Applied Engineering
Telephone Numbers

Technical Support
(214) 241-6069
9 AM to 12:30 PM & 1:35 to 5 PM (CST)
Monday through Friday
Do not return any
product for service without a
Return Material Authorization (RMA) number.
An RMA number can be obtained by calling Technical Support.

Sales
(214) 241-6060
9 AM to 11 PM (CST) 7 days
I/O 32™

User's Manual
Limited Warranty & Disclaimer

Applied Engineering warrants the I/O 32 card against defects in material and workmanship for a period of 5 years from the date of original retail purchase. Applied Engineering also warrants that, under normal use, the magnetic media on which the software is stored is free from defects in materials and workmanship for a period of 30 days from the date of original purchase. Any misuse, abuse, or non-IE authorized alteration, modification and/or repair to the Applied Engineering product will void the warranty. This warranty will also be void if you use the IE product for any other purpose than its intended use. If you discover a defect, Applied Engineering will, at its option, repair or replace only the Applied Engineering product, provided you return the product during the warranty period, transportation prepaid, to Applied Engineering.

This warranty applies to the original retail purchaser only. Therefore, please include a copy of the original invoice or a small service charge may be applied. If the product is to be sent to Applied Engineering by mail, the purchaser will insure the package or assume full responsibility for loss or damage during shipping. Prior to returning the product for warranty consideration, call Applied Engineering Technical Support for a Return Material Authorization (RMA) number and shipping instructions.

Even though Applied Engineering has tested the software and reviewed the documentation, Applied Engineering makes no warranty or representation, either express or implied, with respect to the manual or the software, their quality, performance, merchantability, or fitness for a particular purpose. As a result, the software and manual are sold "as is," and you, the purchaser, not IE or its dealers, are assuming the entire risk as to their quality and performance.

In no event will Applied Engineering be liable for loss or damages of any kind caused either directly or indirectly by the use or possession of its products, even if advised of the possibility of such damages. The Applied Engineering Warranty is for the Applied Engineering Product itself. In particular, Applied Engineering shall have no liability for any other equipment used in conjunction with Applied Engineering products nor for programs or data stored in or used with Applied Engineering products, including the costs of recovering such equipment, programs, or data.

The warranty and remedies set forth above are exclusive and in lieu of all others, oral or written, express or implied. No Applied Engineering dealer, agent, or employee is authorized to make any modification, extension, or addition to this warranty.

Some states do not allow the exclusion or limitation of implied warranties or liability for incidental or consequential damages, so the above limitation or exclusion may not apply to you. This warranty gives you specific legal rights, and you may also have other rights which may vary from state to state.

This manual and the software (computer programs) described herein are copyrighted by Applied Engineering with all rights reserved. Under the copyright laws, this manual or the programs may not be copied, in whole or in part, without the written consent of Applied Engineering, except in the normal use of the software or to make an archival copy. This exception does not allow copies to be made for others, whether or not sold, but all of the materials purchased (with all archive copies) may be sold, loaned, or given to another person. Under the law, copying includes translating into another language or format. You may use this software on any computer owned by you but extra copies cannot be made for any purpose.

Applied Engineering cannot guarantee that you will receive notice of revisions to the software, documentation or products described in this manual. Be sure to check with your dealer or Applied Engineering for information on possible updates. However, Applied Engineering reserves the right to make any improvements to Applied Engineering products without any responsibility toward upgrading previously released products.

Apple and Apple IIgs are registered trademarks of Apple Computer, Inc.
Applied Engineering and I/O 32 are trademarks of Applied Engineering.

©Copyright 1988, Applied Engineering

Applied Engineering
P.O. Box 5100
Carrollton, Texas 75011
Sales: (214) 241-8060 9 AM - 11 PM (CST) 7 days
Technical Support: (214) 241-8069 9 AM - 12:30PM & 1:30PM - 5 PM (CST) Monday - Friday
(The Technical Support telephone lines cannot be accessed through the Sales department.)
Installing the I/O 32 in Your Apple

The I/O 32 board simply plugs into a connector inside your Apple. Care must be exercised however, so follow these instructions exactly.

1) **TURN OFF THE APPLE'S POWER SWITCH.** This is very important to prevent damaging the computer as well as the I/O 32 board. However, you need to leave the computer plugged in throughout the installation to allow the power supply to discharge static electricity from your body.

2) **Remove the cover from your Apple.**

   --Pop the hood of the ][ Plus or //e by pulling up on the cover at the rear edge (the edge farthest from the keyboard) until the two corner fasteners pop apart.

   --The II GS's lid has two fasteners on the sides of the back panel. Push in on the tops of the fasteners with your forefingers while pushing up with your thumbs and heel of your hands on the side of the lid.

3) **Touch the power supply to remove any static electricity from your body.** Do not skip this step! A static shock can damage the chips on your boards and/or the chips on your computer's motherboard.

4) **Locate the “slots” inside the Apple, across the rear of the main circuit board.**

5) **Plug the “fingers” of your I/O 32 board into any 1-7 expansion slot except slot 3. I/O 32 will not work in slot 0 of the ][+ nor the Auxiliary/Memory Expansion slot on the //e and II GS. The “fingers” of the circuit board into the slot you want. The fingers will enter the slot with some friction and will seat firmly. (Refer to picture below.)
6) Replace the Apple's cover by sliding the front edge into place, then press down on the two rear corners until they pop into place.

7) Now turn on your Apple and continue.
Features of the I/O 32

The I/O 32 has several features that make it very versatile. They allow you ease of use and changeability. The most important of the features is the onboard software (firmware). This software is contained in a 2716 EPROM (Erasable Programmable Read-Only Memory). This software allows you easy access to four ports. Within each port, you have eight pins which can be programmed as either input or output. Once you have set the desired direction, you can send or receive data in three formats. The formats are Decimal, Hexadecimal, and Binary. You can send data in any of the three formats and also receive in any of the three. The necessary commands to send or receive will be described in greater detail.

The I/O 32 also has four switches which allow you to configure the board for features other than the default values. The switch settings are described later.

The I/O 32 uses two 6821 Peripheral Interface Adapter which give you the capability of using four 8-bit ports. Port A and Port C are TTL compatible and can drive low TTL loads. Port B and Port D have high impedance inputs and push/pull outputs that can drive two TTL loads. (When used for a switch input, a 5K pull-up resistor is required.

![Switch Block](image)

- 5V ○ ○ GND
- C81 ○ ○ CB2
- PB6 ○ ○ PB7
- PB4 ○ ○ PB5
- PB2 ○ ○ PB3
- PB0 ○ ○ PB1
- N/C ○ ○ N/C
- CA1 ○ ○ CA2
- PA6 ○ ○ PA7
- PA4 ○ ○ PA5
- PA2 ○ ○ PA3
- PA0 ○ ○ PA1
- 5V ○ ○ GND

* PB = Port B, PA = Port A

The I/O 32 has 4 8-bit ports which can be programmed independently. The ports are labeled A, B, C, and D. Within each port there are 8 pins which are used for sending or receiving data. Each of the eight pins can be set as either input or output.

In order for you to use the I/O 32, you must build a cable that will suit your specific needs. The connector that you must acquire is available from several sources. Two are listed below:
Dip Switch Configurations

On the I/O 32, there are four switches which allow you to configure your board for specific applications. The board was shipped to you with the switches in the default positions noted below in the diagram.

![Dip Switch Diagram]

The switches can be used to enable different capabilities that are built into the I/O 32 board. As stated in the Features section of this manual, the I/O 32 has a 2716 EPROM containing the software necessary to make the card operate. This EPROM can store a program that is up to 2K-bytes in length. The software however, takes up only about half of the 2K so there is enough room left in the EPROM for a custom program to be permanently stored. If you had such a custom program in the same EPROM as the original software, you would simply move switch 4 to the open or off position. This will tell the Apple to look for a program in the upper half of the EPROM.

There is also another way to customize the I/O 32 without having to modify the EPROM. For instance, if the software on the board does not fit your needs, you could simply move switches 1 and 2 to their opposite state. This will disable the built-in software by telling the board and Apple that you are using a 6116 RAM (random access memory) chip instead of the 2716 EPROM. Once this is done, you could write your own software to control the input and output ports and have it loaded in from the disk every time you wanted to use it.
For those of you who need to have external devices linked with your I/O 32, you would set switch 3 to the on or closed position. This will enable the use of interrupts through program control. Interrupts using the I/O 32 will not be discussed in this manual.

Programming with the EPROM

Supplied with the I/O 32 board, is a program which will allow you easy access to the I/O ports. In order to use this software however, you must make sure that switch 1 is OPEN, switch 2 is CLOSED, and switch 4 is CLOSED. Once you have set the switches to the required position, you are ready to begin programming.

One of the easiest things to do at first is to locate the board through software. Since the card has software built in, we can search through the seven slots to look for particular parts of that program. To demonstrate this, type in this short program and then run it. When the program finishes running, it will return with a message telling you which slot the I/O 32 is located in. If it does not find it, it will tell you it did not find one.

1000 REM ***** Find the I/O 32 Board *****
1010 SLOT = 0
1020 FOR I = 1 TO 7
1030 ADDR = 12*4096+I*256: REM $Cs00 (where s is IO/32's slot number)
1040 IF PEEK(ADDR) = 44 AND PEEK(ADDR+1) = 88 AND PEEK(ADDR+254) = 201 THEN SLOT = I
1050 NEXT I
1060 IF SLOT = 0 THEN PRINT “No I/O 32 found!”:END
1070 PRINT “I/O 32 found in slot “;SLOT
1080 END

After you have successfully located the slot of the I/O 32, you can proceed to programming the card itself. For the following examples, we will assume that your I/O 32 board is in slot 2.

When programming the I/O 32, you must follow certain steps in order to make sure that the data is correctly sent or received. To do this you must initialize the card. To accomplish this, all you need to do is call the card, then print the command. Type this program in and run it.

1000 REM ***** INITIALIZATION *****
1010 D$ = CHR$(4)
1020 PRINT D$;"PR#2": PRINT D$;"IN#2"
1030 PRINT "I"
1040 PRINT D$;"PR#0": PRINT D$;"IN#0"
1050 END

I/O 32 5
Lines 1010 and 1030 selects and deselects the I/O 32. Line 1020 sends the necessary information to initialize the I/O ports. Sending this command has two effects:

1) Set all four ports to input only.
2) Disable all interrupts.

The next step in programming the I/O 32 ports is to set the desired direction for data. Type this program and run it.

```
1000 REM ***** SET DATA DIRECTION *****
1010 D$ = CHR$ (4)
1020 PRINT D$;"PR#2": PRINT D$;"IN#2"
1030 PRINT "A($FF)"
1040 PRINT D$;"PR#0": PRINT D$;"IN#0"
1050 END
```

As with the initialization command, lines 1010 and 1030 select and deselect the board. Line 1020 is the command that sets the direction for data. The "A" is the port letter. You could have specified any one of the four, but we will use port A. The number is given here in hexadecimal ($FF) but to the I/O 32 it is "11111111" in binary. Notice there are 8 1's. Also notice under the section on the User Connection that in each port there are 8 pins for sending or receiving data. Each 1 corresponds to one pin on the port selected, and a "1" means output, so the above program sets port A to all output. Once both of the above steps have been completed, you are ready to send data.

Let's for a minute, assume that you had 8 light bulbs that you wanted to turn on or off. You could hook up each of the eight lights to each of the 8 pins on port A (see the section "User Connection"). Then to turn a light on, you would just send the appropriate code to do so. For example:

```
1000 REM ***** TURN ON PIN 4 *****
1010 D$ = CHR$ (4)
1020 PRINT D$;"PR#2": PRINT D$;"IN#2"
1030 PRINT "I": PRINT "A($FF)"
1040 PRINT "A$10"
1050 PRINT D$;"PR#0": PRINT D$;"IN#0"
1060 END
```

Line 1020 should be familiar to you. This line combines the initialization command and the direction command. Line 1040 is the data to be sent to port A to turn our light on. The "$10" is again given in hexadecimal but the I/O 32 sees it as "00010000". In this example, the 1 does not mean output as it did in setting the direction, it means on. This command just sends data to turn on or off a selected number of pins. If you wanted to turn light 4 off and lights 7 and 8 on, you would change line 1040 to: 1040 PRINT "A$03"
This command sends the binary code "00000011" to port A and turns off light 4 and then turns on lights 7 and 8.

The above programs are used to turn on and off lights when you already know their present state. However, through program control, you could check to see which ones are on and which ones are off. You would do this with the input command. At this time, we should still have lights 7 and 8 on.

Make sure 7 and 8 are still on with this program.

```
1000 REM ***** CHECK STATE *****
1010 DS = CHR$ (4)
1020 PRINT DS;"PR#2"; PRINT DS;"IN#2"
1030 INPUT "A$";X$    
1040 PRINT DS;"PR#0"; PRINT DS;"IN#0"
1050 PRINT X$
1060 END
```

Lines 1010 and 1020 are the same as for output. The change comes in line 1030. When inputting data, we use the DOS input statement, then the port letter and value mode followed by the variable to store the data into. In line 1050, we print the data that was input in line 1030.

After you have typed this program and executed it, you will have accomplished all the basic commands that are dealt with by the I/O 32. Further programs can be derived from the following table of syntax commands.

**Output Syntax**

```
PRINT "<port><data.value>"
PRINT "<port><ddr.value>"
PRINT "<port><ddr.value><data.value>"
```

**NOTE:** The <ddr.value> is the value that determines which of the 8 pins in each port are input or output.

**Input Syntax**

```
INPUT "<port><value.mode>";<string>
INPUT "<port><ddr.value><value.mode>";<string>
INPUT "<port>";<string>
INPUT <string>
```

**NOTE:** The <ddr.value> is the value that determines which of the 8 pins in each port are input or output.
When `<port>` and/or `<value.mode>` are missing with the INPUT command, the previously specified port and mode will be used. If you have not previously specified both of those, you will get unpredictable results.

**Acceptable Values**

With the built-in software, there are values that must be used in order to prevent errors when running a program. The values are:

- `<port>`: A, B, C, D
- `<value.mode>`: $ - hexadecimal, # - decimal, % - binary, B - binary
- `<digits>`: hexadecimal 0 - 9 and A - F, decimal 0 - 9, binary 0 and 1

The port value and digits are self explanatory however, the value mode needs a little explaining. Whenever you want to send or receive data you must specify one of the three modes. Once you specify which mode you wish to use, you simply send the appropriate value. For example if you wanted to send a value of 26 decimal, you could also send it as IA hexadecimal or 00011010 in binary, just by specifying the mode and then the value you wish to send.

**Maximum Values**

With each of the acceptable values above, there are certain limitations on the maximum number that can be entered into the I/O 32. The maximum values are for `<digits>` only.

- HEXADECIMAL 0 - FF
- DECIMAL 0 - 255
- BINARY 00000000 - 11111111

**DO NOT EXCEED THESE VALUES OR YOUR PROGRAM WILL HALT SUDDENLY!**
Sample Programs

Below is a sample program that will give you some idea of how to use the I/O 32 under software control.

Sample Program

1000 REM ***** DEMONSTRATION *****
1010 GOSUB 10000: REM FIND I/O 32
1020 PRINT CHR$(4);"PR#";SL0T
1030 PRINT "I"; PRINT "A($FF)"
1040 PRINT CHR$(4);"PR#0"
1050 TEXT:HOME
1060 INPUT "ENTER VALUE TO SEND: ";V$: IF V$=" " THEN END
1070 PRINT CHR$(4);"PR#";SL0T: PRINT CHR$(4);"IN#";SL0T
1080 PRINT "A$";V$
1090 INPUT "A$";D1$
1100 INPUT "A#";D2$
1110 INPUT "A%";D3$
1120 PRINT CHR$(4);"PR#0"; PRINT CHR$(4);"IN#0"
1130 PRINT "PORT A - HEXADECIMAL ";D1$
1140 PRINT " - DECIMAL ";D2$
1150 PRINT " - BINARY ";D3$
1160 VTAB 6:PRINT "PRESS ANY KEY TO CONTINUE..."; GET A$
1170 GOTO 1050
1180 END
10000 REM ***** FIND I/O 32 BOARD ***** 10010 SLOT = 0
10020 FOR I = 1 TO 7
10030 ADDR = 12*4096 + I*256: REM $CS00
10040 IF PEEK(ADDR) = 44 AND PEEK(ADDR+1) = 88 AND
PEEK(ADDR+254) = 201 THEN SLOT = I: I = 7
10050 NEXT I
10060 IF SLOT= 0 PRINT "NO I/O 32 FOUND!": END 10070 RETURN

Peripheral Driver Chips for the I/O 32

Following are the part numbers for 500mA (max.) inverting driver chips that can be used to drive DC lamps, relays etc. with the I/O 32. This is just a partial list. Manufacturer data sheets and catalogs may be able to point you to other drivers.

Chips with 8 drivers
- ULN2803 Motorola
- ULN2803 Sprague

Chips with 7 drivers
- MC1413 Motorola
- ULN2003A Sprague

Check the manufacturer's data sheets for pin-out information, etc.
**IO/32 Address Bits**

Machine Language Programmers: the following address register charts are provided to help you address the chips directly.

If you’re not interested in machine language programming, ignore this section.

Access through device select page $C08X + n0
where n = slot number

---

**The 6502 Address Registers in Slot s**

<table>
<thead>
<tr>
<th>Address</th>
<th>CRA Bit 2</th>
<th>CRB Bit 2</th>
<th>Read</th>
<th>Write</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C084+s0</td>
<td>1</td>
<td>-</td>
<td>Port A In</td>
<td>Port A Out</td>
</tr>
<tr>
<td>$C084+s0</td>
<td>0</td>
<td>-</td>
<td>Data Dir A</td>
<td>Data Dir A</td>
</tr>
<tr>
<td>$C085+s0</td>
<td>-</td>
<td>-</td>
<td>Ctrl Reg A</td>
<td>Ctrl Reg A</td>
</tr>
<tr>
<td>$C086+s0</td>
<td>-</td>
<td>1</td>
<td>Port B In</td>
<td>Port B Out</td>
</tr>
<tr>
<td>$C086+s0</td>
<td>-</td>
<td>0</td>
<td>Data Dir B</td>
<td>Data Dir B</td>
</tr>
<tr>
<td>$C087+s0</td>
<td>-</td>
<td>-</td>
<td>Ctrl Reg B</td>
<td>Ctrl Reg B</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>Address</th>
<th>CRA Bit 2</th>
<th>CRB Bit 2</th>
<th>Read</th>
<th>Write</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C088+s0</td>
<td>1</td>
<td>-</td>
<td>Port C In</td>
<td>Port C Out</td>
</tr>
<tr>
<td>$C088+s0</td>
<td>0</td>
<td>-</td>
<td>Data Dir C</td>
<td>Data Dir C</td>
</tr>
<tr>
<td>$C089+s0</td>
<td>-</td>
<td>-</td>
<td>Ctrl Reg C</td>
<td>Ctrl Reg C</td>
</tr>
<tr>
<td>$C08A+s0</td>
<td>-</td>
<td>1</td>
<td>Port D In</td>
<td>Port D Out</td>
</tr>
<tr>
<td>$C08A+s0</td>
<td>-</td>
<td>0</td>
<td>Data Dir D</td>
<td>Data Dir D</td>
</tr>
<tr>
<td>$C08B+s0</td>
<td>-</td>
<td>-</td>
<td>Ctrl Reg D</td>
<td>Ctrl Reg D</td>
</tr>
</tbody>
</table>
In the following Control Register Table, \( x \) is either A, B, C or D.

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>IRQx1 Flag</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>IRQ caused by Cx1 transition (cleared by reading Port x)</td>
</tr>
<tr>
<td>0</td>
<td>No IRQ</td>
</tr>
</tbody>
</table>

When Bit 5 = 0

<table>
<thead>
<tr>
<th>Bit 6</th>
<th>IRQ Flag</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>IRQ caused by Cx2 (cleared by reading Port x)</td>
</tr>
<tr>
<td>0</td>
<td>No IRQ</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 5</th>
<th>Cx2 Mode Select</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Select Cx2 Input Mode</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 4</th>
<th>Cx2 polarity for IRQ</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Set IRQx2 (bit 6) for low-to-high transition of Cx2</td>
</tr>
<tr>
<td>0</td>
<td>Set IRQx2 (bit 6) for high-to-low transition of Cx2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 3</th>
<th>IRQx enable for IRQx2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Enable IRQx</td>
</tr>
<tr>
<td>0</td>
<td>Disable IRQx</td>
</tr>
</tbody>
</table>

When Bit 5 = 1

| Bit 6 | Always 0                    |

<table>
<thead>
<tr>
<th>Bit 5</th>
<th>Cx2 Mode Select</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Select Cx2 Output Mode</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 4</th>
<th>Cx2 Output Control</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Cx2 goes low when a 0 is written to CRx bit 3</td>
</tr>
<tr>
<td></td>
<td>Cx2 goes high when a 1 is written to CRx bit 3</td>
</tr>
<tr>
<td>0</td>
<td>Cx2 goes low in the first high-to-low transition of phase 0 after a read of Port x</td>
</tr>
<tr>
<td></td>
<td>Cx2 goes high as specified by bit 3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 3</th>
<th>Cx2 read strobe restore control (bit 4 = 0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Cx2 goes high on the next phase 0 clock high-to-low transition following a read of Port x</td>
</tr>
<tr>
<td>0</td>
<td>Cx2 goes high on the next active Cx1 transition as specified by bit 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 2</th>
<th>Port x I/O or Data Direction Select</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Port x</td>
</tr>
<tr>
<td>0</td>
<td>Data Direction Register for Port x</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 1</th>
<th>Cx1 polarity for IRQ</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Set IRQx1 (bit 7) for low-to-high transition of Cx1</td>
</tr>
<tr>
<td>0</td>
<td>Set IRQx1 (bit 7) for high-to-low transition of Cx1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 0</th>
<th>IRQx enable for IRQx1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ENable IRQx</td>
</tr>
<tr>
<td>0</td>
<td>Disable IRQx</td>
</tr>
</tbody>
</table>
If a bit in the Data Direction Register is 1, the corresponding bit of the Port is an output; otherwise, it is an input.

The procedure for writing to an output is:
1) Write a 0 to bit 2 of the Control register.
2) Write 1’s to the Data Direction Register of each output bit.
3) Write a 1 to bit 2 of the Control Register.
4) Write data to the Port.

The procedure for reading from an input is:
1) Write a 0 to bit 2 of the Control Register.
2) Write 0’s to the Data Direction Register of each input bit.
3) Write a 1 to bit 2 of the Control Register.
4) Read data from the Port.

Control Reg A $C085+s0
Control Reg B $C087+s0
Control Reg C $C089+s0
Control Reg D $C08B+s0

Examples are included on the following page.
Examples:

Reading all 8 bits from Port A
LDA $C085+s0 ;Read Control Register A
AND #%11111011 ;Turn off bit 2
STA $C085+s0 ;Set CRA for Data Direction
LDX #$00 ;All bits are inputs
STX $C084+s0 ;Set Data Direction for all inputs
ORA #%00000100 ;Turn on bit 2
STA $C085+s0 ;Set CRA for Port A data
LDA $C084+s0 ;Read Port A data

Writing all 8 bits to Port D
LDA $C08B+s0 ;Read Control Register D
AND #%11111111 ;Turn off bit 2
STA $C08B+s0 ;Set CRD for Data Direction
LDX #$FF ;All bits are outputs
STX $C08A+s0 ;Set Data Direction for all outputs
ORA #%00000100 ;Turn on bit 2
STA $C08B+s0 ;Set CRD for Port D data
LDA #$AE ;Data to write
STA $C08A+s0 ;Write Port D data

Reading bit 0, 1, 2 from Port B
LDA $C087+s0 ;Read Control Register B
AND #%11111011 ;Turn off bit 2
STA $C087+s0 ;Set CRB for Data Direction
LDX #%11110000 ;Bits 0,1,2 are inputs
STX $C086+s0 ;Set Data Direction
ORA #%00000100 ;Turn on bit 2
STA $C087+s0 ;Set CRB for Port B data
LDA $C086+s0 ;Read Port B data

Writing bits 4, 5, 6, 7 to Port C
LDA $C089+s0 ;Read Control Register C
AND #%11111011 ;Turn off bit 2
STA $C089+s0 ;Set CRC for Data Direction
LDX #%11110000 ;Bits 4,5,6,7 are outputs
STX $C088+s0 ;Set Data Direction
ORA #%00000100 ;Turn on bit 2
STA $C089+s0 ;Set CRC for Port C data
LDA #$A0 ;Data to write
STA $C088+s0 ;Write Port C data
6821 Data Sheets

The following pages include the technical information about the Hitachi HD6821 chip found on the I/O 32 board. The selected pages have been reprinted from, 8/16-Bit Multi-Chip Micro Computer Data Book, U70, pages 319-335 with permission from:

Hitachi America, Ltd.
Semiconductor & I.C. Sales and Service Division
HD6821, HD68A21, HD68B21—PIA (Peripheral Interface Adapter)

The HD6821 Peripheral Interface Adapter provides the basic means of controlling processors of the HD6800 Microprocessor family. The device is capable of interfacing the MPU to peripheral through the HD6800 Microprocessor's input and output lines. An external logic is required for interfacing to some peripheral devices.

The functional configurations of the PIA are programmed by the MPU through system instructions. Each of the peripheral data lines can be programmed to act as an input or output, and each of the four control signals may be programmed for one of several control modes. This allows a high degree of flexibility in the overall operation of the interface.

**Features**
- Two Bidirectional 8-bit Peripheral Data Bus: Direct Interface to Peripherals
- Two 8-bit Programmable Control Registers
- Two 8-bit Programmable Data Direction Registers
- Four Individually Controlled Input/Output Lines
- Two Programmable Control Outputs
- Two Programmable Control Inputs
- High-Speed I/O Lines and Direct Transistor Drive
- Bidirectional I/O Lines
- Programmed Input/Output and Interrupt Capability

**Applications**
- Direct Control of Graphic Devices
- Direct Control of Digital Devices
- Direct Control of Analog Devices
- Direct Control of Memory Devices

**Operation**
- Data Bus Interface: The PIA has two 8-bit data buses, one for input and one for output. Each data bus can be programmed as an input or output, allowing the MPU to read or write data to peripheral devices.

**Control Signals**
- **Data Direction (DD)**: Controls the direction of data transfer to or from the peripheral device. When set to '0', the data bus is configured as an input; when set to '1', the data bus is configured as an output.

**Interrupt Requests**
- The PIA can generate interrupt requests to the MPU, allowing the MPU to be interrupted when a peripheral device requires attention. The interrupt request is generated when a peripheral device asserts an interrupt signal to the MPU.

**Peripheral Interface Lines**
- Each peripheral data bus can be programmed to act as an input or output. This is accomplished by setting a '1' in the corresponding 'Write' bit of the peripheral control register. The data can be read or written to the peripheral devices in the same manner as the MPU.

**Conclusion**
- The HD6821 Peripheral Interface Adapter provides a flexible means of controlling peripherals connected to the HD6800 Microprocessor family. Its programmable features allow for a wide range of applications, making it a versatile component in system design.
Table 1 Internal Addressing

<table>
<thead>
<tr>
<th>RB</th>
<th>RH</th>
<th>RC</th>
<th>Control Register</th>
<th>Location</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Peripheral Register A</td>
<td>0 0 0 0</td>
<td>Peripheral Register A</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>General Register A</td>
<td>0 0 0 0</td>
<td>General Register A</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Control Register B</td>
<td>0 0 0 0</td>
<td>Control Register B</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>General Register B</td>
<td>0 0 0 0</td>
<td>General Register B</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Total External Register</td>
<td>0 0 0 0</td>
<td>Total External Register</td>
</tr>
</tbody>
</table>

Table 2 Control Word Format

<table>
<thead>
<tr>
<th>CRAB</th>
<th>CRAD</th>
<th>CRAD</th>
<th>CRAD</th>
<th>CRAD</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
</tr>
</tbody>
</table>

Table 3 Control of Interrupt Inputs Cx, Cx, and Cx

<table>
<thead>
<tr>
<th>CRAB</th>
<th>CRAD</th>
<th>CRAD</th>
<th>CRAD</th>
<th>CRAD</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
</tr>
</tbody>
</table>

Table 4 Control of Cx, Cx, and Cx in Input Mode

<table>
<thead>
<tr>
<th>CRAB</th>
<th>CRAD</th>
<th>CRAD</th>
<th>CRAD</th>
<th>CRAD</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
</tr>
</tbody>
</table>
### Table 1: Control of CA0 as an Output

<table>
<thead>
<tr>
<th>CA0</th>
<th>CA1</th>
<th>CA2</th>
<th>Channel</th>
<th>Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>&quot;Low&quot; on the negative transition of E after an MPU Read &quot;A&quot; Data Operation</td>
<td>&quot;High&quot; when the negative transition of E after an MPU Read &quot;A&quot; Data Operation</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>&quot;High&quot; when the negative transition of E after an MPU Read &quot;B&quot; Data Operation</td>
<td>&quot;High&quot; when the negative transition of E after an MPU Read &quot;B&quot; Data Operation</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>&quot;Low&quot; (The content of CA0 is output on CA0)</td>
<td>&quot;High&quot; (The content of CA0 is output on CA0)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>&quot;High&quot; (The content of CA0 is output on CA0)</td>
<td>&quot;High&quot; (The content of CA0 is output on CA0)</td>
</tr>
</tbody>
</table>

### Table 2: Control of CA0 as an Output

<table>
<thead>
<tr>
<th>CA0</th>
<th>CA1</th>
<th>CA2</th>
<th>Channel</th>
<th>Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>&quot;Low&quot; on the negative transition of E after an MPU Read &quot;A&quot; Data Operation</td>
<td>&quot;High&quot; when the negative transition of E after an MPU Read &quot;A&quot; Data Operation</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>&quot;Low&quot; on the negative transition of E after an MPU Read &quot;A&quot; Data Operation</td>
<td>&quot;High&quot; when the negative transition of E after an MPU Read &quot;A&quot; Data Operation</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>&quot;High&quot; (The content of CA0 is output on CA0)</td>
<td>&quot;High&quot; (The content of CA0 is output on CA0)</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>&quot;High&quot; (The content of CA0 is output on CA0)</td>
<td>&quot;High&quot; (The content of CA0 is output on CA0)</td>
</tr>
</tbody>
</table>

### FIA Operation

**Initialization**

When the external reset input RES goes "Low", all internal registers are set to "0". Parameter CA0 (CA0, PA, PB, PB2) is defined as the output data interface (CA0, PA, PB, PB2). This parameter is defined as the 8-bit data interface (CA0, PA, PB, PB2) and is also initialized by the software sequence as follows.

- Clear the direct register.
- Load the direct register into the direct register.
- Set the contents of the direct register into the direct interface.
- Load the direct register into the direct interface.
- Set the contents of the direct interface into the direct register.

**Read/Write Operation**

**Read Operation**

- Set the data transfer register to "DDRA".
- Set the direct register to "DDRA".
- Set the direct interface to "DDRA".
- Write the data into the peripheral interface register.

**Write Operation**

- Set the data transfer register to "DDRB".
- Set the direct register to "DDRB".
- Set the direct interface to "DDRB".
- Write the data into the peripheral interface register.
To read the peripheral data, the data is directly transmitted to the data input Dn while the latch is closed (CPA=CPA). In this case, the peripheral interface register must be set before using the peripheral interface register.

When initializing the control register, interrupt flag set (CR+1, CR+2, CR+3, CR+4) cannot be written from MPU. To overcome this problem, the CR+1, CR+2, CR+3, CR+4 must be set by MPU before using the peripheral interface register.

In the described method, the interrupt flag set (CR+1, CR+2, CR+3, CR+4) cannot be written from MPU. To overcome this problem, the CR+1, CR+2, CR+3, CR+4 must be set by MPU before using the peripheral interface register.

Interrupt service routines
CR+1, CR+2, CR+3, CR+4 cannot be written from MPU. To overcome this problem, the CR+1, CR+2, CR+3, CR+4 must be set by MPU before using the peripheral interface register.

Software polling method
This method requires the MPU to write to the peripheral interface register during operation.

<Handshaking Mode>

When using the handshaking mode, the CR+1, CR+2, CR+3, CR+4 must be set by MPU before using the peripheral interface register.

<Peripheral device mode>

A peripheral device must be read first before using the peripheral interface register.

<Peripheral device mode>

A peripheral device must be read first before using the peripheral interface register.

<Peripheral device mode>

A peripheral device must be read first before using the peripheral interface register.

<Peripheral device mode>

A peripheral device must be read first before using the peripheral interface register.

<Peripheral device mode>

A peripheral device must be read first before using the peripheral interface register.

<Peripheral device mode>

A peripheral device must be read first before using the peripheral interface register.
Getting Help

If you have a technical question relating to the mechanical performance of your I/O 32 card that is not covered in the manual, please contact the dealer from whom you purchased the card. If you are experiencing difficulties with one particular program, contact the program's author or publisher.

In the event that the dealer or the publisher's support personnel cannot answer your question, call Applied Engineering Technical Support. The support representatives are experienced in the applications and uses of Applied Engineering products, but in order to provide a quick and effective answer to your question, they will need to know as much as possible about the hardware and software specifically related to your question. Please provide the technical support representative with the following information:

◊ The Applied Engineering product related to your question and its revision number.
◊ The original and current memory configuration of the card (if applicable).
◊ The model and revision of your computer.
◊ What peripherals are being used and what cards are in each slot.
◊ The name, version, and revision level of the software that you are experiencing problems with.
◊ The results of any test programs, diagnostics, or troubleshooting done by you, your dealer or your software publisher's support department.

Applied Engineering
Technical Support
(214) 241-6069
9 AM to 12:30 PM & 1:35 PM to 5 PM(CST)
Monday Through Friday
(Please call only the number above for technical support. Our sales office cannot transfer calls to the support lines.)
Returning a Product

Include

If your product needs to be returned, the technical support representative will give you a Return Material Authorization (RMA) number.

- Record the RMA number for your own records.
- Write the RMA number on the outside of the package you send to us.
- Write the RMA number at the top of the return form included with your product package.

Fill out the Return Form on back of the yellow sheet marked, "Attention!" A correctly completed form will greatly reduce the time it takes to process and return your product.

Attach a copy of your original invoice to the return form.

⚠️ Warning: If you don’t include an invoice products will be treated as out of warranty products and will be returned to you C.O.D. for the amount of the service charge.

A completed form should look something like the one below.
When You Ship

If you don't have the original packing material, wrap the board in anti-static material (preferably the anti-static bag in which the card was originally shipped, however, aluminum foil will work fine). Pack it in a sturdy box cushioned with wadded papers (i.e. used computer paper or newspaper).

⚠️ Warning: If your product is damaged due to inadequate packing, your warranty will be void.

Include the return form and invoice.

Send the package, shipping prepaid, to:

RMA#_?__
Applied Engineering
Technical Support
3210 Belt Line Road, Suite 154
Dallas TX 75234

You should insure your package. AÉ will not assume any responsibility for inadequate packing or loss or damage during shipping.

When We Receive

Our service department will use your completed form in an attempt to duplicate the problem.

If it is determined that your product is defective due to a manufacturing defect, your card will be repaired or replaced at AÉ's option.

Any misuse, abuse, or non-AÉ authorized alteration, modification and/or repair to the Applied Engineering product will void the warranty. This warranty will also be void if you use the AÉ product for any purpose other than its intended use.

Your product will be fully tested before it is shipped back to you, transportation prepaid, via UPS regular delivery.

Once your product is received by Technical Support, it will be processed and delivered to our shipping department within 7 to 10 working days.