**I/O 1T:**



port1 equ 2A0h

port2 equ 2A8h

code segment

assume cs:code

start:

L1:mov dx,port1

in al,dx

call delay

mov dx,port2

out dx,al

call delay

jmp L1

mov ah,4ch

int 21h

delay proc

mov cx,0

wat2:push cx

mov cx,200h

wat1:loop wat1

pop cx

loop wat2

ret

delay endp

code ends

end start

**I/O T2:**



base   equ 280h;

t0     equ base+0;

t1     equ base+1;

t2     equ base+2;

ctl\_53 equ base+3;

code segment

assume cs:code

start:

       mov dx,ctl\_53;

       mov al,00110110b;

       out dx,al;

       mov dx,t0;

       mov ax,1000;

       out dx,al;

       mov al,ah;

       out dx,al;

       mov dx,ctl\_53;

       mov al,01110100b;

       out dx,al;

       mov dx,t1;

       mov ax,1000;

       out dx,al;

       mov al,ah;

       out dx,al;

       mov ax,4c00h;

       int 21h;

code ends;

end start;

**8253 T:**



**program1:**

; U5 8253

; 8253 Control Word: D7 D6 D5 D4 D3 D2 D1 D0

; D7: SC1, D6: SC0 [Select Counter Channel, 00=Counter0, 01=Counter1, 10=Counter2, 11=Useless]

; D5: RL1, D4: RL0 [Operation Type, 00=latch, 01=R/W lowB, 10=R/W highB, 11=R/W 16B, lowB first]

; D3: M2, D2: M1, D1: M0 [Mode Selection, 000=Mode0, 001=Mode1, X10=Mode2, X11=Mode3, 100=Mode4, 101=Mode5]

; D0: BCD, 0=Bin, 1=BCD.

; Example:

; \* "mov al, 00100111b"

; \* Counter0, R/W lowB, Mode3, BCD.

; Experiment requirements:

; \* Using Counter t0 and t1 to generate a squarewave and a negative

; pulse, with T=1ms.

; \* Clock: 1MHZ

; \* CS: 280h-287h

t0 equ 280H ; Counter0 port

t1 equ 281H ; Counter1 port

t2 equ 282h ; Counter2 port

ctl\_53 equ 283h ; The port of control register, address=283H

code segment

assume cs:code

start:

; Generating squarewave

; (1) write control word of counter0:

mov dx,ctl\_53 ; dx(control register) = ctl\_53(the port of control register)

mov al,00100111b ; t0 control word: 00 10 011 1 b, Counter0, R/W lowB, Mode3, BCD.

; Note: the calculated initail value of counter0 < 256, using lowB

out dx,al ; write control word of counter0

; (2) write initial value of counter0:

mov dx,t0 ; dx(control register) = counter0 port

mov al,10h ; the initial value, 10h = 1000, BCD

out dx,al ; write initial value of counter0, lowB

; if choose binary case, i.e. 00 10 011 0

; mov dx,to

; mov ax,1000

; out dx,al ; lowb first

; mov al,ah

; out dx,al

mov dx,ctl\_53

mov al,01100101b

out dx,al

mov dx,t1

mov al,10h

out dx,al

mov ah,4ch ; reach the end of procedure

int 21h ; return to operating system

code ends

end start

**program2:**

; U5 8253

t0 equ 290H

t1 equ 291H

t2 equ 292h

ctl\_53 equ 293h

code segment

assume cs:code

start:

mov dx,ctl\_53

mov al,10010000b ; counter2, W/R lowB, Mode0, BIN

out dx,al

mov dx,t2

mov al,09h ; initial value

out dx,al

mov ah,4ch

int 21h

code ends

end start

**program3:**

; U5 8253

t0 equ 298h

t1 equ 299h

t2 equ 29Ah

ctl\_53 equ 29Bh

code segment

assume cs:code

start:

mov dx,ctl\_53

mov al,00100111b ; counter0, R/W highB, Mode3, BCD

out dx,al

mov dx,t0

mov al,10h

out dx,al

mov dx,ctl\_53

mov al,01100111b ; counter1, R/W highB, Mode3, BCD

out dx,al

mov dx,t1

mov al,10h

out dx,al

mov ah,4ch

int 21h

code ends

end start

**program4:**

; U5 8253

t0 equ 288h

t1 equ 289h

t2 equ 28Ah

ctl\_53 equ 28Bh

code segment

assume cs:code

start:

mov dx,ctl\_53

mov al,01100101b ; counter1, R/W highB, Mode2, BCD

out dx,al

mov dx,t1

mov al,40h ; 40h

out dx,al

mov dx,ctl\_53

mov al,10010101b ; counter2, R/W lowB, Mode2, BCD

out dx,al

mov dx,t2

mov al,04h ; 04h

out dx,al

mov ah,4ch

int 21h

code ends

end start

**8255 T:**



**program1:**

pa equ 288h

pb equ 289h

pc equ 28Ah

ctl\_55 equ 28Bh

code segment

assume cs:code

start:

mov cx,10 ; for 10 times

; 8255 Initialization

mov dx,ctl\_55 ; register indirect addressing

mov al,10001011b ; control word, A: Mode0, Output, C(A) I/O Input;

; B: Mode0, Input, C(B) I/O Input

out dx,al ; write control word

ag:

mov dx,pb ; read PB0

in al,dx ;

test al,00000001b ; if the lowest bit is not 1, go ag

jz ag

mov dx,pc ; read PC0

in al,dx

mov dx,pa ; send to PA0 to show the LED light

out dx,al

ag2:

mov dx,pb ; read PB0

in al,dx

test al,00000001b ; if the value of al is not 0, loop back

jnz ag2

loop ag ; go back to ag

mov ah,4ch

int 21h

code ends

end start

**program2:**

t0 equ 280h

t1 equ 281h

t2 equ 282h

ctl\_53 equ 283h

pa equ 290h

pb equ 291h

pc equ 292h

ctl\_55 equ 293h

data segment

tab db 0,01h,03h,07h,0fh,1fh,3fh,7fh,0ffh,7eh,3ch,18h

count equ $-tab

data ends

code segment

assume cs:code,ds:data

start:

; load data segment

mov ax,data

mov ds,ax

; 8253 1s pulse

mov dx,ctl\_53

mov al,00100111b ; 00, Counter0; 10, R/W highB; 011, Mode3; BCD

out dx,al

mov dx,t0

mov al,10h

out dx,al

mov dx,ctl\_53

mov al,01100111b ; 01, Counter1; 10 R/W highB; 011, Mode3; BCD

out dx,al

mov dx,t1

mov al,10h

out dx,al

; Initail 8255

mov dx,ctl\_55

mov al,10001000b ; A: 00:Mode0; 0:Output; 1:C(A) I/O Input;

; B: 0:Mode0; 0:Output; 0:C(B) I/O Output.

out dx,al

ag3: ; setup loop

mov cx,count

xor si,si

ag:

mov dx,pc ; read PC7 pulse

in al,dx

test al,10000000b ; if the highest bit is not 1, go ag

jz ag

mov al,tab[si] ; send tab to PA to show the LED light

mov dx,pa

out dx,al

ag2:

mov dx,pc ; read PC7

in al,dx

test al,10000000b ; if the highest bit is not 1, go ag2

jnz ag2

inc si ; si++

loop ag

jmp ag3

mov ah,4ch

int 21h

code ends

end start

**8255A T:**



**program1:**

pa equ 2a0h

pb equ 2a1h

pc equ 2a2h

ctl\_55 equ 2a3h

data segment

tab db 3fh,06h,5bh,4fh,66h,6dh,7dh,07h,7fh,6fh

key db ? ; initialization

data ends

code segment

assume cs:code,ds:data

start:

mov ax,data ; load data segment

mov ds,ax

mov cx,10

; Initial 8255

mov dx,ctl\_55

mov al,10010001b ; A: Mode0, Input, C(A) I/O Output; B: Mode0, Output, C(B) I/O Input

out dx,al

; pulse: 0-1-0-1-...

; read pulse

ag1:

call crt\_8 ; call function crt\_8

mov dx,pc ; read PC0(experiment requirement)

in al,dx

test al,00000001b

jz ag1 ; if read 1, continue, else loop back and wait for 1

mov dx,pa ; read the key value from PA0

in al,dx

mov key,al ; record the key value in 'key'

ag2:

call crt\_8 ;

mov dx,pc ; read PC0

in al,dx

; end of reading pulse

test al,00000001b

jnz ag2 ; if read 0, continue, else loop back and wait for 0

loop ag1 ; loop

mov ah,4ch ; back to OS

int 21h

crt\_8 proc

push cx ; push the value of cx to stack

mov al,key ; show highB

mov cl,04h ;

ror al,cl ; bitwise rotation

and al,0fh ; ignore highest 4b

; lookup the table

mov bx,offset tab ; bx = offset of tab

XLAT ; (al) = ((bx)+(al))

mov dx,pb ; output highb segment code

out dx,al

mov dx,pc ; output highb bit code

mov al,10000000b

out dx,al

mov dx,pc ; close the light

mov al,0

out dx,al

mov al,key ; show lowB

and al,0fh

mov bx,offset tab

XLAT

mov dx,pb ; output lowb segment code

out dx,al

mov dx,pc ; output lowb bit code

mov al,01000000b

out dx,al

mov dx,pc ; close the light

mov al,0

out dx,al

pop cx ; recover

ret

crt\_8 endp

code ends

end start

**program2:**

pa equ 290h

pb equ 291h

pc equ 292h

ctl\_55 equ 293h

t0 equ 280h

t1 equ 281h

t2 equ 282h

ctl\_53 equ 283h

data segment

tab db 3fh,06h,5bh,4fh,66h,6dh,7dh,07h,7fh,6fh

second db 0

data ends

code segment

assume cs:code,ds:data

start:

mov ax,data

mov ds,ax

mov dx,ctl\_53

mov al,00100111b

out dx,al

mov dx,t0

mov al,10h

out dx,al

mov dx,ctl\_53

mov al,01100111b

out dx,al

mov dx,t1

mov al,10h

out dx,al

mov dx,ctl\_55

mov al,10001000b

out dx,al

ag:

call led\_8

mov dx,pc

in al,dx

test al,10000000b

jz ag

mov al,second ; +1s

add al,1

DAA

mov second,al

.IF second==60h ; if 60s, return

mov second,0

.endif

ag1:

call led\_8

mov dx,pc

in al,dx

test al,10000000b

jnz ag1

jmp ag

mov ah,4ch

int 21h

led\_8 proc

mov cl,04h

mov al,second

ror al,cl

and al,0fh

mov bx,offset tab

XLAT

mov dx,pa

out dx,al

mov dx,pb

mov al,10000000b

out dx,al

mov dx,pb

mov al,0

out dx,al

mov al,second

and al,0fh

mov bx,offset tab

XLAT

mov dx,pa

out dx,al

mov dx,pb

mov al,01000000b

out dx,al

mov dx,pb

mov al,0

out dx,al

ret

led\_8 endp

code ends

end start