	Parameter Type	Range of Values	Default
HANDshake*n*	Numeric	None, 0 through 11	0

Comments

- This command will return one of the following strings:
  - NONE
  - LEAD
  - TRA
  - PULS
  - PART
  - STR
- :HANDshake*n* is the keyword used for commands relating to data handshaking at port *n*. The port number *n* must be the last character of the keyword without spaces.

[SOURce:]DIGital:IO*n*?

[SOURce:]DIGital:IO*n*? returns a `0` or `1` indicating the current condition of the I/$\overline{O}$ line on port *n*.

Parameters

Parameter Name	Parameter Type	Range of Values	Default
IO*n*	Numeric	None, 0 through 11	0

Comments

- The I/$\overline{O}$ line polarity is fixed and is as follows:
  - When digital I/O module is programmed to output data, the I/$\overline{O}$ line is set low.
  - When digital I/O module is programmed to input data, the I/$\overline{O}$ line is set high.
- :IO*n* is the keyword used for commands relating to the I/$\overline{O}$ line at port *n*. The port number *n* must be the last character of the keyword without spaces.

[SOURce:]DIGital:TRACe:CATalog?

[SOURce:]DIGital:TRACe:CATalog? lists the currently available data blocks.

Parameters None

Comments

- This command catalogs all blocks in VME memory and all blocks in the mainframe system memory.
- The command returns a string.

Example DIG:TRAC:CAT? would return this string if both alpha and beta had been previously defined: "alpha","beta".

[SOURce:]DIGital:TRACe[:DATA]

[SOURce:]DIGital:TRACe[:DATA] ***<name>,<block_data>*** writes a block of data to a previously defined user memory block.

Parameters

Parameter Name	Parameter Type	Range of Values	Default
<name>	String	Name of user memory block (maximum 12 characters)	None
<block_data>	Numeric/String	Numeric header and ASCII block data	None

Comments

- *<name>* must have been previously defined by a DIGital:TRACe:DEFine command.
- The maximum length for *<name>* is 12 characters.
- *<block_data>* is of the form *<#digits><length><block>* where:
  <#digits> tells how many digits are used to define *<length>*;
  <length> tells how many bytes are to be transferred in *<block>*;
  <block> contains the actual data to transfer.

Example DIG:TRAC:DATA first_block, #210ABCDEFGHIJ sends the data "ABCDEFGHIJ" to the user memory block *first_block*. Since the ASCII character A has a decimal value of 65, the equivalent of 65 is stored in the first byte of first_block (and so on).

[SOURce:]DIGital:TRACe[:DATA]?

[SOURce:]DIGital:TRACe[:DATA]? <*name*> reads a block of data from a previously defined user memory block.

Parameters

Parameter Name	Parameter Type	Range of Values	Default
<*name*>	String	Name of user memory block (maximum 12 characters)	None

Comments

- <*name*> must have been previously defined by a DIGital:TRACe:DEFine command.
- The maximum length for <*name*> is 12 characters.

Example

DIG:TRACe? first_block reads data from a block named *first_block*. If the previous command example is sent, this command will return the string `#210ABCDEFGHIJ`.

[SOURce:]DIGital:TRACe:DEFine

[SOURce:]DIGital:TRACe:DEFine <*name*>, <*size*>,[<*fill*>] defines a block of data as a user memory block, names the block for future reference, and fills the block with the last parameter. If the last parameter is absent, the block is filled with zeros.

Parameters

Parameter Name	Parameter Type	Range of Values	Default
<*name*>	String	Name of user memory block (maximum 12 characters)	None
<*size*>	Numeric	Up to 12 Mbytes (depending on memory installed)	None
<*fill*>	Numeric	0 - 255	0

Comments

- The firmware can handle blocks with a total memory space of up to 12 Mbytes of memory space. The actual amount available depends on the memory installed.
- If the MEMory:VME:STATe ON command has been used, this command will create blocks in the external add-on memory. If the MEMory:VME:STATe OFF command has been used, this command will create blocks in the system memory.

Example

DIG:TRAC:DEF first_block, 256 defines a 256-byte user memory block named *first_block* and fills each byte with a zero.

[SOURce:]DIGital:TRACe:DEFine?

[SOURce:]DIGital:TRACe:DEFine? <*name*> returns the size of a previously defined user memory block in bytes. The command returns a decimal number in the range of 0 to 12,582,912.

Parameters

Parameter Name	Parameter Type	Range of Values	Default
<*name*>	String	Name of user memory block (maximum 12 characters)	None

Comments

- <*name*> must have been previously defined by a DIGital:TRACe:DEFine command. The maximum length for <*name*> is 12 characters.

[SOURce:]DIGital:TRACe:DELete[:NAME]

[SOURce:]DIGital:TRACe:DELete <*name*> deletes a previously defined user memory data block.

Parameters

Parameter Name	Parameter Type	Range of Values	Default
<*name*>	String	Name of user memory block (maximum 12 characters)	None

Comments

- <*name*> must have been previously defined by a DIGital:TRACe:DEFine command. The maximum length for <*name*> is 12 characters.

Example

DIG:TRACe:DEL first_block deletes a user memory block named *first_block*.

[SOURce:]DIGital:TRACe:DELete:ALL

[SOURce:]DIGital:TRACe:DELete:ALL deletes all previously defined user memory data blocks.

Parameters

None

The STATus subsystem controls the SCPI-defined Operation and Questionable Signal status registers and the Standard Event register. Each is comprised of a condition register, an event register, an enable mask, and transition filters.

Each status register works as follows: when a condition occurs, the appropriate bit in the condition register is set or cleared. If the corresponding transition filter is enabled for that bit, the same bit is set in the associated event register. The contents of the event register and the enable mask are logically "ANDed" bit-for-bit; if any bit of the result is set, the summary bit for that register is set in the status byte. The status byte summary bit for the Operation status register is bit 7, for the Questionable Signal status register it is bit 3, and for the Standard Event registers it is bit 5.

This subsystem is provided for compatibility. The Agilent E1458A Digital I/O Module does not use the Operation status or Questionable status registers.

Subsystem Syntax

```
STATus
    :OPERation
        :CONDition?
        :ENABle
        :ENABle?
        [:EVENt]?
    :PRESet
    :QUEStionable
        :CONDition?
        :ENABle
        :ENABle?
        [:EVENt]?
```

STATus:OPERation:CONDition?

STATus:OPERation:CONDition? returns the contents of the Operation Status condition register. Reading the register does not affect its contents.

STATus:OPERation:ENABle

STATus:OPERation:ENABle <*mask*> specifies which bits of the associated event register are included in its summary bit. The summary bit is the bit-for-bit logical AND of the event register and the unmasked bit(s).

Parameters

Parameter Name	Parameter Type	Range of Values	Default
<*mask*>	Numeric or non-decimal numeric	0 through 32767	None

The non-decimal numeric forms are the # H, # Q, or # B formats specified by IEEE-488.2.

STATus:OPERation:ENABle?

STATus:OPERation:ENABle? returns the mask set for the Operation status register.

STATus:OPERation[:EVENt]?

STATus:OPERation[:EVENt]? returns the contents of the Operation Event status register. Reading the register clears all bits in the register.

STATus:PRESet

STATus:PRESet clears both the Operation status enable and Questionable status enable registers.

STATus:QUEStionable:CONDition?

STATus:QUEStionable:CONDition? returns the contents of the Questionable Status condition register. Reading the register does not affect its contents.

STATus:QUEStionable:ENABle

STATus:QUEStionable:ENABle <*mask*> specifies which bits of the associated event register are included in its summary bit. The summary bit is the bit-for-bit logical AND of the event register and the unmasked bit(s).

Parameters

Parameter Name	Parameter Type	Range of Values	Default
<*mask*>	Numeric or non-decimal numeric	0 through 32767	None

The non-decimal numeric forms are the # H, # Q, or # B formats specified by IEEE-488.2.

STATus:QUEStionable:ENABle?

STATus:QUEStionable:ENABle? returns the mask set for the Questionable status register.

STATus:QUEStionable[:EVENt]?

STATus:QUEStionable[:EVENt]? returns the contents of the Questionable status event register. Reading the register clears all bits in the register.

The SYSTem subsystem returns information about the module.

Subsystem Syntax

```
SYSTem
    :CDEScription? <module>
    :CTYPe? <module>
    :ERRor?
    :VERsion?
```

SYSTem:CDEScription?

SYSTem:CDEScription? <*module*> returns the module description.

Parameters

Parameter Name	Parameter Type	Range of Values	Default
<*module*>	Numeric	1	None

Comments

- The <*module*> is the instrument number. Because each Agilent E1458A Digital I/O Module is a single instrument, <*module*> is always 1.
- The command returns the following string:
  `96-Channel Digital I/O`

SYSTem:CTYPe?

SYSTem:CTYPe? <***module***> returns the module number and manufacturer.

Parameters

Parameter Name	Parameter Type	Range of Values	Default
<*module*>	Numeric	1	None

Comments

- The <*module*> is the instrument number. Because each digital I/O module is a single instrument, <*module*> is always 1.
- The command returns the following string (revision number may vary and serial number is always set to 0):
  `HEWLETT-PACKARD,E1458A,0,A.06.00`

SYSTem:ERRor?

SYSTem:ERRor? queries the error register for the error value and returns a string error message to identify the error type. The errors are held in an error buffer and read in a first-in-first-out manner by this command.

- Returns the error number and error string. If no errors are in the error buffer, returns: `+0`,"No error".
- **Related Commands:** *ERR
- ***RST Condition:** NONE

Example

SYST:ERR? queries the mainframe for errors.

SYSTem:VERSion?

SYSTem:VERsion? returns the SCPI version to which this instrument complies.

Comments

- Returns a decimal value in the form: YYYY.R; where YYYY is the year, and R is the revision number within that year.

IEEE 488.2 Common Commands

The following table lists the IEEE 488.2 Common (*) commands that can be executed by the Agilent E1458A 96-Channel Digital I/O Module. For more information on Common commands, refer to *ANSI/IEEE Standard 488.2-1987*.

Note These commands apply to many instruments and are not documented in detail here. See *ANSI/IEEE Standard 488.2-1987* for more information.

Command	Title	Description
*IDN?	Identification Query	Returns identification string of the digital I/O module.
*RST	Reset	Sets all ports to input mode, handshake NONE, and Polarity POS.
*TST?	Selt-Test Query	Returns: 0 = All tests pass 1 = Read register 00h failed 2 = Read register 02h failed 200+n = Port interrupt test at port n failed.
*OPC	Operation Complete	Sets the Request for OPC flag when all pending operations have been completed. Also sets the OPC bit in the Standard Event register
*OPC?	Operation Complete Query	Returns a 1 to the output queue when all pending operations are complete.
*WAI	Wait to Continue	Halts execution of commands and queries until the No Operation Pending message is true.
*CLS	Clear Status Registers	Clears all Event registers, the Request for OPC flag, and all queues (except the output queue).
*ESE <*mask*>	Event Status Enable	Sets the bits in the Event Status Enable register.
*ESE?	Event Status Enable Query	Queries the Event Status Enable register.
*ESR?	Event Status Register Query	Queries and clears contents of the Standard Event Status register.
*SRE <*mask*>	Service Request Enable	Sets the Service Request Enable register bits, and corresponding Serial Poll Status Byte register bits, to generate a service request.
*SRE?	Service Request Enable Query	Queries the contents of the Service Request Enable register.
*STB?	Read Status Byte Query	Queries the Status Byte register.
*TRG	Trigger	
*RCL <*n*>	Recall Instrument State	Recalls stored module configuration in the memory location set by *n*.
*SAV <*n*>	Store Instrument State	Stores the module configuration in the memory location set by *n*.
*EMC <*n*>	Enable Macro	Enables execution of macro *n*.
*EMC? <*n*>	Enable Macro Query	Queries execution state of macro *n*.
*RMC	Remove Macros	Removes all macros.
LMC?	List Macros	Lists macros by name.
*DMC	Define Macro	Defines a macro.
*GMC?	Menu Query	Gets results of menu query.
*PMC	Purge Macros	Purges all system macros.

Command Quick Reference

Command	Description
DISPlay:MONitor:PORT <*port* | AUTO>	Turns the monitor mode on for the specified port.
MONitor:PORT? [<MAX | MIN | DEF>]	Returns the monitored port number.
MONitor[:STATe] <OFF | ON>	Turns the monitor mode of the display on.
MONitor[:STATe]?	Returns the state of the monitor mode.
MEASure:DIGital:DATA*n* [:type]:BIT*m*?	Reads selected bit on selected port after completion of handshake.
DIGital:DATA*n* [:type]:TRACe <*name*>	Reads selected port after completion of handshake and stores block.
DIGital:DATA*n* [:type][:VALue]?	Reads selected port after completion of handshake. Assumes decimal format of input data.
DIGital:FLAG*n*?	Reads FLAG line on selected port. Returns `0` or `1`. Used to implement custom handshakes.
MEMory:DELete:MACRo <*name*>	Deletes a macro.
VME:ADDRess [<*base*>]<*address*>	Sets the address for add-on VME system memory.
VME:ADDRess? [MIN | MAX]	Returns the current add-on VME memory address.
VME:SIZE [<*base*>]<*size*>	Sets the size of the add-on VME memory.
VME:SIZE? [MIN | MAX]	Returns the current size of the add-on VME memory.
VME:STATe <*state*>	Sets the state (ON or OFF) of the assigned VME memory. When this is OFF, all memory commands refer to the base system memory.
VME:STATe?	Returns the current state (`0` or `1`) of the add-on VME memory.
[SOURce:]DIGital:CONTrol*n*:POLarity <POS | NEG>	Sets logical true level of Control line on selected port.
DIGital:CONTrol*n*:POLarity?	Returns current logical true polarity of selected port.
DIGital:CONTrol*n* [:VALue] <0 | 1 or ON | OFF>	Sets or clears control line on selected port. Command used to create custom handshakes when HANDshake is set to NONE.
DIGital:CONTrol*n* [:VALue]?	Returns the current state of the control line on selected port.
DIGital:DATA*n* [:type]:BIT*m* <0 | 1>	Sets selected bit on selected port.
DIGital:DATA*n* [:type]:BIT*m*?	Returns the programmed state of the selected bit on the selected port.
DIGital:DATA*n* [:type]:BIT*m* :MONitor?	Returns the state of the data line associated with the selected bit on the selected port.
DIGital:DATA*n* [:type]:HANDshake :DELay <*time*>	Sets delay between data output and assertion of control line for data output. Also sets strobe pulse for both output and input STROBE handshake.
DIGital:DATA*n* [:type]:HANDshake:DELay? [MIN | MAX | DEF]	Returns the time between data valid and assertion of control line to TRUE.

Command	**Description**
[SOURce:]DIGital:DATA*n*[:type]:HANDshake[:MODE] <NONE | LEADing | TRAiling | PULSe | PARTial | STRobe>	Selects type of handshake to transfer data between selected port and peripheral. Handshakes are initiated by execution of DIG:DATA*n* or MEAS:DATA*n*? commands.
DIGital:DATA*n*[:type]:HANDshake[:MODE]?	Returns the current handshake mode set on the selected port.
DIGital:DATA*n*[:type]:MONitor?	Returns a decimal value showing the state of the data lines associated with the selected port.
DIGital:DATA*n*[:type]:POLarity <POS | NEG>	Sets logical true level of the data lines on the selected port.
DIGital:DATA*n*[:type]:POLarity?	Returns the logical true level set for the data lines on the selected port.
DIGital:DATA*n*[:type]:TRACe <*name*>	Writes the named block of data to the selected port.
DIGital:DATA*n*[:type][:VALue][<*base*>]<*value*>	Writes the value, in the specified base, to the selected port.
DIGital:DATA*n*[:type][:VALue]?	Returns a decimal value indicating the programmed state of the data lines on the selected port.
DIGital:FLAG*n*:POLarity <POS | NEG>	Sets logical true level of the flag line on the selected port.
DIGital:FLAG*n*:POLarity?	Returns the logical true level set for the flag line on the selected port.
DIGital:HANDshake*n*:DELay <*time*>	Sets delay between data output and assertion of control line for data output on the selected 8-bit port. Also sets strobe pulse for both output and input STROBE handshake.
DIGital:HANDshake*n*:DELay? [MIN | MAX | DEF]	Returns the time between data valid and assertion of control line to TRUE on the selected 8-bit port.
DIGital:HANDshake*n*[:MODE] <NONE | LEAD | TRA | PULS | PART | STR>	Selects type of handshake to transfer data between selected 8-bit port and peripheral. Handshakes are initiated by execution of DIG:DATA*n* or MEAS:DATA*n*? commands.
DIGital:HANDshake*n*[:MODE]?	Returns the current handshake mode set on the selected 8-bit port.
DIGital:IO*n*?	Returns the current state of the I/$\overline{O}$ control line on the selected port.
DIGital:TRAce:CATalog?	Returns the currently defined memory blocks.
DIGital:TRAce[:DATA] <*name*><*block_data*>	Writes a block of data to name.
DIGital:TRAce[:DATA]? <*name*>	Reads a block of data from name.
DIGital:TRAce:DEFine <*name*><*size*>[<*fill*>]	Defines name, size, and initial fill for a memory block.
DIGital:TRAce:DEFine? <*name*>	Returns the size, in bytes, of the named memory block.
DIGital:TRAce:DELete[:NAME] <*name*>	Deletes the named memory block.
DIGital:TRAce:DELete:ALL	Deletes all memory blocks.

Command	**Description**
STATus:OPERation:CONDition?	Returns contents of Condition register.
:OPERation:ENABle <*mask*>	Sets mask for Enable register.
:OPERation:ENABle?	Returns mask set in Enable register.
:OPERation[:EVENt]?	Returns the content of the Event register.
:PRESet	Clears Enable registers.
:QUEStionable:CONDition?	Returns contents of Condition register.
:QUEStionable:ENABle <*mask*>	Sets mask for Enable register.
:QUEStionable:ENABle?	Returns mask set in Enable register.
:QUEStionable[:EVENt]?	Returns the content of the Event register.
SYSTemCDEScription? <*number*>	Returns a string description of the module.
:CTYPe? <*number*>	Returns a string of the model number.
:ERRor?	Returns the contents of the system error register.
:VERsion?	Returns the SCPI version to which this instrument complies.

Appendix A
Specifications

Logic Levels:
TTL Compatible, 5V max

Data Lines:
Iout (High): -5.2 mA
@ Vout (High): 2.5 V
(Pullup Enabled)
Iout (Low): 48 mA
@ Vout (Low): 0.5V
Vin (High): >2.0 V; <5.0 V
Vin (Low): <0.8 V
Iin (High): <2.5 mA @ 2.5 V
Iin (Low): <3.2 mA @ 0.4 V

Handshake Lines:
Iout (High): 250 µA
@ Vout (High): 5 V
Iout (Low): 40 mA
@ Vout (Low): 0.7 V
Iout (Low):16 mA
@Vout (Low): 0.4V
Vin (High): >2.0V
Vin (Low): <0.8 V
Iin (Low): <1.75 mA

Module Size/Device Type:
C, register based

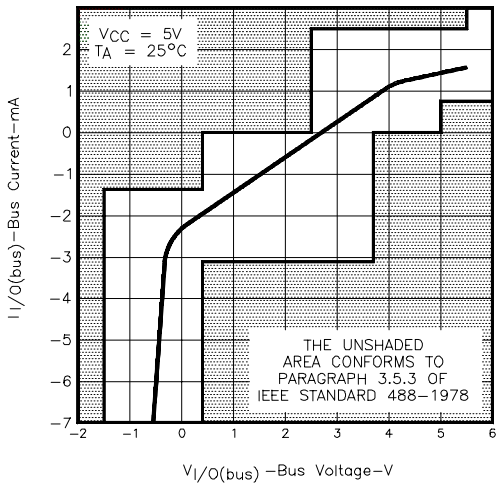
Connectors Used: P1, P2

Number of Slots: 1

VXIbus Interface Capability:
Slave, interrupter, A16, D16, D08EO

Interrupt Level: 1-7, selectable

Typical Data Line Current vs. Data Line Voltage



Power Requirements:
Voltage: +5 V
Peak module current, IPM (A): 1.20
Dynamic module current, IDM (A): 0.01

Watts/Slot: 6.0

Cooling/Slot: 0.05 mm H_2O @ 0.5 liter/sec

Humidity: 65%, 0 to 40°C

Operating Temperature: 0 to 55°C

Storage Temperature: -40 to 75°C

EMC, RFI, Safety:
meets FTZ 1046/1984, CSA 556B, IEC 348, UL 1244

Net Weight (kg): 1.0

Appendix B
Agilent E1458A Register Information

Using This Appendix

The contents of this appendix are:

Note Do not mix register programming and SCPI command programming.

Addressing the Registers

To access a specific register for either read or write operations, the address of the register must be used. Register addresses for the plug-in modules are found in an address space known as VXI A16. The exact location of A16 within a VXIbus master's memory map depends on the design of the the VXIbus master you are using. For the Agilent E1405/E1406 Command Module, the A16 space location starts at $1F0000_{16}$.

The A16 space is further divided so that the modules are addressed only at locations above $1FC000_{16}$ within A16. Further, every module is allocated 64 register addresses (40_{16}). The address of a module is determined by its logical address (set by the address switches on the module) times 64 (40_{16}). In the case of the Agilent E1458A Digital I/O Module, the factory setting is 144 or 90_{16}, so the addresses start at $1FE400_{16}$.

Register addresses for register-based devices are located in the upper 25% of VXI A16 address space. Every VXI device (up to 256) is allocated a 64-byte block of addresses.

Figure B-1 shows the register address location within A16. Figure B-2 shows the location of A16 address space in the Agilent E1405/E1406 Command Module.

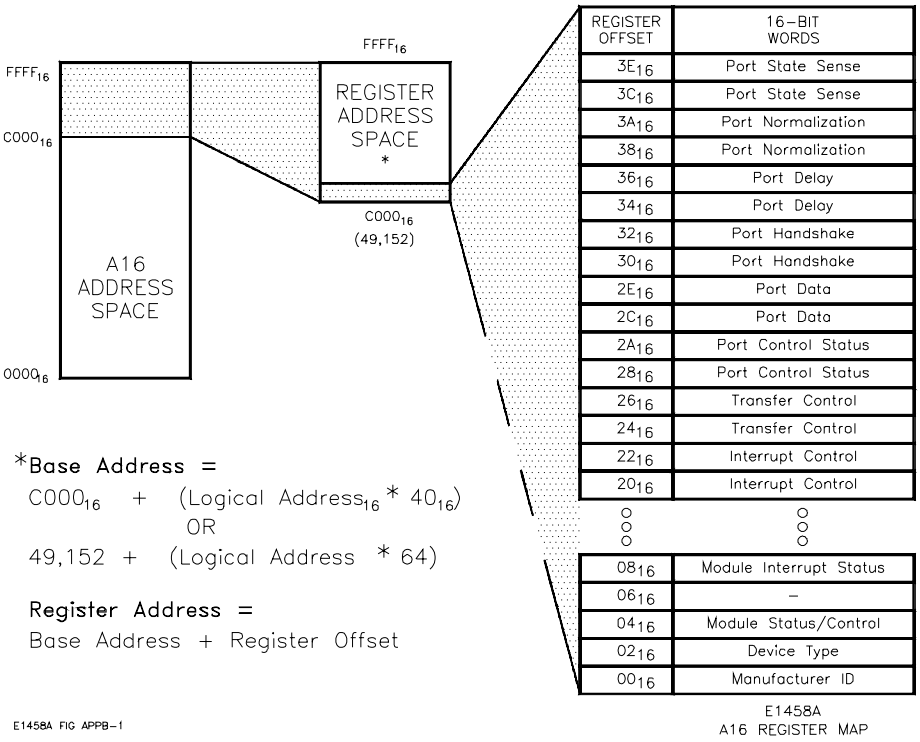


Figure B-1. Register Address Location Within A16

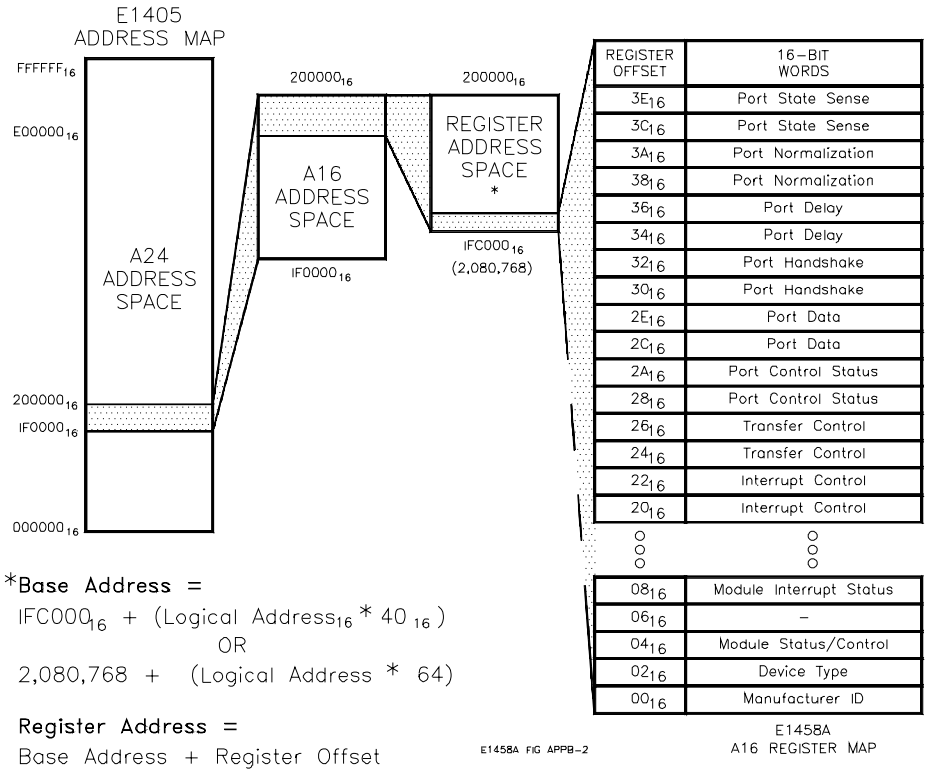


Figure B-2. A16 Address Space in the E1405/E1406

The Base Address

When you are reading or writing to a module register, a hexadecimal or decimal register address is specified. This address consists of a base address plus a register offset. The base address used in register-based programming depends on whether the A16 address space is outside or inside the Agilent E1405/E1406 Command Module.

A16 Address Space Outside the Command Module

When the Agilent E1405/E1406 Command Module is not part of your VXIbus system (Figure B-1), the Agilent E1458A's base address is computed as:

$C000_{16} + (LADDR * 40)_{16}$

or (decimal)

49,152 + (LADDR * 64)

where $C000_{16}$ (49,152) is the starting location of the register addresses, LADDR is the module's logical address, and 64 is the number of address bytes per VXI device. For example, the E1458A's factory-set logical address is 144 (90_{16}), therefore it will have a base address of:

$C000_{16} + (90_{16} * 40_{16}) = C000_{16} + 2400_{16} = \mathbf{E400_{16}}$

or

49,152 + (144 * 64) = 49,152 + 9216 = **58,368**

A16 Address Space Inside the Command Module or Mainframe

When the A16 address space is inside the Agilent E1405/E1406 Command Module (Figure B-2), the module's base address is computed as:

$1FC000_{16} + (LADDR * 40)_{16}$

or

2,080,768 + (LADDR * 64)

where $1FC000_{16}$ (2,080,768) is the starting location of the VXI A16 addresses, LADDR is the module's logical address, and 64 is the number of address bytes per register-based device. Again, the Agilent E1458A's factory-set logical address is 144. If this address is not changed, the module will have a base address of:

$1FC000_{16} + (90_{16} * 64_{16}) = 1FC000_{16} + 2400_{16} = \mathbf{1FE400_{16}}$

or

2,080,768 + (144 * 64) = 2,080,768 + 9216 = **2,089,984**

Register Offset

The register offset is the register's location in the block of 64 address bytes that belong to the module. For example, the module's Status/Control register has an offset of 04_{16}. When you write a command to this register, the offset is added to the base address to form the register address:

$E400_{16} + 04_{16} = \mathbf{E404}_{16}$ $1FE400_{16} + 04_{16} = \mathbf{1FE404}_{16}$

or

58,368 + 4 = **58,372** 2,089,984 + 4 = **2,089,988**

Table B-1 shows the general programming method for accessing the Agilent E1458A Digital I/O Module's registers using different computers.

Table B-1.

System	Typical Commands	Base Address
External Computer (over GPIB to E1405/E1406 Command Module)	VXI:READ? logical_address, offset VXI:WRITE logical_address, offset, data DIAG:PEEK? Base_addr + offset, width DIAG:POKE Base_addr +offset, width, data (width must be either 8 or 16)	logical_address = LADDR offset = register number Base_addr = $1FC000_{16}$+ ($LADDR_{16}$ *40_{16}) or = 2,080,768 + (LADDR * 64) offset = register number
V/360 Embedded Computer	READIO (-18, Base_addr + offset) WRITEIO (-18, Base_addr + offset; data) (positive select code = byte read or write negative select code = word read or write)	Base_addr = $C000_{16}$ + ($LADDR_{16}$ *40_{16}) or = 49,152 + (LADDR * 64) offset = register number
SICL	iwpoke(Base_addr + offset,data) iwpeek(Base_addr + offset)	imap(id,I_MAP_VXIDEV,0,0,NULL)
LADDR = E1458A Logical Address = 144 = 90_{16}		

Reset and Registers

Following power-on or a *RST command, the bits of the registers are put into the following states:

- The Manufacturer ID and Device ID registers remain unaffected.
- The I/$\overline{O}$ bits (bit 6 of the Port Control/Status registers) are set to 1, enabling all ports for input.
- The port delay register is set to 2 µs.
- The port handshake register is set to interrupt driver.
- All other bits of all registers are set to 0.

Register Definitions

You can program the Agilent E1458A 96-Channel Digital I/O Module using its hardware registers. *The procedures for reading or writing to a register depend on your operating system and programming language.* Whatever the access method, you will need to identify each register with its address. These addresses are given in Table B - 2.

Table B-2. Register Map

Register Name	MSB Address		LSB Address	
Manufacturer ID	00_{16}		01_{16}	
Device ID	02_{16}		0316	
Card Status/Control	04_{16}		0516	
Card Interrupt Status	08_{16}		0916	
Port Register Name	**Port *n***	**Port *n*+1**	**Port *n*+2**	**Port *n*+3**
Port Interrupt Control	20_{16}	2116	22_{16}	23_{16}
Port Transfer Control	24_{16}	2516	26_{16}	27_{16}
Port Control/Status	28_{16}	2916	$2A_{16}$	$2B_{16}$
Port Data	$2C_{16}$	2D16	$2E_{16}$	$2F_{16}$
Port Handshake	30_{16}	3116	32_{16}	33_{16}
Port Delay	34_{16}	3516	36_{16}	37_{16}
Port Normalization	38_{16}	3916	$3A_{16}$	$3B_{16}$
Port State Sense	$3C_{16}$	3D16	$3E_{16}$	$3F_{16}$

The module is a register-based slave/interrupter device, supporting VME D16, D8(O), and D8(OE) transfers. The interrupt protocol supported is "release on interrupt acknowledge" – an interrupt is cleared by a VXIbus interrupt acknowledge cycle.

WARNING **Registers have been documented as 8-bit bytes. If you access them using 16-bit transfers from a Motorola CPU, the high and low byte will be swapped. The Agilent command modules (E1405/E1406) use Motorola CPUs. Motorola CPUs place the higher weighted byte in the lower memory location and the lower weighted byte in the higher memory address; Intel processors do just the opposite. VXI registers are memory mapped, thus you will see this Motorola/Intel byte swap difference when doing register programming.**

Register Descriptions

The following pages detail each register in the Agilent E1458A Digital I/O Module. Registers are listed by name in the order shown in the register map, Table B-2.

Manufacturer Identification Register

The Manufacturer Identification Register is a 16-bit read-only register with the Most Significant Byte (MSB) at address 00_{16} and Least Significant Byte (LSB) at address 01_{16}. Reading this register returns the Agilent Technologies identification, $FFFF_{16}$.

Device Identification Register

The Device Identification Register is a 16-bit read-only register at addresses 02_{16} and 03_{16}. The Most Significant Byte (02_{16}) returns 01_{16}. The Least Significant Byte (03_{16}) returns 55_{16}.

Card Status/Control Register

The Card Status/Control Register is a read/write register at addresses 04_{16} and 05_{16}. The following table shows the register bit patterns:

base+04_{16}								base+05_{16}							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	MOD ID	1	1	1	1	BB1	BB0	1	IEN	1	1	1	1	1	SR

SR (soft reset). Writing a 1 and then a 0 to this bit resets all digital I/O module components. SR disables all output ports (all ports become input ports) and sets all other registers to default values. Reads and writes to the other module registers will not transfer valid data when SR is asserted. This bit is cleared by a hard reset.

IEN. Main interrupt enable. Writing a 1 to this bit allows interrupts from port controller ICs to assert interrupt on the VXIbus. Writing a 0 masks these interrupts. This bit is cleared by a hard reset, but not by a soft reset.

CAUTION **A potential race condition exists when clearing this bit or masking interrupts by means of register 08_{16} through $0B_{16}$. If an interrupt occurs just before interrupts are masked, it could be asserted on the VXIbus but not acknowledged by the digital I/O module. Therefore, use care in disabling interrupts once they have been enabled.**

BB0 and **BB1** are the bank enable bits. These bits must be set before accessing any port registers (from 20_{16} to $3F_{16}$). The three legal states of these bits and the port registers they enable access to are shown below:

BB1	BB0	Ports Enabled
0	0	0, 1, 2, and 3
0	1	4, 5, 6, and 7
1	0	8, 9, 10, and 11

NOTE The ports are divided into three banks for register control. Mechanically, the ports are divided into four connectors.

MODID. A `1` in this field indicates the device not selected via the P2 connector MODID line. A `0` indicates that the device is selected. The MODID line and this bit is written by slot 0 device (command modules). The *VMEbus Extensions for Instrumentation System Specification* contains additional descriptions of this bit in the "Device Operation" and "System Resources" sections.

Card Interrupt Status Register

The Card Interrupt Status Register is a 16-bit read-only register at addresses 08_{16} and 09_{16}. When interrupt is enabled, this register indicates which port caused the interrupt. The following table shows the register bit patterns.

base+ 08_{16}								base+09_{16}							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	I10	I11	I8	I9	I6	I7	I4	I5	I2	I3	I0	I1

I*n* set to 1 indicates an interrupt was generated by the port controller *n*. These bits are cleared after a soft or hard reset.

Port Interrupt Control Register

The Port Interrupt Control Register is an 8-bit read/write register and functions as the interrupt register for the port. This register shows the interrupt enable status, the level of interrupt that can signal the controller (always set to 0), and whether an interrupt is pending. The ports affected by this register are set by the BB1 and BB0 bits in the Card Status/Control Register.

base+20_{16}, base+21_{16}, base+22_{16}, base+23_{16}

7	6	5	4	3	2	1	0
PIEN	IP	IL1	IL0	—	—	—	—

Bits (0-3). Unused

IL0 and **IL1 (Interrupt Level)**. Both bits *must* be left at 0 to initialize the Agilent E1458A Digital I/O Module for interrupt operation.

IP (Interrupt pending). When equal to 1, indicates an interrupt is pending. This is a read/write bit. You can force a hardware interrupt by setting this bit to 1 if PIEN is set to 1 and IEN is set to 1 in the Status/Control Register.

PIEN (Port Interrupt enable). When set to 1, enables interrupt. Pending or forced interrupts are ignored if set to 0.

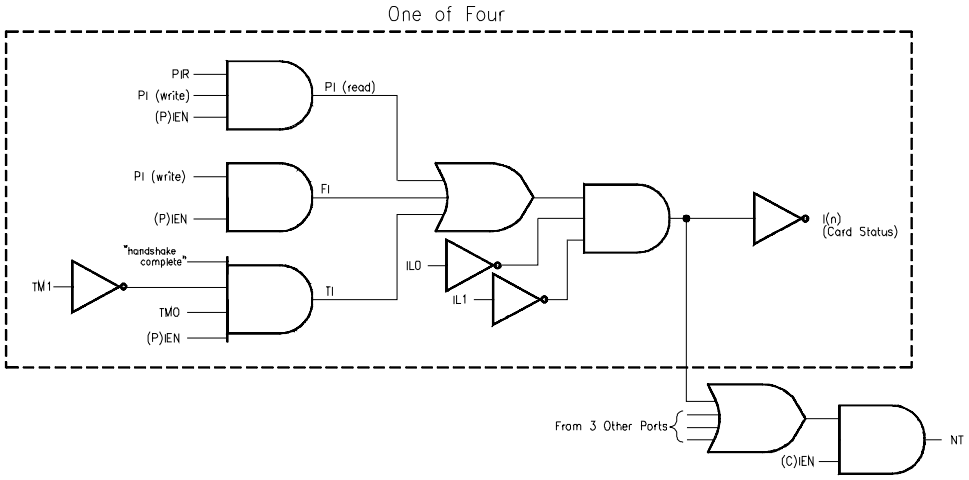


Figure B-3. Interrupt Line Logic Diagram

Port Transfer Control Register

The Port Transfer Control Register controls transfers between the mainframe and port, identifies port interrupts, and identifies forced interrupts from the controller. The ports affected by this register are set by the BB1 and BB0 bits in the Card Status/Control Register.

base+24_{16}, base+25_{16}, base+26_{16}, base+27_{16}

7	6	5	4	3	2	1	0
PI	FI	TI	—	—	—	HE	DRR

DRR (Data Register Ready). A read-only bit. When set to 1 it indicates either that the Port Data Register contains valid data for the mainframe to read, or that the Port Data Register is ready for the mainframe to write a byte of data to it. When the Port Data Register is read, DRR is set to 0.

HE (Handshake Enable). When set to 1 enables handshaking for the port. You can read from or write to this bit. When the registers have been initialized, you can set this bit to 1 to enable handshaking if you are using the port handshake lines to transfer data.

Bits 2 - 4. Unused

TI (Transfer Interrupt). This is a read-only bit. When set to 1 indicates a port transfer has occurred. A port transfer interrupt, if enabled, occurs on a "port data register ready" condition (when bit 0 of this register is set to 1). To enable port transfer interrupts, specify the "interrupt driven" transfer mode of port (refer to Port Handshake Register) and set the "interrupt enable" bit (bit 7 of Interrupt Control Register) equal to 1. When the Port Data Register is read, TI is set to 0.

FI (Forced Interrupt). This is a read-only bit. When set to 1 indicates that a forced interrupt (from the mainframe) has occurred. To force an interrupt, write a 1 to bit 6 and bit 7 of the Port Interrupt Control Register and bit 6 of the Status/Control Register.

PI (Peripheral Interrupt). Bit 7 is a read/write bit. Writing a 1 to bit 7 enables port peripheral interrupts. Writing a 0 disables port peripheral interrupts. When reading bit 7, a 1 indicates a port interrupt has occurred. To clear PI you must write a 0 to PI. Writing a 0 then a 1 to PI is the correct procedure to clear one interrupt and re-enable for a second one.

Note Port peripheral interrupts are caused by a transition in the PIR line. If bit 4 of the Port Normalization Register is 0, a rising-edge (low to high) transition caused the interrupt. If bit 4 is set to 1, a falling-edge (high to low) transition caused the interrupt. Refer to the Port Normalization Register for more information.

Port Control/Status Register

The Port Control/Status Register is a read/write register that controls and shows the status of the port control lines (STS, PIR, FLG, $\overline{\text{RES}}$, I/$\overline{\text{O}}$ and CTL). The ports affected by this register are set by the BB1 and BB0 bits in the Card Status/Control Register.

base+28$_{16}$, base+29$_{16}$, base+2A$_{16}$, base+2B$_{16}$

7	6	5	4	3	2	1	0
CTL	I/$\overline{\text{O}}$	$\overline{\text{RES}}$	FLG	—	—	PIR	STS

STS. Bit 0 is a read-only bit. Read this bit to find the status of the STS line which is an input from the peripheral for the port. A 1 shows that the line is BUSY; a 0 shows that the line is READY.

PIR. Bit 1 is a read-only bit. This bit shows the *normalized* state of the PIR line which is an input line from the peripheral:

- If positive-true logic is in use (bit 4 of the Port Normalization Register is equal to 0), bit 1 is equal to 0 if the line is low, 1 if the line is high.
- If the PIR line is inverted (bit 4 of the Port Normalization Register is equal to 1), bit 1 is equal to 0 if the line is high, 1 if the line is low.

If peripheral interrupts are not enabled, you can use the PIR line as a secondary status line. Just read bit 1 to monitor the state of the line.

If peripheral interrupts are enabled, you can still monitor the status of the PIR line by reading bit 1. However, the current status of the PIR line does not indicate whether a peripheral interrupt has occurred. Port peripheral interrupts are caused by transitions in the state of the PIR line. Read bit 7 of the Port Transfer Control Register to determine whether a port peripheral interrupt has occurred.

Bits 2 and 3. Unused

FLG. This is a read-only bit. Read this bit to find the *normalized* status of the FLG line. A 1 shows that the line is BUSY; a 0 shows that the line is READY. This bit shows the logical state (BUSY or READY) of the FLG line, regardless of the logic sense.

$\overline{\text{RES}}$. This is a read/write bit. Reading this bit shows the current state of the $\overline{\text{RES}}$ line which is an output line to the peripheral. A 1 shows that the line is high; a 0 shows that the line is low. Bit 5 is initially set to 0 by a hardware reset of the interface. This causes the $\overline{\text{RES}}$ line to go low, resetting the peripheral (if the peripheral implements the reset feature). You can control the logical state of the $\overline{\text{RES}}$ line by writing to this bit. Set bit 5 equal to 1 to change $\overline{\text{RES}}$ to the high state. The peripheral will then operate normally. To reset the peripheral, set bit 5 to 0, putting $\overline{\text{RES}}$ in the low state.

I/$\overline{O}$. This is a read/write bit. Read this bit to find the current status of the I/$\overline{O}$ line, which is an output line to the peripheral from the port data transceiver. If bit 6 is equal to 0, the line is FALSE and the transceiver is enabled for output. If bit 6 is equal to 1, the line is TRUE and the transceiver is enabled for input. *This bit is equal to 1 (input) after a hardware reset*. You can select input or output by changing this bit.

Note If you are using the port handshake lines to control transfers, use the I/$\overline{O}$ line to control the direction of data transfer to your peripheral. Make sure that the peripheral is always enabled to send data during input transfers and to receive data during output transfers.

CTL. This is a read/write bit. Read this bit to find the current state of the CTL line. A 1 shows the line is TRUE; a 0 shows the line is FALSE (the bit is not normalized). When handshaking is enabled (bit 1 of the Port Transfer Control Register is set), the CTL line is controlled by the port controller. To prevent incorrect handshaking due to interaction with other lines, before enabling handshaking set the control line to FALSE.

If handshaking is not enabled and the handshake mode is set to NONE, you can control the logical state of the CTL line by writing a *normalized* 1 or 0 to bit 7.

Port Data Register

The Port Data Register is a read/write register. It is used for both output and input. Its operation depends on the state of the I/$\overline{O}$ bit in the Port Status/Control Register. The ports affected by this register are set by the BB1 and BB0 bits in the Card Status/Control Register.

base+2C$_{16}$, base+2D$_{16}$, base+2E$_{16}$, base+2F$_{16}$

7	6	5	4	3	2	1	0
D7	D6	D5	D4	D3	D2	D1	D0

- If I/$\overline{O}$ is set for output (bit 6, Port Transfer Control Register = 0), data written to the Port Data Register is latched and remains until new data is written. The current data in the Port Data Register drives the port data bus. If you read Port Data Register, the value read is the value last written to the register.
- If I/$\overline{O}$ is set for input (bit 6, Port Transfer Control Register =1), the data read from the Port Data Register is the data transmitted by the peripheral on the port data bus. If you write to the Port Data Register, the data is latched for output, but the data lines are not affected until I/$\overline{O}$ is again set for output.

- When the Port Data register is read the following bits are set to 0 on the Port Transfer Control Register: DRR (bit 0), TI (bit 5), and PI (bit 7).

Bits 0-7 of the Port Data Register correspond to data lines D (0-7) where bit 7 is the most significant bit.

Port Handshake Register

The Port Handshake Register determines the type of handshake protocol used for the port data transfers and how the data is transferred from the Agilent E1458A Digital I/O Module to the mainframe on the VXIbus. The ports affected by this register are set by the BB1 and BB0 bits in the Card Status/Control Register.

base+30_{16}, base+31_{16}, base+32_{16}, base+33_{16}

7	6	5	4	3	2	1	0
HT2	HT1	HT0	EI	—	—	TM1	TM0

TM (0,1)(Transfer Mode). These bits control the transfer mode for the port between the digital I/O module and the VXIbus as shown in Table B-3.

Table B-3. Transfer Mode

Transfer Mode	TM1 (Bit 1)	TM0 (Bit 0)
Flag Driven	0	0
Interrupt Driven	0	1
Fast Handshake	1	0

The three transfer modes are used to transfer data between the VXIbus and the digital I/O module:

- Flag Driven – the mainframe polls the Data Register Ready bit (bit 0, Port Transfer Control Register). When this bit is set, it reads data from the Port Data Register or writes data to the Port Data Register.
- Interrupt Driven – the peripheral sets bit 1 of the Port Status/Control Register (via the PIR line) and the digital I/O module interrupts the VXIbus for data transfer with the mainframe.
- Fast Handshake – the peripheral talks directly with the VXIbus's Data Acknowledge line to transfer data between the Port Data Registers and the VXIbus.

Bits 2 and 3. Unused

EI (Enable Inhibit). This bit, if set to 1, enables the STS line to inhibit a transfer cycle during a transfer. If bit 4 is set, the transfer is inhibited when the peripheral puts STS in the BUSY state and resumes when STS returns to the READY state.

HT (5-7)(Handshake Type). These bits determine the type of handshake for port input and output transfers as shown in Table B-4.

Table B-4. Handshake Type

Output/Input Transfer	Bit 7	Bit 6	Bit 5
No Handshake	0	0	0
Leading Edge	0	0	1
Trailing Edge	0	1	0
Pulse	0	1	1
Partial	1	0	0
Strobe	1	0	1

Port Delay Register

The Port Delay Register sets the delay time, T_d. Delay time is the time between data valid and setting the control (CTL) line TRUE. It is used with several handshake modes. You can also read this register to find the current delay time. The ports affected by this register are set by the BB1 and BB0 bits in the Card Status/Control Register.

base+34_{16}, base+35_{16}, base+36_{16}, base +37_{16}

7	6	5	4	3	2	1	0
DF7	DF6	DF5	DF4	—	—	RM1	RM0

RM (0,1)(Range Multiplier). You can specify the range of delay time, T_d, by selecting the one of the range multipliers in Table B-5.

Table B-5. Range Multipliers.

Range Multiplier	RM1 (Bit 1)	RM0 (Bit 0)
1 ms	0	0
100µs	0	1
10µs	1	0
1µs	1	1

Bits 2 and 3. Unused

DF (4-7)(Delay Factor). Regardless of the range multiplier you select, you can specify a delay factor in the range of 0 through 15 (decimal equivalent of the binary value) by setting these bits (0 specifies no delay time). For all output handshake types, the delay period T_d is equal to the range multiplier times the delay factor specified by bits 4 - 7. For example, if you write the value 00010000 to register 5, the multiplier is 1 ms and the delay factor is 1. If you write 11110010 to register 5, then the multiplier is 10 µs and the delay factor is 15; hence, the delay factor is 150 µs. The actual delay for a given transfer may be one count longer due to uncertainty in recognizing a transition of a handshake signal.

Note If you are using the output strobe or pulse handshake, you can specify delay factors in the range 2 through 15, or you can specify 0 (no delay period). Thus, you can specify T_d values from 2 to 15 µs, from 20 to 150 µs, and so forth for these handshakes. Setting T_d to 0 for strobe or pulse handshake is an illegal setting and the pulse width may be too narrow to be detected.

If you are using the input strobe handshake, the delay factor specified by bits 4 through 7 is reduced by one, then multiplied by the range multiplier. For example, the register value 00100000 for an input strobe handshake specifies T_d = 1 ms. (The multiplier is 1 ms and the delay factor is 2-1= 1.) On the other hand, the value 11110010 specifies T_d = 140 µs. (The multiplier is 10 µs and the delay factor is 15-1=14.)

The input strobe handshake is the only *input* handshake that uses a delay period. For the other input handshakes the value in this register has no effect.

Port Normalization Register

The Port Normalization Register allows you to normalize the port handshake and data lines to the correct logic sense for your peripheral. Positive true logic is the default. You can invert a line by setting the appropriate bit equal to 1. The ports affected by this register are set by the BB1 and BB0 bits in the Card Status/Control Register.

base+38_{16}, base+39_{16}, base+$3A_{16}$, base+$3B_{16}$

7	6	5	4	3	2	1	0
ID	ICTL	IFLG	IPIR	—	—	—	—

Bits 0 - 3. Unused.

IPIR (Invert PIR). This bit specifies the logic sense of a peripheral interrupt request. If bit 4 = 0, a rising-edge (low to high) transition of the PIR line triggers an interrupt. If bit 4 = 1, a falling-edge (high to low) transition of the PIR line triggers an interrupt. In either case, no interrupt occurs unless peripheral interrupts are enabled.

IFLG (Invert FLG). This bit specifies the logic sense of the FLG line.
If bit 5 = 0, then positive-true logic is used: HIGH = BUSY, LOW = READY.
If bit 5 = 1, then negative-true logic is used: LOW = BUSY, HIGH = READY.

ICTL (Invert CTL). This bit specifies the logic sense of the CTL line.
If bit 6 = 0, then positive-true logic is used: HIGH = TRUE, LOW = FALSE.
If bit 6 = 1, then negative-true logic is used: LOW = TRUE, HIGH = FALSE.

ID (Invert DATA). This bit specifies the logic sense of the port data lines.
If bit 7 = 0, then positive-true logic is used: HIGH = TRUE, LOW = FALSE.
If bit 7 = 1, then negative-true logic is used: LOW = TRUE, HIGH = FALSE.

Port State Sense Register

The Port State Sense Register is a read-only register. This register performs an active readback of the state of the port data lines, independent of the state of the Port Data Register. The ports affected by this register are set by the BB1 and BB0 bits in the Card Status/Control Register.

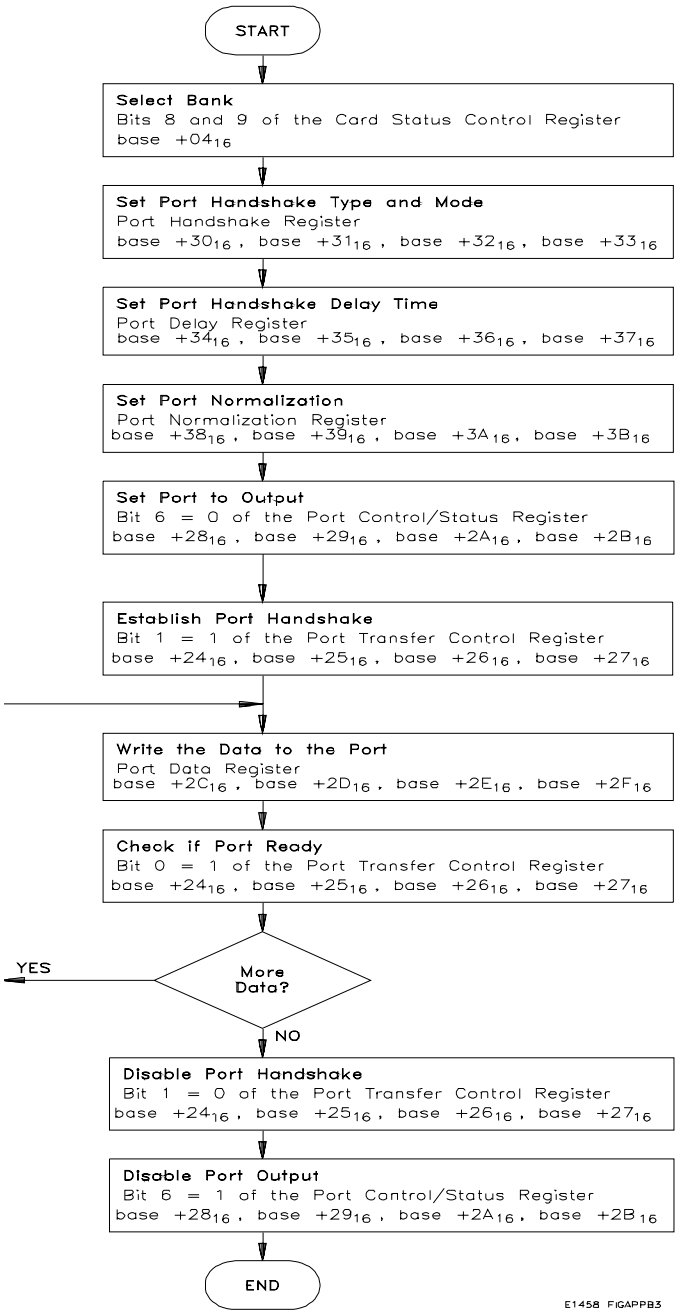
base+2C$_{16}$, base+2D$_{16}$, base+2E$_{16}$, base+2F$_{16}$

7	6	5	4	3	2	1	0
D7	D6	D5	D4	D3	D2	D1	D0

D0-**D7** correspond to data lines D*n*_0 through D*n*_7. Bit 7 is the most significant bit. The values of these bits are not affected by the Port Normalization Register. A 1 indicates a data line in the TTL High condition.

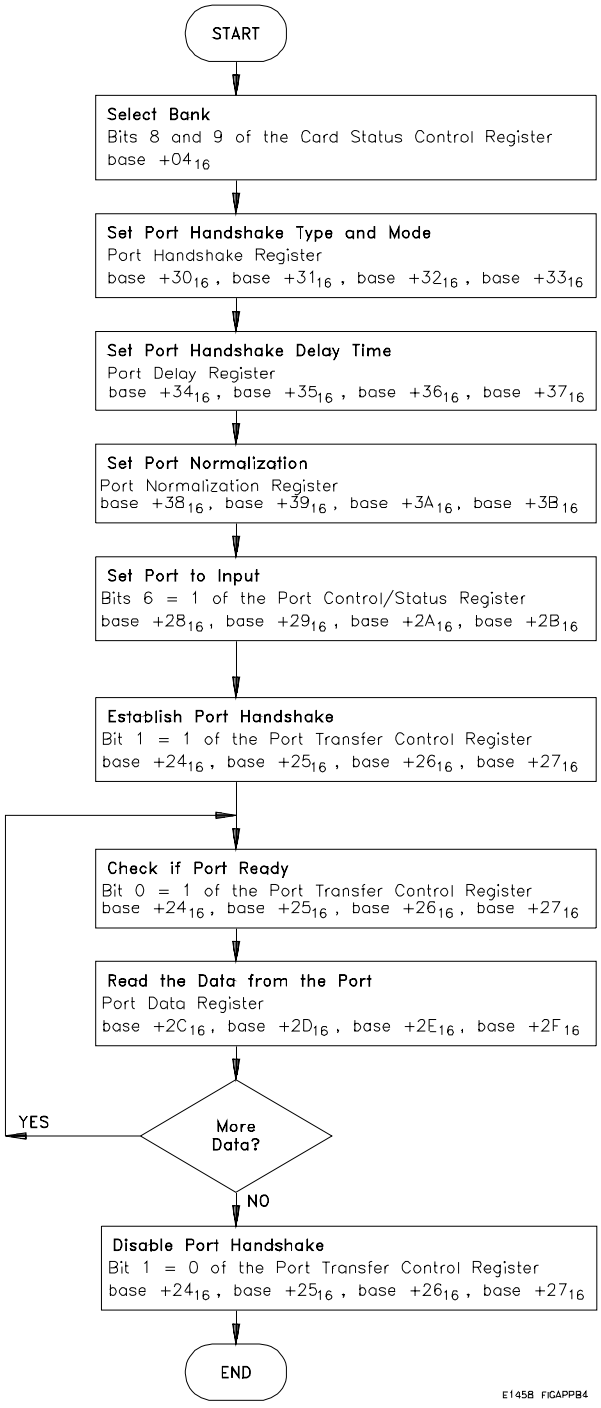
A Register-Based Output Algorithm

The following algorithm describes a procedure to program the registers to transmit a byte of data to a peripheral. The algorithm follows a flag-driven output procedure initiated by the computer. The computer polls the Agilent E1458A Digital I/O Module to see if the data has been accepted by the peripheral by checking the Port Transfer/Control Register, bit 0 (referred to as the acknowledge flag --- hence, the name of flag-driven). Once the flag is TRUE the computer can output new data to the port. The actual path followed by the peripheral and the digital I/O module to set this bit is controlled by the handshake mode you select.



A Register-Based Input Algorithm

The following algorithm describes a procedure to program the registers to read a byte of data from a peripheral. The algorithm follows a flag-driven input procedure initiated by the computer. The computer polls the Agilent E1458A Digital I/O Module to see if the data has been transmitted by the peripheral by checking the Port Transfer/Control Register, bit 0 (referred to as the acknowledge flag --- hence, the name flag-driven). Once the flag is TRUE the computer can read new data from the port. The actual path followed by the peripheral and the digital I/O module to set this bit is controlled by the handshake mode you select.



Programming Example

The example in this section demonstrates how to program the module at the register level. The programs follow the execution and timing models covered in the previous section. The examples in this section include:

- Resetting the module
- Reading the ID, Device Type, and Status Registers
- Writing 8-bit data bytes
- Reading 8-bit data bytes
- Handshaking a port

System Configuration

The following example programs were developed with the module at logical address 144. The **C** language programs were developed on an Agilent V382 using ANSI C programming language and SICL (Standard Instrument Control Library).

```
/* C register programming example */

#include <stdio.h>
#include <sicl.h>

/* Setup the registers and offsets */

#define mfr_id   0x00
#define dev_id   0x02
#define card_stat      0x04
#define port_xfr_0   0x24
#define port_xfr_1   0x25
#define port_xfr_2   0x26
#define port_xfr_3   0x27
#define port_ctl_0   0x28
#define port_ctl_1   0x29
#define port_ctl_2   0x2A
#define port_ctl_3   0x2B
#define port_data_0 0x2C
#define port_data_1 0x2D
#define port_data_2 0x2E
#define port_data_3 0x2F
#define port_hand_0      0x30
#define port_hand_1      0x31
#define port_hand_2      0x32
#define port_hand_3      0x33
#define port_del_0   0x34
#define port_del_1   0x35
#define port_del_2   0x36
```

Continued on next page.

```
#define port_del_3   0x37
#define port_norm_0      0x38
#define port_norm_1      0x39
#define port_norm_2      0x3A
#define port_norm_3      0x3B

/* establish card status/control register bank selection constants */

#define bank_0  0xFCBE
#define bank_1  0xFDBE
#define bank_2  0xFEBE

/* set up bytes to output */

#define pattern_1     0xAA
#define pattern_2     0x55

/* function to test the data ready bit of the port transfer control register */

int test_drr (int reg_addr){
        int test_1;
        unsigned char reg_byte;
        reg_byte = ibpeek (reg_addr);
        reg_byte = reg_byte << 7;
        if (reg_byte = 0x80)
          test_1 = 0; /* port is ready */
        else
          test_1 = 1; /* port not ready */
        return test_1; }

main ()  {

        INST id;
        unsigned short data_word;
        unsigned char data_byte;
        int reg_num;
        char *base_addr;
        int errnum;

/* open a path to digital I/O module */

        id = iopen("vxi,144");
        if (id == 0){
          errnum = igeterrno();
          printf ("iopen failed: error = %d,%s\n\n",errnum,igeterrstr(errnum));
          exit (-1);  }

/* get base address  */
```

Continued on next page.

```
        base_addr = imap(id,I_MAP_VXIDEV,0,0,NULL);
        if (base_addr == NULL){
          errnum = igeterrno();
          printf("imap failure: error = %d,%s\n",errnum,igeterrstr(errnum));
          exit (-1);  }

/* perform a soft reset */

        iwpoke((base_addr + card_stat),0xFCBF);
        iwpoke((base_addr + card_stat),0xFCBE;

/* read MFR and device ID registers */

        data_word = iwpeek (base_addr + mfr_id);
        printf("MFR ID value = %04X\n",data_word);
        data_word = iwpeek (base_addr + dev_id);
        printf("Dev ID value = %04X\n",data_word);

/* output data bytes to ports 0 and 5, no handshake */

        /* port 0 */
        iwpoke((base_addr + card_stat),bank_0);
        ibpoke((base_addr + port_hand_0),0x00);
        ibpoke((base_addr + port_del_0),0x00);
        ibpoke((base_addr + port_norm_0),0x00);
        ibpoke((base_addr + port_ctl_0),0x00);
        ibpoke((base_addr + port_xfr_0),0x00);
        ibpoke((base_addr + port_data_0),pattern_1);


        /* port 5 */
        iwpoke((base_addr + card_stat),bank_1);
        ibpoke((base_addr + port_hand_1),0x00);
        ibpoke((base_addr + port_del_1),0x00);
        ibpoke((base_addr + port_norm_1),0x00);
        ibpoke((base_addr + port_ctl_1),0x00);
        ibpoke((base_addr + port_xfr_1),0x00);
        ibpoke((base_addr + port_data_1),pattern_2);


        /* return ports back to input state  */
        iwpoke((base_addr + card_stat),bank_0);
        ibpoke((base_addr + port_ctl_0),0x40);
        ibpoke((base_addr + port_xfr_0),0x00);
        iwpoke((base_addr + card_stat),bank_1);
        ibpoke((base_addr + port_ctl_1),0x40);
        ibpoke((base_addr + port_xfr_0),0x00);
```

Continued on next page.

```
/* input a data byte at port 9, no handshake */

        iwpoke((base_addr + card_stat),bank_2);
        ibpoke((base_addr + port_hand_1),0x00);
        ibpoke((base_addr + port_del_1),0x00);
        ibpoke((base_addr + port_norm_1),0x00);
        ibpoke((base_addr + port_ctl_1),0x40);
        ibpoke((base_addr + port_xfr_1),0x00);
        data_byte = ibpeek(base_addr + port_data_1);
        printf(("port data register value = %02X\n",data_byte);

/* input a data byte at port 11, leading edge handshake */

        iwpoke((base_addr + card_stat),bank_3);
        ibpoke((base_addr + port_hand_3),0x20);
        ibpoke((base_addr + port_del_3),0xF2);
        ibpoke((base_addr + port_norm_3),0x00);
        ibpoke((base_addr + port_ctl_3),0x40);
        ibpoke((base_addr + port_xfr_3),0x02);
        count = 0;
        while (test_drr(base_addr + port_xfr_3)){
          count = count++;
          if (count == 100) {
               printf("DRR bit not ready ");
               exit (-1); }
        }
        data_byte = ibpeek(base_addr + port_data_3);
        printf(("port data register value = %02X\n",data_byte);

/* disable port handshake */
        iwpoke((base_addr + card_stat),bank_2);
        ibpoke((base_addr + port_xfr_1),0x00);
        iwpoke((base_addr + card_stat),bank_3);
        ibpoke((base_addr + port_xfr_3),0x00);

return 0;  }
```

Appendix C
Agilent E1458A Error Messages

Code	Message	Cause
-101	Invalid character	Unrecognized character in specified parameter.
-102	Syntax Error	Command is missing a space or comma between parameters.
-103	Invalid separator	Command parameters are not separated by a comma.
-104	Data Type Error	The wrong data type (i.e. number, character, string, expression) was used when specifying a parameter.
-108	Parameter not allowed	Parameter specified in a command where none is allowed.
-109	Missing parameter	No parameter specified when required.
-113	Undefined header	Command header was incorrectly specified.
-124	Too many digits	>257 digits were specified for a parameter.
-128	Numeric data not allowed	A number was specified for a parameter when a letter is required.
-131	Invalid suffix	Parameter suffix incorrectly specified.
-138	Suffix not allowed	Parameter suffix is specified when one is not allowed.
-141	Invalid character data	The parameter type specified is not allowed.
-161	Invalid block data	Mismatch between character count in header and actual number of characters.
-178	Expression data not allowed	A parameter is enclosed in parenthesis.
-221	Settings conflict	Digital I/O command settings are in conflict (e.g., control asserted when in a handshake mode other than NONE).
-222	Data out of range	Value specified is out of the legal range for parameter.
-224	Illegal Parameter Value	Inconsistent parameter value or block not found.
-240	Hardware Error	Hardware error detected during power-on cycle. Return digital I/O module to Agilent Technologies for repair.
-410	Query Interrupted	Data is not read from the output buffer before another command is issued.
-420	Query unterminated	Command which generates data not able to finish executing due to a digital I/O module configuration error.
-430	Query deadlocked	Command execution cannot continue since the mainframe's command input and data buffers are full. Clearing the instrument restores control.
1000	Out of memory	No memory available.
2025	Invalid port number for access TYPE	The port number specified is not valid for the [:type] set.
2026	Port number out of range	The port number specified is out of the legal range of port numbers.
2027	Invalid bit number for access TYPE	The bit number specified is not valid for the port [:type] set.
2029	Duplicate memory block name	The memory block name specified already exists.
2030	Invalid number of bytes for TRACE access TYPE	The number of bytes specified in a trace does not match the [:type] set.

Index

A

B

C

C (cont'd)

D

E

F

G

H

M (cont'd)

N

O

P

Q

R

R (cont'd)

S

T

U

V

W