Experiment
On
8255 PPI chip

Objectives: To study how 8255 PPI chip works.

After completing this experiment, you should know the different operation modes of an 8255 PPI chip and how to configure the chip to operate in a particular operation mode. You should also know how to use handshaking to transfer data in an interface.

Software: Text editor, 8051 cross-assembler, 8051 linker and 8051 programmer

Apparatus: 8051 evaluation board and 8255 evaluation board

8255 datasheet
AT89S8252 datasheet (instruction set)

Background

The 8255 PPI chip is a general purpose programmable I/O device which is designed for use with all Intel and most other microprocessors. The 8255 has 24 I/O pins divided into 3 groups of 8 pins each. The groups are denoted by port A, port B and port C respectively. Every one of the ports can be configured as either an input port or an output port.

The 8255 can be programmable in three different modes:
- Mode 0: simple unidirectional input/output without handshaking
- Mode 1: unidirectional input/output with handshaking via some pins of port C
- Mode 2: bidirectional input/output with handshaking via some pins of port C

Handshaking is a common technique used to transfer data in an interface. A computer and a device usually operate at different system clock rates and hence the data transfer between their corresponding I/O interface may not be so reliable. For example, the device might not be fast enough...
to catch the data transmitted from the CPU. Handshaking provides a means to improve the reliability of a data transfer.

**Method and details**

In this lab, you will study how to program an 8255 PPI chip to operate in different operation modes with an 8051 evaluation board and an 8255 evaluation board. Figure 1 shows the setup of the system. You are requested to modify some given 8051 program modules with a text editor in a computer. The modified programs, when they are run in the 8051 evaluation board, should be able to program port A and port B of the 8255 in the 8255 evaluation board to operate in one of their operation modes. You can assemble and link your program modules with the provided cross-assembler and linker to generate executable files. Executable files can then be loaded to the 8051 evaluation board via the printer port of the computer to program the on-board AT89S8252. The AT89S8252 is a low-power, high-performance CMOS 8-bit microcomputer with 8K bytes of Downloadable Flash programmable and erasable read only memory and 2K bytes of EEPROM. The device is manufactured by Atmel and is compatible with the industry standard 80C51 instruction set and pinout.

![Figure 1. Setup of the system](#)

After programming the AT89S8252, the AT89S8252 executes the loaded program to configure the 8255 and the ports of the 8255 should operate in the desired modes.

As there are 3 ports in 8255 and each one of them can be programmed as an input or output port, there are a number of possible configurations. In this lab, four configurations given in Table 1 will be studied.

<table>
<thead>
<tr>
<th>configuration</th>
<th>Port A</th>
<th>Port B</th>
<th>Port C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Mode 0, input</td>
<td>Mode 0, output</td>
<td>Don’t care</td>
</tr>
<tr>
<td>2</td>
<td>Mode 1, input</td>
<td>Mode 0, output</td>
<td>Handshake for port A</td>
</tr>
<tr>
<td>3</td>
<td>Mode 0, input</td>
<td>Mode 1, output</td>
<td>Handshake for port B</td>
</tr>
<tr>
<td>4</td>
<td>Mode 1, input</td>
<td>Mode 1, output</td>
<td>Handshake for ports A and B</td>
</tr>
</tbody>
</table>

Table 1. Some configurations of 8255

You are requested to do the following in this lab.

1. Setup the apparatus as shown in Figure 1.
2. Appendix D1 lists a program for configuration 1 (Ai0Bo0.asm). This program repeatedly reads port A and writes the data read to port B. Assemble, link and load the program into the 8051 evaluation board. You may refer to Appendix C for the details.

Run the program and observe the behavior of the evaluation board. You may define the input with the dip switch connected to port A and the LEDs connected to port B show the data you input.

Study the program. Pay special attention to the procedures of configuring the 8255 and the setting value written to the control register. Try to derive your own setting from datasheet or the information provided in Appendix B. Check if yours is identical to the one provided in the program.

3. Appendix D2 lists a program (Ai0Bo0X.asm) for configuration 1 as well. In this program, a 2.5s delay is added into the loop. By doing so, it simulates the case that the CPU periodically reads port A and reports the result via port B immediately. Port B is programmed to blink before it reports a result.

Load the program into the AT89S8252 evaluation board and run it. See what happens.

Since port A operates at mode 0, no handshaking is exploited. The CPU does not know when a data comes. Suppose every change of the dip switch corresponds to a data transfer from an external device. Answer the following questions.

Q1. If you change the dip switch’s setting at a rate higher than 0.4 times/s, can the CPU receive and report the dip switch setting every time you change it? In other words, can the CPU receive and report all inputs from a device if the device transfers its data at a rate higher than 2 bytes per 5 seconds?

Q2. Suppose now you change the dip switch’s setting every 4 seconds. Can the CPU know that there is no change of the setting when it tries to read a byte from port A? This corresponds to the case that a device transfers 1 byte of data every 4 seconds while the CPU regularly reads the port every 2.5s. Can the CPU know there is no available data from the device when it tries to read a byte from port A? Can it stop reading and reporting rubbish in such a case?
4. Appendix D3 lists a program for configuration 2(Ai1Bo0X.asm). In this program, as port A operates in mode 1, handshaking signal is provided through port C of 8255 and hence the CPU can make use of handshaking to synchronize itself with an external device in a data transfer. This makes the transfer much more reliable.

Load the program into the AT89S8252 evaluation board and run it. Change the setting of the dip switch and press the button marked ‘port A mode 1 input’ in the 8255 evaluation board once. This action corresponds to that an external device generates a strobe to signal the 8255 when its data is ready for transmission. What happens when you do this? Repeat the steps at different speed. Does the CPU miss receiving and reporting your inputs? Does the CPU read something even though you do not do anything?

Study the program carefully. See how the program uses handshaking to improve the performance.

5. Ai1Bo0X.asm does the job with programmed-I/O technique. It keeps checking the handshaking signal and waits until the data is ready. This keeps the CPU busy doing something without contribution. The CPU can be released by using interrupt to handle a data transfer. Appendix D4 lists an incomplete program for configuration 2(Ai1Bo0.asm). It is a better alternative to Ai1Bo0X.asm.

Complete the program by filling up the blank fields. Test your program with the evaluation boards. Study the program to see how it exploits interrupt to do the job.
6. Appendix D5 lists an incomplete program for configuration 3(Ai0Bo1.asm). Complete the program and test your program with the evaluation boards. Record your observation.

7. Based on Ai1Bo0.asm and Ai0Bo1.asm, write a program to configure 8255 to operate in configuration 4. Test your program and verify if it functions with the evaluation boards.

8. Try to configure the 8255 to function at other operation modes if time is allowed. (For more capable student)

- END -

Appendix

Appendix A. Schematic diagrams of the evaluation boards
Appendix B. Summary of the technical information of 8255
Appendix C. Editing, assembling, linking and loading programs to the 8051 evaluation board
Appendix D. Program listing
Appendix E. View of the evaluation boards
Appendix A  Schematic diagrams of the evaluation boards

AT89C52 evaluation board

CSF-LAB/PPI 8255/CC/EIE PolyU/p.6
Appendix B. Summary of the technical information of 8255

- **Internal structure:**

  ![Internal block diagram of 8255A programmable parallel port device](image)

  *(Intel Corporation)*

- **Port and register addresses:**

  - | $A_1$ | $A_0$ | Function |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Port A</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Port B</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Port C</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Command Register</td>
</tr>
</tbody>
</table>

  I/O port assignments for the 8255

- **Port connections:**

<table>
<thead>
<tr>
<th>Port A</th>
<th>Mode 0</th>
<th>Mode 1</th>
<th>Mode 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN</td>
<td>OUT</td>
<td>IN</td>
<td>OUT</td>
</tr>
<tr>
<td>0</td>
<td></td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>7</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Port B</th>
<th>Mode 0</th>
<th>Mode 1</th>
<th>Mode 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN</td>
<td>OUT</td>
<td>IN</td>
<td>OUT</td>
</tr>
<tr>
<td>0</td>
<td></td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>7</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Port C</th>
<th>Mode 0</th>
<th>Mode 1</th>
<th>Mode 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN</td>
<td>OUT</td>
<td>IN</td>
<td>OUT</td>
</tr>
<tr>
<td>0</td>
<td></td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>7</td>
<td></td>
</tr>
</tbody>
</table>

  A summary of the port connections for the 82C55 PIA
• Status word obtained by reading port C:

```
<table>
<thead>
<tr>
<th>Command byte A</th>
</tr>
</thead>
<tbody>
<tr>
<td>7 6 5 4 3 2 1 0</td>
</tr>
</tbody>
</table>

(a) Programs ports A, B, and C
```

The command byte of the command register in the 82C55.
• Operation modes:

Mode 1 operation

Mode 2 operation
Set/reset IRTEs:

<table>
<thead>
<tr>
<th>MODE 1</th>
<th>Port C Interrupt Signal Pin Number</th>
<th>To enable Interrupt Request Set Port C bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port A IN</td>
<td>PC3</td>
<td>PC4</td>
</tr>
<tr>
<td>Port B IN</td>
<td>PC0</td>
<td>PC2</td>
</tr>
<tr>
<td>Port A OUT</td>
<td>PC3</td>
<td>PC6</td>
</tr>
<tr>
<td>Port B OUT</td>
<td>PC0</td>
<td>PC2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MODE 2</th>
<th>Port A IN</th>
<th>PC3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port A OUT</td>
<td>PC3</td>
<td>PC6</td>
</tr>
</tbody>
</table>
Appendix C. Editing, assembling, linking and loading programs to the 8051 evaluation board

You may use any text editor such as Notepad in Windows to edit your 8051 program. Then you can assemble and link your program so as to make it loadable to the evaluation board for debugging.

Suppose your program is ready and is now stored in the working directory where the 8051 cross-assembler(X8051.exe) and the 8051 linker(Link.exe) are in. Run X8051.exe to activate the cross-assembler. Figure C1 shows the user interface of the cross-assembler. In the interface, the cross-assembler will prompt for inputting listing destination, input filename and output filename. You have to specify the input filename. As for others, you can skip them by just entering ‘\n’. If no error is detected by the cross-assembler, an object file with extension ‘.obj’ will be generated.

Run Link.exe to activate the linker. Figure C2 shows the user interface of the linker. The linker will prompt for inputting parameters. All you need to do is to specify the input filename. It should be an object file with extension ‘.obj’. As an example, Figure C2 shows the case that the input file is pgm8051.obj. You can skip all other prompts by just entering ‘\n’. If no error is detected, a binary file with extension ‘.hex’ will be generated.

A universal programmer called PonyProg is provided in this lab. Figure C3 shows the user interface provided by the programmer. This programmer can program a specified binary file into the flash
memory of an 8051-compatible controller via the printer port of a computer system. To order to do it successfully, you have to make sure that the device you want to program is AT89S8252. You can check (and select) via the listbox in the interface as shown Figure C3. Besides, you have to check the interface setup by selecting ‘Setup’ in the pulldown menu ‘Options’. Select the setting shown in Figure C4.

After configuring the programmer, one can load a program, namely, a file of extension ‘.hex’, into the working environment and program the AT89S8252 in the evaluation board. To load the program into the working environment, you can push the fourth pushbutton from the left in the toolbar and then select the desired file. Figure C5 shows a snapshot of the user interface after program ‘pgm8051.hex’ was loaded into the environment. Then you can push the second pushbutton from the left in the toolbar to load the program into the AT89S8252.
Figure C4. Setting for the interface between the evaluation board and the computer

Figure C5. A snapshot of the user interface after a program is loaded into the working environment
Appendix D. Program listing

D.1 listing of Ai0Bo0.asm

; Ai0Bo0.asm
; Port A -> mode 0 input
; Port B -> mode 0 output
; Input data from port A and output it to port B

pa equ 8000h ; port a
pb equ pa+1 ; port b
pc equ pa+2 ; port c
cr equ pa+3 ; control register

org 00h
ajmp main

;------------------------------------

main:
    mov sp,#60h ; set stack pointer to address 60h
    mov r5,#10 ; delay 10ms for
    call delay ; 8255 initialization
    mov a,#90h ; set port a to mode 0 input
    mov dptr,#cr ; and port b to mode 0 output
    movx @dptr,a

    jmp loop

;------------------------------------

delay:    ; delay time = r5*10ms
    djnz r5,delay
    djnz r6,$1
    djnz r7,$2
ret

;------------------------------------

end

loop:
    mov dptr,#pa ; input from port a
    movx a,@dptr
    mov dptr,#pb ; output to port b
    movx @dptr,a
    jmp loop

;------------------------------------

S1:     mov r7,#100
S2:     djnz r7,$2

D.2 listing of Ai0Bo0X.asm

; Ai0Bo0X.asm
; Port A -> mode 0 input
; Port B -> mode 0 output
; Input data from port A and output it to port B

pa equ 8000h ; prot a
pb equ pa+1 ; prot b
pc equ pa+2 ; prot c
cr equ pa+3 ; control register

org 00h
ajmp main

;------------------------------------
main:  mov  sp,#60h  ; set stack pointer to address 60h
        mov r5,#10  ; delay 100ms for
        call delay  ; 8255 initialization
        mov a,#90h  ; set port a to mode 0 input
        mov dptr,#cr  ; and port b to mode 0 output
        movx @dptr,a
        loop:    ; periodically wait 2.5s, get a data
        mov r5,#250  ; delay 2.5s
        call delay
        mov a,#0  ; clear port b for 20ms
        mov dptr,#pb ;
        movx @dptr,a
        mov r5,#2
        call delay
        mov a,#255  ; set port b for 20ms
        mov dptr,#pb ;
        movx @dptr,a
        mov r5,#2
        jmp loop

;------------------------------------
delay:    ; delay time = r5*10ms
        mov r6,#50
        $1: mov r7,#100
        $2: djnz r7,$2
        djnz r6,$1
        djnz r5,delay
        ret

;------------------------------------
end
D.3 listing of Ai1Bo0X.asm

; Ai1Bo0X.asm
; Port A -> mode 1 input
; Port B -> mode 0 output
; Input data from port A and output it to port B

pa equ 8000h ; port a
pb equ pa+1 ; port b
pc equ pa+2 ; port c
cr equ pa+3 ; control register

org 00h
ajmp main

;------------------------------------
main:  mov  sp,#60h  ; set stack pointer to address 60h
mov r5,#10  ; delay 10ms for
call delay  ; 8255 initialization
mov a,#b0h  ; set port a to mode 1 input
mov dptr,#cr  ; and port b to mode 0 output
movx @dptr,a

loop:
mov dptr,#pc  ; get status word of 8255
movx a,@dptr  ; to check if IBF(bit 5)=1
anl a,#20h  jz loop

mov a,#0  ; clear port b for 20ms
mov dptr,#pb
movx @dptr,a
mov r5,#2  ;
call delay  ;

mov a,#255  ; set port b for 20ms
mov dptr,#pb
movx @dptr,a
mov r5,#2  ;

call delay  ;
mov dptr,#pa  ; input from port a
movx a,@dptr
mov dptr,#pb  ; output to port b
movx @dptr,a
jmp loop

;------------------------------------
delay:    ; delay time = r5*10ms
mov r6,#50
$1: mov r7,#100
$2: djnz r7,$2
    djnz r6,$1
    djnz r5,delay
    ret

;------------------------------------
end
D.4 listing of Ai1Bo0.asm

; Ai1Bo0.asm
; Port A -> mode 1 input
; Port B -> mode 0 output
; Input data from port A and output it to port B

pa equ 8000h ; port a
pb equ pa+1 ; port b
pc equ pa+2 ; port c
cr equ pa+3 ; control register

org 00h
ajmp main

org 13h
ajmp int1

;------------------------------------

main:
    mov sp,#60h ; set stack pointer to address 60h
    setb it1 ; set int1 to negative edge trigger
    setb ea ; enable hardware interrupt
    setb ex1 ; enable int1
    mov r5,#10 ; delay 10ms for
    call delay ; 8255 initialization
    mov a,_____ ; set port a to mode 1 input
    mov dptr,#cr ; and port b to mode 0 output
    movx @dptr,a
    mov a,_____ ; enable interrupt request
    mov dptr,#cr ; for port a
    movx @dptr,a

loop:
    mov dptr,#pb ; output to port b
    movx @dptr,a
    jmp loop

;------------------------------------

int1:
    mov r5,#200 ; delay 200ms to make
call delay ; IBF visible
    mov dptr,#pa ; input from port a
    movx a,@dptr
    reti

;------------------------------------

delay: ; delay time = r5*10ms
    mov r6,#50
$1:    mov r7,#100
$2:    djnz r7,$2
        djnz r6,$1
        djnz r5,delay
        ret

;------------------------------------

end
D.5 listing of Ai0Bo1.asm

; Ai0Bo1.asm
; Port A -> mode 0 input
; Port B -> mode 1 output
; Input data from port A and output it to prot B

pa equ 8000h ; prot a
pb equ pa+1 ; prot b
pc equ pa+2 ; prot c
cr equ pa+3 ; control register

org 00h
ajmp main
org 03h
ajmp int0

-------------------------------------------------------------------------------------
main:
    mov sp,#60h ; set stack pointer to address 60h
    setb it0 ; set int0 to negative edge trigger
    setb ea ; enable hardware interrupt
    setb ex0 ; enable int0
    mov r5,#10 ; delay 10ms for
    call delay ; 8255 initialization
    mov a,_____ ; set port a to mode 0 input
    mov dptr,#cr ; and port b to mode 1 output
    movx @dptr,a
    mov a,_____ ; enable interrupt request
    mov dptr,#cr ; for port b
    movx @dptr,a

loop:
    mov dptr,#pa ; input from port a
    movx a,@dptr
    jmp loop

-------------------------------------------------------------------------------------
int0:

mov dptr,#pb ; output to port b
movx @dptr,a

reti

-------------------------------------------------------------------------------------
delay:
    mov r6,#50
S1:     mov r7,#100
S2:     djnz r7,S2
        djnz r6,S1
        djnz r5,delay
    ret

-------------------------------------------------------------------------------------
end
D.6 Listing of Ai1Bo1.asm

; Ai1Bo1.asm
; Port A -> mode 1 input
; Port B -> mode 1 output
; Input data from port A and output it to prot B

pa equ 8000h ; prot a
pb equ pa+1 ; prot b
pc equ pa+2 ; prot c
cr equ pa+3 ; control register

org 00h
ajmp main

org 03h
ajmp int0
org 13h
ajmp int1

main:
    mov sp,#60h ; set stack pointer to address 60h
    setb it0 ; set int0 to negative edge trigger
    setb it1 ; set int1 to negative edge trigger
    setb ea ; enable hardware interrupt
    setb ex0 ; enable int0
    setb ex1 ; enable int1

    mov r5,#10 ; delay 10ms for
    call delay ; 8255 initialization

    mov a,_______ ; set port a to mode 1 input
    mov dptr,_____ ; and port b to mode 1 output
    movx @dptr,a

    mov a,_______ ; enable interrupt request
    mov dptr,_____ ; for port b
    movx @dptr,a

    mov a,_______ ; enable interrupt request
    mov dptr,_____ ; for port a
    movx @dptr,a

loop:
    jmp loop

;------------------------------------

int0:
    mov dptr,_____ ; output to port b
    movx @dptr,a

    reti

;------------------------------------

int1:
    mov dptr,_____ ; input from port a
    movx a,@dptr

    reti

;------------------------------------

delay:
    mov r6,#50
    $1: mov r7,#100
    $2: djnz r7,$2
    djnz r6,$1
    djnz r5,delay
    ret

;------------------------------------

end

CSF-LAB/PPI 8255/CC/EIE PolyU/p.20
Appendix E. Views of the evaluation boards

Program listing

Figure A.E-1  8051 evaluation board

Figure A.E-2  8255 evaluation board