

17XXA-002

Parallel Interface



17XXA-002

Parallel Interface

P/N 717249

JUNE 1984

©1984 John Fluke Mfg. Co., Inc.
All rights reserved. Litho in U.S.A.



WARRANTY

John Fluke Mfg. Co., Inc. (Fluke) warrants this instrument to be free from defects in material and workmanship under normal use and service for a period of one (1) year from date of shipment. Software is warranted to operate in accordance with its programmed instructions on appropriate Fluke instruments. It is not warranted to be error free. This warranty extends only to the original purchaser and shall not apply to fuses, computer media, batteries or any instrument which, in Fluke's sole opinion, has been subject to misuse, alteration, abuse or abnormal conditions of operation or handling.

Fluke's obligation under this warranty is limited to repair or replacement of an instrument which is returned to an authorized service center within the warranty period and is determined, upon examination by Fluke, to be defective. If Fluke determines that the defect or malfunction has been caused by misuse, alteration, abuse, or abnormal conditions of operation or handling, Fluke will repair the instrument and bill purchaser for the reasonable cost of repair. If the instrument is not covered by this warranty, Fluke will, if requested by purchaser, submit an estimate of the repair costs before work is started.

To obtain repair service under this warranty purchaser must forward the instrument, (transportation prepaid) and a description of the malfunction to the nearest Fluke Service Center. The instrument shall be repaired at the Service Center or at the factory, at Fluke's option, and returned to purchaser, transportation prepaid. The instrument should be shipped in the original packing carton or a rigid container padded with at least four inches of shock absorbing material. **FLUKE ASSUMES NO RISK FOR IN-TRANSIT DAMAGE.**

THE FOREGOING WARRANTY IS PURCHASER'S SOLE AND EXCLUSIVE REMEDY AND IS IN LIEU OF ALL OTHER WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR ANY PARTICULAR PURPOSE OR USE. FLUKE SHALL NOT BE LIABLE FOR ANY SPECIAL, INDIRECT, INCIDENTAL, OR CONSEQUENTIAL DAMAGES OR LOSS WHETHER IN CONTRACT, TORT, OR OTHERWISE.

CLAIMS

Immediately upon arrival, purchaser shall check the packing container against the enclosed packing list and shall, within thirty (30) days of arrival, give Fluke notice of shortages or any nonconformity with the terms of the order. If purchaser fails to give notice, the delivery shall be deemed to conform with the terms of the order.

The purchaser assumes all risk of loss or damage to instruments upon delivery by Fluke to the carrier. If an instrument is damaged in-transit, **PURCHASER MUST FILE ALL CLAIMS FOR DAMAGE WITH THE CARRIER** to obtain compensation. Upon request by purchaser, Fluke will submit an estimate of the cost to repair shipment damage.

Fluke will be happy to answer all questions to enhance the use of this instrument. Please address your requests or correspondence to: **JOHN FLUKE MFG. CO., INC., P.O. BOX C9090, EVERETT, WA 98206, ATTN: Sales Dept.** For European Customers: **Fluke (Holland) B.V., P.O. Box 5053, 5004 EB, Tilburg, The Netherlands.**

Contents

1	HOW TO USE THIS MANUAL	1-1
	Introduction	1-2
	Organization	1-3
	Usage Guide	1-4
	Evaluators	1-4
	Beginning Interface Designers	1-4
	Experienced Interface Designers	1-4
	Troubleshooters	1-5
2	INSTALLATION	2-1
	Introduction	2-2
	Unpacking And Shipping Information	2-3
	Installation	2-4
	Configuration Procedure	2-5
	Software Compatibility	2-7
	Installation Procedure	2-9
	Getting Started	2-11
3	INTERFACE DESCRIPTION	3-1
	Introduction	3-2
	Hardware Interface Description	3-3
	Port Lines	3-3
	Port Signals	3-3
	Port Control Jumpers	3-6
	Notes on the Port Control Lines	3-8
	Port Direction Line (PDIR)	3-8
	Port Output Enable Line (POE)	3-10

CONTENTS, *continued*

Interface Circuits	3-11
Typical Receiver Circuit	3-14
Typical Transmitter Circuits	3-14
Transceiver for Bidirectional Data Transfer	3-15
Handshake Modes	3-17
Handshake Configuration Examples	3-17
Introduction to the Handshake Timing Diagrams	3-19
Mode 0 - No Handshake	3-21
Mode 1 - Handshake Input	3-23
Using Mode 1 to Strobe Data into the Port	3-24
Peripheral Device Strobing Too Rapidly	3-25
Mode 2 - Handshake Output	3-27
Mode 3 - Strobe Output Handshake	3-29
Using Mode 3 for Partial Handshake Operation	3-30
Some Notes On Using The Parallel Interface Board	3-31
Inverting the Handshakes	3-31
Timeout Mechanism	3-32
Using Interrupts to Drive the Parallel Interface	3-33
FORTRAN Programs	3-34
4	
PARALLEL INTERFACE BOARD LIBRARY	
(PIBLIB.OBJ)	4-1
Introduction	4-3
Subroutine Summary	4-3
Parameters	4-4
Integer Equivalents of Binary Numbers	4-6
Example 1. Converting A Binary Digit to an Integer	
(Direction Mask)	4-6
Example 2. Converting an Integer to a Binary Number .	4-7
Event Interrupts	4-8
Program Examples	4-8
Subroutine Reference Pages	4-9
CHKBIT	4-9
CLRBIT	4-11
SETBIT	4-13
RDWRD	4-15
WTWRD	4-17
RDBLK	4-19
WTBLK	4-21
FRDBLK	4-23
FWTBLK	4-25
POPEN	4-27

PCLOSE	4-29
Error Messages	4-30

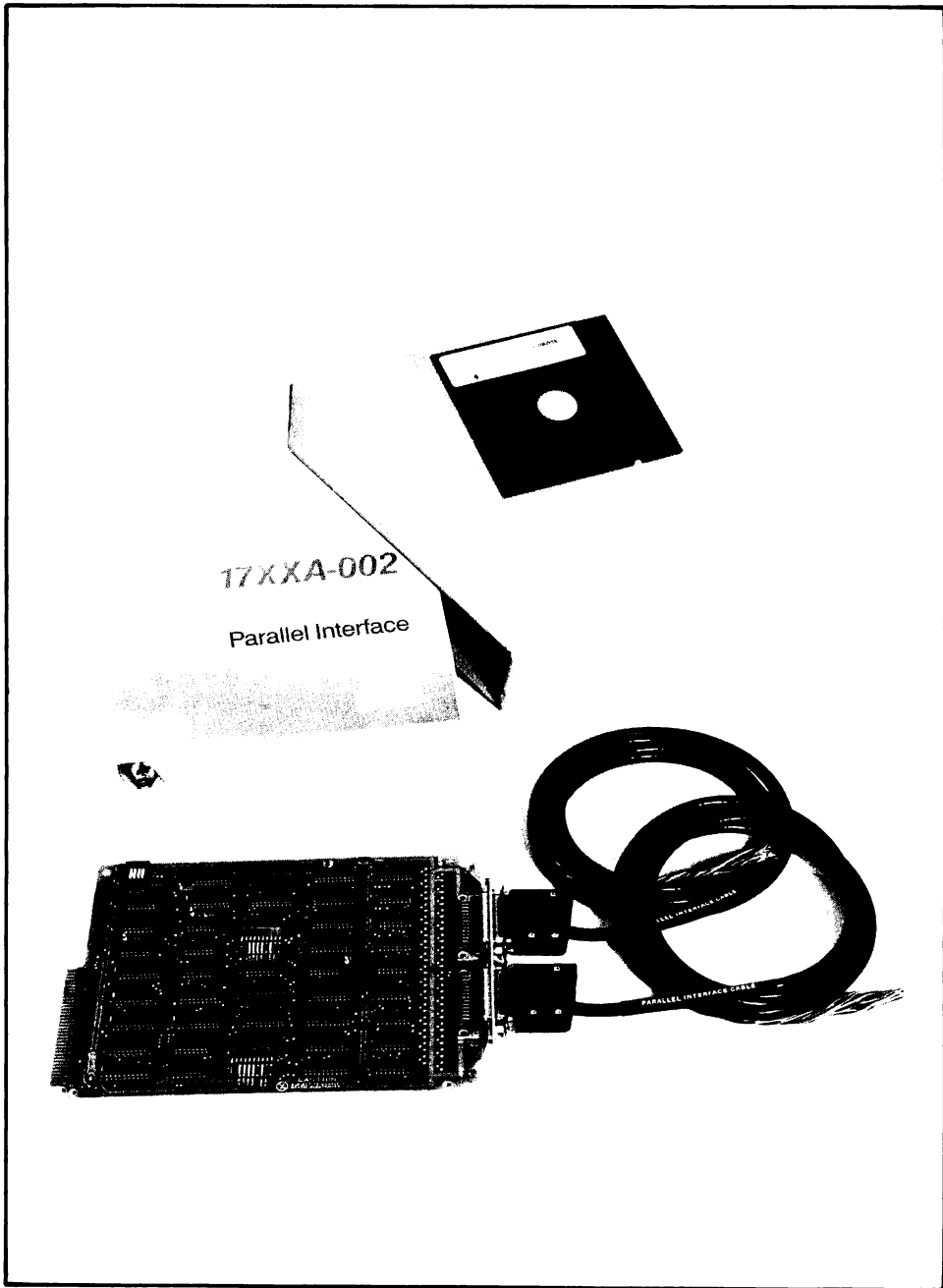
5 THEORY OF OPERATION 5-1

Introduction	5-3
Address Logic	5-3
Address Buffers	5-3
Address Switches and Decode Logic	5-3
Address Acknowledge Logic	5-4
Data Valid Flip-Flop	5-4
Register Decoder	5-4
Data Buffer and Interrupt Logic	5-4
Reset Buffer	5-4
Data, Status, and Control	5-5
Control Register	5-5
Status Register	5-6
Configuration Jumpers	5-7
State Machine	5-7
Interrupt Enable Logic	5-7
Data Transfer Port	5-7
Output Driver Enable	5-7
32-Bit Data Register	5-8
Data Bus Drivers	5-8

APPENDICES

A Specifications	A-1
B Interface Connector Pinout	B-1
C Sample BASIC Programs	C-1
D Test Points	D-1
E Signal Glossary	E-1
F Using the Parallel Interface With a 1720A Instrument Controller	F-1
G Performance Testing	G-1
H Schematics	H-1

INDEX



Parallel Interface

Section 1

How To Use This Manual

CONTENTS

Introduction	1-2
Organization	1-3
Usage Guide	1-4
Evaluators	1-4
Beginning Interface Designers	1-4
Experienced Interface Designers	1-4
Troubleshooters	1-5

INTRODUCTION

This manual is the primary reference for the Option 17XXA-002 Parallel Interface. The manual describes the hardware and software requirements for creating a customized parallel interface for Fluke Instrument Controllers.

The Parallel Interface manual provides a complete reference for effectively using the Parallel Interface Option. You do not need to be proficient in Assembly Language programming to program a parallel interface. Languages available are Interpreted BASIC, Compiled BASIC, FORTRAN, and TMS-99000 Assembly Language. You should be able to use timing diagrams; they illustrate the sequence of events required by hardware, and understanding them can assist you in using the software effectively.

ORGANIZATION

Section 1 How To Use This Manual

Describes the organization of the manual and how to use it.

Section 2 Installation

Describes how to install a parallel interface. This section contains unpacking instructions and general installation and setup procedures.

Section 3 Interface Description

This section describes the interface as a hardware device controlled by software. It gives the signal names for each pin on the connector and briefly discusses the timing constraints of each signal. The four operating modes are presented with timing diagrams and a complete description of the conditions occurring during the entire handshake.

Section 4 PIBLIB Routines

This section discusses software control of the interface. It begins with a description of how to access the PIB library (PIBLIB) from the high-level languages and Assembly Language. Examples illustrate the steps needed to get the module performing an I/O function. This section describes in detail each of the routines in the library and presents the syntax for BASIC, FORTRAN, and Assembly Language programs.

Section 5 Theory Of Operation

This section describes the architecture and hardware design of the Parallel Interface module. A block diagram serves as the focal point for a discussion of how the module works. Each functional circuit is briefly described.

Appendices

The appendices contain useful reference material, including the specifications of the module, the connector pinout, test points, example programs, and a glossary of signal names.

Index

An index is provided to assist you in locating topical references.

USAGE GUIDE

This guide is intended to assist potential PIB (Parallel Interface Board) designers and users to evaluate this manual and the Parallel Interface option.

Evaluators

If your task is to evaluate the option for a particular application, refer to Appendix A, Specifications, for a summary of the hardware and software that make up the Fluke Parallel Interface option. You might also review Sections 3, 4, and 5 for a more detailed look at the option's capabilities.

Beginning Interface Designers

If you have never designed a parallel interface, start with Section 2, Installation. That section contains preliminary setup instructions and shows the various configuration possibilities the PIB affords. Next, before going on to Sections 3 and 4, look over the contents of Appendix C, Sample Programs. These programs were all developed to meet real needs, and while they may have no direct applicability to your particular installation, they should serve as a source of ideas about how to proceed.

Experienced Interface Designers

Those who have designed a programmable parallel interface in the past may not need to use the sample programs in Appendix C, but will want to proceed directly to Section 3 or 4 for a detailed look at the hardware and software that make up the option. However, Appendix A, Specifications, is recommended first because it is a concise reference to the hardware and software. Also, Section 2 will help with your preliminary setup.

Troubleshooters

Depending on the level of troubleshooting to be done, there are several ways to proceed:

- If there is uncertainty about whether the module or a program is causing a failure, the System Diagnostic Software disk includes a test program called PIBTST that can be used to verify the operation of the hardware. You might also use one of the sample loop programs in Appendix C to ensure the ability of each port to read and write data and to do handshaking.
- If the PIB seems to work with some programs, but not others, the hardware rather than the program may be at fault. In this case, the theory of operation discussion in Section 5 may be helpful. This information breaks the PIB into logical blocks, and may help uncover the reason a particular program is not working.
- If the PIB is known not to operate, more detailed servicing information and the schematic are presented in the 1722A Service Manual, which can be ordered as Fluke P/N 732156.

In any of these cases, Appendix C lists two short test programs that can verify both ports. The test programs use a loopback connector that connects to both ports and allows the program to send data out one port and back into the other. The test cable can be ordered from FLuke, or it can be constructed easily. Appendix C tells how to order or build a test cable.

The Parallel Interface Board is covered by the Module Exchange Program. Consult your local Fluke Service Center for full details.

Section 2 Installation

CONTENTS

Introduction	2-2
Unpacking And Shipping Information	2-3
Installation	2-4
Configuration Procedure	2-5
Software Compatibility	2-7
Installation Procedure	2-9
Getting Started	2-11

INTRODUCTION

This section describes how to unpack and install the Option 17XXA-002 Parallel Interface. The installation procedures include information about:

- Verifying the configuration switches and jumpers.
- Checking for software compatibility.
- Physically installing the module into the Controller.

The section ends with a few tips on how to get started in designing and programming the interface.

UNPACKING AND SHIPPING INFORMATION

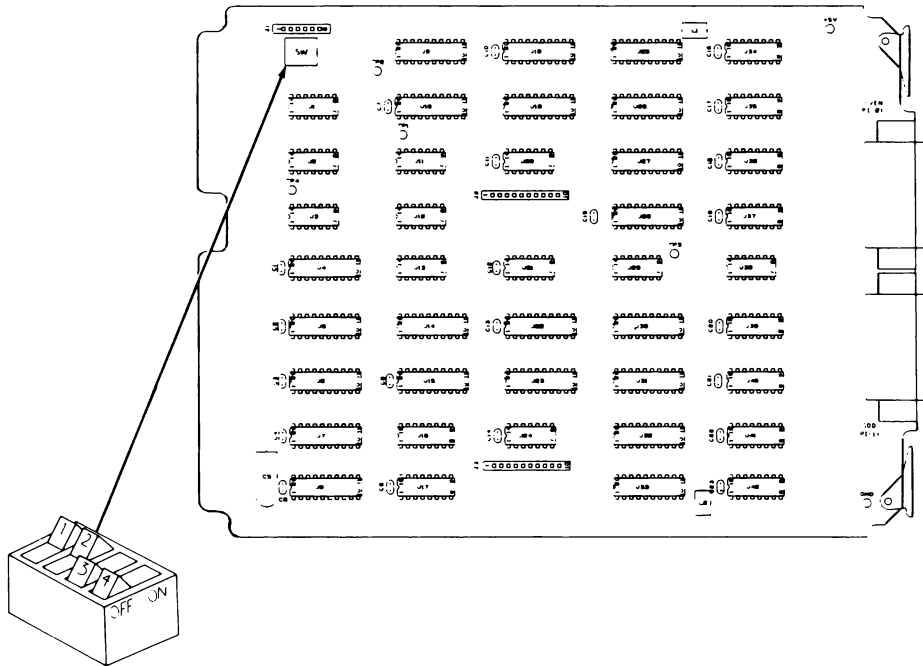
The Option 17XXA-002 Parallel Interface is packaged and shipped in a protective container. When you receive the option, make a thorough inspection for proper contents and possible shipping damage. Special instructions for inspection and claims are included with the shipping container. In the event the module must be shipped again later, use the original container. If the container is not available, a new one can be obtained from the John Fluke Mfg. Co., Inc. Please specify the instrument model number when requesting a new shipping container.

17XXA-002 Parallel Interface Option Contents

ITEM	JOHN FLUKE PART NUMBER
Parallel Interface Module	611947
Instruction Manual	732230
(2) Parallel Interface Cables	733907
1720A PIB Software Disk	630699

INSTALLATION

The instructions that follow describe how to install a Parallel Interface option in a 1722A Instrument Controller. Installation into a 1720A Controller is similar; the differences are described in Appendix F, "Using the Parallel Interface with a 1720A".



Configuration Procedure

The operating configuration of the Parallel Interface is set up by a board address switch and a configuration jumper block on the module. The information here describes the proper switch settings for various configurations. The Parallel Interface is shipped in the following operating configuration:

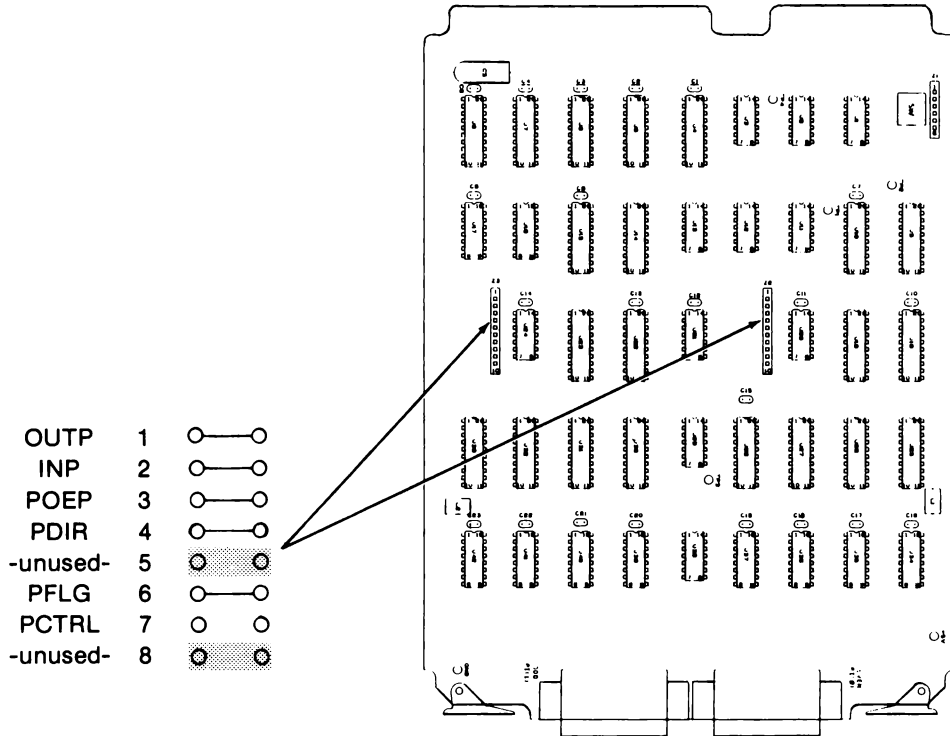
1. Set the address switch for ports PI0 and PI1. Use the table below to ensure that the switch is set properly for port addressability.

Switch/Address Operating Configuration

BOARD	SW4	SW3	SW2	SW1	PORT PI0	ADDRESS PI1	ADDRESS RANGE
1	N	ON	ON	ON	0	1	F340-F346
2	O	ON	ON	OFF	2	3	F348-F34E
3	T	ON	OFF	ON	4	5	F350-F356
4		ON	OFF	OFF	6	7	F358-F35E
5	U	OFF	ON	ON	8	9	F360-F366
6	S	OFF	ON	OFF	10	11	F368-F36E
7	E	OFF	OFF	ON	12	13	F370-F376
8	D	OFF	OFF	OFF	14	15	F378-F37E

Installation Configuration

- Each port also has a configuration jumper block, shown in the illustration below. The shipping configuration is identical for both ports, and is summarized in the table following the illustration.



This table indicates the meanings of each of the jumper positions. Notice that the shipping configuration results in these characteristics:

NAME	SHIPPING CONFIGURATION
OUTP	Jumpered = Normal data output.
INP	Jumpered = Normal data input.
POEP	Jumpered = POE HI enables outputs, LO disables them.
PDIRP	Jumpered = PDIR HI for all lines bidirectional, LO for output.
PFLGP	Jumpered = PFLG LO is asserted, HI not asserted.
PCTRLP	Open = PCTRL LO is asserted, HI not asserted.

3. If the shipping configuration is satisfactory, proceed to the Software Compatibility check. If you need to change the shipping configuration, use the table to select the desired operating configuration before proceeding.

SOFTWARE COMPATIBILITY

For the 1722A Instrument Controller, the Parallel Interface software is supplied on the 1722A System Disk. These machine language programs are interdependent and are compatible only in the combinations of versions supplied by Fluke and documented in published Fluke manuals. Using incompatible system software modules may give unpredictable results. In this case, Fluke cannot provide software support except for identification of compatible combinations.

NOTE

The floppy disk supplied with the Parallel Interface option contains software for operating the interface when it is installed in a 1720A Instrument Controller ONLY. The same routines for the 1722A Controller are already supplied on the System Disk that is shipped with the instrument.

There are some important (but easily overlooked) differences between the two sets of routines. For one thing, they are not interchangeable. This manual assumes that the Parallel Interface will be installed into a 1722A. If you are working with the 1720A Instrument Controller, be sure to refer to Appendix F for complete details on using the Parallel Interface with a 1720A Instrument Controller.

Software modules are easily erased and copied, so it is important to record modules onto a disk in the same combinations as they are supplied by Fluke. Place a Write-Protect tab on final disks as added insurance that no incompatible modules are accidentally recorded.

Check that the versions of the system software programs listed table below are the same. For example, all of them should be 1.0, 1.1, 1.2, and so forth. This manual supports system software modules up to and including Version 1.3.

These modules must have the same version numbers:

FDOS.SYS
TIME.FD2
SET.FD2
FUP.FD2
BASIC
PIBLIB.OBJ

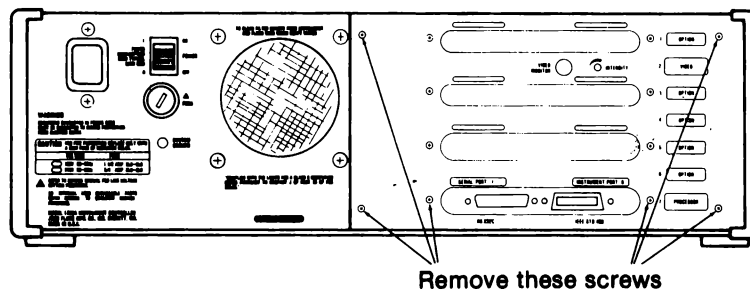
NOTE:

As subsequent software releases become available, all version numbers are incremented. If these releases are also supported by this manual, an addendum will be supplied that identifies the compatible versions and any functional changes and additions.

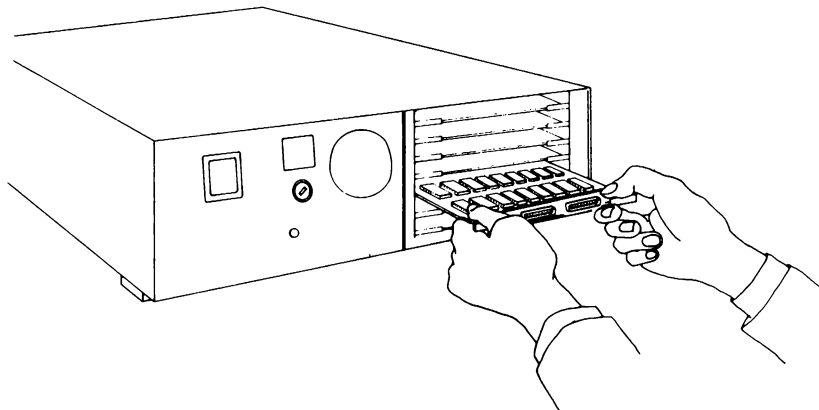
INSTALLATION PROCEDURE

After setting the switches and jumpers, and confirming software compatibility, follow these steps to install the Parallel Interface module in the 1722A Instrument Controller. Installation instructions for the 1720A are given in Appendix F.

1. Power down the Controller, and remove the line cord.
2. Remove the rear card cage cover, illustrated below. Depending on which modules are already installed, more screws than those indicated may also have to be removed. The six screws shown here are those that are removed from a new Controller with no options installed.



3. Carefully slide the option module into slot 1, 3, or 5 in the card cage. Make sure the module is fully seated so that it makes solid contact with the card-edge connector.



4. Remove the two screws holding the plate with the oval opening onto the card cage cover. After reinstalling the card cage cover, thread these screws through the card cage cover and into the shield plate surrounding the interface connectors.
5. Reinstall the card cage cover and the power cord.
6. Once installation is complete, power up the Controller and test the module by running the System Diagnostic "PIBTST". This program, which is part of the System Diagnostics software, verifies that the module is operating properly.

NOTE

For more rigorous testing, a loopback connector, John Fluke Part Number 632968, is available that connects the two ports and sends data between them. Listings for two loop-around programs are included in Appendix C.

7. Connect the plug of the Parallel Interface Cable to the connector on the Parallel Interface module. Connect the other end of the cable to the instrumentation system. Refer to Section 3, Interface Description, for complete information about the signals carried on each of the lines.
8. In case of any problems, recheck your work to ensure that switches are set properly and that the correct jumpers are in place. Be sure that the board is fully seated in the card cage. If everything is in order, but the failure continues, refer to the System Guide, Appendix G, System Diagnostics, for troubleshooting information, or call your local Fluke Service Center.

GETTING STARTED

Once the interface has been designed, it is a simple matter to use the PIBLIB routines in a program. Here are a few things to keep in mind:

- Early in the program, link to the library. In Interpreted BASIC programs, the line reads, LINK "PIBLIB". For other languages, consult the programming manual.
- Most of the routines can be used merely by giving the program name followed by any parameters or arguments required. However, using SETBIT in a BASIC program requires the "CALL" statement because of possible ambiguity with other BASIC "SET" statements like "SET SHELL".
- Although it isn't always necessary, make it a practice to use PCLOSE (Close a Port) prior to POPEN (Open a Port), as a housekeeping measure. No errors result from closing an already closed port, but errors are reported if you try to open a port that is already open.
- Be sure to include a CTRL/C handler in all programs. When <CTRL>/C is pressed, close the ports to ensure that no errors occur the next time you run the program.

The most important thing to keep in mind as you begin the design is to carefully outline the requirements of the intended system before beginning. This task breaks conveniently into two parts. First determine the direction, polarity, and level of the data lines, and of the handshake lines if they will be used. Begin software design only after the hardware requirements have been defined.

Finally, remember that the Option 17XXA-002 is an extremely versatile addition to the Fluke Instrument Controller. The PIB has the built-in ability to adapt to some of the most unusual interface requirements of the connected device(s).

Section 3

Interface Description

CONTENTS

Introduction	3-2
Hardware Interface Description	3-3
Port Lines	3-3
Port Signals	3-3
Port Control Jumpers	3-6
Notes on the Port Control Lines	3-8
Port Direction Line (PDIR)	3-8
Port Output Enable Line (POE)	3-10
Interface Circuits	3-11
Typical Receiver Circuit	3-14
Typical Transmitter Circuits	3-14
Transceiver for Bidirectional Data Transfer	3-15
Handshake Modes	3-17
Handshake Configuration Examples	3-17
Introduction to the Handshake Timing Diagrams	3-19
Mode 0 - No Handshake	3-21
Mode 1 - Handshake Input	3-23
Using Mode 1 to Strobe Data into the Port	3-24
Peripheral Device Strobing Too Rapidly	3-25
Mode 2 - Handshake Output	3-27
Mode 3 - Strobe Output Handshake	3-29
Using Mode 3 for Partial Handshake Operation	3-30
Some Notes On Using The Parallel Interface Board	3-31
Inverting the Handshakes	3-31
Timeout Mechanism	3-32
Using Interrupts to Drive the Parallel Interface	3-33
FORTRAN Programs	3-34

INTRODUCTION

The information in this section describes the hardware interface, the interface cable, and the software protocols. The hardware interface information defines in detail the port lines and illustrates a few typical handshake configurations and interfacing circuits.

The software protocol descriptions show the timing of each of the handshake modes in a reference page format. In most cases, timing is not crucial; this information is included to assist those who design systems where timing is a concern.

The section ends with some notes on using the timeout mechanism and interrupts. The software library is described in the next section.

HARDWARE INTERFACE DESCRIPTION

This section describes the hardware interface and the signals that the Parallel Interface sends and receives.

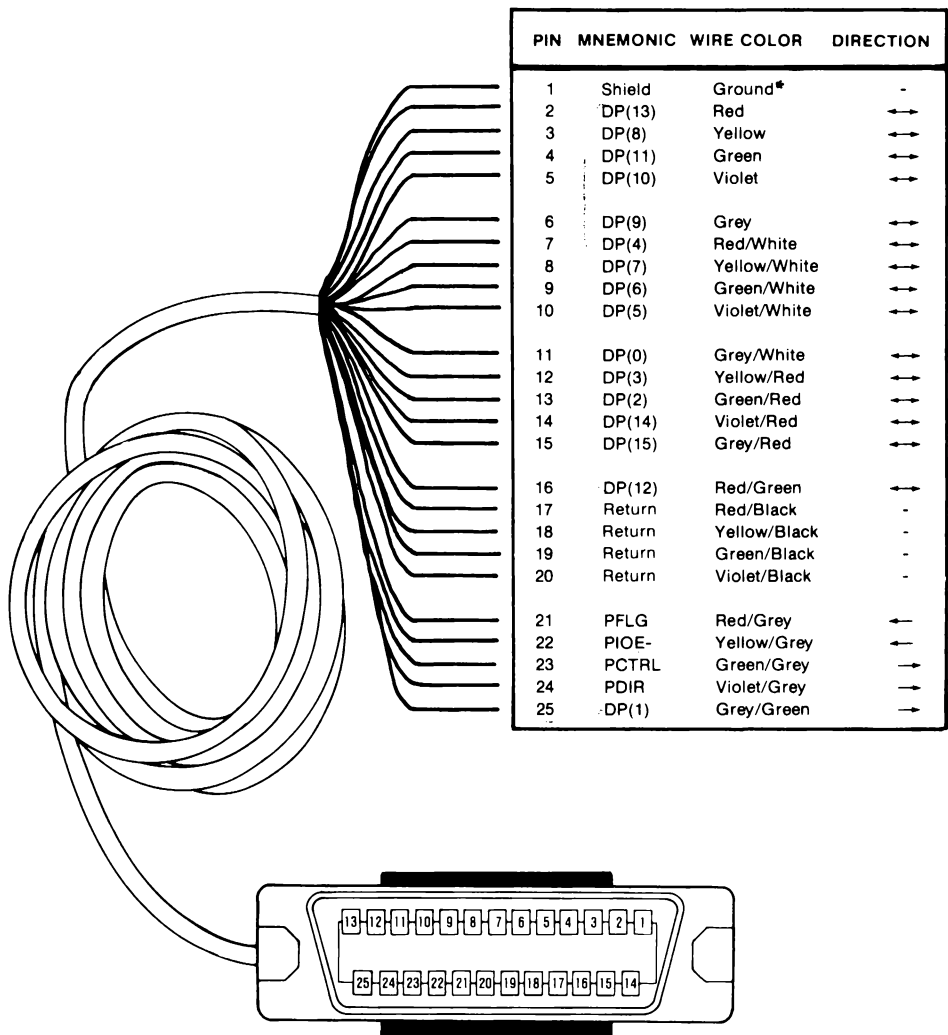
Port Lines

The port lines provide bidirectional data transmission between the Instrument Controller and compatible external devices. The drawing below shows the connector and lists the signal definition and logic states for each of the lines at the interface.

Port Signals

On the Parallel Interface module, several signals control the direction of data and the sense of the handshake lines. These signals, and the jumpers which can invert them, are defined in the following tables. Notice that by reconfiguring the jumpers, the handshake can be fully or partly inverted.

Interface Description



*Pin 1 is connected to the cable shield. This pin is not a reference point for signals. Its purpose is to connect the cable shield to a system ground. Make sure that use of this connection does not result in circular grounding paths (loops) through the system. Such ground loops can conduct enough current to interfere with data transmission.

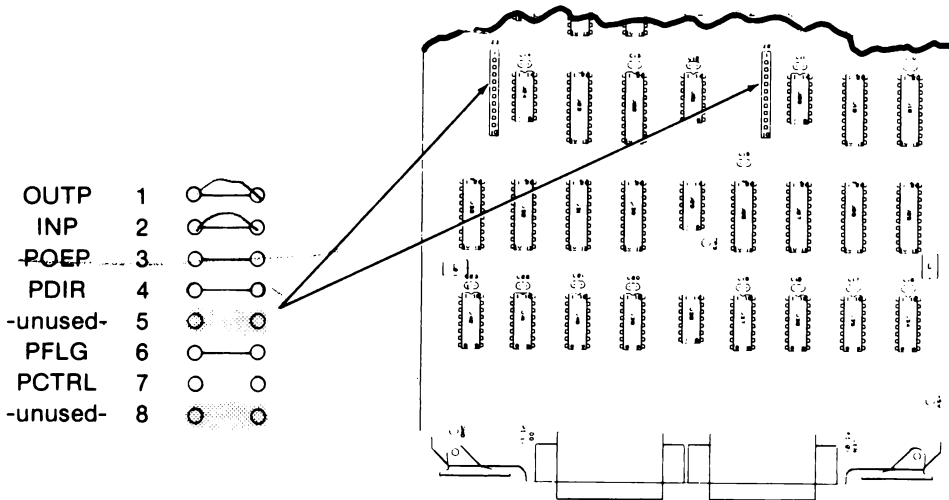
- ↔ indicates bidirectional data lines
- ← indicates input to PIB
- indicates output by PIB

Port Signal Definitions

SIGNAL NAME	DIRECTION	DEFINITION
PDIR	Output	Port Direction: Direction of data on port. Logic 1 = all lines bidirectional or input only Logic 0 = all lines output
POE	Input	Port Output Enable: Used by external device to disable the output from the ports. This is done typically if the device is sending data to the port. Logic 1 = outputs enabled Logic 0 = outputs disabled
PFLG	Input	Port Flag: One of the two handshake lines; the signal originates with the peripheral device. For input operations (device to port), PFLG, when asserted, indicates that input data is valid. For output operations, PFLG, when asserted, acknowledges that the device is reading the output data from the port.
PCTRL	Output	Port Control: One of the two handshake lines; the signal originates with the Parallel Interface module. For output operations (port to external device), PCTRL, when asserted, indicates that the output data is valid and the device may read it. For input operations, PCTRL, when asserted, indicates that the port is reading the incoming data from the device.
Data Lines	Bidirectional	Each data line can be used either as an input or as an output. For a full discussion, please refer to the discussion of the Direction Mask in Section 4.

Port Control Jumpers

Each port has a configuration jumper block, shown in the illustration below. The shipping configuration is identical for both ports, and is summarized in the table following the illustration.



Port Signal Definitions

JUMPER	DEFINITIONS
PCTRP	<p>Port Control Polarity: Controls the sense of PCTRL. When the jumper is removed (default):</p> <p>Logic I = PCTRL is not asserted Logic O = PCTRL is asserted</p> <p>When the jumper is in place:</p> <p>Logic I = PCTRL is asserted Logic O = PCTRL is not asserted</p>
INP	<p>Input Polarity: At the software's option, allows data to be inverted before it is input. When the jumper is in place (default), input data is not inverted. When the jumper position is open, the input data is inverted after being read.</p>

Interface Description Configuration Jumpers

OUTP	<p>Output Polarity: At the software's option, allows data to be inverted before output. When the jumper is in place (default), output data is not inverted; when the jumper position is open, the output data is inverted.</p>
PDIRP	<p>Port Direction Polarity: Inverts the sense of PDIR. When the jumper is in place (default), the sense of PDIR is defined as:</p> <p>Logic I = all lines bidirectional or input only Logic O = all lines are output only</p> <p>When the jumper is open, the sense of PDIR is defined as:</p> <p>Logic I = all lines are output only Logic O = all lines are bidirectional or input only</p> <p>Logic I = outputs enabled Logic O = outputs disabled</p>
POEP	<p>Port Output Enable Polarity: Inverts the sense of POE. When the jumper is in place (default), the sense of POE is defined as:</p> <p>Logic I = outputs enabled Logic O = outputs disabled</p> <p>When the POEP jumper position is open, the sense of POE is defined as:</p> <p>Logic I = outputs disabled Logic O = outputs enabled</p>
PFLGP	<p>Port Flag Polarity: Controls the sense of PFLG.</p> <p>When the jumper is in place (default):</p> <p>Logic I = PFLG is asserted Logic O = PFLG is not asserted</p> <p>When the jumper is removed:</p> <p>Logic I = PFLG is not asserted Logic O = PFLG is asserted</p>

Notes on the Port Control Lines

For this discussion, refer to the illustration on the next page. Each port consists of 16 data lines and 4 control lines. The control lines PCTRL and PFLG are used for handshake operation and are only active during Modes 1, 2, and 3 (not in Mode 0). The normal sense of these signals is active low. Their sense can be inverted by the PCTRP and PFLGP jumpers.

The other control lines, PDIR and POE (Port Direction and Port Output Enable) can be sensed or controlled by the external peripheral. When a port is opened using the POPEN command, the data lines are initially reset to logic high levels.

Port Direction Line (PDIR)

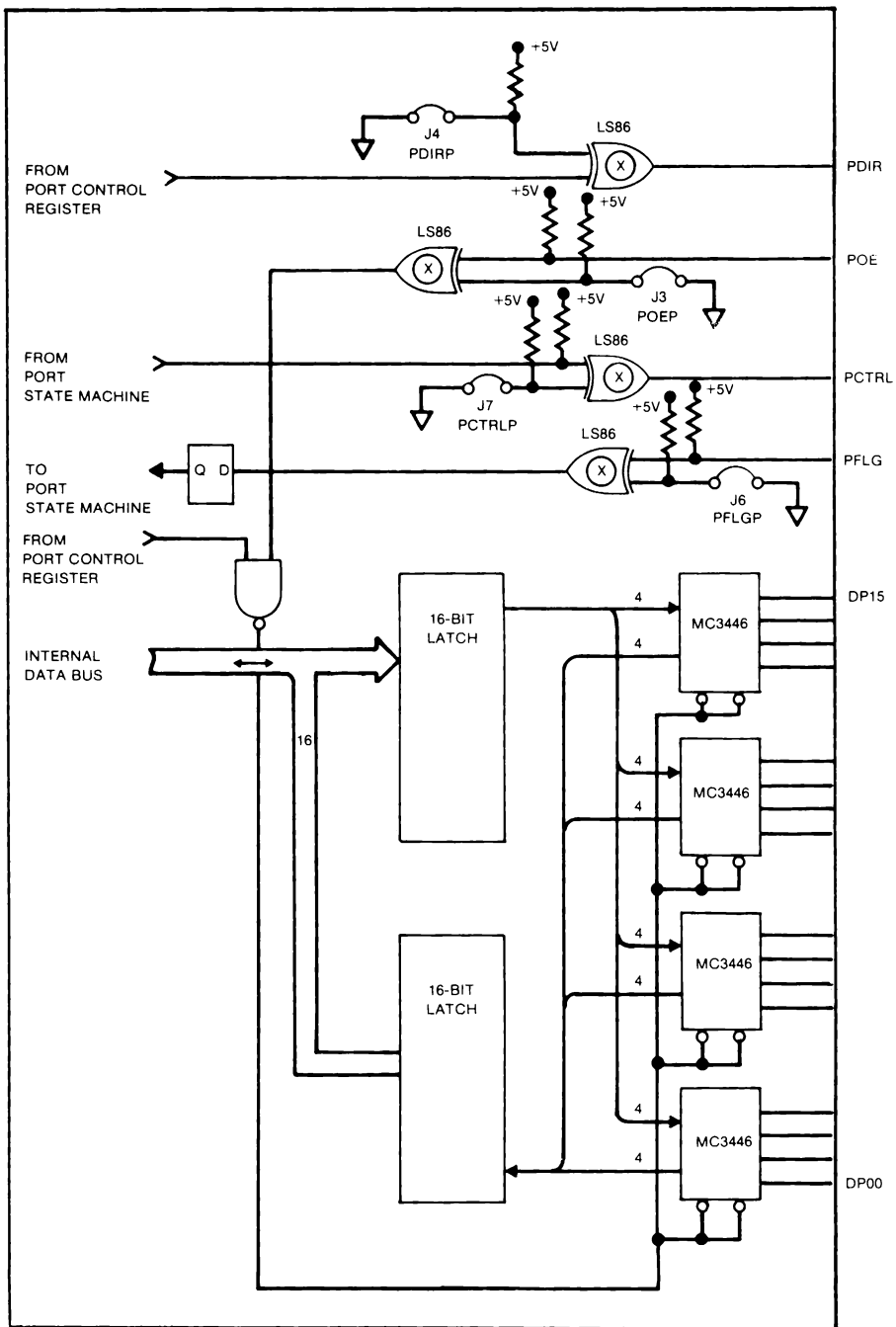
The Port Direction line is set to either a logic high or a logic low when the port is opened. If the port is opened in Modes 0 or 1, PDIR is set to a logic 1; if it is opened in Modes 2 or 3, the direction line is set to 0, and the polarity will not change unless the port is closed and reopened in another mode. See the table below. When the port is closed, PDIR is reset to 0.

HANDSHAKE MODE	PDIR
0, 1	1
2, 3	0

The handshake modes are described fully later in this section.

Interface Description

Port Control Lines



Port Output Enable Line (POE)

The Port Output Enable line enables and disables the output drivers. It is controllable by the external device. The MC3446A transceiver can be set to operate as a transparent buffer, or can be disabled. If it is disabled, the data lines are pulled to a logic high, the condition when the port is opened or closed. The MC3446A is not a tristate device. This means that the transceiver is never put into a high-impedance state; the port is always either sinking or sourcing current on the data lines.

The following table summarizes the actions the external device can affect on the transceivers by manipulating the POE line.

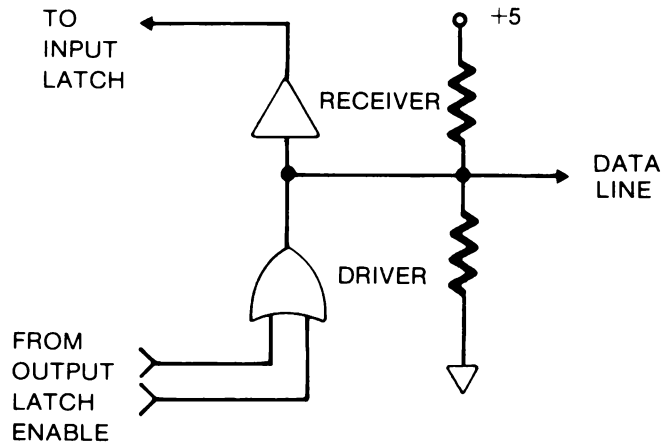
POE FROM EXTERNAL DEVICE	ACTION
Logic High	Enables outputs. Outputs reflect state of registered data.
Logic Low	Disable outputs. Outputs are pulled to a logic high.
No Connection	POE pulled to logic high; therefore, outputs are enabled.

INTERFACE CIRCUITS

The next few pages illustrate some typical interface circuits. Note that John Fluke Mfg. Co. does not guarantee these circuits for any particular application, and is not liable for their use. The component values shown are typical, and may have to be adjusted in the actual implementation.

The Parallel Interface data lines are driven by the Motorola MC3446A Transceiver. This component is compatible with RTL, DTL, TTL, and MOS devices. It also meets the requirements of the IEEE-488-1978 standard. Some critical electrical specifications of the MC3446A are illustrated on the next page.

MC 3446 DATA LINE INTERFACE CIRCUITRY

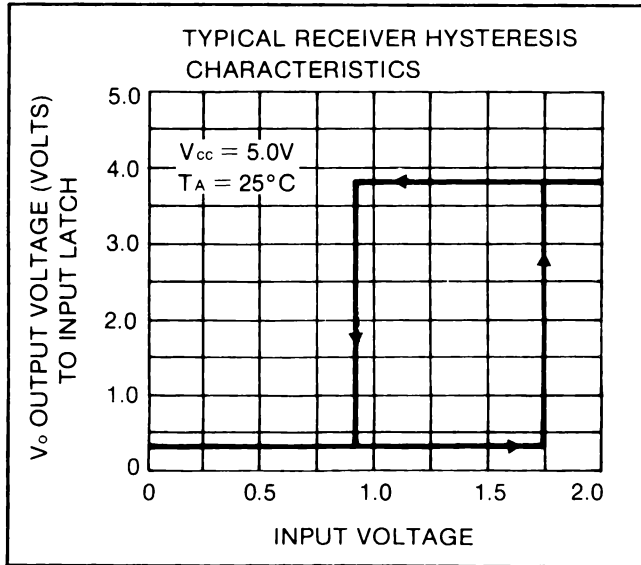


MAXIMUM RATINGS (T_A = 25°C unless otherwise noted.)

RATING	SYMBOL	VALUE	UNIT
Input Voltage	V _I	5.5	Vdc
Driver Output Current	I _{O(D)}	150	mA

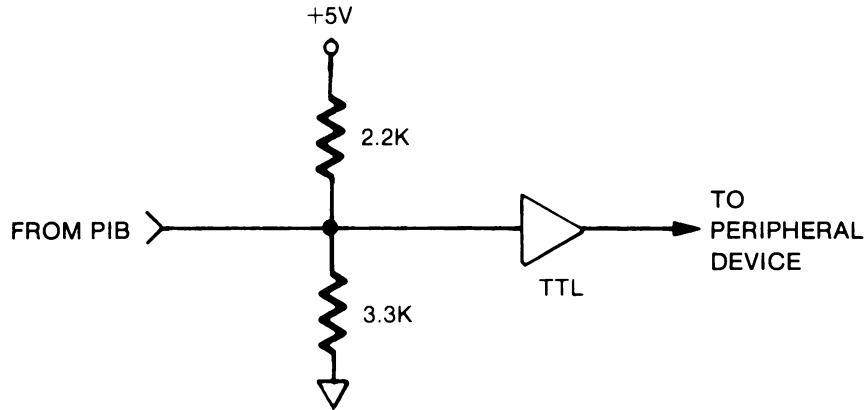
ELECTRICAL CHARACTERISTICS

CHARACTERISTIC	MIN	TYP	MAX	UNIT
DRIVER PORTION				
Input Voltage — High Logic State	2.0	—	—	V
Input Voltage — Low Logic State	—	—	0.8	V
Input Current — High Logic State (V _{IH} = 2.4V)	—	5.0	40	μA
Input Current — Low Logic State (V _{IL} = 0.4 V, V _{CC} = 5.0V, T _A = 25°C)	—	-0.2	-0.25	mA
Output Voltage — High Logic State (1) (V _{IH(S)} = 2.4V or V _{IH(D)} = 2.0V)	2.5	3.3	3.7	V
Output Voltage — Low Logic State (V _{IL(S)} = 0.8V, V _{IL(D)} = 0.8V, I _{OL(D)} = 48 mA)	—	—	0.5	V
RECEIVER PORTION				
Input Hysteresis	400	625	—	mV
Input Threshold Voltage — Low to High Output Logic State	—	1.66	2.0	V
Input Threshold Voltage — High to Low Output Logic State	0.8	1.03	—	V
Output Voltage — High Logic State (V _{IH(R)} = 2.0V, I _{OH(R)} = -400 μA)	2.4	—	—	V
Output Voltage — Low Logic State (V _{IL(R)} = 0.8V, I _{OL(R)} = 8.0 mA)	—	—	0.5	V



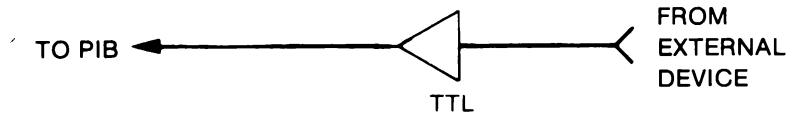
Typical Receiver Circuit

The connected device is receiving the signal(s) from the PIB and represents one TTL load.

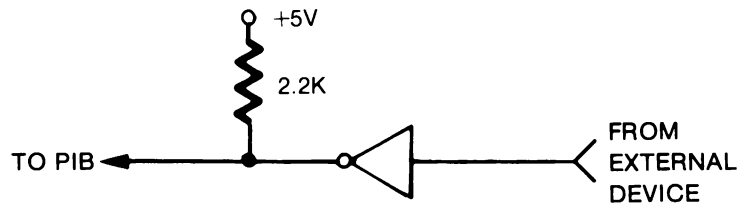


Typical Transmitter Circuits

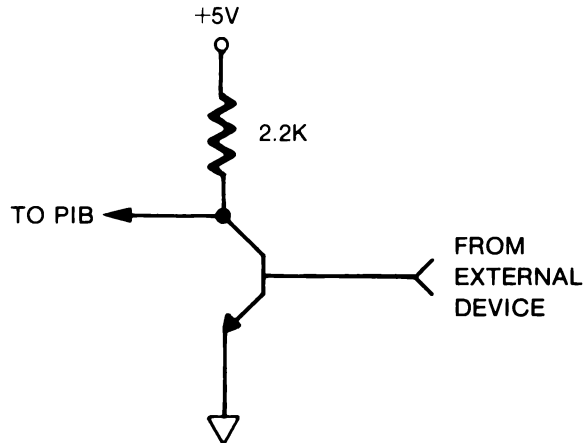
The external device is transmitting at TTL levels.



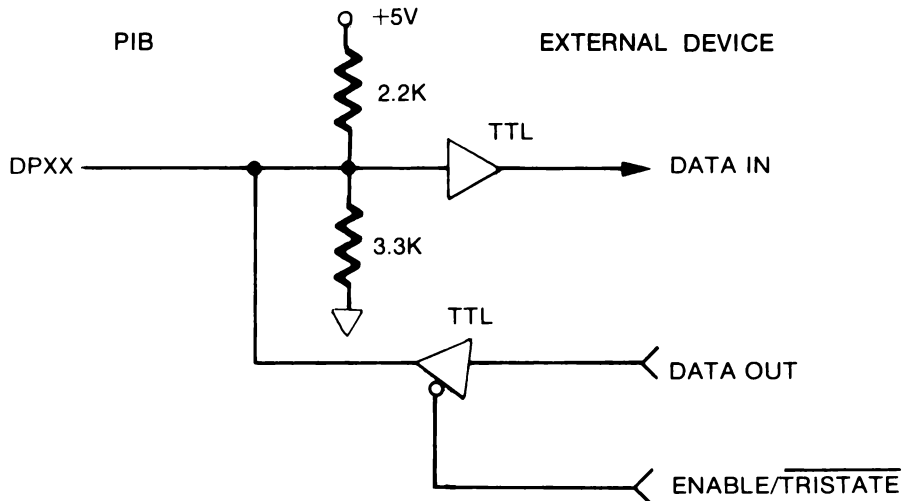
The peripheral device is transmitting using an open collector device.



The peripheral device has an undefined output level. This circuit ensures that the signal approximates the desirable TTL level of the PIB input. Remember that this circuit inverts the sense of the signal applied to the transistor's base.

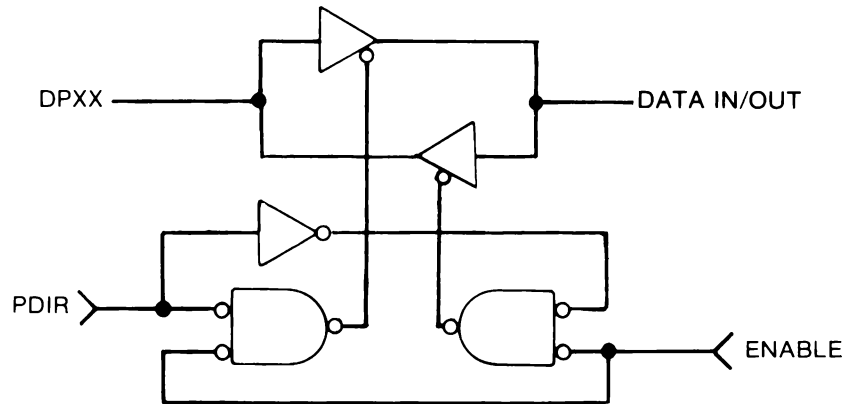


Transceiver for Bidirectional Data Transfer



Interface Description
Interface Curcuits

This transceiver circuit uses the Parallel Interface module's Port Direction signal FDIR and ENABLE signal at the external device to control tristating of the external device transmitter and receiver.



HANDSHAKE MODES

The Parallel Interface module operates in one of four modes: No Handshake, Full Handshake, and Strobe Input and Output. With the exception of No Handshake, the handshakes synchronize incoming and outgoing data. The table below lists the handshake names and the mode number that the software recognizes. Timing diagrams later in this section illustrate the timing requirements of each of the modes.

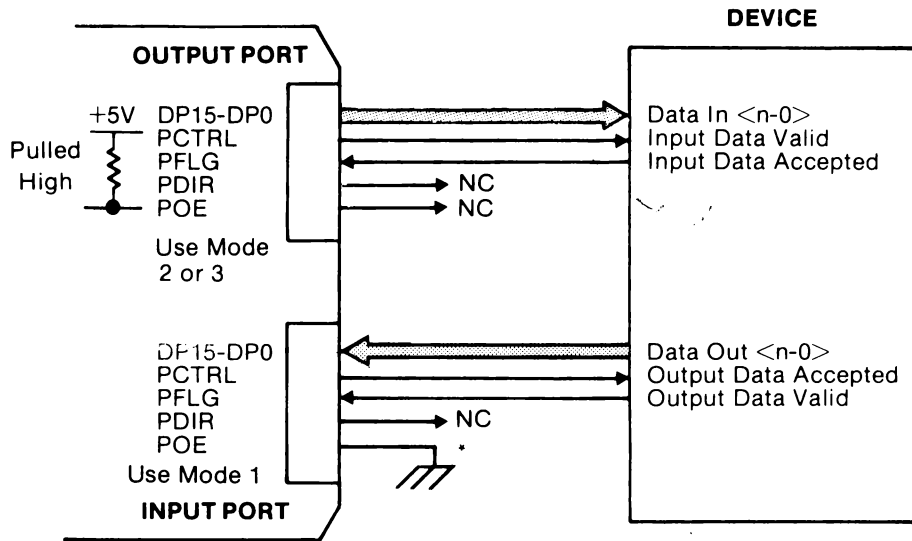
Handshake Modes		
NAME	DEFINITION	MODE
—	No Handshake	0
HNDSHKIN	Full Handshake Input	1
STROBEIN	Strobe Input	1
HNDSHKOUT	Full Handshake Output	2
STROBEOUT	Strobe Output	3

Handshake Configuration Examples

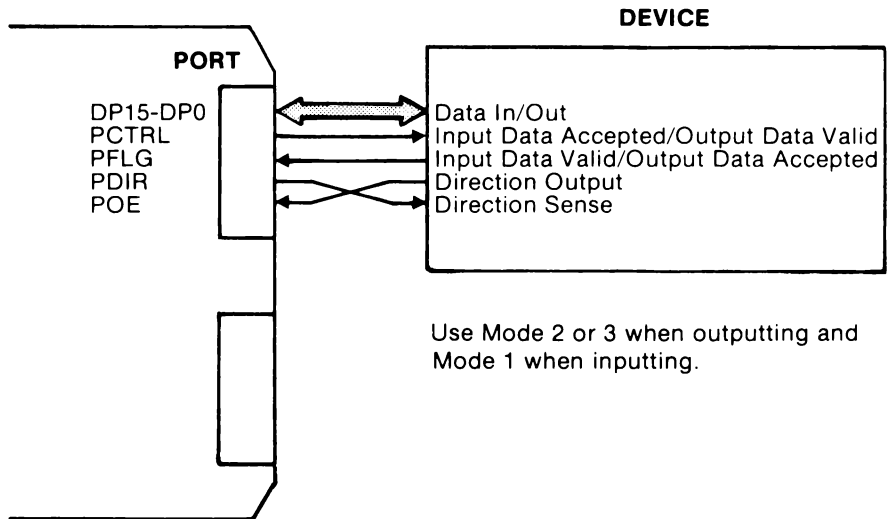
The drawings that follow illustrate two possible connection methods and show the modes that would be used for them.

29.

Interface Description
Handshake Examples



*POE is grounded to force the output devices off since this port is input only.



Introduction to the Handshake Timing Diagrams

The pages that follow describe the timing of each of the handshakes. Some of the terms used in these diagrams may not be familiar or in general use, so they are defined here:

- tr_{resp}* Response time, the time between an event and the response. This may be the time that the port takes to respond to a device initiating a handshake, or the time it takes the device to respond to a port-initiated handshake.
- t_{in}* Input time, the length of time that the port will hold the Handshake Line PCTRL active, or the length of time the external device holds PFLG active. During *t_{in}*, the CPU or the peripheral device is reading data.
- t_{rdy}* Ready time, the length of time until the port is ready after completion of the last event of the handshake.
- t_{strobe}* The pulse width of a pulse or strobe in Modes 2 and 3 (HNDSHKOUT and STROBEOUT). *t_{strobe}* is 200 ns minimum.
- t_w* Strobe width time. In Mode 3, the width of the pulse generated by PCTRL. It is 670 ns in duration.
- t_{ack}* Acknowledgement time; during a full handshake, the time that the receiving device (CPU or external peripheral) acknowledges that data has been transferred.

NO HANDSHAKE

Mode 0

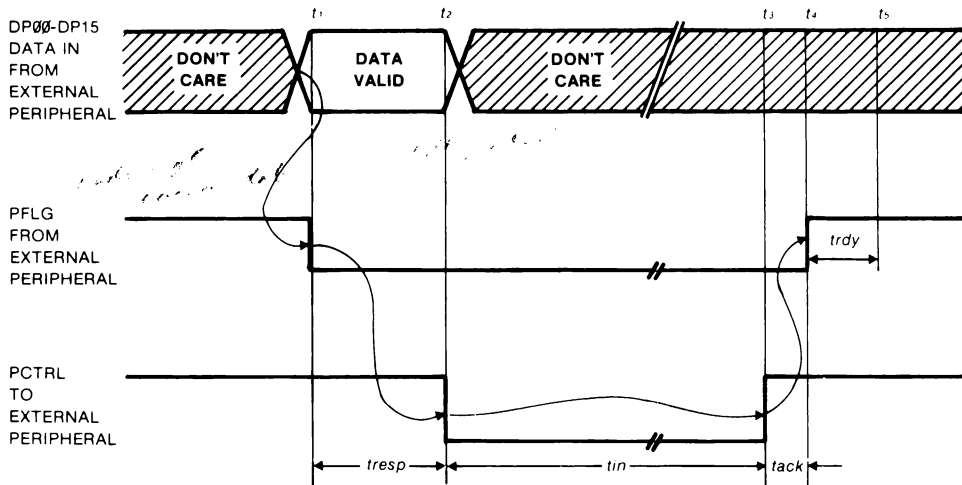
In the most simple mode, there is no handshaking, so no timing diagram is needed. The handshake lines remain idle, and the interface latches new data when the basic statement is executed.

In this mode of operation, the handshake lines PFLG and PCTRL are not asserted. When a call is made to a port in Mode 0, the data at the port is latched either in or out, depending upon whether the data is read from or written to the port.

The port must be opened in Mode 0. The port will not time out in Mode 0, and there will be no interrupts.

✓ HANDSHAKE INPUT

Mode 1



EVENT	DURATION
t_{resp}	670 ns maximum, no minimum.
t_{in}	24 μ s maximum, no minimum.
t_{ack}	no maximum.
t_{rdy}	200 ns maximum, no minimum.

- t_1 Data should be valid at the time that PFLG goes active.
- t_2 The port responds within 670 ns by making PCTRL active. This indicates that the data has been latched and that the CPU has been notified. However, note that the CPU has not yet read the data. Between times t_2 and t_3 , PCTRL remains active until the CPU reads the data from the latch.
- t_3 PCTRL is deasserted indicating to the peripheral device that the data has now been stored. The port will wait until PFLG has deasserted itself, if it hasn't yet done so.
- t_4 PFLG deasserts itself, and the port resets. The handshake is now complete.
- t_5 Since it has been reset, the port is now ready for another handshake.

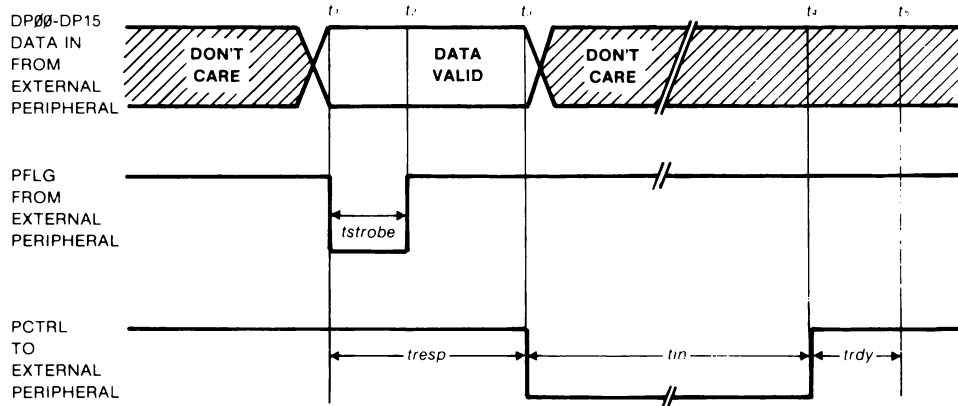
Note that unless the port is transferring data in the Block Mode, the next handshake will not begin until another call to the driver subroutine is made.

HANDSHAKE INPUT

Mode 1

Using Mode 1 to Strobe Data into the Port

It is not necessary to use the full handshake to transfer data to the port. The port also recognizes a pulse on the PFLG line. The minimum duration of the strobe is 200 ns. This method of using Mode 1 is illustrated below:



EVENT	DURATION
t_{strobe}	200 ns minimum.
t_{resp}	670 ns maximum.
t_{in}	24 μ s maximum, no minimum.
t_{rdy}	200 ns maximum, no minimum.

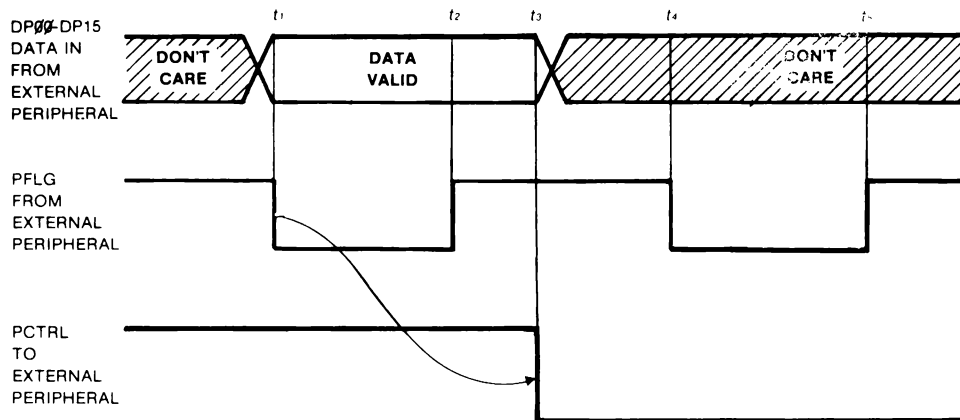
- t_1 The peripheral device has valid data, and therefore initiates the handshake by setting PFLG low. Since it is only strobing the port, it deasserts PFLG at time t_2 . In order for the port to recognize it, the length of strobe must be > 200 ns.
- t_2 The peripheral device deasserts PFLG. Data has been valid for at least 200 ns.
- t_3 The CPU has responded to the strobe and has latched the data.
- t_4 After t_{in} , the CPU has finished transferring the data, and PCTRL is deasserted. The port checks the status of PFLG to see that it was deasserted.

HANDSHAKE INPUT

Mode 1

Peripheral Device Strobing Too Rapidly

It is very important that the strobe rate of the peripheral device does not exceed the speed of the port because data can be lost. To ensure that PCTRL has been deasserted before attempting another data transfer, the external peripheral must either monitor the PCTRL line, or it must use an interval timer of at least t_{in} duration. Here is an example of a peripheral device attempting to send data too rapidly.

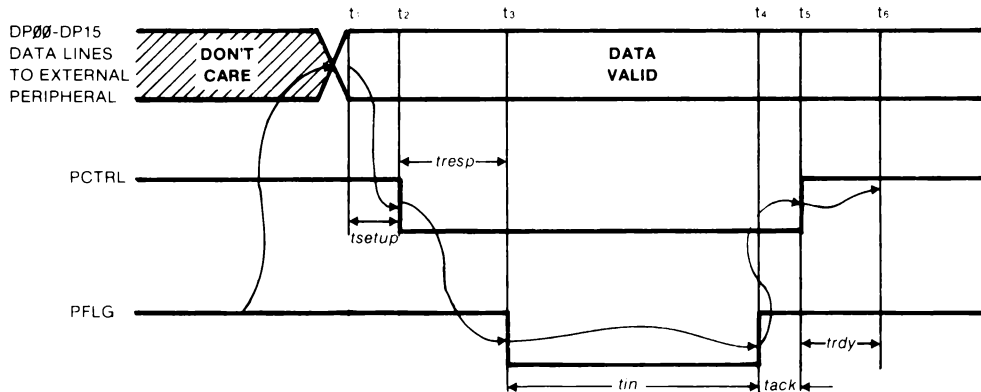


By strobing PFLG at t_1 , the peripheral device has notified the port that there is valid data. The port proceeds with the handshake and by time t_3 , the data has been latched.

However, the peripheral device doesn't wait for the CPU to read the latch and for the handshake to end with PCTRL being deasserted. Instead, the peripheral strobes PFLG again at time t_4 . The port is not available for data transfers while PCTRL is active, so it misses this new piece of data because the CPU is still busy attempting to read the original data.

HANDSHAKE OUTPUT

Mode 2



EVENT	DURATION
<i>tsetup</i>	200 ns maximum, no minimum.
<i>tresp</i>	Depends on external device; no maximum.
<i>tack</i>	200 ns maximum.
<i>tin</i>	Depends on external device; no maximum.
<i>trdy</i>	200 ns maximum.

- t1* Before registering valid data, the port checks the state of the PFLG line. If PFLG is inactive, the port proceeds to latch the valid data.
- t2* The port initiates the handshake at time *t2* by setting PCTRL low, indicating to the peripheral device that valid data is available and latched at the port.
- t3* From this point on, the port operates autonomously without CPU monitoring. The peripheral responds to the new state of PCTRL by setting PFLG low. This event may occur before *tresp* (~ 400 ns). However, the port only recognizes that PFLG has gone active no sooner than 400 ns after PCTRL has become active.

There is no maximum time period for *tresp*. The port continues to monitor the PFLG line until PFLG goes active. After response time *tresp*, the peripheral device holds PFLG active for at least the data input time (*tin*), ~ 200 ns in order for this signal to be recognized.

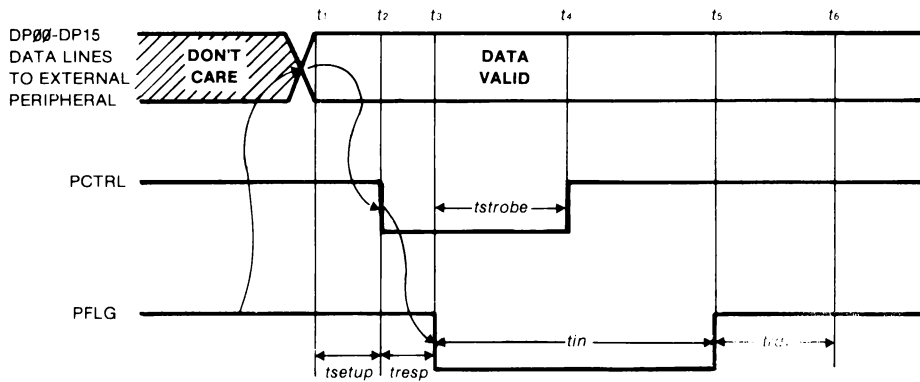
HANDSHAKE INPUT

Mode 2

- t4* The peripheral device allows PLFG to go inactive. Within 200 ns, the port makes PCTRL inactive at time *t5*.
- t6* The port is available again for handshaking no later than *trdy* at time *t6*, a maximum of 200 ns after *t5*. Note that this event only signifies the end of the handshake. Another handshake will only begin if the port is transferring blocks of data using WTBLK or FWTBLK. Otherwise, another transfer will occur when WTWRD is called from the calling program.

STROBE OUTPUT HANDSHAKE

Mode 3



EVENT	DURATION
t_{setup}	200 ns maximum, no minimum.
t_{resp}	Depends on external device; no maximum.
t_{strobe}	400 ns maximum.
t_{in}	Depends on external device; no maximum.
t_{rdy}	200 ns maximum.

- t_1 The port latches valid data.
- t_2 The port makes PCTRL active, indicating that new data has been latched.
- t_3 The peripheral device recognizes that the port has initiated a handshake, and responds by setting PFLG active.

PFLG may be brought active low before t_{resp} . However, it will not be recognized by the port sooner than 400 ns after PCTRL is first made low. Also, PFLG must remain low for at least 200 ns to guarantee that the port recognizes the new state.

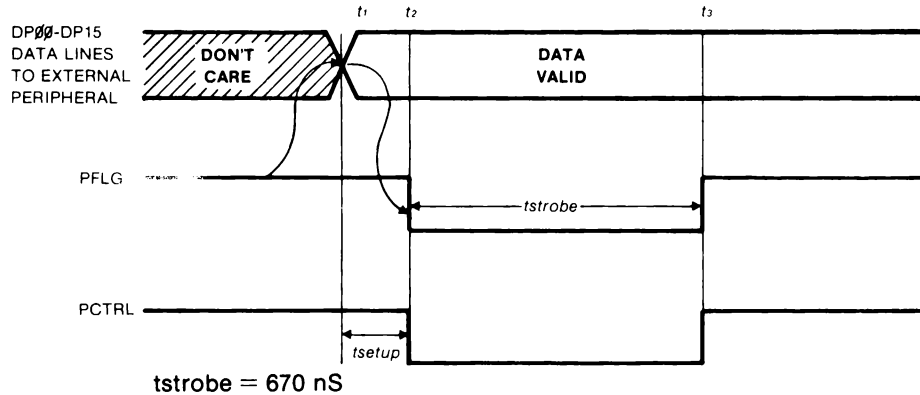
- t_4 PCTRL returns to its inactive state no later than 400 ns after PFLG becomes active. After it has finished reading data, the external peripheral resets the PFLG line to its inactive state. The port recognizes this action, and after t_{rdy} (200 ns) is again available for data transfer.

STROBE OUTPUT HANDSHAKE

Mode 3

Using Mode 3 for Partial Handshake Operation

By physically tying the PCTRL and PFLG lines to one another, the port can be configured to operate in a partial handshake mode.



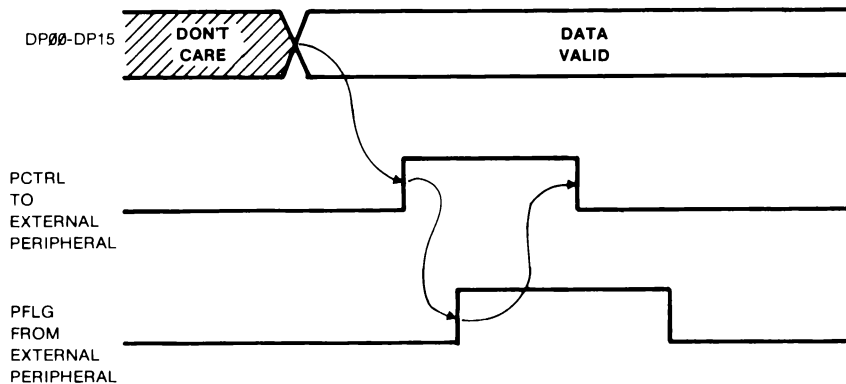
In this mode of operation, a strobe of 670 ns duration indicates to the peripheral device that valid data has been latched and is available. It is not a full handshake because PFLG is already active, and the port does not wait to receive it as the indication that the peripheral is ready.

SOME NOTES ON USING THE PARALLEL INTERFACE BOARD

Inverting the Handshakes

Before it is shipped, the Parallel Interface module is configured for the handshake lines PFLG and PCTRL to operate active low. This means that in their active states, these signals are at a logic low, and when they are inactive, they are at a logic high.

The handshakes can be configured by jumpers to operate either active high or active low. As an example, to configure Port 0 to handshake active high, cut the PFLGP jumper, and place a jumper on PCTRLP. If this is done, the timing diagram below illustrates the Mode 3 Strobeout active high handshake.



EXAMPLE OF INVERTED STROBE OUTPUT HANDSHAKE

Timeout Mechanism

The timeout mechanism provides a way to recover from an incomplete handshake after a time period specified by the software. The timeout mechanism operates differently for the different handshake modes. The timeout error is recoverable.

□ Mode 0

In Mode 0, No Handshake, the port never times out. Data is read from and written to the port regardless of the states of the handshake lines.

□ Mode 1

In Mode 1, Handshake Input, the port times out if the peripheral device does not initiate a handshake by making PFLG active within the specified timeout period.

□ Modes 2 and 3

When writing to a port in Mode 2 or Mode 3 (Handshake Output), the software first checks the port to determine whether the last handshake has been completed. The software continually checks the port until either the last handshake is complete (in which case a new handshake begins), or until the specified timeout period ends with the last handshake still incomplete. If the second case occurs, a timeout error is reported.

Using Interrupts to Drive the Parallel Interface

When working with slow peripheral devices, it is possible to use the Parallel Interface module in an interrupt-driven mode. Doing so requires that the Port Mode parameter be set up properly. For more information, see the discussion of parameters in Section 4. A port-to-port loop program using interrupts is shown in Appendix C, Sample Programs.

In the interrupt-driven method of operation, the Instrument Controller can be freed to do other tasks while it is collecting data from the Parallel Interface in a “background” process, thus allowing some degree of parallel processing.

To operate in the interrupt-driven mode, use the BASIC statement

```
ON PPORT GOTO (linenumber)
```

The OFF PPORT statement disables interrupts from the PIB. For complete information on how to use these statements in a BASIC program, see the BASIC Reference Manual, reference entries 96 (ON PPORT), and 84 (OFF PPORT).

In addition, the mode parameter of the POPEN statement must be set appropriately. See the discussion on parameters in Section 4.

In Mode 1, when the PIB senses that PFLG has been asserted, it latches the data and simultaneously notifies the CPU via interrupt that new data is available. When the BASIC Interpreter is notified, it finishes executing the current command and jumps to the interrupt handler subroutine in this fashion:

```
100  | Interrupt Handler for P.I.B.  
105  |  
110  | RDWRD (parameters)  
120  |  
130  |   -- code for processing data --  
140  |  
150  |  
160  | RESUME      ! return to main program
```

Interrupts can be used in Mode 2 or Mode 3 to indicate the end of a handshake. After outputting a data word to a port, the driver software does not wait for the handshake to complete. Instead, it returns to the next BASIC statement. The interrupt can be used to indicate that another data transfer can be made.

FORTRAN Programs

FORTRAN does not handle interrupts. Therefore, the PIB cannot be used in the interrupt-driven mode if the Controller's programs are written in FORTRAN. This is one of the reasons that a FORTRAN program executes so quickly. A FORTRAN program does not have the time-consuming overhead necessary for interrupt handling.

Section 4

Parallel Interface Board Library (PIBLIB.OBJ)

CONTENTS

Introduction	4-3
Subroutine Summary	4-3
Parameters	4-4
Integer Equivalents of Binary Numbers	4-6
Example 1. Converting A Binary Digit to an Integer (Direction Mask)	4-6
Example 2. Converting an Integer to a Binary Number .	4-7
Event Interrupts	4-8
Program Examples	4-8
Subroutine Reference Pages	4-9
CHKBIT	4-9
CLRBIT	4-11
SETBIT	4-13
RDWRD	4-15
WTWRD	4-17
RDBLK	4-19
WTBLK	4-21
FRDBLK	4-23
FWTBLK	4-25
POPEN	4-27
PCLOSE	4-29
Error Messages	4-30

INTRODUCTION

This section presents each of the subroutines in the 17XXA-002 Parallel Interface Board Library (PIBLIB.OBJ). The subroutines are alphabetically arranged and described in a reference page format. Error messages that might be returned for each routine are included as a part of the reference page, and are also given in a table at the end of this section.

Subroutine Summary

This table summarizes each of the routines in the Parallel Interface Board Library. A full discussion with program examples can be found in reference pages at the end of this section. All parameters are integers.

Summary of PIBLIB Routines

SUBROUTINE	DESCRIPTION	PARAMETERS
CHKBIT	Check a bit	Port, Bit, Bool
CLRBIT	Clear a bit on a port	Port, Bit
SETBIT	Set a bit on a port	Port, Bit
RDWORD	Read word at a port	Port, Word
WTWORD	Write a word to a port	Port, Word
RDBLK	Read to a port from an array	Port, Block, Count
WTBLK	Write to a port from an array	Port, Block, Count
FRDBLK	Same as RDBLK, but faster	Port, Block, Count
FWTBLK	Same as WTBLK, but faster	Port, Block, Count
POPEN	Open a port	Port, Mode, Mask, Timeout
PCLOSE	Close a port	Port

Parameters

When the routines are used in a program, one or more parameters are specified by the programmer. All parameters must be specified as integers. The reference page for each routine includes the required parameters, which are defined below.

PORT The port number for the routine to operate on, expressed as an integer in the range 0 - 15. Use the table below to assign port numbers to any number of Parallel Interface modules.

BOARD	PORTS
0	0, 1
1	2, 3
2	4, 5
3	6, 7
4	8, 9
5	10, 11
6	12, 13
7	14, 15

MODE The mode parameter consists of two one-byte parameters in a 16-bit integer. The high order byte determines whether or not interrupts from the interface will be acknowledged by the Controller (using the ON PPORT command in BASIC, for example). The low order byte determines the mode of operation.

High Order Byte: set to one (1) to enable interrupts from the PIB, and zero (0) to disable them. This is easily accomplished in decimal by adding 256 to the low order byte. This example opens Port 0 in the No Handshake mode with interrupts enabled, all bits input, and a timeout of 2 seconds:

POPEN (0%, 256% + 1%, -1%, 200%)

Lower Order Byte: indicates the mode of operation as shown below. For more information, see Section 3.

VALUE	NAME	DEFINITION
0	No Handshake	Bidirectional input and output with no handshaking. PCTRL and PFLG remain inactive.
1	Hndshkin	Handshake input. Handshaking under control of PCTRL and PFLG.
2	Hndshkout	Handshake output. Handshaking under control of PCTRL and PFLG.
3	Strobeout	Strobe handshake output.

TIME OUT The wait time before an incomplete handshake is terminated. Timeout is a programmable integer in the range 0 - 32767; each count is one 10-millisecond "tick". The longest timeout, 32767 x 10 ms = 328 seconds, or about five and a half minutes. Specifying 0 disables timeout.

BIT The bit number to be checked, read, or written to, expressed as an integer. The least significant bit is 0 and the most significant bit is 15.

BOOL When a bit or word is read or checked, as in the CHKBIT routine, the routine returns the value into a variable specified as **BOOL**. This value is an integer in the range of 0 to 32768, corresponding to $bool = 2^n$, where n is the bit position. This variable can be used in Boolean expressions, because its value will always be either zero or non-zero.

DIRECTION MASK An integer that indicates the desired transmission direction of each bit on the port. The Direction Mask is the decimal equivalent of the two's complement of the binary number. Input lines are encoded as 1-bits and outputs as 0-bits. If the port is to be used for output only, the direction mask parameter would be set to 0. For input on all lines, use -1. The Direction Mask insures that an input bit is not driven as an output bit.

For more detail, Example 2 (following) shows how to convert a binary word to a decimal integer.

- BLOCK** This parameter is the array to which data will be transferred using the Block subroutines (RDBLK, WTBLK, FRDBLK, and FWTBLK.)
- COUNT** Indicates how many array elements to transfer.

Integer Equivalents of Binary Numbers

During programming the Parallel Interface, it is often necessary to convert a 16-bit binary word to a decimal integer, and vice versa. Use the table below to decode BOOL and to assist in calculating the integer to be used for various direction masks.

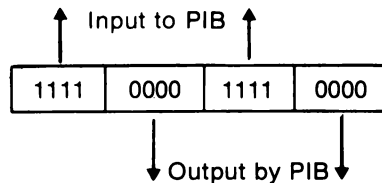
2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
32768	16384	8192	4096	2048	1024	512	256	128	64	32	16	8	4	2	1

NOTE

While it is true that the value of 2^{15} is 32768, bit 15 is never assigned that value. Instead, bit 15 is used as a sign bit, because the allowable range of integers is -32768 to $+32767$ inclusive. Whenever bit 15 is a 1-bit, it indicates that the rest of the binary word must be encoded or decoded as the two's complement of the actual number. See the following example for more details.

Example 1. Converting A Binary Digit to an Integer (Direction Mask)

The port below is grouped into four sets of four lines each. The data direction alternates for each set:



Each input line is a 1-bit, so the binary value of the required integer is 1111000011110000. Since bit 15 is to be a 1, take the two's complement (invert all bits, add 1, and make the result negative). Note that the binary number can be encoded directly if the MSB is a zero.

1111 0000 1111 0000

(-) 0000 1111 0000 1111 (+ 1)

which encodes into the integer as follows:

$$- [1 + 2 + 4 + 8 + 256 + 512 + 1024 + 2048 (+1)] = \mathbf{-3656}$$

Example 2. Converting an Integer to a Binary Number

This conversion has many uses when you are working with the PIBLIB subroutines. It is used mainly when reading or writing a word at a port. Assume that RDWRD has just read the value of the word at a port to be -21846, and has placed that value into R%, the integer that was assigned to receive the returned value.

There are several options open to the programmer: first, the program might simply display the value, permitting it to be decoded manually (either by successive division by two, or by successive subtraction by powers of two, using the table above). The more efficient way, however, is to have the program itself do the decoding. Fluke BASIC has an easy way to do this conversion: the RAD\$ function.

The program:

```

100 PRINT "ENTER A DECIMAL NUMBER: "; \ INPUT R%
120 B$ = RAD$(R%, 2%)
130 BB$ = DUPL$("0", 16-LEN(B$)) + B$
140 PRINT "Decimal          Binary"
150 PRINT R%, BB$

```

would print the following if the number -21846 were entered:

DECIMAL	BINARY
-21846	10101010101010

To better understand how all this works, take a few moments to type in this short routine and plug a few numbers in.

Event Interrupts

The Parallel Interface Module can generate an interrupt to the CPU. The types and priorities of all interrupts are listed below.

Priority	Interrupt Type
1	ON ERROR
1	ON <CTRL>/C
2	ON #n (RS-232-C Channel)
3	ON KEY (the Touch-Sensitive Display)
4	ON PORT
5	ON PPORT (Parallel Interface Interrupt)
6	ON SRQ (IEEE-488 Bus instrument service request)
7	ON PPOL (IEEE-488 Bus parallel poll)
8	ON CLOCK
9	ON INTERVAL Interrupts

For more information, refer to the 1722A BASIC Reference Manual, Reference Entry 96, ON PPORT.

Program Examples

Each subroutine is illustrated in program examples. Note that these program fragments can be made part of a larger program. All examples are shown as they would appear in an Interpreted BASIC program. Several complete sample programs appear in Appendix C.

Usage

BASIC: [CALL] CHKBIT (port%, bit%, bool%)

FORTRAN: CALL CHKBIT (port, bit, bool, error)

Assembly Language:

```
ref chkbit
blwp @chkbit
data 3
data port
data bit
data *bool
```

Description

This routine checks a particular bit on a Parallel Interface Port. It is equivalent to reading the port and then isolating the specified bit position.

- The CHKBIT routine is often used in the No Handshake mode but can also be used with the other modes.
- If the port was opened with a Handshake Input mode, then CHKBIT initiates a handshake.
- The routine checks the DIPOL bit of the Status Register. If the INP jumper is open, then the DIPOL bit will be logic high, and data will be inverted by the CPU.

Parameters

- port The number of the port to be opened, expressed as an integer.
- bit The bit to be checked (0 though 15), expressed as an integer.
- bool The returned value of the bit at the position, expressed as an integer in the range 0-2¹⁵. The value can be used in a Boolean expression since it will either be zero or not zero.

See Also

SETBIT and CLRBIT

CHKBIT

PIBLIB

Example

This example uses chkbit to loop in a portion of a program until the value of bit 0 is true.

```

:
:
370 FOR IX = 1% TO 15%
380   CLRBIT (PT%, IX) \ WAIT 50   ! clear and set each bit
390   SETBIT (PT%, IX)           ! in order
400 NEXT IX
410 CHECKBIT (PT%, 0%, RV%)      ! return bit 0 value to RV%
420 IF RV% THEN 370             ! when set, do it again
:
:
```

Possible Error Messages

1201 Port Not Available
1204 Port Not Opened
1207 Port Timed Out
1220 Illegal Port Number
1222 Illegal Bit Value

Usage


BASIC: [CALL] CLRBIT (port%, bit%)





FORTRAN: CALL CLRBIT (port, bit, error)

Assembly Language:

```
ref clrbit
blwp @clrbit
data 2
data port
data bit
```

Description

 When the OOTP jumper is in the shipping position, (in place), this routine clears to zero a particular bit on a selected port. It is equivalent to reading the port, ANDing the bit with 0 in the appropriate position, and then writing the data word back out to the port.

-  This routine is used mainly in the No Handshake mode, but can also be used with the other output modes.
-  The bit must be opened as an output.
-  If the port was opened with a Handshake Output mode then CLRBIT initiates a handshake.
-  The routine checks the DOPOL bit of the Status Register. If OOTP is open, then the DOPOL bit will be logic high, and all data bits will be inverted by the CPU.

Parameters

port The number of the port to be opened, expressed as an integer.

bit The bit to be checked (0 though 15), expressed as an integer.

See Also

SETBIT and CHKBIT

CLRBIT PIBLIB

Example

This example uses CLRBIT and SETBIT to sequentially turn all the bits off and on.

```
110 POPEN (0%, 0%, 0%, 0%)  
120 FOR I% = 0% TO 15%  
130   CLRBIT (0%, I%) \ WAIT 50  
140   SETBIT (0%, I%)  
150 NEXT I%
```

Possible Error Messages

- 1201 Port Not Available
- 1204 Port Not Opened
- 1205 Attempted Write to a Read-Only Bit
- 1207 Port Timed Out
- 1220 Illegal Port Number
- 1222 Illegal Bit Value

Usage

BASIC: [CALL] SETBIT (port%, bit%)

FORTRAN: CALL SETBIT (port, bit, error)

Assembly Language:

```
ref setbit  
blwp @setbit  
data 2  
data port  
data bit
```

Description

This routine sets a particular bit (output = high) on a selected port. It is equivalent to reading the data latched at the port, ORing the bit in the specified position with 1, and then writing the data word back out to the port.

- This routine is used mainly in the No Handshake mode, but can also be used with the other output modes.
- The bit must be opened as an output.
- If the port was opened with a handshake output mode, then the output of the word after the bit has been set starts a new handshake.
- The routine checks the DOPOL bit of the Status Register. If the OUTP jumper is open, then the DOPOL bit will be logic high, and all data bits will be inverted by the CPU.

Parameters

port The number of the port to be opened, expressed as an integer.

bit The bit to be checked (0 though 15), expressed as an integer.

See Also

CLRBIT and CHKBIT

SETBIT PIBLIB

Example

This example uses CLRBIT and SETBIT to sequentially turn all the bits off and on.

```
      .  
110 POPEN (0%, 0%, 0%, 0%)  
120 FOR I% = 0% TO 15%  
130   CLRBIT (0%, I%) \ WAIT 50  
140   SETBIT (0%, I%)  
150 NEXT I%  
      .  
      .
```

Possible Error Messages

1201 Port Not Available
1204 Port Not Opened
1205 Attempted Write to a Read-Only Bit
1207 Port Timed Out
1220 Illegal Port Number
1222 Illegal Bit Value

Usage

BASIC: [CALL] RDWRD (port%, bool%)

FORTRAN: [CALL] RDWRD (port, bool, error)

Assembly Language:

```
ref rdwrd
blwp @rdwrd
data 2
data port
data *bool
```

Description

This routine accepts an integer value from the specified port and writes it into 'BOOL'. The following features are associated with RDWRD:

- The routine checks the DIPOL bit of the Status Register. If the INP jumper is open, then the DIPOL bit will be logic high, and data will be inverted by the CPU.
- Output data is masked with the Direction Mask that was specified when the port was opened, ensuring that no output drivers are turned on for bits defined as inputs.
- If the port mode is HNDSHKIN, HNDSHKOUT, or STROBEOUT, then the routine waits until the port obtains the next piece of data, thus synchronizing the software to the hardware.
- If the port is unable to handshake the data in, then the software causes a timeout error (recoverable).

Parameters

port The port to be opened, expressed as an integer.

bool The integer where the word value of the port will be written to.

See Also

WTWRD, RDBLK, FRDBLK

RDWRD PIBLIB

Example

An ATE program is learning the characteristics of a module. Under varying conditions, it reads the port and stores the word values into an integer array which is then recorded on floppy disk. Later, the elements of this array can be compared to the values obtained from a Unit Under Test.

Possible Error Messages

1201 Port Not Available
1204 Port Not Opened
1207 Port Timed Out on Handshake

Usage

BASIC: [CALL] WTWRD (port%, word%)

FORTRAN: CALL WTWRD (port, word, error)

Assembly Language:

```
ref wtwrd
blwp @wtwrd
data 2
data port
data word
```

Description

This routine writes a word out to a specified port. The routine has several features:

- WTWRD checks the DOPOL bit of the Status Register. If the OUTP jumper is open, then the DOPOL bit will be logic high, and data will be inverted by the CPU.
- The output data is masked (ORed) with the Direction Mask that was specified when the port was opened. The routine outputs up to 16 bits of data. Masking ensures that no output driver is turned on for bits defined as inputs.
- If HNDSHKIN, HNDSHKOUT, or STROBEOUT is the port mode, then the routine waits until the port becomes available for the next piece of data. This synchronizes the software to the hardware of the port.
- WTWRD may initiate a handshake and hang up if the peripheral device does not complete the handshake.

Parameters

port The port number, expressed as an integer.

word The integer value of the word to be written to the port.

See Also

RDWRD

WTWRD PIBLIB

Possible Error Messages

1201 Port Not Available
1204 Port Not Opened
1205 Attempted Write to a Read-Only Bit
1207 Port Timed Out
1220 Illegal Port Number

Usage

BASIC: [CALL] RDBLK (port%, block%, count%)

FORTRAN: CALL RDBLK (port, block, count, error)

Assembly Language:

```
ref rdblk
blwp @rdblk
data 3
data port
data block
data count
```

Description

This routine reads a block of data from a port.

- Input is assumed to be all 16 bits.
- The routine checks the DIPOL bit of the Status Register. If the INP jumper is open, then the DIPOL bit will be logic high, and data will be inverted by the CPU.
- If HNDSHKIN is the port mode, then the routine waits until the port becomes available before reading the next block.
- Data is input until the number of words specified in the “COUNT” argument are transferred.

Parameters

- port The number of the port to be opened, expressed as an integer.
- block An integer array pointer. The block read is placed into the array starting at the array element indicated.
- count The number of array elements to be transferred, expressed as an integer.

RDBLK PIBLIB

See Also

WTBLK, FRDBLK, and FWTBLK

Possible Error Messages

1201 Port Not Available
1204 Port Not Opened
1220 Illegal Port Number

Usage

BASIC: [CALL] WTBLK (port%, block%, count%)

FORTRAN: CALL WTBLK (port, block, count, error)

Assembly Language:

```
ref wtblk
blwp @wtblk
data 3
data port
data block
data count
```

Description

This routine writes a block of data to a port.

- Output is assumed to be all 16 bits.
- WTBLK checks the DOPOL bit of the Status Register. If the OUTP jumper is open, then the DOPOL bit will be logic high, and data will be inverted by the CPU.
- If HNDSHKOUT or STROBEOUT are the port mode, then the routine waits until the port becomes available before sending the next block, thus synchronizing the software to the port's state machine.
- Data is output until the number of words specified in the "COUNT" argument are transferred.

Parameters

- port The value of port (0 or 1) to be opened, expressed as an integer.
- block The block to be written to the port is sent from an integer array. The BLOCK argument points to the first element in the array.
- count The number of array elements to be transferred, expressed as an integer.

WTBLK

PIBLIB

See Also

RDBLK, FRDBLK, and FWTBLK

Possible Error Messages

1201 Port Not Available
1204 Port Not Opened
1205 Attempted Write to a Read-Only Bit
1220 Illegal Port Number

Usage

BASIC: [CALL] FRDBLK (port%, block%, count%)

FORTRAN: CALL FRDBLK (port, block, count, error)

Assembly Language:

```
ref frdblk
blwp @frdblk
data 3
data port
data block
data count
```

Description

This routine reads a block of data from a port, as fast as possible. This means that it does not perform the error checking that is done in a normal read of a block using RDBLK. Like the Fast Write Block routine, FWTBLK, the Fast Read Block routine also ignores the Direction Mask that was set when the port was opened and does not check the states of the INP or OUTP jumpers.

- Input is assumed to be all 16 bits. The routine ignores the Direction Mask that was set when the port was opened.
- The DIPOL bit in the Status Register is ignored, so input data is never inverted.
- If HNDCHKIN is the port mode, then the routine waits until the port becomes available before reading the next block.
- Data is input until the number of words specified in the “COUNT” argument are transferred.
- The objective of this routine is to maximize the transfer rate of the port. However, only use FRDBLK when RDBLK is not fast enough.

CAUTION

This routine will not time out. If the external device does not send data, the routine may stop attempting to read the port, and yet not give an error message.

FRDBLK

PIBLIB

Parameters

- port The number of the port to be opened, expressed as an integer.
- block An integer array pointer. The block read is placed into the array starting at the array element indicated.
- count The number of array elements to be transferred, expressed as an integer.

See Also

FWTBLK, RDBLK, RDWRD

Possible Error Messages

1201 Port Not Available
1204 Port Not Opened
1220 Illegal Port Number

Usage

BASIC: [CALL] FWTBLK (port%, block%, count%)

FORTRAN: CALL FWTBLK (port, block, count, error)

Assembly Language:

```
ref fwtblk
blwp @fwtblk
data 3
data port
data block
data count
```

Description

This routine writes a block of data to a port as fast as possible. This means that it does not perform the error checking that is done in a normal block write using WTBLK. Like the Fast Read Block routine, FRDBLK, the Fast Write Block routine also ignores the Direction Mask that was set when the port was opened and does not check the INP or OUTP jumpers.

- Output is assumed to be all 16 bits. The routine ignores the Direction Mask that was set when the port was opened.
- The DOPOL bit in the Status Register is ignored, so output data is never inverted.
- This routine will not time out. If the external device does not respond, FWTBLK may stop attempting to write to the port and yet return no error message (it will “hang”).
- If HNDCHKOUT or STROBEOUT are the port mode, then the routine waits until the port becomes available before sending the next block, thus synchronizing the software to the port’s state machine.
- Data is output until the number of words specified in the “COUNT” argument are transferred.
- The objective of this routine is to maximize the transfer rate of the port. However, only use FWTBLK when WTBLK is not fast enough.

FWTBLK PIBLIB

CAUTION

Do not use FWTBLK if the port mode is HNDSHKIN.

Parameters

- port The value of port (0 or 1) to be opened, expressed as an integer.
- block The block to be written to the port is sent from an integer array.
 The BLOCK argument points to the first element in the array.
- count The number of array elements to be transferred, expressed as
 an integer.

See Also

FRDBLK, WTBLK, WTWDR

Possible Error Messages

- 1201 Port Not Available
- 1204 Port Not Opened
- 1205 Attempted Write to a Read-Only Bit
- 1220 Illegal Port Number

Usage

BASIC: [CALL] POPEN (port%, mode%, mask%, timeout%)

FORTRAN: CALL POPEN (port, mode, mask, timeout, error)

Assembly Language:

```
ref popen
blwp @popen
data 4
data port
data mode
data mask
data timeout
```

Description

POPEN opens a port in preparation for data transfer at the interface.

NOTE

Always use POPEN before attempting to read from or write to the port. Not doing so will return Error 1208, Port Not Open.

Parameters

- | | |
|---------|---|
| port | The port number expressed as an integer. |
| mode | The mode (0, 1, 2, or 3) that indicates the Control Mode to be used for handshaking. |
| mask | The Direction Mask integer that indicates the direction of data transfer. |
| timeout | The length of time that the port waits before terminating an unsuccessful handshake. The integer is expressed in terms of 10 millisecond "ticks". For a 10-second wait, the integer would be 1000. Use the argument 0 (zero) for immediate timeout. |

See Also

PCLOSE

POPEN PIBLIB

Examples

To open Port 0 in Strobe Output mode, interrupts disabled, all bits set as outputs, and timeout disabled:

```
POPEN (0%, 3%, -1%, 0%)
```

To open Port 0 in Handshake Input mode, interrupts enabled, all bits output, and a 1-second timeout:

```
POPEN (0%, 256% + 1%, 0%, 100%)
```

Possible Error Messages

1201 Port Not Available
1203 Port Already Open
1220 Illegal Port Number
1222 Illegal Mode

Usage

BASIC: [CALL] PCLOSE (port%)

FORTTRAN: CALL PCLOSE (port, error)

Assembly Language:

```
ref pclose
blwp @pclose
data 1
data port
```

Description

PCLOSE closes the specified port and returns the hardware to a passive state.

- The port is left in a condition similar to that of the power-up condition; PDIR is set to allow the outputs of any external device to be on.
- It is not necessary that the port be open to use PCLOSE. Because of this, PCLOSE can be used to put the port into a known state, all data lines pulled high.

Parameters

port The port to be closed, expressed as an integer.

See Also

POPEN

Example

To close Port 0:

```
PCLOSE (0%)
```

Possible Error Messages

1201 Port Not Available
1220 Illegal Port Number

ERROR MESSAGES

This table gives all possible error messages that can be returned by the PIBLIB routines. Notice that these are the error numbers returned by BASIC programs. For FORTRAN, the error numbers are the last two digits only. All Parallel Interface errors are recoverable.

ERROR	MEANING	SOLUTION
1200	PIB Software Drivers Not Linked with FDOS.	Use the System Generation Utility program to link in PIB drivers.
1201	Port Not Available	The specified port has not been assigned by the address switch on the module. See Section 2, Installation, to set the switch.
1202	Illegal Function Call	The library does not contain the specified function. Look for typing errors, or an obsolete PIBLIB function (like PSETUP).
1203	Port Already Opened	Port was not closed before it was opened by the program which generated the error. Always use PCLOSE before POPEN as a housekeeping measure.
1204	Port Not Opened	An attempt was made to read or write at a port that has not yet been opened. There is probably no POPEN statement in the program, or the wrong port was opened.
1205	Attempted Write to a Read-Only Bit	Both of these errors may result from an incorrectly set Direction Mask, or from improper setup of the INP or OUTP jumpers.
1206	Attempted Read from a Write-Only Bit	
1207	Port Timed Out	See the discussion on timeout in Section 4.
1220	Illegal Port Number	A port number greater than 15 was specified by the program.
1221	Illegal Mode	A mode parameter less than zero or greater than three was specified.
1222	Illegal Bit Value	A bit greater than 15 was specified.

Section 5

Theory of Operation

CONTENTS

Introduction	5-3
Address Logic	5-3
Address Buffers	5-3
Address Switches and Decode Logic	5-3
Address Acknowledge Logic	5-4
Data Valid Flip-Flop	5-4
Register Decoder	5-4
Data Buffer and Interrupt Logic	5-4
Reset Buffer	5-4
Data, Status, and Control	5-5
Control Register	5-5
Status Register	5-6
Configuration Jumpers	5-7
State Machine	5-7
Interrupt Enable Logic	5-7
Data Transfer Port	5-7
Output Driver Enable	5-7
32-Bit Data Register	5-8
Data Bus Drivers	5-8

INTRODUCTION

The information here is expected to assist the Parallel Interface user in designing and programming an interface. The descriptions refer to each of the blocks on the PIB block diagram on the last page.

ADDRESS LOGIC

The block in the upper left of the drawing monitors the system address bus to determine if the PIB is being selected. Address lines ADR5-3 select the board, and ADR2-1 in combination with R/W select the operation.

Address Buffers

This block buffers the entire 16-bit address and these signals:

ADVAL- Address Valid
 BUSWR- Bus Read/Write
 RINT- Refresh Interrupt

Address Switches and Decode Logic

The states of switches S1, S2, and S3 are compared to the incoming address lines A3, A4, and A5, respectively. If the switches match the states of these lines, the board is selected. The table below shows board addresses and associated switch settings.

Switch/Address Operating Configuration

BOARD	SW4	SW3	SW2	SW1	PORT PI0	ADDRESS PI1	ADDRESS RANGE
1	N	ON	ON	ON	0	1	F340-F346
2	O	ON	ON	OFF	2	3	F348-F34E
3	T	ON	OFF	ON	4	5	F350-F356
4		ON	OFF	OFF	6	7	F358-F35E
5	U	OFF	ON	ON	8	9	F360-F366
6	S	OFF	ON	OFF	10	11	F368-F36E
7	E	OFF	OFF	ON	12	13	F370-F376
8	D	OFF	OFF	OFF	14	15	F378-F37E

Address Acknowledge Logic

If the board is selected, it notifies the CPU via the signal ADACK- that a valid address has been received and that the board has been selected.

Data Valid Flip-Flop

The board-select signal BRDSLCT- is the input to a D flip-flop. The next high-going clock pulse SYNC sends the data valid signal DVAL- to the Register Decoder.

Register Decoder

When DVAL- is active low, a 3-to-8 multiplexer reads address lines ADR1 and ADR2 and the Read/Write line to select one of the four registers at one of the two ports. The four registers are the same for each of the two ports. They are the input and output data registers and the control and status registers.

DATA BUFFER AND INTERRUPT LOGIC

This block, located at the left center of the diagram, sends interrupt signals to the system control bus based on an interrupt from either of the two ports. The Data Buffer is a bidirectional buffer for incoming and outgoing data.

RESET BUFFER

The signal DCOK indicates that the power supply is stable. When DCOK is true, the Reset Buffer resets the various registers, buffers, and flip-flops on the board.

DATA, STATUS, AND CONTROL

The Data, Status, and Control block and the Data Transfer block are identical for both ports except for the addresses and signal names.

Control Register

Data lines DI15-DI8 are latched at the Control Register, which generates the port control signals. These signals are briefly described below. In general, they control the direction of data and direct the mode of the state machine.

CONTROL REGISTER															
PORT 1								PORT 0							
15	14	13	12	11	10	9	8	15	14	13	12	11	10	9	8
			X	X							X	X			

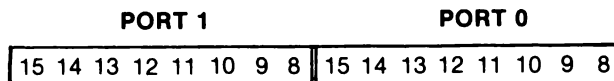
Bit	Signal Name	Meaning	Explanation
15	INTEN	Interrupt Enable	Allows interrupts to occur for this port.
14	OUTEN	Output Enable	A low level enables the output drivers; a high level disables them.
13	SANITY-	State Machine Reset	A low on this control line resets the state machine for that port to a known state.
12	X	Don't Care	
11	X	Don't Care	
10	PDIR	Port Direction	The level on this line indicates data input or output, depending on the setting of the configuration switch.
9, 8	MODE	Handshake Mode	These two bits select the mode that the state machine is to operate in. There are two mode bits and, therefore, four possible modes. See the following table.

Handshake Modes vs Control Register Bits

BIT 9 8	MODE	NAME
0 0	0	No Handshake
0 1	1	HANDSHKIN
1 0	2	HNDSHKOUT
1 1	3	STROBEOUT

Status Register

Receives port status information and sends it by way of the data bus to the CPU.



Bit	Signal Name	Meaning
15	READY	Port is ready for next transfer
14	INTO-	Port 0 Interrupt Request
13st(1) 12 st(0)		The value of current state.
11	DOPOL	Data Output Polarity, indicates polarity of output data to software.
10	DIPOL	Data Input Polarity, indicates polarity of input data to software.
9 MODE0 8 MODE1		Indicates in what mode the port is operating.

Configuration Jumpers

Each of the two ports has six configuration jumpers. Their placement defines the polarity of:

- Input and output data (INP and OUTP)
- Port Output Enable and Port Direction (POEP and PDIRP)
- The handshaking lines (PCTRP and PFLGP)

State Machine

The state machine controls the functions of the port. The state machine is made up of a 512 by 8-bit PROM, and an 8-bit register. The PROM is addressed by control signals from the Control and Status Registers. The PROM's output data is latched into the register, and four of the register's outputs (ST0 - ST3) are fed back to the input of the PROM, resulting in seven possible states using four state-variables. Two of the other four outputs are combined with other on-board control signals to generate the interrupt and port control signals for the port. The two PROM output signals LO and LI clock the data registers.

Interrupt Enable Logic

Based on the current state of the state machine, and on the state of the signal INTEN, this block either enables or disables interrupts. When INTEN is low, interrupts are disabled, and when it is high, they are enabled. Interrupt status occupies one bit in the status register. Note that INTEN is sent to the ports on data line 15, and the interrupt status bit is read from the status registers on data line 14.

DATA TRANSFER PORT

The port itself is the last block on the right hand side of the page. The port is made of three functional blocks and the 25-pin connector on the edge of the module.

Output Driver Enable

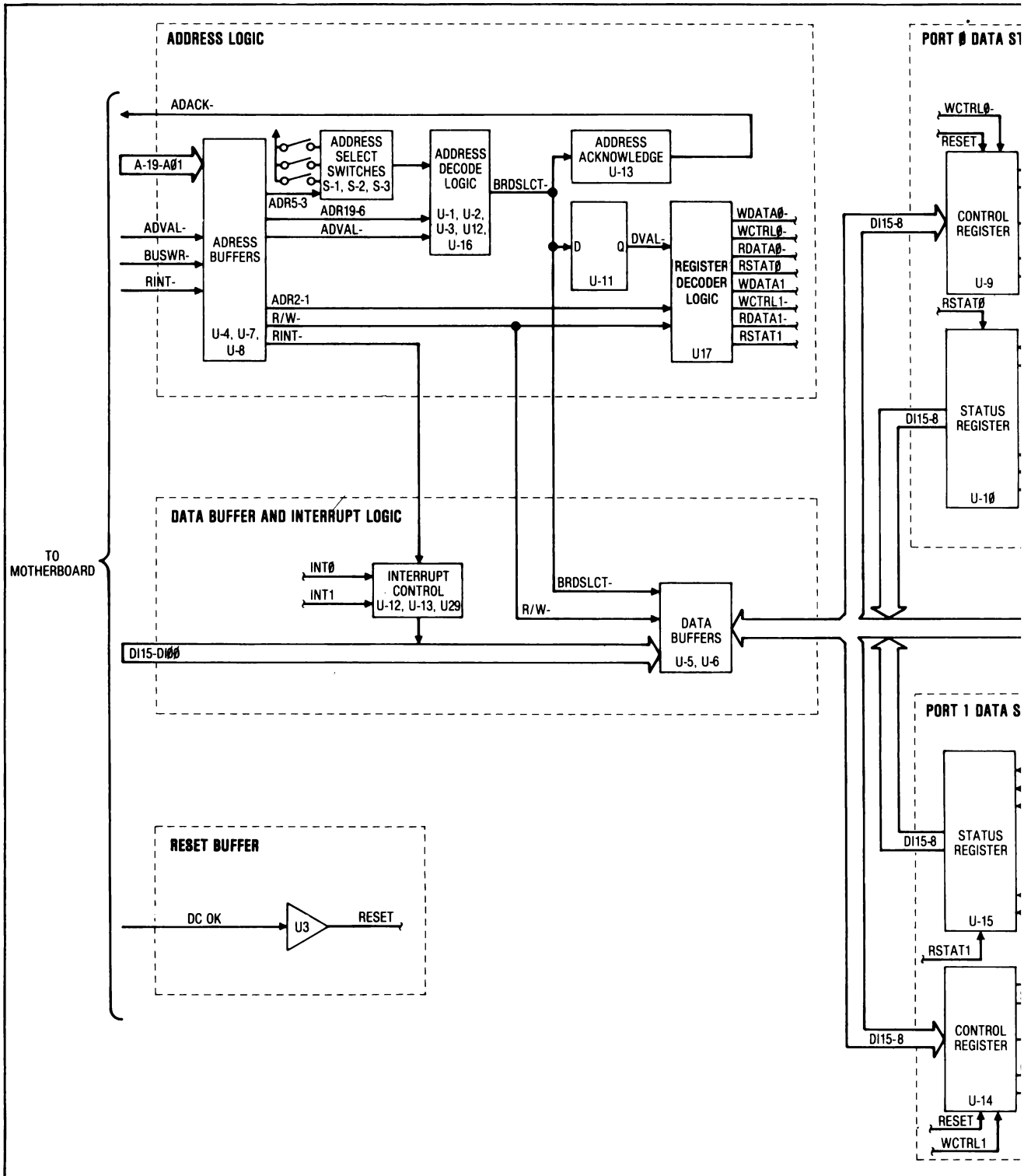
This block enables or disables the port's output drivers. When the output enable signal OUTEN is high, the port is enabled to output data; when OUTEN is low, data output is disabled, and the output data lines are pulled high.

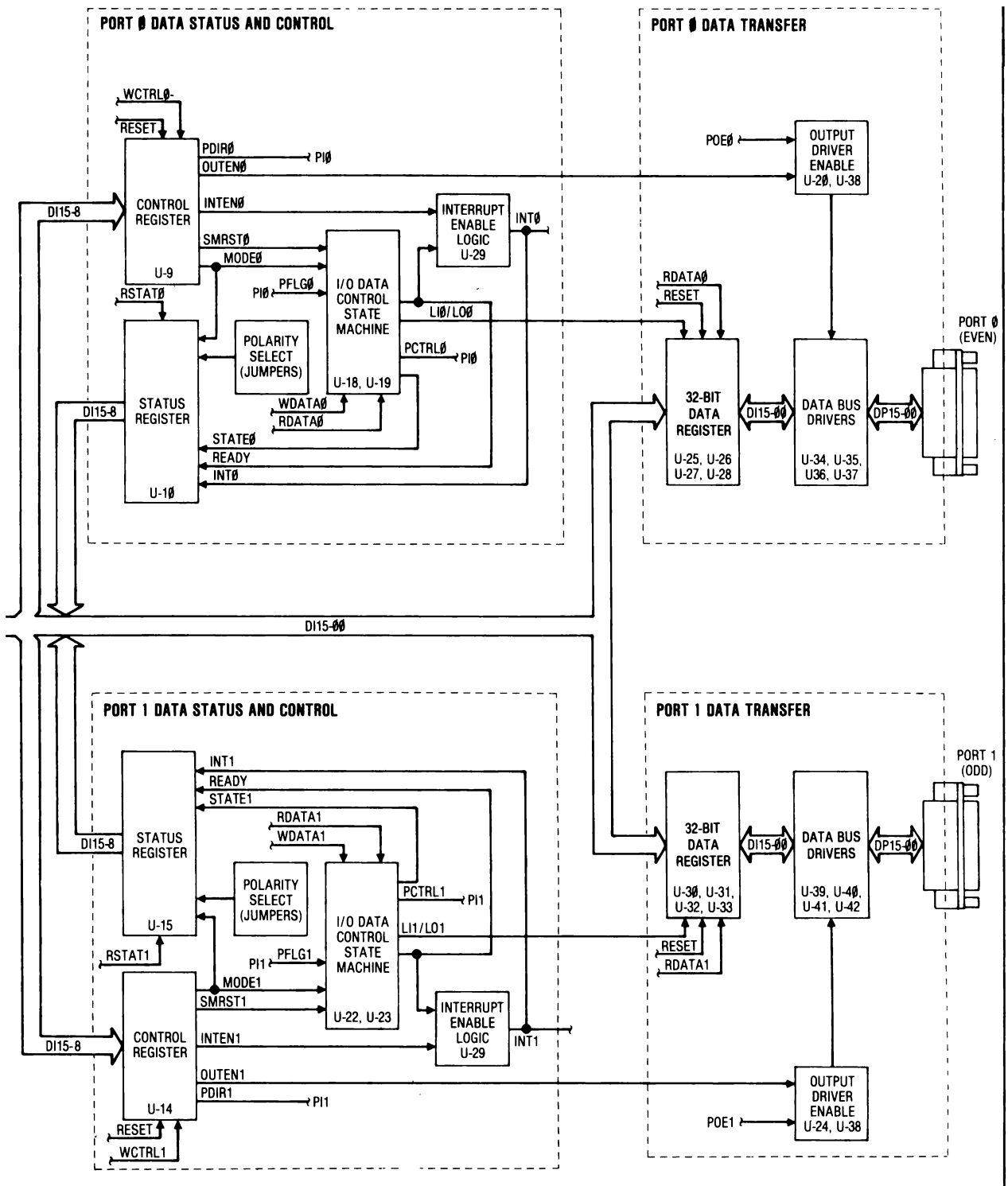
32-Bit Data Register

This 32-bit register latches input and output data.

Data Bus Drivers

These drivers buffer incoming and outgoing data to and from the board.





Parallel Interface Board Block Diagram

Section 9 Options

CONTENTS

Introduction	9-2
Peripherals	9-4
Accessories	9-5
Options	9-6
Option Configuration Table	9-7
User Information	9-9
Installing Hardware Options	9-9
-002 Parallel Interface Board	9-13
-004/-005 Magnetic Bubble Memory	9-25
-006/-007 Memory Expansion Modules	9-29
-008 IEEE-488/RS-232 Interface	9-33
-009 Dual Serial Interface	9-37
1760A and 1761A Disk Drive Systems	9-55
1765A Winchester Disk Drive System	9-63

P/N 760611

January 1985

©1985, John Fluke Mfg. Co., Inc. All rights reserved. Litho in USA.

INTRODUCTION

This section of the manual gathers into one area all information about options available at the time of printing. Because new options are always being investigated and released throughout the life of the instrument, this section is expected to grow between printings of the manual.

When a new option becomes available, first shipments are supported by “User Information” sheets, which can be incorporated into this section after the option is installed. The next printing of the manual will then include the new sheets in this section. Be sure to contact your local Fluke representative for the latest information about available options.

The items discussed in this section are not included in the shipment unless ordered at the same time as the Controller. If specified on the order, options will be installed at the factory. Otherwise, they may be packaged separately.

PERIPHERALS

All the peripherals listed here are separate products, and can be ordered by the model numbers shown.

1760A Disk Drive System, 400K Byte

1761A Dual Disk Drive System, 800K Byte

1765A/AB Winchester Disk Drive, 10M Byte

1780A InfoTouch Display

ACCESORIES

Y1700 Keyboard

Y1706 Ten-pack of Blank Unformatted floppy disks (Certified)

P/N 533547

Pad of 50 Programmers Worksheets

Y1711 Reinforced Shipping Case

IEEE-488 Cables

Y8021 Shielded, 1 meter

Y8022 Shielded, 2 meters

Y8023 Shileded, 4 meters

RS-232C Interface Cables

Standard (For DCE devices)

Y1707 2 meter

Y1708 10 meter

Null Modem (For other DTE devices)

Y1702 2 meter

Y1703 4 meter

Y1705 0.3 meter

Printer Cable

For connecting a serial printer.

Y1709 2 meter

Rack Mount Kits

Y1790 Rack Mount Kit with 24-inch slides

Y1794 Rack Mount Kit with 18-inch slides

Side Carrying Handle

Y1795

OPTIONS

Options are listed by a unique three-digit number appended to the product family identifier. For example, the -004 option 256K Byte Bubble Memory option is ordered using model number 17XXA-004.

Memory Expansion

Memory Expansion options greatly increase the available on-line storage capabilities of the Controller. Memory Expansion Modules can be placed in any of the five unused options slots in the card cage. The maximum dynamic RAM configuration increases the total on-line system memory to about 2.6 Megabytes; Bubble Memory can provide up to approximately 1.3 Megabytes. Combinations are possible; please consult the Option Configuration Table later in this section for complete details.

- 004 256K Byte Bubble Memory
- 005 512 Kbyte Bubble Memory
- 006 256 Kbyte RAM Expansion
- 007 512 Kbyte RAM Expansion

Interface Additions

Interface options expand the Input/Output possibilities of the Controller. They may only be used in card cage slots 1, 3, and 5.

- 002 Parallel Interface
- 008 IEEE-488/RS-232C Interface
- 009 Dual Serial Interface

Configuration Information

Use this table to determine allowable configurations of available hardware options. A dot in a column indicates the slots that the option can be placed in. For example, if all available slots are used for 512K byte RAM expansion memory modules, the system has the maximum memory configuration for this type of memory: 2.6M bytes, but no slots are available for other modules. On the other hand, slots 1, 3, 4, and 6 could be used for additional memory, resulting in 2.1M bytes of additional memory, and slot 5 would still be available for one of the I/O options.

Option Configuration Table

OPTIONS							SLOTS
IEEE-488/RS-232C Interface	Dual Serial Interface	Parallel Interface	512K byte RAM Expansion	256K byte RAM Expansion	512K byte Bubble Memory	256K byte Bubble Memory	
•			•	•	•	•	1
Reserved for Video/Graphics/Keyboard Interface							2
•			•	•	•	•	3
				•	•	•	4
•	•	•	•	•	•	•	5
			•	•	•	•	6
Reserved for Single Board Computer							7

• = Allowable Slot for Option

Software

For increased flexibility, these software options are available to allow programming the Controller in languages other than Interpreted BASIC, which is supplied as the standard programming language. Each language option is supplied as a floppy disk with an accompanying Programming manual.

- 201 Assembly Language Software Development System
- 202 FORTRAN Software Development System
- 203 Compiled BASIC Software Development System
- 205 Extended BASIC Software Development System

1722A USER INFORMATION

Installing Hardware Options

INTRODUCTION

This information is provided to assist you in installing hardware options into a 1722A Instrument Controller. All options are installed in the same way. The instructions here describe how to install options into the Controller's card cage, and give general directions on how to check them out.

Some of the options require some initial set-up. Be sure to refer to the appropriate option User Information for a module's unique requirements before you start the installation.

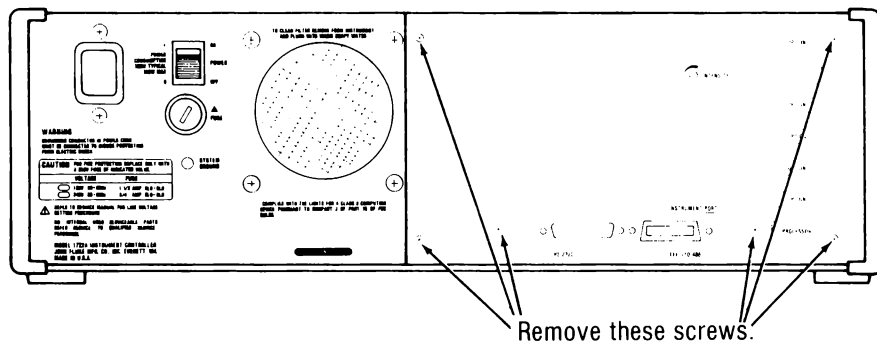
If a new module fails to perform correctly on first applying power, be sure to recheck your work. It may be that a small but important step was missed, resulting in a failure of the new module to operate.

PRE-INSTALLATION CHECKOUT

Inspect the shipping carton for damage, and notify the shipper immediately if it appears to have been damaged. Unwrap the module and inspect it for damage. If everything seems to be in order, set any selection switches or jumpers that are unique to the module. This information is available in the individual sections that cover each of the options.

INSTALLATION

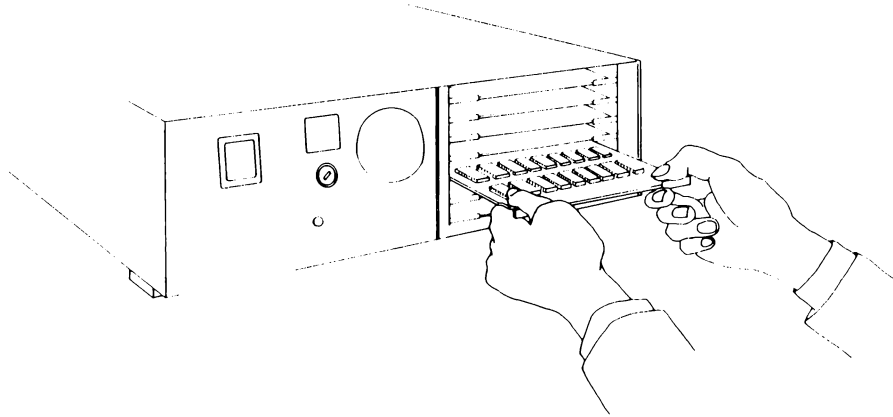
1. Refer to the option User Information provided with the module and perform any preliminary set-up steps.
2. Power down the Controller, and remove the AC line cord.
3. Remove the rear card cage cover, illustrated below. Depending on which modules are already installed, more screws than those indicated may also have to be removed. The 6 screws shown here are those that are removed from a new Controller with no options installed.



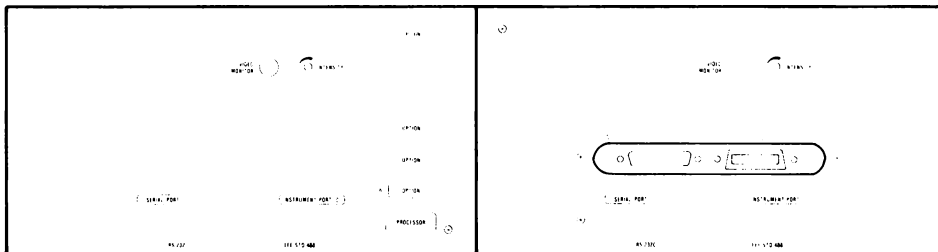
4. Determine which location to use for the option:

Memory Options:	Any open slot
17XX-008 Option:	Slot 5 only
Other Input Output Options:	Slots 1, 3, or 5 only

- Carefully slide the option module into the card cage. Make sure the module is fully seated in the card cage so that it makes solid contact with the card-edge connector.



- If an input/output option is being installed, remove the two screws holding the solid oval plate onto the card cage cover. After reinstalling the card cage cover, use the two screws to attach the shield plate supplied with the option, to the card cage cover. Affix the identification sticker to the outlined area on the card cage cover.



7. Reinstall the power cord.
8. Once installation is complete, power up the Controller and test the module. Refer once again to the option User Information provided with the module for any special requirements or procedures.
9. In case of problems with any new option, recheck your work to insure that switches are set properly, and that the correct jumpers are in place. Be sure that the board is fully seated in the card cage. If everything is in order, but the failure continues, refer to Appendix G, System Diagnostics, for troubleshooting information, or call your local Fluke Service Center.

1722A USER INFORMATION

OPTION 17XXA-002

PARALLEL INTERFACE BOARD

INTRODUCTION

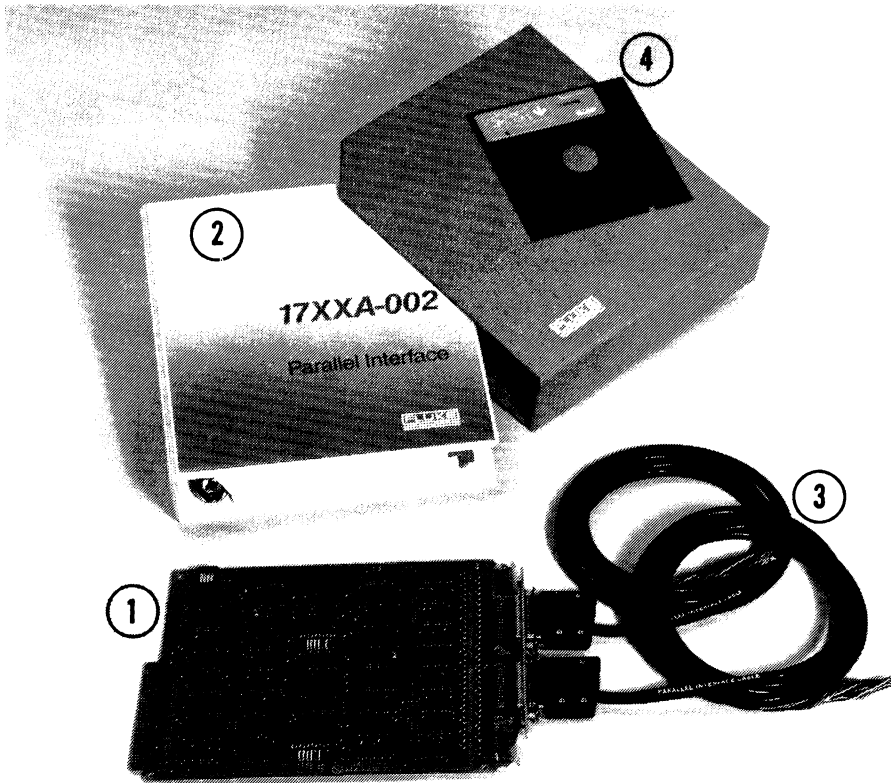
The Option 17XXA-002 Parallel Interface (PIB) is an extremely versatile addition to the Fluke Instrument Controller. The PIB has the built-in ability to adapt to some of the most unusual interface requirements of connected devices.

Option Contents

The photograph and table below show the contents of the 17XXA-002 option. The floppy disk is supplied to support the PIB in 1720A installations. For the 1722A Controller, the PIB support software is supplied on the standard system disk. With minor variations, which are fully explained in the manual, both sets of routines operate the same.

17XXA-002 PARALLEL INTERFACE OPTION CONTENTS

NUMBER	ITEM	JOHN FLUKE PART NUMBER
1	Parallel Interface Module	611947
2	Instruction Manual	732230
3	(2) Parallel Interface Cables	733907
4	1720A PIB Software Disk	630699



INSTALLATION AND CHECKOUT

The instructions that follow describe how to install a Parallel Interface option in a 1722A Instrument Controller.

Configuration Procedure

The operating configuration of the Parallel Interface is set up by a board address switch on the module.

1. Set the address switch for ports PI0 and PI1. Use the table below to ensure that the switch is set properly for port addressability.

SWITCH/ADDRESS OPERATING CONFIGURATION

BOARD	SW4	SW3	SW2	SW1	PORT PI0	ADDRESS PI1RANGE	ADDRESS
1	N	ON	ON	ON	0	1	F340-F346
2	O	ON	ON	OFF	2	3	F348-F34E
3	T	ON	OFF	ON	4	5	F350-F356
4		ON	OFF	OFF	6	7	F358-F35E
5	U	OFF	ON	ON	8	9	F360-F366
6	S	OFF	ON	OFF	10	11	F368-F36E
7	E	OFF	OFF	ON	12	13	F370-F376
8	D	OFF	OFF	OFF	14	15	F378-F37E

2. Once the switch has been set, use the directions in the Options section "Installing Hardware Options" to install the -002 option into the 1722A.
3. Power up the Controller and test the new interface by using the System Diagnostic software. Appendix G of the System Guide explains how to use the System Diagnostic software to test the -002 option.
4. In case of problems with the new module, recheck your work to ensure that the board is fully seated in the card cage, and that port connectors are attached securely. If everything is in order but the failure continues, refer to Appendix G for troubleshooting information, or call your local Fluke Service Center. The Parallel Interface module is included in Fluke's Module Exchange Program.

SOFTWARE COMPATIBILITY

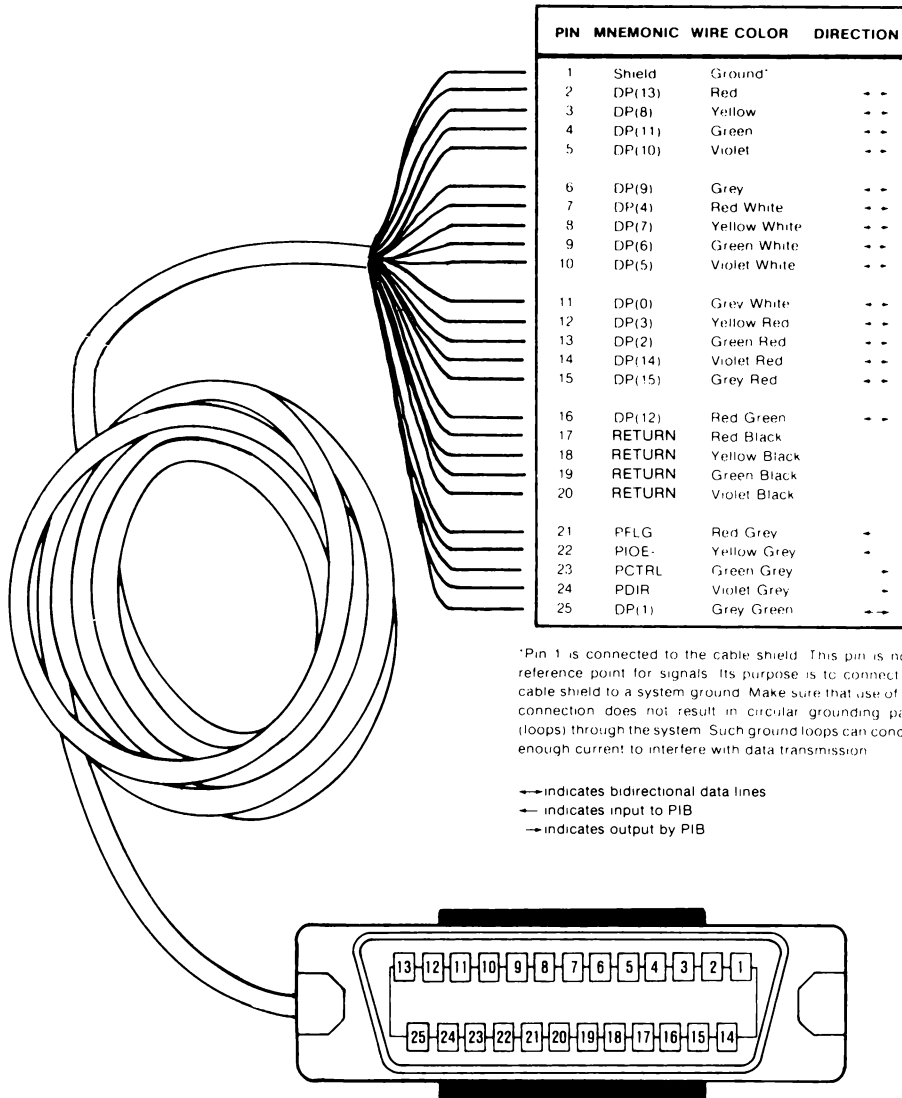
For the 1722A Instrument Controller, the Parallel Interface software is supplied on the 1722A System Disk. These machine language programs are interdependent and are compatible only in the combinations of versions supplied by Fluke and documented in published Fluke manuals. Using incompatible system software modules may give unpredictable results. In this case, Fluke cannot provide software support except for identification of compatible combinations.

NOTE

The floppy disk supplied with the Parallel Interface option contains software for operating the interface when it is installed in a 1720A Instrument Controller ONLY. The same routines for the 1722A Controller are already supplied on the System Disk that is shipped with the instrument.

INTERFACE DESCRIPTION

The port lines provide bidirectional data transmission between the Instrument Controller and compatible external devices. The drawing below shows the cable connector and lists the signal definition and logic states for each of the lines at the interface.



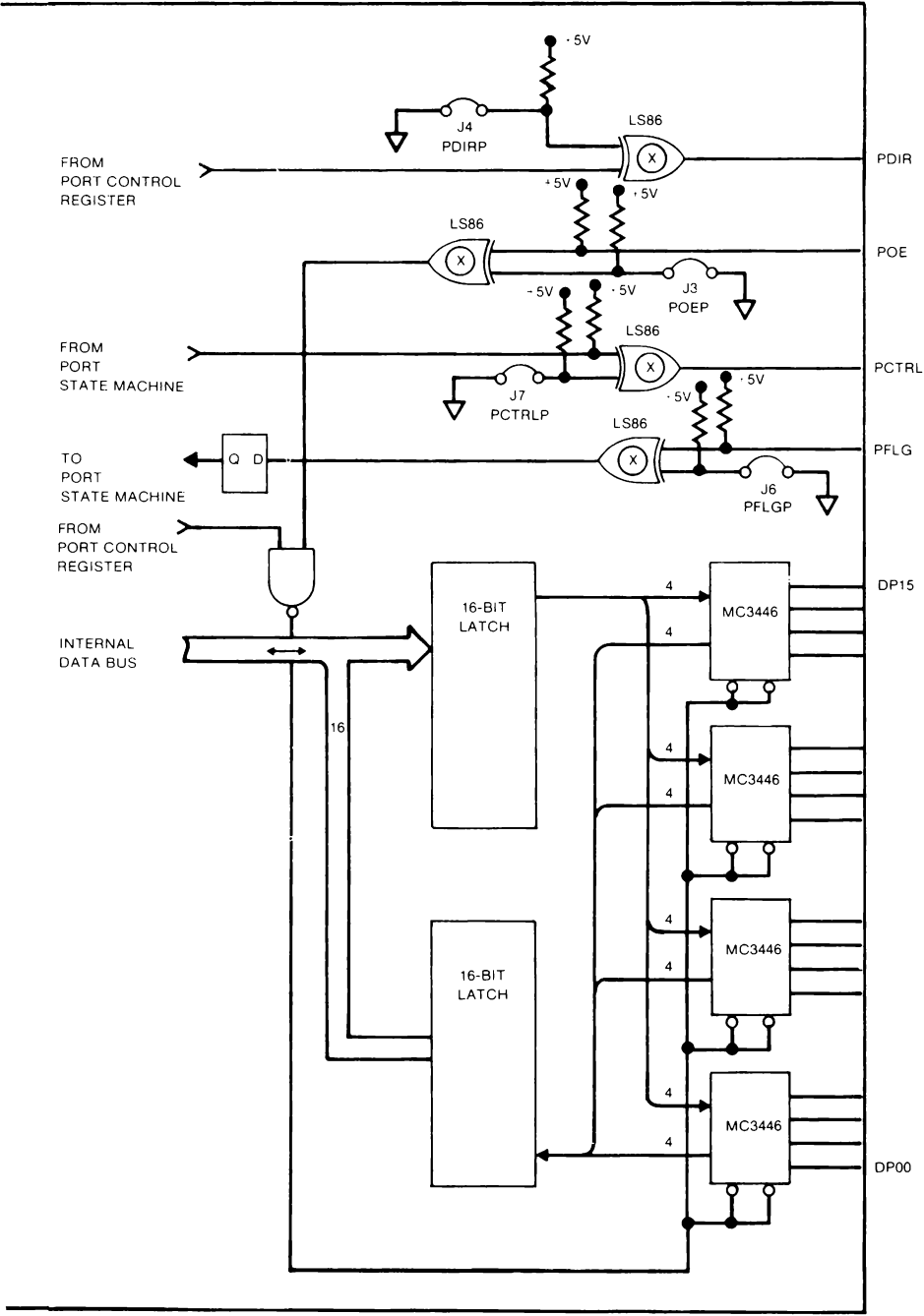
Port Signals

On the Parallel Interface module, several signals control the direction of data and the sense of the handshake lines. These signals are defined in the table below. The jumpers shown on the port schematic permit inverting handshakes.

PORT SIGNAL DEFINITIONS

SIGNAL NAME	DIRECTION	DEFINITION
PDIR	Output	Port Direction: Direction of data on port.
POE	Input	Port Output Enable: Used by external device to disable the output from the ports.
PFLG	Input	Port Flag: One of the two handshake lines; this signal originates with the peripheral device. For input operations (device to port), PFLG, when asserted, indicates that input data is valid. For output operations, PFLG, when asserted, acknowledges that the device is reading the output data from the port.
PCTRL	Output	Port Control: The other handshake line; the signal originates with the Parallel Interface module. For output operations (port to external device), PCTRL, when asserted, indicates that the output data is valid and the device may read it. For input operations, PCTRL, when asserted, indicates that the port is reading the incoming data from the device.
Data	Bidirectional	Each data line can be used either as an input or as an output.

User Information
Parallel Interface Board



SOFTWARE LIBRARY (PIBLIB.OBJ)

The next few pages summarize the available software routines. The parameters are described immediately following the descriptions of the routines.

CHKBIT Check bit

Parameters: port, bit, bool

Description: This routine checks a particular bit on a Parallel Interface Port. It is equivalent to reading the port and then isolating the specified bit position.

CLRBIT Clear bit

Parameters: port, bit

Description: This routine clears to zero a particular bit on a selected port. It is equivalent to reading the port, ANDing the bit with 0 in the appropriate position, and then writing the data word back out to the port.

SETBIT Set bit

Parameters: port, bit

Description: This routine sets a particular bit (output = high) on a selected port. It is equivalent to reading the data latched at the port, ORing the bit in the specified position with 1, and then writing the data word back out to the port.

RDWORD Read word

Parameters: port, word

Description: This routine reads a selected port and writes the value into a variable.

WTWORD Write word

Parameters: port, word

Description: This routine writes a word from a variable to a specified port.

RDBLK Read block

Parameters: port, block, count

Description: This routine reads multiple words from a port into an array.

WTBLK Write block

Parameters: port, block, count

Description: This routine writes multiple words to a port from an array.

FRDBLK Fast Read block

Parameters: port, block, count

Description: This routine reads a block of data from a port, as fast as possible. It does not perform the error checking that is done in a normal read of a block using RDBLK.

FWTBLK Fast Write block

Parameters: port, block, count

Description: This routine writes a block of data to a port as fast as possible. It does not perform the error checking that is done in a normal block write using WTBLK.

POPEN Open a port

Parameters: port, mode, mask, timeout

Description: POPEN opens a port in preparation for data transfer at the interface.

PCLOSE Close a port

Parameters: port

Description: PCLOSE closes the specified port and returns the hardware to a passive state similar to the power-up condition.

Parameters

When the routines are used in a program, one or more parameters are specified by the programmer. All parameters must be specified as integers.

PORT The port number for the routine to operate on, expressed as an integer in the range 0 - 15.

MODE The Parallel Interface module operates in one of four modes: No Handshake, Full Handshake, and Strobe Input and Output Handshakes. With the exception of No Handshake, the handshakes synchronize incoming and outgoing data. The table below lists the handshake names and the mode number that the software recognizes.

HANDSHAKE MODES

NAME	DEFINITION	MODE
—	No Handshake	0
HNDSHKIN	Full Handshake Input	1
STROBEIN	Strobe Input	1
HNDSHKOUT	Full Handshake Output	2
STROBEOUT	Strobe Output	3

TIME OUT The wait time before an incomplete handshake is terminated.

BIT The bit number to be checked, read, or written to, expressed as an integer.

BOOL When a bit or word is read or checked, as in the CHKBIT routine, the routine places a value into the variable specified as **BOOL**.

DIRECTION MASK An integer that indicates the desired transmission direction of each bit on the port.

- BLOCK** This parameter is the array to which data will be transferred using the Block subroutines RDBLK, WRTBLK, FRDBLK, and FWTBLK.
- COUNT** Indicates how many array elements to transfer.

CONCLUSION

This information has been taken from the manual supplied with the PIB. Besides being the definitive source of information about the topics presented here, the PIB manual includes a great deal more information, including:

- Specifications
- Interface timing
- Sample programs
- Performance Testing
- Theory of Operation
- Schematic

1722A USER INFORMATION

Option 17XXA-004/005

Magnetic Bubble Memory

INTRODUCTION

The Option 17XXA-004/-005 Bubble Memory Modules provide additional memory for the 1722A.

Like a floppy disk, the Bubble Memory is treated by FDOS as a file-structured device. Information is retained in the device when the power is turned off.

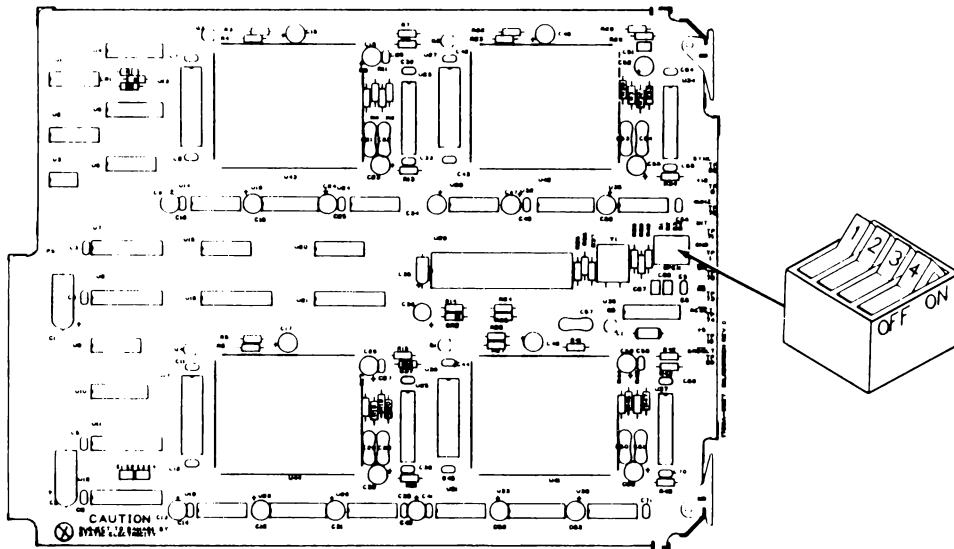
The 17XXA-004 Bubble Memory Module contains 256K bytes of memory; the 17XXA-005 module contains 512K bytes. The maximum amount that can be installed in a system is 1.5M bytes (any combination of three modules).

PRE-INSTALLATION CHECKOUT

Inspect the shipping carton for damage. Notify the shipper immediately if the carton appears to have been damaged in shipping. Unwrap the module and inspect it for damage. If everything seems to be in order, go on to the next step, setting the board's address switch.

Board Addressing

In order for the Operating System (FDOS) to operate properly using bubble memory, there must be a unique address associated with each Bubble Memory Module installed. Use the drawing below to locate the board address switch (SW1).



The 1722A has five slots available for options. Bubble Memory Modules can be installed into any three of them. The SW1 address switch settings determine the device names, and are identical for Option -004 (256K bytes) or Option -005 (512K bytes).

User Information
Magnetic Bubble Memory

DEVICE NAME	ADDRESS CODE	SWITCH POSITIONS			
		1	2	3	4
MB0:	111X	on	on	on	X
MB1:	110X	on	on	off	X
MB2:	101X	on	off	on	X
MB3:	100X	on	off	off	X

NOTES: 1. "1" = on, "0" = off, "X" = don't care.

2. Four device names are available. However, only three are used at a time because only three modules can be installed at one time.

INSTALLATION AND CHECKOUT

1. Once address switch SW1 has been set, follow the directions in the Options Section “Installing Hardware Options” to install the module into the Controller’s card cage. Be sure to turn the power off before beginning.
2. Power up the Controller and insert the 1722A System Software disk. If required, use the System Generation Utility program (SYSGEN) to make a new System Software disk that includes the Bubble Memory driver. Refer to Section 3 of the System Guide for instructions.
3. When a new System Disk has been generated, press the RESTART button and let the disk load the new FDOS.
4. Use the File Utility Program (FUP) to format the Bubble Memory Module. Type MBx:/F, where x is the device number given the module by its address switch setting. See Section 4 of the System Guide for complete details about using FUP.
5. Test the new memory board by running the Bubble Memory diagnostic, as described in Appendix G, “System Diagnostic Software”, or by transferring files to and from the bubble memory device using FUP.
6. If the diagnostic test executes successfully, your Bubble Memory Module is now available for use.
7. If any trouble develops, first make sure that the address selection switch is set properly, and that the module is seated firmly in the card cage. If everything seems to be in order, but the failure continues, refer to Appendix G, “System Diagnostic Software”, or call your Fluke Technical Service Center for assistance in tracking down the problem. The Bubble Memory Module is included in Fluke’s Module Exchange Program.

1722A USER INFORMATION

Option 17XX-006/007

Memory Expansion Module

INTRODUCTION

The Option 17XX-006/007 Memory Expansion Modules provide additional memory for the 1722A. This added memory can also be configured as Electronic disk.

The Operating System treats memory configured as E-Disk as an electronic version of a floppy disk. This means that files are stored and retrieved from E-Disk in a formatted fashion like a floppy disk. See the File Utility Program in Section 4 of the System Guide for instructions on how to configure E-Disk space.

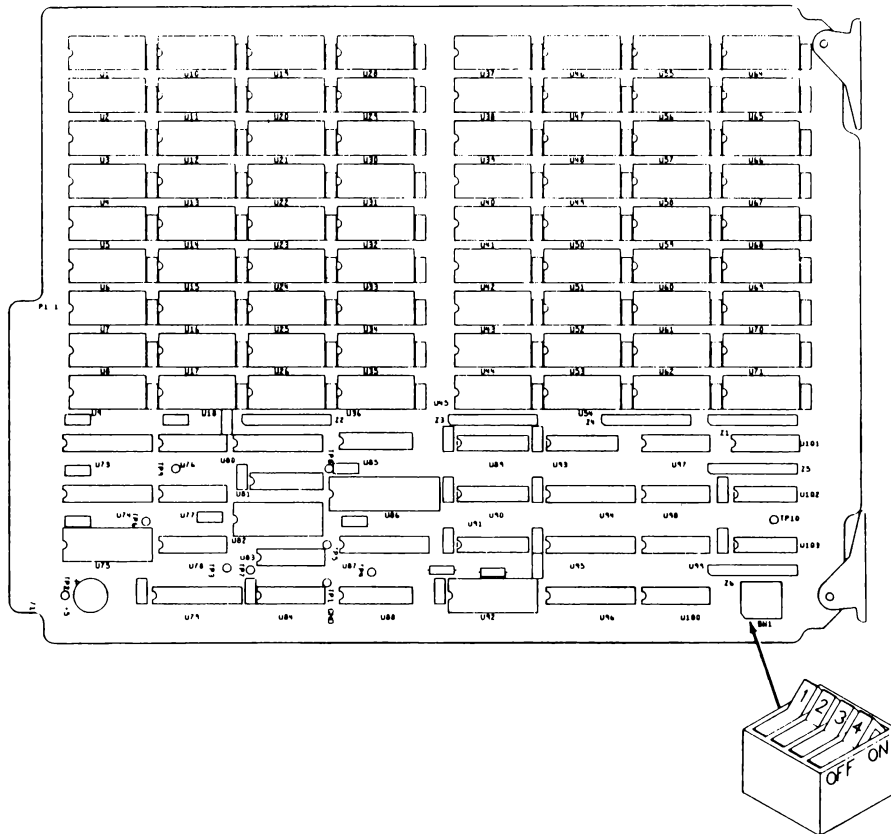
The 17XX-006 Memory Expansion Module contains 256K bytes of dynamic RAM; the 17XX-007 module contains 512K bytes. Any combination of up to five modules can be installed in a 1722A at a time. If all of the slots are filled with 17XX-007 boards, then the total expansion memory added to the 1722A would be over 2.6M bytes.

PRE-INSTALLATION CHECKOUT

Inspect the shipping carton for damage. Notify the shipper immediately if the carton appears to have been damaged in shipping. Unwrap the module and inspect it for damage. If everything seems to be in order, go on to the next step, setting the board's address switch.

Board Addressing

In order for the Operating System to operate properly using expanded memory, each memory board installed must have unique addresses set. Each module has a memory switch, located in the drawing below.



The 1722A has five slots available for options. Expansion memory modules can be installed into any or all of them. To assure proper operation of diagnostics, the first module added should be given the address for Unit One as shown in the table below. Subsequently added modules are given addresses in ascending unit number order. The SW1 address switch settings shown below are identical for Option -006 (256K bytes) or Option -007 (512K bytes).

UNIT NUMBER	ADDRESS CODE	SWITCH POSITIONS			
		1	2	3	4
1	1110	off	off	off	on
2	1100	off	off	on	on
3	1010	off	on	off	on
4	1000	off	on	on	on
5	0110	on	off	off	on

NOTES

- 1) "0" = on and "1" = off on the option's address switch label.
- 2) Although the Controller may operate properly with addresses set out of order, setting them in the recommended order ensures that diagnostic software can correctly identify faulty components, and will prevent possible bus contention problems when mixing -006 and -007 options.

Example:

Two -007 Options, and two -006 Options are to be installed. In this case, we will set the addresses for the larger memory sizes first by setting their switches to 1110 and 1100. Next, the first -006 option's switches are set to 1010, and the second one to 1000.

These settings leave a 256K byte gap between the memory addresses occupied by the -006 modules. This gap is transparent when the module is in use.

INSTALLATION AND CHECKOUT

1. Follow the directions in the Options Section titled “Installing Hardware Options” to install the module into the Controller’s card cage. Be sure to turn the power off.
2. To check the new memory module, power up the system and load the Operating System software. Observe the amount of memory message that appears when FDOS loads. It should indicate the additional memory that is now available, both in bytes and blocks. (1 block = 512 bytes.)
3. To exercise the new memory, use the File Utility Program to configure all available free blocks as E-Disk, then transfer a large amount of files to the E-Disk, and see that they can be read to the screen. If everything is in order, this is an adequate check that the Memory Expansion Module is operational. Section 4 of the 1722A System Guide, Devices and Files, explains all the operations of the File Utility Program.
4. If any trouble develops, first recheck your work. Make sure that the address selection switches are set properly, and that the module is properly seated into the connector on the motherboard. If everything seems to be in order, refer to Appendix G, System Diagnostics, or call your Fluke Technical Service Center for assistance in tracking down the trouble. The Memory Expansion module is included in Fluke’s Module Exchange Program.

1722A USER INFORMATION

Option 17XXA-008

IEEE-488/RS-232C Interface

INTRODUCTION

This section of the 1722A System Guide covers the Option 17XXA-008 IEEE-488/RS-232 Interface Module. The module provides the 1722A Instrument Controller with one additional IEEE-488 port, and one additional RS-232 port.

As shipped, the standard 1722A Instrument Controller has a single IEEE-488 port and one RS-232 port. The IEEE-488 port has the device name GP0: when used as a serial device (output only), and Port 0 when used by a program as an instrument port. The standard configuration RS-232 port has the device name KB1:.

The IEEE-488 port on the -008 option has the device name GP1: or Port 1, and the RS-232 port has the device name KB2:. See Section 4 of the System Guide for more information on devices.

PRE-INSTALLATION CHECKOUT

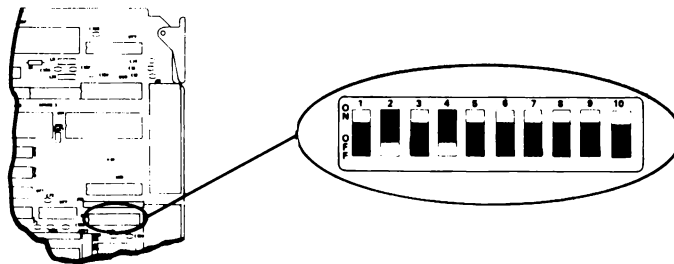
Inspect the shipping carton for damage. Notify the shipper immediately if the carton appears to have been damaged in shipping. Unwrap the module and inspect it for damage. If everything seems to be in order, go on to the next step, Installation.

INSTALLATION

1. Refer to the drawing below to locate and set the configuration switches. The initial setup establishes the module's IEEE-488 address as 0, and its function as "system controller". This switch setting also sets the RS-232 port to 4800 baud for power up, but the baud rate can easily be changed later using the Set Utility Program. See Section 5 of the System Guide for details.

NOTE

Both the standard IEEE-488 port and the one added by the -008 option can be set up as "system controller", because the two ports are effectively two separate systems. However, if both of them will be connected to the same bus, then one of the ports must be set up as "idle controller".



SWITCH 1

1	2	3	4	5	6	7	8	9	10	
—	S2	S1	S0	SC		A4	A3	A2	A1	
x						unused				
						IEEE-488 ADDRESS				
						0	0	0	0	0
						0	0	0	1	1
						0	0	1	0	2
						0	0	1	1	3
						0	1	0	0	4
						0	1	0	1	5
						0	1	1	0	6
						0	1	1	1	7
						1	0	0	0	8
						1	0	0	1	9
						1	0	1	0	10
						1	0	1	1	11
						1	1	0	0	12
						1	1	0	1	13
						1	1	1	0	14
						1	1	1	1	15
						IEEE-488 CONTROLLER				
				0	0	System Controller				
				1	1	Idle Controller				
						BAUD RATE				
	0	0	0			110				
	0	0	1			300				
	0	1	0			600				
	0	1	1			1200				
	1	0	0			2400				
	1	0	1			4800				
	1	1	0			9600				
	1	1	1			19200				

2. Once the switch has been set, use the directions in the Options section “Installing Hardware Options” to install the -008 option into the 1722A.
3. Power up the Controller and test the new interface by using the System Diagnostic software. Appendix G of the System Guide explains how to use the System Diagnostic software to test the -008 option.
4. In case of problems with the new module, recheck your work to ensure that the board is fully seated in the card cage, and that port connectors are attached securely. If everything is in order but the failure continues, refer to Appendix G for troubleshooting information, or call your local Fluke Service Center. The IEEE-488/RS-232 Interface module is included in Fluke’s Module Exchange Program.

1722A USER INFORMATION

OPTION 17XXA-009

DUAL SERIAL INTERFACE

INTRODUCTION

The 17XXA-009 Dual Serial Interface (DSI) provides the 1722A Instrument Controller with two additional serial communications ports. The ports are addressed as SP0: - SP9: through the Set Utility program and high level languages. They are treated similarly to KB0: and KB1: on the Single Board Computer (SBC), and KB2: on the IEEE/RS-232 Option (-008).

Up to three DSI modules can be installed in the 1722A. Each port may be configured for these electrical interfaces:

- RS-232-C
- RS-422
- 20 mA Current Loop

Each port buffers incoming and outgoing data and signals the external device when the buffers are nearly full to prevent loss of data. The signaling method or protocol may be selected as discussed below. The ports are controlled by a microprocessor which reduces the overhead on the Single Board Computer (SBC). System throughput is a function of the data being transferred at the floppy disk and the IEEE-488 and KBx: ports. If the load from these devices is heavy, external devices will be held off more frequently regardless of the data rate selected.

The Operating System (FDOS) which is supplied on the 1722A System Disk includes a device driver for the -009 option. This operating system must be used in order for your software to be able to access the Dual Serial Interface. If necessary, you can reconfigure the operating system by using this disk and following the instructions in Section 3 of the 1722A System Guide.

The disk also includes the Serial Port Software Driver (SPIO.OBJ). This driver allows a BASIC program to directly monitor and control the various lines of the serial interface, including RS-422 ENABLE, which some users may require. Information later in this section describes the operation of the driver.

PRE-INSTALLATION CHECKOUT

Inspect the shipping carton for damage. Notify the shipper immediately if the carton appears to have been damaged in shipping. Unwrap the module and inspect it for damage.

INSTALLATION

The Option has been configured at the factory as follows:

Electrical Interface	RS-232-C
Data rate	4800 Baud
Data Bits	7
Parity	none
Stop Bits	2
Flow Control	off
Board Address	0 (SP0: and SP1:)

The jumpers and switches are set at the factory as shown in the figure on the next page. Jumpers JPR1, 2, 3, 4, 5, 6, 29, and 35 are not user-configurable and must remain in the positions shown. Check all jumpers and switches to be sure they match the factory configuration.

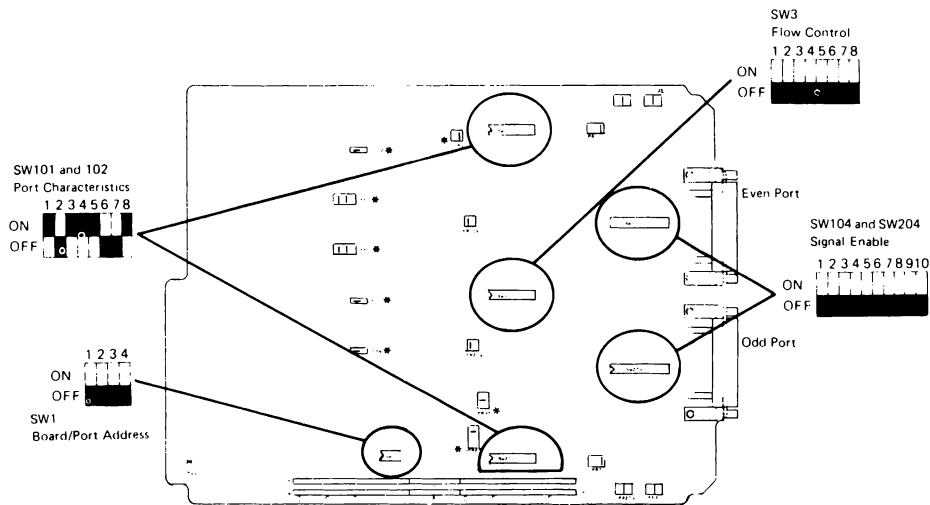
Use the directions in the Options Section of the System Guide "Installing Hardware Options" to install the -009 option into the 1722A.

If you have reconfigured the Operating System with the SYSGEN Program, make sure that the Dual Serial Interface Driver has been included. You can check this by re-booting the controller with your version of the Operating System and then running the CONFIG Program supplied on the standard System Disk.

The signals for all three interface types are available on each of the port connectors. Since this is a modification of the RS-232-C standard, the redefined pins can be switched off using SW104 and SW204 when RS-232-C is desired. This allows using cables with connections for signals defined by the standard to be used. Refer to the section “Electrical Interfaces” for connections to external devices.

In case of problems with the new module, recheck your work to ensure that the board is fully seated in the card cage, and that the port connectors, jumper positions, and switch settings are correct. If the failure continues, refer to Appendix G for troubleshooting information, or call your local Fluke Service Center. The Dual Serial Interface is included in Fluke’s Module Exchange Program.

FACTORY CONFIGURATION



Note: * indicates jumpers that are not user-configurable.
Do not move these jumpers.

POWER ON CONFIGURATION

When the Dual Serial Interface is powered on, these things take place:

- The configuration switches are read.
- The RS-422 drivers are enabled.
- The DSI waits for input or output.

The initial states of the control lines are shown later. See “Electrical Interfaces - RS-232-C”.

PROTOCOLS

XON/XOFF

The ASCII Standard defines two codes that may be used to control data transfer between devices. If the input buffer of the device receiving data is full or nearly full, XOFF is sent to the transmitting device to request the transmission be stopped. When the receiver can accept more data, the XON code is sent to resume the transmission. The Set Utility program can be used to enable or disable this protocol, and refers to the protocol as “stall input” and “stall output”.

Secondary Request to Send (RS-232)

SRTS is a handshake line that is used for flow control with external devices that cannot respond to XON/XOFF codes. The polarity of SRTS is set by the Flow Control configuration switch (SW3).

RECONFIGURATION

The electrical interfaces and power-up configuration can be changed using the switches and jumpers described in the following tables. After the tables, the next sections discuss each interface and typical setups. The Set Utility program can also be used to change port characteristics. See Section 5 of the System Guide for details.

NOTE

The Port Characteristics and Flow Control switches are read only at power-up.

Board/Port Addresses (SW1)

BOARD	PORT	1	2	3	4
1	1, 0	0	0	0	0
2	3, 2	0	0	0	1
3	5, 4	0	0	1	0
4	7, 6	0	0	1	1
5	9, 8	0	1	0	0

ON = 1 = closed = enabled
OFF = 0 = open = disabled

J1 = EVEN Port
J2 = ODD Port

Port Characteristics (SW101 AND SW201)

<u>1 2 3 4</u>		<u>5 6 7 8</u>			
1 1 1 1	19200 Baud	1			Data Bits: 7
1 1 1 0	19200	0			8
1 1 0 1	9600				
1 1 0 0	7200	1 1			Parity: even
1 0 1 1	4800	1 0			odd
1 0 1 0	3600	0 1			none
1 0 0 1	2400	0 0			none
1 0 0 0	2000				
0 1 1 1	1800		1		Stop Bits: 2
0 1 1 0	1200		0		1
0 1 0 1	600				
0 1 0 0	300				
0 0 1 1	150				
0 0 1 0	134				
0 0 0 1	110				
0 0 0 0	75				

User Information
Dual Serial Interface

Flow Control (SW3)

1 2 3 4 5 6 7 8

1									<u>Odd Port</u>
0									enable
1									disable
0									active high
									active low
		x	x						not used
									<u>Even Port</u>
									enable
									disable
									active high
									active low
									not used

Port Connector Signal Disable Switches (SW104 and SW204)

1	RS-422	Rx+
2		Rx-
3	20mA	Rx+
4		Rx-
5		I1-/-12v
6		I2-/-12v
7	RS-422	Tx+
8		Tx-
9	20mA	Tx+
10		Tx-

Clear To Send (CTS) Input to UART (JPR116 and JPR216)

Left	CTS
Right	CTS and SRLSD

UART Receive Input (JPR117 and JPR217)

Left	20mA loop
Middle	RS-422
Right	RS-232-C

Voltage/Current Sources (JPR102, 103, 202, 203)

Left	20mA
Middle	OFF
Right	+12v

ELECTRICAL INTERFACES

RS-232-C

Maximums

Distance	50ft
Data Rate	19200 Baud

Typical Applications

Data Communications Equipment (DCE)
(Use RS-232-C Cable)

Modems

Data Terminal Equipment (DTE)
(Use “Null Modem Cable”)

1780A
VT100
Printers

Port Connections

To DCE			From DCE		
Pin	Circuit Function		Pin	Circuit Function	
1	AA	Shield			
7	AB	Signal Common			
2	BB	Transmitted Data	3	BA	Received Data
4	CA	Request to Send	5	CB	Clear to Send
20	CD	Data Terminal Ready	6	CC	Data Set Ready
19	SCA	Secondary Request to Send			
11	UND	RS-232-B			
			22	CE	Ring Indicator
			12	SCF	Secondary Received Line Signal Detector
			8	CF	Received Line Signal Detector

The Clear To Send jumpers (JPR116 and 216) allow the CTS input to the UART to be either the CB circuit or the logical AND of CB and SCF. This feature is useful for external devices that use the SCF circuit as a “busy” indicator.

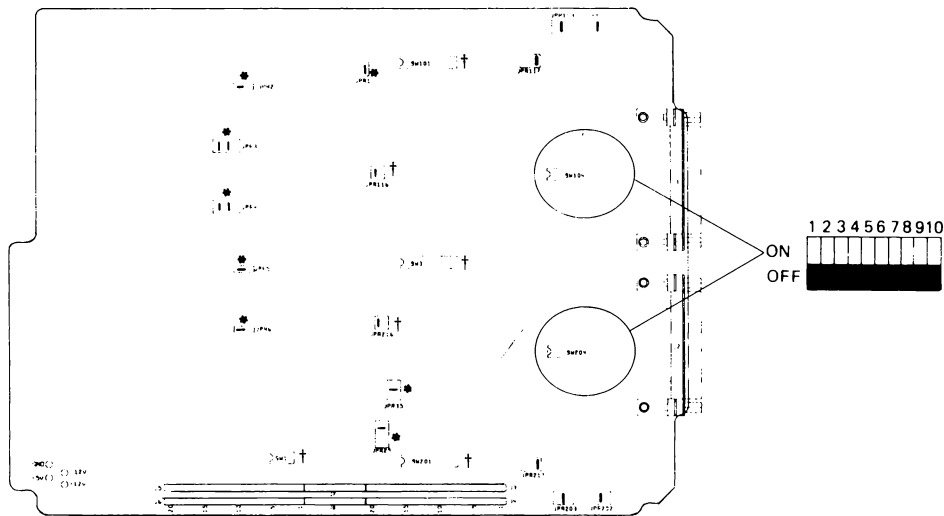
Power-on States of the Control Lines

SIGNAL	PIN	STATE
BB	2	MARK
CA	4	ON
SCA	11, 19	OFF
CD	20	ON

NOTE

Clear To Send (Pin 5) controls the transmission from the port. When it is ON, the corresponding UART is permitted to transmit. When it is OFF, the UART stops transmitting, beginning at the next character boundary. This behavior is a function of the UART hardware, and always applies. It cannot be disabled. Leave Pin 5 unterminated if not used by the receiving device.

RS-232-C CONFIGURATION



*Notes: * indicates jumpers that are not user-configurable. Do not move these jumpers.*

† indicates switches or jumpers that can be set as desired. Refer to tables in the text for settings.

RS-422

Maximums

Distance 4000 feet
Data Rate 19200 Baud

Typical Applications

2400B
1780A/AU

Protection Networks

The circuitry incorporates protection networks on the drivers and receivers to reduce susceptibility to high voltage transients and faults.

Port Connections

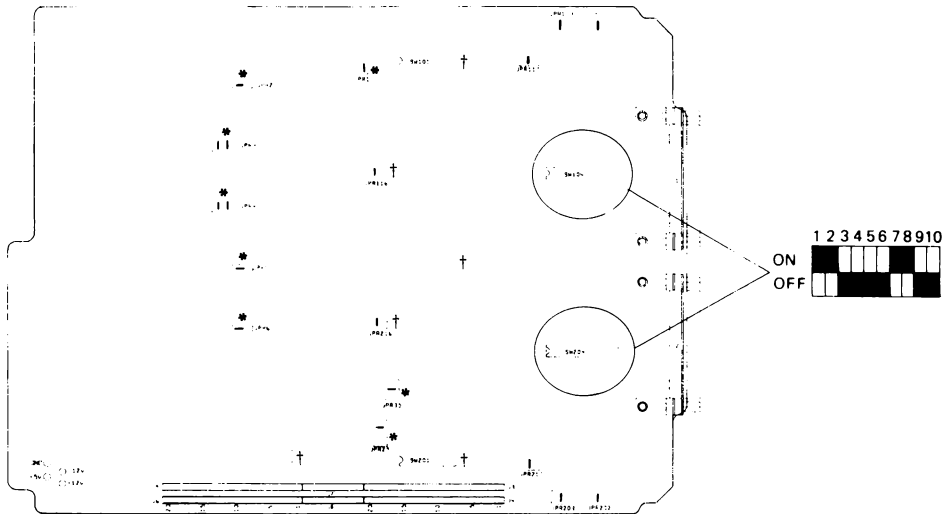
To External Device

PIN	SWITCH	SIGNAL
9	7	Tx+
10	8	Tx-
7		Signal Ground

From External Device

PIN	SWITCH	SIGNAL
14	1	Rx+
15	2	Rx-

RS-422 CONFIGURATION



*Notes: * indicates jumpers that are not user-configurable. Do not move these jumpers.*

† indicates switches or jumpers that can be set as desired. Refer to tables in the text for settings.

20 MA LOOP

Maximums

Distance 1000 feet
Data Rate 4800 Baud
Voltage 30 Vdc

Typical Application

• TTY

Port Connections

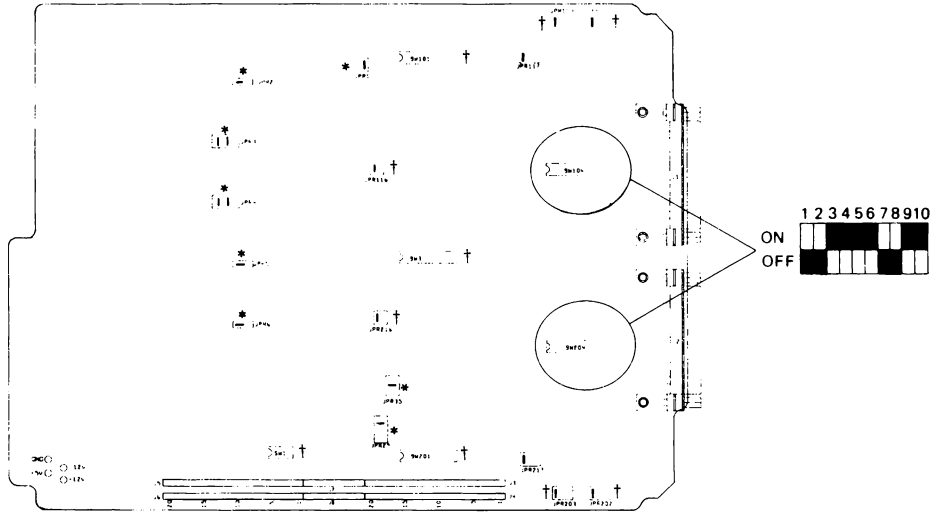
To External Device

PIN	SWITCH	SIGNAL
12	9	Tx+
23	10	Tx-
18	5	I1-/-12v
16	6	I2-/-12v

From External Device

PIN	SWITCH	SIGNAL
24	3	Rx+
25	4	Rx-
17	JPRx02	I1+ /+12v
13	JPRx03	I2+ /+12v

20 MA CURRENT LOOP CONFIGURATION



Notes: * indicates jumpers that are not user-configurable. Do not move these jumpers.

† indicates switches or jumpers that can be set as desired. Refer to tables in the text for settings.

USING THE DUAL SERIAL INTERFACE

The -009 device driver in FDOS allows the programmer to use the DSI from all languages and utility programs just as any other serial device. The ports are specified as SP0: through SP9: depending on the board address selected.

Two additional routines are provided to gain more direct control of the interface lines. The user can read the current state of all the input and output lines and can set the output lines to any state he desires. The user can also enable and disable the RS-422 drivers. These routines are supplied in the following files for each programming language:

LANGUAGE	FILENAME	SUPPLIED WITH
Interpreted BASIC	SPIO.OBJ	1722A System Disk
Compiled BASIC	BASIC.LIB	17XXA-203 Compiled BASIC
Extended BASIC	BASIC.LIB	17XXA-205 Extended BASIC
FORTRAN	FLUK22.LIB	17XXA-202 FORTRAN

PROGRAMMING IN BASIC

The routines can be used from Interpreted BASIC, by using the LINK statement as follows:

```
LINK "SPIO" <RETURN>
```

To use the routines in Compiled BASIC, use the FIND command in the Linking Loader, as follows:

```
F BASIC <RETURN>
```

Description

The module consists of two routines: SPGETS and SPSETS. SPGETS is called to get a "snapshot" of the inputs and outputs. A port number between 0 and 9 is specified, along with an integer word into which the status is to be stored by SPGETS. When it is called, SPGETS retrieves the current input and output status and returns the value to the caller in the specified integer variable. The state of each line is represented by a corresponding bit in the status word. The bit assignments are illustrated below.

SPSETS sets the state of the output lines. A port number between 0 and 9 is specified, along with an integer word which contains the control data to be output to the control lines. The bit assignments in the control word are identical to the assignments in the status word. The input portion of the status/control word is ignored by this function.

Usage

CALL SPGETS(port%, status%)

INTEGER port%

0 for SP0:
1 for SP1:
.....
9 for SP9:

status%

RS-232 input/output line status		
bit		
value	pin	description
1	20	Data Terminal Ready
2	19	Secondary Request To Send
4	11	Undefined
8	4	Request To Send
128	-	RS-422 Output Enable
256	12	Sec Rcv Line Sig Detector
512	22	Ring Indicator
1024	8	Rcv Line Signal Detector
2048	6	Data Set Ready
4096	5	Clear To Send

CALL SPSETS(port%, status%)

INTEGER port%

0 for SP0:
1 for SP1:
.....
9 for SP9:

status%

RS-232 output line control		
bit		
value	pin	description
1	20	Data Terminal Ready
2	19	Secondary Request To Send
4	11	Undefined
8	4	Request To Send
128	-	RS-422 Output Enable

Errors

All errors are recoverable:

- 5100 FDOS function call failed (usually means option missing or faulty)
- 5101 invalid port number

External Effects

The RS-232 output lines may be changed. Turning off Request To Send inhibits transmission on the corresponding port.

FORTRAN PROGRAMMING

From FORTRAN, the routines are called in the same manner as they are in BASIC, except that an error parameter is passed as follows:

CALL SPGETS (port, status, error)

CALL SPSETS (port, status, error)

If the error value returned is non-zero, then one of the following has occurred:

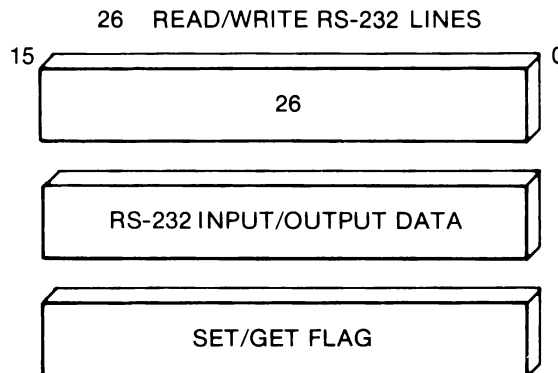
No device driver in FDOS
Option missing or faulty
Illegal port number specified

ASSEMBLY LANGUAGE PROGRAMMING

The FDOS driver for the option supports all the applicable functions, which are listed below. For a description of the call conventions, refer to the FDOS direct I/O functions for KBx: ports in the Assembly Language Manual. The functions are:

- 0 Read a Record
- 2 Write a Record
- 4 (unused)
- 6 Initialize Driver
- 8 Get Port Configuration
- 10 Set Port Configuration
- 12 Return Number of characters/lines in Input Buffer
- 14 Get a Character
- 16 Put a Character
- 18 Send a Break
- 20 (unused)
- 22 (unused)
- 24 Return Number of Characters in Output Buffer

Full control of the UARTs on the Dual Serial Interface option requires one additional FDOS call, described below.



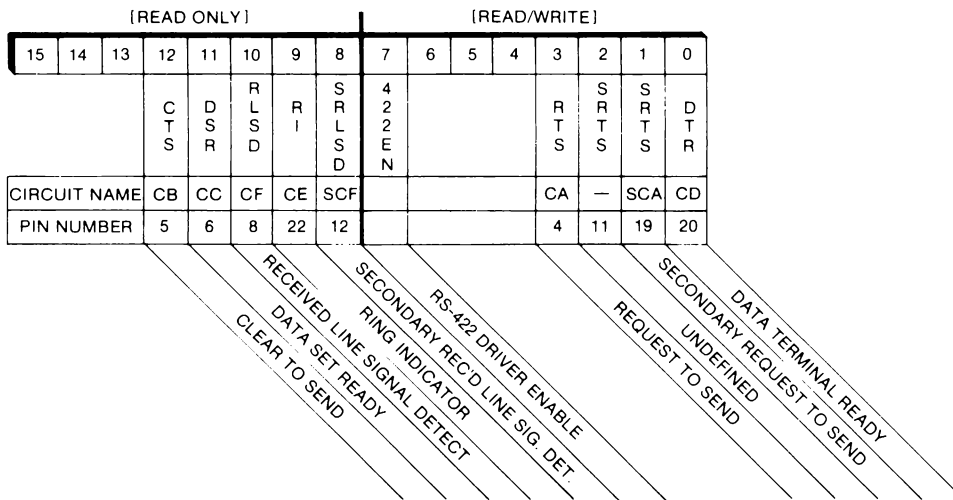
Description

This function either sets or gets the the current state of the RS-232 signal lines. When the Set/Get flag is non-zero, the lower byte of the RS-232 data word is loaded into the latches that drive the RS-232 lines. When the set/get flag is zero, the entire RS-232 data word is returned with the current state of the RS-232 interface, including both input and output lines.

In addition, this word also controls the RS-422 data driver. When the 422EN bit is set to one (1), the RS-422 driver is active. When 422EN is set to zero (0), the RS-422 output drivers are tri-stated.

The set function has no effect on the input signal lines. They are always sampled directly during a get function.

Format of RS-232 Data Word



1722A USER INFORMATION

1760A and 1761A DISK DRIVE SYSTEMS

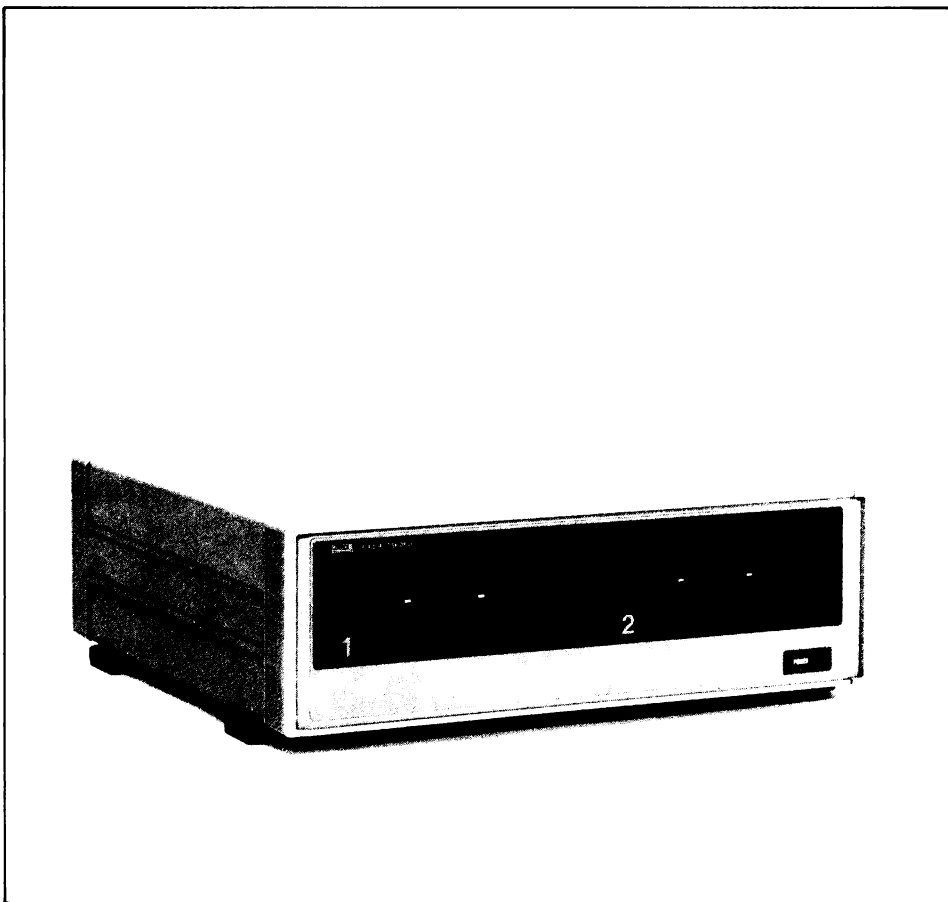
WARNING

This equipment generates and uses radio frequency energy and if not installed and used in accordance with the instruction manual, may cause interference to radio and television reception. It has been tested and found to comply with the limits for a Class B computing device pursuant to Subpart J of Part 15 of FCC Rules, which are designed to provide reasonable protection against such interference in a residential installation. If this equipment does cause interference, which can be determined by turning the the equipment off and on, the user is encouraged to try to correct the interference by relocating the equipment with respect to the receiver or plugging the computer into a different outlet so that the computer and receiver are on separate branch circuits.

INTRODUCTION

The Fluke 1760A and 1761A Disk Drive Systems provide additional floppy disk mass storage for the 1722A Instrument Controller. They connect to the Controller's IEEE-488 connector, and are recognized and addressed as additional logical devices MF1: - MF4:.

In addition to the internally mounted floppy disk drive (MF0:), up to four other drives can be connected to the Controller. The Fluke model 1760A is a single 5-1/4' unit, and the 1761A has two drives. When they are installed, they act similar to the internal drive; operation over the IEEE-488 bus is transparent to the user.



PRE-INSTALLATION CHECKOUT

The Disk Drive System is carefully inspected at the factory before shipment. Remove it from the shipping carton and inspect it for any signs of physical damage that might have occurred during shipment. If you find anything wrong, notify the nearest Fluke Service Center immediately and file a claim with the carrier.

INSTALLATION

These procedures describe how to install the Disk Drive System. There are two main steps:

- Setting the proper line voltage.
- Setting the IEEE-488 address.

Setting the Line Voltage

The Disk Drive System is shipped configured to the line voltage specified on the order form. If no voltage is specified, the factory ships the drive to operate on 120V ac, 47 to 63 Hz. If that is the voltage you will be using, skip this procedure, and go on to the next step, setting the IEEE-488 address. This procedure is only required if the voltage setting is different than that required, but it just takes a moment to check, so we recommend that you check the voltage as part of the incoming inspection.

1. Place the drive on a suitable work surface with the rear facing you, and with the line cord disconnected.
2. Slide the clear plastic cover (A) to the side, so that it covers the line plug connector.
3. Look at the wafer (B) at the top of the AC line connector. It shows the voltage for which the drive is configured. If the number shown differs from the voltage at the installation site, remove the wafer from its slot, and re-install it so that the correct number shows. This may be either 100, 120, 200, or 240.
4. Slide the cover back over the wafer, and double check that the correct voltage is still visible.

NOTE

Changing the wafer's position changes the voltage configuration. No frequency conversion is needed.

5. Check the fuse (C) against this table, and replace it for the correct value if needed.

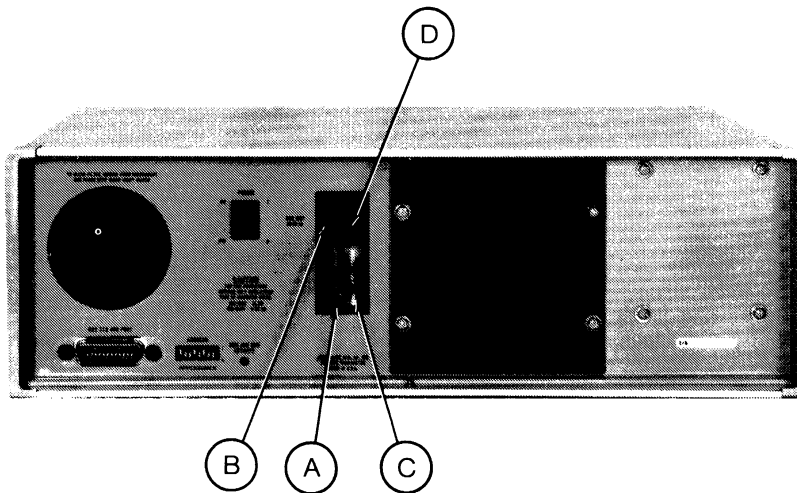
Voltage Setting	Fuse Rating
100 to 120	1 ampere, 250 volt, slow blow
200 to 240	1/2 ampere, 250 volt, slow blow

6. Check the power cord supplied with the Disk Drive System to ensure that it matches the receptacle. If it does not match, contact your sales representative to obtain the correct cord.

WARNING

If it is necessary to replace the power cord, the replacement must have the same polarity as the original. If it does not, a shock hazard exists, and the Disk Drive System may be damaged.

7. Plug the AC line plug into the connector (D) at the rear of the Disk Drive System. Do not apply power until the complete installation procedures have been done.



Selecting the IEEE-488 Address

After the power requirements have been taken care of, the next step is to connect the Disk Drive System to the Controller's IEEE-488 port.

1. Disconnect power from the Controller and the Disk Drive System.
2. Remove the Single Board Computer module from the Controller (slot 7). Check configuration switch S1 to be sure that position 5 and 6 are set to the OFF position. Doing so makes the Controller the System Controller rather than an idle controller. Only a System Controller can manipulate the bus control lines to initialize the disk drive.
3. Replace the Single Board Computer Module into slot 7, and reattach the Controller's rear cover.
4. Locate the switch labeled "Device Address" on the back of the Disk Drive System. Set it to IEEE-488 Address 20, Parallel Poll Address 4.

PARALLEL POLL ADDRESS			IEEE-488 ADDRESS					
1	2	3	4	5	6	7	8	
1			1		1			1
	0	0		0		0	0	0

NOTE

If the Controller is installed in a system, make sure that no other connected equipment is set to Address 20, because bus contention problems might occur, in which case neither the drive nor the other device will operate properly. Also make sure that the Controller itself is not set to address 20.

If a second Disk Drive System is connected to the Controller, it must be set to Device Address 21, Parallel Poll Address 5. (Switches ON: 1, 3, 4, 6, and 8.)

5. Attach the Disk Drive System IEEE-488 connector to the IEEE-488 connector on the Controller's Single Board Computer module. If you need to order the cable, see the table below:

Fluke P/N	Length
Y8021	1 meter
Y8022	2 meters
Y8023	4 meters

TESTING THE DISK DRIVE SYSTEM

At this point, all the steps have been completed to prepare for using the new Disk Drive System. To begin using it, remember that it provides new logical devices: MF1:, and so on. Also be sure to format floppy disks prior to use, as they are not formatted at the factory prior to shipment.

WARNING

Whenever the Controller is accessing the floppy disk drive, do not press the RESTART or ABORT buttons or enter <CTRL>/P. Doing so may damage the directory or data files on the disk. The disk is being accessed any time the red LED indicator is lit.

1. Connect the Disk Drive System to the Controller's IEEE-488 Port 0 connector, and power up both units.
2. Follow the instructions in Appendix G of the System Guide to run the diagnostic *MFXTST*. This test will check the disk drive speed and media detection logic, then seek, format, write, and read data back from a disk installed in each drive.

USING THE DISK DRIVE SYSTEM

There is a software device driver for the 1760A and 1761A that can be linked into the 1722A operating system (FDOS). The device driver allows the system software and applications programs to treat the Disk Drive System as another logical device in the system. The device names are MF1: and MF2: for a Disk Drive System at IEEE-488 Address 20, and MF3: and MF4: for a Disk Drive System at Address 21.

The operating system (FDOS2.SYS) supplied on the 1722A System Disk does not contain the device driver for the 1760A and 1761A. You must run the SYSGEN program to create a new operating system which includes this driver before you can use the Disk Drive System. Follow the instructions in Section 3 of the 1722A System Guide to reconfigure the operating system.

1722A USER INFORMATION

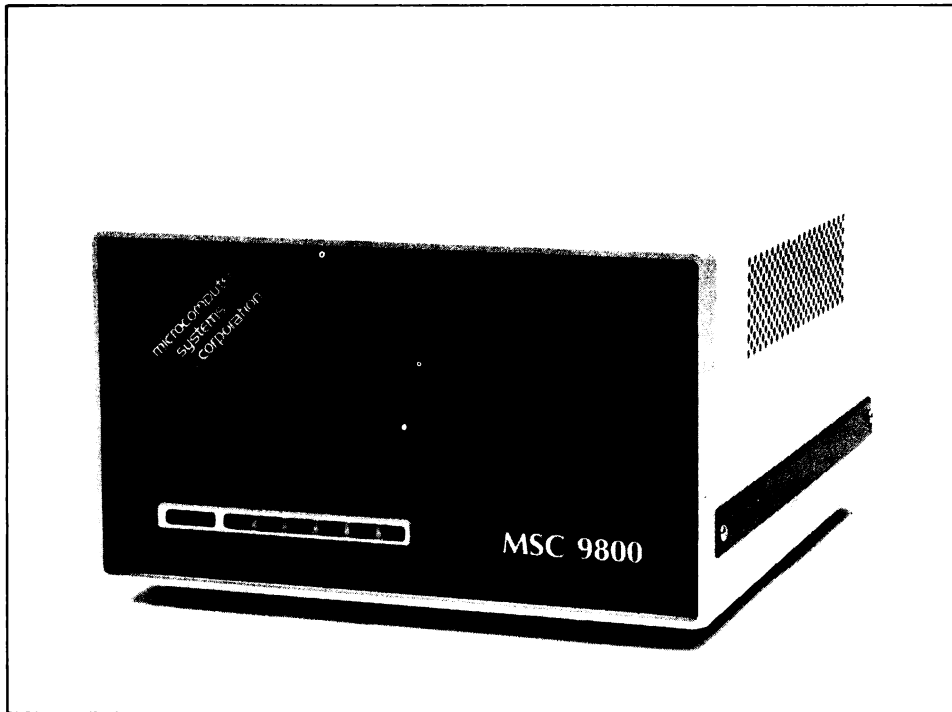
1765A/AB WINCHESTER DISK DRIVE

WARNING

This equipment generates, uses, and can radiate radio frequency energy and if not installed and used in accordance with the instruction manual, may cause interference to radio communications. It has been tested and found to comply with the limits for a Class A computing device pursuant to Subpart J of Part 15 of FCC Rules, which are designed to provide reasonable protection against such interference when operated in a commercial environment. Operation of this equipment in a residential area is likely to cause interference in which case the user at his own expense will be required to take whatever measures may be necessary to correct the interference.

INTRODUCTION

The Fluke 1765A/AB Winchester Disk Storage System provides 10 megabytes of mass storage for the 1722A Instrument Controller. It connects to the Controller's IEEE-488 connector, and is recognized and addressed as four logical devices: WD0:, WD1:, WD2:, and WD3:.



PRE-INSTALLATION CHECKOUT

The disk drive is carefully inspected at the factory before shipment. Remove it from the shipping carton and carefully inspect it for any signs of physical damage that might have occurred during shipment. If you find any damage, notify the nearest Fluke Service Center immediately and file a claim with the carrier.

These procedures describe how to install the disk drive. There are two main steps:

- Setting the proper line voltage.
- Setting the IEEE-488 address.

Setting the Line Voltage

The 1765A/AB is shipped configured to the line voltage specified on the order form. If no voltage is specified, the factory ships the drive to operate on 120V ac, 47 to 63 Hz. If that is the voltage you will be using, skip this procedure, and go on to the next step, setting the IEEE-488 address. This procedure is only required if the voltage setting is different than that required, but it just takes a moment to check, so we recommend that you check the voltage as part of the incoming inspection.

1. Place the drive on a suitable work surface with the rear facing you, and with the line cord disconnected.
2. Look at the wafer (A) at the top of the AC line connector. It shows the voltage for which the drive is configured. If the number shown differs from the voltage at the installation site, remove the wafer from its slot, and re-install it so that the correct number shows. This may be either 100, 120, 200, or 240.

NOTE

Changing the wafer's position changes the voltage configuration. No frequency conversion is needed.

3. Check the fuse (B) against this table, and replace it for the correct value if needed.

Voltage Setting	Fuse Rating
100 to 120	2 ampere, 250 volt, slow blow
200 to 240	1 ampere, 250 volt, slow blow

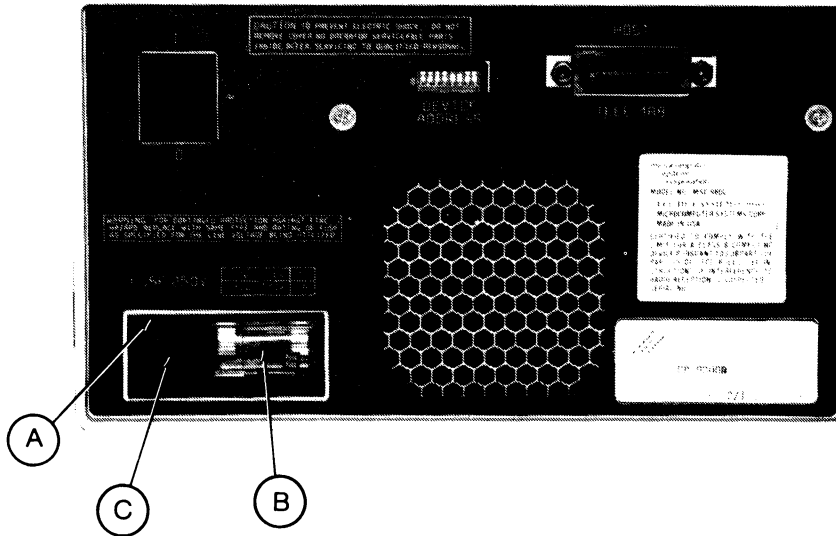
4. Check the power cord supplied with the disk drive to ensure that it matches the receptacle. If it does not match, contact your sales representative to obtain the correct cord.

WARNING

If it is necessary to replace the power cord, the replacement must have the same polarity as the original. If it does not, a shock hazard exists, and the disk drive may be damaged.

5. Plug the AC line plug into the connector (C) at the rear of the disk drive. Do not apply power until the complete installation procedures have been done.

User Information
1765A/AB Winchester Disk Drive



Selecting the IEEE-488 Address

After the power requirements have been taken care of, the next step is to connect the 1765A/AB to the Controller's IEEE-488 port.

1. Disconnect power from the Controller and the disk drive.
2. Remove the Single Board Computer module from the Controller (slot 7). Check configuration switch S1 to be sure that positions 5 and 6 are set to the OFF position. Doing so makes the Controller the System Controller rather than a Controller in Charge. Only a System Controller can manipulate the bus control lines to initialize the disk drive.
3. Replace the Single Board Computer Module into slot 7, and reattach the Controller's rear cover.
4. Locate the switch labeled "Device Address" on the back of the 1765A/AB. Set it to IEEE-488 Address 12. The orientation of the switch differs between revisions of the Disk Drive System, but the back panel makes clear the required direction to place a switch ON switch settings. When a switch is set to 1, it is ON (closed). For Address 12, switches 5 and 6 should be ON, and all others OFF as shown:

1	2	3	4	5	6	7	8	1
0	0	0	0	1	1	0	0	0

NOTE

If the Controller is installed in a system, make sure that no other connected equipment is set to Address 12, because bus contention problems may occur, in which case neither the drive nor the other device will operate properly. Also make sure that the Controller itself is not set to address 12.

5. Attach the disk drive IEEE-488 connector to the IEEE-488 connector on the Controller's Single Board Computer module. If you need to order a cable, see the table below:

<u>Fluke P/N</u>	<u>Length</u>
Y8021	1 meter
Y8022	2 meters
Y8023	4 meters

Testing the Hard Disk Drive

At this point, all the steps have been completed to prepare for using the new hard disk. To begin using it, remember that it provides four new logical devices: WD0: - WD3:. There is no need to format the disk prior to use, as it has been formatted at the factory prior to shipment.

WARNING

Whenever the Controller is accessing the Winchester disk drive, do not press the RESTART or ABORT buttons or enter <CTRL>/P. Doing so may damage the directory or data files on the disk. The disk is being accessed any time the red LED indicator is lit.

1. Connect the drive to the Controller's IEEE-488 Port 0 connector, and power up both units.
2. Follow the instructions in Appendix G of the System Guide to run the diagnostic WDXTST. This test program performs two functions:

Self Test This program sends a self-test command to the disk controller inside the 1765A/AB. When it receives this command, the 1765A/AB performs an internal disk controller test and communications test.

Verify This program checks each sector on the disk for errors, and reports any errors to the display.

3. When the program has been selected, four options will be presented on the screen:

Option 1 - Self Test

Option 2 - Verify

Option 3 - Self Test and Verify

Option 4 - Exit (Exits to FDOS.)

4. Select the option desired. For new installations, Option 3 is probably the most useful. If the message “correctable ECC error” is displayed during the Verify operation, press <RETURN> to continue the test. This indicates that a software correctable error occurred. Any other errors indicate a faulty unit, and should be reported to the local Fluke Service Center.

Using the Winchester Disk Drive

There is a software device driver for the 1765A/AB that can be linked into the 1722A operating system (FDOS). The device driver allows the system software and applications programs to treat the Winchester Disk Drive as four additional logical devices in the system. The device names are WD1:, WD2:, WD3: and WD4:.

The operating system (FDOS2.SYS) supplied on the 1722A System Disk does not contain the device driver for the 1765A/AB. You must run the SYSGEN program to create a new operating system which includes this driver before you can use the Winchester Disk Drive. Follow the instructions in Section 3 of the 1722A System Guide to reconfigure the operating system.

Appendices

CONTENTS

A	Specifications	A-1
B	Interface Connector Pinout	B-1
C	Sample BASIC Programs	C-1
D	Test Points	D-1
E	Signal Glossary	E-1
F	Using the Parallel Interface With a 1720A Instrument Controller	F-1
G	Performance Testing	G-1
H	Schematics	H-1

Appendix A

Specifications

HARDWARE SPECIFICATIONS

Ports	Two independent 16-bit parallel ports, 25-pin D-type subminiature female pin connectors.
Logical Interface	Memory mapped, two memory locations per port: data and status/control.
Line Characteristics	Data I/O lines are terminated resistively with diode input protection, 2400 to +5V, 5000 to ground.
Line Sense	Independent jumper-configurable active sense level for each control line, and for input and output data lines.
Data Out	Low: <0.4V @ 48 mA High: >2.4V @ -0.4 mA
Data In	Low: <0.8V. High: >2.0V.
Control Out	Low: <0.4V @ 8 mA. High: >2.4V @ 400 uA
Control In	Low: <0.4V High: >2.4V
Control Lines (each port)	PCTRL (output) PFLAG (input) PDIR (output) POEN (input)
Temperature	0 - 55 C storage 10 - 40 C operating
Dimensions	190.5 mm x 254.0 mm x 1.6 mm (7-1/2" x 10" x 1-1/16")
Supplied Cable	1.98 m (6.5 ft) of 25 conductor 26-gauge stranded copper wire.

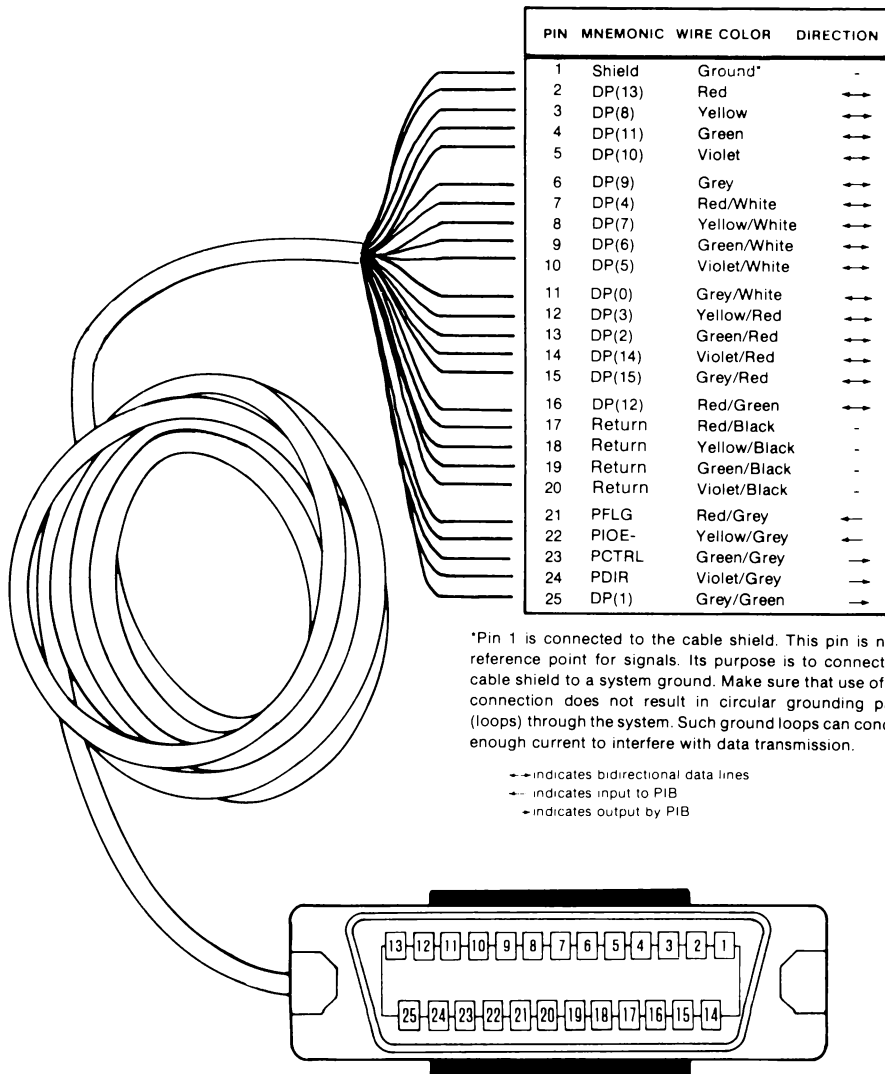
SOFTWARE SPECIFICATIONS

Drivers	Supplied on System Software Disk: Read/Write Bit Read/Write Words (16 bits) Read/Write Blocks (multiple words)
Subroutine Library	Supplied on System Software Disk: chkbit - Check a bit clrbit - Clear a bit setbit - Set a bit rdwr - Read a word wtwr - Write a word rdbl - Read a block wtbl - Write a block frdbl - Fast read a block fwtbl - Fast write a block popen - Open a port pclose - Close a port
Control Modes	Mode 0: No Handshake Mode 1: Input Handshake Mode 2: Full Output Handshake Mode 3: Strobe Output Handshake
Handshake Modes	No Handshake Full Handshake Input Strobe Input Handshake Output Strobe Output

Appendix B

Interface Connector

This appendix illustrates the interface connector and cable. For reference, it repeats some of the information from Section 3, Interface Description.



Appendix C

Sample BASIC Programs

INTRODUCTION

In this section are several sample programs, taken from actual applications. In each case, the system to which the Parallel Interface is connected is described in the text. Some of the programs are introduced by a block diagram, followed by the program description and listing. The program listings include enough comments to show how the program accomplishes the task it was designed to do. The programs are:

1. Loop-Around Tests

By connecting the ports to one another, these test programs send data out of one port and back into the other. The transfer can be observed by scoping one data line at a time, or the entire 16-bit data word can be watched by placing a logic analyzer in the loop.

2. Pulse Measurement Program

This short program times the length of an incoming pulse and displays the pulsewidth in milliseconds.

3. Digital Sine Wave Generator

This program uses an A/D converter and a low-pass filter to generate a sine wave whose frequency is that of the program's tightest loop, and whose varying amplitude is a function of the bit pattern output by the PIB.

4. Parallel Interface Test Circuit

A small test circuit and this sample program vividly demonstrate the operation of the Parallel Interface hardware and software.

5. Radix Converters

Since it is often necessary to convert between numbering systems, two programs are included that perform these tedious tasks.

PROGRAM 1: LOOP PROGRAMS

These two programs demonstrate how a simple loop between the two ports on a Parallel Interface module can be accomplished. The first one uses Mode 1, Handshake Output, to synchronize the data flow. The second program uses the BASIC language ON PPORT statement to generate an interrupt each time a word is received by Port 1.

These two programs operate properly only when the two ports are electrically connected. Either order the test cable, Fluke P/N 632968, or prepare an equivalent. To make your own cable:

1. Connect the data lines. Pins 2 through 16 and pin 25 on each connector go to the matching pin numbers on the other connector.
2. Connect the RETURN lines. Pins 17 and 20 on each connector go to the matching pin numbers on the other.
3. Reverse the connections of PCTRL and PFLG. Pin 21 on each connector goes to Pin 23 on the other.
4. Do not connect pin 22 (PIOE) or pin 24 (PDIR). These programs do not use the signals carried on these pins.

Appendix C
Sample BASIC Programs

```

10 :           * Loop With Handshaking *
20 :
30 LINK 'PIBLIB'           : link in assembly language drivers
40 PCLOSE(0%)\PCLOSE(1%)  : make sure the ports are closed
50 POPEN(0%, 1%, -1%, 0%) : open port 0 in handshake in mode
60 POPEN(1%, 2%, 0%, 0%)  : open port 1 in pulse output mode
70 A%=0%                  : initialize
80 FOR I% = 1% TO 2000%   : for-next loop
90   WTWRD(1%, I%)        : write the loop counter to port 1
100  RDWRD(0%, A%)        : read the word from port 0
110  PRINT A%             : print it
120 NEXT I%               : repeat until 2000
130 END

```

```

10 :           * Interrupt-Driven Loop Program *
20 :
30 LINK 'PIBLIB'           : link in assembly language drivers
40 OFF PPORT(0%)          : disable interrupts
50 PCLOSE(0%)\PCLOSE(1%)  : make sure ports are closed
60 POPEN(0%, 256%+1%, -1%, 0%) : open port 0 in handshake in mode
70 POPEN(1%, 256%+2%, 0%, 0%) : open port 1 in pulse output mode
80 A%=0%                  : initialize array
90 ON PPORT(0%) GOTO 170   : enable interrupts
100 FOR I% = 1% TO 2000%  : for-next loop
110   WTWRD(1%, I%)        : write the loop counter to port 1
120 NEXT I%               : continue until 2000
130 END                   : cease
140 :
150 : - Interrupt Handler -
160 :
170 RDWRD(0%, A%)         : read the word at port 0
180 PRINT A%             : print it out
190 RESUME               : go back, output next word

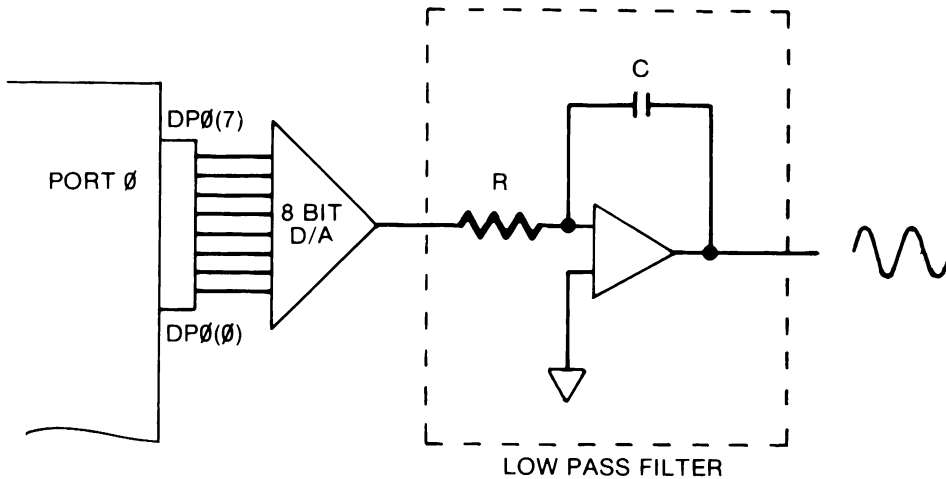
```

PROGRAM 2: PULSE MEASUREMENT PROGRAM

```
10 ! This program measures the length of an active high signal on port 0.
20 !
30 ON ERROR GOTO 265 \ ON CTRL/C GOTO 270
40 !
50 PRINT "Program PULSE"
60 !
70 LINK 'PIBLIB'           ! link in the PIB subroutines
80 PCLOSE(0%) \ PCLOSE(1%) ! close both ports
90 POPEN(0%, 0%, -1%, 0%) ! open port 0, no handshake,
!                          ! input on all bits, timeout disabled.
110 BL% = 0%               ! initialize bit to zero
120 BT% = 0%               ! check for pulse on data line 0
130 PT% = 0%               ! use port zero
140 BO% = 0%               ! initialize to zero
150 CHKBIT(PT%, BT%, BO%) ! check the bit
160 IF BO%=0% THEN GOTO 150 ! if low, check again
170 CHKBIT(PT%, BT%, BO%) ! looking for 0 -> 1
180 IF BO% THEN GOTO 190 \ GOTO 170 ! loop until bit transition is seen
190 T1 = TIME              ! find pulse start time
200 CHKBIT (PT%, BT%, BO%) ! looking for 1 -> 0 transition
210 IF BO% THEN GOTO 200 \ GOTO 220 ! still high? y-keep checking; n-go on
220 T2 = TIME              ! find pulse end time
230 TI = ((T2 - T1)/1000) ! calculate pulsewidth
240 !
250 PRINT "Pulse length: "; TI; "seconds"
260 GOTO 150
265 PRINT "Error number";ERR;"occured on line";ERL
270 PCLOSE(0%)
280 END
```

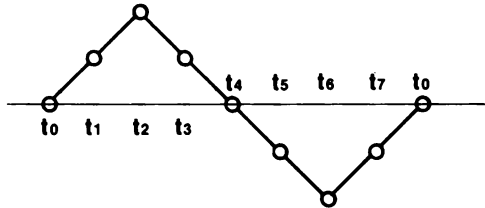
PROGRAM 3: DIGITAL SINE WAVE GENERATOR

This circuit uses the lower 8 bits to generate a sine wave. The program listing which follows this discussion shows that the length of time between writing one word to the port and writing the next word is dependent on how long the program takes to perform the loop at program lines 220 and 230.



Appendix C
Sample BASIC Programs

The illustration below shows how a sine wave is built. The table gives the word values the PIB sends to the D/A converter at regular intervals. The D/A converter outputs a sine wave whose frequency (about 75 Hz.) is proportional to the speed of the PIB output. The amplitude is a direct function of the value of the data word.



time	Word Value
t0	127
t1	191
t2	255
t3	191
t4	127
t5	64
t6	0
t7	64

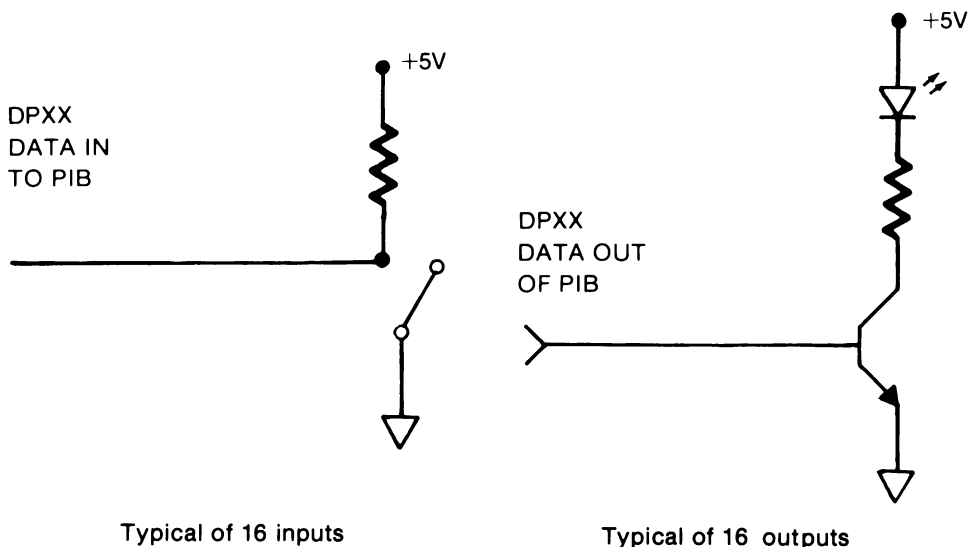
```

10 ON ERROR GOTO 260 \ ON CTRL/C GOTO 270
20 |
30 PRINT "Program: Sine Wave Generator"
40 |
50 LINK 'PIBLIB'           ! link in the PIB drivers
60 PCLOSE(0%)             ! make sure port is closed
70 BL% = 0%                ! initialize bit to zero
80 MD% = 0%                ! mode 0: no handshake
90 PT% = 0%                ! port 0
100 TM% = 0%              ! no timeout
110 DM% = 0%              ! direction mask - all bits output
120 POPEN(PT%, MD%, DM%, TM%) ! open port 0; no handshake;
130 |                       ! output on all bits; timeout disabled.
140 DIM AX(7%)            ! dimension the array
150 |
160 |                       * Set Up Values in Array *
170 |
180 AX(0%)=127% \ AX(1%)=191% \ AX(2%)=255% \ AX(3%)=191%
190 AX(4%)=127% \ AX(5%)= 64% \ AX(6%)= 0% \ AX(7%)=64%
200 |
210 |                       * Output Binary Numbers to D/A Converter *
220 FOR I%=0% TO 7% \ WTWRD (PT%, AX(I%)) \ NEXT I% \ GOTO 210
230 |
240 |                       * Error Handler *
250 |
260 PRINT "Error number"; ERR; "occurred on line"; ERL;
270 PCLOSE(0%)
280 END

```


PROGRAM 4: PARALLEL INTERFACE TEST CIRCUIT

In this example, the system designer wanted to test whether the PIB was outputting the proper data patterns, and whether it was properly receiving the input data from a bank of switches. A technician built a circuit that would light LEDs for Port 0 (entirely output), and that would input to Port 1 a 5-volt level for any of 16 switches that were turned on. The drawing below shows the circuitry that was used for one input and one output data line.



Since no handshaking was to be used, the PCTRL and PFLG lines were left disconnected on both ports.

A test fixture such as the one described here can be useful in testing not only the Parallel Interface, but also programs in development. In this case, the fixture was built on a 6"x 12" piece of perf board with wire-wrap sockets for the switches and LEDs. The actual test fixture used 74LS00 ICs for debouncing the switches.

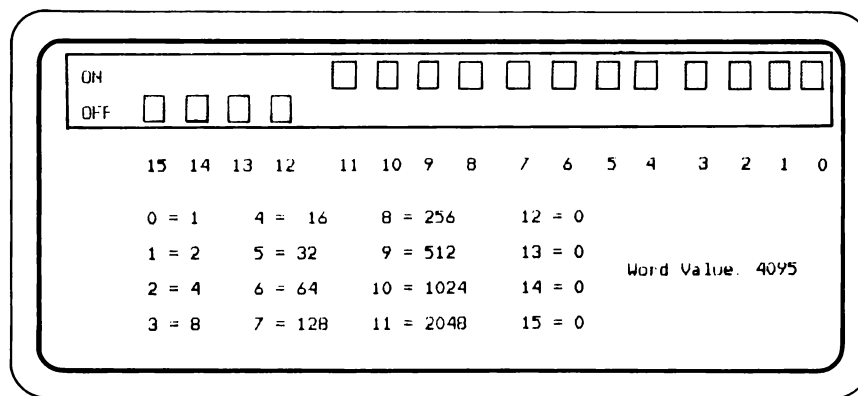
A program was written to use the test fixture to observe the interface conditions. After first asking whether to check the input or output, it branches to one of two subprograms.

Input Test Subprogram

This portion of the program reads the word value of the input port, and displays it in three ways:

- As a graphical indication of the switch position.
- As a table showing the value of each switch.
- As the decimal integer of the input data lines that results with various combinations of the switches on.

Here is a sample of the screen appearance when the input test is selected:



Output Test Subprogram

The output test performs a “walking bit” pattern up and down the 16 LEDs connected to the output port. After asking how fast to walk the bit (in milliseconds), it generates the various word values needed to light first the LED on DP0, then DP1, and on down the line to DP15. It counts in powers of two.

Test Fixture Program

```

10  : Filename:  PARAL.BAS
20  : Version 0.3
30
40  :***** PROGRAM DESCRIPTION *****
50
60  : This program uses the 17XXA-002 Parallel Interface and a test fixture
70  : to check the operations of the Parallel Interface and the routines
80  : in the PIB Library. Port 0 is used for an input, and Port 1 as
90  : an output.
100
110 : If "Output" is selected, the program asks how fast to turn
120 : on and off the LEDs on the test fixture (time in milliseconds).
130
140 : If "Input" is selected, the program displays the positions of each of
150 : the switches connected to Port 0, as well as the resultant word value.
160
170
180
190 :***** SET UP: DISPLAY MODES *****
200
210 ES$ = CHR$(27)+"["      : escape initialization
220 EK$ = ES$ + "4p"       : enable keyboard
230 DK$ = ES$ + "5p"       : disable keyboard
240 NS$ = ES$ + "p"        : normal size characters
250 DS$ = ES$ + "1p"       : double size characters
260 DC$ = ES$ + "2p"       : disables graphics mode
270 EC$ = ES$ + "3p"       : enables graphics mode
280 US$ = ES$ + "4m"       : underscore
290 BE$ = ES$ + "5m"       : blink
300 AO$ = ES$ + "m"        : attributes off
310 IV$ = ES$ + "7m"       : reverse video
320 HL$ = ES$ + "1m"       : increased intensity
330 CH$ = ES$ + "2J" + ES$ + ";H" : clear screen & home cursor
340 CB$ = ES$ + "1;4;5m"   : combination
350
360 FF$=CHR$(12)           : form feed
370 LF$=CHR$(10)          : line feed
380 BP$=CHR$(7)           : beep
390
400 :***** SET UP: ERASE MODES *****
410
420
430 ED$ = ES$ + "J"        : from active position to end of screen
440 EU$ = ES$ + "1J"       : from start of screen to active position
450 EP$ = ES$ + "2J"       : page
460 EE$ = ES$ + "OK"       : from active position to end of line
470 EF$ = ES$ + "1K"       : from first of line to active position
480 EL$ = ES$ + "2K"       : line
490 EC$ = ES$ + "D " + ES$ + "D" : last character
500
510
520
530 :***** PARALLEL INTERFACE TEST PROGRAM *****
540
550 ON CTRL/C GOTO 1920
560 LINK 'PIBLIB'         : link to the library
570 POX=1%                : initialize port 1 - output
580 PNZ=0%                : initialize port 0 - input
590 MDX=0%                : port mode - no handshake
600 MSX=0%                : direction mask - bidirectional
610 TMX=2000%            : timeout after 2 seconds
620
630 PCLOSE (POX)          : close both ports as insurance
640 PCLOSE (PNZ)
650 POPEN (POX, MDX, MSX, TMX) : now open them
660 POPEN (PNZ, MDX, MSX, TMX)
670 AX = 0%               : initialize A to hold word value
680 DIM BX(15%), B1%(15%) : arrays to hold bit numbers
690
700 PRINT CH$; CPOS(3, 25); "* PARALLEL INTERFACE TEST PROGRAM *"
710 PRINT CPOS(3, 25); "Please Select Test - Input or Output"
720 PRINT BP$; CPOS(6, 25); "Indicate I -or- O "; \INPUT A$
730 IF UCASE$(A$) = "I" THEN 1030
740 IF UCASE$(A$) = "O" THEN 810 ELSE 720

```

Appendix C
Sample BASIC Programs

```

750 |
760 |
770 | ***** SELECTION : OUTPUT *****
780 | !
790 | - marches a single bit back and forth - all bits -
800 |
810 | PRINT CPOS(8,25); "Output Selected."
820 | PRINT CPOS(9,25); "How Fast To Scan (milliseconds)"; \INPUT BTX
830 | WTWRD(POX,AX) | clear all bits
840 | FOR IX=0% TO 15% | set sequence
850 | CALL SETBIT(POX,IX) | output the specific bit
860 | WAIT BTX | wait for specified time
870 | CLRBIT(POX,IX) | clear the bit
880 | NEXT IX | next bit
890 |
900 | IX=14% | set the first return bit
910 | CALL SETBIT(POX,IX) | output the specific bit
920 | IF IX=0% GOTO 830 | check for last bit
930 | WAIT BTX | if not, wait
940 | CLRBIT(POX,IX) | now clear the bit
950 | IX=IX-1% | decrease bit # by one
960 | GOTO 910 | forever
970 |
980 |
990 | ***** SELECTION: INPUT *****
1000 |
1010 | - reads the states of all 16 switches on the input port-
1020 |
1030 | RDWRD(PNX,AX)\BX=0% | read port n, load value into int.A\initialize B
1040 |
1050 | FOR IX = 0% TO 15% | for-next loop to read each switch
1060 | CHKBIT (PNX,IX,BX) | check bit(at port n, bit #, place into B)
1070 | BX(IX) = BX | load the array
1080 | NEXT IX
1090 |
1100 | FOR IX=0% TO 15% | copy the array
1110 | BIX(IX)=BX(IX) | into a help file
1120 | NEXT IX | for later comparison
1130 |
1140 | * - Make the Display - *
1150 | PRINT EP$ | first clear the screen
1160 | CX = 77% | initial cursor position
1170 | FOR IX = 0% TO 3% | first four switches
1180 | IF BX(IX)=0% THEN LX=4% ELSE LX=3% | check if bit is on or off
1190 | GOSUB 1870 | draw first four switches
1200 | CX = CX - 1% | decrement cursor position
1210 | NEXT IX
1220 | CX=CX-2% | move cursor left
1225 |
1230 | FOR IX=4% TO 7% | draw next four switches
1240 | IF BX(IX)=0% THEN LX=4% ELSE LX=3%
1250 | GOSUB 1870
1260 | CX=CX-1%
1270 | NEXT IX
1280 | CX=CX-2%
1285 |
1290 | FOR IX=8% TO 11% | draw the next four
1300 | IF BX(IX)=0% THEN LX=4% ELSE LX=3%
1310 | GOSUB 1870
1320 | CX=CX-1%
1330 | NEXT IX
1340 | CX=CX-2%
1345 |
1350 | FOR IX=12% TO 15% | now draw the last four
1360 | IF BX(IX)=0% THEN LX=4% ELSE LX=3%
1370 | GOSUB 1870
1380 | CX=CX-1%
1390 | NEXT IX
1400 |
1410 | PRINT CPOS(3,2); HL$; "ON"; CPOS(5,2); HL$; "OFF"; AD$; | label on & off
1420 | PRINT EG$; | character graphics
1430 | PRINT CPOS(1,1); "1"; DUPL$(56%,76%); "0";
1440 | FOR I = 2 TO 5
1450 | PRINT CPOS(I,1); "9"; CPOS(I,78); "9"; | draw box
1460 | NEXT I | around switches
1470 | PRINT CPOS(6,1); "3"; DUPL$(56%,76%); "2";
1480 | PRINT DG$; | disable graphics
1485 |
1490 | PRINT CPOS(7%,9%); "15 14 13 12"
1500 | PRINT CPOS(7%,27%); "11 10 9 8" | under each switch,
1510 | PRINT CPOS(7%,46%); "7 6 5 4" | print its bit position
1520 | PRINT CPOS(7%,64%); "3 2 1 0"

```

Appendix C
Sample BASIC Programs

```

1530 !
1540 PRINT CPOS(9%,6%); "0 ="; BOX(0%); ! draw a table showing
1550 PRINT CPOS(11%,6%); "1 ="; BOX(1%); ! the switch positions
1560 PRINT CPOS(13%,6%); "2 ="; BOX(2%); ! and their values
1570 PRINT CPOS(15%,6%); "3 ="; BOX(3%);
1580 !
1590 PRINT CPOS(9%,19%); "4 ="; BOX(4%);
1600 PRINT CPOS(11%,19%); "5 ="; BOX(5%);
1610 PRINT CPOS(13%,19%); "6 ="; BOX(6%);
1620 PRINT CPOS(15%,19%); "7 ="; BOX(7%);
1630 !
1640 PRINT CPOS(9%,32%); "8 ="; BOX(8%);
1650 PRINT CPOS(11%,32%); "9 ="; BOX(9%);
1660 PRINT CPOS(13%,31%); "10 ="; BOX(10%);
1670 PRINT CPOS(15%,31%); "11 ="; BOX(11%);
1680 !
1690 PRINT CPOS(9%,45%); "12 ="; BOX(12%);
1700 PRINT CPOS(11%,45%); "13 ="; BOX(13%);
1710 PRINT CPOS(13%,45%); "14 ="; BOX(14%);
1722 PRINT CPOS(15%,45%); "15 ="; BOX(15%);
1730 !
1740 !
1750 PRINT CPOS(11%,60%); "Word value: "; AX; ! print the word value
1760 !
1770 FX = 0% ! initialize flag
1780 FOR IX=0% TO 15% ! check each bit for a change
1790 CHKBIT(PNZ, IX, BX)
1800 BOX(IX)=BX ! reload B, compare to help file
1810 IF BOX(IX)(>)BX(IX) THEN IX=15\FX=1% ! if different, set flag
1820 NEXT IX
1830 IF FX=0% THEN 1770 ELSE 1030 ! otherwise keep checking
1840 !
1850 ! * - Subroutine to Draw the Switches - *
1860 !
1870 PRINT CPOS(LX,CX); AOS; \CX=CX-3% ! this subroutine uses
1880 PRINT CPOS(LX,CX); IVS; \LX=LX+1%\CX=CX+3% ! two inverse video blocks
1890 PRINT CPOS(LX,CX); AOS; \CX=CX-3% ! to represent the switches
1900 PRINT CPOS(LX,CX); IVS;
1910 RETURN
1920 !
1930 PRINT CH$ ! clear screen, home cursor
1940 PCLOSE(PO%) ! close both ports
1950 PCLOSE(PNZ)
1984 END !

```

PROGRAM 5: RADIX CONVERSION PROGRAMS

Effective use of the PIB sometimes requires conversion between numbering systems. The programs here illustrate how to do some of these conversions, and with minor changes they can be made subroutines in other programs.

BCD-Decimal

The BCD number required by this program must be expressed in terms of the decimal value of each digit. The program does not accept the binary digits themselves; e.g., enter '9' instead of 1001.

```
10 ON CTRL/C GOTO 250
20
30 * BCD to Decimal Conversion Program *
40
50 This program converts a BCD number in the range 0 - 9999
60 to its decimal equivalent.
70
80 PRINT "Please enter a BCD number"
90 INPUT BC          ! variable to hold binary code
100 IN = BC          ! keep input number
110 DE = 1000        ! decimal position
120 A = 15           ! conversion up to 2^15
130 BP = 2 ^ A       ! bit position
140 BI = 0           ! initialize variable
150 FOR X = 1 TO 4   ! for a 4-digit number
160   SC = 0         ! initialize variable
170   FOR Y = 1 TO 4 ! for each bit
180     BP = 2 ^ A   ! bit position
190     IF (BC / BP) > 1 THEN BC = BC - BP \ SC = SC + 2^(4 - Y)
195     ! check if bit is there
200     A = A - 1    ! decrement the bit position
210   NEXT Y
220   BI = SC * DE + BI ! calculate bit value
230   DE = DE / 10     ! calculate next decimal position
240 NEXT X            ! get next digit
250 PRINT IN: 'dec = 'BI; 'bcd'; ! print values
260 GOTO 30          ! ask again
```

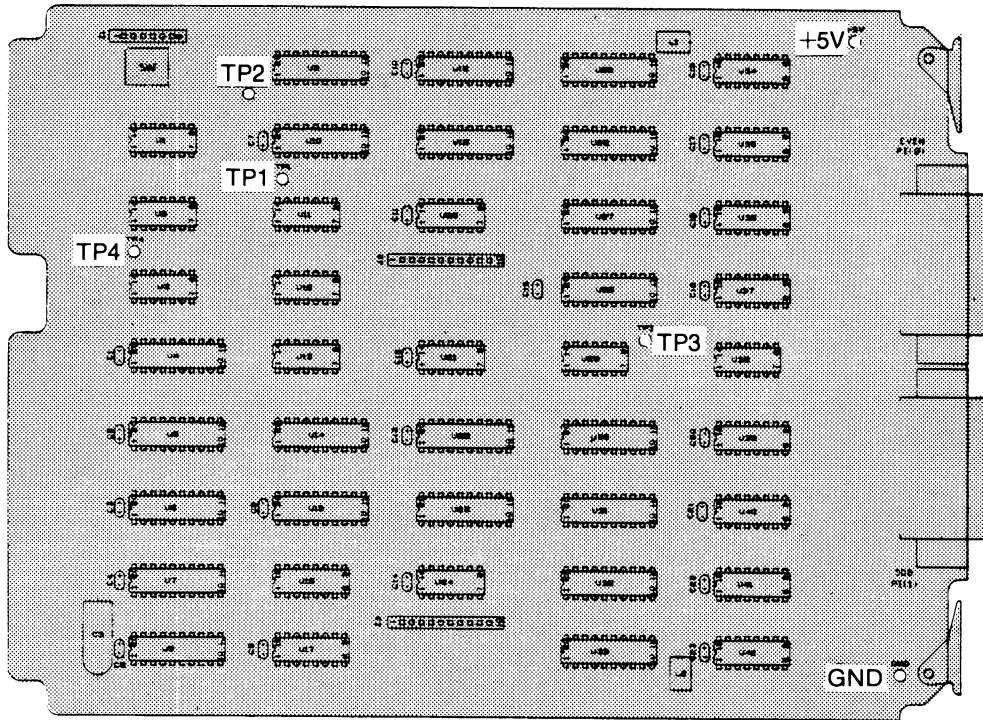
Decimal Integer-Binary

```
10 ! * Decimal to Binary Conversion Program *
20 !
30 PRINT "ENTER A DECIMAL NUMBER: "; \ INPUT DX
40 PRINT "Decimal", "Binary"
50 B$ = RAD$(DX, 2)
60 BB$ = DUPL$("0", 16-LEN(B$)) + B$
70 PRINT DX, BB$
```

Appendix D

Test Points

This appendix illustrates the test points on the Parallel Interface module. The table following the illustration describes the measurement conditions and the signals that should be observed at each of the test points.



TEST POINT	NAME	DESCRIPTION
1	BRDSL T	Board Select Signal. Active (low) when Parallel Interface is selected by software.
2	RESET	Active (low) when device is first powered on.
3	INTP	Interrupt; active (high) when interrupts are enabled and an interrupt is generated.
4	SYNC	The system clock; a 6 MHz square wave at all times.

Appendix E

Signal Glossary

This appendix gives the definitions of all Parallel Interface module signals. The table uses the convention “(0 or 1)” after signals that are duplicated on both ports.

SIGNAL NAME	DEFINITION
ADVAL-	Address Valid; indicates the address currently on the bus is valid.
AD 15-0	Address lines; the system Address Bus.
BUSWR-	Bus Write; Data Bus direction indicator.
RINT-	Refresh or Interrupt; gates an interrupt request onto the bus.
SYNC-	The system clock, 6 MHz.
DI 15-0	Data; the system Data Bus.
DCOK-	DC power is now ok. Clears the registers, buffers, and flip-flops on the PIB.
BRDSLCT-	Board Select; active when the system talks to the PIB.
R/W-	Read or Write; high indicates a read operation, low indicates a write.
WDATA(0 or 1)	Write Data Register of the indicated port.
WCTRL(0 or 1)	Write Control Register of the indicated port.
RDATA(0 or 1)	Read Data Register of the indicated port.
RSTAT(0 or 1)	Read Status Register of the indicated port.
ADACK-	Address Acknowledge; the PIB indicates to the CPU that the address has been received.

Appendix E

Sample BASIC Programs

INTEN(0 or 1)	Interrupt Enable. Allows interrupts to occur for the indicated port.
OUTEN(0 or 1)	Output Enable. Allows output drivers to go low.
SANITY(0 or 1)	Puts state machine for the indicated port into a known state.
DIR(0 or 1)	Direction. Indicates direction of data on the indicated port.
MODE0 1-0	Puts handshake state machine into 1 of 4 modes.
READY(0 or 1)	The indicated port is ready for next transfer.
INT(0- or 1-)	Interrupt Request from the indicated port.
ST(0 or 1) 3-0	State Value. The value of the current state (0, 1, 2, or 3) of the state machine of the indicated port.
DOPOL(0 or 1)	Data Output Polarity. Indicates to software the polarity of output data.
DIPOL(0 or 1)	Data Input Polarity. Indicates to software the polarity of input data.
PFLG(0 or 1)	Port Flag. Handshake signal input from device on the indicated port.
PCTRL(0 or 1)	Port Control. Handshake signal output to device on indicated port.
INTR(0 or 1)	Interrupt. The indicated port is ready and can generate an interrupt.
LO(0 or 1)	Latch Outputs; latches output data.
LI(0 or 1)	Latch Inputs; latches input data.
PDIRP(0 or 1)	Port Direction Polarity; inverts the sense of PDIR on the indicated port.
PFLGP(0 or 1)	Port Flag Polarity; inverts the sense of PFLG on the indicated port.

PCTRP(0 or 1)		Port Control Polarity; inverts the sense of PCTRL on the indicated port.
DO(0 or 1)	15-0	Data Output; the outputs of the Output Data Latch of the indicated port.
DN(0 or 1)	15-0	Data Input; the inputs to the Data Input Latch of the indicated port.
DP(0 or 1)	15-0	Port Data; the PIB bidirectional Data Bus at the indicated port.
POE(0 or 1)		Port Output Enable; external device output driver enable.
POEP(0 or 1)		Port Output Enable Polarity; inverts the sense of POE on the indicated port.

Appendix F

Using the Parallel Interface with a 1720A Instrument Controller

INTRODUCTION

The software disk supplied with the Option 17XXA-002 Parallel Interface contains the driver routines for operating the Parallel Interface installed in a 1720A Instrument Controller. This appendix documents the differences between this software disk and the software supplied on the System Disk shipped with the 1722A Instrument Controller.

There are also some minor differences in the hardware installation technique and in how to use the routines in a program. These topics are discussed here as well.

SOFTWARE

The Revision 4 Parallel Interface floppy disk supplied with the Option 17XXA-002 Parallel Interface is for use **ONLY** with the 1720A Instrument Controller. The disk contains several types of files:

- 1720A System Software files, including:

<u>NAME</u>	<u>VERSION</u>
BOOT	1.5
FDOS.SYS	1.8
COMMON.SYS	1.2
TIME.CIL	1.1
SET.CIL	2.0
FUP.CIL	1.3
BASIC	1.6

- Assembly language source code files containing the 1720A routines, which are a superset of those included in the object file PIBLIB.OBJ on the 1722A System Disk:

PSETUP.PRE (1720A initialization routine, described later.)
POPEN.PRE
PCLOSE.PRE
SETBIT.PRE
CLRBIT.PRE

CHKBIT.PRE
RDWRD .PRE
WTWRD .PRE
RDBLK .PRE
WTBLK .PRE
FRDBLK.PRE
FWTBLK.PRE

- Assembly Language source code files containing internal management routines. They are listed below, but should not be tampered with because they control the operation of the PIB routines.

CHKLIM.PRE	Checks that values are within legal limits.
F\$RGMY.PRE	Transfers arguments from BASIC calls.
F\$RNER.PRE	Aborts and returns error numbers.
PIBLIB.H	Header file of time constants.
PVARS .PRE	Controls structure and workspace.
TILRDY.PRE	Waits until the port is ready for the next data.

- Command Files which reassemble PIBLIB, and clean up the disk afterwards:

MAKE.CMD	Assembles pvars, psetup, popen, and pclose; calls pwrds.
PWRDS.CMD	Assembles wtwr, rdwr, setbit, clrbit, and chkbit; calls pblks.
PBLKS.CMD	Assembles wtblk, rdblk, fwtblk, frdblk; calls prntim.
PRNTIM.CMD	Assembles tilr, chkl, f\$rgmy, and f\$rner; calls plink.
PLINK.CMD	Links object files for the other command files to create piblib.obj.
CLEAN.CMD	Removes object files for the source.

- Sample BASIC Programs:
 - TEST.BAS A program which uses No Handshake mode to estimate data transfer speeds.
 - SLOW.BAS A program which uses two instrument controllers to transfer data using parallel interfaces.
 - FAST.BAS A program which uses two instrument controllers and the FRDBLK and FWTBLK routines to transfer data using parallel interfaces.
- These additional files:
 - PIBLIB.OBJ Object file for PIBLIB.
 - PIBLIB.MAP Map file for PIBLIB showing the size and relative location of the modules.
 - PIBTST.CIL Parallel Interface diagnostic program.

Rebuilding PIBLIB.OBJ

If a routine is added, modified, or deleted, follow these steps to rebuild PIBLIB:

1. Copy the 1720A Assembly Language package software to the device containing the PIBLIB source document.
2. Place the 1720A into COMMON.
3. Type:
 - make <RETURN> -or- make [*device*] <RETURN>where [*device*] is the FDOS device on which the listing is to be produced.
 - a. The Assembler and Preprocessor act on each *.pre* file to produce corresponding *.obj* files.

- b. Next, the Link program is called. It links the *.obj* files together, resolves all external references, and produces *piblib.obj* and *piblib.map*.
- c. The command files form a functional chain. As the functions of each file are successfully completed, the next command file is called. The command file chain is:

make → pwrds → pblks → prntim → plink

4. After *make.cmd* is successfully run, call *clean.cmd* to get rid of all the intermediate *.obj* files.
5. The E-Disk is the preferred system device for the command file operations since the system device is repacked after each assembly, link, and deletion.

PSETUP

The PSETUP routine is only for using the PIB with a 1720A and is a required command in all 1720A programs that use the Parallel Interface. It takes no arguments. The 1722A software has incorporated the setup function in the POPEN routine. Note that if programs written for the PIB in a 1720A are to be converted for use with a 1722A, all PSETUP statements must either be deleted or commented out.

The reference page for PSETUP follows this appendix, and may be incorporated at any convenient place in Section 4 of this manual.

Interrupt Handling

The 1720A Controller does not handle PIB interrupts. The BASIC language commands ON PPORT and OFF PPORT can only be used if the Parallel Interface is installed in a 1722A.

HARDWARE INSTALLATION

Be sure to note these differences between the hardware installation in a 1720A and 1722A.

1. The 17XXA-002 module is electrically compatible with both the 1720A and 1722A Instrument Controllers. However, because of the differences in case design, the 1720A must be operated with no rear card cage cover when the option is installed. This won't cause any problems except for slightly greater EMI susceptibility.
2. To maintain full FCC and VDE certifiability of the 1722A Controller, all I/O option modules are provided with a shield plate attached. If this plate interferes with other modules in a 1720A installation, it can be removed with no deterioration of performance.

ERROR MESSAGES

This table shows the error messages returned by the 1720A PIB software. All Parallel Interface errors are recoverable.

ERROR	MEANING	SOLUTION
1201	No Port There	The program specified a port number that does not exist. Be sure that the address switch setting matches the port number specified.
1202	Bad Hardware on Port	This error indicates either a probable hardware failure or no PSETUP statement in the program. IF PSETUP is OK, use the PIBTST diagnostic to confirm whether th hardware is faulty.
1203	Port Already Opened	The program has two POPEN statements.
1204	Port Not Opened	An attempt was made to read or write at a port that has not yet been opened. There is probably no POPEN statement in the program, or the wrong port was opened.
1205	Attempted Write to a Read-Only Bit	These errors may result from an incorrectly set Direction Mask, or from improper setup of the INP or OUTP jumpers.
1206	Attempted Write to a Read-Only Port	
1207	Port Timed Out	See the discussion on timeout in Section 4.
1220	Illegal Port Number	A port number greater than 15 was specified by the program.
1222	Illegal Bit Value	A bit greater than 15 was specified.

PSETUP

PIBLIB

Usage

BASIC: [CALL] PSETUP

FORTRAN: CALL PSETUP

Assembly Language:

```
ref psetup
blwp @psetup
data 0
```

Description

The port setup routine initializes the hardware of all Parallel Interfaces in the system. It is only used for the 1720A Instrument Controller, and must not be used in 1722A applications.

- PSETUP finds all the ports in the system.
- The routine insures that all the output drivers are off and that the hardware is in a passive state as far as external devices are concerned.
- The routine sets up and initializes internal tables.

NOTE

The PSETUP routine must be called before any other PIBLIB routines are called in a program, and should only be called once.

Parameters

none

See Also

POPEN

PSETUP PIBLIB

Example

Always use PSETUP prior to any other calls to PIBLIB in 1720A PIB programs:

```
10      PSETUP  
20      POPEN (0%, 0%, 0%, 0%)  
  
      .  
      .  
      .  
32000   PCLOSE (0%)  
32768   END
```

Possible Error Messages

If called by 1722A program: Undefined subroutine call.

If not called by 1720A program: 1202 Bad Hardware on Port

Appendix G

Performance Testing

INTRODUCTION

This Appendix describes how to use the PIB performance test, PIBTST. For 1720A Instrument Controllers, the test is included on the 1720A-002 floppy disk that is supplied with the 17XXA-002 option. For 1722A Instrument Controllers, PIBTST is included as part of the System Diagnostic Software disk. While the test is the same in both cases, its operation differs because the 1722A diagnostics are menu-driven, rather than command line oriented. The set-up procedure therefore is the same for both Controllers, but the actual test operation differs.

The performance test is designed to verify the overall operation of the Parallel Interface module, and can be used as an acceptance test and/or a periodic maintenance check.

Equipment

The equipment required for this test are the Instrument Controller, the appropriate floppy disk containing the test, and a Parallel Interface Test Cable.

Parallel Interface Test Cable

The purpose of the Parallel Interface Test Cable is to tie the two ports on the module together. This connection is essential to check for proper operation of handshake and port interface lines. The test cable also ensures an extra degree of confidence in the integrity of the data lines. Order the Parallel Interface Test Cable (Fluke Part Number 632968) from your nearest John Fluke Service Center.

Set-Up Procedure

Before running the performance test, the Parallel Interface module must be restored to the shipping configuration. Perform the following operations to ensure that the module is in the proper configuration for the performance test.

1. On the Instrument Controller, set the rear panel POWER to OFF.
2. Remove all Parallel Interface modules from the Instrument Controller.
3. Restore the jumpers to their original configuration. Refer to Section 2 for proper jumper configuration.
4. Restore the Parallel Interface module address switches to their original configuration. S3, S2, and S1 should be ON, and S4 should be OFF.
5. Attach the Parallel Interface Test Cable to both ports of the module to be tested.
6. Insert the Parallel Interface module to be tested into a spare slot. This can be any open slot in the 1720A, but must be either 1, 3, or 5 in the 1722A.
7. Set the rear panel POWER switch to ON.

Procedure for 1720A Instrument Controllers

The 1720A-002 floppy disk, provided with the Parallel Interface module, contains all the system software to run the 1720A Controller plus a software program to test the 1720A-002 module. Use this floppy disk to run the performance test on 1720A Controllers. The software reports the status of the module under test by flashing either PASS or FAIL on the display. Replace the parallel interface module if a FAIL indication is displayed.

To run the performance test complete the following procedure:

1. Insert the 1720A-002 floppy disk into the 1720A disk drive.
2. Complete the start-up procedure on the 1720A as described in the 1720A Instrument Controller User Manual.
 - a. Check for software compatibility using the software compatibility table in Section 3 of the 1720A User Manual.
 - b. Make sure that the 1720A passes all start-up self-tests.
 - c. Load the COMMON program into the 1720A (enter CTRL/P) with the system floppy disk mounted in the disk drive.
3. Enter PIBTST (RETURN). The 1720A loads and starts the 1720A-002 Self-test program.
4. The Parallel Interface module is faulty if the self-test indicates any of the following:
 - a. A blinking FAIL at the bottom of any column.
 - b. Board 0 is not tested (indicated by XXXX in column 0).
 - c. Any other of the 7 columns are tested (indicated with 0000 in appropriate column).
5. Allow the test to run through 100 cycles (total loops = 100 is displayed on the screen).
6. Touch the EXIT block displayed on the screen to terminate the self-test program.

Procedure for 1722A Instrument Controllers

This program tests up to three Parallel Interface Modules (Option 17XXA-002). The program performs three separate tests:

- Writing to a port and reading back from the same port (Readback)
- Writing to one port and reading back on the other port (Loopback)
- Interrupt test.

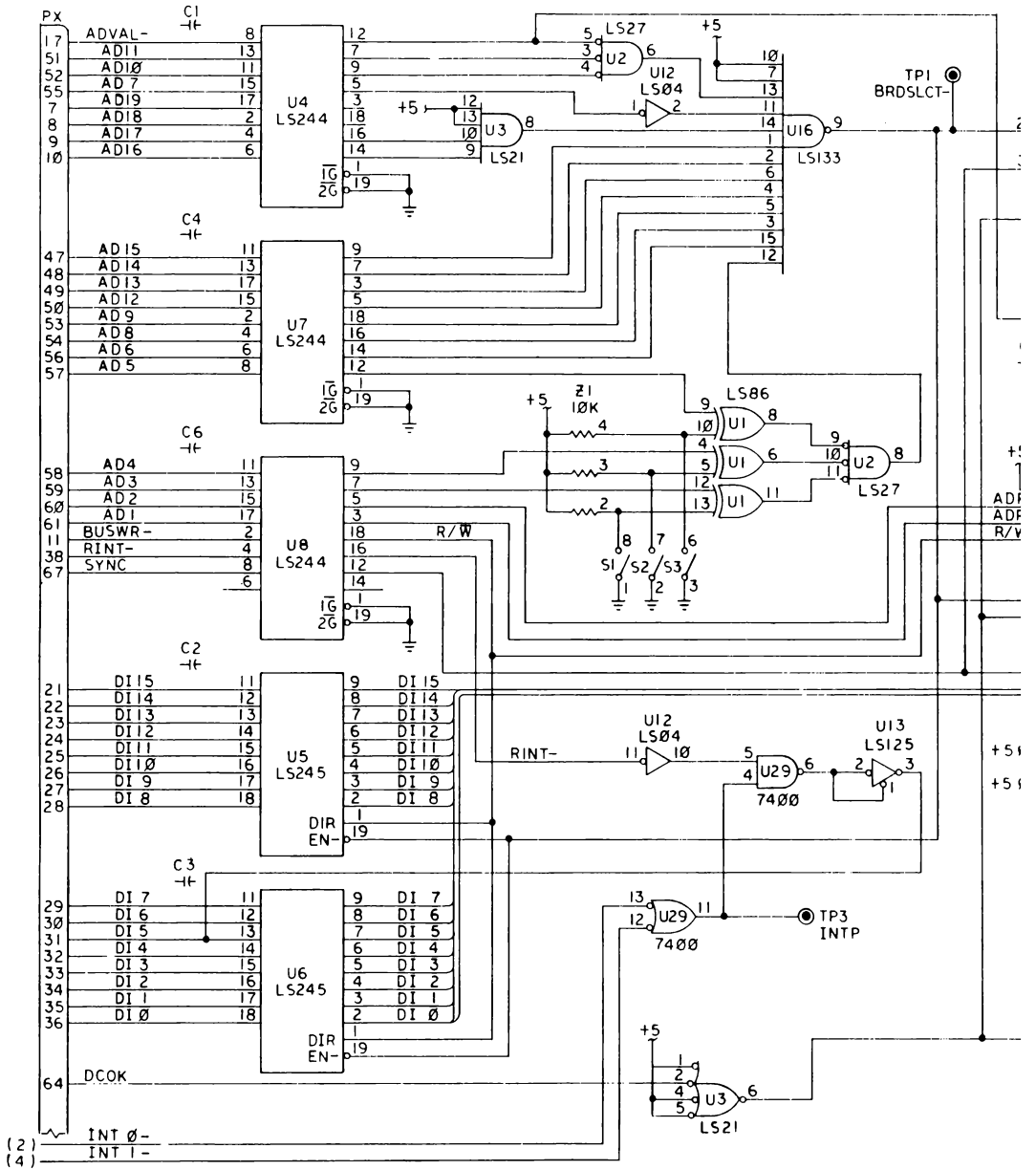
When the program runs, the numbers of the modules under test are displayed across the top of the screen and the tests that are executing are displayed down the left side of the screen.

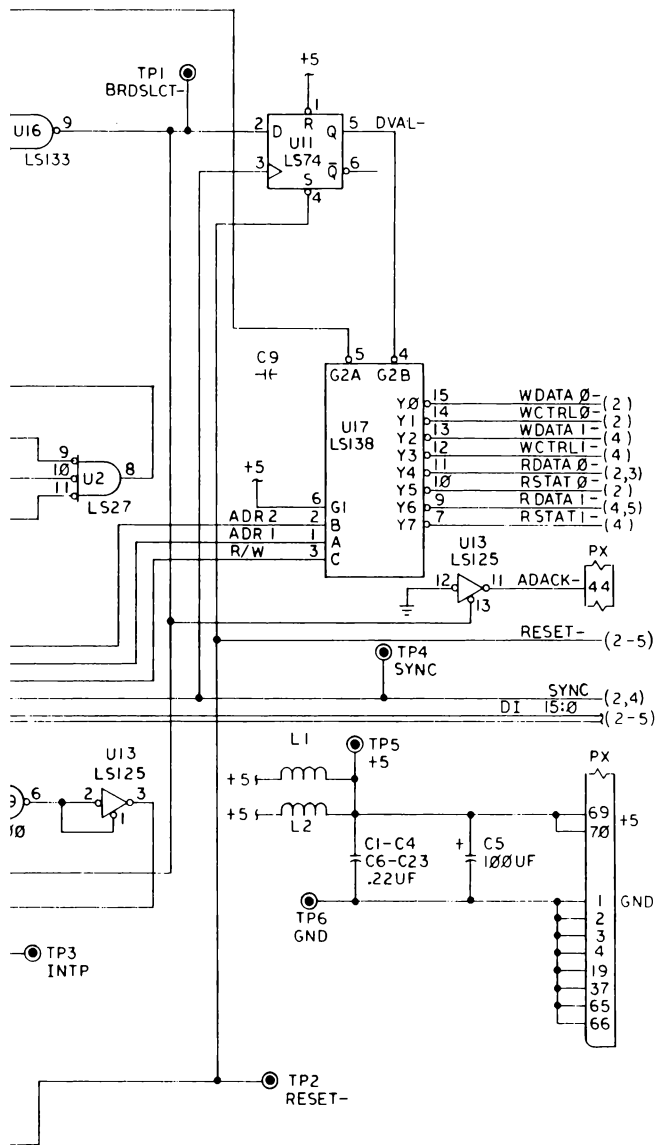
In order to pass the Loopback test, the Parallel Interface loopback test cable (JF/PN 632968) must be connected between the two ports on the module. If the test cable is not available, the Readback and Interrupt tests may be run individually by touching the PASS/FAIL block on the screen at the bottom of the column corresponding to the module under test. At this point a second menu is displayed. The user may run the tests individually on either port of the selected module by touching the screen at the appropriate point. ʹ

After performing the Set-Up Procedure described earlier, refer to Appendix G, System Diagnostic Software, in the 1722A System Guide for complete instructions on how to interact with the 1722A Diagnostics Software.

Appendix H

Schematics



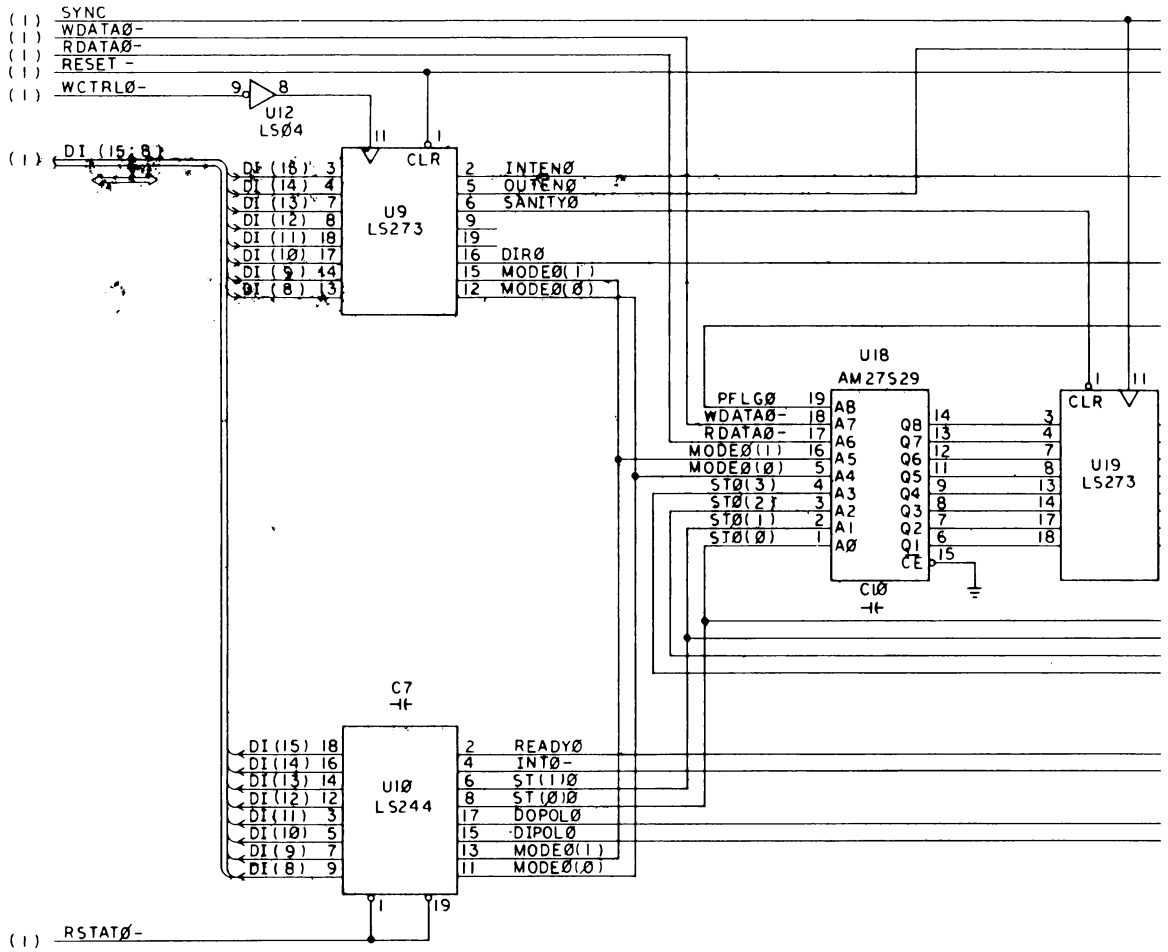


REFERENCE	
TEST POINTS	SPARE
TP1 BRDSELCT-	LS74 1/2 OF U11
TP2 RESET-	LS86 1/4 OF U1
TP3 INTP	LS125 2/4 OF U13
TP4 SYNC	LS04 2/6 OF U12
TP5 +5	LS27 1/3 OF U2
TP6 GND	LS00 2/4 OF U38

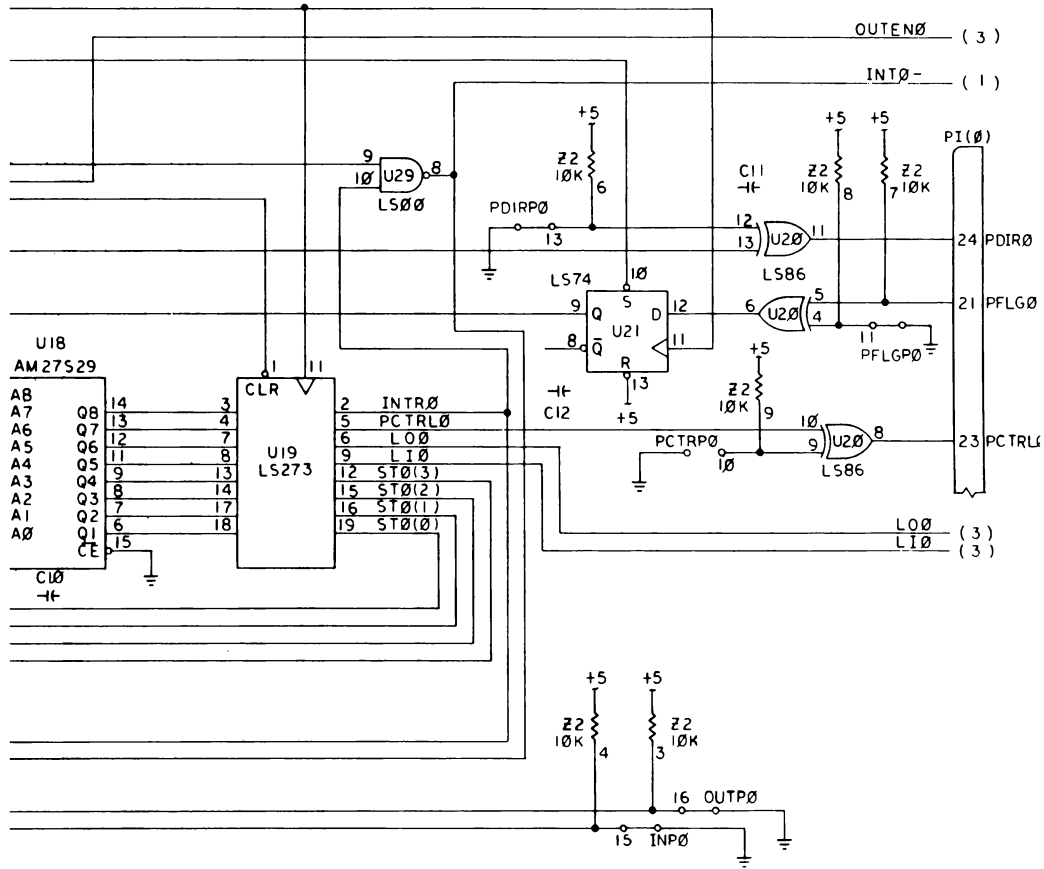
- NOTES-UNLESS OTHERWISE SPECIFIED:
- +5 AND GND ARE NOT SHOWN EXPLICITLY FOR MOST CHIPS. OTHER SUPPLY VOLTAGES ARE SHOWN EXPLICITLY.
 - SIGNALS TEND TO FLOW (ENTER) FROM LEFT AND EXIT FROM THE RIGHT.
 - (1)-SIGNAL (2-5) FROM SHEET TO SHEETS

Circuitry Common To Both Parts

PORT Ø STATUS/CONTROL

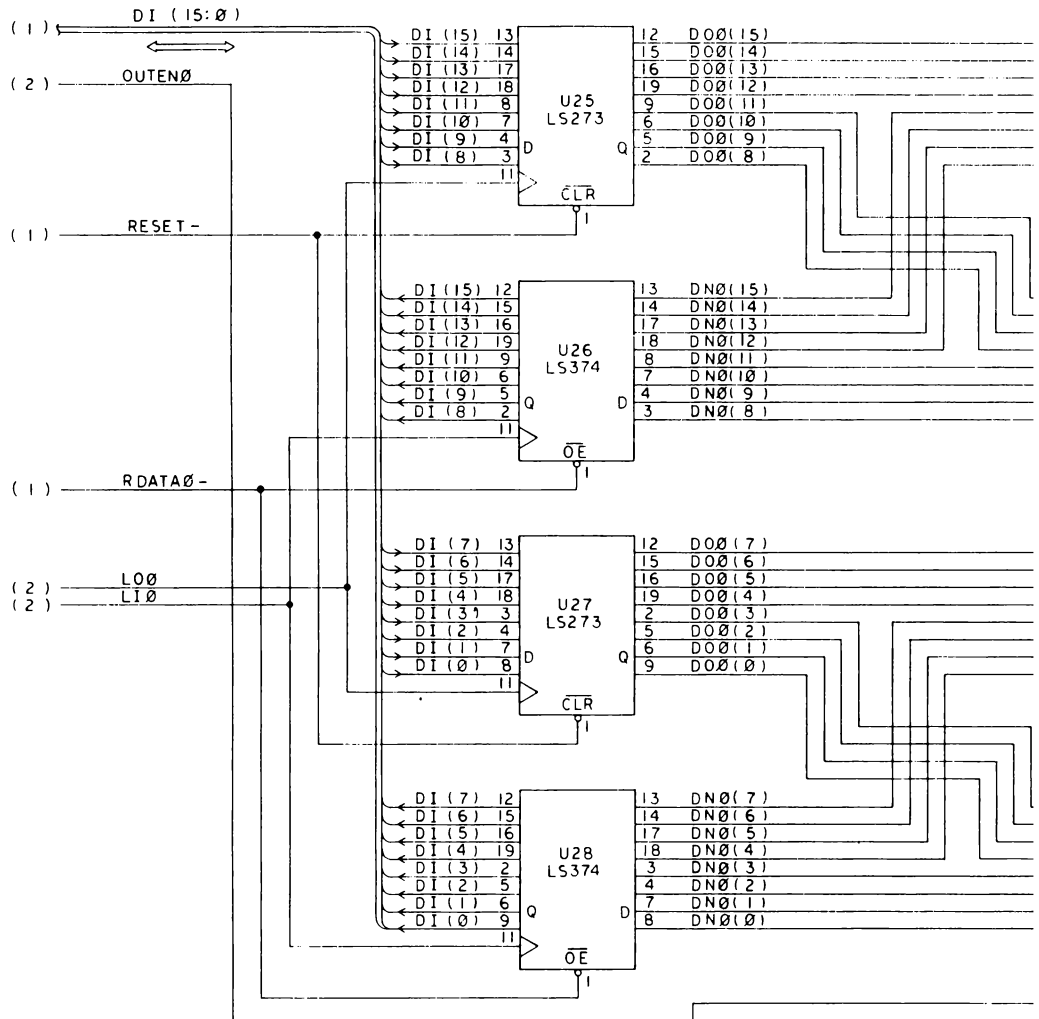


Γ Ø STATUS/CONTROL

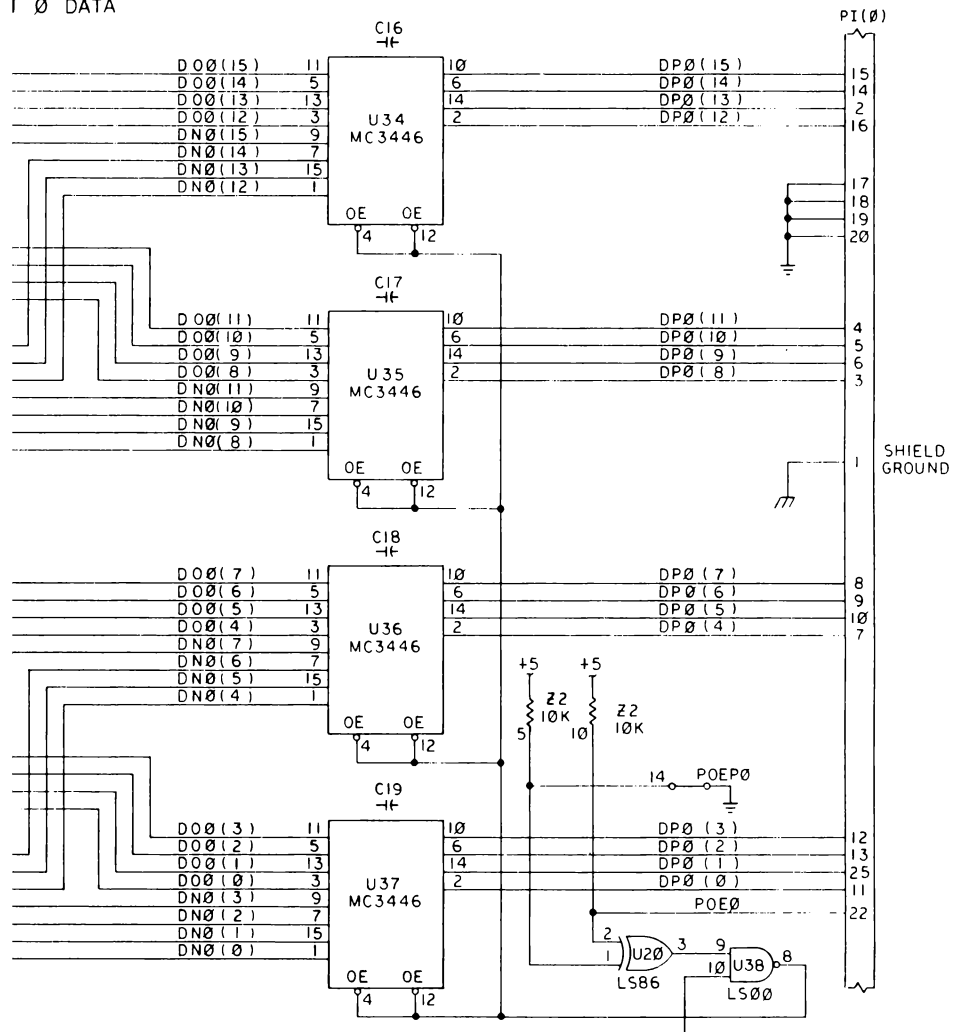


Port 0 Status/Control

PORT 0 DATA

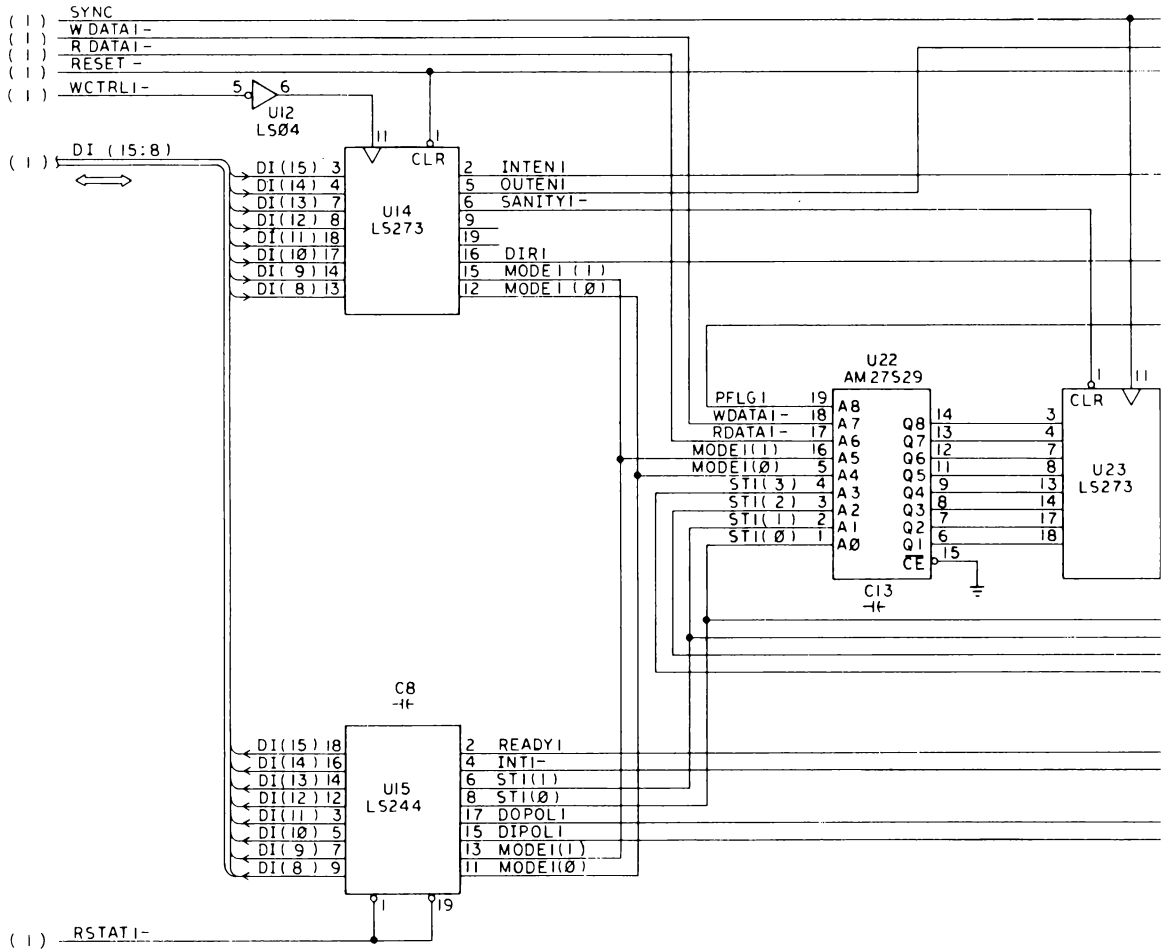


Γ Ø DATA

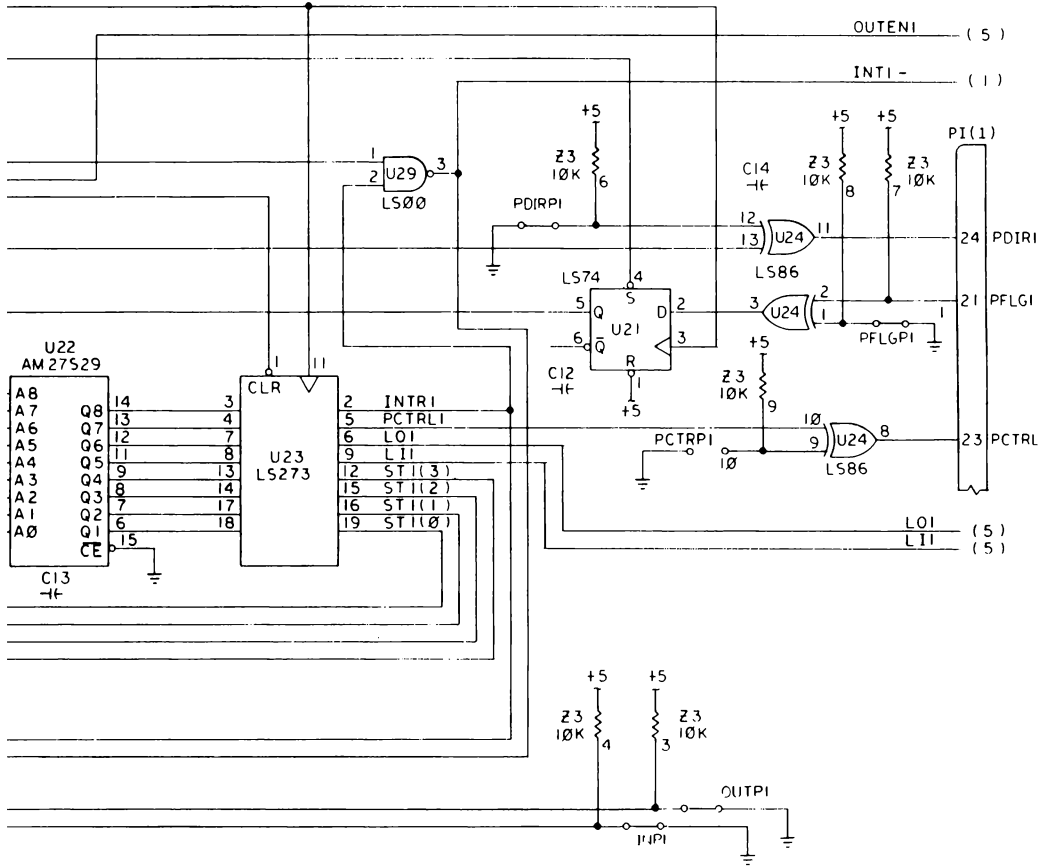


Port 0 Data

PORT I STATUS/CONTROL

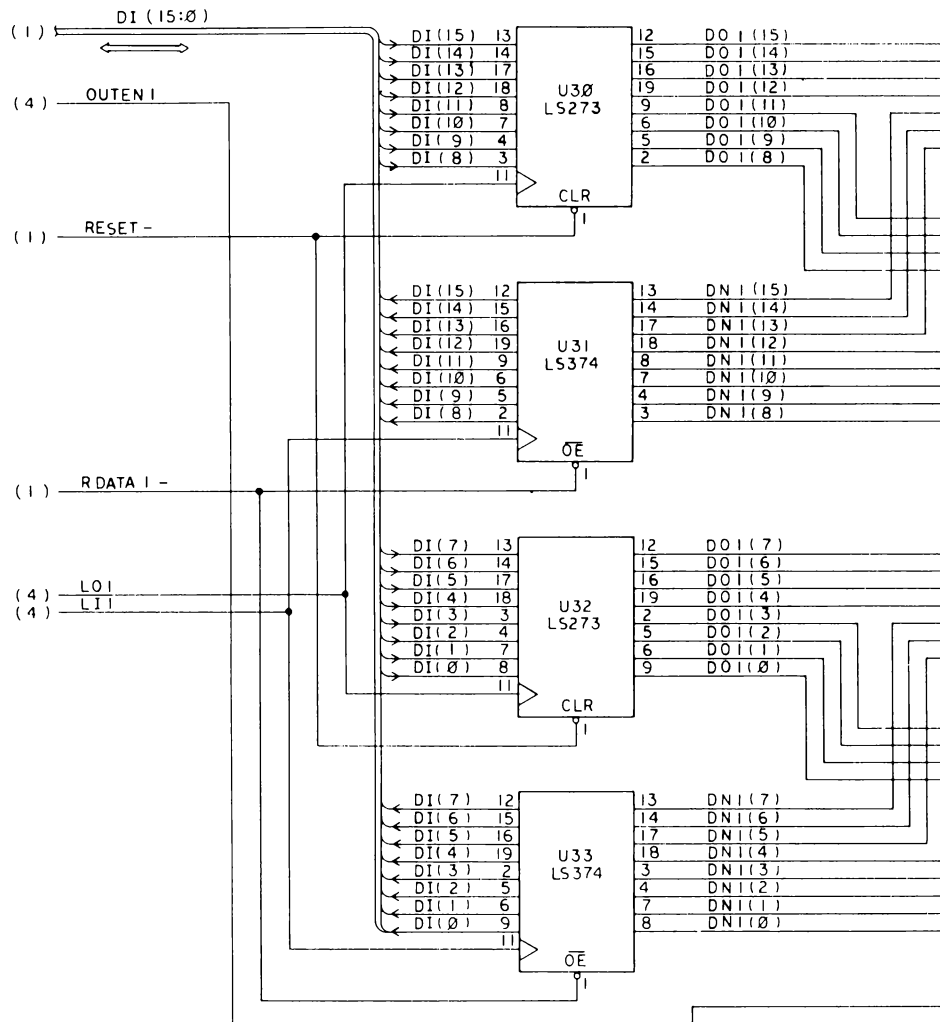


PORT 1 STATUS/CONTROL

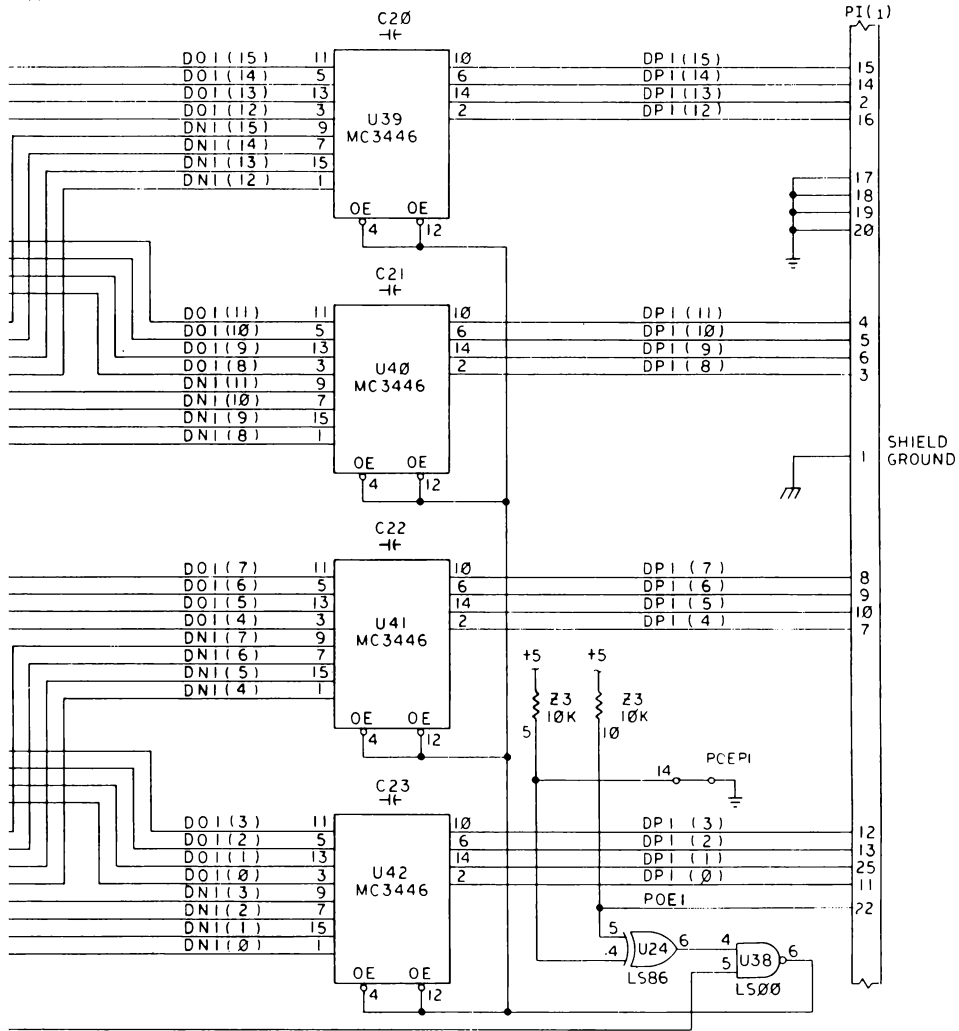


Port 1 Status/Control

PORT I DATA



ATA



Port 1 Data

INDEX

- Address Switch
 - location of, **2-4**
 - setting, **2-5**
- BIT Parameter, **4-5**
- Block Diagram, **5-9**
- BOOL Parameter, **4-4**
- CHKBIT, **4-9**
- CALL, **2-11**
- CLRBIT, **4-11**
- Configuration procedure, **2-5**
- Connector
 - interface, **3-4** , **B-1**
 - loopback, **2-10**, **C-2**
- Control
 - lines, **3-5**
 - jumpers, **3-6**
- Data
 - lines, **3-3** thru **3-5**
 - direction, **3-8**, **4-5**, **4-6**
- FRDBLK, **4-23**
- FWTBLK, **4-25**
- Getting Started, **2-11**
- Glossary (Signal names), **G-1**
- Handshake, **3-17**
 - example connections, **3-17**
 - Mode 0, No Handshake, **3-21**
 - Mode 1, Handshake Input, **3-23**
 - Mode 2, Handshake Output, **3-27**
 - Mode 3, Strobe Output, **3-29**
 - mode parameter, **4-2**
- Hardware
 - installation, **2-4** (for **1720A**, see **Apx. F**)
 - specifications, **A-1**
 - theory of operation, **5-1**
- HNDSHKIN (Mode 1), **3-23**
- HNDSHKOUT (Mode 2), **3-27**
- Integer Conversion
 - by a program, **4-7**, **C-12**
 - by hand, **4-6**
- Interface
 - circuits, **3-11**
 - description, **Section 3**
 - connector, **3-4** , **B-1**
- Interrupts
 - using in a program, **C-2**
 - ON PPORT, **4-8**
 - priority, **4-8**
 - how to enable/disable, **3-33**
- INP, **3-7**
- Inverted Handshake, **3-31**

Index

- Jumpers, 3-6
- Loopback
 - connector, 2-10, C-2
 - programs, C-3
- MC 3446A, 3-11
- Mode
 - defined, 3-17
 - No Handshake, 3-21
 - Handshake Input, 3-23
 - Handshake Output, 3-27
 - Strobe Handshake, 3-29
- MODE Parameter, 4-4
- No Handshake (Mode 0), 3-21
- OUTP, 3-7
- Parameters
 - defined, 4-4
 - Direction Mask example, 4-6
 - how to use, 4-4
- Partial Handshake Using Mode 3, 3-30
- Part Numbers
 - Option 17XXA-002, 2-3
 - loopback connector, C-2
 - 1722A Service Manual, 1-5
- Performance Testing, **Apx. G**
- PIBLIB, **Section 4**
- Port
 - control lines, 3-8
 - parameter, 4-4
 - partial schematic, 3-9
 - theory, 5-7, 5-8
- PCLOSE, 4-29
- PCTRL, 3-5
- PCTRLP, 3-7
- PDIR, 3-8
- PDIRP, 3-6
- PFLG, 3-5
- PFLGP, 3-7
- PIBLIB.OBJ, 4-1
- PIBTST, **G-1**
- POE, 3-5
- POEP, 3-7
- POPEN, 4-27
- Programs
 - digital sine wave generator, C-5
 - loop, C-2
 - pulse measurement, C-4
 - radix conversion, C-12
 - test fixture, C-7
- PSETUP (1720A Port Initialization), F-7
- RAD, 4-7, C-12
- RDBLK, 4-19
- RDWRD, 4-15
- Schematic, **Apx. H**
- SETBIT, 2-11, 4-13
- Software
 - compatibility, 2-7
 - diagnostics, **Apx. G**
 - FORTTRAN programs, 3-34
 - subroutines, 4-9
 - 1720A, F-1
- Software Routines
 - CHKBIT, 4-9
 - CLRBIT, 4-11
 - FRDBLOCK, 4-23
 - FWTBLOCK, 4-25
 - PCLOSE, 4-29
 - POPEN, 4-27
 - PSETUP (1720A), F-4, F-7
 - SETBIT, 4-13
 - RDBLOCK, 4-19
 - RDWORD, 4-15
 - WTBLOCK, 4-21
 - WTWORD, 4-17
- STROBEOUT (Mode 3), 3-29
- Specifications, **Apx. A**
- Switches, 2-5
- Test Points, **Apx. D**
- Timeout mechanism, 3-32

- TIMEOUT parameter, **4-5**
- Theory of Operation, **Sect. 5**
- Timing diagrams
 - terms defined, **3-19**
 - Mode 0 (No Handshake), **3-21**
 - Mode 1 (Handshake Input), **3-23**
 - Mode 2,(Handshake Output), **3-27**
 - Mode 3, (Strobe Output), **3-29**
- Strobe Output Partial Handshake, **3-30**
- Strobe Output Inverted Handshake, **3-31**
- Transceiver
 - circuits, **3-15, 3-16**
 - MC 3446A, **3-11**
- WTBLK, **4-21**
- WTWRD, **4-17**

