Circuitos Digitales y Microcontroladores **Facultad de Ingeniería** **UNLP**	**Trabajo Práctico N°2**	**Fecha de Entrega: 23/05/2022** **Profesor Titular:** **Juárez, José María**
 	**Adrian Daniel Barral 01840/5** **Lautaro La Vecchia 02031/2**	

1. Circuito Eléctrico

Para este trabajo se utilizó el kit de desarrollo provisto por la cátedra, el mismo se ilustra en la Figura 1. Consiste de 1 display LCD de 2 líneas, 1 teclado en matricial de 4x4 y el microcontrolador ATMega328P.

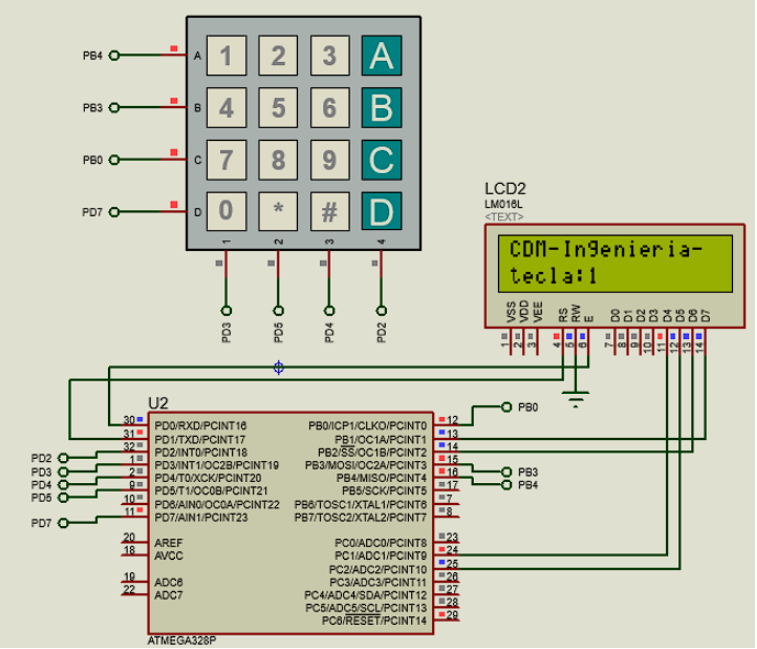

Figura 1. Esquemático del kit.

2. Programa

2.1 Problema

Implementar con el MCU un reloj con fecha y hora como el mostrado en la Figura 2. Para esto se dispone de un display LCD de 2 líneas, un teclado matricial 4x4 y el Atmega328p. La implementación deberá hacerse con máquinas de estados finitos temporizadas con Timer.

El sistema debe satisfacer los siguientes requerimientos:

A. En el estado por defecto se deberá mostrar en la primera línea del LCD un reloj funcionando con el formato HH:MM:SS (horas, minutos y segundos) y en la segunda línea la fecha en formato DD/MM/AA (día, mes y año). El valor inicial del reloj se establece en tiempo de compilación por ejemplo 23:59:59 de 31/12/21.
B. Para modificar los datos (fecha y hora) se deberá presionar ‘A’. Para cancelar y no aceptar la modificación se deberá presionar ‘D’ en cuyo caso se volverá al estado por defecto.
C. El proceso de modificación es secuencial, empezando por el año y siguiendo con el mes, el día, luego la hora, minutos y por último segundos. Para pasar de un campo al siguiente se deberá presionar ‘A’. Si es el campo segundos al presionar ‘A’ se establecerán los nuevos datos al sistema volviendo al estado por defecto.
D. Cada campo a modificar tendrá su rango de validez, por ejemplo la hora tiene un rango de 0 a 23, los minutos y segundos de 0 a 59, el año de 0 a 99, los meses de 1 a 12. Los días dependen de cada mes. Para incrementar el valor actual de un campo se deberá presionar ‘B’ y para decrementarlo se deberá presionar ‘C’.
E. Mientras transcurre la modificación de un campo, el valor del mismo deberá mostrarse parpadeando cada un segundo hasta que el usuario finalice presionando ‘A’ para pasar al campo siguiente o para finalizar, o presionando ‘D’ para cancelar.

2.2 Interpretación

El programa consistirá de:

- Un **reloj**, activado por un periférico **timer**, que se encargará de llevar la cuenta de la fecha y hora.
- Una **máquina de estado finito** que esencialmente se encargará de actualizar la pantalla LCD, leer el teclado y ejecutar la lógica de modificación de fecha y hora.

La tarea del **reloj** debe ser únicamente incrementar una variable que represente la fecha de forma periódica (cada 1 segundo). Este comportamiento facilita la implementación del ***requerimiento B*** de forma simple: permite tomar una copia de la fecha y hora actual, la cual puede ser modificada mientras se mantiene la actualización de la original, y así en caso de cancelar la modificación, no se habrá perdido información y el reloj se mantiene "en hora". Para asegurar esto último, es necesario que se eviten los retardos bloqueantes, por lo que tanto la lectura de input como la solución al efecto rebote deberán resolverse mediante otra técnica distinta a la aplicada en la entrega 1.

Podemos pensar la información del reloj como la "salida" del programa, y las teclas del teclado matricial como las "entradas". Esto nos permite diseñar una máquina de estados finita sencilla que lea el teclado, se encargue de la lógica de modificación de campos y se comunique con el display. Si además temporizamos esta MEF con nuestro periférico timer ya tendremos una solución para la consulta de teclado mediante un método no bloqueante.

Para la temporización, como se mencionó anteriormente será necesario utilizar un timer. Nuestro microcontrolador cuenta con 3 periféricos timer que podrían satisfacer nuestra necesidad; en esta ocasión, debido a que el periférico condicionará el tiempo de respuesta del teclado y la precisión de nuestro reloj, decidimos utilizar el periférico Timer1, que cuenta con una resolución de 16 bits y nos permite una mayor flexibilidad y precisión en la periodicidad.

2.3 Resolución

2.3.1 Timer1

Se utilizará el Timer1 en modo CTC (Clear Timer on Compare match). El mismo se configura para generar interrupciones cada 100 ms, un tiempo de respuesta razonable para la interfaz con el usuario (teclado).

Para obtener la mayor precisión posible se intenta utilizar el prescaler más pequeño posible, teniendo en cuenta que el Timer1 solo puede contar hasta 2^{16} y que la frecuencia de reloj con la que contamos es de 16 Mhz, en este caso el valor para el prescaler más conveniente es 64, como se puede ver en la Figura 2.

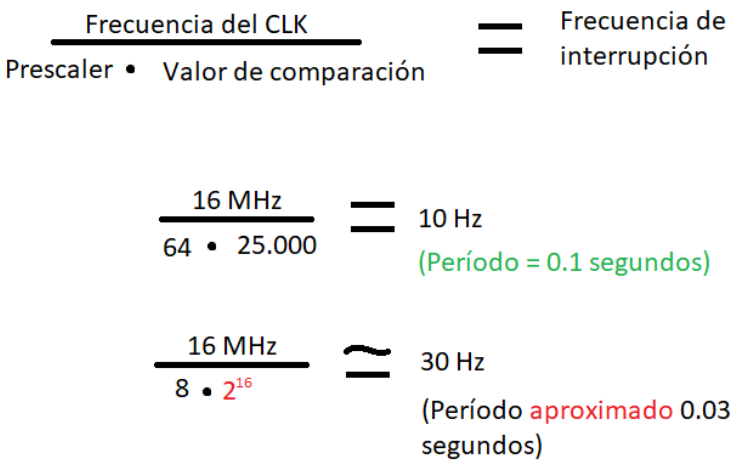

Figura 2. Cálculo de frecuencia de interrupción.

2.3.2 Reloj

El reloj puede verse como una cadena de contadores que se alimentan el uno al otro. En efecto, cada 1 segundo se incrementa en 1 el contador de segundos hasta llegar a 60, momento en el cual se reinicia a 0 y se incrementa en 1 el contador de minutos. Esto se repite en cada uno de los contadores (hora, día, mes, año), teniendo en cuenta que el límite para reiniciar cada uno de los contadores es diferente:

- Para las horas, el límite es 24
- Para los días, el límite depende del mes y también del año (debido a los años bisiestos).
- Para los meses, el límite es 13.
- Los años son el único campo que no tiene límite*

** A excepción del límite impuesto por la arquitectura del sistema, que en nuestro caso debido al tipo de dato elegido será de 2^{32}.*

2.3.3 MEF

La máquina de estados finita se utilizará para leer el teclado, ejecutar la lógica de modificación de fecha, y mostrar la fecha en el display.

El comportamiento puede modelarse con 6 estados, que se enumeran a continuación:

- MOSTRAR
- YEAR
- MNTH
- DAY
- HOUR
- MIN
- SEC

"MOSTRAR" es el estado inicial, envía al display la fecha actual para ser mostrada.

El resto de los estados permiten modificar cada uno de los campos de la fecha, enviando al display en primer lugar los campos que ya han sido modificados, parpadeando el campo que se está modificando en ese momento (asociado al estado) y para los campos que no se han modificado aún, envía el valor de "tiempo real".

Un diagrama que muestra las transiciones de estados puede verse en la Figura 3, las mismas dependen del estado actual y de las entradas. Nótese que debido a esto último, esta máquina de estados es una máquina de Mealy.

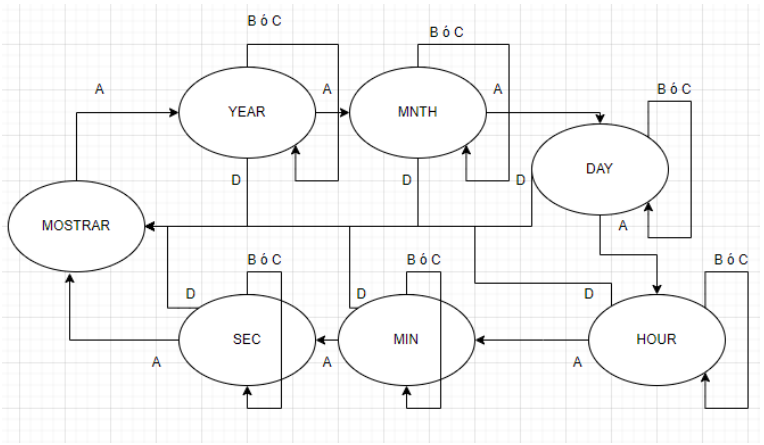

Figura 3. Diagrama de transición de estados.

2.3.4 Main loop. Planificación de tareas.

A fin de lograr el comportamiento no bloqueante, las tareas se planifican en el main loop mediante flags que son activadas por el timer. Cada 100 ms se ejecutará la MEF, leyendo el teclado y actualizando el display, mientras que el reloj actualizará la fecha cada 1 segundo.

2.3.5 Drivers

Para el display LCD se utilizó el driver puesto a disposición por la cátedra.

Para el teclado matricial fue necesario desarrollar uno, teniendo en cuenta la necesidad de evitar retardos bloqueantes al evitar rebotes de tecla.

3. Validación

3.1 Frecuencia de interrupciones

Observamos que el programa tiene un tiempo de inicialización de 0.3695335 segundos. La primera interrupción ocurre a los 100 ms, ilustrado en la figura 4 (0.4695335 segundos). Para realizar la comparación adjuntamos en la figura 5 la próxima interrupción que ocurre a los 0.5695375 segundos de ejecución, esto valida que el periodo de las interrupciones es aproximadamente 100 ms, con un error de 0.000004 que lo atribuimos al tiempo de ejecución de una iteración del main loop.

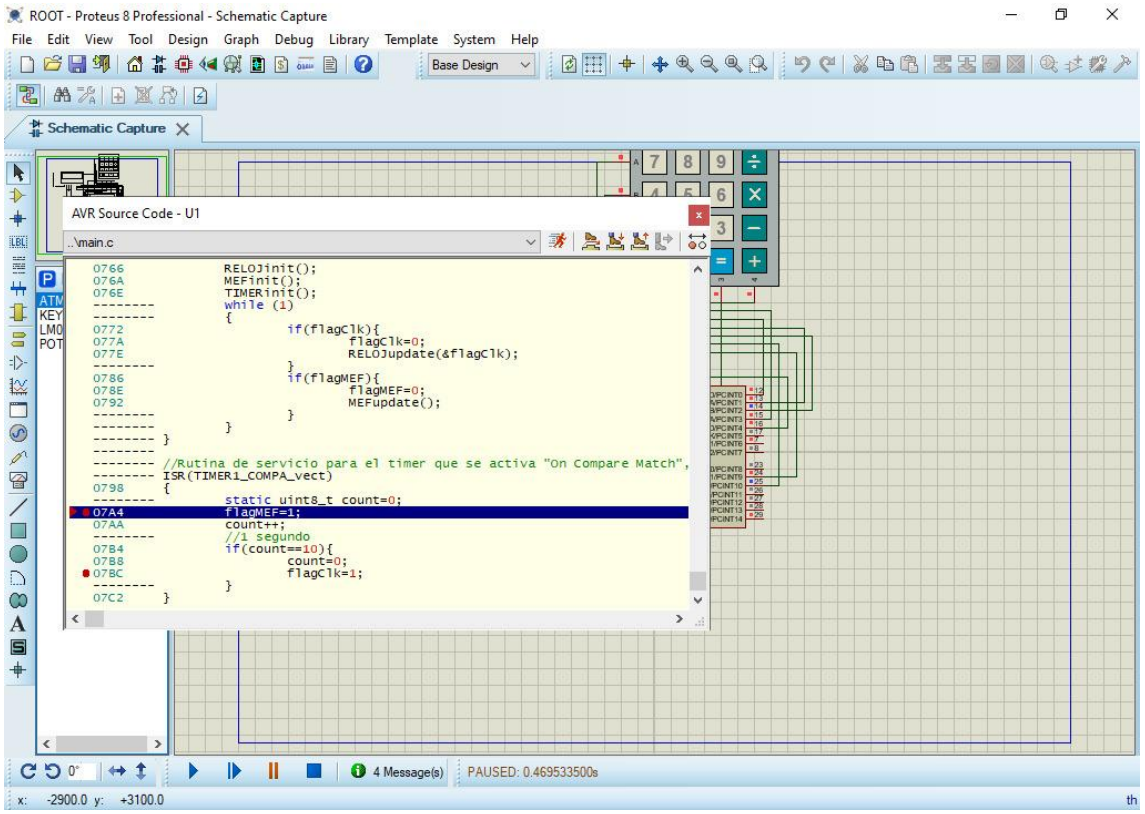

Figura 4. Primera interrupción de Timer.

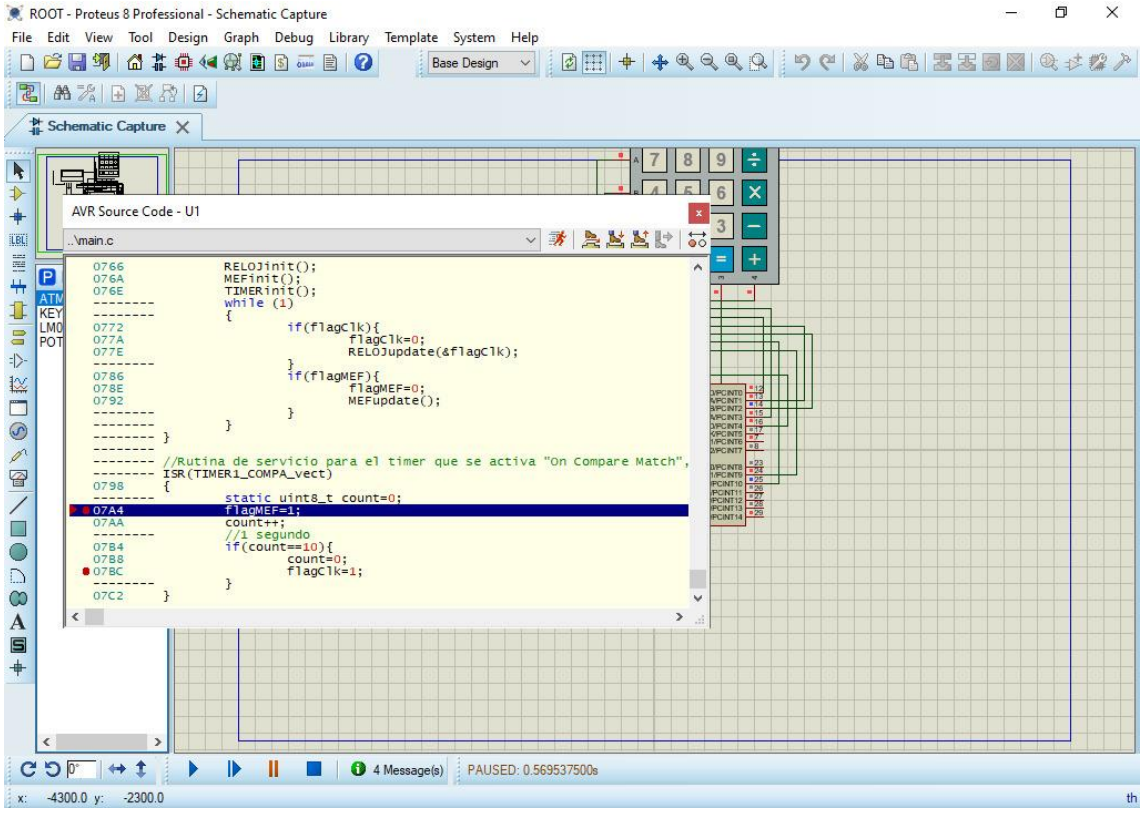

Figura 5. Segunda interrupción de Timer.

3.2 Planificación de tareas

Recordando que el tiempo de inicialización es de 0.3695335 segundos, confirmamos que el reloj es actualizado cada 1 segundo al observar la figura 6, captura del momento en que se hace verdadera la variable correspondiente a la tarea del reloj (flagClk).

$$1.369570\ segundos\ -\ 0.3695335\ segundos\ \simeq\ 1\ segundo$$

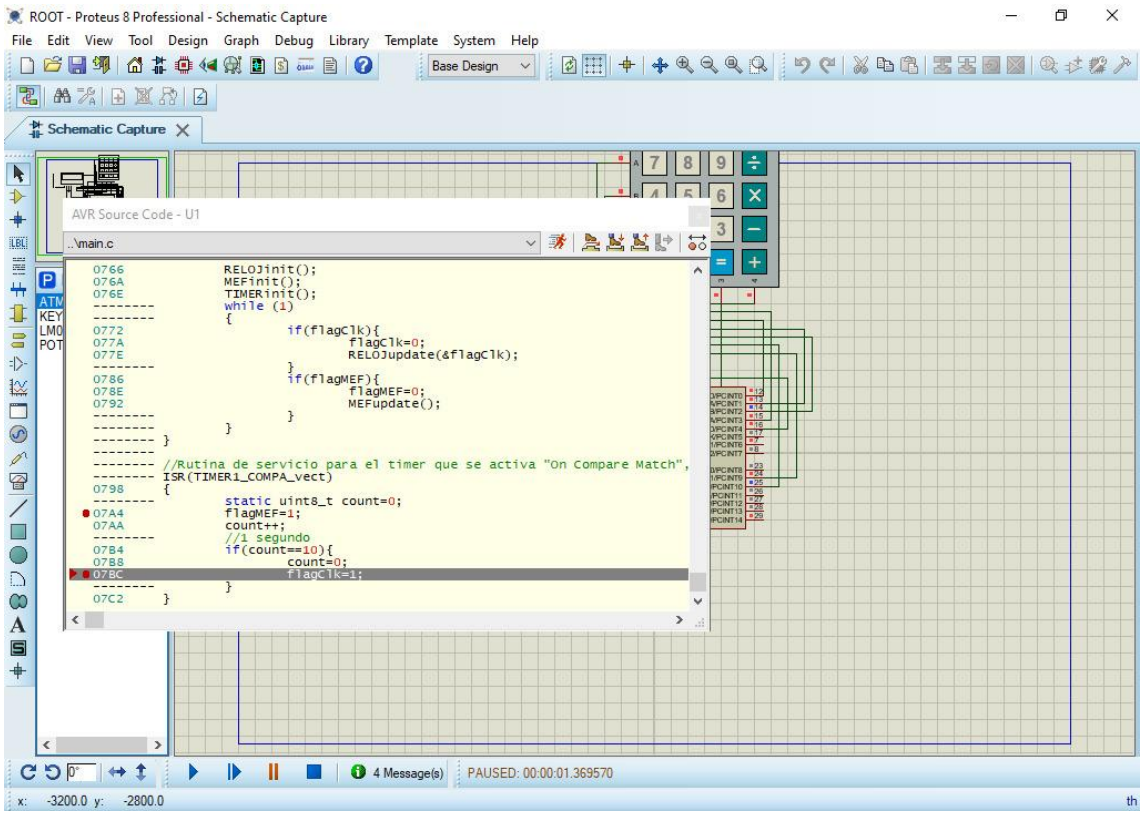

Figura 6. Primer set de flagClk.

3.3 Funcionamiento de la MEF

3.3.1 Modificando la fecha

Al presionar la tecla “A”, se ingresa al modo de modificación de los datos del reloj, comenzando por el año, ver Figura 7.

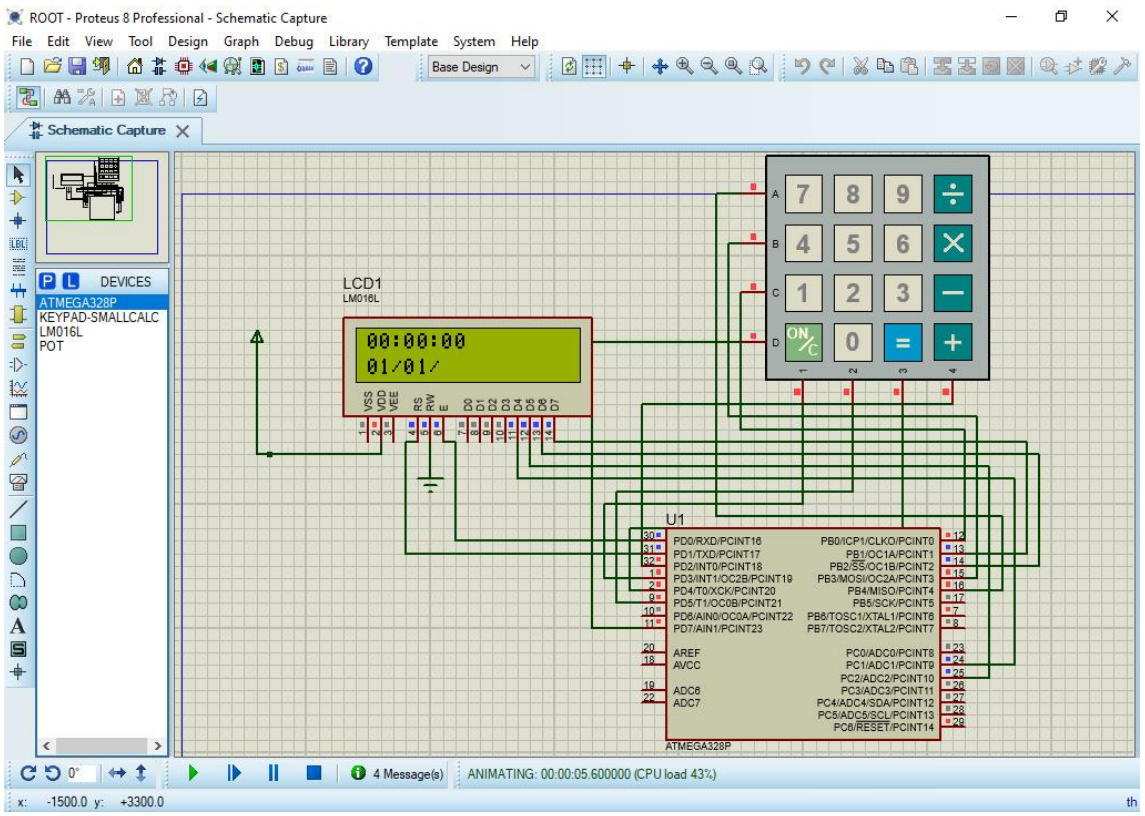

Figura 7. Modificando el campo de año.

Si se continúa la edición, confirmando cada campo con la misma tecla “A”, luego de confirmar los segundos, la fecha queda actualizada con el nuevo valor elegido. Ver Figura 8.

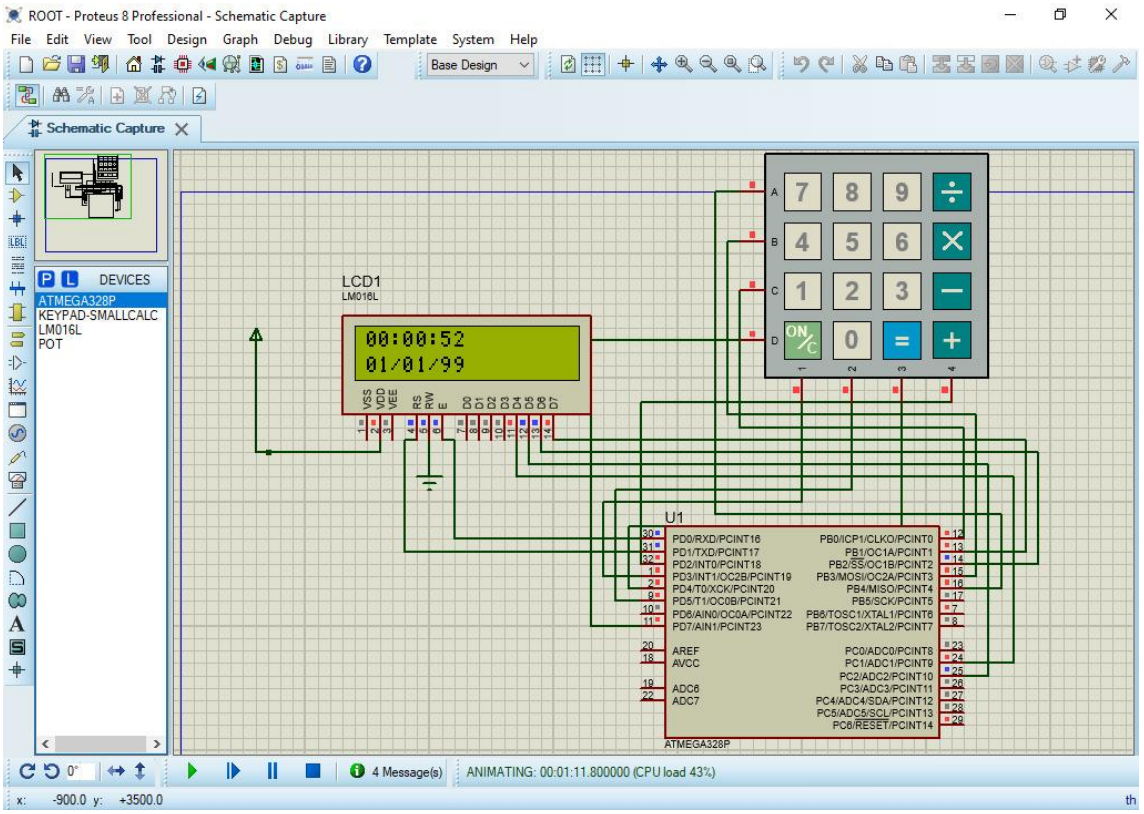

Figura 8. El año se modificó a 99. Esto es evidente ya que el tiempo de simulación transcurrido no se corresponde con el valor del reloj.

3.3.2 Descartando los cambios

Por otro lado, si se vuelve a repetir la experiencia, pero esta vez se avanza hasta los minutos y se espera que continúe el conteo (ver Figura 9), luego se presiona la tecla "D" y se cancela la edición, se observa como el reloj retorna a su valor de conteo esperado que se corresponde con el tiempo de simulación (ver Figura 10).

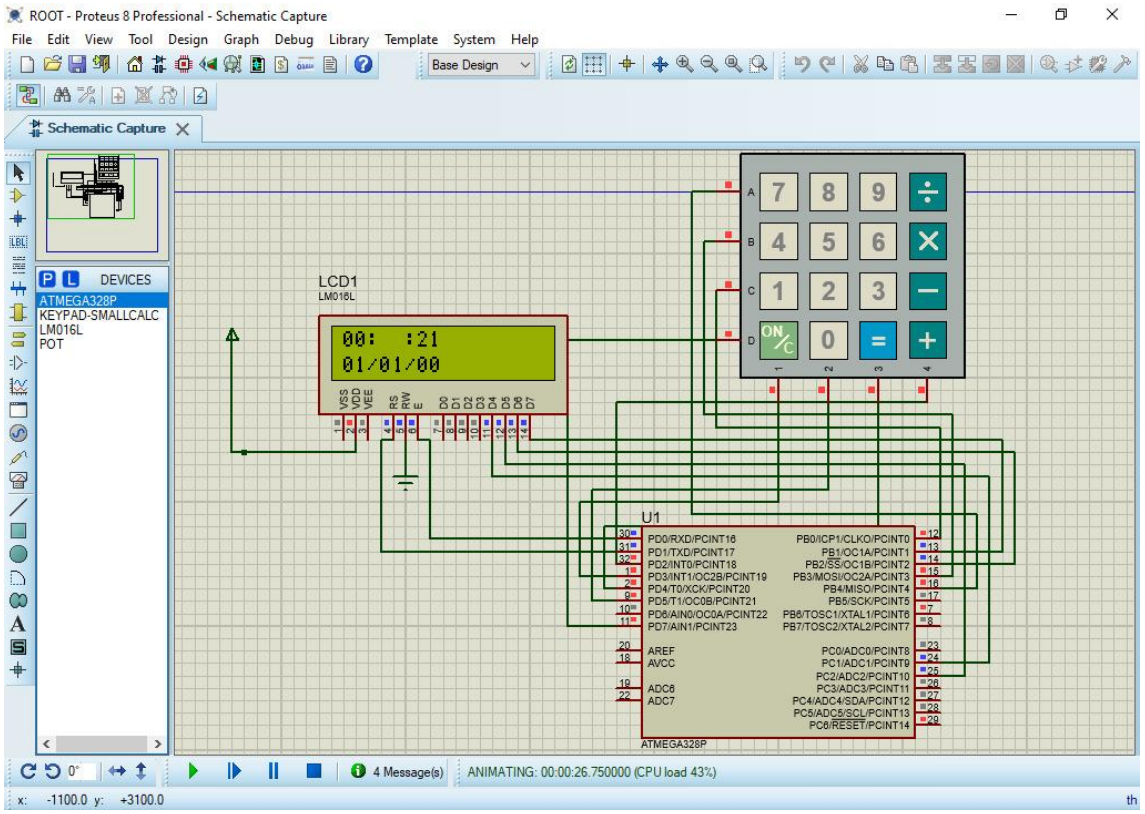

Figura 9. Modificación de los minutos.

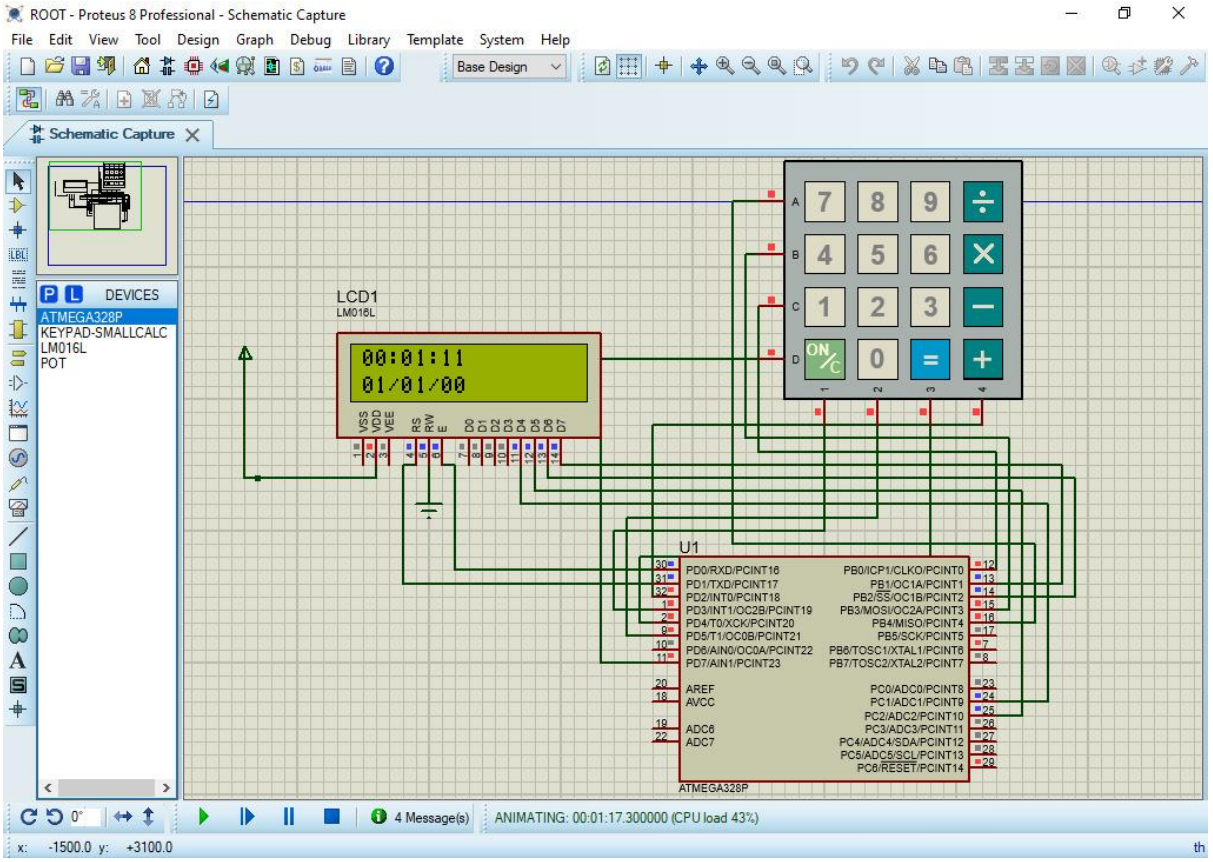

Figura 10. Cancelando el cambio.

Esto es así puesto que el conteo es independiente de la modificación. Los cambios en la modificación sólo se verán reflejados si el usuario confirma la totalidad de los campos, y el conteo avanzará independientemente del tiempo que el usuario permanezca en el modo de modificación.

4. Código

4.1 Timer

```
/*timer.h
 *
 */
#ifndef TIMER_H_
#define TIMER_H_
#include <avr/interrupt.h>
void TIMERinit();
#endif /* TIMER_H_ */
```

```c
/* timer.c
 *
 */
#include "timer.h"

/*****************************************************************
Utiliza el timer 1 de 16 bits para lograr interrupciones cada 100ms
Debe contar hasta 25.000 con un prescaler de 64 para este fin,
logrando interrupciones precisas de 100ms;
*****************************************************************/
void TIMERinit(){
    TCCR1A = 0;
    TCCR1B = 0;
    TCNT1  = 0;  //Inicializa el contador
    TCCR1B |= (1 << WGM12)|(1<<CS11)|(1 << CS10); // modo CTC, prescaler 64
    OCR1A = 0x61A8; // configura el contador de comparación para 25.000
    TIMSK1 |= (1 << OCIE1A);  //habilita interrupción por compare match
    sei(); //habilita las interrupciones para el MCU
}
```

4.2 Librería de funciones auxiliares y typedef's

```c
/* utils.h
 */
#ifndef UTILS_H_
#define UTILS_H_
#include <stdio.h>
#include <stdbool.h>
#include <stdlib.h>
/*****************************************************************/
/* Modelo de la fecha y hora. Si bien el año es mostrado con dos dígitos,
internamente se almacena el año completo para poder realizar los
cálculos para años bisiestos de manera precisa*/
/*****************************************************************/
typedef struct{
    unsigned long year;
    uint8_t mnth;
    uint8_t day;
    uint8_t hour;
    uint8_t min;
    uint8_t sec;} fecha;
uint8_t leap(uint8_t);
uint8_t maximo(uint8_t,uint8_t);
#endif /* UTILS_H_ */
```

```c
/*
 * utils.c
 *
 */

#include "utils.h"
/*********************************************************************
    Calcula los cantidad de días de febrero teniendo en cuenta años
        bisiestos, bajo la regla de que estos son:
            >divisibles por 4 y no por 100
                        o
            >divisibles por 4, por 100 y por 400
*********************************************************************/
uint8_t leap(uint8_t year){
    if (!(year%4)){
        if(year%100){
            return 29;
        }
        else if (!(year%400)){
            return 29;
        }
    }
    return 28;
}

/*********************************************************************
Calcula la máxima cantidad de días de cada mes dependiendo del año y la
retorna.

        Funciona asumiendo que solo los días pares tienen 30 días,
        luego corrige para febrero teniendo en cuenta años bisiestos.
*********************************************************************/
uint8_t maximo(uint8_t month,uint8_t year){
    if(!(month % 2)){
        if (month!=2){
            return 30;
        }else{
            return leap(year);
        }
    }
    return 31;
}
```

4.2 Reloj

```
/* reloj.h
 *
 */
#ifndef RELOJ_H_
#define RELOJ_H_
#include "utils.h"
void RELOJinit();
void RELOJupdate();
fecha RELOJgetTime();
void RELOJsetTime(fecha);
#endif /* RELOJ_H_ */
```

```
/*reloj.c
 *
 */
#include "reloj.h"
fecha actual;
/*****************************************************************/
/* Inicializa el reloj configurando una fecha en tiempo de compilación*/
/*****************************************************************/
void RELOJinit(){
    actual.day=31;
    actual.mnth=12;
    actual.year=1999;

    actual.hour=23;
    actual.min=59;
    actual.sec=55;
}
```

```
/*********************************************************************
Actualiza todos los campos de la fecha actual, teniendo en cuenta los límites
de cada rango
*********************************************************************/
void RELOJupdate(){
    if(++actual.sec == 60){
        actual.sec=0;
        if (++actual.min == 60){
            actual.min=0;
            if(++actual.hour == 24){
                actual.hour = 0;
                if(++actual.day > maximo(actual.mnth,actual.year)){
                    actual.day=1;
                    if(++actual.mnth == 13){
                        actual.year++;
                        actual.mnth=1;
                    }
                }
            }
        }
    }
}
/*********************************************************************/
/* Actualiza el valor de la fecha al valor pasado por argumento.*/
/*********************************************************************/
void RELOJsetTime(fecha new){
    actual=new;
}
/*********************************************************************/
/* Devuelve el valor actual de la fecha*/
/*********************************************************************/
fecha RELOJgetTime(){
    return actual;
}
```

4.3 Teclado

```
/* keypad.h
 *
*/
#ifndef KEYPAD_H_
#define KEYPAD_H_
#include <avr/io.h>
void KEYPADinit();
uint8_t KEYPAD_Scan(uint8_t *);
uint8_t KEYPAD_update(uint8_t *);
#endif /* KEYPAD_H_ */
```

```
/* keypad.c
 *
 */
#include "keypad.h"
#define F_CPU 16000000L
#include <util/delay.h>
//pines para las filas, de A a D
const uint8_t row[4]={4,3,0,7};
//pines para las columnas, de 1 a 4
const uint8_t col[4]={3,5,4,2};
//valores ASCII de cada tecla
const char ascii[16]= { '1','2','3','A',
    '4','5','6','B',
    '7','8','9','C',
    '0','*','#','D' };
/*********************************************************************
Esta función configura los puertos correspondientes a las filas como
salida, y los pines correspondientes a las columnas como entrada con pull up.

Inicializa las salidas de cada fila como desactivada (1)
*********************************************************************/
void KEYPADinit(){
//inicializo puertos de filas como salida, inicialmente inactivas (salida 1)
    for(int i=0;i<3;i++){
        DDRB|=1<<row[i];
        PORTB|=1<<row[i];
    }
    DDRD|=1<<row[3];
    PORTD|=1<<row[3];
```

```
//inicializo pines de columnas como entrada con pull up
    for(int i=0;i<4;i++){
        DDRD&=~(1<<col[i]);
        PORTD|=1<<col[i];
    }
}
/************************************************************
La función keypad_scan realiza polling fila por fila en el teclado
matricial, devolviendo 1 si detecta una tecla pulsada o 0 caso contrario.
Este proceso lo llamamos "escaneo".

Argumentos:

*pkey: referencia a la variable donde se almacenará la tecla que se detecte
pulsada

Nota: debido al funcionamiento del teclado, solo es posible detectar una tecla
por cada llamada a la función. Si hay más de una tecla presionada
se devuelve la que se encuentra primero teniendo en cuenta
el orden de escaneo (fila por fila, de izquierda a derecha)
************************************************************/
uint8_t KEYPAD_Scan(uint8_t *pkey) {
    uint8_t i;
    uint8_t j;

    for(i=0;i<3;i++){
        //activo la fila i para el polling
        PORTB&=~(1<<row[i]);
        for(j=0;j<4;j++){
            //busco columna activa en bajo
            if (!( PIND & (1 << col[j]) )){
                //desactivo fila i
                PORTB|=1<<row[i];
                *pkey=ascii[i*4 + j];
                return 1;
            }
        }
        //desactivo la fila i
        PORTB|=1<<row[i];
    }
```

```
    //activo la ultima fila
    PORTD&=~(1<<7);
    for(j=0;j<4;j++){
        //busco columna activa en bajo
        if (!( PIND & (1 << col[j]) )){
            PORTD|=1<<7;
            //el primer boton de la fila 4 es el indice 12 del arreglo
            *pkey=ascii[12 + j];
            return 1;
        }
    }
    //desactivo ultima fila
    PORTD|=(1<<7);
    return 0;
}
/*********************************************************************/
/*La función KEYPAD_update evita el efecto rebote de manera no bloqueante,
descartando pulsaciones dobles consecutivas de la misma tecla*/
/*********************************************************************/
uint8_t KEYPAD_update(uint8_t *pkey){
    static uint8_t Old_key;
    uint8_t Key;
    static uint8_t Last_valid_key = 0xFF;
    //lee el teclado
    if (!KEYPAD_Scan(&Key)){
        //si no se recibe una tecla se resetean los valores y se devuelve 0
        Old_key = 0xFF;
        Last_valid_key = 0xFF;
        return 0;
    }
//Chequea si la última tecla leída sigue presionada.
//Esto permite que se estabilice la señal del teclado.
    if (Key == Old_key){
        //verifica que no sea una pulsación consecutiva y la valida
        if (Key != Last_valid_key){
            *pkey = Key;
            Last_valid_key = Key;
            return 1;
        }
    }
    //Se detecta la primera pulsación de la tecla
    Old_key = Key; return 0;}
```

4.4 MEF

```
/* MEF.h
 *
 */
#ifndef MEF_H_
#define MEF_H_
#include "lcd.h"
#include "keypad.h"
#include "utils.h"
#include "reloj.h"
/*********************************************************************/
/* Estados de la máquina de estados
MOSTRAR: estado inicial, envía el valor de la fecha actual al display
YEAR: parpadea el campo correspondiente al año y permite incrementar o
decrementar
MNTH: idem con el campo correspondiente al mes
DAY: idem con el campo correspondiente al dia
HOUR: idem con el campo correspondiente a la hora
MIN: idem con el campo correspondiente a los minutos
SEC: idem con el campo correspondiente a los segundos.

En cualquiera de los estados, confirmar el valor ingresado transiciona al
siguiente.
Con excepción del estado SEC que transiciona al estado inicial y actualiza el
valor de la fecha y hora.

En cualquiera de los estados cancelar descarta los valores ingresados y
devuelve al estado inicial.

Confirmar: Tecla 'A'
Incrementar campo: Tecla 'B'
Decrementar campo: Tecla 'C'
Cancelar: Tecla 'D'
/*********************************************************************/
typedef enum{MOSTRAR,YEAR,MNTH,DAY,HOUR,MIN,SEC}state;
void MEFinit();
void MEFupdate();
void actualizarLCD();
#endif /* MEF_H_ */
```

```c
/* MEF.c
 *
 * Máquina de estado finito para modelar el funcionamiento del reloj. La MEF lee el teclado y permite modificar todos los campos de la fecha y hora.
 * Tiene en cuenta años bisiestos.
 */
#include "MEF.h"
state estado;
uint8_t pkey;
uint8_t pkey_event;
uint8_t count=10;
bool blink=false;
fecha modificada;

/*****************************************************************/
/*Inicializa la mef inicializando el display y el teclado 4x4. Setea el estado inicial*/
/*****************************************************************/
void MEFinit(){
    LCDinit();
    LCDclr();
    LCDhome();
    KEYPADinit();
    estado=MOSTRAR;
}

/*****************************************************************/
/* La actualización de la MEF consiste en leer las entradas (el teclado 4x4) y en función de los estados realizar la transición correspondiente.
/*****************************************************************/
void MEFupdate(){
//si se encuentra modificando un campo el output debe parpadear
    switch (estado){
        case MOSTRAR: count=10;
        break;
        default: count--;
            if (!count){
                 blink=!blink;
                 count=10;
            }
            break;
    }
```

```
    //lee las entradas
    if(KEYPAD_update(&pkey)){
        switch(estado){
            case MOSTRAR:
            if(pkey=='A'){
                modificada.year=RELOJgetTime().year;
                estado=YEAR;
            }
            break;
            case YEAR:
            switch(pkey){
                case 'A':
                modificada.mnth=RELOJgetTime().mnth;
                estado=MNTH;
                break;
                case 'B':
                modificada.year++;
                break;
                case 'C':
                modificada.year--;
                break;
                case 'D':
                estado=MOSTRAR;
                break;
            }
            break;
            case MNTH:
            switch(pkey){
                case 'A':
                modificada.day=RELOJgetTime().day;
                //me aseguro de que la fecha sea valida antes de confirmar
                 if (modificada.day > maximo(modificada.mnth,modificada.year))
modificada.day=maximo(modificada.mnth,modificada.year);
                estado=DAY;
                break;
                case 'B':
                if (++modificada.mnth==13) modificada.mnth=1;
                break;
                case 'C':
                if (--modificada.mnth==0) modificada.mnth=12;
                break;
                case 'D':
```

```
                estado=MOSTRAR;
                break;
            }
            break;
            case DAY:
            switch(pkey){
                case 'A':
                modificada.hour=RELOJgetTime().hour;
                estado=HOUR;
                break;
                case 'B':
                                                                        if
(++modificada.day>(maximo(modificada.mnth,modificada.year))) modificada.day=1;
                break;
                case 'C':
                                                  if   (--modificada.day==0)
modificada.day=maximo(modificada.mnth,modificada.year);
                break;
                case 'D':
                estado=MOSTRAR;
                break;
            }
            break;
            case HOUR:
            switch(pkey){
                case 'A':
                modificada.min=RELOJgetTime().min;
                estado=MIN;
                break;
                case 'B':
                if (++modificada.hour==24) modificada.hour=0;
                break;
                case 'C':
                if (--modificada.hour<0) modificada.hour=23;
                break;
                case 'D':
                estado=MOSTRAR;
                break;
            }
            break;
            case MIN:
            switch(pkey){
```

```
                case 'A':
                modificada.sec=RELOJgetTime().sec;
                estado=SEC;
                break;
                case 'B':
                if (++modificada.min==60) modificada.min=0;
                break;
                case 'C':
                if (modificada.min--==0) modificada.min=59;
                break;
                case 'D':
                estado=MOSTRAR;
                break;
            }
            break;
            case SEC:
            switch(pkey){
                case 'A':
                RELOJsetTime(modificada);
                estado=MOSTRAR;
                break;
                case 'B':
                if (++modificada.sec==60) modificada.sec=0;
                break;
                case 'C':
                if (modificada.sec--==0) modificada.sec=59;
                break;
                case 'D':
                estado=MOSTRAR;
                break;
            }
            break;
        }
    }
    //actualiza las salidas
    actualizarLCD();
}
```

```
/*****************************************************************
Actualiza la pantalla dependiendo del estado actual de la MEF.
Si el usuario se encuentra modificando algun campo, ademas de actualizarse
debe parpadear.

        Parpadea 1 vez por segundo. Esto se interpreta como:
    mostrar el campo q se esta modificando durante 1 segundo,
    apagarlo durante 1 segundo.
*****************************************************************/
void actualizarLCD(){
    LCDclr();
    LCDGotoXY(0,0);
    fecha actual=RELOJgetTime();

    switch(estado){
        case MOSTRAR:
            LCDescribeDato(actual.hour,2);
            LCDsendChar(':');
            LCDescribeDato(actual.min,2);
            LCDsendChar(':');
            LCDescribeDato(actual.sec,2);

            LCDGotoXY(0,1);

            LCDescribeDato(actual.day,2);
            LCDsendChar('/');
            LCDescribeDato(actual.mnth,2);
            LCDsendChar('/');
            LCDescribeDato(actual.year,2);
        break;
        case YEAR:
            LCDescribeDato(actual.hour,2);
            LCDsendChar(':');
            LCDescribeDato(actual.min,2);
            LCDsendChar(':');
            LCDescribeDato(actual.sec,2);

            LCDGotoXY(0,1);

            LCDescribeDato(actual.day,2);
            LCDsendChar('/');
            LCDescribeDato(actual.mnth,2);
```

```
            LCDsendChar('/');
            if(blink){
                LCDsendChar(' ');
                LCDsendChar(' ');
                }else{
                LCDescribeDato(modificada.year,2);
            }
        break;
        case MNTH:
            LCDescribeDato(actual.hour,2);
            LCDsendChar(':');
            LCDescribeDato(actual.min,2);
            LCDsendChar(':');
            LCDescribeDato(actual.sec,2);

            LCDGotoXY(0,1);

            LCDescribeDato(actual.day,2);
            LCDsendChar('/');
            if(blink){
                LCDsendChar(' ');
                LCDsendChar(' ');
                }else{
                LCDescribeDato(modificada.mnth,2);
            }
            LCDsendChar('/');
            LCDescribeDato(modificada.year,2);
        break;
        case DAY:
            LCDescribeDato(actual.hour,2);
            LCDsendChar(':');
            LCDescribeDato(actual.min,2);
            LCDsendChar(':');
            LCDescribeDato(actual.sec,2);

            LCDGotoXY(0,1);

            if(blink){
                LCDsendChar(' ');
                LCDsendChar(' ');
                }else{
                LCDescribeDato(modificada.day,2);
```

```
            }
            LCDsendChar('/');
            LCDescribeDato(modificada.mnth,2);
            LCDsendChar('/');
            LCDescribeDato(modificada.year,2);
        break;
        case HOUR:
            if(blink){
                LCDsendChar(' ');
                LCDsendChar(' ');
                }else{
                LCDescribeDato(modificada.hour,2);
            }
            LCDsendChar(':');
            LCDescribeDato(actual.min,2);
            LCDsendChar(':');
            LCDescribeDato(actual.sec,2);

            LCDGotoXY(0,1);

            LCDescribeDato(modificada.day,2);
            LCDsendChar('/');
            LCDescribeDato(modificada.mnth,2);
            LCDsendChar('/');
            LCDescribeDato(modificada.year,2);
        break;
        case MIN:
            LCDescribeDato(modificada.hour,2);
            LCDsendChar(':');
            if(blink){
                LCDsendChar(' ');
                LCDsendChar(' ');
                }else{
                LCDescribeDato(modificada.min,2);
            }
            LCDsendChar(':');
            LCDescribeDato(actual.sec,2);

            LCDGotoXY(0,1);

            LCDescribeDato(modificada.day,2);
            LCDsendChar('/');
```

```
            LCDescribeDato(modificada.mnth,2);
            LCDsendChar('/');
            LCDescribeDato(modificada.year,2);
        break;
        case SEC:
            LCDescribeDato(modificada.hour,2);
            LCDsendChar(':');
            LCDescribeDato(modificada.min,2);
            LCDsendChar(':');
            if(blink){
                LCDsendChar(' ');
                LCDsendChar(' ');
                }else{
                LCDescribeDato(modificada.sec,2);
            }


            LCDGotoXY(0,1);

            LCDescribeDato(modificada.day,2);
            LCDsendChar('/');
            LCDescribeDato(modificada.mnth,2);
            LCDsendChar('/');
            LCDescribeDato(modificada.year,2);
        break;
    }
}
```

4.5 Main Loop

```
/* main.c
 */
#include <stdio.h>
#include <stdlib.h>
#include <avr/io.h>
#include "reloj.h"
#include "timer.h"
#include "MEF.h"
volatile uint8_t flagMEF=1;
volatile uint8_t flagClk=0;
int main(void){
    RELOJinit();
    MEFinit();
    TIMERinit();
    while (1){
        if(flagClk){
            flagClk=0;
            RELOJupdate(&flagClk);
        }
        if(flagMEF){
            flagMEF=0;
            MEFupdate();
        }
    }
}
//Rutina de servicio para el timer que se activa "On Compare Match", definido
en el timer a 100ms
ISR(TIMER1_COMPA_vect){
    static uint8_t count=0;
    flagMEF=1;
    count++;
    //1 segundo
    if(count==10){
        count=0;
        flagClk=1;
    }
}
```