Universidad del Valle de Guatemala

Facultad de ingeniería

Departamento de electrónica

Programación de microcontroladores

Proyecto I
Reloj Digital

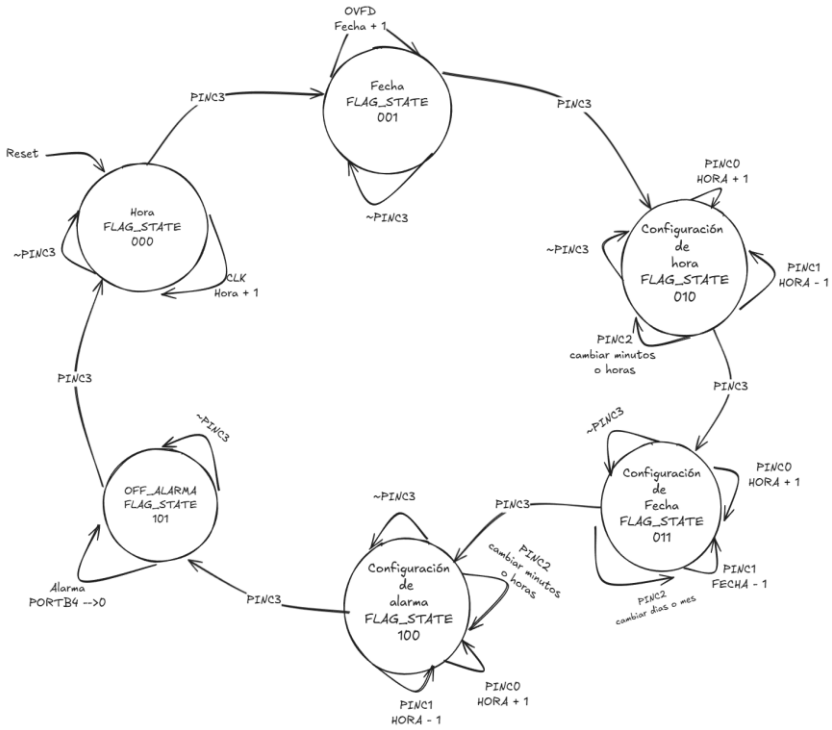
Juan Antonio Rodríguez Pineda

221593

21/03/2025

ÍNDICE

Diagrama de transiciones de estados



Estado Hora: Despliega la hora actual.

Estado Fecha: Despliega la fecha actual.

Estado configuración hora: Modifica la hora actual. Enciende un led de estado.

Estado configuración fecha: Modifica la fecha actual. Enciende un led de estado.

Estado configuración alarma: Configura la alarma en horas y minutos. Enciende un led de estado.

Estado de apagado de alarma: La alarma se apaga.

Explicación por bloques:

SETUP:

```
.include "M328PDEF.inc"
.def    SET_PB_N=R17                    //Estado de los botones
.def    CONTAD0R=R18                    //PUERTO C
.def    DISPLAY=R19                     //PUERTO D
.def    FLAG_STATE=R20                  //Bandera de Modos
.def    FLAGS_MP=R21                    //Bandera Multiproposito 0
.def    FLAGS_MP1=R22                   //Bandera Multiproposito 1
.def    LIMIT_OVF=R23                   //Contador de dias y meses
.def    DIAS=R24                        //Contador de dias y meses
.equ    T1VALUE= 64558                  //Valor inicial para la interrupcion de 60 seg
.equ    T0VALUE=11                      //Valor para interrupcion de 250 ms
.equ    T2VALUE=224                     //Valor para interrupcion de 2 ms
.dseg
```

Declaración de registros y constantes.

```
26
27   .org    SRAM_START
28   UMIN:   .byte   1
29   DMIN:   .byte   1
30   UHOR:   .byte   1
31   DHOR:   .byte   1
32   UDIAS:  .byte   1
33   DDIAS:  .byte   1
34   UMES:   .byte   1
35   DMES:   .byte   1
36   UHA:    .byte   1
37   DHA:    .byte   1
38   UMA:    .byte   1
39   DMA:    .byte   1
40
```

Uso de la ram para guardar las unidades y decenas de las horas, minutos, días y meses.

```
.cseg

.org 0x0000
    RJMP    SETUP                           //Ir a la configuraciOn al inicio

.org PCI1addr
    RJMP    ISR_PCINT1

.org OVF1addr
    RJMP    ISR_TIMER1

.org OVF0addr
    RJMP    ISR_TIMER0
```

Vectores de Interupcción.

```
        //Configuracion de pila //0x08FF
    LDI     R16, LOW(RAMEND)                    // Cargar 0xFF a R16
    OUT     SPL, R16                            // Cargar 0xFF a SPL
    LDI     R16, HIGH(RAMEND)                   //
    OUT     SPH, R16                            // Cargar 0x08 a SPH

//Configurar MCU
SETUP:
    CLI                                         //Deshabilitar interrupciones globales

    // Configurar Prescaler "Principal"
    LDI R16, (1 << CLKPCE)
    STS CLKPR, R16                              // Habilitar cambio de PRESCALER
    LDI R16, 0b00000100
    STS CLKPR, R16                              // Configurar Prescaler a 16 F_cpu = 1MHz
```

Configuración de pila y del reloj principal del ATMega328P. La frecuencia configurada es de 1MHz.

```
//Configuracion de TIMER2
LDI     R16, T2VALUE
STS     TCNT2, R16                              //Cargar el valor inic
LDI     R16, (1 << CS21) | (1 << CS20)          //Prescaler de 64
STS     TCCR2B, R16

//Configuracion de TIMER0
LDI     R16, (1<<CS01) | (1<<CS00)              //Prescaler a 64
OUT     TCCR0B, R16
//Activar las interrupciones del timer0
LDI     R16, (1<<TOIE0)
STS     TIMSK0, R16                                 //Activar las inte
LDI     R16, T0VALUE
OUT     TCNT0, R16                                  //establecer el va

//Configuracion de TIMER1
LDI     R16,  0x05                              //Prescaler a 1024
STS     TCCR1B, R16
LDI     R16, (1 << TOIE1)                       //Activar interrupcion
STS     TIMSK1, R16
//Cargar el valor inicial al timer1 para interrupci?n cada segundo
LDI     R16, HIGH(T1VALUE)
STS     TCNT1H, R16
LDI     R16, LOW(T1VALUE)
STS     TCNT1L, R16
```

Configuración de timers.

Interrupción en modo normal del timer0 (0.25 ms) y timer1 (60 s). Timer esta configurado para realizar desborde cada 2 ms (sin interrupción).

```
//Configuracion de puerto C
LDI     R16, 0x30                           //PINC0/3 entrada y PC5/4 salida
OUT     DDRC, R16
LDI     R16, 0b00001111                     //PINC0/4 pullup activados y PC5/4 conduce 0 logico
OUT     PORTC, R16

//Configuracion de puerto B
LDI     R16, 0x1F                           //Todos los pines como salida excepto PB4
OUT     DDRB, R16
LDI     R16, 0x00                           //Todos los pines conducen  logico y activar pullup PB4
OUT     PORTB, R16

//Confifuracion de puerto D
LDI     R16, 0xFF                           //Todos los pines como salida
OUT     DDRD, R16
LDI     R16, 0x00                           //Todos los pines conducen  logico
OUT     PORTD, R16

//Habilitar interrupciones en el puerto C
LDI     R16, (1<<PCIE1)                     //Setear PCIE1 en PCICR
STS     PCICR, R16
LDI     R16, 0x0F                           //Activar las interrupciones solo en los pines de botones
STS     PCMSK1, R16
```

Configuración de puertos.

```
//Deshabilitar comunicacion serial
LDI     R16, 0x00
STS     UCSR0B, R16
```

Deshabilitar la comunicación serial en los pines PD0 y PD1.

```
/*****************LOOP**********************/
MAIN:
    CPI     FLAG_STATE, 0x00
    BREQ    HORA
    CPI     FLAG_STATE, 0x01
    BREQ    FECHA
    CPI     FLAG_STATE, 0x02
    BREQ    CONFI_HORA
    CPI     FLAG_STATE, 0x03
    BREQ    CONFI_FECHA
    CPI     FLAG_STATE, 0x04
    BREQ    CONFIA
    CPI     FLAG_STATE, 0x05
    BREQ    OFFAA
    RJMP    MAIN
/******************LOOP**********************/
```

Bucle principal. Básicamente es un bucle donde el reloj se dirige hacía el estado que debe en función de las banderas de estado.

REGISTROS:

FLAGS_STATE							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
					ESTADOS		

FLAGS_STATE	R20

0b101	0b100	0b011	0b010	0b001	0b000
OFF ALARMA	CONF. ALARMA	CONF. FECHA	CONF. HORA	FECHA	HORA

El código utiliza dos registros muy importantes que indican la acción que debe realizar en función del estado y que bit de estos registros esten encendidos. Se puede decir que estos dos registros son registros de banderas como el SREG del ATMega328.

FLAGS_MP							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
FECHA	HORA	CLK	Alarma	UNIDEC	decrementar	Incrementar	OVFD

FLAGS_MP:	R21

OVFD: Se activa cuando ocurre un el reloj llega a las 00:00 es para actualizar la fecha.
Incrementar: se activa cuando se presiona el botón de incremento.
decrementar: se activa cuando se presiona el botón de decremento.
UNIDEC: cuando es 0 se van a modificar el display0 y display1, es decir, los minutos/mes, cuando es 1 se va a modificar el display2 y el display3, es decir, horas/días.
Alarma: esta HIGH cuando la alarma esta encendida, apagar en OFF ALARMA.

CLK: Se activa cada 60 segundos para incrementar las unidades del tiempo del reloj.
HORA: SI esta HIGH usas los datos de horas de la RAM. para el multiplexeo y los modos de configuración
FECHA: SI esta HIGH usas los datos de fechas de la RAM. para el multiplexeo y los modos de configuración

FLAGS_MP1:							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	ALARMA_CONF	BUCLE_CA	UDA	FLASH	FDISP23	FDISP01	FLED

FLED: Se enciende/apaga en Timer0 cada 0,5 segundos. Si esta high inicia un parpadeo de leds en los displays del centro (1 y 2).
FDISP01: Se activan en los estados de configuración cuando UNIDEC --> 0. Inicia un parpadeo en los display0 y display1
FDISP23: Se activan en los estados de configuración cuando UNIDEC --> 1. Inicia un parpadeo en los display2 y display3
FLASH: Se activa en los modos de configuración. Para permitir el parpadeo
UDA: Se activa en el modo de configuración de alarma y ahí mismo se apaga. Para permitir el uso de los datos de alarma
BUCLEA: Se activa en el modo de configuración de alarma y se apaga en las interrupciones del pin change con el botón de modos. Mientras este encendido
ALARMA_CONF: Se activa cuando se sale de la configuración de alarma. Indica que la alarma ha sido configurada.

FLAGS_STATE	R21

Lista de registros utilizados y sus funciones:

Registro	Descripción	Nombre
R12	Guardar y recargar el valor de la hora actual en el modo de alarma	NA
R13		
R14		
R15		
R16	Multipróposito	NA
R17	Estado nuevo de los botones	SET_PB_N
R18	Contar cada 25 ms	CONTAD0R
R19	Salida del displays	DISPLAY
R20	FLAG_STATE	FLAG_STATE
R21	FLAGS_MP	
R22	FLAGS_MP1	FLAGS_MP
R23	Utilizado para los límites de dias en función del mes	LIMIT_OVF
R24	Conteo de dias	DIAS
R25	Comparador de días	NA
R26	Comparador de días	NA
R27	Comparador de alarma	Sirve para saber si encender la alarma
R28	NA	NA
R29	NA	NA
R30	Puntero Z	NA
R31		NA

MODOS:

- *Hora:*

```
HORA:
    //Verificar los datos para el multiplexeo HORAS
    LDI     R16, 0x40                           //LDI   R16, (1<<HORA)
    //Encender la bandera de HORA
    SBRS    FLAGS_MP, 6
    EOR     FLAGS_MP, R16
    LDI     R16, 0x80                           //LDI   R16, (1<<FECHA)
    //Apagar la bandera de fecha
    SBRC    FLAGS_MP, 7
    EOR     FLAGS_MP, R16
    //Apagar todas las leds de estado
    SBI     PORTC, 4
    SBI     PORTC, 5
    //Actualizar reloj
    SBRC    FLAGS_MP, 5                         //Si el bit CLK esta LOW saltar
    CALL    LOGICH
    //Actualizar fecha
    SBRC    FLAGS_MP, 0
    CALL    LOGICF
    //Multiplexeo
    CALL    MULTIPLEX
    RJMP    MAIN
```

Enciendo la bandera de HORA para indicar que usare los datos de hora y minutos, a su vez apago la bandera de FECHA. Apago los leds estado. Llamo a las subrutinas para actualizar el tiempo del reloj, actualizar la fecha (de ser necesario) y desplegar las horas y minutos actuales.

- *Fecha:*

```
FECHA:
    //Verificar los datos para el multiplexeo FECHAS
    LDI     R16, 0x40                           //LDI   R16, (1<<HORA)
    //Apagar la bandera de HORA
    SBRC    FLAGS_MP, 6
    EOR     FLAGS_MP, R16
    LDI     R16, 0x80                           //LDI   R16, (1<<FECHA)
    //Encender la bandera de fecha
    SBRS    FLAGS_MP, 7
    EOR     FLAGS_MP, R16
    //Apagar todas las leds de estado
    SBI     PORTC, 4
    SBI     PORTC, 5
    //Actualizar CLK
    SBRC    FLAGS_MP, 5                         //Si el bit CLK esta LOW saltar
    CALL    LOGICH
    //Mostrar fecha
    SBRC    FLAGS_MP, 0                         //Si FLAG OVFD >> SET actualizar fecha
    CALL    LOGICF
    CALL    MULTIPLEX
    RJMP    MAIN
```

Enciendo la bandera FECHA y apago la bandera HORA para desplegar los datos

de fecha en los displays. Apago los leds de estado. Actualizo el reloj, y la fecha y despliegue la fecha actual.

- *Configuración de hora:*

```
CONFI_HORA:
    //encender la led de la hora (AZUL)
    CBI     PORTC, 4
    CBI     PORTC, 5
    //Verificar los datos para el multiplexeo HORAS
    LDI     R16, 0x40                           //LDI   R16, (1<<HORA)
    //Encender la bandera de HORA
    SBRS    FLAGS_MP, 6
    EOR     FLAGS_MP, R16
    LDI     R16, 0x80                           //LDI   R16, (1<<FECHA)
    //Apagar la bandera de fecha
    SBRC    FLAGS_MP, 7
    EOR     FLAGS_MP, R16
    //Incrementar o decrementar
    SBRC    FLAGS_MP, 1                         // Si Incrementar --> 1 incrementar
    CALL    INCREMENTAR
    SBRC    FLAGS_MP, 2                         //Si Decrementar --> 1 decrementar
    CALL    DECREMENTAR
    CALL    MULTIPLEX
    RJMP    MAIN
```

Enciendo el led del estado correspondiente. Enciendo la bandera HORA para trabajar con los registros de horas y minutos, a su vez pago la bandera de FECHA. Leo las banderas de incrementar y decrementar y realizo estas acciones si las banderas están encendidas. Por último, despliego la hora y minutos configurados.

- *Configuración de fecha:*

```
362 CONFI_FECHA:
363     //Encender la led de la fecha (VERDE)
364     SBI     PORTC, 4
365     CBI     PORTC, 5
366     //Verificar los datos para el multiplexeo FECHAS
367     LDI     R16, 0x40                       //LDI   R16, (1<<HORA)
368     //Apagar la bandera de HORA
369     SBRC    FLAGS_MP, 6
370     EOR     FLAGS_MP, R16
371     LDI     R16, 0x80                       //LDI   R16, (1<<FECHA)
372     //Encender la bandera de fecha
373     SBRS    FLAGS_MP, 7
374     EOR     FLAGS_MP, R16
375     //Incrementar o decrementar
376     SBRC    FLAGS_MP, 1                     // Si Incrementar --> 1 incrementar
377     CALL    INCREMENTAR
378     SBRC    FLAGS_MP, 2                     //Si Decrementar --> 1 decrementar
379     CALL    DECREMENTAR
380     CALL    MULTIPLEX
381     RJMP    MAIN
```

Enciendo el led del estado correspondiente. Enciendo la bandera FECHA para trabajar con los registros de meses y días, a su vez pago la bandera de HORA. Leo

las banderas de incrementar y decrementar y realizo estas acciones si las banderas están encendidas. Por último, despliego el mes y día configurados.

- *Configuración de Alarma:*

```
CONFI_ALARMA:
    //Encender la led de alarma (ROJA)
    SBI     PORTC, 5
    CBI     PORTC, 4
    //Bandera de modo configuracion de alarma
    LDI     R16, 0x30                           //LDI   R16, (1<<UDA) | (1<<BUCLEA)
    EOR     FLAGS_MP1, R16
    //Es la misma logica que en configuración de hora lo unico que no se debe modificar la hora
    //Se guardan las unidades y decenas de horas y minutos
    LDS     R16, UHOR
    MOV     R15, R16
    LDS     R16, DHOR
    MOV     R14, R16
    LDS     R16, UMIN
    MOV     R13, R16
    LDS     R16, DMIN
    MOV     R12, R16
```

Enciende el led estado correspondiente. Se encienden las banderas de UDA y BUCLEA. Se trasladan los registros de horas y minutos actuales a distintos registros.

```
//Establecemos las decenas y unidades de la alarma configurada anteriormente
LDS     R16, DMA
STS     DMIN, R16
LDS     R16, UMA
STS     UMIN, R16
LDS     R16, DHA
STS     DHOR, R16
LDS     R16, UHA
STS     UHOR, R16
//Limpiar la bandera ALARMA_CONF
LDI     R16, 0x40                               //LDI   R16, (1<<ALARMA_CONF)
SBRC    FLAGS_MP1, 6
EOR     FLAGS_MP1, R16                          //Si esta encendida apagar bandera
```

Los registros de hora y minutos actuales son reemplazados por los registros de hora y minutos de la alarma configurada (Al inicio son todos 0). Se limpia la bander de ALARMA_CONF.

```
BUCLE:
    //Verificar los datos para el multiplexeo HORAS
    LDI     R16, 0x40                           //LDI   R16, (1<<HORA)
    //Encender la bandera de HORA
    SBRS    FLAGS_MP, 6
    EOR     FLAGS_MP, R16
    LDI     R16, 0x80                           //LDI   R16, (1<<FECHA)
    //Apagar la bandera de fecha
    SBRC    FLAGS_MP, 7
    EOR     FLAGS_MP, R16
    //Incrementar o decrementar
    SBRC    FLAGS_MP, 1                         // Si Incrementar --> 1 incrementar
    CALL    INCREMENTAR
    SBRC    FLAGS_MP, 2                         //Si Decrementar --> 1 decrementar
    CALL    DECREMENTAR
    //Guardar los valores de minutos y horas en las localidades de las alarmas
    LDS     R16, DMIN
    STS     DMA, R16
    LDS     R16, UMIN
    STS     UMA, R16
    LDS     R16, UHOR
    STS     UHA, R16
    LDS     R16, DHOR
    STS     DHA, R16
    CALL    MULTIPLEX
    SBRC    FLAGS_MP1, 5                        //Sale del bucle con el botón de modo
    RJMP BUCLE
END BUCLE:
```

Inicia un bucle para configurar la alarma. Este bucle me permite usar los registros de horas y minutos actuales (debido a que las subrutinas incrementar y decrementar usan estos registros) para guardarlos en las localidades de las horas y minutos de la alarma. Sin el bucle no se puede configurar la alarma ni tampoco desplegarla en los displays. Para salir se presiona el botón de cambio de modo.

```
END_BUCLE:
    //Reestablecemos las unidades y decenas de horas y minutos las reestablecemos
    STS     DMIN, R12
    STS     UMIN, R13
    STS     DHOR, R14
    STS     UHOR, R15
    //Apagamos la bandera de UDA y encendemos ALARMA_CONF
    LDI     R16, 0x50                           // LDI  R16, (1<<UDA) | (1<<ALARMA_CONF)
    EOR     FLAGS_MP1, R16
    //Actualizar CLK
    SBRC    FLAGS_MP, 5                         //Si el bit CLK esta LOW saltar
    CALL    LOGICH
    //Actualizar fecha
    SBRC    FLAGS_MP, 0                         //Si FLAG OVFD >> SET actualizar fecha
    CALL    LOGICF
    RJMP    MAIN
```

El fin del bucle. Reestablezco lo valores de las horas y minutos actuales a estado antes de entrar al bucle. Limpio la bandera de UDA y enciendo la bandera ALARMA_CONF. Por último, se actualiza el reloj y la fecha.

```
CONFIA:
    RJMP    CONFI_ALARMA
```

Este pedazo es para poder llegar al modo de configuración de alarma. La instruccion BREQ tiene un límite y el inicio del modo estaba fuera de su rango.

- *Alarma apagada:*

```
OFFAA:
    //Verificar los datos para el multiplexeo HORAS
    LDI     R16, 0x40                           //LDI   R16, (1<<HORA)
    //Encender la bandera de HORA
    SBRS    FLAGS_MP, 6
    EOR     FLAGS_MP, R16
    LDI     R16, 0x80                           //LDI   R16, (1<<FECHA)
    //Apagar la bandera de fecha
    SBRC    FLAGS_MP, 7
    EOR     FLAGS_MP, R16
    //Parpadear la leds de estado roja
    SBI     PORTC, 5
    CBI     PORTC, 4
    //Apagar todas las leds de estado
    SBI     PORTC, 4
    SBI     PORTC, 5
    //Actualizar CLK
    SBRC    FLAGS_MP, 5                         //Si el bit CLK esta LOW saltar
    CALL    LOGICH
    //Actualizar fecha
    SBRC    FLAGS_MP, 0                         //Si FLAG OVFD >> SET actualizar fecha
    CALL    LOGICF
    //Multiplexeo
    CALL    MULTIPLEX
    SBRS    FLAGS_MP, 4
    RJMP    MAIN
    LDI     R16, 0X10                           //Apagar bandera de alarma
    EOR     FLAGS_MP, R16
    CBI     PORTB, 4                            //Apagar la alarma
    RJMP    MAIN
/*******************Modos*******************/
```

Enciendo la bandera de HORA y apago la bandera de FECHA. Enciendo la led de estado, actualizo hora y fecha. Despliego hora y si bandera Alarma está encendida se apaga alarma y bandera.

INTERRUPCIONES:

- *ISRPINCT1:*

```
ISR_PCINT1:
    //Guardar SREG y R16
    PUSH    R16
    IN      R16, SREG
    PUSH    R16
    //Progra de antirebote
    IN      SET_PB_N, PINC                          //Leer el puerto C
    //Botones de configuracion.
    LDI     R16, 0x02                               //LDI   R16, (1<<Incrementar)
    SBRS    SET_PB_N, 0                             //Si presiono el boton 0, el bit 0 esta en LOW
    EOR     FLAGS_MP, R16                           //encender la bandera de incremento
    LDI     R16, 0x04                               //LDI   R16, (1<<Decrementar)
    SBRS    SET_PB_N, 1                             //Si presiono el boton 1, el bit 1 esta en LOW
    EOR     FLAGS_MP, R16                           //Encender la bandera de decremento
    LDI     R16, 0x08                               //LDI   R16, (1<<UNIDEC)
    SBRS    SET_PB_N, 2                             //Si presiono el boton 2, el bit 2 esta en LOW
    EOR     FLAGS_MP, R16                           //Encender la bandera de UNIDEC
    //Boton de cambio de modo
    SBRS    SET_PB_N, 3
    INC     FLAG_STATE
    CPI     FLAG_STATE,0x06
    BRNE    FIN_CA
    LDI     FLAG_STATE, 0x00
FIN_CA:
    CPI     FLAG_STATE, 0x05
    BRNE    RETORNO
    LDI     R16, 0x20                               //LDI   R16, (1<<BUCLE_DA)
    EOR     FLAGS_MP1, R16                          //Salir del bucle
RETORNO:
    POP     R16
    OUT     SREG, R16
    POP     R16
    RETI
//********Rutinas de interrupcion del pin C*******
```

En esta rutina, leo el puerto la entrada del puerto C y activo las banderas correspondientes, según dicha lectura. Enciendo las banderas de INCREMENTAR, DECREMENTAR, UNIDEC y incremento las banderas estado (FLAG_STATE). A su vez si estoy en el modo de configuración de alarma y realizo un cambio de estado, apago la bandera BUCLE_DA para salir del bucle.

- *ISR_TIMER0*

```
ISR_TIMER0:
    PUSH    R16
    IN      R16, SREG
    PUSH    R16

    LDI     R16, T0VALUE
    OUT     TCNT0, R16                          //establecer el valor inicial a TCNT0 para interrumpir cada 10ms

    INC     CONTAD0R
    CPI     CONTAD0R, 2                         //Cada interrupcion es 0.25 s si contador=2 pasaron 0.5 s
    BRNE    RETORN0
    LDI     CONTAD0R, 0x00                      //Reinciar el contad0r
    LDI     R16, 0x01                           //LDI   R16, (1<<FLED)
    EOR     FLAGS_MP1, R16
RETORN0:
    POP     R16
    OUT     SREG, R16
    POP     R16
    RETI
```

Reestablezco TCNT0 para tener una interrupción consistente. Realizo un conteo

de dos interrupciones para llegar a los 500 ms. Al realizar esto se enciende la bandera de FLED.

- *ISR_TIMER1*

```
ISR_TIMER1:
    PUSH    R16
    IN      R16, SREG
    PUSH    R16
    //Reiniciar el contador del timer
    LDI     R16, HIGH(T1VALUE)
    STS     TCNT1H, R16
    LDI     R16, LOW(T1VALUE)
    STS     TCNT1L, R16

    //Activar bandera para incrementar unidades de tiempo
    LDI     R16, 0x20                       //LDI R16, (1<<CLK)
    EOR     FLAGS_MP, R16

    //Saltar solo si la alarma ha sido configurada
    SBRS    FLAGS_MP1, 6                    //Salta si Flag ALARMA_CONF --> 1
    JMP     RTIMER1                         //Si no ha sido configurada no sonara.
    //Logica de alarma
    LDS     R16, DHOR
    LDS     R27, DHA
    CP      R16, R27                        //Comparar decenas de hora
    BRNE    RTIMER1
    LDS     R16, UHOR
    LDS     R27, UHA
    CP      R16, R27                        //Comparar unidades de hora
    BRNE    RTIMER1
    //Logica de alarma
    LDS     R16, DMIN
    LDS     R27, DMA
    CP      R16, R27                        //Comparar decenas de minuto
    BRNE    RTIMER1
    LDS     R16, UMIN
    LDS     R27, UMA
    CP      R16, R27                        //Comparar unidades de minuto
    BRNE    RTIMER1
    //Si llegamos aca es porque los registros de alarma son iguales a los de la hora
    SBI     PORTB, 4                        //Encender la alarma
    LDI     R16, 0x10                       //LDI   R16, (1<<Alarma)
    EOR     FLAGS_MP, R16                   //Encender la bandera de alarma encendida
    //Retorno
RTIMER1:
    POP     R16
    OUT     SREG, R16
    POP     R16
```

Reestablecer TCNT1 para una interrupción consistente. Activo la bander CLK para incrementar el reloj y realizo una comparación entre los registros de la hora actual y los registros de la alarma pasa saber si es momento de encenderla o no.

SUBRUTINAS:

- *DELAY*

```
DELAY:
    IN      R16, TIFR2
    SBRS    R16, TOV2                       //Hasta que la bandera de overflow se active
    RJMP    DELAY                           //Se va a repetir el ciclo
    SBI     TIFR2, TOV2                     //Limpiar la bandera
    LDI     R16, T2VALUE
    STS     TCNT2, R16                      //Cargar el valor inicial
    RET
```

Función para perder 2 ms. La utilizo para realizar el multiplexeo.

- *MOVPOINTER*

```
MOV_POINTER:
    LDI     ZH, HIGH(TABLA<<1)
    LDI     ZL, LOW(TABLA<<1)
    ADD     ZL, R16                                 //Se incrementa la parte baja
    ADC     ZH, R1                                  //Se suma 0 y el carro de la parte baja
    LPM     DISPLAY, Z
    RET

MOV_POINTER2:
    LDI     ZH, HIGH(TABLA2<<1)
    LDI     ZL, LOW(TABLA2<<1)
    ADD     ZL, R16                                 //Se incrementa la parte baja
    ADC     ZH, R1                                  //Se suma 0 y el carro de la parte baja
    LPM     DISPLAY, Z
    LDI     R16, 0x08                               // LDI  DISPLAY, (1<<PT) 0x08
    SBRC    FLAGS_MP1, 0                            //Salta si FLED es 0
    EOR     DISPLAY, R16                            //Encender el punto display 2 (volteado)s*/
    OUT     PORTD, DISPLAY
    RET
```

Estas subrutinas las utilizo para cargar el valor a los displays. Utilizo dos porque tengo un display volteado. En ambas sumo a la parte baja del puntero R16, es decir, que antes de llamar la subrutina es necesario cargarle un valor a R16. En el segundo MOVPOINTER implemente la lógica para parpadear el punto del display cada que la bandera FLED este encendida.

- *MULTIPLEX*

```
517 MULTIPLEX:
518     //Unidades de minutos/MES   Display 3
519     LDS     R16, UMA                            //Mostrar las unidades de minutos de la alarma
520     SBRC    FLAGS_MP1, 4                        //Salta si UDA --> 0
521     RJMP    DISPLAY3
522     SBRC    FLAGS_MP, 6                         // HORA --> 1 usar unidades de minuto
523     LDS     R16, UMIN
524     SBRC    FLAGS_MP, 7                         // FECHA --> 1 usar unidades mes
525     LDS     R16, UMES
526 DISPLAY3:
527     CALL    MOV_POINTER
528     OUT     PORTD, DISPLAY
529     SBI     PORTB, 3
530     CALL    DELAY
531     CBI     PORTB, 3
532     //Decenas de minutos/MES Display 2
533     LDS     R16, DMA                                //Mostrar las decenas de minutos de la alarma
534     SBRC    FLAGS_MP1, 4                            //Salta si UDA --> 0
535     RJMP    DISPLAY2
536     SBRC    FLAGS_MP, 6                             // HORA --> 1 usar decenas de minuto
537     LDS     R16, DMIN
538     SBRC    FLAGS_MP, 7                             // FECHA --> 1 usar decenas mes
539     LDS     R16, DMES
540 DISPLAY2:
541     CALL    MOV_POINTER2
542     SBI     PORTB, 2
```

La lógica de los displays 0, 2 y 3 es la misma. Esta subrutina está hecha para desplegar los valores de horas/minutos o los valores de meses/días en función de las banderas FECHA y HORA. Ádemas también despliega los datos guardados en los registros de alarma.

```
DISPLAY1:
    CALL    MOV_POINTER
    //Parpadeo de punto
    LDI     R16, 0x04                       // LDI  DISPLAY, (1<<PT)
    SBRC    FLAGS_MP1, 0                    //Salta si FLED es 0
    EOR     DISPLAY, R16                    //Encender el punto display 2 (volteado)s
    OUT     PORTD, DISPLAY
    SBI     PORTB, 1
    CALL    DELAY
    CBI     PORTB, 1
    //Decenas de horas/dias Display0
    LDS     R16, DHA                        //Mostrar las decenas de horas de la alarma
    SBRC    FLAGS_MP1, 4
    RJMP    DISPLAY0                        //Saltar solo si estamos en modo config Alarma
    SBRC    FLAGS_MP, 6                     // HORA --> 1 usar decenas de hpras
    LDS     R16, DHOR
    SBRC    FLAGS_MP, 7                     // FECHA --> 1 usar decenas dias
    LDS     R16, DDIAS
```

La diferencia entre el Display1 y los otros es que este tiene implementado la lógica para indicar un parpadeo en el segmento del punto cada 500 ms.

- *INCREMENTAR*

```
INCREMENTAR:
    //Limpiar la bandera de incremento
    LDI     R16, 0x02                       //LDI   R16, (1<<Incrementar)
    EOR     FLAGS_MP, R16

    SBRC    FLAGS_MP, 3                     //UNIDEC ---> 1
    LDI     R16, 0x00                       //Trabajar con minutos/mes
    SBRS    FLAGS_MP, 3                     //UNIDEC --> 0
    LDI     R16, 0x01                       //Trabajar con horas/dias
    CPI     R16, 0x00
    BRNE    MINMES                          //Si es diferente salta a MINMES

    //Trabajar horas y dias
    SBRC    FLAGS_MP, 6                     //Si HORA --> 1 se trabajan con horas
    CALL    INCHOUR
    SBRC    FLAGS_MP, 7                     //Si Fecha -->1 se trabaja con dias
    CALL    INCDAYS
    RET
MINMES:
    //Trabajar con minutos/mes
    //Trabajar minitos y mese
    SBRC    FLAGS_MP, 6                     //Si HORA --> 1 se trabajan con min
    CALL    INCMINS
    SBRC    FLAGS_MP, 7                     //Si Fecha -->1 se trabaja con meses
    CALL    INCMES
    RET
```

Limpio la bandera de INCREMENTAR. Reviso si UNIDEC esta encendida o apagada, ya que de ella depende si incrmento horas/dias o minutos/meses. Utilizo las banderas de FECHA y HORA para indicar que registos voy a incrementar. El resto del código del bloque es la lógica empleada para incrementar horas y OVF que son similares a las detalladas en LOGICF y LOGICH

- *DECREMENTAR:*

```
DECREMENTAR:
    //Limpiar la bandera de incremento
    LDI     R16, 0x04                                   //LDI   R16, (1<<Decrementar)
    EOR     FLAGS_MP, R16

    //Configurar las de horas/dias - minutos/mes
    SBRC    FLAGS_MP, 3                                 //UNIDEC ---> 1
    LDI     R16, 0x00                                   //Trabajar con minutos/mes
    SBRS    FLAGS_MP, 3                                 //UNIDEC --> 0
    LDI     R16, 0x01                                   //Trabajar con horas/dias
    CPI     R16, 0x00
    BRNE    MINMESD                                     //Si es diferente salta a MINMES

    //Trabajar horas y dias
    SBRC    FLAGS_MP, 6                                 //Si HORA --> 1 se trabajan con horas
    CALL    DECHOUR
    SBRC    FLAGS_MP, 7                                 //Si Fecha -->1 se trabaja con dias
    CALL    DECDAYS
    RET
MINMESD:
    //Trabajar con minutos/mes
    SBRC    FLAGS_MP, 6                                 //Si HORA --> 1 se trabajan con min
    CALL    DECMINS
    SBRC    FLAGS_MP, 7                                 //Si Fecha -->1 se trabaja con meses
    CALL    DECMES
    RET

//Subrutinas de decremento
DECHOUR:
    LDS     R16, DHOR                           //Comparar si las decenas son 0
    CPI     R16, 0
    BRNE    UNDFUH                                  //Mientras se diferente a 0 el undf de la unidades es en 9
```

Limpio la bandera de decrementar. Reviso si UNIDEC esta encendida o apagada, ya que de ella depende si decremento horas/dias o minutos/meses. Utilizo las banderas de FECHA y HORA para indicar que registos voy a decrementar. El resto del código del bloque es la lógica empleada para decrementar horas y dias y los diferente UNF. Esta lógica solo cambia los límites de comparación y el valor a donde resteamos los registros. Debido al tamaño de este bloque no incluyo demasiada información.

- *LOGICH*

```
581 LOGICH:
582     //Limpiar bandera de clock
583     LDI     R16, 0x20                               //LDI   R16, (1<<CLK)
584     EOR     FLAGS_MP, R16
585     //incrementar el contador de unidades
586     LDS     R16, UMIN                           //Pasar las UMIN al contador
587     INC     R16                                 //Incrementar contador
588     STS     UMIN, R16                           //Actualizar el valor de UMIN
589     //Overflow en unidades de minuto (10 minutos)
590     CPI     R16, 10                         //ovf
591     BRNE    RETORN1
592     //Reiniciar el contador de Unidades de minutos
593     LDI     R16, 0x00
594     STS     UMIN, R16
595     //Incrementar el contador de decenas de minutos
596     LDS     R16, DMIN
597     INC     R16
598     STS     DMIN, R16
599     //Overflow en decenas de minuto
600     CPI     R16, 6
601     BRNE    RETORN1
602     //Reiniciar el contador de decenas de minutos (60)
603     LDI     R16, 0x00
604     STS     DMIN, R16
605     //Incrementar el contador de unidades de hora
606     LDS     R16, UHOR
607     INC     R16
608     STS     UHOR, R16
```

Limpio la bandera de CLK. Empiezo incrementando las unidades de minuto y cada OVF lo realizará en 10, así que cada 10 minutos incremento las decenas de minuto y cada 59 minutos incremento las unidades de hora,

```
    //El overflow de las unidades de hora dependen de las decenas de hora
    // si decenas= 1 | 0 el overflow >>> es en 9
    // si decenas=2  el overflow >>> es en 4
    LDS     R16, DHOR
    CPI     R16, 2
    BREQ    OVERF_2
    //Overflow de unidades de hora para decenas 0-1
    LDS     R16, UHOR                           //Se vuelve a cargar las unidades para comparar
    CPI     R16, 10
    BRNE    RETORN1
    LDI     R16, 0x00                           //reiniciar el contador de unidades
    STS     UHOR, R16
    //Incrementar el contador de decenas de horas
    LDS     R16, DHOR
    INC     R16
    STS     DHOR, R16
    RJMP    RETORN1
//OVF de unidades para decenas de 2
OVERF_2:
    LDS     R16, UHOR                           //se cargan las unidades para comparar
    CPI     R16, 4                              //Esta vez el limite es 4
    BRNE    RETORN1
    //Reiniciar los contadores de unidades y decenas de hora
    LDI     R16, 0x00                           //reiniciar contadores de unidades y decenas de horas
    STS     UHOR, R16
    STS     DHOR, R16
    //Encender bandera que incrementa DIAS
    LDI     R16, 0x01                               //LDI   R16, (1<<OVFD)
    EOR     FLAGS_MP, R16
RETORN1:
    RET
/**************Logica de CLK*********************/
```

El OVF de las horas tiene la peculiaridad que depende de sus decenas, si las decenas son 0 | 1 el OVF de las unidades se hace en 9 y si las decenas son 2, el OVF se realiza en 3. Por lo que realizo la comparación de decenas para determinar que límite debo utilizar. Si se realiza el OVF de las 23:59 se enciende la bandera que indica el incremento del día.

- *LOGICF*

```
/**************Logica para ovf de modo fecha***************/
LOGICF:
    //Resetear la bandera CLK
    LDI     R16, 0x01                                   //LDI   R16, (1<<OVFD)
    EOR     FLAGS_MP, R16
    //Ver que la logica a usar dependende si estamos antes de agosto o despues
    LDS     R16, UMES
    CPI     R16, 8                                      //Si es igual a 7 ir LOVF2
    BRNE    LOVF1                                       //mientra no sea igual a 7 ir LOVF1
LOVF2:
    /*De Agosto (0x07) a diciembre (0x0B) los meses de 31 dias terminan en 0
    Los de 30 terminan en 1*/
    LDI     R25, 31                                     //Se cambia la l♦gica
    LDI     R26, 32
LOVF1:
    //De enero (0x01) a Julio (0x07) los meses de 31 dias terminan en 1
    //Los de 30 terminan en 0 excepto febrero.
    SBRC    R16, 0                                      //Revisar si el mes termina en 0 o en 1
    MOV     LIMIT_OVF, R25                              // 31
    SBRS    R16, 0
    MOV     LIMIT_OVF, R26                              // 30
    CPI     R16, 2                                      //Mientras no sea febrero usar 30 o 31 como l
    BRNE    INCREMENTAR_FECHA
    LDI     LIMIT_OVF, 29                               // 1C 0001 1100
    //Incrementar dias
INCREMENTAR_FECHA:
    INC     DIAS                                        //Incrementar dias
    CP      DIAS, LIMIT_OVF                             //Comparar con el limite para el ovf
    BREQ    RESET_UD                                        //Si es distinto al limite saltar
```

Reseteo las banderas de OVFD. Para saber cuantos días tiene el mes en el que estoy utilice la siguiente tablita. Amarillos 31 días, blancos 30 días y azul febrero.

Lógica de meses		
Mes	Numero	Binario
Enero	1	1
Febrero	2	10
Marzo	3	11
Abril	4	100
Mayo	5	101
Junio	6	110
Julio	7	111
Agosto	8	1000
Septiembre	9	1001
Octubre	10	1010
Noviembre	11	1011
Diciembre	12	1100

Se puede notar que existe un patrón de Enero a Julio. El bit menos significativo termina en 1 si el mes es de 31 y en 0 si es el mes es de 30(excepto febrero). En agosto se invierte la lógica y regresando a Diciembre se reestablece.

Primero determino si hemos llegado a agosto, si esa así invertimos lógica, de lo contrario estamos con la lógica de enero a Julio. Se incrementa el registro de días y se compara con Limit_ovf que guarda los días del mes en función a la lógica mencionada.

```
INC_UD:
    //Incrementar unidades de dias
    LDS     R16, UDIAS
    INC     R16                                     //Incrementar las unidades de dias
    STS     UDIAS, R16
    CPI     R16, 10                                 //Si es distinto a 10 saltar
    BRNE    RETORNF
    //Se ejecuta unicamente cuando hay OVF en unidades de dias
    //Reiniciar las unidades e incrementar las decenas
    LDI     R16, 0x00                               //Reiniciar contador de unidades dias
    STS     UDIAS, R16                              //Guardar el contador de unidades dias
    //Incrementar decenas de dias
    LDS     R16, DDIAS
    INC     R16
    STS     DDIAS, R16
    RJMP    RETORNF
RESET_UD:
    //Reiniciar los dias, unidades y decenas de dias
    LDI     DIAS, 8
    STS     UDIAS, DIAS
    LDI     DIAS, 2
    STS     DDIAS, DIAS
    LDI     DIAS, 28
    //Incrementar mes, unidades y decenas de mes
    LDS     R16, DMES
    CPI     R16, 1                                  //Si es igual a 1
    BRNE    OVFUM                                   // No salta
    //Incrementar unidades de mes cuando decenas es 1
    LDS     R16, UMES
    INC     R16
    STS     UMES, R16
    CPI     R16, 3                                  //El overflow ocurre en 2
    BRNE    RETORNF
    //Aca pasaron todos lo meses del a♦o.
    //Resetear unidades y decenas de mes
    LDI     R16, 0x00
    STS     UMES, R16
    STS     DMES, R16
    //Se reestablece la logica a la inicial
    LDI     R25, 32
    LDI     R26, 31
OVFUM:
    //Incrementar unidades de mes cuando decenas es 0
    LDS     R16, UMES
    INC     R16
    STS     UMES, R16
    CPI     R16, 10
    BRNE    RETORNF                                 //Mientras no se igual a 10 salta
    //Resetear unidades y aumentar decenas de mes
    LDI     R16, 0x00
    STS     UMES, R16
    LDS     R16, DMES
    INC     R16
    STS     DMES, R16
RETORNF:
    RET
```

Si no es igual al límite incrementara unidades dias, decenas de días hasta llegar al límite y reiniciar el registro de los días y el contador de DIAS. Al realizar este reseteo se incrementan las unidades de mes y también las decenas (de ser el caso) hasta que estas llegan a 1 y las unidades lleguen a 2 para realizar el OVF.