MICROPORT–XL

FOR ATARI XL COMPUTERS

TABLE OF CONTENTS

****
*** IMPORTANT NOTICE ***
****

Supra Corporation warrants MICROPORT XL to be free from any defects in material and workmanship for a period of one (1) year from the date of purchase by the original owner, or until the device is assembled by the purchaser. Supra Corporation does not make any warranties as to the abilities of this device to meet any federal or state regulations concerning emitted radiation. Any liability incurred as a result of the construction of this device or its operation is incurred by the purchaser. Supra Corporation can not under any circumstances be held liable for any damages, direct or incidental, arising from the use or misuse of this product. Supra Corporation can not under any circumstances be held liable for any damage to any computer equipment, peripheral equipment, or interfaced equipment which is connected to this device by the purchaser or any person other than a properly authorized employee of Supra Corporation.

****

INTRODUCTION

MICROPORT XL for the Atari XL computers is an expansion peripheral for the parallel bus to allow interfacing of real world devices to Atari personal computers. It provides 20 I/O lines to be used as input, output, or any combination of input and output lines.

DESCRIPTION

MICROPORT XL implements a PIA (6520/6821) chip to expand the Atari XL computers input/output capabilities. This is the same chip the Atari XL computers use for joystick ports and memory bank switching. The MICROPORT XL PIA can be mapped into the Atari computer at one of four(4) possible locations and can generate interrupts on user programmed events. It has two 8-bit bi-directional I/O ports with a separate data direction register and two handshake lines for each port.

The chips on MICROPORT XL are socketed for easy replacement and servicing. All lines on the Atari expansion bus are brought out to solder points for easy access. Ten square inches of proto-board area are provided with solder-thru holes for easy experimenting and testing. A separate +5 volt power supply is included to power the PIA and any interfacing electronics.

MICROPORT XL can be used for many applicatinns. Some examples are LED display, printer interface (parallel and/or serial), home lighting controller, home security controller, eprom programmer, video tape/disk controller, etc. The possibilities are endless, and left up to the users imagination. Some of these ideas will be used on a limited basis for examples in this documentation. Each example will include a schematic on how to wire it using MICROPORT XL and a simplified software example showing how to drive the hardware. Example software will be in either assembly and/or BASIC. These examples are intended to be used for demonstration purposes only, therefore the software and/or the hardware may not be practical for normal use.

Supra Corporation will be very interested in any practical applications the user may implement with MICROPORT XL. If these applications are accepted they may be published in a quarterly MICROPORT XL users newsletter, included with future printings of the MICROPORT XL users guide as examples, or even possibly a Supra Corporation product.

All submitted applications should include all the following information. This data will be used to supply the necessary information about the application. Incomplete submissions will not be considered.

MICROPORT XL APPLICATION PACKET

I. Documentation
   - A. General Description
   - B. Theory of Operation
   - C. Explanation
     1. Hardware
     2. Software
   - D. Construction Hints and Helps
   - E. Checkout and Calibration
   - F. Instruction and Usage

II. Hardware Information
   - A. Schematic Drawing
   - B. Parts List
   - C. Pictorial Drawing of Board
   - D. Black & White Pictures (optional)

III. Software Documentation
   - A. Commented Source code.
   - B. Disk Media
     1. Source Code
     2. Object Code
     3. Documentation
   - C. Flowchart (optional)

Send all correspondence to:

Supra Corporation
1133 Commercial Way
Albany, OR 97321
ATTN: MICROPORT XL User Support

INTERFACING

COMPUTER SIDE

The PIA chip can be located at any one of four(4) locations in the $0000 - $0FFF input/output block on the Atari XL computer.

PAGE		DIP SWITCHES (4-3-2-1)	
$0100	(53504)	0111	
$0500	(54528)	0101	(1 = ON)
$0600	(54784)	1001	(0 = OFF)
$0700	(55040)	0001	

NOTE:

If other pages are selected there may be conflicts with existing chips in that area.

The registers of the PIA are arranged in memory just like the PIA inside the Atari computers. The first two memory locations are the data registers, and the next two memory locations are the control registers.

Page + 0 => Porta data register
Page + 1 => Portb data register
Page + ? => Control register for porta
Page + 3 => Control register for portb

NOTE:

The rest of the page is mirror images of the first four(4) locations.

This arrangement is not necessarily standard for PIA chips, but to remain like the Atari computers the registers in MICROPORT XL were arranged in this same order.

The MICROPORT XL PIA is tied into the Atari XL IRQ interrupt line to allow for interrupt processing. Both ports in the PIA have an interrupt that can be enabled to pause the computer during normal processing and force it to handle that interrupt. These interrupts are normally turned off and must be enabled through software before they can he used.

To test for an interrupt generated by the PIA, the program must change the vector at $0216 and $0217 to point to a new routine. This is the vector that ALL IRQ interrupts are passed through before going to the normal interrupt processor. A test should be make to see if MICROPORT XL PIA generated the IRQ, if it did not then control must be passed back to the normal IRQ processor. If MICROPORT XL did generate the IRQ then control should be passed to the appropriate user software.

EXTERNAL SIOE

Interfacing to the world outside of the computer is the real purpose for MICROPORT XL. The two ports and associated handshake lines can be used for both input from external devices and/or controlling of external devices. All of the lines are at TTL levels and should only be used to drive other chips. To drive more powerful devices opto-isolators and/or transistors drivers are recommended. The MICROPORT XL can be used for many creative purposes, some of which are included with your MICROPORT XL information packet.

MICROPORT XL SPECIFICATIONS

To use the PIA chip on the MICROPORT XL to its maximum capability a thorough understanding of the specification sheets from the manufacturer of the PIA is recommended. These specifications sheets are included with this documentation and go into much more detail that this discussion. They are recommended study material. This documentation will attempt to give the user a general understanding of the PIA and how to interface it to some applications.

I/O LINES

The MICROPORT XL has 20 I/O lines which can be used for many different purposes. The two data ports (Porta and Portb) can be used for either input or output, or any combination of inputs and outputs. These ports are not completely identical, each is slightly different in its drive capability and input sensitivity. For simplicity sake this documentation will assume that each port is identical and interchangeable. The remaining four(4) I/O lines are generally used for hand-shaking and have some special features associated with them. The CA1 and CB1 lines can be used for input only. These two lines can generate an IRQ request if this option is enabled. The CA2 and CB2 lines are more versatile handshake lines. They can be used as interrupt input lines or as output control lines. As inputs they function like the CA1 and CB1 lines, but as outputs they are very powerful. They can be used as simple output, or made to transition on a change of their associated CA1 or CB1 line. Each handshake line is controlled through a bit in the control register associated with that port. An abbreviated discussion of the cnntrol register and its functions follows.

CONTROL REGISTER

The control registers are the heart of the control system for each port. Each bit in the register controls a different function for that port.

BIT	FUNCTION
7	IRQ has occurred on the Cx1 line
6	IRQ has occurred on the Cx2 line
5	Mode select for Cx2 line
4	Cx2 IRQ transition select / nutput control
3	Cx2 IRQ enable / strobe restore
2	Port mode select
1	Cxl IRQ transition select
0	Cxl IRQ enable

NOTE: The small 'x' in the Cx1 and Cx2 labels corresnonds to the 'a' or 'b' in Porta and Portb. Insert the appropriate letter for the port in reference.

BIT 7 Bit 7 shows the status of the last transition of the Cx1 line that meets the criteria set in bit 1.
1 = A valid transition has occurred on Cx1
0 = No valid transition has occurred on Cx1
For example using Porta Control Register:
Bit 1 = 1 (flag on a low-to-high transition)
Bit 7 = 0 (NO transition to high yet on CA1)
Bit 7 = 1 (CA1 has gone high)
To clear Bit 7 for the next transition simply read the port associated with that register. In the example above to clear the transition just read Porta.

BIT 6 Bit 6 is used only when the Cx2 line is used as an input. If the Cx2 line is being used as an output this bit is unused.
This bit works like Bit 7, except on the Cx2 line, and transition criteria is set by bit 4. It is also cleared by a read of the assnciated port.

BIT 5 Bit 5 sets the mode for the Cx2 line.
1 = the Cx2 line is an output line.
0 = the Cx2 line is an input line.

BIT 4 — Bit 4 has different purposes for each mode of the Cx2 line. If Cx2 is being used as input (bit 5 = 0) it sets the transition criteria for the Cx2 line. This criteria is used by bit 6 and the IRQ (if enabled) to flag a transition
1 = Set bit 6 on a low-to-high (positive) transition
0 = Set bit 6 on a high-to-low (negative) transition
If Cx2 is being used as an output (bit 5 = 1) this bit controls the output set criteria for Cx2.
1 = Cx2 will follow the status of bit 3. If bit 3 is a '1' the Cx2 will be high, and if bit 3 is a '0' then Cx2 will be low.
0 = Cx2 line will go low after a processor read of the data on the port associated with the control register. This can be used to tell the sending device that the data has been read by the computer. This assumes that the port is being used as an input

BIT 3 — Bit 3 is the enable IRQ for the Cx2 line when Cx2 is being used as input (bit 5 = 0).
1 = generate an IRQ interrupt when bit 6 is set
0 = DO NOT generate an IRQ for the Cx2 line
If Cx2 is being used as an output (bit 5 = 1) this bit works in conjunction with bit 4 and controls the reset criteria for the Cx2 line.
1 = return Cx2 high one clock cycle after the read of the data port. This can be used with the above set criteria to create a negative pulse on every processor read of the associated port.
0 = return Cx2 high on the next transition of the Cx1 input line as set by the transition criteria from bit 1. This allows for an acknowledge hand-shake of received data.

BIT 2 — Bit 2 selects the mode of the data port.
1 = the data port is used as input/output.
0 = the data port is used to set the data direction for each bit of the port. By writing a value out to the data port in this mode, the data direction of each line is selected. A '1' in a bit position causes that bit to be used as an output, and a '0' causes that bit to be used as an input. To make all the lines outputs write a $FF (255) to the data port, or to make all the lines inputs write a 0. Any combination of '1' and '0' can be used. A $F0 (240) will make bits 7-4 outputs and bits 3-0 inputs.
Remember to set bit 2 of the control register back to a '1' to use the data port after setting the data direction.

BIT 1 — Bit 1 sets the transition criteria for the Cx1 line.
1 = set bit 7 on a low-to-high (positive) transition
0 = set bit 7 on a high-to-low (negative) transition

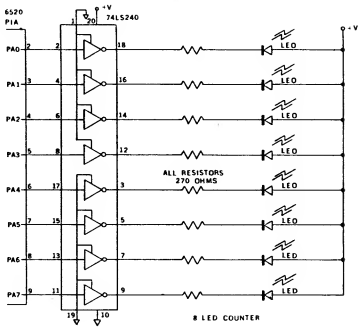
Bit 0 — Bit 0 is the enable IRQ interrupt bit for the Cx1 line.
1 = generate an IRQ interrupt when bit 7 is set
0 = DO NOT generate an IRQ from Cx1

USING MICROPORT XL

Using MICROPORT XL is really very simple. The technical data can be hard to read, but it is very necessary information to use the PIA chip to its fullest. To use MICROPORT XL in its simplest mode just set the data direction and then read or write data to the port.

Figure 1 is a simple example for learning how to use Porta to turn on and off 8 LEO's. The following programs will setup Porta and make the 8 LEO's count from 0 to $FF (255) in binary. Example 1 is in BASIC and example 2 is in 6502 assembly. Both examples assume that MICROPORT XL has been located at $0700 (55040) in memory.

FIGURE 1



EXAMPLE 1

```
10 PORTA = 55040     : REM DATA PORT A
20 PACTL = 55042     : REM CONTROL REGISTER FOR PORT A
30 POKE PACTL, 0     : REM SET DATA DIRECTION
40 POKE PORTA, 255   : REM MAKE ALL LINES OF PORTA OUTPUTS
50 POKE PACTL, 4     : REM TURN PORT BACK TO DATA
60 FOR LOOP = 0 TO 255       : REM COUNT FROM 0 TO 255
70 POKE PORTA, LOOP          : REM WRITE DATA TO PORT A
80 FOR TIME = 0 TO 100       : REM DELAY A LITTLE SO THAT
90 NEXT TIME                 : REM ...WE CAN SEE THINGS
100 NEXT LOOP                : REM GO DO LOOP AGAIN
110 END
```

EXAMPLE 2

```
Porta = $0700                ;Data porta
Pactl = $0702                ;Control register for porta
                             ;
Rtclok = $14                 ;Clock (increments every 1/60 sec)
                             ;
*= $4000                     ;Put program at a safe place
                             ;
Main                         ;Main loop of counter program
          LDA #$00           ;Set control register to
          STA Pactl          ;...setup of data direction
          LDA #$FF           ;Make all lines of porta
          STA Porta          ;...into outputs
          LDA #$04           ;Set porta back to data
          STA Pactl          ;...with no interrupts
                             ;
          LDX #$00           ;Start counter at 0
Loop                         ;Loop arnund till done
          STX Porta          ;Put counter value in porta
          LDA #$20           ;Set value (1/2 sec) to wait
          JSR Delay          ;Go wait a bit
          INX                ;Count up one
          BNE Loop           ;Done yet? - go around again
          RTS                ;All done sn return
                             ;
Delay                        ;Subroutine to delay 'A 60th sec
          LDY #$00           ;Set clock to zero
          STY Rtclok         ;
Delay.1                      ;
          CMP Rtclok         ;Has clock got to delay value
          BNE Delay.1        ;Nope - wait some more
          RTS                ;Done - so return
```


ATARI PARALLEL BUS

MICROPORT XL gives access to the Atari Parallel Bus although it was not intended to be used as a standard device for the parallel bus. If the user intends to try and modify MICROPORT XL to be used as a parallel device, we recommend reading the series of articles published in ANTIC magazine by Earl Rice that started in January 1985. This series of articles will give some idea of how to set up a device on the parallel bus. It was not the intention of this product to teach the use of the parallel bus. To allow the experienced user of MICROPORT XL to use the power of the parallel bus, a brief description of the available lines will follow.

NOTE: Active low lines are preceded by the word NOT.

PARALLEL BUS LINE DESCRIPTION

1. Ground
2. NOT EXTSEL input
3. Address line A0
4. Address line A1
5. Address line A2
6. Address line A3
7. Address line A4
8. Address line A5
9. Address line A6
10. Ground
11. Address line A7
12. Address line A8
13. Address line A9
14. Address line A10
15. Address line A11
16. Address line A12
17. Address line A13
18. Address line A14
19. Ground
20. Address line A15
21. Data I/O D0
22. Data I/O D1
23. Data I/O D2
24. Data I/O D3
25. Data I/O D4
26. Data I/O D5
27. Data I/O D6
28. Data I/O D7
29. Ground
30. Ground
31. Phase 2 Clock output
32. Ground
33. Reserved
34. NOT RESET input
35. NOT IRQ input
36. R/W input to processor
37. Reserved
38. EXTEN (external enable output)
39. Reserved
40. NOT REF
41. NOT CAS (RAM column select)
42. Ground
43. NOT MPD (math pack disable)
44. NOT RAS (RAM row select)
45. Ground
46. L R/NOT W (late read/write)
47. +5 volts regulated (600 only)
48. +5 volts regulated (600 only)
49. Audio in input
50. Ground

ASSEMBLY INSTRUCTIONS FOR MICROPORT XL

PACKAGE LIST

1. ( ) 1 ea MICROPORT XL Circuit Board
2. ( ) 1 ea 50 pin edge connector
3. ( ) 1 ea Power Jack
4. ( ) 1 ea Power pack (110 VAC to 9 VDC)
5. ( ) 1 ea 6520/6821 PIA (40 pin chip)
6. ( ) 1 ea 74LS688 or AM25LS252 8 bit Comparator (20 pin chip)
7. ( ) 1 ea 7805 +5 volt regulator
8. ( ) 1 ea 4 position DIP Switch
9. ( ) 1 ea 40 pin socket
10. ( ) 1 ea 20 pin socket
11. ( ) 1 ea 100 uF 10v electrolytic capacitor
12. ( ) 1 ea .22 uF 25v disc capacitor (224 on it)
13. ( ) 2 ea .1 uF 25v disc capacitor (104 on it)
14. ( ) 4 ea 47K 1/4 watt resistor (Yellow Purple Red)
15. ( ) 1 ea 12 Ohm 1/4 watt resistor (Brown Red Black)
16. ( ) 1 ea MICROPORT XL Documentation
17. ( ) 1 ea MICROPORT XL CLOCK application packet
18. ( ) 1 ea MICROPORT XL EPROMer application packet

TOOLS NEEDED

1. Soldering Pencil (MAX 45 WATT) DO NOT USE SOLDERING IRON!!!
2. Small Diameter Solder (60/40 tin/lead) DO NOT USE ACID CORE!!
3. Side cutters

ASSEMBLY

Locate the MICROPORT XL circuit board. Position the board flat on the table with the words 'MICROPORT XL' are at the top back, but still correct for reading. All instructions will assume that the board is in this position in front of you. All components will be inserted from the TOP of the board and soldered from the BOTTOM.

(1) First insert the 50 pin edge connector. This connector fits into the two rows of 25 holes along the front edge of the board. Insertion of this connector can he tricky. The easiest method is to start at one end and work your way to the other end. An Xacto knife or small flat blade screwdriver can be used to gently move the pins to align them with the correct holes. Solder in this connector.

(2) Insert and solder the two sockets. Place the alignment notch or white dot to the right side of the board. This will allow you to easily see which pin is number 1 on the chips that are to be inserted into these sockets. Do NOT insert the chips at this time.

(3) Insert and solder the two(2) .1 uF bypass capacitors. One goes at the right end of each chip socket. These capacitors will usually have a 104 on them.

(4) Insert and solder the four(4) 47K resistors into the holes forward from the 20 pin chip.

(5) Insert and solder the 4 position DIP switch to the left of the four resistors. Position switch number 1 to the rear of the board and number 4 to the front of the board.

(6) Insert and solder the power plug into the holes at the left rear of the board.

(7) Insert and solder the 10 ohm resistor into the two horizontal holes to the immediate right of the power plug.

(8) Insert and solder the .22 uF capacitor into the two vertical holes to the right of the resistor. This capacitor will usually have a 224 on it.

(9) Insert and solder the 100 uF capacitor into the holes about an inch from the right of the board at the rear. Note the polarity of this capacitor. The + side goes to the + on the circuit board.

(10) Last insert and solder the 7805 +5 volts regulator into the 3 holes at the center rear of the board. The tab side goes to the right with the writing on top. The pins can be bent to allow the regulator to lay flag on the board.

CHECKOUT

Before powering up this device double check the board, both top and bottom, for any solder bridges or shorts. Without plugging in MICROPORT XL into a computer plug in the power supply to a wall outlet and the other end into the power jack. Using a volt meter check between the two power rails on the left side of the circuit board. The meter should show +5 volts. Then check between pins 10 and 20 on the 20 socket chip. There should also be +5 volts. Last check for +5 volts between pins 1 and 20 on the 40 socket chip. If all this checks out OK you are ready to check out MICROPORT XL on a computer, If not recheck solder joints, polarity of the 100 uF capacitor, and the direction of the 7805 regulator.

Insert the IC into the appropriate sockets. Make sure that the notch or detent on the chip aligns with the notch or detent on the socket. Before continuing double check each chip to ensure that every pin was not bent under or missed the hole on the socket.

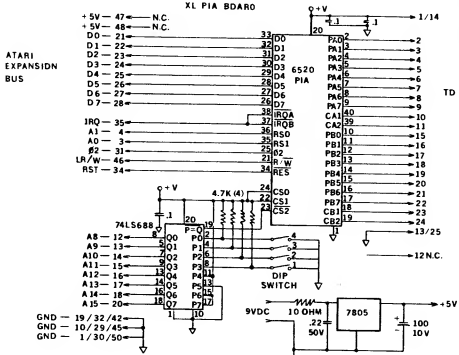
Set the DIP switch on MICROPORT XL to the $0700 page (1 ON the rest OFF). Unplug the power cord from the power jack on the MICROPORT XL and plug the 50 pin connector into the parallel expansion bus of an Atari XL computer. You may need to remove the protective plastic cover from the back of the computer. Atari ships the computers with this cover to keep stray material from entering and shorting out this bus. Turn on the computer power and verify that the computer powers up correctly. If it does not look for a short or solder bridge on the 50 pin connector and on the two chips.

Turn the computer off and plug the power cord back into the power jack on MICROPORT XL. Repower up the computer. If the computer powers up correctly you have probably wired your MICROPORT XL correctly and are ready for the last check out.

Load and run a monitor or debugger program. Display memory

at the $0700 page. Put a $00 into locations $0702 and $0703. This allows us to set the data direction for the ports. Put a $FF into $0700 and $0701. This sets the direction to all outputs on both ports. Put a $04 into $0702 and $0703 to set the ports back to data. Put a logic probe or volt meter on a pin from 2 to 9 of the 6520/6821 PIA. Put a $FF into $0700 and the probe or meter should show +5 volts. Put a $00 there and the probe or meter should show 0 volts. Do the same for pins 10 to 17 of the 6502/6821 PIA using $0701. If all the lines toggle your assembly was successful and you are ready to build your first project on MICROPORT XL.

If the above test was not successful check the OIP switches and the wiring on both chips. If only one line does not function but the rest seem OK, check for bent pins on the 6520/6821 PIA or on the socket.



CLOCK

Clock for MICROPORT XL is a four(4) digit multiplexed clock display driven completely by interrupts. The circuit can be used as a four digit clock display or it can be used to display other current information while the normal computer system is running. If the program is positioned correctly in memory it will run transparent from the normal system or other programs.

THEORY OF OPERATION

The display used in CLOCK is a four(4) digit multiplexed display. This system is the simplest to use as it only requires 11 lines to control it. Seven(7) lines are used to access the segments, one to each identical segment of all the digits, and four(4) are used to access the digits, one to each of the digits. In other words each identical segment of each digit is connected together giving seven(7) lines total, and one line is connected to the common cathode of each digit giving four(4) lines total. All totaled requiring only 11 lines to control the complete display.

The outputs of most PIA's can NOT directly drive a LED, so a 74LS240 octal tri-state inverter is used to drive the LED's. Since we have eight(8) inverters and only need seven(7) to drive the segments, the left inverter one will be used to drive the colon between the hours and minutes digits. To drive the common cathode side of each digit a general purpose NPN transistor is used.

The theory behind a multiplexed display is to display one digit of the necessary four(4) every VBL (vertical blank interrupt). The digit will stay on until the next VBL. At this time it will be turned off and the next digit will be displayed. This continues until all four digits have been displayed, then the whole procedure starts over again.

EXPLANATION

HARDWARE

The hardware for the MICROPORT XL CLOCK consists primarily of a four(4) digit common cathode clock display. The simplest method of building this display is to buy a multiplexed four(4) digit display. If one cannot be found at a reasonable price, there are two alternatives.

The easiest method is to make the display out of four(4) separate digits by wiring all the identical segments together, ie. wire segment A of digit 1 to segment A of digit 2 to segment A of digit 3 to segment A of digit 4. This line would then be connected to the 270 ohm resistor as shown in the schematic. This would continue for all seven(7) segments. The common cathode lead of each digit would be connected as per the schematic. This method will entail more wiring but may be

cheaper due the the low cost of surplus seven segment displays.

The second method requires some careful disassembly of a common four(4) digit non-multiplexed clock display. This method is the one I used due to the fact that I already had the display in my junk box. These displays have been seen at Radio Shack for a couple of bucks. To make this display work all the segments must he wired as in the above example. The hard part is to open up the display by removing the red plastic cover and finding the common lead that supplies a return for ALL the digits. This line must be disconnected from each digit. I accomplished this by making a small cut with an Xacto knife in the circuit trace in between all the digits. This freed up each digit to be driven independently from each of the other digits. This line must also he cut from the colon. A small scrape is then made on the trace for each digit and a wire soldered there. This line is then used to drive each digit and wired as per the schematic. This method is NOT recommended for those not handy with an xacto knife and not steady of hand. A mistake could ruin a segment of the display. With the cover off there is no protection for the LED's. If you slip you may touch one of these LED's and ruin it.

Once the four(4) digit display is made the rest of the circuit can be wired. This display can be expanded to six(6) digits, by adding two more displays and their appropriate transistors. They should be connected to pins 11 and 10 on the PIA. The added time to update six(6) digits will cause some additional flicker of the displays, but it should not be too annoying.

SOFTWARE

The software for MICROPORT XL CLOCK consists of a initialization routine and an interrupt routine. The initialization routine sets up the VBL sequence. The VBL routine includes a simple clock routine to update our clock variables every 60 VBL's, and a display routine to display the next digit on the LED's. This program should run with existing software (probably NDT games) as long as $D8 (216) bytes of free space can be found where the CLOCK routine can be placed. The included assembly shows it being placed at $600. This will not work with most cartridges. Note that all the variables have been kept internal and do not require the use of zero page.

The INIT routine clears all the digits to zero, sets up the PIA to all outputs, and links in our code to the VBL sequence. It then returns to whatever called it. This allows the code to run with other programs as long as they do not play with VBL and do not conflict with our program area.

CLOCK is the routine to update the time variables. It is only executed every 60 VBL's. The clock variables are kept in common binary, ie one byte for seconds, one byte for 10 seconds, one byte for minutes, etc. This is not the most space effective but it is the simplest to demonstrate. Note that the clock routine is a 24 hour clock and resets to zero at that time.

DISPLAY is the routine to display the next digit to the LED display. Note that it could be changed to display six(6) digits by changing the reset value from 2 to 0. To get the flashing colon the right most bit of the seconds counter is shifted into carry then into bit 7 of the accumulator. The rest of the accumulator is then masked off and the data for the digit to be displayed is OR'ed into the accumulator. The accumulator is then put on port A to turn on the correct segments. This gives a flash of the colon once a second. The routine ends with a jump to the normal VBL routine.

DIGTABLE is a table to convert binary numbers to seven segment data. The data is inverted to work correctly with the 74LS240 inverter, ie. a 1 in a bit position means do not display that segment and a 0 means to display that segment.

BITTABLE is a table to convert a digit number to a bit position to correspond to the digit to turn on.

CONSTRUCTION HINTS AND HELPS

Construction of MICROPORT XL CLOCK is straight forward and simple. It is recommended that a socket be used for the 74LS240 and a 14 or 16 pin socket for the four(4) transistors. If separate seven segment displays are used they should also be socketed. The transistors are common general purpose NPN types and substitutions can be used.

INSTRUCTION FOR USE

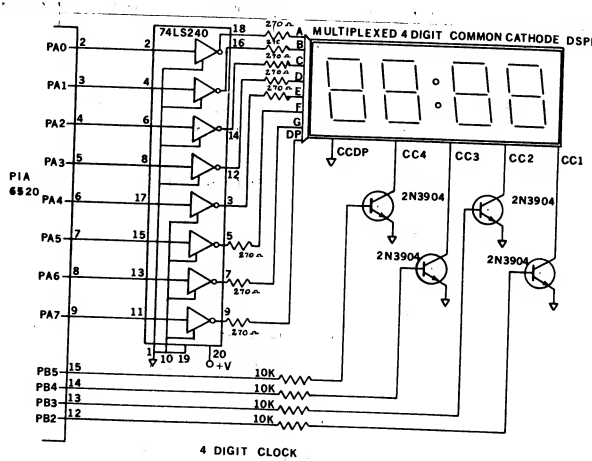
Plug the MICROPORT XL board with the clock circuit into the Atari XL computer. Load the assembled program and run it at the INIT address. The clock should display 00:00. After one(1) minute it should change to 00:01 and continue to update every minute. The slight flicker is due to the fact that each digit is turned on for only 1/4 of the time. The clock should remain running until the machine is turned off or until there is a conflict with its memory space.

If it fails to work check all wiring. If unrecognizable characters are displayed there are probably crossed wires in the segment drive lines, or the DIGTABLE is incorrect.

The display can be expanded to six(6) digits to display seconds. It could be used to display other information other than the time by changing the data to be displayed. The DIGTABLE could be expanded to allow some alpha characters to be displayed. The other possible applications are endless.

PARTS LIST

1. ( ) 1 ea. 74LS240 tri-state inverter
2. ( ) 1 ea. 4 digit multiplexed display (see text)
3. ( ) 4 ea. 2N2222 general purpose NPN transistors
4. ( ) 8 ea. 270 ohm 1/4 watt resistors
5. ( ) 4 ea. 10K ohm 1/4 watt resistors
6. ( ) 1 ea. 20 pin socket



4 DIGIT CLOCK

```
                    10 ; SAVE @D2:CLOCK.MAC
                    20 ;
                    30 ;
                    40 ; ASM ,,@D2:CLOCK.COM
                    50 ;
                    60 ;     .OPT NO LIST
                    70 ;
                    80 ;
                    90 ; Define Equates
                   100 ;
      =003C         110 ONESECOND = 60  ;VBL's per second
                   120 ;
                   130 ;
                   140 ; MICROPORT XL PIA equates
                   150 ;
      =D700        160 MPORTA = $D700  ;Segment port
      =D701        170 MPORTB = $D701  ;Digit port
      =D702        180 MACTL = $D702   ;Porta control
      =D703        190 MBCTL = $D703   ;Portb control
                   200 ;
                   210 ;
                   220 ; Atari system equates
                   230 ;
      =E45C        240 SETVBV = $E45C  ;Set new VBL
      =E45F        250 SYSVBV = $E45F  ;VBL entry
                   260 ;
                   270 ;
                   280 ;
0000               290     *=  $0600
                   300 ;
                   310 ;
                   320 ; Routine to setup MICROPORT XL,
                   330 ;   VBL interrupt, and clear
                   340 ;   time variables
                   350 ;
0600               360 INIT
0600 A900          370     LDA #$00    ;Configure the
0602 8D02D7        380     STA MACTL   ;..direction
0605 8D03D7        390     STA MBCTL   ;..registers
0608 8DD706        400     STA HOUR    ;Clear time
060B 8DDB06        410     STA HOUR+$01
060E 8DD506        420     STA MINUTE
0611 8DD606        430     STA MINUTE+$01
0614 8DD306        440     STA SECOND
0617 8DD406        450     STA SECOND+$01
061A A9FF          460     LDA #$FF    ;Make all lines
061C 8D00D7        470     STA MPORTA  ;..to outputs
061F 8D01D7        480     STA MPORTB
0622 A904          490     LDA #$04    ;Turn ports back
0624 8D02D7        500     STA MACTL   ;..to data with
0627 8D03D7        510     STA MBCTL   ;..no IRQ's
062A 8DD106        520     STA COUNTER ;Just non-zero
062D 8D9B06        530     STA DISPLAY ;Just non-zero
                   540 ;
0630 A03A          550     LDY # <CLOCK ;Setup VBL
0632 A206          560     LDX # >CLOCK
0634 A906          570     LDA #$06
```

MICROPORT-XL
J1
J2
C1
C2
C3
C4
U1
U2
Q1
R1
R2
R3
R4
S1-4
P1
1
2
49
50
mpp
© 1984
Microbits
Peripheral
Products, Inc.

70-40000-1

```
0636 205CE4   0580      JSR SETVBV
0639 60       0590      RTS           ;Back to system
              0600 ;
              0610 ;
              0620 ; Routine to update and show
              0630 ;   clock display to PIA card
              0640 ;
              0650 ;
063A          0660 CLOCK
063A CED106   0670      DEC COUNTER ;Time to update?
063D D05C     0680      BNE DISPLAY ;No..
              0690 ;
              0700 ; Routine to update time
              0710 ;
063F          0720 CLOCK.1
063F A200     0730      LDX #$00    ;Reset value
0641 A93C     0740      LDA #ONESECOND
0643 8DD106   0750      STA COUNTER ;Reset counter
0646 EED306   0760      INC SECOND  ;Up one second
0649 ADD306   0770      LDA SECOND
064C C90A     0780      CMP #10     ;Past 10?
064E 904B     0790      BCC DISPLAY ;NO..
0650 8ED306   0800      STX SECOND  ;Yes - reset it
              0810 ;
0653 EED406   0820      INC SECOND+$01 ;Up 10 secnds
0656 ADD406   0830      LDA SECOND+$01
0659 C906     0840      CMP #6      ;Past 6?
065B 903E     0850      BCC DISPLAY ;NO..
065D 8ED406   0860      STX SECOND+$01 ;Yes-reset it
              0870 ;
0660 EED506   0880      INC MINUTE  ;Up one minute
0663 ADD506   0890      LDA MINUTE
0666 C90A     0900      CMP #10     ;Past 10?
0668 9031     0910      BCC DISPLAY ;NO..
066A 8ED506   0920      STX MINUTE  ;Yes - reset it
              0930 ;
066D EED606   0940      INC MINUTE+$01 ;Up 10 minuts
0670 ADD606   0950      LDA MINUTE+$01
0673 C906     0960      CMP #6      ;Past 6?
0675 9024     0970      BCC DISPLAY ;NO..
0677 8ED606   0980      STX MINUTE+$01 ;Yes-reset it
              0990 ;
067A EED706   1000      INC HOUR    ;Up one hour
067D ADD706   1010      LDA HOUR
0680 C904     1020      CMP #4      ;24 hour test
0682 D00D     1030      BNE CLOCK.2 ;Nope..
0684 ADD806   1040      LDA HOUR+$01 ;Yes-test rest
0687 C902     1050      CMP #2
0689 D010     1060      BNE DISPLAY ;Nope..
068B 8ED706   1070      STX HOUR    ;Reset hour
068E 8ED806   1080      STX HOUR+$01 ;Reset 10 hour
              1090 ;
0691          1100 CLOCK.2
0691 C90A     1110      CMP #10     ;Past 1?
0693 9006     1120      BCC DISPLAY ;NO..
0695 8ED706   1130      STX HOUR    ;Yes-reset it
0698 EED806   1140      INC HOUR+$01 ;Up 10 hours
              1150 ;
```

```
                        1160 ;
                        1170 ; Routine to show one digit of
                        1180 ;   of time for one VBL avery
                        1190 ;   VBL
                        1200 ;
 069B                   1210 DISPLAY
 069B AED206            1220     LDX NEXTDIG ;Which digit
 069E E8                1230     INX         ;For next time
 069F E006              1240     CPX #6      ;At last digit
 06A1 9002              1250     BCC DISPLAY.1 ;NO..
 06A3 A202              1260     LDX ##02    ;Start at minute
                        1270 ;
 06A5                   1280 DISPLAY.1
 06A5 8ED206            1290     STX NEXTDIG ;Save till next
 06A8 AD0306            1300     LDA SECOND  ;Role bit 0 into
 06AB 6A                1310     ROR A       ;..bit 7 to
 06AC 6A                1320     ROR A       ;..flash colon
 06AD 2980              1330     AND ##80    ;Mask off rest
 06AF BCD306            1340     LDY SECOND,X ;Get digit
 06B2 19C106            1350     ORA DIGTABLE,Y ;Make 7 segment
 06B5 8D00D7            1360     STA MPORTA  ;Put in port
 06B8 BDCB06            1370     LDA BITTABLE,X ;Make binary
 06BB 8D01D7            1380     STA MPORTB  ;Put in port
                        1390 ;
 06BE 4C5FE4            1400     JMP SYSVBV  ;Continue VBL
                        1410 ;
                        1420 ;
                        1430 ; Table to convert Binary digit
                        1440 ;   into 7 segment display
                        1450 ;
 06C1                   1460 DIGTABLE
 06C1 407C1218          1470     .BYTE #40,#7C,#12,#18 ;0123
 06C5 2C09215C          1480     .BYTE #2C,#09,#21,#5C ;4567
 06C9 000C              1490     .BYTE #00,#0C ;89
                        1500 ;
                        1510 ;
                        1520 ; Table to convert position in
                        1530 ;   in table to bit position
                        1540 ;   to drive correct display
                        1550 ;
 06CB                   1560 BITTABLE
 06CB 01020408          1570     .BYTE 1,2,4,8,#10,#20
 06CF 1020
                        1580 ;
                        1590 ;
                        1600 ; Our variables
                        1610 ;
                        1620 ;
 06D1                   1630 COUNTER #= #+#01 ;Second counter
 06D2                   1640 NEXTDIG #= #+#01 ;Next digit
                        1650 ;
 06D3                   1660 SECOND #= #+#02 ;Our clock
 06D5                   1670 MINUTE #= #+#02 ;..variables
 06D7                   1680 HOUR #= #+#02
                        1690 ;
                        1700 ;
                        1710 ;
```


EPROMer

EPROMer is a project for MICROPORT XL designed to allow programming or reading of 2764 EPROM's. The hardware has been kept very simple and the software is structured in style to allow the user to expand its capabilities and usability. The current version of software is a bare bones version. It is not very user friendly, but it is well documented to allow for expansion and improvement. The hardware is designed for use with only 2764 EPROM's but could easily be adapted to allow reading and programming of other EPROM's or EEPROM's.

THEORY OF OPERATION

An EPROM is an Erasable (through the use of ultra-violet light) Programmable Read Only Memory. It can be programmed by an EPROM Programmer such as the one being described in this documentation. If the code in the EPROM needs to be changed, the EPROM need only be erased by exposure to shortwave ultra-violet light then reprogrammed. This procedure can be repeated many times before the EPROM ceases to work. EPROM's are not as easy to use or as fast as RAM, but they DO retain what you programmed into them when the power is turned off.

To program an EPROM the programmer must select the address of the location in the EPROM to be programmed, write the data desired at that location to the data lines, then toggle the PGM (program line) on the EPROM in 1 millisecond pulses for the manufactures designated number of pulses. After this has been done the programmer must select the next address to program and go through the same process. While this is happening +5 volts must be supplied to the EPROM on the normal Vcc pin, and +21 volts must be supplied on the Vpp pin. This +21 volts is what really does the programming of the EPROM.

The MICROPORT XL EPROMer uses what is known as the intelligent method of programming EPROM's. This method entails zapping the EPROM with 1 millisecond pulses until the data read back from the EPROM is the same as that written to the EPROM. Four(4) times that many pulses are then given to the EPROM to ensure the data will hold before moving on to the next memory location.

OESCRIPTION

HAROWARE

The hardware for the MICROPORT XL EPROMer is very simple to build. All parts are readily available at local electronics parts stores and substitutions can be made for almost all the parts.

The circuit uses the natural latching characteristics of the 6520/6821 PIA ports to simplify the design. This ability supplies 16 of the needed 21 latched lines. To get the other needed latched lines a 74LS373 OCTAL LATCH is used. Port B will be used to access the 74LS373 latch and to read or program the data to the EPROM. This gives port B double duties. The 74LS373 latch will be used to hold the lower 8 address lines of the EPROM. Port A will be used to access the remainder 5 upper address lines, control the latch state of the 74LS373 latch, and control the PGM (program) and OE the (output enable) lines. Only two more control lines are needed to switch the two power supply voltages. The CA2 and CB2 lines will be used to switch the +21 volts program line and the +5 volt power line respectively.

The +21 volts programming voltage is derived from four(4) 9 volt transistor batteries in series and regulated by a 21 volt zener diode. To arrive at the 21 volt zener diode two 5.6 volt and two 5.1 volt zeners are connected in series. Both the +5 volt and +21 volt switchers are identical. The control signal from the PIA is used to switch a general purpose NPN transistor on. This transistor then enables a PNP pass transistor to supply the necessary voltages to the EPROM. Both of the voltages need the option of being turned off to alleviate the possibility of burning up the EPROM when it is inserted into the ZIF (zero insertion force) socket. The LED's show the status of the power voltages. When ever either or both LED's are on, the socket has power supplied to it and the EPROM should not be inserted or removed.

SOFTWARE

The software for the MICROPORT XL EPROMer is a very bare bones implementation of an EPROM programmer. This description and the included source code documentation is intended to give the user an idea of one possible way to use the hardware to program EPROM's. This software is in no way intended to be user friendly or thorough. It was primarily written to instruct and teach.

The program is divided into many subroutines to allow for easy expansion and improvement. To use this current version of software it must be called and used from a machine language monitor or debugger. There are five(5) bytes starting at location $80 used to specify the start of the RAM area that holds the data to be written to the EPROM, the length of code to write out, and the mode in which the software is to run (read or program).

START (locations $80 and $81) points to the start of the RAM area to copy to the EPROM or the area to copy the EPROM to. LENGTH (locations $82 and $83) is the length of the data block to be programmed into or read from the EPROM. This length must be less then the defined PROMSIZE variable, currently set to $2000 (8k). The last byte in this area is MODE (location $84). This byte controls whether the software will read the EPROM or program it. A '0 value causes the EPROM to be programmed and a non-zero value caused it to be read into memory.

The main part of the program first calls the INIT subroutine to configure the ports, turn on the +5 volts, and disable the chip. It then checks the mode and branches to either the WRITELOOP or the REAOLOOP. The WRITELOOP turns on the +21 and calls PROGRAMIT until all the code is written to the EPROM or there is a writing error. The REAOLOOP just reads the EPROM into memory. Both routines finish by calling CLEAR to return all chip lines low (0 volts) and to turn power back off.

The border color is used to signal the status of the programmer. While the socket has power applied and data is being read or written to the EPROM the border will be green to signify that things are going well. If an error occurs the border will turn to RED. If the reading or programming finishes without an error the screen border will be yellow.

The rest of the subroutines will be described below:

INIT - Configure port A to output, turn the +5 volts on and the +21 volts off and disable the OE (output enable) and PGM (program) lines.

CLEAR- Make all ports into outputs then make all the output lines low (0 volts). Turn both power supplies off and clear the latch outputs to low (0 volts). This makes sure all the lines are low to keep from burning up EPROM's when they are inserted or removed from the socket.

SETCOLOR- This sets the border color to the value passed in the accumulator.

SETINPUT- Makes port B into an input to allow for reading of the data from the EPROM.

SETOUTPUT- Sets port B to output to allow for setting up of the latch and for writing data to the data lines of the EPROM before each program zap.

SETAOORESS- Takes the address in the POINTER variable and puts this on the address lines of the EPROM. Automatically sets up the latch.

CHECKENO- First increments POINTER, then checks to see if the end of the current code block has been reached. Is POINTER minus START equal to LENGTH?

READIT- Sets up the EPROM to the address pointed to by POINTER then reads into the X register the data at that location.

WRITEBYTE- Writes the data in the Accumulator to the EPROM at the address pointed to by POINTER.

DELAYMILI- Delays for one(1) millisecond. Used for timing the length of each zap pulse.

HITPROM- Zap the EPROM the number of times passed in the Y register. Each pulse is only one(1) millisecond long and all interrupts are disabled for the length of the pulse to ensure nothing will lengthen the pulse.

PROGRAMIT- Routine to program the EPROM until the data read back is the same as that written and then multiply that number by four(4) and zap it that many more times, or else error out if the EPROM did not program. The equal flag will be set in the status register if things are OK.

CONSTRUCTION HINTS

All the parts used in the MICROPORT XL EPROMer are non-critical except for the zener diodes. These zener diodes must equal 21 volts plus or minus .5 volts. If they total too much, the EPROM will be damaged, and if they are too low the EPROM will never get programmed. The NPN transistors (2N390A) are general purpose types, and the PNP transistors (ECG189) are general purpose medium power types. Most reasonable substitutions should work.

Layout is non-critical, but a ZIF (Zero Insertion Force) socket is highly recommended for the EPROM. It will greatly decrease the chance of bent pins.

CHECKOUT

Double check all wiring before proceeding to power up your computer and MICROPORT XL EPROMer. When first powered up, usually one of the LED's will be lit. This signifies that power is being applied to the EPROM socket and NO EPROM should be inserted or removed.

To test the power switchers insert a debugger cartridge or load one into your computer. Deposit a $34 into $0702 and $0703. This should turn any LED's off. Use a volt meter to check that no voltage is present at pins 1 and 28 of the EPROM socket. Next put a $3C into $0703. This should turn on the +5 volt LED. Check pin 28 for +5 volts with a volt meter, also check pin 1 for about +4.5 volts. Then put a $3C into $0702. This should turn on the +21 volt LED, and raise the voltage on only pin 1 to +21 volts. Put a $34 back into both $0702 and $0703. This should

extinguish both LED's and return the voltages on pins 1 and 28 back to 0 volts. If the above procedure checks out OK you can continue. But if it failed, recheck all wiring and try again. If the above procedure fails the EPROM programmer will NOT function and damage can be inflicted on the EPROM.

USING MICROPORT XL EPROMer

To use the MICROPORT XL EPROMer you will need the following:

Wired MICROPORT XL EPROMer
MICROPORT XL EPROMer software
Erased 2764 EPROM
6502 Assembler
Debugger or monitor

Enter in the MICROPORT XL EPROMer software with your favorite assembler and assemble it. The programmer software can be located anywhere in memory, but remember to put it where the code to be written into the EPROM will not conflict with it.

Load in your favorite debugger or monitor and run it. Load in the assembled EPROMer software. Load in the section of code to be written to the EPROM. Set locations $80 through $84 with the necessary information. For example lets suppose your code is from $4000 to $5FFF and you want to write all 8K out to the EPROM. Put a $00 in $80 and a $40 in $81 - this sets the beginning of the code to be written. Put a $00 in $82 and a $20 in $83 - this teils the programmer to write out 8K. Put a $00 in $84 - this signals that you want to program the EPROM.

If both LED's are not off then execute the programmer code without an EPROM in the socket. This will cause an error, but will clear the socket so that the EPROM can be safely inserted. Insert a 2764 EPROM in the socket. Remember to orientate it correctly with pin 1 in the correct place. Execute the programmer and in approximately 1 - 2 minutes you will have a programmed EPROM.

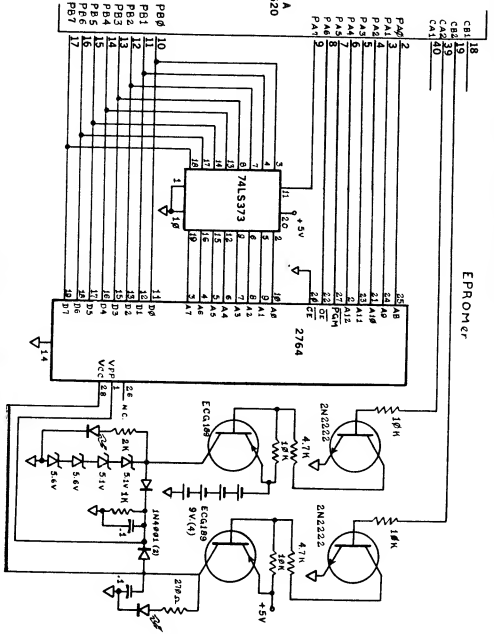
If you receive an error condition there could be any number of reasons why. (1) The EPROM is bad or not erased. (2) There is a wiring error on your MICROPORT XL EPROMer board. (3) There is an error in your assembly code. (4) You specified more than 8K of code to program into the EPROM. The next byte over 8K will go back into the first position of the EPROM causing an error, since that location is already programmed.

To read an EPROM set up bytes $80 through $83 as above, but put a non-zero ($FF) into $84. Insert the EPROM to be read into the socket and execute the code. The code will be copied into memory starting at the specified location.

There are many improvements that could he made to this project. The included information is intended to acquaint the user with how to use MICROPORT XL and give background information to facilitate building projects.

PARTS LIST

1. ( ) 1 ea. 74LS373 latch
2. ( ) 2 ea. 2N3904 general purpose NPN transistor
3. ( ) 2 ea. ECG 189 medium power general purpose PNP transistor
4. ( ) 2 ea. LED general purpose
5. ( ) 2 ea. 1N4002 general purpose rectifier
6. ( ) 1 ea. 21v zener (2 - 5.1 volts and 2 - 5.6 volts)
7. ( ) 1 ea. 270 ohm 1/4 watt resistor
8. ( ) 1 ea. 1K ohm 1/4 watt resistor
9. ( ) 1 ea. 2K ohm 1/4 watt resistor
10. ( ) 2 ea. 4.7K ohm 1/4 watt resistor
11. ( ) 4 ea. 10K ohm 1/4 watt resistor
12. ( ) 2 ea. .1 uF 50v disc capacitors
13. ( ) 1 ea. 20 pin socket
14. ( ) 1 ea. 28 pin ZIF socket (Zero Insertion Force)
15. ( ) 4 ea. 9v transistor batteries




```
                        10 ; SAVE #D2:EPROMER.MAC
                        20 ;
                        30 ;
                        40 ; ASM ,,#D2:EPROMER.COM
                        50 ;
                        60 ;     .OPT NO LIST
                        70 ;
                        80 ;
                        90 ; Define Equates
                        0100 ;
=003B                   0110 MCTLSET = $3B      ;Set DDR registr
=003C                   0120 MCTLRUN = $3C      ;Use ports
=003C                   0130 MPWRON = $3C       ;Turn power on
=0034                   0140 MPWROFF = $34      ;Power off
=007F                   0150 LATCHON = $7F      ;Latch on
=0080                   0160 LATCHOFF = $80     ;Dump latch data
=00BF                   0170 OUTENABLE = $BF    ;Enable output
=0040                   0180 OUTDISABLE = $40   ;Disable output
=00DF                   0190 PGMON = $DF        ;PGM line on
=0020                   0200 PGMOFF = $20       ;PGM line off
=000F                   0210 MAXTRY = 15        ;Max zap times
=2000                   0220 PROMSIZE = $2000   ;8K EPROM
=0030                   0230 RED =    $30       ;ERROR color
=00E0                   0240 GREEN = $E0        ;OK color
=0020                   0250 YELLOW = $20       ;OFF color
                        0260 ;
                        0270 ;
                        0280 ;
                        0290 ; MICROPORT XL PIA equates
                        0300 ;
=D700                   0310 MPORTA = $D700     ;High & control
=D701.                  0320 MPORTB = $D701     ;Low & data
=D702                   0330 MACTL = $D702      ;Porta control
=D703                   0340 MBCTL = $D703      ;Portb control
                        0350 ;
                        0360 ;
                        0370 ; Atari system equates
                        0380 ;
=02C8                   0390 COLOR4 = $02C8     ;Border color
=D40F                   0400 NMIEN = $D40F      ;NMI control
                        0410 ;
                        0420 ;
                        0430 ; Variables
                        0440 ;
0000                    0450       *=  $80
                        0460 ;
0080                    0470 START *= *+2       ;Start Ram locat
0082                    0480 LENGTH *= *+2      ;Length to copy
0084                    0490 MODE *= *+1        ;0=pgm  $FF=read
0085                    0500 POINTER *= *+2     ;Our pointer
0087                    0510 DATABYTE *= *+1    ;Byte value
0088                    0520 COUNTER *= *+1     ;Zap counter
                        0530 ;
                        0540 ;
                        0550 ;
0089                    0560       *=  $6000
                        0570 ;
```

```
                    0580 ;
                    0590 ; Main routine to program a
                    0600 ;   2764 Eprom.  Code to write
                    0610 ;   must pointed to by:
                    0620 ;   START  = $80/$81
                    0630 ;   LENGTH = $82/$83
                    0640 ;
6000                0650 MAIN
6000 38             0660      SEC          ;Check Length
6001 A582           0670      LDA LENGTH
6003 E900           0680      SBC # <PROMSIZE
6005 A583           0690      LDA LENGTH+1
6007 E920           0700      SBC # >PROMSIZE
6009 902C           0710      BCC ERROR    ;To much code
                    0720 ;
6008 A9E0           0730      LDA #GREEN   ;OK color
600D 20BF60         0740      JSR SETCOLOR
                    0750 ;
6010 A580           0760      LDA START
6012 8585           0770      STA POINTER
6014 A581           0780      LDA START+1
6016 8586           0790      STA POINTER+1
6018 205260         0800      JSR INIT
                    0810 ;
6018 A584           0820      LDA MODE     ;Read or write
601D D01D           0830      BNE READLOOP
                    0840 ;
601F A93C           0850      LDA #MPWRON ;Turn on +21v
6021 8D02D7         0860      STA MACTL
6024                0870 WRITELOOP
6024 A000           0880      LDY #$00     ;Get next byte
6026 B185           0890      LDA (POINTER),Y
6028 203961         0900      JSR PROGRAMIT
602B D00A           0910      BNE ERROR
                    0920 ;
602D 20DA60         0930      JSR CHECKEND
6030 D0F2           0940      BNE WRITELOOP
                    0950 ;
6032 A920           0960      LDA #YELLOW ;OK color
6034 4C4B60         0970      JMP DONE
                    0980 ;
                    0990 ;
6037                1000 ERROR
6037 A930           1010      LDA #RED     ;ERROR color
6039 4C4B60         1020      JMP DONE
                    1030 ;
                    1040 ;
                    1050 ;
603C                1060 READLOOP
603C 20F160         1070      JSR READIT
603F 8A             1080      TXA
6040 A000           1090      LDY #$00     ;Get next byte
6042 9185           1100      STA (POINTER),Y
                    1110 ;
6044 20DA60         1120      JSR CHECKEND
6047 D0F3           1130      BNE READLOOP
                    1140 ;
6049 A920           1150      LDA #YELLOW ;OK color
```



```
                    1160 DONE
6048 20BF60         1170      JSR SETCOLOR
604E 206C60         1180      JSR CLEAR    ;Clear socket
6051 60             1190      RTS
                    1200 ;
                    1210 ;
                    1220 ;
                    1230 ;
                    1240 ;
                    1250 ;
                    1260 ; Subroutines:
                    1270 ;
                    1280 ; Setup control lines
                    1290 ;
6052                1300 INIT
6052 A93B           1310      LDA #MCTLSET ;Configure port
6054 8D02D7         1320      STA MACTL
6057 A9FF           1330      LDA #$FF     ;All outputs
6059 8D00D7         1340      STA MPORTA
605C A934           1350      LDA #MPWROFF ;Back to data
605E 8D02D7         1360      STA MACTL    ;..& 21v off
6061 A93C           1370      LDA #MPWRON ;Turn +5v on
6063 8D03D7         1380      STA MBCTL
6066 A960           1390      LDA #(OUTDISABLE+PGMOFF)
6068 8D00D7         1400      STA MPORTA
606B 60             1410      RTS
                    1420 ;
                    1430 ;
                    1440 ; Clear EPROM socket
                    1450 ;
606C                1460 CLEAR
606C A93B           1470      LDA #MCTLSET ;Configure port
606E 8D02D7         1480      STA MACTL
6071 8D03D7         1490      STA MBCTL
6074 A9FF           1500      LDA #$FF     ;All outputs
6076 8D00D7         1510      STA MPORTA
6079 8D01D7         1520      STA MPORTB
607C A934           1530      LDA #MPWROFF ;Power off
607E 8D02D7         1540      STA MACTL
6081 8D03D7         1550      STA MBCTL
6084 A980           1560      LDA #LATCHOFF ;Disable latch
6086 8D00D7         1570      STA MPORTA
6089 A900           1580      LDA #$00     ;All lines low
608B 8D01D7         1590      STA MPORTB
60BE 60             1600      RTS
                    1610 ;
                    1620 ;
                    1630 ; Put new color into border
                    1640 ;
608F                1650 SETCOLOR
608F AA             1660      TAX          ;Save new
6090 ADCB02         1670      LDA COLOR4   ;Get old color
6093 290F           1680      AND #$0F     ;Mask off border
6095 8DCB02         1690      STA COLOR4   ;Save luminace
6098 8A             1700      TXA          ;Get back new
6099 0DCB02         1710      ORA COLOR4   ;Add in new colr
609C 8DCB02         1720      STA COLOR4   ;Save it
609F 60             1730      RTS
```

```
                    1740 ;
                    1750 ;
                    1760 ; Set data port to input
                    1770 ;
60A0                1780 SETINPUT
60A0 A938           1790      LDA #MCTLSET ;Configure port
60A2 8D03D7         1800      STA MBCTL
60A5 A900           1810      LDA #$00     ;Make all inputs
60A7 8D01D7         1820      STA MPORTB
60AA A93C           1830      LDA #MCTLRUN ;Back to data
60AC 8D03D7         1840      STA MBCTL
60AF 60             1850      RTS
                    1860 ;
                    1870 ;
                    1880 ; Set data port to output
                    1890 ;
60B0                1900 SETOUTPUT
60B0 A938           1910      LDA #MCTLSET ;Configure port
60B2 8D03D7         1920      STA MBCTL
60B5 A9FF           1930      LDA #$FF     ;Set all outputs
60B7 8D01D7         1940      STA MPORTB
60BA A93C           1950      LDA #MCTLRUN ;Back to data
60BC 8D03D7         1960      STA MBCTL
60BF 60             1970      RTS
                    1980 ;
                    1990 ;
                    2000 ; Set address to EPROM socket
                    2010 ;
60C0                2020 SETADDRESS
60C0 20B060         2030      JSR SETOUTPUT ;Portb output
60C3 AD00D7         2040      LDA MPORTA
60C6 09B0           2050      ORA #LATCHOFF ;Clear latch
60C8 8D00D7         2060      STA MPORTA
60CB A6B5           2070      LDX POINTER ;Put in low
60CD 8E01D7         2080      STX MPORTB
60D0 29E0           2090      AND #$E0     ;Mask off addres
60D2 0586           2100      ORA POINTER+1 ;Add in high
60D4 297F           2110      AND #LATCHON ;Latch portb
60D6 8D00D7         2120      STA MPORTA
60D9 60             2130      RTS
                    2140 ;
                    2150 ;
                    2160 ; Check for end of code
                    2170 ;
60DA                2180 CHECKEND
60DA E685           2190      INC POINTER ;Up pointer
60DC D002           2200      BNE CHECKEND.1
60DE E686           2210      INC POINTER+1
60E0                2220 CHECKEND.1
60E0 38             2230      SEC          ;Check for end
60E1 A586           2240      LDA POINTER+1
60E3 E581           2250      SBC START+1
60E5 C583           2260      CMP LENGTH+1
60E7 D007           2270      BNE CHECKEND.9 ;Not yet
60E9 38             2280      SEC
60EA A585           2290      LDA POINTER
60EC E580           2300      SBC START
60EE C582           2310      CMP LENGTH
```



```
60F0 60             2320 CHECKEND.9
                    2330      RTS
                    2340 ;
                    2350 ;
                    2360 ;
                    2370 ; Read byte from EPROM
                    2380 ;   Byte returned in 'X register
                    2390 ;
60F1                2400 READIT
60F1 20C060         2410      JSR SETADDRESS ;Set address
60F4                2420 READBYTE
60F4 20A060         2430      JSR SETINPUT ;Portb input
60F7 AD00D7         2440      LDA MPORTA
60FA 29BF           2450      AND #OUTENABLE
60FC 8D00D7         2460      STA MPORTA
60FF AE01D7         2470      LDX MPORTB  ;Get byte
6102 0940           2480      ORA #OUTDISABLE
6104 8D00D7         2490      STA MPORTA
6107 60             2500      RTS
                    2510 ;
                    2520 ;
                    2530 ; Write byte to EPROM
                    2540 ;   Byte must be in 'A register
                    2550 ;
6108                2560 WRITEBYTE
6108 85B7           2570      STA DATABYTE ;Save it
610A 20C060         2580      JSR SETADDRESS ;Set address
610D 20B060         2590      JSR SETOUTPUT ;Portb output
6110 60             2600      RTS
                    2610 ;
                    2620 ;
                    2630 ; Delay one millisecond for zap
                    2640 ;
6111                2650 DELAYMIL1
6111 A2B4           2660      LDX #180    ;Wait one mili
6113                2670 DELAYMIL1.1
6113 EA             2680      NOP
6114 EA             2690      NOP
6115 CA             2700      DEX
6116 D0FB           2710      BNE DELAYMIL1.1
6118 60             2720      RTS
                    2730 ;
                    2740 ;
                    2750 ; Zap EPROM 'Y times 1 millisec
                    2760 ;
6119                2770 HITPROM
6119 78             2780      SEI          ;NO INTERUPTS
611A A920           2790      LDA #$20
611C 8D0FD4         2800      STA NMIEN
611F AD00D7         2810      LDA MPORTA  ;PGM line on
6122 29DF           2820      AND #PGMON
6124 8D00D7         2830      STA MPORTA
                    2840 ;
6127                2850 HITPROMLOOP
6127 201161         2860      JSR DELAYMIL1 ;Wait one mili
612A 88             2870      DEY
612B D0FA           2880      BNE HITPROMLOOP
                    2890 ;
```

```
612D 0920          2900          ORA #PGMOFF ;PGM line off
612F 8D00D7        2910          STA MPORTA
6132 A9FF          2920          LDA #$FF
6134 8D0FD4        2930          STA NMIEN     ;Interrupts on
6137 58            2940          CLI
6138 60            2950          RTS
                   2960 ;
                   2970 ;
                   2980 ;  Program byte till it takes
                   2990 ;    or until timeout
                   3000 ;
6139               3010 PROGRAMIT
6139 200B61        3020          JSR WRITEBYTE ;Setup zap
613C A900          3030          LDA #0        ;Clear counter
613E 8588          3040          STA COUNTER
                   3050 ;
6140               3060 PROGRAMLOOP
6140 E688          3070          INC COUNTER
6142 A588          3080          LDA COUNTER
6144 C50F          3090          CMP MAXTRY  ;Max tries yet?
6146 B014          3100          BCS PROGRAMFINAL
                   3110 ;
6148 208060        3120          JSR SETOUTPUT
614B A587          3130          LDA DATABYTE ;Put byte there
614D 8D01D7        3140          STA MPORTB
                   3150 ;
6150 A001          3160          LDY #1        ;Hit 1 Milisec
6152 201961        3170          JSR HITPROM
                   3180 ;
6155 20F460        3190          JSR READBYTE ;..if it took
6158 E487          3200          CPX DATABYTE
615A D0E4          3210          BNE PROGRAMLOOP ;Not yet
                   3220 ;
615C               3230 PROGRAMFINAL
615C 208060        3240          JSR SETOUTPUT
615F A587          3250          LDA DATABYTE ;Put back in
6161 8D01D7        3260          STA MPORTB
                   3270 ;
6164 A588          3280          LDA COUNTER ;Hit tries
6166 0A            3290          ASL A         ;..times 4
6167 0A            3300          ASL A
6168 AB            3310          TAY
6169 201961        3320          JSR HITPROM
                   3330 ;
616C 20F460        3340          JSR READBYTE
616F E487          3350          CPX DATABYTE ;Did it take?
6171 60            3360          RTS
                   3370 ;
                   3380 ;
                   3390 ;

ASSEMBLY ERRORS: 0     26512 BYTES FREE
```


R6520 — Peripheral Interface Adapter (PIA)

DATA INPUT REGISTER (DIR)

When the microprocessor writes data into the PIA, the data which appears on the data bus during the Ø2 clock pulse is latched into the Data Input Register (DIR). The data is then transferred into one of six internal registers of the PIA after the trailing edge of the Ø2 clock. This assures that the data on the peripheral output lines will make smooth transitions from high to low (or from low to high) and the voltage will remain stable except when it is going to the opposite polarity.

CONTROL REGISTERS (CRA and CRB)

Table 1 illustrates the bit designation and functions in the two control registers. The control registers allow the microprocessor to control the operation of the Interrupt Control inputs (CA1, CA2, CB1, CB2), and Peripheral Control outputs (CA2, CB2). Bit 2 in each register controls the addressing of the Data Direction Registers (DDRA, DDRB) and the Output Registers (ORA, ORB). In addition, two bits (bit 6 and 7) in each control register indicate the status of the Interrupt Input lines (CA1, CA2, CB1, CB2). These Interrupt Status bits (IRQA1, IRQA2 or IRQB1, IRQB2) are normally interrogated by the microprocessor during the IRQ interrupt service routine to determine the source of the interrupt.

DATA DIRECTION REGISTERS (DDRA, DDRB)

The Data Direction Registers (DDRA, DDRB) allow the processor to program each line in the Peripheral I/O port to be either an input or an output. Each bit in DDRA controls the corresponding line in the Peripheral A port and each bit in DDRB controls the corresponding line in the Peripheral B port. Writing a "0" in a bit position in the Data Direction Register causes the corresponding Peripheral I/O line to act as an input. A "1" causes it to act as an output.

Bit 2 (DDRA, DDRB) in each Control Register (CRA and CRB) controls the accessing to the Data Direction Register or the Peripheral interface. If bit 2 is a "1", a Peripheral Output register (ORA, ORB) is selected, and if bit 2 is a "0," a Data Direction Register (DDRA, DDRB) is selected. The Data Direction Register Access Control bit, together with the Register Select lines (RS0, RS1) selects the various internal registers as shown in Table 2.

In order to write data into DDRA, DDRB or ORB registers, bit 2 in the proper Control Register must first be set. The desired register may then be accessed with the address determined by the address interconnect technique used.

PERIPHERAL OUTPUT REGISTERS (ORA, ORB)

The Peripheral Output Registers (ORA, ORB) store the output data from the Data Bus Buffers (DBB) which appears on the Peripheral I/O port. If a line on the Peripheral A Port is programmed as an output by the DDRA, writing a 0 into the corresponding bit in the ORA causes that line to go low (0.4 V). Writing a 1 causes the line to go high. The lines of the Peripheral B port are controlled by ORB in the same manner.

INTERRUPT STATUS CONTROL (ISCA, ISCB)

The four interrupt peripheral control lines (CA1, CA2, CB1, CB2) are controlled by the Interrupt Status Control logic (A, B). This logic interprets the contents of the corresponding Control Register and detects active transitions on the interrupt inputs.

PERIPHERAL I/O PORTS (PA0-PA7, PB0-PB7)

The Peripheral A and Peripheral B I/O ports allow the microprocessor to interface to the input lines on the peripheral device by writing data into the Peripheral Output Register. They also allow the processor to interface with the peripheral device output lines by reading the data on the Peripheral Port input lines directly onto the data bus and into the internal registers of the processor.

Each of the Peripheral I/O lines can be programmed to act as an input or an output. This is accomplished by setting a "1" in the corresponding bit in the Data Direction Register for those lines which are to act as outputs. A 0 in a bit of the Data Direction Register causes the corresponding Peripheral I/O lines to act as an input.

The buffers which drive the Peripheral A I/O lines contain passive pull-up devices. These pull-up devices are resistive in nature and therefore allow the output voltage to go to VCC for a logic 1. The switches can sink a full 3.2 mA, making these buffers capable of driving two standard TTL loads.

In the input mode, the pull-up devices are still connected to the I/O pin and still supply current to this pin. For this reason, these lines also represent two standard TTL loads in the input mode.

The Peripheral B I/O port duplicates many of the functions of the Peripheral A port. The process of programming these lines to act as an input or an output is similar to the Peripheral A port, as is the effect of reading or writing this port. However, there are several characteristics of the buffers driving these lines which affect their use in peripheral interfacing.

Table 1. Control Registers Bit Designations

	7	6	5	4	3	2	1	0
CRA	IRQA1	IRQA2	CA2 Control			DDRA Access	CA1 Control	
	7	6	5	4	3	2	1	0
CRB	IRQB1	IRQB2	CB2 Control			DDRB Access	CB1 Control	

The Peripheral B I/O port buffers are push-pull devices i.e., the pull-up devices are switched OFF in the 0 state and ON for a logic 1. Since these pull-ups are active devices, the logic 1 voltage will not go higher than +2.4V.

Another difference between the PA0-PA7 lines and the PB0 through PB7 lines is that they have three-state capability which allows them to enter a high impedance state when programmed to be used as input lines. In addition, data on these lines will be read properly, when programmed as output lines, even if the data signals fall below 2.0 volts for a high state or are above 0.8 volts for a low state. When programmed as output, each line can drive at least a two TTL load and may also be used as a source of up to 1 milliampere at 1.5 volts to directly drive the base of a transistor switch, such as a Darlington pair.

Because these outputs are designed to drive transistors directly, the output data is read directly from the Peripheral Output Register for those lines programmed to act as inputs.

The final characteristic is the high-impedance input state which is a function of the Peripheral B control buffers. When the Peripheral B I/O lines are programmed to act as inputs, the output buffer enters the high impedance state.

DATA BUS BUFFERS (DBB)

The Data Bus Buffers are 8-bit bidirectional buffers used for data exchange on the D0-D7 Data Bus, between the microprocessor and the PIA. These buffers are tri-state and are capable of driving a two TTL load when operating in an output mode) and recresent a one TTL load to the microprocessor (when operating in an input mode)

INTERFACE SIGNALS

The PIA interfaces to the R6500, R6500 ° or the R65C00 microprocessor family with a reset line, a Ø2 clock line, a read write line, two interrupt request lines, two register select lines three chip select lines and an 8-bit bidirectional data bus.

The PIA interfaces to the peripheral devices with four interrupt control lines and two 8-bit bidirectional date busses

Figure 1 on the front page) shows the pin assignments for these interface signals and Figure 3 shows the interface relationship of these signal as they pertain to the CPU and the peripheral devices

CHIP SELECT (CS0, CS1, $\overline{CS2}$)

The PIA is selected when CS0 and CS1 are high and $\overline{CS2}$ is low. These three chip select lines are normally connected to the processor address lines either directly or through external decoder circuits. When the PIA is selected, data will be transferred between the data lines and PIA registers, and or peripheral interface lines as determ ned by the R/$\overline{W}$, RS0, and RS1 lines and the contents of Control Registers A and B.

RESET SIGNAL ($\overline{RES}$)

The Reset ($\overline{RES}$) input initializes the R65C2* PIA. A low signal on the $\overline{RES}$ input causes all internal registers to be cleared

CLOCK SIGNAL (Ø2)

The Phase 2 Clock Signal (Ø2) is the system clock that triggers all data transfers between the CPU and the PIA. Ø2 is generated by the CPU and is therefore the synchronizing signal between the CPU and the PIA

READ/WRITE SIGNAL (R/$\overline{W}$)

Read Write (R/$\overline{W}$) controls the direction of data transfers between the PIA and the data lines associated with the CPU and the peripheral devices. A high on the R/$\overline{W}$ line permits the peripheral devices to transfer date to the CPU from the PIA. A low on the R/$\overline{W}$ line allows date to be transferred from the CPU to the peripheral devices from the PIA.

REGISTER SELECT (RS0, RS1)

The two Register Select lines (RS0, RS1), in conjunction with the Control Registes (CRA, CRB) Data Direction Register access bits (see Table 1, bit 2) select the various R6520 registers to be accessed by the CPU. RS0 and RS1 are normally connected to the microprocessor (CPU) address output lines. Through control of these lines, the CPU can write directly into the Control

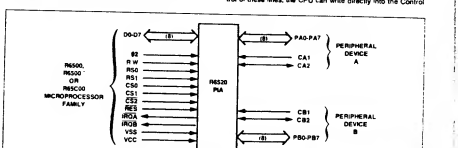


Figure 3. Interface Signals Relationship

Registers (CRA, CRB) the Data Direction Registers (DDRA, DDRB) and the Peripheral Output Registers (ORA, ORB). In addition, the processor may directly read the contents of the Control Registers and the Data Direction Registers. Accessing the Peripheral Output Register for the purpose of reading data back into the processor operates differently on the ORA and the ORB registers and therefore are shown separately in Table 2.

Table 2. ORA and ORB Register Addressing

Register Address (Hex)	Register Select Lines		Data Direction Control		Register Operation	
	RS1	RS0	CRA (Bit 2)	CRB (Bit 2)	R/$\overline{W}$=H	R/$\overline{W}$=L
0	L	L	1	—	Read PBA	Write ORA
0	L	L	0	—	Read DDRA	Write DDRA
1	L	H	—	—	Read CRA	Write CRA
2	H	L	—	1	Read PBB	Write ORB
3	H	L	—	0	Read DDRB	Write DDRB
3	H	H	—	—	Read CRB	Write CRB

INTERRUPT REQUEST LINES (IRQA, IRQB)

The active low Interrupt Request lines (IRQA and IRQB) act to interrupt the microprocessor either directly or through external interrupt priority circuitry. These lines are open drawn and are capable of sensing 1.6 milliamps from an external source. This permits all interrupt request lines to be tied together in a wired-OR configuration. The A and B in the titles of these lines correspond to the peripheral port A and the peripheral port B so that each interrupt request line services one peripheral data port.

Each Interrupt Request line has two interrupt flag bits which can cause the Interrupt Request line to go low. These flags are bits 6 and 7 in the two Control Registers (CRA, CRB). These flags act as the link between the peripheral interrupt signals and the microprocessor interrupt inputs. Each flag has a corresponding interrupt disable bit which allows the processor to enable or disable the interrupt from each of the four interrupt inputs (CA1, CA2, CB1, CB2). The four interrupt flags are set (enabled) by active transitions of the signal on the interrupt input (CA1, CA2, CB1, CB2).

CRA bit 7 (IRQA1) is always set by an active transition of the CA1 interrupt input signal. However, IRQA can be disabled by setting bit 0 in CRA to ø 0. Likewise, CRA bit 6 (IRQA2) can be set by an active transition of the CA2 interrupt input signal and IRQA can be disabled by setting bit 3 in CRA to ø 0

Both bit 6 and bit 7 in CRA are reset by ø Read Peripheral Output Register A operation. This is defined as an operation in which the read write, proper data direction register and register select signals are provided to allow the processor to read the Peripheral A I/O port. A summary of IRQA control is shown in Table 3.

Control of IRQB is performed in exactly the same manner as that described above for IRQA. Bit 7 in CRB (IRQB1) is set by an active transition on CB1 and IRQB from this flag is controlled by CRB bit 0. Likewise, bit 6 (IRQB2) in CRB is set by an active transition on CB2, and IRQB from this flag is controlled by CRB bit 3.

Also, both bit 6 and bit 7 of CRB are reset by a "Read Peripheral B Output Register" operation. A summary of IRQB control is shown in Table 3

Table 3. IRQA and IRQB Control Summary

Control Register Bits	Action
CRA-7 = 1 and CRA-6 = 1	IRQA goes low (Active)
CRA-6 = 1 and CRA-3 = 1	IRQA goes low (Active)
CRB-7 = 1 and CRB-6 = 1	IRQB goes low (Active)
CRB-6 = 1 and CRB-3 = 1	IRQB goes low (Active)

Note

The flags act as the link between the peripheral interrupt signals and the processor interrupt inputs. The interrupt disable bits allow the processor to control the interrupt function.

INTERRUPT INPUT/PERIPHERAL CONTROL LINES (CA1, CA2, CB1, CB2)

The four interrupt input peripheral control lines provide a number of special peripheral control functions. These lines greatly enhance the power of the two general purpose interface ports (PA0-PA7 PB0-PB7). Figure 4 summarizes the operation of these control lines

CA1 is an interrupt input only. An active transition of the signal on this input will set bit 7 of the Control Register A to a logic 1. The active transition can be programmed by setting a 0 in bit 1 of the CRA if the interrupt flag (bit 7 of CRA) is to be set on a negative transition of the CA1 signal or a 1 if it is to be set on a positive transition

Note:

A negative transition is defined as a transition from a high to a low, and a positive transition is defined as a transition from a low to a high voltage

CA2 can act as a totally independent interrupt or as a peripheral control output. As an input (CRA. bit 5 = 0) it acts to set the interrupt flag, bit 6 of CRA to a logic 1 on the active transition selected by bit 4 of CRA.

These control register bits end interrupt inputs serve the same basic function as that described above for CA1. The input signal sets the interrupt flag, which serves as the link between the peripheral device and the processor interrupt structure. The interrupt disable bit allows the processor to exercise control over the system interrupt.

In the output mode (CRA. bit 5 = 1), CA2 can operate independently to generate a simple pulse each time the microprocessor reads the date on the Peripheral A I/O port. This mode is selected by setting CRA. bit 4 to a 0 and CRA. bit 3 to a 1. This pulse output can be used to control the counters, shift registers, etc. which make sequential data available on the Peripheral input lines

CONTROL REGISTER A (CRA)

CA2 INPUT MODE (BIT 5 = 0)

7	6	5	4	3	2	1	0
IRQA1 FLAG	IRQA2 FLAG	CA2 INPUT MODE SELECT (= 0)	IRQA2 POSITIVE TRANSITION	IRQA ENABLE FOR IRQA2	ORA SELECT	IRQA1 POSITIVE TRANSITION	IRQA ENABLE FOR IRQA1
			IRQA IRQA2 CONTROL			IRQA IRQA1 CONTROL	

CA2 OUTPUT MODE (BIT 5 = 1)

7	6	5	4	3	2	1	0
IRQA1 FLAG	0	CA2 OUTPUT MODE SELECT (= 1)	CA2 OUTPUT CONTROL	CA2 RESTORE CONTROL	ORA SELECT	IRQA1 POSITIVE TRANSITION	IRQA ENABLE FOR IRQA1
			CA2 CONTROL			IRQA IRQA1 CONTROL	

CA2 INPUT OR OUTPUT MODE (BIT 5 = 0 or 1)

Bit 7 IRQA1 FLAG
1 A transition has occurred on CA1 that satisfies the bit 1 IRQA1 transition polarity criteria. This bit is cleared by a read of Output Register A or by RES
0 No transition has occurred on CA1 that satisfies the bit 1 IRQA1 transition polarity criteria

Bit 2 OUTPUT REGISTER A SELECT
1 Select Output Register A
0 Select Data Direction Register A

Bit 1 IRQA1 POSITIVE TRANSITION
1 Set IRQA1 Flag bit from a low-to-high transition of CA1
0 Set IRQA1 Flag bit from a negative-going low-to-high transition of CA1

Bit 0 IRQA ENABLE FOR IRQA1
1 Enable assertion of IRQA when IRQA1 Flag bit 7 is set
0 Disable assertion of IRQA when IRQA1 Flag bit 7 is set

CA2 INPUT MODE (BIT 5 = 0)

Bit 6 IRQA2 FLAG
1 A transition has occurred on CA2 that satisfies the bit 4 IRQA2 transition polarity criteria. This flag is cleared by a read of Output Register A or by RES
0 No transition has occurred on CA2 that satisfies the bit 4 IRQA2 transition polarity criteria

Bit 5 CA2 MODE SELECT
0 Select CA2 Input Mode

Bit 4 IRQA2 POSITIVE TRANSITION
1 Set IRQA2 Flag bit 6 on a positive-low-to-high transition of CA2
0 Set IRQA2 Flag bit 6 on a negative-high-to-low transition of CA2

Bit 3 IRQA ENABLE FOR IRQA2
1 Enable assertion of IRQA when IRQA2 Flag bit 6 is set
0 Disable assertion of IRQA when IRQA2 Flag bit 6 is set

CA2 OUTPUT MODE (BIT 5 = 1)

Bit 5 NOT USED
0 Always zero

Bit 5 CA2 MODE SELECT
1 Select CA2 Output Mode

Bit 4 CA2 OUTPUT CONTROL
1 CA2 goes low when a zero is written into CRA bit 3
CA2 goes high when a one is written into CRA bit 3
0 CA2 goes low on the first negative-high-to-low E2 clock transition following a read of Output Register A
CA2 returns high as specified by bit 3

Bit 3 CA2 READ STROBE RESTORE CONTROL (4 = 0)
1 CA2 returns high on the next E2 clock negative transition following a read of Output Register A
0 CA2 returns high on the next active CA1 transition following a read of Output Register A as specified by bit 1

Figure 4. Control Line Operations Summary (1 of 2)

A second output mode allows CA2 to be used in conjunction with CA1 to "handshake" between the processor and the peripheral device. On the A side, this technique allows positive control of data transfers from the peripheral device into the microprocessor. The CA1 input signals the processor that data is available by interrupting the processor. The processor reads the data and sets CA2 low. This signals the peripheral device that it can make new data available.

The final output mode can be selected by setting be 4 of CRA to a 1. In this mode, CA2 is a simple peripheral control output which can be set high or low by setting bit 3 or CRA to a 1 or a 0 respectively.

CB1 operates as an interrupt input only in the same manner as CA1. Bit 7 of CRB s set by the active transition sevected by be 0 of CRB. Likewise, the CB2 input mode operates exactly the same as the CA2 input modes. The CB2 output modes, CRB bit 5 = 1, differ somewhat from those of CA2. The pulse output occurs when the processor writes data into the Peripheral B Output Register. Also, the "handshaking" operates on data transfers from the processor into the peripheral device.

READING THE PERIPHERAL A I/O PORT

Performing a Read operation with RS1 = 0, RS0 = 0 and the Data Direct on Reg ster Access Control bit (CRA-2) = 1, directly transfers the data on the Peripheral A I/O lines to the data bus. In this situation, the data bus will contain both the input and output data. The processor must be programmed to recognize and interpret only those bits which are important to the particular peripheral operation being performed.

Since the processor always reads the Peripheral A I/O port pins instead of the actual Peripheral Output Register (ORA), it is possible for the data read by the processor to differ from the contents of the Peripheral Output Register for an output line. This is true when the I/O pin is not allowed to go to a full +2.4V DC when the Peripheral Output register contains a logic 1. In this case, the processor will read a 0 from the Peripheral A pin, even though the corresponding bit in the Peripheral Output register is a 1.

READING THE PERIPHERAL B I/O PORT

Reading the Peripheral B I/O port yields a combination of input and output data in a manner similar to the Peripheral A port. However, data is read directly from the Peripheral B Output Register (ORB) for those lines programmed to act as outputs. It is therefore possible to load down the Peripheral B Output lines without causing incorrect data to be transferred back to the processor on a Read operation.

ABSOLUTE MAXIMUM RATINGS*

Parameter	Symbol	Value	Unit
Supply Voltage	V_{CC}	−0.3 to +7.0	Vdc
Input Voltage	V_{in}	−0.3 to +V_{CC}	Vdc
Operating Temperature Range Commercial Industrial	T_A	T_L to T_{Hi} 0 to +70 −40 to +85	°C
Storage Temperature	T_{stg}	−55 to +150	°C

*NOTE: Stresses above those listed may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the other sections of this document is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

OPERATING CONDITIONS

Parameter	Symbol	Value
Supply Voltage	V_{CC}	5V ±5%
Temperature Range Commercial Industrial	T_A	0°C to 70°C −40°C to +85°C

DC CHARACTERISTICS

(V_{CC} = 5.0V ±5%, V_{SS} = 0, T_A = T_L to T_{Hi} unless otherwise noted)

Parameter	Symbol	Min.	Typ.[1]	Max.	Unit	Test Conditions
Input High Voltage	V_{IH}	+2.0	—	V_{CC}	V	
Input Low Voltage	V_{IL}	−0.3	—	+0.8	V	
Input Leakage Current IRQA, IRQB, RES, RS0, RS1, CS0, CS1, CS2, CA1, CB1, Ø2	I_{in}	—	±1	±2.5	µA	V_{in} = 0V to 5.25V V_{CC} = 0V
Output Leakage Current for Three-State Off D0-D7, PB0-PB7, CB2	I_{TSI}	—	±2	±10	µA	V_{in} = 0.4V to 2.4V V_{CC} = 5.25V
Input High Current PA0-PA7, CA2	I_{IH}	−100	−250	—	µA	V_{in} = 2.4V
Input Low Current PA0-PA7, CA2	I_{IL}	—	−1	−1.6	mA	V_{in} = 0.4V
Output High Voltage All outputs PB0-PB7, CB2 (Darlington Drive)	V_{OH}	2.4 1.5	—	—	V V	V_{CC} = 4.75V I_{load} = −100µA I_{load} = −1.0 mA
Output Low Voltage	V_{OL}	—	—	+0.4	V	V_{CC} = 4.75V I_{load} = 1.6 mA
Output High Current (Sourcing) Logic PB0-PB7, CB2 (Darlington Drive)	I_{OH}	−100 −1.0	−1000 −2.5	— −10	µA mA	V_{out} = 2.4V V_{out} = 1.5V
Output Low Current (Sinking)	I_{OL}	1.6	—	—	mA	V_{out} = 0.4V
Output Leakage Current (Off State) IRQA, IRQB	I_{OH}	—	1	±10	µA	V_{out} = 2.4V V_{CC} = 5.25V
Power Dissipation	P_D		200	500	mW	
Input Capacitance D0-D7, PA0-PA7, PB0-PB7, CA2, CB2 RW, RES, RS0, RS1, CS0, CS1, CS2, CA1, CB1, Ø2	C_{in}	— —	— —	10 7.0	pF	V_{CC} = 5.0V V_{in} = 0V f = 1.0MHz T_A = 25°C
Output Capacitance	C_{OUT}	—	—	10	pF	

Notes:
1. All units are direct current (dc) except for capacitance.
2. Negative sign indicates outward current flow, positive indicates inward flow.
3. Typical values are shown for V_{CC} = 5.0V and T_A = 25°C