Programación de Microcontroladores (Ciclo 1 - 2025)

Proyecto 1: Reloj

Andrés Barrientos – 23288

Explicación del código (bloques, rutinas e interrupciones importantes).

Link a video de YouTube: https://youtu.be/yivhhOX2_rQ

```
.def    unitdia = R0
.def    decdia = R1
.def    unitmes = R2
.def    decmes = R3
.def    Aunitmin = R4
.def    Adecmin = R5
.def    Aunithora = R6
.def    Adechora = R7
.def    contador500ms = R18
.def    contador1s = R19
.def    estado = R20
.def    unitseg = R21
.def    decseg = R22
.def    unitmin = R23
.def    decmin = R24
.def    unithora = R25
//  dechora = R26
```

En esta parte del código lo que se hace es asignar nombres a ciertos registros de propósito general para mantener la consistencia a lo largo del programa. R26 fue el único que se utilizó como registro como tal en el código ya que éste tiene otras funciones internas, lo que dificulta el asignarle un nombre directo.

```
.org 0x00
    jmp MAIN

.org 0x0006
    jmp ISR_PCINT0

.org 0x0020
    jmp ISR_TIMER0_OVF

MAIN:
    LDI R16, LOW(RAMEND)
    OUT SPL, R16
    LDI R17, HIGH(RAMEND)
    OUT SPH, R17

TABLA: .DB 0xC0, 0xF9, 0xA4, 0xB0, 0x99, 0x92, 0x82, 0xF8, 0x80, 0x90
```

Este fragmento inicia definiendo las direcciones de memoria para el inicio del programa, interrupciones por cambio de pin y desbordamiento en el TIMER0. Luego, en **MAIN**, se inicializa el puntero de la pila para que apunte al final de la RAM, lo cual es necesario para manejar llamadas a subrutinas e interrupciones. Abajo se muestra la tabla de los valores a mostrarse en el display de 7 segmentos.

```
; INTERRUPCIONES POR PULSADORES
            ; PB2           PB1           PB0
LDI R16, (1 << PCINT2)|(1 << PCINT1)|(1 << PCINT0)
STS PCMSK0, R16

LDI R16, (1 << PCIE0)
STS PCICR, R16

SEI

CLR R27
LDI ZH, HIGH(TABLA << 1)
LDI ZL, LOW(TABLA << 1)
ADD ZL, R27
LPM R27, Z
```

Este bloque configura las interrupciones por cambio de estadi en los pines PB(0, 1, 2). Se habilita la detección de cambios mediante el registro PCMSK0. Luego, se activa la interrupción por cambio de pin (PCIE0) en el registro correspondiente. El SEI habilita las interrupciones globales para que el sistema responda a los eventos y luego se configura la tabla y el puntero Z para manejarla en la memoria del sistema.

```
LOOP:
    SBRs ESTADO, 0
    JMP EX000
    JMP EXXX1

    RJMP PULSO

//************************
// PULSO
//************************
PULSO:
    CPI CONTADOR500MS, 50
    BRNE LOOP
    CLR CONTADOR500MS

    ;DOS PUNTOS
    SBRc R8, 7
    SBI PINB, PB3
    SBRs R8, 7
    CBI PORTB, PB3

    INC CONTADOR1S
    CPI CONTADOR1S, 2
    BRNE LOOP
    CLR CONTADOR1S

    LDI R28, 9
    CPSE UNITSEG, R28
    CALL INC_UNITSEG
    CLR UNITSEG

    LDI R28, 5
    CPSE DECSEG, R28
    CALL INC_DECSEG
    CLR DECSEG

    LDI R28, 9
    CPSE UNITMIN, R28
    CALL INC_UNITMIN
    CLR UNITMIN

    LDI R28, 5
    CPSE DECMIN, R28
    CALL INC_DECMIN
    CLR DECMIN

    LDI R28, 9
    CPSE UNITHORA, R28
    CALL INC_UNITHORA
    CLR UNITHORA

    LDI R28, 2
    CPSE R26, R28
    CALL INC_DECHORA
    CLR R26

    RJMP LOOP
```

En el bucle principal se verifica el bit 0 del registro estado para ver en qué parte del código iniciará, luego ejecuta la rutina **PULSO** que actualiza los contadores del reloj, siendo cada segundo, reiniciándolos hasta alcanzar sus límites, como 59 o 24 según corresponda. Además, controla un indicador visual.

```
;;
EXXX0:
    SBRS ESTADO, 1
    JMP EXX00
    JMP EXX10
```

Este bloque declara y gestiona los estados de la FSM mediante la verificación de bits específicos en el registro **ESTADO**. Hay varios bloques de este tipo y utilizan toda la misma lógica.

```
E0000:
    LDI R28, 0b1000_0000
    MOV R8, R28
    CALL UNITMIN_DISPLAY
    CALL DECMIN_DISPLAY
    CALL UNITHORA_DISPLAY
    CALL DECHORA_DISPLAY

    SBRS R29, 0
    CBI PORTB, PB5
    SBRC R29, 0
    CALL ALARMA

    RJMP PULSO
    JMP LOOP
```

En este bloque se configura un estado en específico del programa. Estos igual se repiten a lo largo del código y utilizan la misma lógica. Al final de cada uno, siempre saltan a **PULSO** para continuar con la lógica o regresa al bucle principal.

```
ALARMA:
    CPSE ADECHORA, R26
    RET
    CPSE AUNITHORA, UNITHORA
    RET
    CPSE ADECMIN, DECMIN
    RET
    CPSE AUNITMIN, UNITMIN
    RET
    LDI ESTADO, 11
    RET
```

Este bloque implementa la lógica para comparar una alarma. Se comparan los valores en las variables del reloj con los de la alarma. Si alguna comparación no coincide, solo retorna sin hacer nada.

```
INIT_TIMER0:
    LDI R16, (1 << CS02)|(1 << CS00)    ;config prescaler 1024
    OUT TCCR0B, R16
    LDI R16, 99                         ;valor desbordamiento
    OUT TCNT0, R16                       ; valor inicial contador
    LDI R16, (1 << TOIE0)
    STS TIMSK0, R16
    RET

//************************************************************
// ISR Timer0 Overflow
//************************************************************
ISR_TIMER0_OVF:
    LDI R17, 99
    OUT TCNT0, R17
    SBI TIFR0, TOV0
    INC CONTADOR500MS

    RETI
```

Este bloque configura el Timer 0. Se establece un preescalador de 1024 y se carga un valor inicial de 99 en el contador. También se habilita la interrupción por desbordamiento. Seguido, maneja la interrupción por desbordamiento del Timer 0. Recarga el contador, limpia la bandera de desbordamiento e incrementa un contador de tiempo.

```
ISR_PCINT0:
    PUSH R16
    IN R16, SREG
    PUSH R16

    SBRs ESTADO, 0
    JMP ISR_EXX00
    JMP ISR_EXX01
```

Maneja la interrupción por cambio de pin. Guarda el estado del registro R16 y el de SREG en la pila para preservarlos. Luego, verifica el bit 0 de **ESTADO** para ver a qué parte del código saltar.

```
ISR_EXXX0:
    SBRS ESTADO, 1
    JMP ISR_EXX00
    JMP ISR_EXX10
```

Maneja una interrupción basada en el estado del sistema. Verifica el bit indicado de **ESTADO** para ver qué instrucción o rutina prosigue. Esta lógica se repite durante el código.

```
ISR_E0000:
    IN R16, PINB
    SBRS R16, PB0
    LDI ESTADO, 1
    IN R16, PINB
    SBRS R16, PB1
    LDI ESTADO, 2
    IN R16, PINB
    SBRS R16, PB2
    LDI ESTADO, 3
    JMP ISR_POP_PCINT0
```

Este fragmento maneja la interrupción del estado correspondiente. Lee los pines necesarios para cambiar al estado indicado, luego, salta a **ISR_POP_PCINT0** para finalizar la interrupción.

```
ISR_POP_PCINT0:
    SBI PCIFR, PCIF0

    POP R16
    OUT SREG, R16
    POP R16
    RETI
```

Este bloque finaliza la interrupción **PCINT0**. Limpia la bandera de interrupción para permitir nuevas interrupciones y luego restaura los registros R16 y SREG desde la pila. Finalmente, retorne de la interrupción permitiendo que el programa continúe con su ejecución normal.

```
UNITMIN_DISPLAY:
    CBI PORTD, PD2
    CBI PORTD, PD3
    CBI PORTD, PD4
    CBI PORTD, PD6
    CBI PORTD, PD7

    MOV R27, UNITMIN
    LDI ZH, HIGH(TABLA << 1)
    LDI ZL, LOW(TABLA << 1)
    ADD ZL, R27
    LPM R27, Z

    CBI PORTD, PD2
    CBI PORTD, PD3
    CBI PORTD, PD4
    CBI PORTD, PD6
    CBI PORTD, PD7

    SBRS R27, PC6
    CBI PORTB, PB4
    SBRC R27, PC6
    SBI PORTB, PB4

    OUT PORTC, R27
    SBI PORTD, PD5

    CBI PORTD, PD2
    CBI PORTD, PD3
    CBI PORTD, PD4
    CBI PORTD, PD6
    CBI PORTD, PD7

    RET
```

Este bloque actualiza la visualización de las unidades de minutos en un display de 7 segmentos. Primero, desactiva los pines del puerto D para evitar interferencias, luego carga el valor indicado en R27 y accede a la tabla para obtener el patrón de bits correspondiente al dígito. Después, configura PB4 según el bit 6 del patrón. Finalmente, envía el patrón al PORTC para mostrar el dígito en el display. Esta lógica se repite en todas las rutinas de este tipo.

```
MES_30D:
    LDI R28, 3
    CPSE DECDIA, R28
    RJMP MES_30D_2

    LDI R28, 0
    CPSE UNITDIA, R28
    CALL INC_UNITDIA_2
    LDI R28, 1
    MOV UNITDIA, R28
    CLR DECDIA
    RJMP INC_UNITMES
```

Este bloque maneja el incremento de días en meses de 30 días. Verifica si las decenas de día son 3. Si no lo son, salta a la siguiente. Luego establece las unidades de días en 1, reinicia las decenas y salta para incrementar el mes. Una lógica similar se utiliza en todos los bloques de este tipo.

```
ISR_INC_UNITMIN:
    LDI R28, 9
    CPSE UNITMIN, R28
    JMP ISR_INC_UNITMIN_2
    CLR UNITMIN
    JMP ISR_INC_DECMIN
```

Este bloque maneja el incremento de las unidades de minutos durante una interrupción. Verifica si las unidades de minutos son 9. Una lógica similar se utiliza para todos los bloques.