UNIVERSIDAD AUTONOMA DE AGUASCALIENTES Departamento de Sistemas Electrónicos	**Centro de Ciencias Básicas** **Sistemas Digitales 2** **Prof. Gerardo Leyva H.** **Noviembre de 2017**

Sistemas Digitales 2

Unidad 5 y 6: Ejemplos del uC8051

1. Blink

```
;blink puerto p1.0

    Org 0H
    Mov sp, #50h
Ciclo:
    setb    p1.0
    acall   delay25k
    Clr     p1.0
    aCall delay25k
    jmp ciclo

Delay25k:
    Mov     r7,#07h
Etq2:
    Mov r6,#0ffh
Etq1:
    Dec     r6
    Cjne r6, #0, etq1
    Dec     r7
    Cjne    r7,#0, etq2
    ret
    END
```

2. Luces del Auto Fantástico

```
;luces del auto fantastico
      ORG 0H
      MOV    SP,#50H

INICIO:
      MOV A,#80H
      MOV R0,#7
OTRO:
      MOV P1,A
      RR A
      DJNZ R0,OTRO
      MOV R0,#7
SIGUE:
      MOV P1,A
      RL A
      DJNZ R0,SIGUE
      SJMP OTRO

      End
```

3. Puerto paralelo mapeado en RAM externa

```
;contador en ASM para 8051 con valor duplicado en P1 y en ;dir externa de RAM FFFFh
;8051 arq de 8bits. DPTR es un registro de 16bits
;dptr -> dph & dpl en donde dph y dpl son reg de 8 bits
    Org     0h
    Mov         sp,#50h
    Sjmp    inicio

    Org     32h                         ;directiva de compilador
Inicio:
    Mov         acc, #0h                ; clr a
    Mov         dph, #0ffh      ;dptr <- ffffh
    Mov         dpl, #0ffh
Ciclo:
    Mov         p1, acc         ;p1 <- acc
    Movx    @dptr, acc      ;[dptr] <- acc
    Inc         acc
    Call        delay
    Call        delay
    Sjmp    ciclo

Delay:
    Mov         r7,#0ffh                ; r7 <- #ffh
Etq2:
    Mov     r6,#0ffh
Etq1:
    Dec         r6                      ;decremento
    Cjnz        r6, etq1                ;comp y salta si diferente de 0
    Dec         r7
    Cjnz        r7, etq2
    Ret                                 ;return
    END
```

4. Control de Motor de Pasos

;Diseñe el HW y SW de un sistema con uC8051 que controle el giro de un motor de pasos de
;4 fases con las siguientes entradas:
;Encendido/apagado en el puerto p3.2
;Derecha/izquierda en el puerto p3.3
;Salidas:
;Puerto P1

```
ORG     0H
MOV     SP,#50H
LEER_SW
MOV     C, P3.3         ;MOVIMIENTO DE BIT
JC      GIRA_DER
GIRA_IZQ:
    MOV   A,#0F7H           ;A <= 11110111B
MOV     R7,#4H
IZQ1:
MOV     P1,A            ;P1 <= A
RR      A               ;Rotacion DER
DJNZ    R7, IZQ1
SJMP    LEER_SW

GIRA_DER:
    MOV   A,#0FEH           ;A <= FEH
MOV     R7,#4H
DER1:
MOV     P1,A            ;P1 <= A
RL      A               ;Rotacion izq
DJNZ    R7, DER1
SJMP    LEER_SW

END
```

5. Suma números en RAM externa

```
;SUMADOR DE 4 BYTES EN MEMORIA RAM
;PTR1       0100H, (0103 MSB)
;PTR2       0110H, (0103 MSB)
;PTR3       0120H, (0103 MSB)

PTR1       EQU          0100H
PTR2       EQU          0110H
PTR3       EQU          0120H

RAMI       EQU          20H
NUMDIG     EQU          04H

     ORG   0H
     MOV          SP, #50H
     CLR          C
     ACALL PREP_PTR
     MOV          R7,#NUMDIG

CICLO:
     MOV          R0,#RAMI
     ACALL RD_RAM_EX
     MOV          B,A
     MOV          R0,#RAMI
     ACALL WR_PTR

     MOV          R0,#RAMI+2H
     ACALL RD_RAM_EX
     ADDC  A,B
     MOV          R0,#RAMI+2H
     ACALL WR_PTR

     MOV          R0,#RAMI+4H
     ACALL WR_RAM_EX
     MOV          R0,#RAMI+4H
     ACALL WR_PTR

     DJNZ  R7, CICLO

     SJMP  $
```

```
;=========================================================
; RESPALDA LOS APUNTADORES EN RAM INTERNA
PREP_PTR:
    MOV         R0,#RAMI
    MOV   DPTR,#PTR1
    ACALL WR_PTR
    INC R0

    MOV   DPTR,#PTR2
    ACALL WR_PTR
    INC R0

    MOV   DPTR,#PTR3
    ACALL WR_PTR
    RET

;=========================================================
; ESCRIBE DPTR EN @R0 DE RAM INTERNA
; ENTRADAS: R0, DPTR
WR_PTR:
    MOV   @R0, DPL
    INC    R0
    MOV @R0, DPH
    RET


;=========================================================
; LEER EL DATO DE @DPTR
; ENTRADAS: R0, DPTR
; SALIDA:  A
RD_RAM_EX:
    MOV   DPL, @R0
    INC    R0
    MOV   DPH, @R0
    MOVX  A, @DPTR
    INC    DPTR
    RET
```

```
;=========================================================
; ESCRIBIR EL DATO EN  @DPTR
; ENTRADAS: R0, A
WR_RAM_EX:
     MOV          DPL, @R0
     INC          R0
     MOV          DPH, @R0
     MOVX  @DPTR, A
     INC          DPTR
     RET

;=========================================================
     END
```

6. Lee datos almacenados en ROM

```
;caso 1: lectura de una cadena de caracteres que termina en 0h
;caso 2: localizar un dato en una tabla dependiendo del valor de A

    ORG         0H
    MOV         SP, #50H

;LEER UN MENSAJE DE LA ROM
    MOV         DPTR,#0040H
CICLO:
    MOV         A, #0
    MOVC  A,@A+DPTR
    INC         DPTR
    CJNE   A, #0, CICLO


;LEER UN VALOR DE UNA TABLA
ETQ1:
    MOV         A,P1
    ANL         A,#0FH
    MOV         DPTR, #0050H
    MOVC  A,@A+DPTR
    SJMP   ETQ1


;================================================;DATOS
ALMACENADOS EN ROM
;================================================
    ORG         40H
DB  "HOLA",0

    ORG         50H
DB  0H, 2H, 4H, 6H, 8H, 0AH, 0CH, 0EH, 10H, 12H, 14H, 16H,18H, 1AH, 1CH, 1EH, 0,0,0,0

    END
```

7. Contador BCD en el puerto serie

a. Introducción al problema

baud rates comunes

9600, 19200, 38499, 57600, 115200

```
MOV   SBUF, A      ;copia el acc al buffer serie (Tx)
MOV   A, SBUF      ;copia el buffer serie al acc (Rx)
```

Manipulación del contenido de B para convertir cada uno de sus dígitos en ASCII
B= 43H

	ACC	
MOV A,B	43H	0100_0011
ANL A, #F0H	40H	0100_0000
RRC A	20H	0010_0000
RRC A	10	0001_0000
RRC A	08	0000_1000
RRC A	04	0000_0100
ADD A,#30H	34H	ASCII 4H
MOV A,B	43H	0100_0011
ANL A,#0FH	03H	0000_0011
ADD A,#30H	33H	ASCII 3H

```
ANL A,#3C          ;ACC = 43
ACC 0100_0011
CTE 0011_1100
AND 0000_0011
```

b. Programa

```
;envia un contador bcd al puerto serie
;9600 baudios
```

```
        org     00h
        mov     sp,#50h
        ACALL INI_SERIE
        mov     b, #0h          ;este es el contador
CICLO:
        mov     a, b
        add     a,#1
                                ;#1b = #1h =1(decimal por omision)
        da      a               ;ajuste decimal (conversion a BCD)
        MOV     B,A
        ANL A,#0F0H             ;GUARDA EL NUEVO VALOR DE CONTEO
        RRC     A
        RRC     A
        RRC     A
        RRC     A
        ADD     A,#30H          ;CONVIERTE ASCII
        ACALL TX_SERIE
        MOV     A,B
        ANL     A,#0FH
        ADD     A,#30H          ;CONVIERTE ASCII
        ACALL TX_SERIE
        ;ASCII DEL ESPACIO
        MOV     A,#20H
        ACALL TX_SERIE
        SJMP    CICLO

INI_SERIE:
        MOV SCON,#50h ;UART modo 1 (8 bits, sin paridad)
        MOV TMOD,#21h ;Timer1 en modo 2
        MOV TH1,#0FDh ;9600 baudios (F3 en XTAL=12 MHz.)
        MOV TL1,#0FDh ;Autorecarga en el TIMER 1
        SETB TR1 ;Pone en marcha el TIMER 1
        RET

TX_SERIE:
        MOV SBUF,A ;Pone el dato en puerto serie
        JNB TI , $ ;Espera a que se envíe el dato
        CLR TI ;Desactiva la interrupción del puerto
        RET
        END
```

8. Timer 20 mS

```
;ejemplo de temporización 20ms
;COMPLEMENTA P1.7
; Ejemplo: Implementamos un contador de 0 hasta 49999
;       cual es el valor hexadecimal de 50000? r= C350H
;       CALCULAMOS FFFFH - C350H para color en los registros del timer1. R=3CAF
;       TH1 = 3C y TL1= AF

        Org     0h
        Mov     sp,#50h
        MOV     TMOD,#1H
        MOV     TL0, #E0H
        MOV     TH0, #B1h
        SETB    TR0                     ;HABILITA TIMER 0

CICLO:
        JNB     TF0, $                  ;JUMP IF NO BIT
        MOV     TL0, #E0H
        MOV     TH0, #B1h
        CLR             TF0
        CPL             P1.0                    ;ACCION DESEADA
        SJMP    CICLO

        END
```

9. RTC (real time clock)

```
;RTC (real time clock)
;En este ejemplo se construye un reloj en formato HH:MM:SS que estará alojado en las direcciones de RAM 20:21:22, respectivamente
;El ajuste se realiza por dos entradas (p3.7 y p3.6) para los minutos y las horas, respectivamente.
;Un pulso en cada entrada aumenta el valor de su registro.

hrs     equ     20h
min     equ     21h
segun   equ     22h
pMIN    EQU     P3.7
pHR     EQU     P3.6

    ORG     0H
    MOV SP,#50H
    ACALL INI_TIMER
    mov     b, #14h                         ;b cuenta 20 veces 50 milisegundos
OTRO:
    ACALL ESPERA_50mS
    ACALL AJUSTA_TIMER
    ACALL LEE_ENTRADAS
    SJMP OTRO

;SUBRITUNAS ==================================

;ajusta horas, minutos y segundos
AJUSTA_TIMER:
        djnz    b, ajusta_fin
        mov     b,#14h                      ;listo para contar otra vez 20 ciclos
        mov     r0,#segun                   ;r0 = 22
        mov     a, @r0                      ;a= [22]
        add     a,#1
                        ;#1b = #1h =1(decimal por omision)
        da      a               ;ajuste decimal (conversion a BCD)
        cjne    a,#60h, ajusta1
        clr     a
        mov     @r0,a
```

```
            acall ajusta_min
            sjmp ajusta_fin
ajusta1:
            mov    @r0,a
ajusta_fin:
            ret

ajusta_min:
            mov    r0,#min                          ;r0 = 21
            mov    a, @r0                           ;a= [21]
            add    a,#1
                                ;#1b = #1h =1(decimal por omision)
            da     a            ;ajuste decimal (conversion a BCD)
            cjne   a,#60h, min1
            clr    a
            mov    @r0,a
            acall ajusta_hrs
            sjmp   min_fin
min1:
            mov    @r0,a
min_fin:
            ret

ajusta_hrs:
            mov    r0,#hrs                          ;r0 = 20
            mov    a, @r0                           ;a= [20]
            add    a,#1
                                ;#1b = #1h =1(decimal por omision)
            da     a            ;ajuste decimal (conversion a BCD)
            cjne   a,#24h, hrs1
            clr    a
;           mov    @r0,a
hrs1:
            mov    @r0,a
            ret

INI_TIMER:
;x= ffffh- 3cafh = c350
;c350h = 50000d
    MOV   TMOD, #10H          ;TIMER 1 MODO 1 16 BITS
    MOV           TL1, #0AFH  ;PREPARA EL TIMER PARA EL VALOR DE CONTEO
```

```
        MOV         TH1, #3CH
        SETB  TR1
        RET

ESPERA_50Ms:
        JNB         TF1, $
        MOV         TL1, #0AFH   ;PREPARA EL TIMER PARA EL VALOR DE CONTEO
        MOV         TH1, #3CH
        CLR         TF1
        RET

LEE_ENTRADAS:
        JNB     pMIN, LEE_HORA
        mov     r0,#min
        MOV A, @r0          ;A <- [21h]
        add     A,#1
        DA      A
        cjne a,#60h, guarda_min
        clr     a
guarda_min:
        MOV @r0,A           ;[21h] <- A
LEE_HORA:
        JNB     pHR, FIN_LEER
        mov     r1,#hrs
        MOV A, @r1          ;A <- [21h]
        add     A,#1
        DA      A
        cjne a,#24h, guarda_hrs
        clr     a
guarda_hrs:
        MOV @r1,A           ;[21h] <- A
FIN_LEER:
        RET


        END
```

10. Interrupciones del timer

```
;TIMER 1 con Interrupcion
;Complementa p1.0
     ORG    0H
     SJMP INICIO

     ORG 001BH
     PUSH PSW
     PUSH ACC
     CALL ISR_TIMER1
     POP     ACC
     POP  PSW
     RETI

     ORG 30H
INICIO:
     ACALL INI_TIMER
     SETB   EA
     SETB   ET1
     SJMP $                ;ESPERA AQUI POR SIEMPRE

;============== SUBRUTINAS ==========================
INI_TIMER:
;x= ffffh- 3cafh = c350
;c350h = 50000d
     MOV    TMOD, #10H          ;TIMER 1 MODO 1 16 BITS
     MOV          TL1, #0AFH    ;PREPARA EL TIMER PARA EL VALOR DE CONTEO
     MOV          TH1, #3CH
     SETB   TR1
     RET

ISR_TIMER1:
     MOV          TL1, #0AFH    ;PREPARA EL TIMER PARA EL VALOR DE CONTEO
     MOV          TH1, #3CH
     CLR          TF1
     CPL          P1.0
     MOV          A,#0A5H
     SETB   C
     RET
     END
```

11. Interrupciónes externas

```
;Interrupción EXTERNA 1 (baja activa)
;Complementa el p1.1

     ORG    0H
     SJMP INICIO

     ORG 0013H
     PUSH PSW
     PUSH ACC
     CALL ISR_ext1
     POP      ACC
     POP  PSW
     RETI

     ORG 30H

INICIO:
     ACALL INI_ext1
     SETB   EA
     SJMP $                ;ESPERA AQUI POR SIEMPRE

INI_ext1:
     CLR            IE1
     SETB   EX1
     SETB   IT1
     RET

ISR_ext1:
     CLR    IE1
     CPL    P1.1
     MOV            A,#03CH
     RET


     END
```

Definiciones

CARRY. BANDERA DE ACARREO DE UN BIT ALTERADA POR LAS OPERACIONES LOGICAS-ARITMETICAS. MANIPULABLE POR BIT. ESTA DENTRO DEL PSW

PSW. PROGRAM STATUS WORD (BANDERAS DE ESTADO DE LA OPERACIÓN DEL PROCESADOR) C, CY, Z, V…

REGISTROS R0 A R7 DE PROPÓSITO GENERAL, OCUPAN LA DIR DE RAM DE 00H A 07H

Instrucciones

ACC.	REGISTRO ACUMULADOR (8 BITS). SIRVE PA'TODO.
NOP	No operation
RR	RIGTH ROTATE
JC	JUMP IF CARRY
JNC	JUMP IF NOT CARRY
MOVC A,@A+DPTR	INSTR PARA RECUPERAR ;DATOS DE LA MEMORIA DE PROGRAMA
ANL	AND LOGIC OF ACC AND CONSTANT. A = A AND CTE
DA	DECIMAL ADJUSTMENT (CONVIERTE A BCD)
ACALL	LLAMADA ABSOLUTA DE UNA SUBRUTINA
SETB	Pone a 1 lógico el bit especificado
JMP	Salta a una dirección de memoria de programa
CJNE	Compara y salta si no es igual un registro con otro registro o una constante
DJNZ	Decrementa un registro y salta a una dirección sino ha llegado a cero

Conclusiones

En este documento se presentan ejemplos de programación del microcontrolador MCS51.

Espero que haya sido de utilidad. Sugerencias al correo Gerardo.leyva.h@gmail.com