Hong Kong Polytechnic University
Department of Electronic and Information Engineering

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 handshake 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

Reference : H-P. Messmer, "The indispensable PC hardware book," 3rd Ed, Addison-Wesley, 1997 Chapter 29 Section 2.
Barry B. Bery, “The Intel Microprocessors: 8086/8088, 80186/80188, 80286, 80386, 80486, Pentium, Pentium Pro Processor, Pentium II, Pentium III, Pentium 4 - Architecture, Programming, and Interfacing”, 6th Ed, Chapter 11, Section 3.
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 handshake
- Mode 1: unidirectional input/output with handshake via some pins of port C
- Mode 2: bidirectional input/output with handshake via some pins of port C

Handshake 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. Handshake 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.

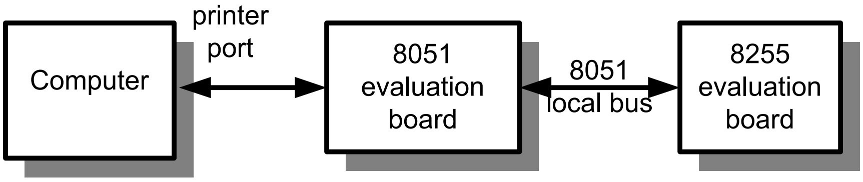


Figure 1. Setup of the system

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.

configuration	Port A	Port B	Port C
1	Mode 0, input	Mode 0, output	Don’t care
2	Mode 1, input	Mode 0, output	Handshake for port A
3	Mode 0, input	Mode 1, output	Handshake for port B
4	Mode 1, input	Mode 1, output	Handshake for ports A and B

Table 1. Some configurations of 8255

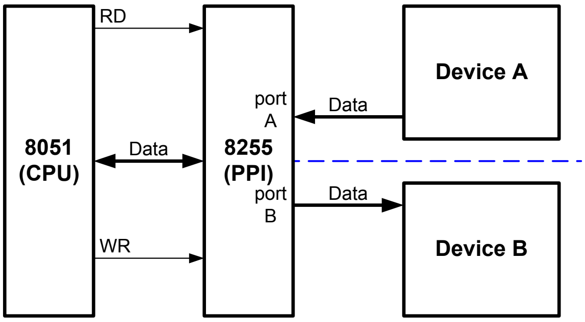
You are requested to do the following in this lab.

1. Setup the apparatus as shown in Figure 2.

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 of 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.

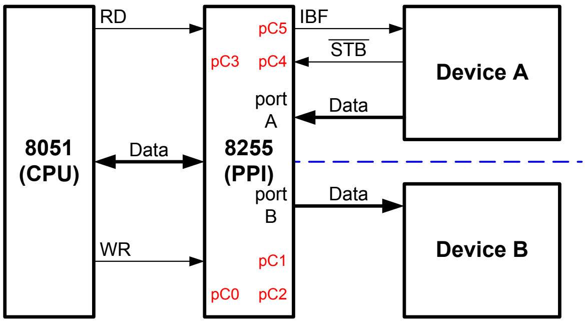
   Load the program into the AT89S8252 evaluation board and run it. See what happens.

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

   Q1. Can the CPU receive and report all inputs from the device if the device transfers its data at a rate of 4 bytes per second?

   Q2. Suppose now the device transfers its data at a rate of 1 byte per second. 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?

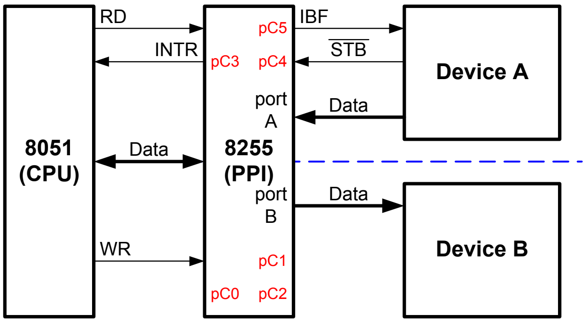
4. Appendix D3 lists a program for configuration 2(Ai1Bo0X.asm). In this program, as port A operates in mode 1, handshaking signal is provided through port C of 8255 and hence the CPU can make use of handshake to synchronize itself with an external device in a data transfer. This makes the transfer much 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?

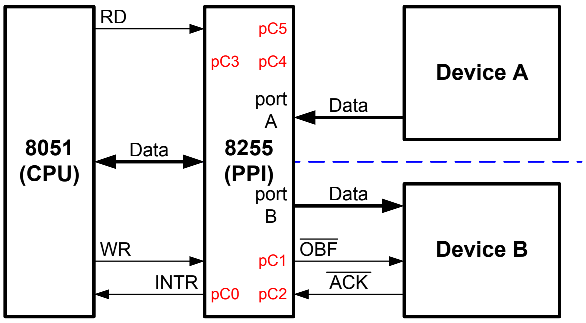
Study the program carefully. See how the program uses handshake to improve the performance.

5. Ai1Bo0X.asm does the job with programmed-I/O technique. It keeps checking the handshake 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.

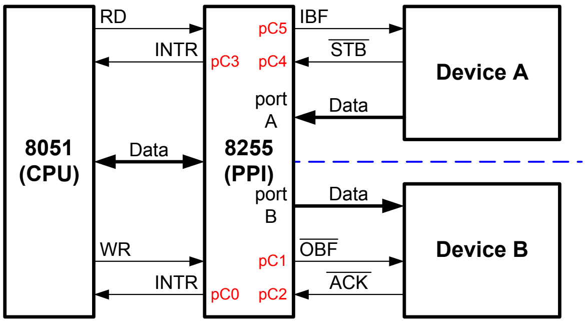


Complete the program by filling up the blank fields. Test your program with the evaluation boards. Study the program to see how it exploits interrupt to do the job.

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.



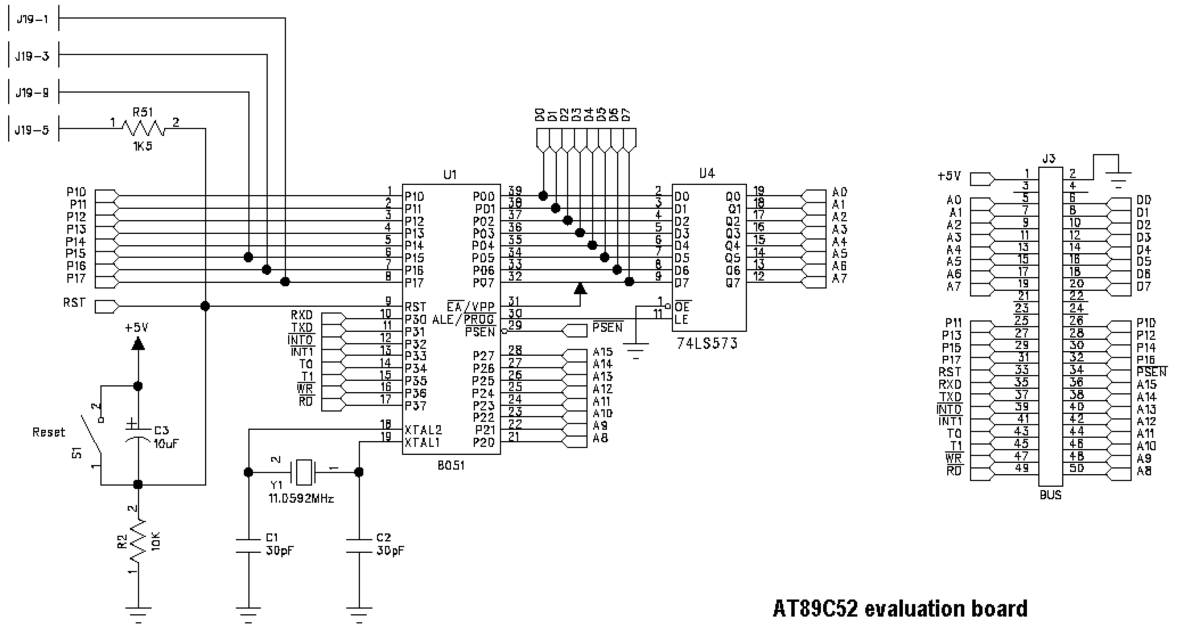
8. Try to configure the 8255 to function at other operation modes if time is allowed. (For more capable student)

- END -

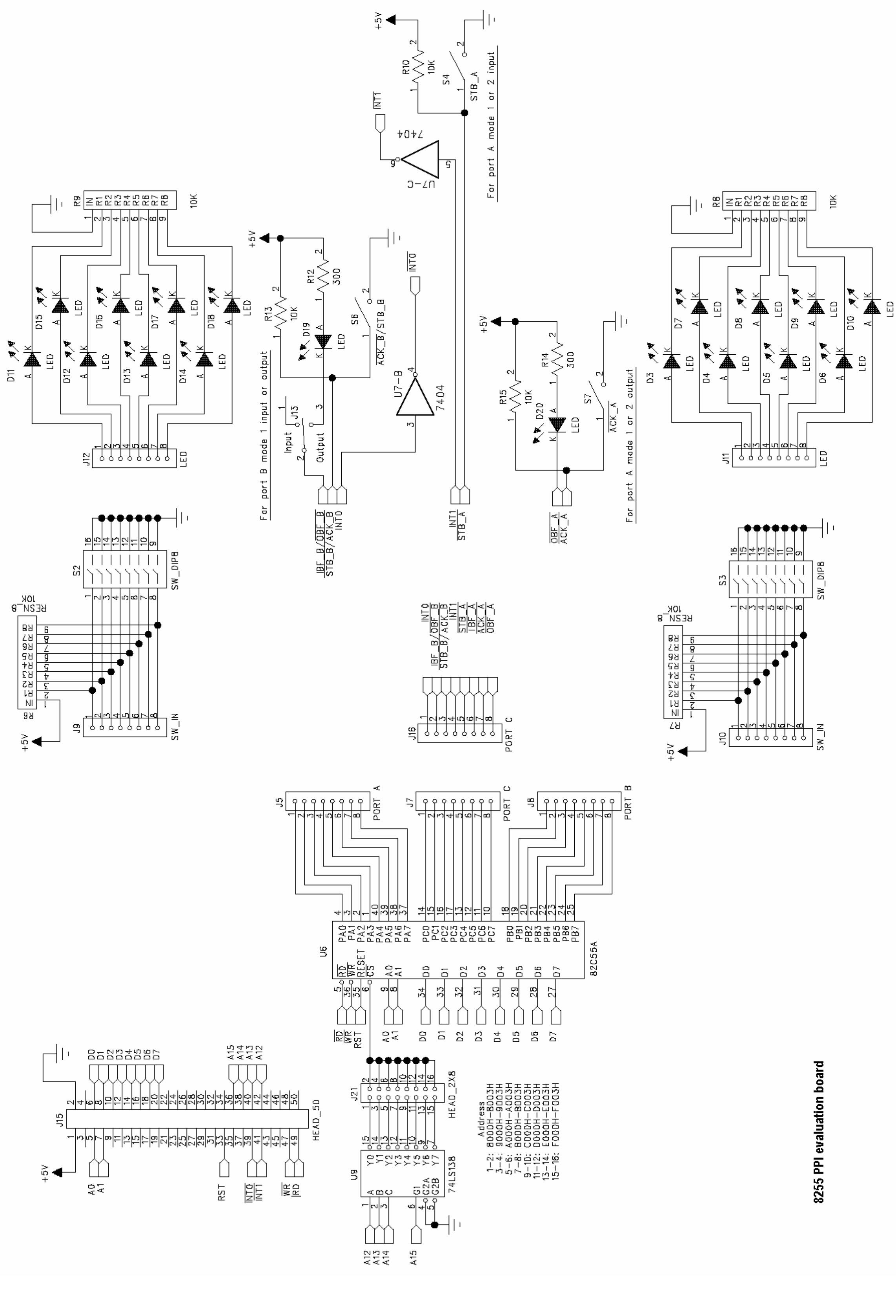
Appendix

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



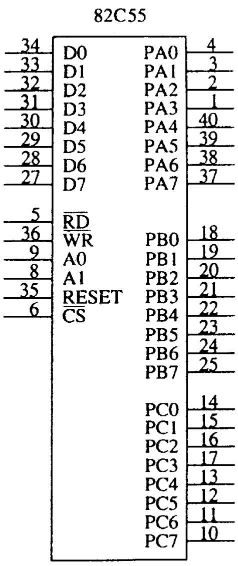
AT89C52 evaluation board

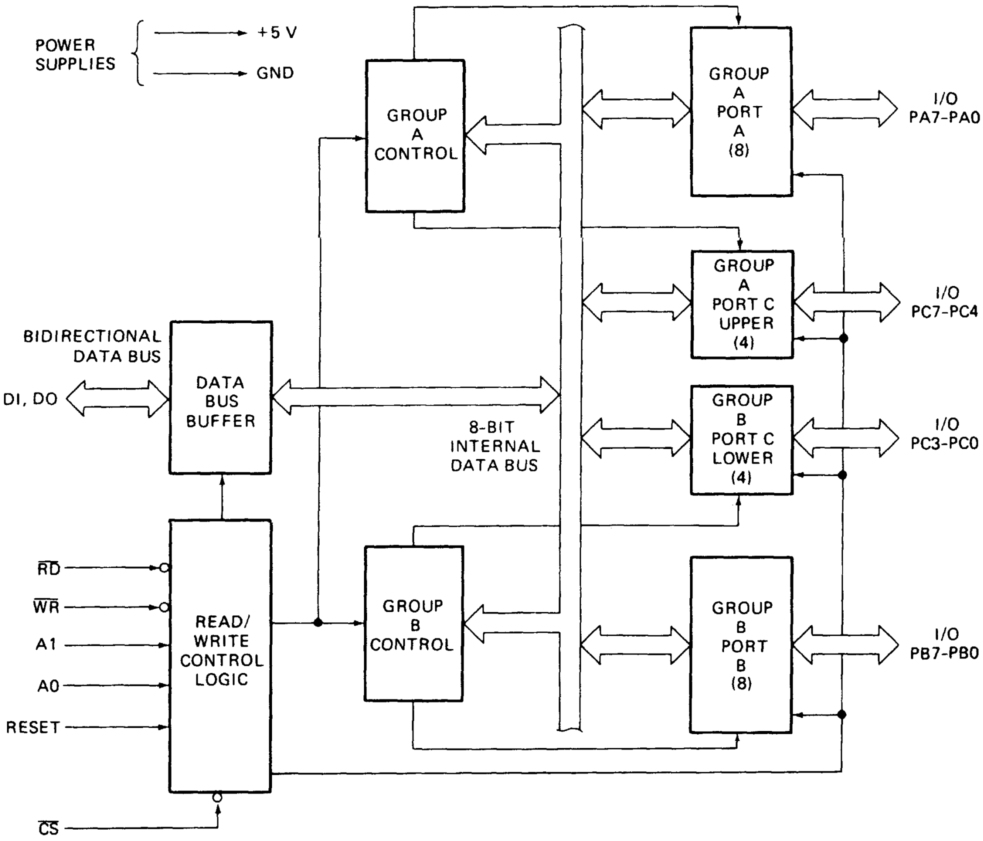


8255 PPI evaluation board

Appendix B. Summary of the technical information of 8255

- Internal structure:





Internal block diagram of 8255A programmable parallel port device. *(Intel Corporation)*

- Port and register addresses:

A_1	A_0	*Function*
0	0	Port A
0	1	Port B
1	0	Port C
1	1	Command Register

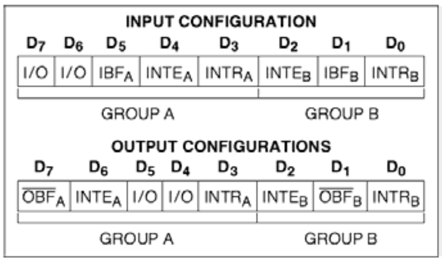
I/O port assignments for the 8255

- Port connections:

		Mode 0		Mode 1		Mode 2
Port A		IN	OUT	IN	OUT	I/O
Port B		IN	OUT	IN	OUT	Not used
Port C	0	IN	OUT	INTR_B	INTR_B	I/O
	1			IBF_B	$\overline{\text{OBF}}_B$	I/O
	2			$\overline{\text{STB}}_B$	$\overline{\text{ACK}}_B$	I/O
	3			INTR_A	INTR_A	INTR
	4			$\overline{\text{STB}}_A$	I/O	$\overline{\text{STB}}$
	5			IBF_A	I/O	IBF
	6			I/O	$\overline{\text{ACK}}_A$	$\overline{\text{ACK}}$
	7			I/O	$\overline{\text{OBF}}_A$	$\overline{\text{OBF}}$

A summary of the port connections for the 82C55 PIA

- Status word obtained by reading port C:



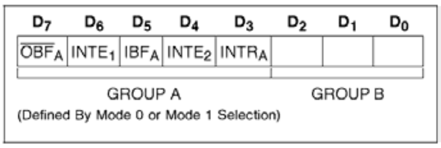
INPUT CONFIGURATION

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
I/O	I/O	IBF_A	$INTE_A$	$INTR_A$	$INTE_B$	IBF_B	$INTR_B$
GROUP A					GROUP B		

OUTPUT CONFIGURATIONS

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
$\overline{OBF}_A$	$INTE_A$	I/O	I/O	$INTR_A$	$INTE_B$	$\overline{OBF}_B$	$INTR_B$
GROUP A					GROUP B		

MODE 1 Status Word Format

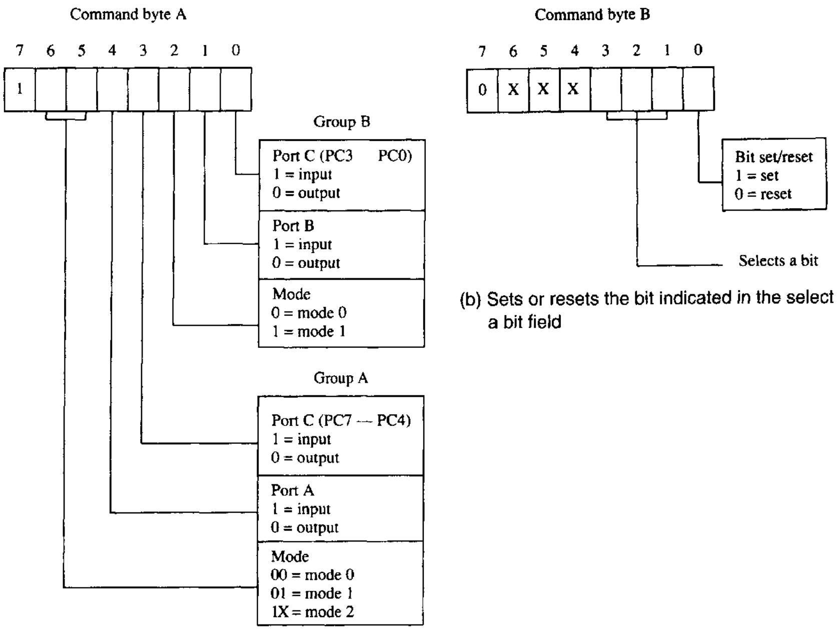


D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
$\overline{OBF}_A$	$INTE_1$	IBF_A	$INTE_2$	$INTR_A$			
GROUP A					GROUP B		

(Defined By Mode 0 or Mode 1 Selection)

MODE 2 Status Word Format

- Command words:

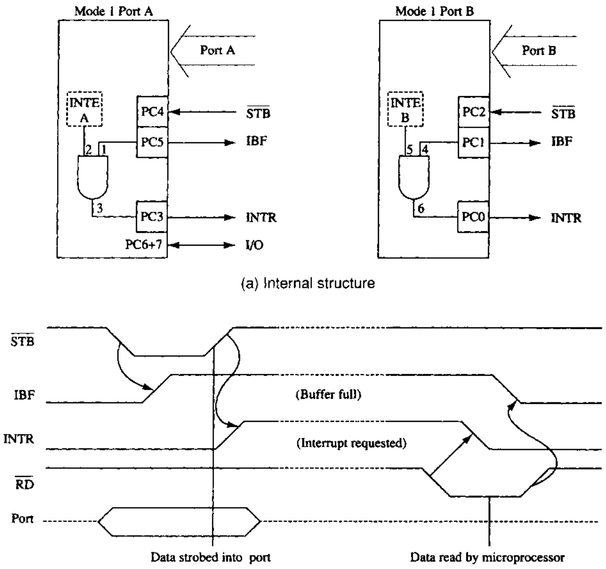


(b) Sets or resets the bit indicated in the select a bit field

(a) Programs ports A, B, and C

The command byte of the command register in the 82C55.

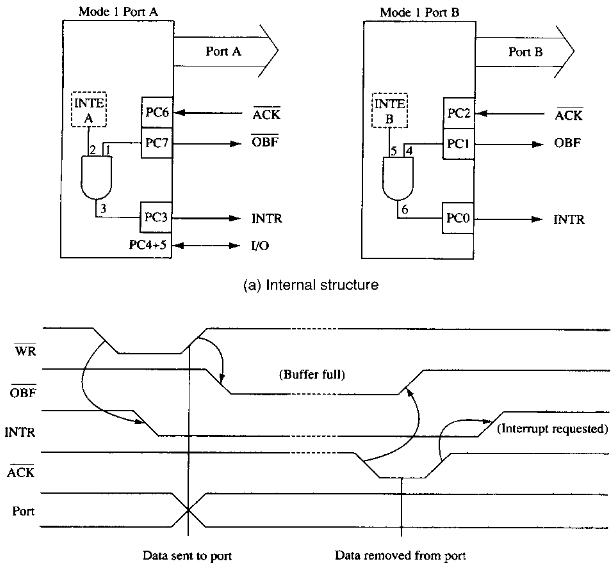
- Operation modes:



(a) Internal structure

(b) timing diagram

Strobed input operation (mode 1) of the 82C55.

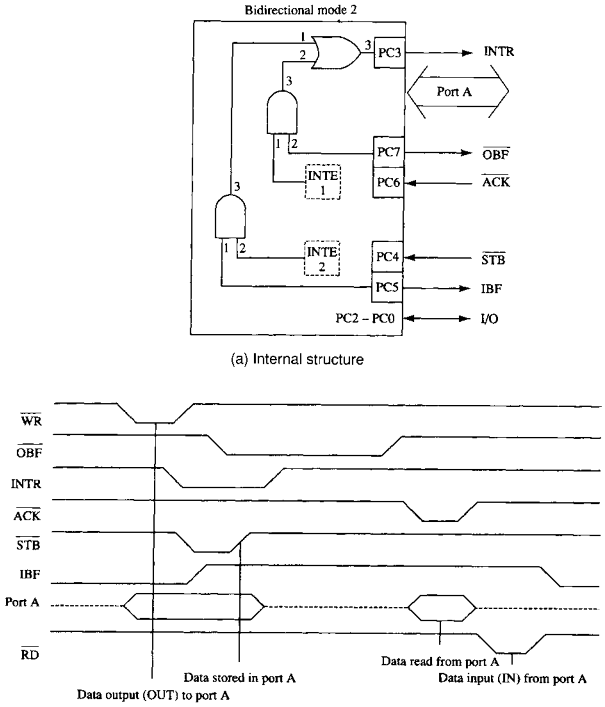


(a) Internal structure

(b) timing diagram

Strobed output operation (mode 1) of the 82C55.

Mode 1 operation



(a) Internal structure

(b) timing diagram

Mode 2 operation

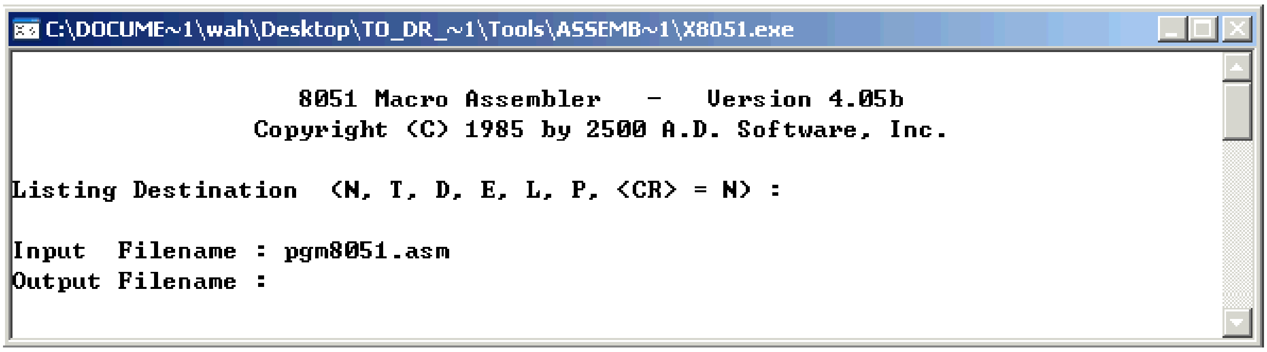
- Set/reset IRTEs:

	Port C Interrupt Signal Pin Number	To enable Interrupt Request Set Port C bit
MODE 1		
Port A IN	PC3	PC4
Port B IN	PC0	PC2
Port A OUT	PC3	PC6
Port B OUT	PC0	PC2
MODE 2		
Port A IN	PC3	PC4
Port A OUT	PC3	PC6

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.

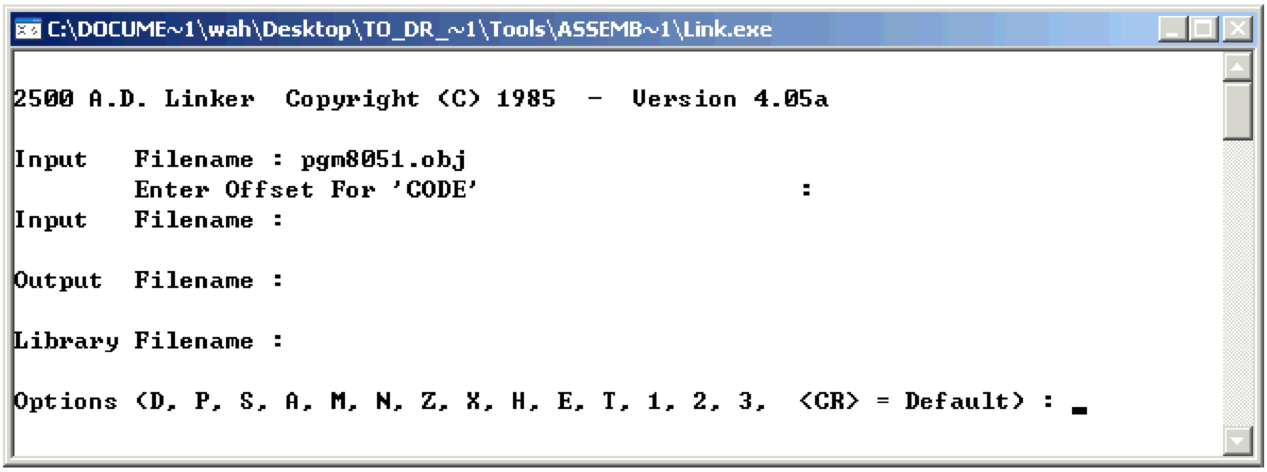
Suppose your program is ready and is now stored in the working directory where the 8051 cross-assembler(X8051.exe) and the 8051 linker(Link.exe) are in. Run X8051.exe to activate the cross-assembler. Figure C1 shows the user interface of the cross-assembler. In the interface, the cross-assembler will 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 '↵'. If no error is detected by the cross-assembler, an object file with extension '.obj' will be generated.



```
                    8051 Macro Assembler   -   Version 4.05b
                 Copyright (C) 1985 by 2500 A.D. Software, Inc.

Listing Destination  (N, T, D, E, L, P, <CR> = N) :

Input  Filename : pgm8051.asm
Output Filename :
```

Figure C1. User interface of X8051.exe

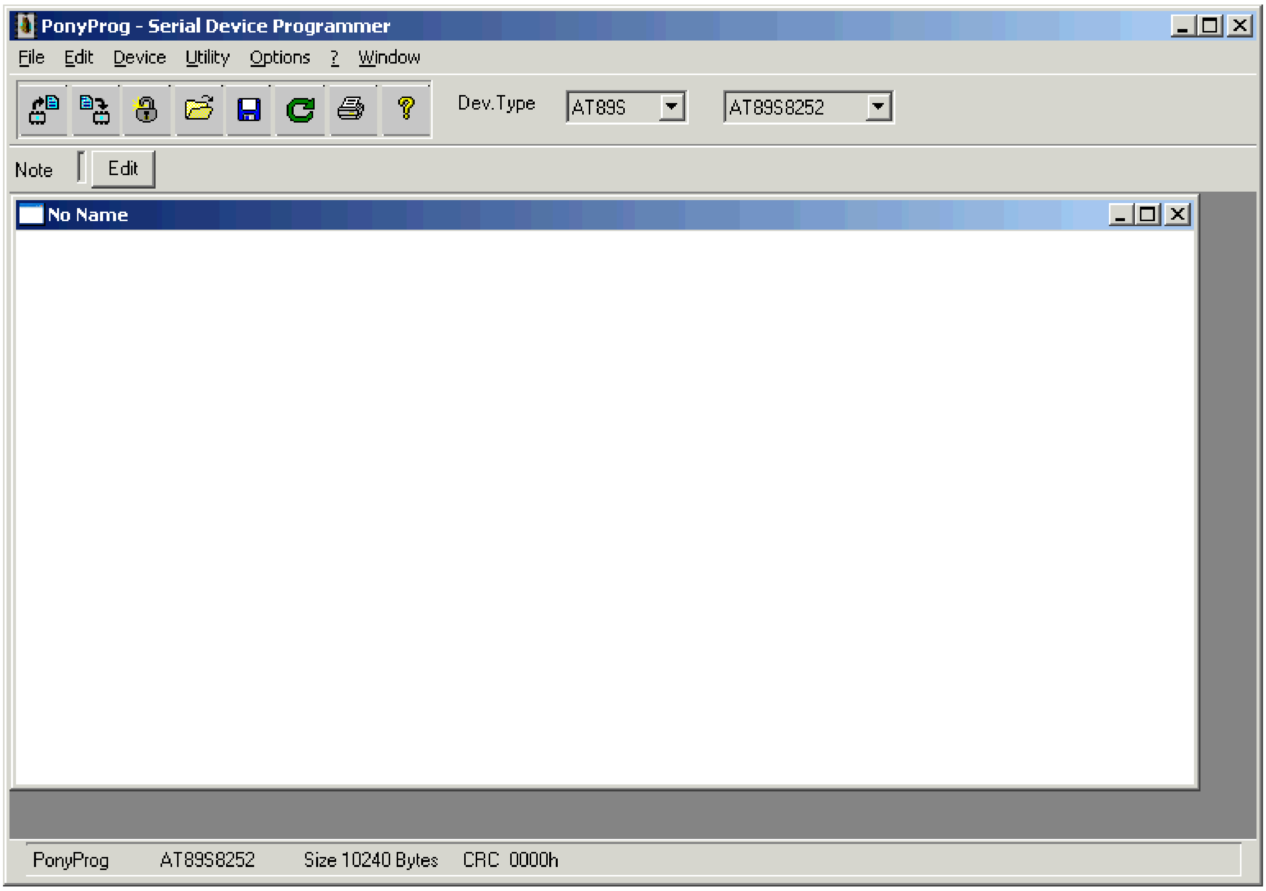
Run Link.exe to activate the linker. Figure C2 shows the user interface of the linker. The linker will prompt for inputting parameters. All you need to do is to specify the input filename. It should be an object file with extension '.obj'. As an example, Figure C2 shows the case that the input file is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



```
2500 A.D. Linker  Copyright (C) 1985  -  Version 4.05a

Input   Filename : pgm8051.obj
        Enter Offset For 'CODE'                  :
Input   Filename :

Output  Filename :

Library Filename :

Options (D, P, S, A, M, N, Z, X, H, E, T, 1, 2, 3,  <CR> = Default) : _
```

Figure C2. User interface of Link.exe

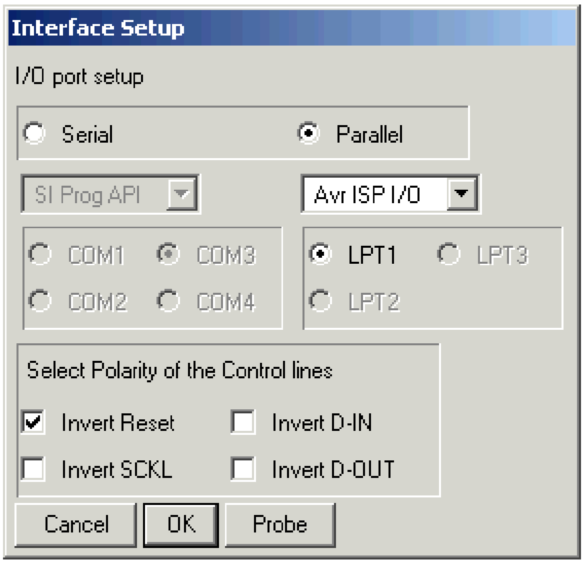
A universal programmer called PonyProg is provided in this lab. Figure C3 shows the user interface provided by the programmer. This programmer can program a specified binary file into the flash memory of an 8051-compatiable controller via the printer port of a computer system. To order to do it successfully, 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.


PonyProg - Serial Device Programmer
File Edit Device Utility Options ? Window
Dev.Type
AT89S
AT89S8252
Note
Edit
No Name

PonyProg AT89S8252 Size 10240 Bytes CRC 0000h

Figure C3. User interface of the programmer

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.



Interface Setup

I/O port setup

○ Serial ● Parallel

SI Prog API ▾ Avr ISP I/O ▾

○ COM1 ● COM3 ● LPT1 ○ LPT3

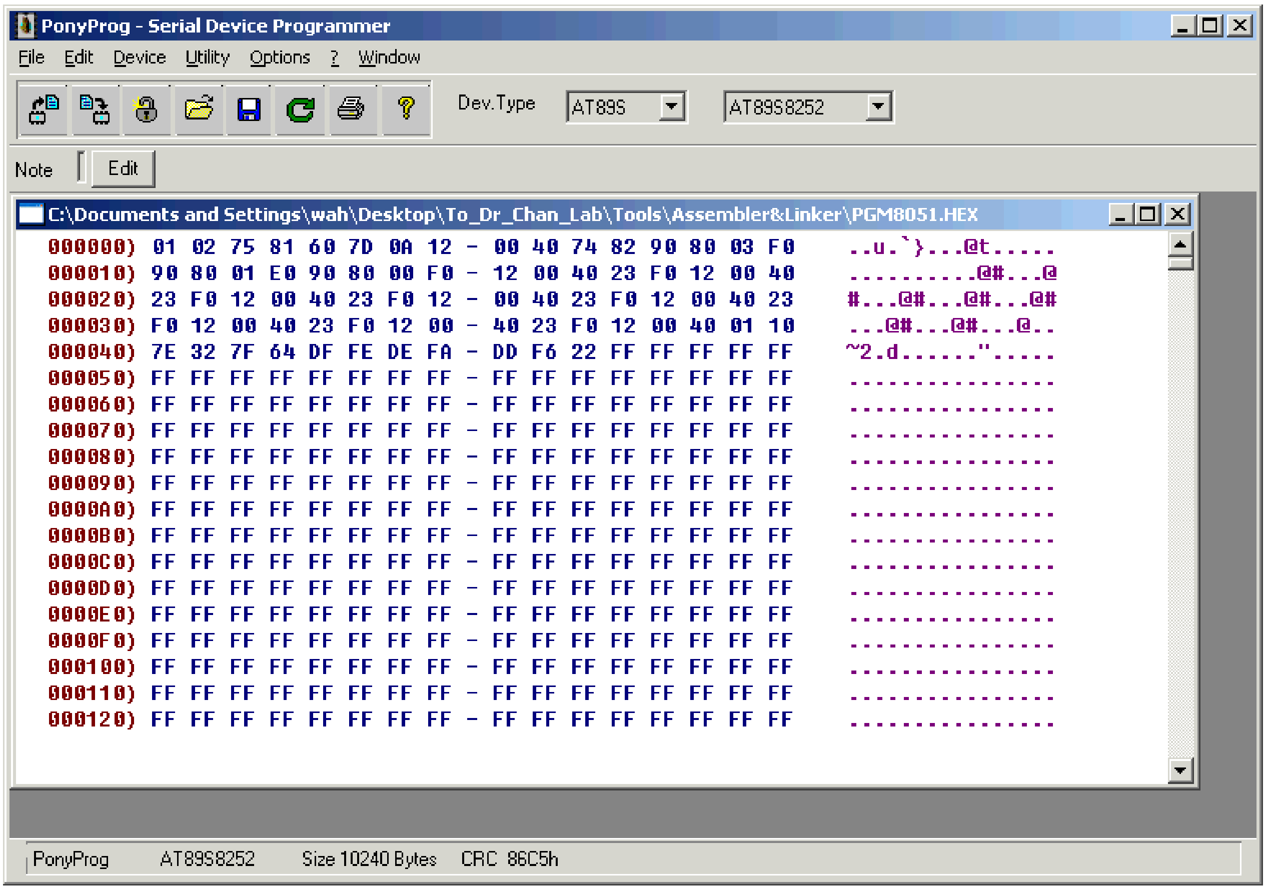
○ COM2 ○ COM4 ○ LPT2

Select Polarity of the Control lines

☑ Invert Reset ☐ Invert D-IN

☐ Invert SCKL ☐ Invert D-OUT

Figure C4. Setting for the interface between the evaluation board and the computer



PonyProg - Serial Device Programmer

Dev.Type AT89S ▾ AT89S8252 ▾

Note Edit

C:\Documents and Settings\wah\Desktop\To_Dr_Chan_Lab\Tools\Assembler&Linker\PGM8051.HEX

```
000000) 01 02 75 81 60 7D 0A 12 - 00 40 74 82 90 80 03 F0   ..u.`}...@t.....
000010) 90 80 01 E0 90 80 00 F0 - 12 00 40 23 F0 12 00 40   ..........@#...@
000020) 23 F0 12 00 40 23 F0 12 - 00 40 23 F0 12 00 40 23   #...@#...@#...@#
000030) F0 12 00 40 23 F0 12 00 - 40 23 F0 12 00 40 01 10   ...@#...@#...@..
000040) 7E 32 7F 64 DF FE DE FA - DD F6 22 FF FF FF FF FF   ~2.d......".....
000050) FF FF FF FF FF FF FF FF - FF FF FF FF FF FF FF FF   ................
000060) FF FF FF FF FF FF FF FF - FF FF FF FF FF FF FF FF   ................
000070) FF FF FF FF FF FF FF FF - FF FF FF FF FF FF FF FF   ................
000080) FF FF FF FF FF FF FF FF - FF FF FF FF FF FF FF FF   ................
000090) FF FF FF FF FF FF FF FF - FF FF FF FF FF FF FF FF   ................
0000A0) FF FF FF FF FF FF FF FF - FF FF FF FF FF FF FF FF   ................
0000B0) FF FF FF FF FF FF FF FF - FF FF FF FF FF FF FF FF   ................
0000C0) FF FF FF FF FF FF FF FF - FF FF FF FF FF FF FF FF   ................
0000D0) FF FF FF FF FF FF FF FF - FF FF FF FF FF FF FF FF   ................
0000E0) FF FF FF FF FF FF FF FF - FF FF FF FF FF FF FF FF   ................
0000F0) FF FF FF FF FF FF FF FF - FF FF FF FF FF FF FF FF   ................
000100) FF FF FF FF FF FF FF FF - FF FF FF FF FF FF FF FF   ................
000110) FF FF FF FF FF FF FF FF - FF FF FF FF FF FF FF FF   ................
000120) FF FF FF FF FF FF FF FF - FF FF FF FF FF FF FF FF   ................
```

PonyProg AT89S8252 Size 10240 Bytes CRC 86C5h

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

loop:
        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.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 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
;-----------------------------------
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
                                ; and dump it

        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           ;
        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.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 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

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

;------------------------------------
        end
```

Appendix E. Views of the evaluation boardsProgram listing

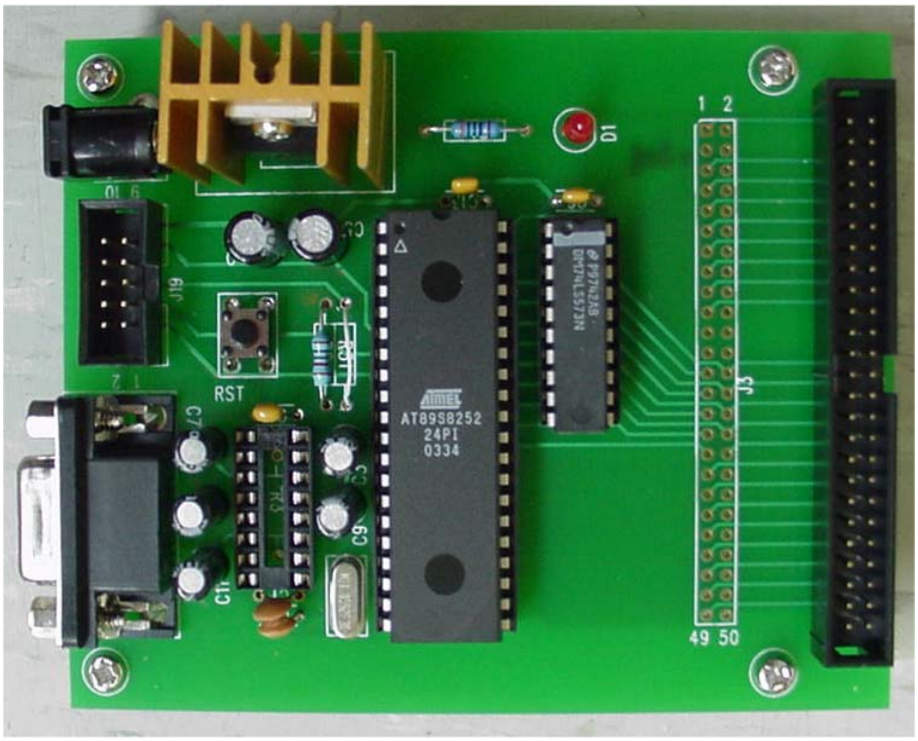


Figure A.E-1   8051 evaluation board

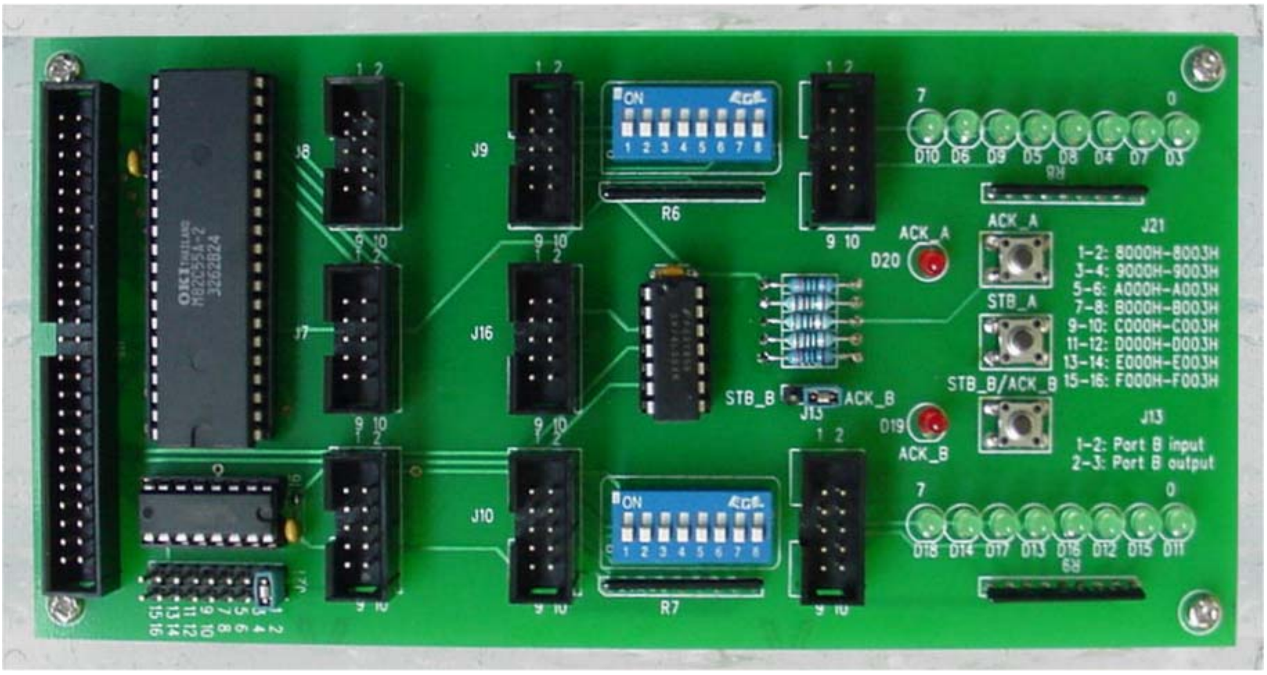


Figure A.E-2   8255 evaluation board