Fecha:20/05/2024

Circuitos Digitales y Microcontroladores

GRUPO 6

Chanquía, Joaquín (02887/7)

Forden Jones, Ian E. W. (02543/3)

Ollier, Gabriel (02958/4)

ÍNDICE

1. INTRODUCCIÓN

1.1 - Enunciado

Implementar el juego interactivo “ENCUENTRE EL NÚMERO”. Para esto se dispone de un display LCD de 2 líneas, un teclado matricial 4x4 y el MCU Atmega328p. La implementación deberá hacerse con máquina de estados temporizadas con Timer.

A continuación, se enumeran los requerimientos que debe satisfacer el sistema:

a) Cuando el equipo se inicia deberá mostrar en la primera línea del LCD un mensaje de bienvenida y en la segunda línea el texto: “P/ JUGAR PULSE A”.

b) Si el usuario presiona ‘A’ se mostrará en la primera línea “JUGANDO”, se iniciará un temporizador interno, y se generará un número aleatorio entre 0 y 99 sin mostrarlo. En la segunda línea se mostrará “INGRESE NUM”

c) El jugador deberá ingresar por teclado un número entre 0 y 99. Si está fuera de rango se le solicitará otro. Si el número ingresado coincide con el generado internamente se mostrará en la primera línea “GANADOR” y en la segunda línea se mostrará el tiempo que tardó en resolverlo, por ejemplo “TIME: 1:45:682”. Luego de 3 segundos el sistema volverá al estado inicial.

d) Si el número ingresado es menor que el generado internamente, se mostrará al lado del número el símbolo ‘>’ para indicar que debe ingresar un número más alto que el anterior. En caso contrario se mostrará ‘<’ para indicar que debe ingresar un número más bajo que el anterior.

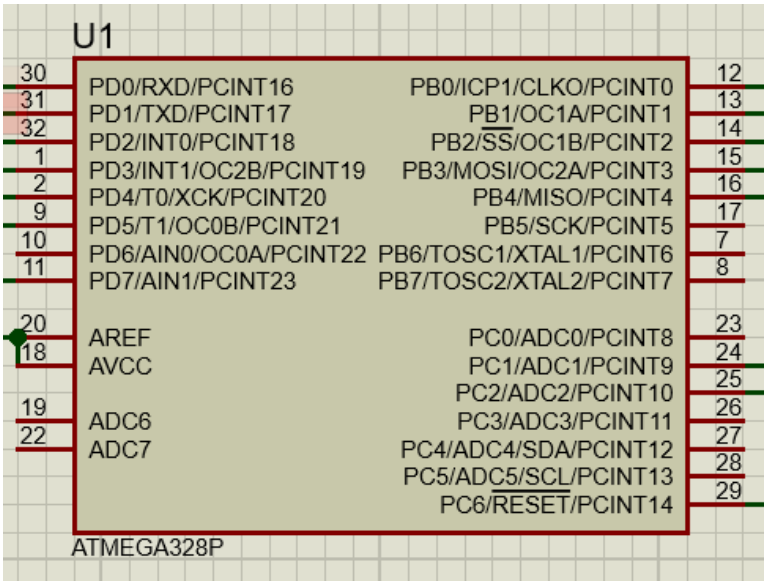
e) Por último, si el jugador quiere descartar la partida puede presionar la tecla ´D´ y volver al estado inicial.

1.2 - Interpretación

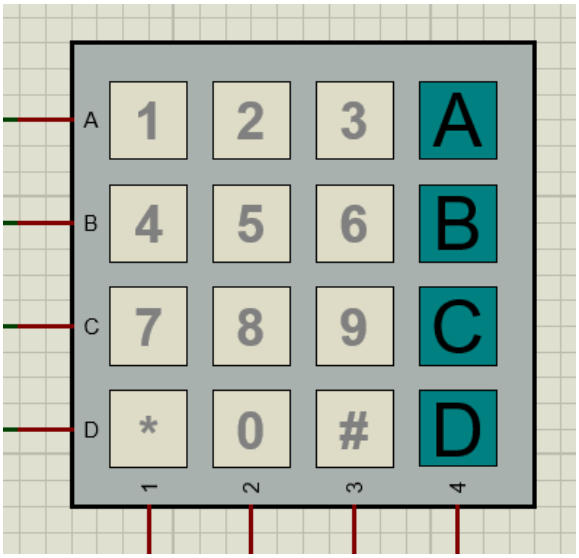
El objetivo del trabajo es crear un juego basado en adivinar un numero del 0 al 99. La interfaz con el usuario esta compuesta por una pantalla LCD y un teclado matricial. Para comenzar el juego el usuario debe presionar la tecla A. Luego de esto se le pide al usuario ingresar un digito, que aparecerá en pantalla, y al ingresar un segundo digito numérico se mostrara en pantalla si el numero es mayor, menor o igual al numero generado. En caso de ser igual se mostrará también el tiempo transcurrido entre que empezó este juego hasta que se ingresó el numero correcto, y se esperará durante 3 segundos. Este sistema no admite números fuera de rango. En cualquier momento del juego si se presiona la tecla D se volverá a la pantalla de inicio, excepto en la pantalla de juego ganado.

Componentes utilizados:

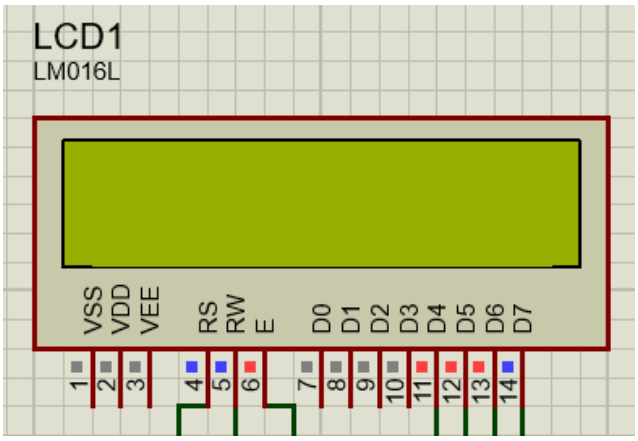
MCU Atmega328p



Teclado matricial 4x4



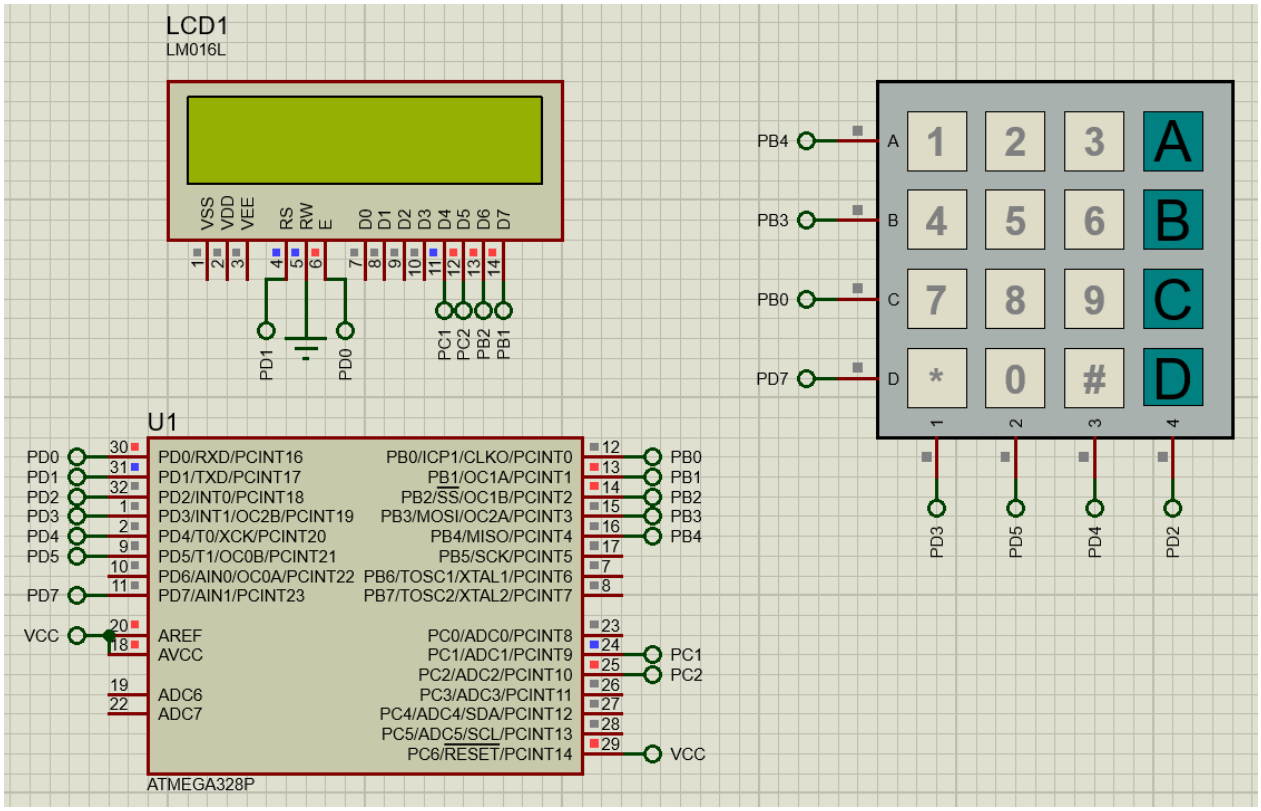
Display LCD de 2 líneas por 16 caracteres



2. RESOLUCIÓN

2.1 Funcionamiento de periféricos

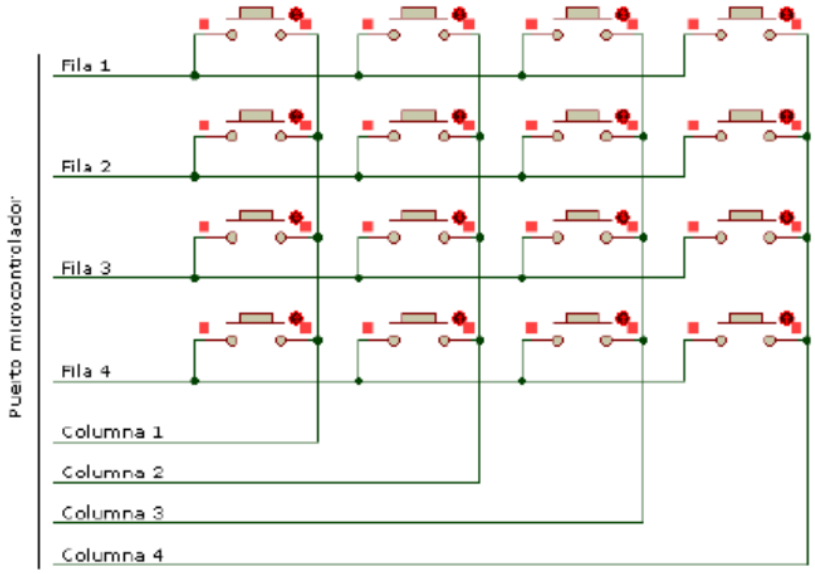
El conexionado de los periféricos se realiza siguiendo la conexión dada por la práctica, el uso de conexiones virtuales se hace para simplificar la conexión y la visualización en el Proteus.



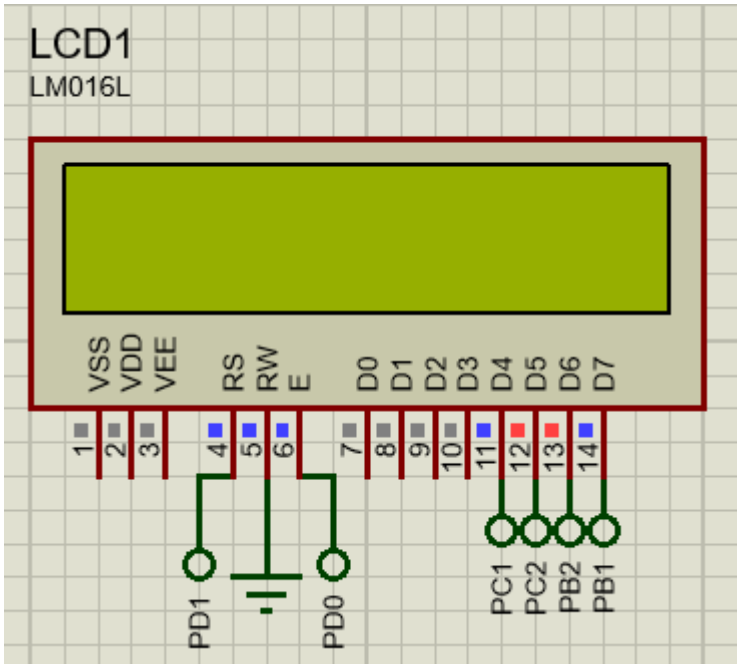
Para la lectura del teclado matricial 4x4 se conecta cada fila de este a una salida del microcontrolador y cada columna a una entrada. Para la conexión del teclado se requiere del uso de 8 pines. Dada la conexión interna del teclado al presionar una tecla se produce un cortocircuito que conecta la salida de la fila correspondiente a esta con su columna. Esto permite detectar si la salida en cuestión tiene un valor alto o bajo en el pin conectado.

Para el fin de este trabajo se activará un pullup interno en las entradas del microcontrolador por lo que no será posible detectar si fue configurado un 1 en la salida, pero si cambiará su valor si en la salida hay un 0 y se presiona la tecla.

Conexión del teclado matricial:



En cuanto al Display LCD de 2 líneas con 16 segmentos se configuran varios puertos, como queremos enviar (write) un dato al LCD, hacemos R/W=0, mientras que por el puerto E (Enable) se envía una señal de control para habilitar el acceso al LCD, tanto para leer o escribir en el mismo necesitamos aplicar un flanco descendente al pin, el RS (Register select) debe alterar entre 0 o 1 ya que el primero selecciona el data register interno, en cambio el segundo selecciona el command register interno. Para ahorrar terminales del MCU el LCD se configura para trabajar con solo 4 bits de datos en lugar de los 8bits, en este caso la conexión se realiza en los pines D4 – D7



2.2 Funcionamiento del software

Para este problema se decidió crear un total de 4 archivos de tipo ".c" y sus respectivos archivos cabecera para una mejor organización de las funciones y una posible reutilización en futuras aplicaciones. Los archivos son los siguientes:

- main.c: este archivo contiene la función principal y la lógica perteneciente a la máquina de estados.
- TimerMEF.c: Contiene funciones relacionadas con la temporización del programa.
- ScreensMEF.c: En este archivo están las funciones para la impresión de la pantalla en los distintos estados del juego.
- KeyPad.c: Tiene las funciones para la lectura del teclado matricial.

main.c

En este archivo se encuentran las funciones pertenecientes a la máquina de estados y también está el programa principal que será ejecutado. El archivo cuenta con las siguientes funciones:

Función de manejo de la lógica de la MEF y transición entre estados:

```
void Actualizar_MEF();
```

Funciones de cambio a cada estado:

```
void ChangeToSTART();
void ChangeToPLAY();
void ChangeToFIRST(uint8_t k);
void ChangeToAGAIN();
void ChangeToWIN();
```

Funciones para verificar que un carácter presionado sea un número y para comparar el resultado ingresado por el jugador y el almacenado internamente:

```
uint8_t EsNum(uint8_t k);
uint8_t Comparar(uint8_t num, uint8_t rsp);
```

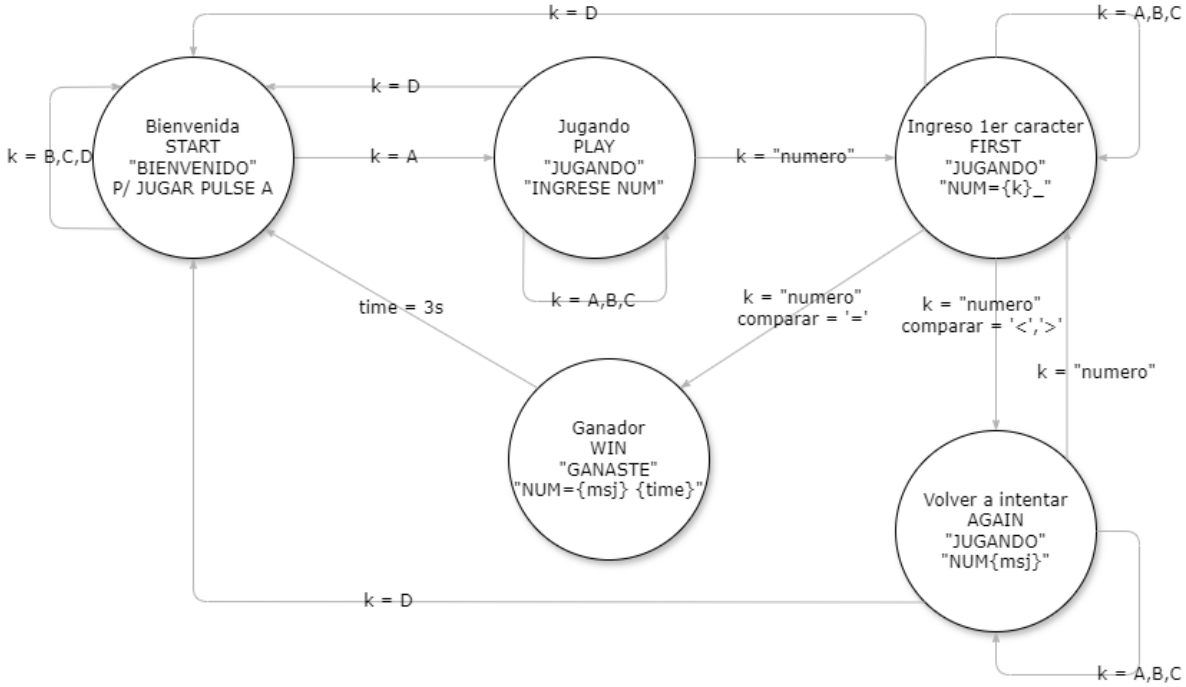
Programa principal

En el programa principal se realiza la inicialización de las distintas funciones del código. Y se ingresa al bucle infinito en el que se quedará revisando hasta el momento en el que se produzca una interrupción.

```
int main(void){
    Inicializacion del LCD
    Configuracion del timer
    Estado inicial de la MEF
    while(1) // loop infinito
    {
        Si se produjo una interrupción{
            Se actualiza la MEF
        }
    }
}Final de programa
```

Función de actualización de la máquina de estados

Esta Función contiene toda la lógica de transición de la máquina de estados, para su implementación primero se mostrará un diagrama de máquina de estado tipo Moore para explicar el funcionamiento de esta.



La máquina de estados, como se puede observar, recibe dos dígitos y arma el numero ingresado por el usuario mediante estos. Esto significa que si se quiere ingresar un número de un digito como respuesta debe ingresarse de la forma “07” por ejemplo.

La función de actualización emplea el funcionamiento de la MEF y agrega funcionalidades internas a la máquina de estados:

- Secuencia de números aleatorios: la inicialización de la secuencia de números se realiza una única vez y cada vez que se comienza a jugar se recibe un nuevo número. En el estado START, la primera vez que se quiera jugar se utilizará el tiempo desde la inicialización del timer para la inicialización de una secuencia de números aleatoria. Se eligió usar este método y no usar otro por la aleatoriedad que brinda el mismo usuario con el tiempo que tarda en empezar a jugar.
- Cronometro: Se mantiene una cuenta del tiempo entre que el jugador pasa al estado PLAY y entre que pasa al estado WIN.
- Se realiza la comparación entre el numero ingresado y el numero objetivo.

El único caso en el que no se lee el teclado es en el estado de WIN, en el que se esperan 3 segundos, independientemente de si se presiona o no una tecla. Y tanto este estado como en el de START son los únicos en los que al presionar la D no se vuelve al último estado mencionado.

A continuación, se muestra un pseudocódigo de la función, el cambio a un estado incluye la impresión en pantalla del mismo.

```
void Actualizar_MEF(void){
    Dependiendo del estado actual se realizará una de las siguientes:
        START:
            Se lee el teclado
            Si se presionó una A{
                Si no se había jugado nunca todavía
                    Se inicializa la secuencia de números
                }
                Se busca un numero aleatorio
                Se inicia el cronómetro de juego
                Se cambia al estado de PLAY
            }
        PLAY:
            Se lee el teclado
            Si se presionó un numero{
                Se cambia al estado de FIRST
            }
            Si se presionó una D{
                Se cambia al estado de START
            }
        FIRST:
            Se lee el teclado
            Si se presionó un numero{
                Se compara el numero ingresado con el buscado
                Si el número es correcto{
                    Se detiene el cronometro de juego
                    Se cambia al estado de WIN
                } Si el número es incorrecto{
                    Se cambia al estado de AGAIN
                }                   }
            Si se presionó una D{
                Se cambia al estado de START
            }
        AGAIN:
            Se lee el teclado
            Si se presionó un numero{
                Se cambia al estado de FIRST
            }
            Si se presionó una D{
                Se cambia al estado de START
            }
        WIN:
            Si pasaron 3 segundos
                Se cambia al estado de START
    }
}
```

TimerMEF.c

Este archivo contiene funciones para el manejo de tiempo en el programa, aunque este archivo fue creado específicamente para esta aplicación, fue creado con funciones lo más generales posibles para que tras pocas modificaciones pueda ser reutilizado en otra aplicación que lo requiera.

Para comenzar a utilizar cualquiera de las funciones se debe realizar una llamada a la función Timer_Init(). Esta función realiza la configuración de los registros necesarias para el funcionamiento de la interrupción e inicializa el registro de cuenta de tiempo global del programa. También recibe como parámetro cada cuanto tiempo se debe actualizar el flag utilizado por la MEF.

Se decidió utilizar el Timer0 como fuente de interrupción con un preescalado de 64 para llegar a contar con los bits necesarios para tener una interrupción cada 1 ms. El cálculo que justifica la elección de 250 como tope de comparación es el siguiente:

Se cuenta con una frecuencia de CPU de 16 MHz.

Con un preescalado de 64, se emula una frecuencia de $\frac{16MHz}{64} = 250KHz$

Si se calcula el tiempo que tarda con esta frecuencia $\frac{1}{250KHz} = 4us$

Y al querer un tiempo de 1 ms, teniendo el periodo de reloj $\frac{1ms}{4us} = 250$

Por lo tanto, la configuración necesaria del reloj tiene que usar como limite el conteo a 250, y como el máximo con 8 bits es de 255, este alcanza para nuestro propósito. Para utilizar como top este número configuramos el reloj con el modo CTC con TOP: OCRA, y guardamos en este registro el valor 250 que necesitamos. Registros que se modifican en esta función:

<u>Modo CTC con TOP: OCR0A</u>

TCCR0A: WGM00 = 0 ; WGM01 = 1

TCCR0B: WGM02 = 0

<u>Preescalador clk/64</u>

TCCR0B: CS02 = 0 ; CS00 y CS01 = 1

<u>Valor máximo para la comparación</u>

OCR0A = 250

<u>Habilitación de la interrupción por COMP A</u>

TIMSK0: OCIE0A = 1

Funciones del timer:

Como fue mencionado antes las funciones en este archivo intentan ser lo más generales posibles por lo que las 4 funciones que ofrece son:

- ActualTime(): Devuelve el tiempo que pasó desde que se inicializó el timer. La variable utilizada tiene un tamaño de 32 bits ya que con el tamaño de 16 bits solo alcanzaba para representar 2^16 ms, que equivale a poco más de 1 minuto. Tiempo que fue considerado muy poco para el uso dado a este contador.

static uint32_t init_count=0;

- Contador "Game": Este contador funciona de forma similar a la que lo hace un cronometro, se comienza un conteo desde que se ejecuta la función GameCountInit(), y con la funcion GameCountStop() se detiene el conteo y se devuelve el tiempo transcurrido en ms. El flag se encarga de señalizar si se esta realizando un conteo y la variable de cuenta tiene el mismo tamaño que el de conteo global por la misma razón.

static uint8_t FLAG_COUNTING=0;
static uint32_t cuentaG=0;

- Contador "Win": Este contador funciona como un temporizador. Al ejecutarse la funcion WinCountInit() y pasarle como parámetro un número, se utiliza este como valor inicial de un conteo regresivo. Con la funcion WinCount() se puede ver el tiempo restante. El tamaño de la variable es de 16 bits porque, a diferencia de los casos anteriores, se consideró que el tiempo de 1 minuto si era suficiente para la cuenta regresiva que en este programa es de tan solo 3 segundos.

static uint16_t cuentaW=0;

- Acceso a un flag de interrupción: Esta función se compone del acceso a el getter y el setter de un flag que se actualiza cada un tiempo establecido al inicializar el timer. En el programa se utiliza un tiempo de 10 ms. Las variables de cuenta y top tienen un tamaño de 8 bits porque se consideró que el uso que se le da a esta variable es de una interrupción periódica que no necesita un tamaño mayor a esto.

static uint8_t FLAG_MEF=0;
static uint8_t cuentaMEF=0, topMEF=0;

Llamado a interrupción:

Se utiliza una única interrupción para manejar las funciones antes mencionadas, en la ISR correspondiente se incrementan los contadores de cuenta global, de cuenta para el flag de la MEF y si se está realizando una cuenta de juego el contador "Game". Tambien se disminuye el valor del contador "Win" si es que este tiene un valor distinto de 0. Como esta interrupción se produce cada 1 ms, todos los tiempos, ya sea que aumenten o disminuyan, lo hacen cada este tiempo

ScreensMEF.c

En este archivo se encuentran las funciones dedicadas a la impresión de las pantallas para los distintos estados del juego. Las funciones tienen todas como nombre "P_" y a continuación el nombre del estado al que corresponde la pantalla.

Las funciones arrancan todas siguiendo los siguientes pasos:

- Se limpia la pantalla con un LCDclr()
- Se posicionan en la primer linea con un LCDGotoXY(0,0)
- Se imprime el texto de la primera linea con LCDstring(string,length), en el que el primer carácter corresponde al string a imprimir y el segundo al tamaño de este.
- Se posiciona al comienzo de la segunda linea.

A partir de este punto difieren en el contenido de la segunda linea:

- P_START y P_PLAY: En estas funciones tan solo se imprime un string en la segunda linea utilizando la funcion LCDstring().
- P_FIRST(uint8_t k): Esta funcion recibe como parámetro un carácter de un número. Al comienzo de la segunda linea se imprime el string "NUM:" usando la funcion LCDstring() y luego se imprime el carácter del número correspondiente, utilizando la funcion LSDsendChar().
- P_AGAIN(uint8_t msj[3]): El parámetro que recibe esta función es una cadena de 3 posiciones. En la posición 0 se debe guardar un carácter correspondiente a una comparación, ya sea '<' o '>'. Y en los siguientes dos los dos dígitos que conforman el numero ingresado por el usuario. Utilizando el LCDstring() primero se realiza la impresión de "NUM" y luego la del mensaje recibido. Por ejemplo, finalmente quedaría "NUM<28".
- P_WIN(uint8_t msj[3], uint32_t win_time): Esta función recibe dos parámetros: El primero es igual que el recibido por la función anterior, pero en la primer posición debe tener un '='. La impresión hasta este punto se realiza de la misma forma que en P_AGAIN. Luego re realiza la impresión de un espacio con LCDsendChar(). El segundo parámetro representa en ms el tiempo que debe ser impreso. Para su impresión en minutos y segundos se realiza la siguiente cuenta:

  ```
  ms = win_time % 1000;
  win_time = win_time / 1000;
  seg = win_time % 60;
  min = win_time / 60;
  ```

  Y luego se realiza la impresión de los datos utilizando la función LCDescribeDato() separados por caracteres ':' impresos por la función LCDsendChar(). La primer función funciona como LCDstring() pero en lugar de recibir un string recibe un entero

KeyPad.c

En este archivo se encuentran las dos funciones correspondientes a la lectura del teclado matricial. Ambas fueron tomadas de la clase entregada por la catedra, pero modificadas para funcionar con la configuración de la conexión específica del problema.

Funcion KepadUpdate():

Esta es la funcion encargada de realizar el escaneo del teclado matricial, para ello primero se realiza la configuración de los puertos y se establecen en los archivos DDRB y DDRD respectivamente:

- Salidas: Pines 4, 3 y 0 del puerto B y el pin 7 del puerto D.
- Entradas: Pines 5, 4, 3 y 2 del puerto D. Para estas entradas también se configuro como pull-up interno desde el PORTD.

El archivo cuenta con una matriz que representa los valores que se encuentran en el teclado matricial. Tambien cuenta con dos vectores que fueron creados para recorrer en orden las entradas y salidas, siguiendo el orden en el que fueron conectados al teclado.

Las salidas siguen el orden: B4, B3, B0, D7

Las entradas siguen el orden: D3, D5, D4, D2

Luego de esto se ingresa a dos for anidados, el primero es el encargado de establecer todas las salidas como 1 y luego establecer uno en 0 siguiendo el orden de salidas explicado. La conexión de las salidas tiene sus tres primeros pines en el puerto B, pero el ultimo se encuentra en el puerto D, por lo que se debe diferenciar a este del resto para cambiar el valor en PORTD en lugar de PORTB.

El segundo for se encarga de revisar las entradas para ver si alguna cambió su valor 1 dado por el pull-up interno a un 0 causado por la presión de una tecla. Para esto revisa los pines de D conectados a las columnas del teclado en el orden dado. En el caso de encontrar una tecla presionada se devuelve el carácter correspondiente.

Funcion KEYPAD_Scan():

Esta funcion es la explicada por la catedra, no sufrió modificaciones. Sirve para evitar El escaneo múltiple de un mismo valor y para corregir el posible rebote mecánico añadiendo una segunda verificación. Cuando se leyó un valor valido devuelve un 1 y en el caso de no leer un valor en el teclado o de encontrarse en la primera verificación devuelve un 0.

3. VALIDACIONES

A la hora de correr el programa se observa el estado inicial del juego donde al presionar cualquier botón del teclado no realizará ninguna modificación a excepción del carácter A, al presionarse este inicia el contador del tiempo de juego y se muestra el estado PLAY. Al momento de pulsar algún número entre el cero y el nueve se desplaza al estado FIRST, luego se espera a que se introduzca el segundo dígito válido, en ese momento se compara el valor proporcionado por el usuario con el valor a adivinar. En caso de que la comparación sea exitosa el programa te redirige al estado WIN por un tiempo de 3 segundos, informando el tiempo de juego transcurrido. En cambio, si el número ingresado no coincide, el juego muestra si el número ingresado fue mayor o menor que el número generado, indicando al jugador que intente de nuevo, pasando al estado AGAIN. Durante el juego, si en cualquier momento se presiona 'D', el juego se reinicia y retorna al estado inicial.

El teclado matricial 4x4 utilizado cuenta con dígitos que se consideran válidos para la solución que son '0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'A', y 'D' donde al pulsarlos el programa debe procesarlo en cambio hay otros que son el 'B', 'C', '*', y '#' los cuales su pulsador no debe ser tenido en cuenta en ningún momento.

Estas características del funcionamiento se ven demostradas en las siguientes pruebas, una muestra el programa funcionando en el kit entregado por la catedra y el otro muestra una simulación realizada en el programa Proteus.

Prueba de funcionamiento en el kit:

https://drive.google.com/file/d/1MMpT4mvnFsWWxAFKgFszFdNmbEDKQn8b/view?usp=sharing

Prueba de funcionamiento en Proteus:

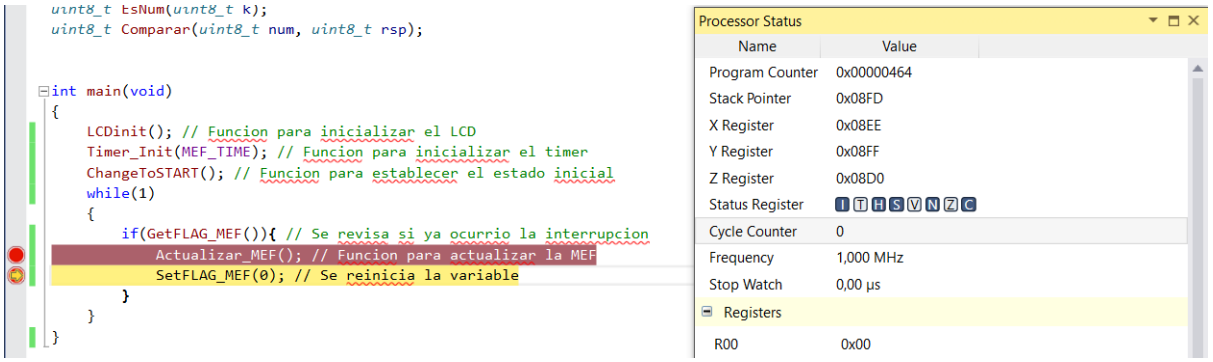
https://drive.google.com/file/d/1Lt9BkZe98R1YlJMRHzNMLVof45PrOtwD/view?usp=drivesdk

Como puede verse, al tratarse de una simulación, el tiempo que devuelve el programa en Proteus no refleja el tiempo real transcurrido, y para una mejor experiencia de simulación por esta misma razón se disminuyo el tiempo de espera en la pantalla ganadora.

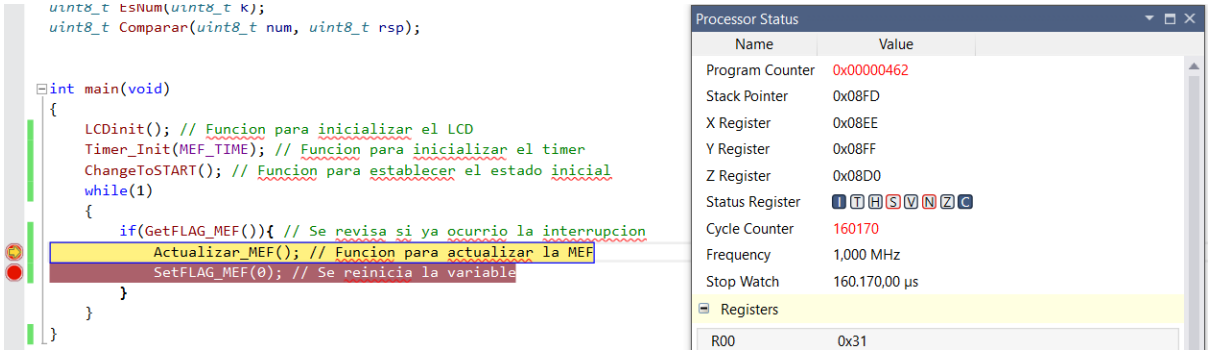
Verificación de tiempo:

La validación de la temporización se realizará mediante Microchip Studio. Para esto se verificará el tiempo transcurrido entre dos actualizaciones de la maquina de estados, que para nuestro programa sucede cada 10 ms.

Tomando como estado inicial el momento de salida desde una actualización:



Se tiene que la cantidad de ciclos transcurridos desde que se puso el flag de la MEF en 0 hasta que volvió a activarse (pasadas 10 interrupciones de 1 ms) es de 160170 ciclos, que aproximadamente dan 10 ms para un procesador con 16 MHz de frecuencia.



Esta medida verifica el correcto funcionamiento del temporizador utilizado en el timer, y al estar todas las medidas de tiempo del programa manejadas por la misma interrupción, esta sirve para demostrar al resto de funcionalidades.

4. CONCLUSIÓN

El programa desarrollado cumple con los requerimientos establecidos en la consigna, proporcionando una experiencia de juego interactiva mediante el uso de un display LCD, un teclado matricial y el microcontrolador Atmega328p. La implementación con una máquina de estados temporizadas asegura una transición ordenada entre los diferentes estados del juego, lo que incluye la bienvenida, el inicio del juego, la evaluación de los intentos del jugador y la finalización del juego con la indicación del tiempo transcurrido.

La modularización permite dividir el programa en partes más pequeñas, manejables e independientes, cada una de las cuales realiza una función específica, esto no solo facilita el desarrollo y la depuración del código, sino que también permite reutilizar módulos en futuros proyectos.

La abstracción ayuda a simplificar la complejidad del sistema al ocultar los detalles de implementación de los componentes, por ejemplo, las funciones para interactuar con el hardware, como escribir en el LCD o leer el teclado, están encapsuladas en funciones de archivos específicos con ese propósito. Esto posibilita centrarse en el flujo del juego sin preocuparse por los detalles de bajo nivel, permitiendo hacer el programa más fácil de verificar, de optimizar y mantener.

Un tiempo de respuesta rápido asegura que las acciones del usuario, como presionar una tecla, se reflejen inmediatamente en el sistema, la implementación con una máquina de estados temporizada y la planificación periódica garantiza que el sistema responda de manera oportuna a las entradas del usuario, proporcionando una experiencia de juego fluida y sin demoras perceptibles.