I/O 32™

User's Manual

APPLIED ENGINEERING™

A DIVISION OF AE RESEARCH CORPORATION

v2.0

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/0 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-Æ 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 Æ 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 Æ 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.

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.

Applied Engineering
P.O. Box 5100
Carrollton, Texas 75011
Sales: (214) 241-6060 9 AM - 11 PM (CST) 7 days
Technical Support: (214) 241-6069 9 AM - 12:30PM & 1:35PM - 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.**

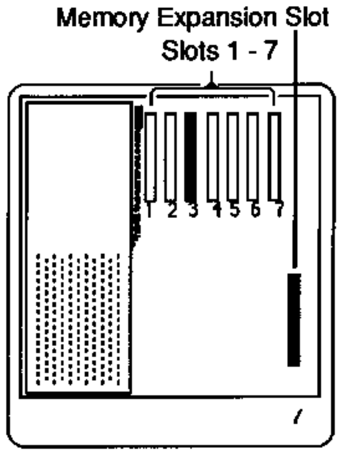
   --Pop the hood of the][Plus or //e by pulling up on the cover at the rear edge (the edge farthest from the keyboard) until the two corner fasteners pop apart.

   --The IIGS'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.

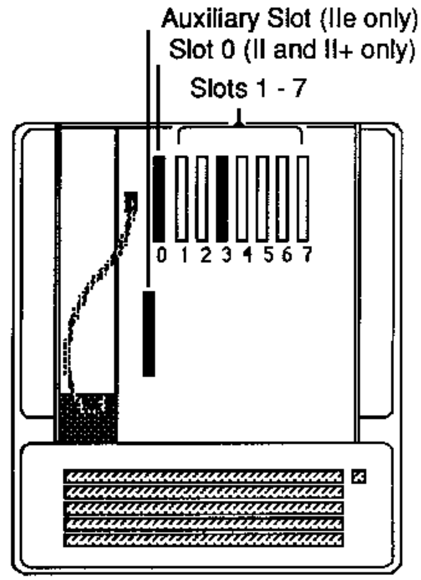
3) **Touch the power supply** to remove any static electricity from your body. Do not skip this step! A static shock can damage the chips on your boards and/or the chips on your computer's motherboard.

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 IIGS. 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.)



Inside the IIGS



Inside the][,][+ and //e

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.

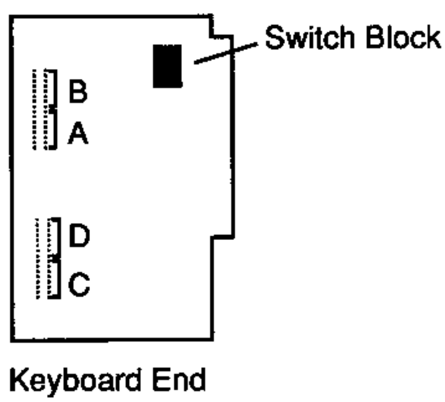
7) **Now turn on your Apple and continue.**

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/Ò 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 tow 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.



+5V	GND
CBI	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.

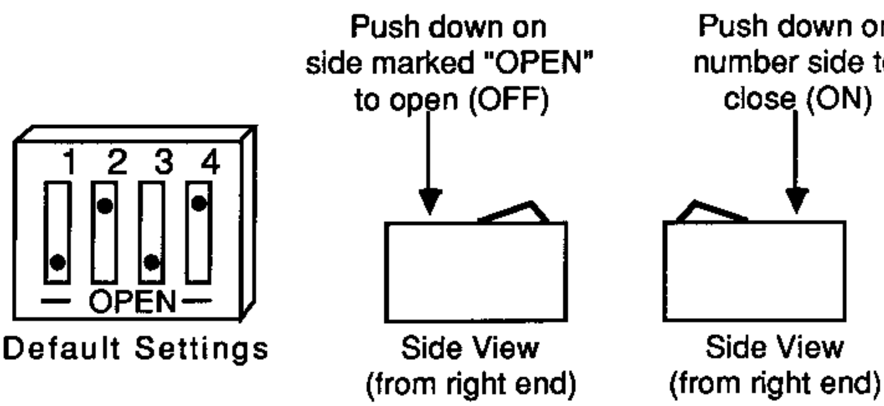
In order for you to use the I/O 32, you must build a cable that will suit your specific needs. The connector that you must acquire is available from several sources. Two are listed below:

Digi-Key	1-800-344-4539 Part No. R304-ND
AP Products	1-800-321-9668 Part No. 925110-26-R

These connectors will crimp onto standard 26 conductor ribbon cable.

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.



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
```

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  D$ = CHR$ (4)
1020  PRINT D$;"PR#2": PRINT D$;"IN#2"
1030  INPUT "A%";X$
1040  PRINT D$;"PR#0": PRINT D$;"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 "A#15"
PRINT "<port><ddr.value>"                 PRINT "A($FF)"
PRINT "<port><ddr.value><data.value>"     PRINT "A($FF)#15"
```

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 "A$";X$
INPUT "<port><ddr.value><value.mode>";<string>  INPUT "A($00)$",X$
INPUT "<port>";<string>                         INPUT "A";X$
INPUT <string>                                  INPUT X$
```

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 decimal binary	0 - 9 and A - F 0 - 9 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 lA 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$(4);"PR#";SLOT
1030  PRINT "I": PRINT "A($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#";SLOT: PRINT CHR$(4);"IN#";SLOT
1080  PRINT "A$";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

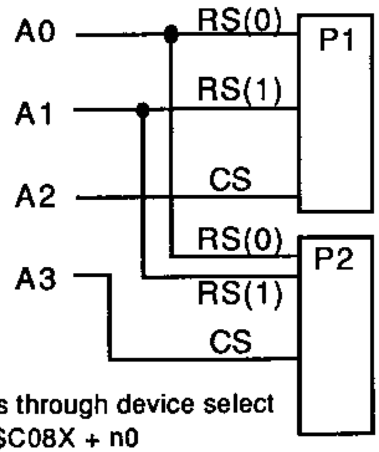
Chips with 7 drivers
- MC1413 Motorola
- ULN2003A Sprague

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

Address	CRA Bit 2	CRB Bit 2	Read	Write
$C084+s0	1	-	Port A In	Port A Out
$C084+s0	0	-	Data Dir A	Data Dir A
$C085+s0	-	-	Ctrl Reg A	Ctrl Reg A
$C086+s0	-	1	Port B In	Port B Out
$C086+s0	-	0	Data Dir B	Data Dir B
$C087+s0	-	-	Ctrl Reg B	Ctrl Reg B

Adress	CRA Bit 2	CRB Bit 2	Read	Write
$C088+s0	1	-	Port C In	Port C Out
$C088+s0	0	-	Data Dir C	Data Dir C
$C089+s0	-	-	Ctrl Reg C	Ctrl Reg C
$C08A+s0	-	1	Port D In	Port D Out
$C08A+s0	-	0	Data Dir D	Data Dir D
$C08B+s0	-	-	Ctrl Reg D	Ctrl Reg D

In the following Control Register Table, x is either A, B, C or D.

Bit 7 IRQx1 Flag
- 1 IRQ caused by Cx1 transition (cleared by reading Port x)
- 0 No IRQ

When Bit 5 = 0

Bit 6 IRQ Flag
- 1 IRQ caused by Cx2 (cleared by reading Port x)
- 0 No IRQ

Bit 5 Cx2 Mode Select
- 0 Select cx2 Input Mode

Bit 4 Cx2 polarity for IRQ
- 1 Set IRQx2 (bit 6) for low-to-high transition of Cx2
- 0 Set IRQx2 (bit 6) for high-to-low transition of Cx2

Bit 3 IRQx enable for IRQx2
- 1 Enable IRQx
- 0 Disable IRQx

When Bit 5 = 1

Bit 6 Always 0

Bit 5 Cx2 Mode Select
- 1 Select cx2 Output Mode

Bit 4 Cx2 Output Control
- 1 Cx2 goes low when a 0 is written to CRx bit 3
  Cx2 goes high when a 1 is written to cRx bit 3
- 0 Cx2 goes low in the first high-to-low transition of phase 0 after a read of Port x
  Cx2 goes high as specified by bit 3

Bit 3 Cx2 read strobe restore control (bit 4 = 0)
- 1 Cx2 goes high on the next phase 0 clock high-to-low transition following a read of Port x
- 0 Cx2 goes high on the next active Cx1 transition as specified by bit 1

Bit 2 Port x I/O or Data Direction Select
- 1 Port x
- 0 Data Direction Register for Port x

Bit 1 Cx1 polarity for IRQ
- 1 Set IRQx1 (bit 7) for low-to-high transition of Cx1
- 0 Set IRQx1 (bit 7) for high-to-low transition of Cx1

Bit 0 IRQx enable for IRQx1
- 1 ENable IRQx
- 0 Disable IRQx

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 #%11111011      ;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 #%11111000      ;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 universal means of interfacing peripheral equipment to the HD6800 Microprocessing Unit (MPU). This device is capable of interfacing the MPU to peripherals through two 8-bit bi-directional peripheral data buses and four control lines. No external logic is required for interfacing to most peripheral devices.

The functional configuration of the PIA is programmed by the MPU during system initialization. Each of the peripheral data lines can be programmed to act as an input or output, and each of the four control/interrupt lines may be programmed for one of several control modes. This allows a high degree of flexibility in the over-all operation of the interface.

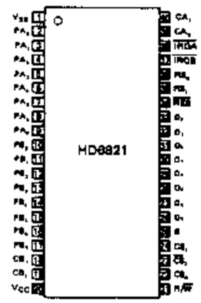
HD6821P, HD68A21P, HD68B21P

(DP-40)

■ FEATURES

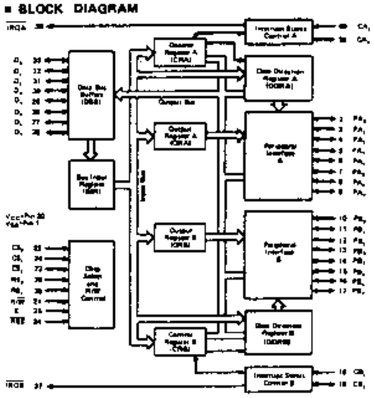
- Two Bi-directional 8-Bit Peripheral Data Bus for interface to Peripheral devices
- Two Programmable Control Registers
- Two Programmable Data Direction Registers
- Four Individually-Controlled Interrupt Input Lines: Two Usable as Peripheral Control Outputs
- Handshake Control Logic for Input and Output Peripheral Operation
- High-Impedance 3-State and Direct Transistor Drive Peripheral Lines
- Program Controlled Interrupt and Interrupt Disable Capability
- CMOS Drive Capability on Side A Peripheral Lines
- Two TTL Drive Capability on All A and B Side Buffers
- N Channel Silicon Gate MOS
- Compatible with MC6821, MC68A21 and MC68B21

■ PIN ARRANGEMENT



(Top View)

■ BLOCK DIAGRAM



■ PIA INTERFACE SIGNALS FOR MPU

The PIA interfaces to the HD6800 MPU with an eight-bit bi-directional data bus, three chip select lines, two register select lines, two interrupt request lines, read/write line, enable line and reset line. These signals, in conjunction with the HD6800 VMA output, permit the MPU to have complete control over the PIA. VMA should be utilized in conjunction with an MPU address line into a chip select of the PIA.

- **PIA Bi-Directional Data (D_0~D_7)**

The bi-directional data lines (D_0 ~ D_7) allow the transfer of data between the MPU and the PIA. The data bus output drivers are three-state devices that remain in the high-impedance (off) state except when the MPU performs a PIA read operation. The $R/\overline{W}$ line is in the Read ("High") state when the PIA is selected for a Read operation.

- **PIA Enable (E)**

The enable pulse, E, is the only timing signal that is supplied to the PIA. Timing of all other signals is referenced to the leading and trailing edges of the E pulse. This signal will normally be a derivative of the HMCS6800 System ϕ_2 Clock. This signal must be continuous clock pulse.

- **PIA Read/Write ($R/\overline{W}$)**

This signal is generated by the MPU to control the direction of data transfers on the Data Bus. A "Low" state on the PIA line enables the input buffers and data is transferred from the MPU to the PIA on the E signal if the device has been selected. A "High" on the $R/\overline{W}$ line sets up the PIA for a transfer of data to the bus. The PIA output buffers are enabled when the proper address and the enable pulse E are present.

- **Reset ($\overline{RES}$)**

The active "Low" $\overline{RES}$ line is used to reset all register bits in the PIA to a logical zero "Low". This line can be used as a power-on reset and as a master reset during system operation.

- **PIA Chip Select (CS_0, CS_1 and $\overline{CS_2}$)**

These three input signals are used to select the PIA. CS_0 and CS_1 must be "High" and $\overline{CS_2}$ must be "Low" for selection of the device. Data transfers are then performed under the control of the E and $R/\overline{W}$ signals. The chip select lines must be stable for the duration of the E pulse. The device is deselected when any of the chip selects are in the inactive state.

- **PIA Register Select (RS_0 and RS_1)**

The two register select lines are used to select the various registers inside the PIA. These two lines are used in conjunction with Internal Control Registers to select a particular register that is to be written or read.

The register and chip select lines should be stable for the duration of the E pulse while in the read or write cycle.

- **Interrupt Request ($\overline{IRQA}$ and $\overline{IRQB}$)**

The active "Low" Interrupt Request lines ($\overline{IRQA}$ and $\overline{IRQB}$) act to interrupt the MPU either directly or through interrupt priority circuitry. These lines are "open drain" (no load device on the chip). This permits all interrupt request lines to be tied together in a wire-OR configuration.

Each $\overline{IRQ}$ line has two internal interrupt flag bits that can cause the $\overline{IRQ}$ line to go "Low". Each flag bit is associated with a particular peripheral interrupt line. Also four interrupt enable bits are provided in the PIA which may be used to inhibit a particular interrupt from a peripheral device.

Servicing an interrupt by the MPU may be accomplished by a software routine that, on a prioritized basis, sequentially reads and tests the two control registers in each PIA for interrupt flag bits that are set.

The interrupt flags are cleared (zeroed) as a result of an MPU Read Peripheral Data Operation of the corresponding data register. After being cleared, the interrupt flag bit cannot be enabled to be set until the PIA is deselected during an E pulse. The E pulse is used to condition the interrupt control lines (CA_1, CA_2, CB_1, CB_2). When these lines are used as interrupt inputs at least one E pulse must occur from the inactive edge to the active edge of the interrupt input signal to condition the edge sense network. If the interrupt flag has been enabled and the edge sense circuit has been properly conditioned, the interrupt flag will be set on the next active transition of the interrupt input pin.

■ PIA PERIPHERAL INTERFACE LINES

The PIA provides two 8-bit bi-directional data buses and four interrupt/control lines for interfacing to peripheral devices.

- **Section A Peripheral Data (PA_0~PA_7)**

Each of the peripheral data lines can be programmed to act as an input or output. This is accomplished by setting a "1" in the corresponding Data Direction Register bit for those lines which are to be outputs. A "0" in a bit of the Data Direction Register causes the corresponding peripheral data line to act as an input. During an MPU Read Peripheral Data Operation, the data on peripheral lines programmed to act as inputs appears directly on the corresponding MPU Data Bus lines.

The data in Output Register A will appear on the data lines that are programmed to be outputs. A logical "1" written into the register will cause a "High" on the corresponding data line while a "0" results in a "Low". Data in Output Register A may be read by an MPU "Read Peripheral Data A" operation when the corresponding lines are programmed as outputs. This data will be read properly if the voltage on the peripheral data lines is greater than 2.0 volts for a logic "1" output and less than 0.8 volt for a logic "0" output. Loading the output lines such that the voltage on these lines does not reach full voltage causes the data transferred into the MPU on a Read operation to differ from that contained in the respective bit of Output Register A.

- **Section B Peripheral Data (PB_0~PB_7)**

The peripheral data lines in the B Section of the PIA can be programmed to act as either inputs or outputs in a similar manner to PA_0~PA_7. However, the output buffers driving these lines differ from those driving lines PA_0~PA_7. They have three-state capability, allowing them to enter a high impedance state when the peripheral data line is used as a input. In addition, data on the peripheral data lines PB_0~PB_7 will be read properly from those lines programmed as outputs even if the voltages are below 2.0 volts for a "High". As outputs, these lines are compatible with standard TTL and may also be used as a source of up to 2.5 milliampere (typ.) at 1.5 volts to directly drive the base of a transistor switch.

- **Interrupt Input (CA_1 and CB_1)**

Peripheral Input lines CA_1 and CB_1 are input only lines that set the interrupt flags of the control registers. The active transition for these signals is also programmed by the two control registers.

- **Peripheral Control (CA_2)**

The peripheral control line CA_2 can be programmed to act as an interrupt input or as a peripheral control output. As an output, this line is compatible with standard TTL. The function of this signal line is programmed with Control Register A.

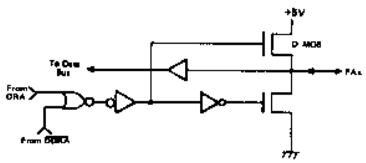
- **Peripheral Control (CB_2)**

Peripheral Control line CB_2 may also be programmed to act as an interrupt input or peripheral control output. As an input, this line has "High" input impedance and is compatible with standard TTL. As an output it is compatible with standard TTL and may also be used as a source of up to 2.5 milliampere (typ) at 1.5 volts to directly drive the base of a transistor switch. This line is programmed by Control Register B.

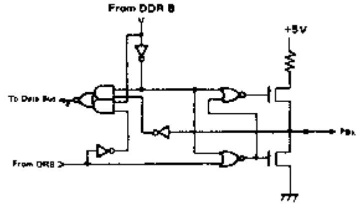
(NOTE) 1. Interrupt inputs CA_1, CA_2, CB_1 and CB_2 shall be used at normal "High" level. When interrupt inputs are "Low" at reset ($\overline{RES}$ = "Low"), interrupt flags CRA6, CRA7, CRB6 and CRB7 may be set.
2. Pulse width of interrupt inputs CA_1, CA_2, CB_1 and CB_2 shall be greater than a E cycle time. In the case that "High" time of E signal is not contained in Interrupt pulse, an interrupt flag may not be set.

- **The equivalent Circuit of the Lines on Peripheral side**

The equivalent circuit of the lines on Peripheral side is shown in Fig. 15. The output circuits of A port is different from that of B port. When the port is used as input, the input is pullup to V_{CC} side through load MOS in A port and B port becomes "Off" (high impedance).



(a) Section A



(b) Section B

Figure 15 Peripheral Data Bus

■ **INTERNAL CONTROLS**

There are six locations within the PIA accessible to the MPU data bus: two Peripheral Registers, two Data Direction Registers, and two Control Registers. Selection of these locations is controlled by the RS_0 and RS_1 inputs together with bit 2 in the Control Register, as shown in Table 1.

Table 1 Internal Addressing

RS_1	RS_0	Control Register Bit CRA2	Control Register Bit CRB2	Location Selected
0	0	1	x	Peripheral Register A*
0	0	0	x	Data Direction Register A
0	1	x	x	Control Register A
1	0	x	1	Peripheral Register B*
1	0	x	0	Data Direction Register B
1	1	x	x	Control Register B

x = Don't Care
* Peripheral interface register is a generic term containing peripheral data bus and output register.

- **Initialization**

A "Low" reset line has the effect of zeroing all PIA registers. This will set PA_0~PA_7, PB_0~PB_7, CA_2 and CB_2 as inputs, and all interrupts disabled. The PIA must be configured during the restart program which follows the reset.

Details of possible configurations of the Data Direction and Control Register are as follows.

- **Data Direction Registers (DDRA and DDRB)**

The two Data Direction Registers allow the MPU to control the direction of data through each corresponding peripheral data line. A Data Direction Register bit set at "0" configures the corresponding peripheral data line as an input; a "1" results in an output.

- **Control Registers (CRA and CRB)**

The two Control Registers (CRA and CRB) allow the MPU to control the operation of the four peripheral control lines CA_1, CA_2, CB_1 and CB_2. In addition they allow the MPU to enable the interrupt lines and monitor the status of the interrupt flags. Bits 0 through 5 of the two registers may be written or read by the MPU when the proper chip select and register select signals are applied. Bits 6 and 7 of the two registers are read only and are modified by external interrupts occurring on control lines CA_1, CA_2, CB_1 or CB_2. The format of the control words is shown in Table 2.

Table 2 Control Word Format

	7	6	5	4	3	2	1	0
CRA	IRQA1	IRQA2	CA_2 Control			DDRA Access	CA_1 Control	
CRB	IRQB1	IRQB2	CB_2 Control			DDRB Access	CB_1 Control	

Data Direction Access Control Bit (CRA2 and CRB2)

Bit 2 in each Control register (CRA and CRB) allows selection of either a Peripheral Interface Register or the Data Direction Register when the proper register select signals are applied to RS_0 and RS_1.

Interrupt Flags (CRA6, CRA7, CRB6, and CRB7)

The four interrupt flag bits are set by active transitions of signals on the four Interrupt and Peripheral Control lines when those lines are programmed to be inputs. These bits cannot be set directly from the MPU Data Bus and are reset indirectly by a Read Peripheral Data Operation on the appropriate section.

Control of CA_1 and CB_1 Interrupt Lines (CRA0, CRB0, CRA1, and CRB1)

The two lowest order bits of the control registers are used to control the interrupt input lines CA_1 and CB_1. Bits CRA0 and CRB0 are used to enable the MPU interrupt signals $\overline{IRQA}$ and $\overline{IRQB}$, respectively. Bits CRA1 and CRB1 determine the active transition of the interrupt input signals CA_1 and CB_1 (Table 3)

Control of CA_2 and CB_2 Peripheral Control Lines (CRA3, CRA4, CRA5, CRB3, CRB4, and CRB5)

Bits 3, 4 and 5 of the two control registers are used to control the CA_2 and CB_2 Peripheral Control lines. These bits determine if the control lines will be an interrupt input or an output control signal. If bit CRA5 (CRB5) is "0" CA_2 (CB_2) is an interrupt input line similar to CA1 (CB1) (Table 4). When CRA5 (CRB5) is "1", CA_2 (CB_2) becomes an output signal that may be used to control peripheral data transfers. When in the output mode, CA_2 and CB_2 have slightly different characteristics (Table 5 and 6).

Table 3 Control of Interrupt Inputs CA_1 and CB_1

CRA1 (CRB1)	CRA0 (CRB0)	Interrupt Input CA_1 (CB_1)	Interrupt Flag CRA7 (CRB7)	MPU Interrupt Request $\overline{IRQA}$ ($\overline{IRQB}$)
0	0	↓ Active	Set "1" on ↓ of CA_1 (CB_1)	Disabled – $\overline{IRQ}$ remains "High"
0	1	↓ Active	Set "1" on ↓ of CA_1 (CB_1)	Goes "Low" when the interrupt flag bit CRA7 (CRB7) goes "1"
1	0	↑ Active	Set "1" on ↑ of CA_1 (CB_1)	Disabled – $\overline{IRQ}$ remains "High"
1	1	↑ Active	Set "1" on ↑ of CA_1 (CB_1)	Goes "Low" when the interrupt flag bit CRA7 (CRB7) goes "1"

(Notes) 1. ↑ indicates positive transition ("Low" to "High")
2. ↓ indicates negative transition ("High" to "Low")
3. The Interrupt flag bit CRA7 is cleared by an MPU Read of the A Peripheral Register and CRB7 is cleared by an MPU Read of the B Peripheral Register.
4. If CRA0 (CRB0) is "0" when an interrupt occurs (Interrupt disabled) and is later brought "1", $\overline{IRQA}$ ($\overline{IRQB}$) occurs after CRA0 (CRB0) is written to a "1".

Table 4 Control of CA_2 and CB_2 as Interrupt Inputs – CRA5 (CRB5) is "0"

CRA5 (CRB5)	CRA4 (CRB4)	CRA3 (CRB3)	Interrupt Input CA_2 (CB_2)	Interrupt Flag CRA6 (CRB6)	MPU Interrupt Request $\overline{IRQA}$ ($\overline{IRQB}$)
0	0	0	↓ Active	Set "1" on ↓ of CA_2 (CB_2)	Disabled – $\overline{IRQ}$ remains "High"
0	0	1	↓ Active	Set "1" on ↓ of CA_2 (CB_2)	Goes "Low" when the interrupt flag bit CRA6 (CRB6) goes "1"
0	1	0	↑ Active	Set "1" on ↑ of CA_2 (CB_2)	Disabled – $\overline{IRQ}$ remains "High"
0	1	1	↑ Active	Set "1" on ↑ of CA_2 (CB_2)	Goes "Low" when the interrupt flag bit CRA6 (CRB6) goes "1"

(Notes) 1. ↑ indicates positive transition ("Low" to "High")
2. ↓ indicates negative transition ("High" to "Low")
3. The interrupt flag bit CRA6 is cleared by an MPU Read of the A Peripheral Register and CRB6 is cleared by an MPU Read of the B Peripheral Register.
4. If CRA3 (CRB3) is "0" when an interrupt occurs (Interrupt disabled) and is later brought "1", $\overline{IRQA}$ ($\overline{IRQB}$) occurs after CRA3 (CRB3) is written to a "1".

Table 5 Control of CB_2 as an Output — CRB5 is "1"

CRB5	CRB4	CRB3	CB_2 Cleared	CB_2 Set
1	0	0	"Low" on the positive transition of the first E pulse after MPU Write "B" Data Register operation.	"High" when the interrupt flag bit CRB7 is set by an active transition of the CB_1 signal. (See Figure 16)
1	0	1	"Low" on the positive transition of the first E pulse after an MPU Write "B" Data Register operation.	"High" on the positive edge of the first "E" pulse following an "E" pulse which occurred while the part was deselected. (See Figure 16)
1	1	0	"Low" (The content of CRB3 is output on CB_2)	
1	1	1	"High" (The content of CRB3 is output on CB_2)	

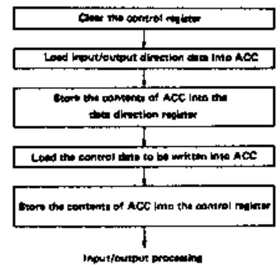
Table 6 Control of CA_2 as an Output — CRA5 is "1"

CRA5	CRA4	CRA3	CA_2 Cleared	CA_2 Set
1	0	0	"Low" on negative transition of E after an MPU Read "A" Data Operation.	"High" when the interrupt flag bit CRA7 is set by an active transition of the CA_1 signal. (See Figure 16)
1	0	1	"Low" on negative transition of E after an MPU Read "A" Data operation.	"High" on the negative edge of the first "E" pulse which occurs during a deselect. (See Figure 16)
1	1	0	"Low" (The content of CRA3 is output on CA_2)	
1	1	1	"High" (The content of CRA3 is output on CA_2)	

■ PIA OPERATION

• **Initialization**

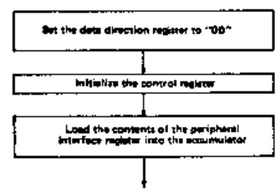
When the external reset input $\overline{RES}$ goes "Low", all internal registers are cleared to "0". Peripheral data port (PA_0~PA7, PB_0~PB_7) is defined to be input and control lines (CA_1, CA_2, CB_1 and CB_2) are defined to be the interrupt input lines. PIA is also initialized by software sequence as follows.



- Program the data direction register access bit of the control register to "0" to allow to access the dada direction register.
- The data of the control line function is set into the accumulator, of which Data Direction Register Access Bit shall be programmed to "1".
- Transfer the control data from the accumulator into the control register.

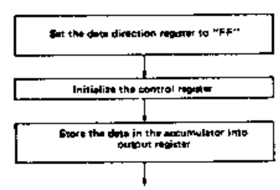
• **Read/Write Operation Not Using Control Lines**

<Read Operation>




```
CLR   CRA    • Clear the DDRA access bit of the control register to "0".
CLR   DDRA   • Clear all bits of the dada direction register.
LDAA  #$04   • Set DDRA access bit of the control register to "1" to allow
STAA  CRA      to access the peripheral interface register.

LDAA  PIRA
```

<Write Operation>



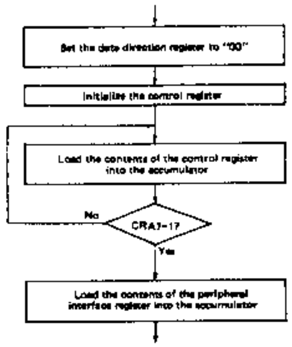

```
CLR   CRA    • Set DDRB access bit of the control register to "0".
LDAA  #$FF } • Set all bits of the data direction register to "FF".
STAA  DDRB }

LDAA  #$04 } • Set DDRB access bit of the control register to "1" to allow to
STAA  CRB  }   access the peripheral interface register.

LDAA  DATA } • Write the data into the peripheral interface register.
STAA  PIRB }
```

- **Read/Write Operating Using Control Lines**

Read/write request from peripherals shall be put into the control lines as an interrupt signal, and then MPU reads or writes after detecting interrupt request.

< Read >

The following case is that Port A is used and that the rising edge of CA_1 indicates the request for read from peripherals.



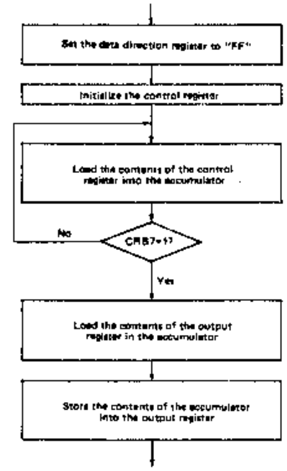

```
      CLR   CRA    • Set the DDRA access bit to "0".
      CLR   DDRA   • Set all bits of the data direction register to "0".
      LDAA  #$05   • Program the rising edge of CA1 to be active. IRQA is masked
      STAA  CRA      and DDRA access bit is set to 1.
LOOP  LDAA  CRA    • Check whether the read request comes from peripherals
      BPL   LOOP     or not.
      LDAA  PIRA   • Load the data from the peripheral interface register into the
                     accumulator. CRA flag is reset after this read operation.
```

To read the peripheral data, the data is directly transfered to the data buses D_0~D_7 through PA_0~PA_7 or PB_0~PB_7 and they are not latched in the PIA. If necessary, the data should be held in the external latch until MPU completes reading it.

When initializing the control register, interrupt flag bit (CRA7, CRA6, CRB7, CRB6) cannot be written from MPU. If necessary the interrupt flag must be reset by dummy read of Peripheral Register A and B.

<Write>

Write operation using the interrupt signal is as follows. In this case, B port is used and interrupt request is input to CB_1. And the $\overline{IRQ}$ flag is set at the rising edge of CB_1.




```
      CLR   CRB    • Set DDRB access bit to "0"
      LDAA  #$FF } • Set all bits of DDRB to output "1".
      STAA  DDRB }
      LDAA  #$06 } • Program the rising edge of CB1 to be active. IRQB is masked
      STAA  CRB  }   and DDRB access bit is set to "1".
LOOP  LDAA  CRB  } • Check whether the write request comes from peripherals
      BPL   LOOP }   or not.
      LDAA  PIRB   • Reset the CRB7 flag by the dummy read of the
                     peripheral interface register.
      STAB  PIRB   • Store the data of the accumulator B to the peripheral
                     interface register.
```

Interrupt request flag bits (CRA7, CRA6, CRB7 and CRB6) cannot be written and they cannot be also reset by write operation to the peripheral interface register. So dummy read of peripheral interface register is needed to reset the flags.

To accept the next interrupt, it is essential to reset indirectly the interrupt flag by dummy read of peripheral interface register.

Software poling method mentioned above requires MPU to continuously monitor the control register to detect the read/write request from peripherals. So other programs cannot run at the same time. To avoid this problem, hardware interrupt may be used. The MPU is interrupted by $\overline{IRQA}$ or $\overline{IRQB}$ when the read/write request is occurred from peripherals and then MPU analyzes cause of the interrupt request during interrupt processing.

- **Handshake Mode**

The functions of CRA and CRB are similar but not identical in the hand-shake modes. Port A is used for read hand-shake operation and Port B is used for write hand-shake mode.

CA_1 and CB_1 are used for interrupt input requests and CA_2 and CB_2 are control outputs (answer) in hand-shake mode.

Fig. 16, Fig. 17 and Fig 18 show the timing of hand-shake mode.

< Read Hand-shake Mode >

CRA5="1", CRA4="0" and CRA3="0"

1. A peripheral device puts the 8-bit data on the peripheral data lines after the control output CA_2 goes "Low".
2. The peripheral requests MPU to read the data by using CA_1 input.
3. CRA7 flag is set and CA_2 becomes "High" (CA_2 automatically becomes "High" by the interrupt CA_1). This indicates the peripheral to maintain the current data and not to transfer the next data.
4. MPU accepts the read request by $\overline{IRQA}$ hardware interrupt or CRA read. Then MPU reads the peripheral register A.
5. CA_2 goes "Low" on the following edge of read Enable pulse. This informs that the peripheral can set the next data to port A.

<Write Hand-shake>

CRB5 = "1", CRB4 = "0" and CRB3 = "0"

1. A peripheral device requests MPU to write the data by using CB_1 input. CB_2 output remains "High" until MPU write data to the peripheral interface register.
2. CRB7 flag is set and MPU accepts the write request.
3. MPU reads the peripheral interface register to reset CRB7 (dummy read).
4. Then MPU write data to the peripheral interface register. The data is output to port B through the output register.
5. CB_2 automatically becomes "Low" to tell the peripheral that new data is on port B.
6. The peripheral read the data on Port B peripheral data lines and set CB_1 to "Low" to tell MPU that the data on the peripheral data lines has been taken and that next data can be written to the peripheral interface register.

<Pulse mode>

CRA5 = "1", CRA4 = "0" and CRA3 = "1"
CRB5 = "1", CRB4 = "0" and CRB3 = "1"
This mode is shown in Figure 16, Figure 19 and Figure 20.

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.

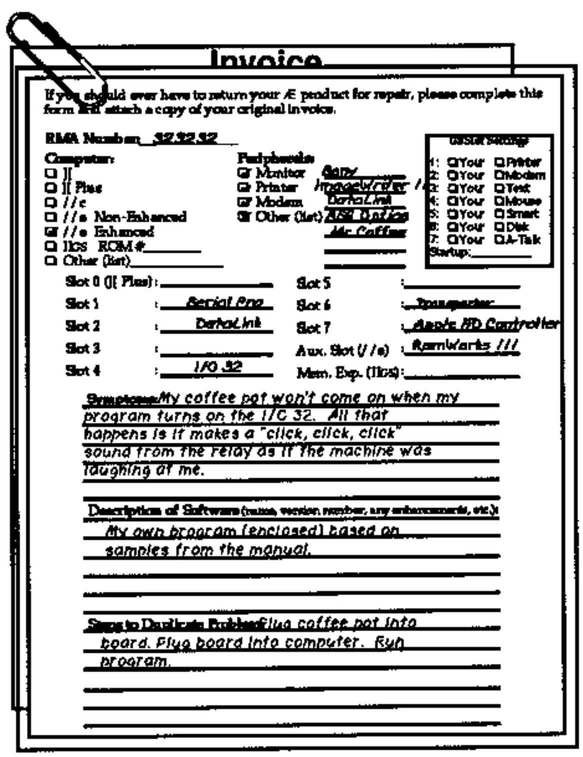
- ❑ Record the RMA number for your own records.
- ❑ Write the RMA number on the outside of the package you send to us.
- ❑ Write the RMA number at the top of the return form included with your product package.

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 (preferrably 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. Æ 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 Æ's option.

Any misuse, abuse, or non-Æ 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 Æ 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.