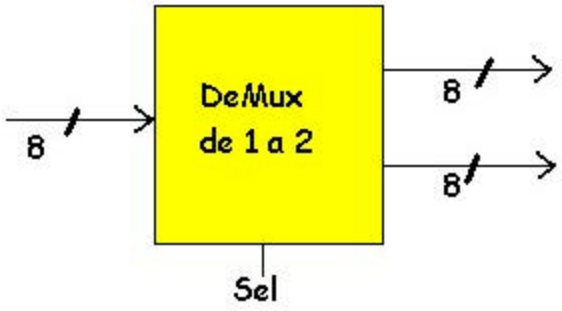
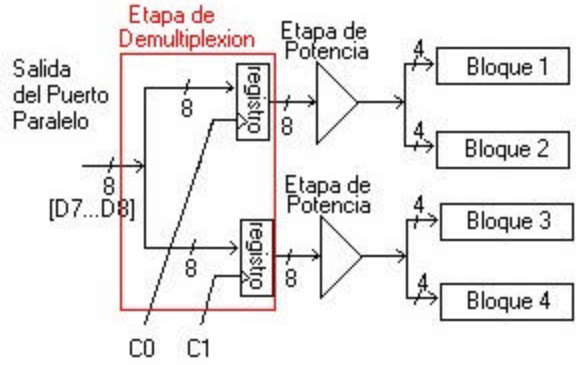
control de Luces por el Puerto Paralelo 378H (16 Luces 4 bloques de 4)

1.- Lo primero
Tenemos que tener un cuenta que el puerto paralelo tiene solo 8 salidas de datos (D7...D2) y nosotros necesitamos controlar 16 luces.

Para esto utilizamos una técnica de Demultiplexion:



Solo que en este, en vez de usar un selector(Sel) usaremos 2 registros de desplazamiento, y un esquema mas amplio seria:



Aquí C0 y C1 (Salidas de control del Puertos) sirven para demultiplexar la salida lograr obtener 16 salidas de 4 entradas por bloque.
La etapa de potencia es para que se puedan usar focos de 12V, ya que la salida del Puerto es TTL (5V)

2.-Como se envía la información.-
Acabamos de ver como funciona el circuito, ahora vamos a ver como se envía la información para que se pueda apreciar el juego de luces que queremos:
La etapa por las que tiene que pasar la información para que se vea un solo momento de un efecto es:

- Se envía, la información del bloque 1 y bloque 2.
- Se hace que en C0 (se ocasione un Clock).
- Se envía, la información del bloque 3 y bloque 4.
- Se hace que en C1 (se ocasione un Clock).

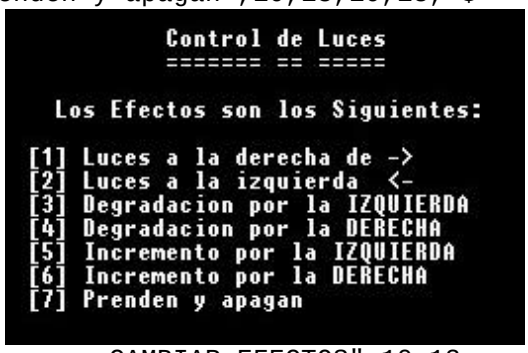
Esa es la manera como se envía una etapa, pero recordemos que un efecto completo tiene 4 etapas, que deben ser mandadas de la manera antes dicha.

3.- Explicando el Código.-
El código fuente esta en el archivo luces.txt y luces.asm

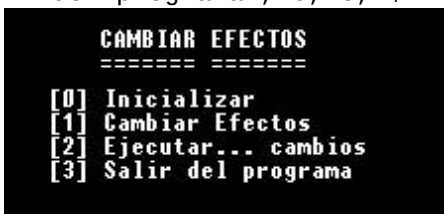
Encabezado:
.286 →Ya que usamos instrucciones del procesador 286 para arriba
El ejemplo mas claro esta en pusha (guarda todos los registros en la pila) y popa (recupera los registros)
.model small →Usaremos el modelo small
.stack 100h →Guardamos un segmento de 256 bytes (100H) para la pila

Mi segmento de datos:

```
.data
;Aquí estan mis menus:
;El primero que es mostrado cuando se realizan los efectos
Titulo  db "            Control de Luces",10,13
        db "            ======= == ====="
Tefectos db 2 dup (10,13),"    Los Efectos son los Siguientes:"
        db 2 Dup(10,13)
        db "  [1] Luces a la derecha de ->",10,13
        db "  [2] Luces a la izquierda  <-",10,13
        db "  [3] Degradacion por la IZQUIERDA",10,13
        db "  [4] Degradacion por la DERECHA",10,13
        db "  [5] Incremento por la IZQUIERDA",10,13
        db "  [6] Incremento por la DERECHA",10,13
        db "  [7] Prenden y apagan",10,13,10,13,'$'
```

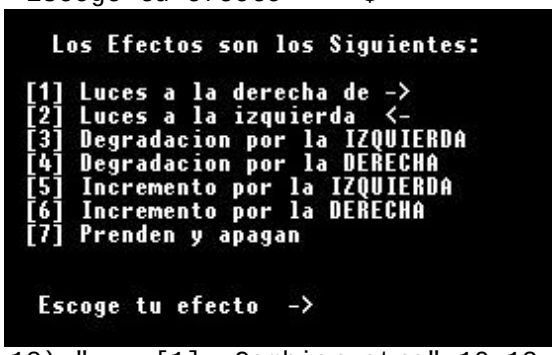



```
Tcambia db "               CAMBIAR EFECTOS",10,13
        db "               ======= =======",2 dup(10,13)
        db "    [0] Inicializar",10,13
        db "    [1] Cambiar Efectos",10,13
        db "    [2] Ejecutar... cambios ",10,13
        db "    [3] Salir del programa",10,13,'$'
```

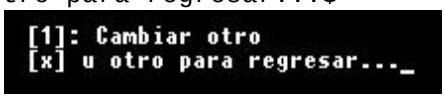



```
Tsel1   db "               SELECCIONE",2 Dup(10,13)
        db 10,13,"    Bloque a cambiar [1-4]    -> $"
```

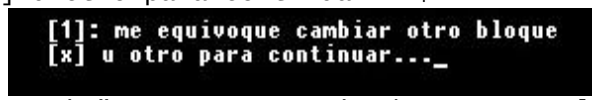

```
Tsel2   db 10,13,"   Escoge tu efecto  -> $"
```



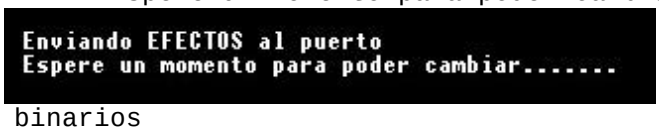

```
Tsel3   db 2 Dup(10,13),"    [1]: Cambiar otro",10,13
        db "    [x] u otro para regresar...$"
```




```
Tsel4   db 2 Dup(10,13),"    [1]: me equivoque cambiar otro bloque",10,13
        db "    [x] u otro para continuar...$"
```

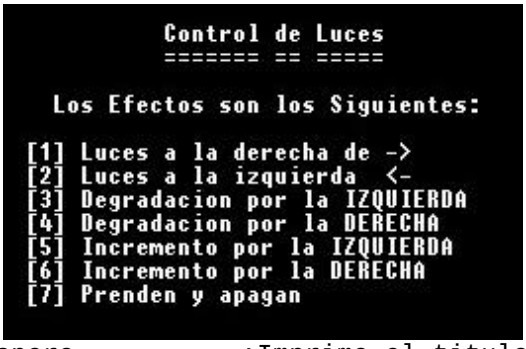



```
ESPERA  db 2 dup(10,13),"           Enviando EFECTOS al puerto",10,13
        db "           Espere un momento para poder cambiar..... $"
```




```
;Mis Efectos  en binarios
EFECTOSTOTALES LABEL BYTE
E1 db 00001000b, 00000100b, 00000010b, 00000001b
E2 db 00000001b, 00000010b, 00000100b, 00001000b
E3 db 00001111b, 00000111b, 00000011b, 00000001b
E4 db 00001111b, 00001110b, 00001100b, 00001000b
E5 db 00001000b, 00001100b, 00001110b, 00001111b
E6 db 00000001b, 00000011b, 00000111b, 00001111b
E7 db 00001111b, 00000000b, 00001111b, 00000000b

EFECTOS dw 4 Dup (4 Dup (0))

BLOQUES LABEL Byte
    B1 db 4 Dup(0)
    B2 db 4 Dup(0)
    B3 db 4 Dup(0)
    B4 db 4 Dup(0)

.data?
;Mis datos no inicializados
NUMERO  db ?      →Usado para guardar el numero de bloque y de efecto
CBLOQUE dw ?      →Dirección del Bloque inicial a ser cambiado
CEFECTO dw ?      →Dirección del Efecto inicial a mover al bloque
```

```
.code       ;Aquí comienza el código
            Mov ax,@data ;→Comienzo definiendo mi segmento de datos
            mov ds,ax
            call inicializar;→Inicializo mis efectos
            jmp cmabini;→Salto a la rutina de cambios
REPITE:     call limpiapantalla     ;->Limpia la pantalla
            lea dx,titulo           ;Imprimo el Menu principal
            mov ah,9                ;Usando el servicio 9 de la int 21H
            int 21h
```




```
            lea dx,espera           ;Imprimo el titulo de espera
            mov ah,9                ;Imprimo mensaje de espera
            int 21h
```


```
            mov cx,4                ;Muestro 4 veces
MUESTRA:    call mostrar            ;usando rutina mostrar
            loop muestra
cmabini:    call cambiar            ;Llamar a rutina de cambios
            cmp ax,0                ;Si retorno en ax=’0’ error
            jz salir                ;Si hay error salgo
            jmp repite              ;Si no hay error salto a repite
salir:      call limpiapantalla     ;Limpio pantalla
            mov ah,4ch              ;Regreso al sistema operativo
            int 21h                 ;Usando el servicio 4Ch de int 21h
;*****************************************************************
                        Aquí COMIENZAN MIS RUTINAS
;*****************************************************************
mostrar proc near
            pusha ;Guardo mis registros ante posibles cambios
            mov di,0;Pongo mi puntero de destino en ‘0’
            mov cx,4;Voy a mostrar 4 etapas, tal como lo explique
                    ;anteriormente
EFE:        mov ax,[EFECTOS+di];Muevo etapa a ax(Bloq4,Bloq3,Bloq2,Bloq1)
            mov bl,al;muevo etapa(bloque2,bloque1) en bl
            xchg ah,al;hago cambio y muevo (bloque4 y bloque3) en b1
            call imph;imprimo la parte alta (bloque4 y bloque3)
            call impl;imprimo parte baja
            call seg_1;Hago retardo de 1 segundo
            inc di;Incremento direccion de mi puntero
            inc di;Es decir paso a siguite etapa de (16bits=2bytes)
            loop EFE
            popa
            ret
mostrar endp
```

```
imph proc near
            mov dx,378h;Mando mi bloque4 y bloque3
            out dx,al   ;Al Puerto 378H
            mov al,1    ;Y ocasiono en Clock en C0, primero mando un '0'
            mov dx,37ah
            out dx,al
            mov al,0
            out dx,al   ;Luego mando un '1'
            ret
imph endp
impl proc near
            mov al,bl   ;Muevo el Bloque2 y Bloque1  a al
            mov dx,378h ;Y lo mando al puerto
            out dx,al
            mov al,2    ;Ocasiono un clock en C2, mando un '1'
            mov dx,37ah
            out dx,al
            mov al,0
            out dx,al   ;Luego mando un '0'
            ret
impl endp
;*********************************************************************
;Rutina que usa el reloj del sistema para lograr el retardo deseado
;No es exacto, pero el resultado aproximadamente es el deseado
;*********************************************************************
seg_1 proc  near
            pusha
            mov ah,2ch;Pido reloj del sistema
            int 21h
            mov bh,dh;dh->Segundos del sistema, lo guardo en bl
            add bh,1;Aumento en 1 a la cantidad
            cmp bh,60;Comparo si es mayor que 60
            jl contseg;si esn menor salto a contseg
            sub bh,60;si es mayo le quito 60
contseg:
            mov ah,2ch;Pido nuevamente reloj del sistema
            int 21h
            cmp dh,bh;comparo si aumento en 1
            jne contseg;si no aumento salto a cont seg
            popa
            ret
seg_1 endp
;         LIMPIA LA PANTALLA
limpiapantalla    PROC      NEAR
          pusha
          MOV       AX,0600H            ;REQUEST SCROLL SCREEN
          MOV       BH,07H              ;NORMAL
          MOV       CX,0000H            ;FROM 00,00
          MOV       DX,184FH            ;TO 24,79
          INT       10H                 ;CALL BIOS
          MOV       DX,0000        ;PONE EL CURSOR A FILA 00, COLUMNA 00
          MOV       AH,02               ;REQUEST SET CURSOR
          MOV       BH,00               ;PAGE 10
          INT       10H                 ;CALL BIOS
          popa
          RET
limpiapantalla    ENDP
```

```
;Rutina de Cambios, doria considerarse como la principal
cambiar proc near
ventana1:   call limpiapantalla;limpio la pantalla
            lea dx,TCAMBIA;muestro mensaje de cambios
            mov ah,9
            int 21h
```

```
      CAMBIAR EFECTOS
      ======= =======

  [0] Inicializar
  [1] Cambiar Efectos
  [2] Ejecutar... cambios
  [3] Salir del programa
```

```
cambi:      mov ah,7    ;Pido entrada de carácter sin eco
            int 21h     ;Para seleccion de menu
            cmp al,'0'  ;Si es ‘0’ salto a iniefe
            jz  iniefe
            cmp al,'1'  ;Si es ‘1’ salto a ventana2
                        ;Que es el menu de cambio de bloques
            jz ventana2
            cmp al,'2'  ;Si es ‘2’ salto a fin de cambios
                        ;y luego muestro efectos nuevmanete repite
            jz FINCAMBIOS
            cmp al,'3'; si he presionado 3 continuo
            jnz cambi;Y si no he presionado ‘3’ salto a cambi
            mov ax,0    ;Si presione ‘3’ hago que retorne pero con ax=0
                        ;Que es salir del programa
            jmp FINCAMBIOS
iniefe:     call inicializar  ;Hago rutina de inicializacion
            ;En realidad copia los primeros 4 efectos a Bloques
            ;Y luego los transforma y coloca en efectos
            ;para que puedan ser enviadas por el puerto 378
            jmp ventana1

ventana2:   call Limpiapantalla    ;Limpia la pantalla
            lea dx,Tsel1      ;Aparece el mensaje de escoger Bloque
            mov ah,9
            int 21h
```

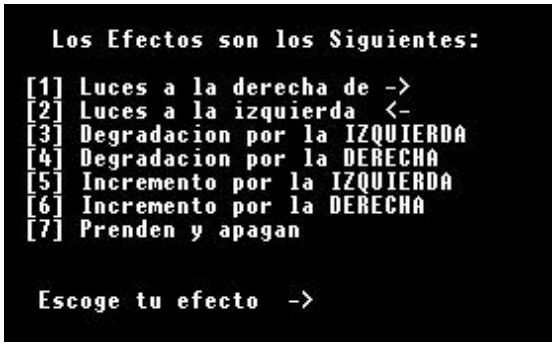
```
            SELECCIONE

  Bloque a cambiar [1-4]    -> _
```

```
            mov al,4          ;Va usarse en tecla, es decir maximo numnero
                              ;a ingresar es 4
            call tecla        ;rutina de ingreso de tecla
            call cambiabloq   ;Rutina de cambio de Bloque
            lea dx,Tsel4      ;Aparece el mensaje de confirmacion
            mov ah,9          ;Si es
            int 21h
```

```
  [1]: me equivoque cambiar otro bloque
  [x] u otro para continuar..._
```

```
            mov ah,7    ;Luego ingreso dato
            int 21h
            cmp al,'1'  ;Y  si es ‘1’
            jz ventana2 ;Salto a ventana2
```

```
            call limpiapantalla ;Limpio pantalla
            lea dx,Tefectos
            mov ah,9
            int 21h
            lea dx,Tsel2
            mov ah,9
            int 21h
```

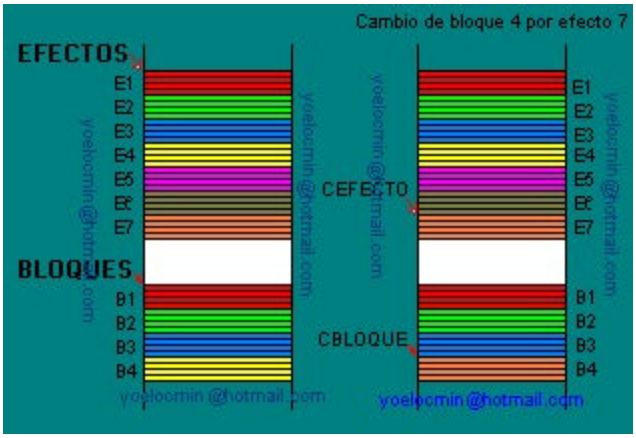



```
            mov al,7;máximo de efectos es 7
            call tecla
            call cambiaefecto
            call transforma
            lea dx,Tsel3
            mov ah,9
            int 21h
            mov ah,7;Ingreso dato sin eco
            int 21h
            cmp al,'1';Si es ‘1’
            jz ventana2;Salto a ventana2
            jmp ventana1;Si es otro salto ventana1
FINCAMBIOS: ret
CAMBIAR      endp
tecla proc near
            mov cl,al;Tenemos en al el máximo numero a entrar en Hexa
            add cl,30h;lo convertimos en Ascii y lo guardamos en cl
espt:       mov ah,7h;Ingreso caracter sin eco
            int 21h
            cmp al,'1';Comparo si tecla ingresada esta en el rango
            jl error;es menor que ‘1’
            cmp al,cl
            jg error;Es mayor que Máximo, si lo es salto a error
            mov dl,al;Si esta en el rango lo muestro en pantalla
            mov ah,2;Usando el servicio 2 de la int 21h
            int 21h
            sub al,30h;Luego al valor le quito 30H
            mov ah,0
            mov byte ptr[NUMERO],al;Lo muevo a numero
            ret
error:      mov dl,7;En caso de ser error, imprimo un beep
            mov ah,2;Cuyo codigo en hexa es ‘7’
            int 21h
            jmp espt; y salto a espt
tecla endp
```

```
;Aquí vienen las rutinas mas difíciles de entender y se puede decir
;que es el núcleo para poder lograr los cambios de forma independiente
;En los bloques y en los Efectos
;**************************************
;Esta rutina lo que hace es poner en CBLOQUE
;la dirección del bloque que deseo cambiar
 cambiabloq proc near
            mov al,byte ptr[numero], para eso copio el numero en al
            dec ax;lo disminuyo en '1'
            mov bl,4;y lo multiplico por 4, ya que son 4 bytes por bloque
            mul bl
            lea bx,bloques;luego obtengo la dirección de Bloques en bx
            add ax,bx;y le sumo el dato anterior
            mov word ptr [CBLOQUE],ax
            ret
cambiabloq endp
;Esta rutina guarda en CEFECTO la dirección del efecto
;Y luego copia el bloque correspondiente de 4 bytes
;a la dirección CBLOQUES
cambiaefecto proc near
            mov al,byte ptr[numero]
            dec ax
            mov bl,4
            mul bl
            lea bx,EFECTOSTOTALES
            add ax,bx
            mov di,ax
            mov ax,word ptr [CBLOQUE]
            mov si,ax
            mov cx,4
mueve:      mov al, byte ptr[DI]
            mov byte ptr[si],al
            inc di
            inc si
            loop mueve
            ret
cambiaefecto endp
```

En forma grafica hace lo que se observa en la figura:
Se posiciona en Efectos y le suma el nuemro del efecto deseado –1 *4

CEFECTO=(numero-1)*4+EFECTOSTOTALES
CBLOQUE=(numero-1)*4+BLOQUES
Luego se copia los 4 bytes de Cefecto a Cbloque



```
;Rutina de transformación, ahora que ya tenemos los efectos  que queremos
;En Bloques, estos los tenemos que transformar en un código que
;Se pueda enviar por el Puerto 378H
```

Como sabemos cuando definimos los datos, en EFECTOSTALES, estan en codigo de ‘1’ y ‘0’ (‘1’ = ON, ‘0’ = OFF), pero solo nos sirven los 4 primeros bits de cada dato, y estos son copiados a BLOQUES (call inicializar al inicio del código)
Donde lo que ocasiona es que los tengamos de la siguiente manera

```
BLOQUES LABEL BYTE
B1 db 00001000b, 00000100b, 00000010b, 00000001b
B2 db 00000001b, 00000010b, 00000100b, 00001000b
B3 db 00001111b, 00000111b, 00000011b, 00000001b
B4 db 00001111b, 00001110b, 00001100b, 00001000b
```

Pero no nos sirven de esta manera, ya que no podemos enviarlos asi al puerto, debemos transformarlos

```
transforma proc near
            mov si,0;inicializo Puntero Si
            mov di,0;inicializo Puntero Di
            mov cx,4;Cantidad de veces a hacer el cambio
                    ;Tambien lo puedo tomar como cantidad de efectos
                    ;Que van ser transformados
;Voy a poner los cambios de la primera etapa
camblef:    mov ah,byte ptr B1[DI];Muevo el bloque1 a ah=00001000b
            mov al,byte ptr B2[DI];Muevo el Bloque2 a al=00000001b
            shl al,4;Desplazo 4 bits ← AL => AL=10000000b
            shl ax,4;Desplazo 4 bits ← AX => AH=00011000b
            mov bh,ah;copio a BH=00011000b
            mov ah,byte ptr B3[DI];Muevo el bloque3 a ah=00001111b
            mov al,byte ptr B4[DI];Muevo el bloque4 a al=00001111b
            shl al,4;Desplazo 4 bits ← AL => AL=11110000b
            shr ax,4;Desplazo 4 bits → AX => AL=11111111b
            mov ah,bh;Copio bh en al => AX=00011000111111b
            mov word ptr EFECTOS[SI],ax; y muevo mi dato a efectos
            inc si ;incremento mi puntero Si en 2 ya
            inc si      ;Que se mueve 2bytes un word
            inc di      ;incremento mi puntero interno un byte
            loop camblef;Hago lo mismo para los siguiente 3 bytes
            ret
transforma endp
```

Quedando de esta forma

```
      EFECTOS     DW 1111111100011000b
                  DW 1110011100100100b
                  DW 1100001101000010b
                  DW 1000000110000001b
```

Donde si podemos mandarlos a AX y luego ir mandando parte ALTA y BAJA segun se vio en un principio en la rutina mostrando

```
;Esta rutina inicializa los cambios; es decir, regresa a Bloques
;A su estado inicial copiando los 16 primeros bytes de efectostotales
;A bloques es decir copia E1, E2, E3, E4 a B1, B2 B3 B4.
inicializar proc near
            pusha
            lea di,Bloques
            lea si,EFECTOSTOTALES
            mov cx,16
```

```
inicial:    mov al,byte ptr[si]
            mov byte ptr[di],al
            inc di
            inc si
            loop inicial
            call transforma
            popa
            ret
inicializar endp
end
```

Bueno Carlos espero que con esta explicación no tengas problemas en entender el código

yoelocmin@homail.com
http://yoelocmin.tk
http://proyectos-fie.tk