Laboratorio 1:
Introducción a microcontroladores y manejo de GPIOS

Profesor: Marco Villalta
Estudiante: Amy Herrera Mora B53473

I. Introducción

El presente laboratorio introduce los conceptos de microcontroladores y manjeo de GPIOS. Para este caso se desarrolla un simulador de tómbola simplificada de bingo que utilizando dos displays de 7 segmentos para mostrar números generados de forma aleatoria. Para esto se utiliza un botón, el microcontrolador PIC12F683 así como componentes necesarios para su implementación.

El objetivo de este laboratorio es introducir al manejo de GPIOs y generación de números aleatorios. Cada vez que se presiona el botón se debe desplegar en el display un número que represente a una de las bolas de la tómbola que van del 00 al 99. Se debe mantener un registro de los números que salen para no repetir. Para efectos de simplificar la simulación se consideran solamente 16 bolas. Finalmente, después de sacar 16 bolas/números, deberá parpadear la pantalla con el numero 99 y reiniciar el juego.

AL finalizar el laboratorio, se encontró que este microcontrolador es bastante pequeño y que genera problemas de memoria y de tener cuidado a la hora de crear un algoritmo muy complejo. Sin embargo, se pudo solucionar la mayor parte de los problemas para que funcionara de la mejor manera.

II. Nota Teórica

A. *Características generales*

Los microcontroladores (Main Control Units) se han desarrollado para cubrir diversas aplicaciones. Estos se usan en automoción, equipos de comunicaciones y de telefonía, instrumentos electrónicos, equipos médicos, electrodomésticos, juguetes, entre otros. Se requiere de estos para necesidades puntuales, o sea, cuando se quiera realizar un pequeño numero de tareas al menor costo posible. Su manera de operación es ejecutar un programa que se almacena de forma permanente en su memoria para trabajar con algunos datos temporales. Con esto interactúa con el exterior a través de lineas de entrada y de salida con las que dispone. Es un controlador incrustado que combina las partes fundamentales de un microcomputador. Estos son: memoria y recursos de entrada y salida en un único circuito integrado. *cita*

Dentro de las características generales mas importantes para este microcontrolador PIC12F683 se tienen que es un flash de 8 bits con 8 pines parte de los microcontroladores CMOS con tecnología NanoWatt. Su procesador tiene arquitectura RISC (Reduced Instruction Set Computer) con solamente 35 instrucciones y todas monociclo. Una corriente de operación de $11\mu A$ a 32 kHz y 2.0V; y a $220\mu A$ con 4 MHz y a 2.0V. Para las características periféricas están un modulo de comparador análogo, un convertidor A/D de 10 bits de resolución con 4 canales, un contador de programa de 13 bits capaz de direccionar un programa con espacio de memoria de 8k x 14. Solamente el primer 2k x 14 (0000h-07FFh) para el PIC12F683 esta físicamente implementado. Finalmente, se habla que contiene los 6 pines I/O.

Para el caso especifico del PIC12F683, se tienen 8 pines de los cuales V_{DD} y V_{SS} son sus alimentaciones, y los seis restantes son GPIOS I/O como se mencionó anteriormente, o sea, General Purpose Input Output. Estos son aquellos pines con los que se puede programar o decir como se van a utilizar. La figura 1 muestra el diagrama de pines para este microcontrolador.

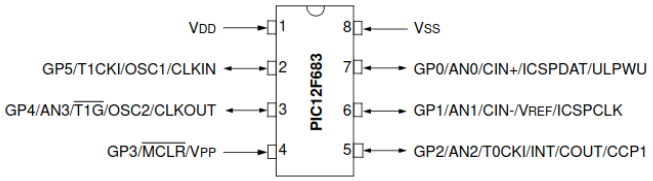

Figure 1: Diagrama de pines para el PIC12F683

Para poder configurar estos GPIOS se hace a través de registros, y depende del uso que se le de al MCU se hará necesario modificar mas de un registro. Con esto es que se especifican cuales seran las salidas y las entradas. Cuando se quiera una entrada, se hace la lectura con polling, y cuando se necesite una salida, se maneja el estado en el programa como un Alto: 1 y Bajo: 0.

Para operar un PIC12F683 digitalmente se utilizan los registros: TRISIO, GPIO, ANSEL y/o CMCON y a veces el CONFIG. Estos se explicaran a continuación.

En primer lugar, se aprecia en la figura 2 el registro TRISIO que configura el flujo de datos. Este indica el modo de operación de cada pin. Un bit en alto: 1 servirá como entrada y un bit en bajo: 0 sera una salida.

REGISTER 4-2: TRISIO GPIO TRI-STATE REGISTER

U-0	U-0	R/W-1	R/W-1	R-1	R/W-1	R/W-1	R/W-1
—	—	TRISIO5(2,3)	TRISIO4(2)	TRISIO3(1)	TRISIO2	TRISIO1	TRISIO0
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

Figure 2: Registro TRISIO

En segundo lugar, el registro que se observa en la figura 3 es el GPIO donde se guarda el estado del puerto GPIO. Con este se lee el el estado del pin para escribir el latch de salida.

REGISTER 4-1: GPIO: GENERAL PURPOSE I/O REGISTER

U-0	U-0	R/W-x	R/W-0	R-x	R/W-0	R/W-0	R/W-0
—	—	GP5	GP4	GP3	GP2	GP1	GP0
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

Figure 3: Registro GPIO

En tercer lugar, se encuentra en la figura 4 el registro ANSEL. Este se debe inicializar para poder configurar la conversión y selección analógica. Se modifica si se utiliza cualquier pin como entrada.

REGISTER 4-3: ANSEL: ANALOG SELECT REGISTER

U-0	R/W-0	R/W-0	R/W-0	R/W-1	R/W-1	R/W-1	R/W-1
—	ADCS2	ADCS1	ADCS0	ANS3	ANS2	ANS1	ANS0
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

Figure 4: Registro ANSEL

Finalmente, en la figura 5 se muestra el registro CMCON que configura el comparador analógico. Este se modifica si se utilizan como entradas los pines GP0, GP1 y GP2.

REGISTER 8-1: CMCON0: COMPARATOR CONFIGURATION REGISTER

U-0	R-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	COUT	—	CINV	CIS	CM2	CM1	CM0
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

Figure 5: Registro CMCON

Este microcontrolador inicia por defecto con el watchdog timer habilitado y con el GPIO3 configurado como MCLRE (reset). El registro CONFIG esta localizado fuera de la memoria del programa porque pertenece al espacio de la configuración de esta. Este espacio puede ser accedido cuando se programa y mediante instrucciones de preprocesador según la documentación del archivo de encabezado, el compilador y la hoja de datos de este MCU.

Para configurar los pines, esto se puede realizar a bit a bit o todos de una vez con diferentes métodos. El primero es el de la sobre escritura de un registro, pero no se recomienda porque reemplaza configuraciones previas en el mismo programa y tiende a generar pulgas. Luego, esa el método enmascarado que es una buena solución para no modificar bits ya configurados anteriormente. Finalmente, el método por campo de bit. Este ultimo modifica bits uno a uno. Con esto ya se pueden leer y escribir sus estados.

Una vez que se completa la configuración de los pines, es indispensable revisar con la hoja de datos las capacidades eléctricas de estos para comenzar a construir el circuito. Se vuelve importante agregar algunos componentes electrónicos para que funcione bien y se obtenga un comportamiento estable. Las características se aprecian en la figura 6. Cabe mencionar que pasarse de los rangos máximos absolutos que vienen listados puede generar daños permanentes en el dispositivo.

15.0 ELECTRICAL SPECIFICATIONS

Absolute Maximum Ratings(†)

Ambient temperature under bias	-40° to +125°C
Storage temperature	-65°C to +150°C
Voltage on V_{DD} with respect to V_{SS}	-0.3V to +6.5V
Voltage on $\overline{MCLR}$ with respect to V_{SS}	-0.3V to +13.5V
Voltage on all other pins with respect to V_{SS}	-0.3V to (V_{DD} + 0.3V)
Total power dissipation(1)	800 mW
Maximum current out of V_{SS} pin	95 mA
Maximum current into V_{DD} pin	95 mA
Input clamp current, I_{IK} (V_I < 0 or V_I > V_{DD})	± 20 mA
Output clamp current, I_{OK} (V_O < 0 or V_O > V_{DD})	± 20 mA
Maximum output current sunk by any I/O pin	25 mA
Maximum output current sourced by any I/O pin	25 mA
Maximum current sunk by GPIO	90 mA
Maximum current sourced GPIO	90 mA

Figure 6: Especificaciones eléctricas

B. Diseño del circuito

Ahora, como se establece en el enunciado para este laboratorio, los números aleatorios se generan cada vez que se presiona el botón. Entonces, esta sera la primera parte del diseño para el circuito total. Sin embargo, hay que tomar en cuenta el efecto rebote que se puede presentar. Para esto existe una teoría que explica que cuando se presiona un switch, dos partes de metal se juntan.

Para el usuario este contacto se observa casi que instantáneamente, pero realmente el contacto no es uniforme y dura algún tiempo es cerrarse completamente. En este proceso el switch queda entre el contacto y no contacto de las partes de metal, o sea, se conectan y reconectan entre si. Esto genera una pequeña inestabilidad que debe ser controlada.

Para solucionar esto, se puede agregar un capacitor tal y como se observa en la figura 7. En la parte izquierda se aprecia el Push que genera una respuesta inestable, y en la parte derecha ya el comportamiento controlado. Usualmente lo que se hace es agregar un delay de 50 ms después de presionar el pulsador la primera vez.

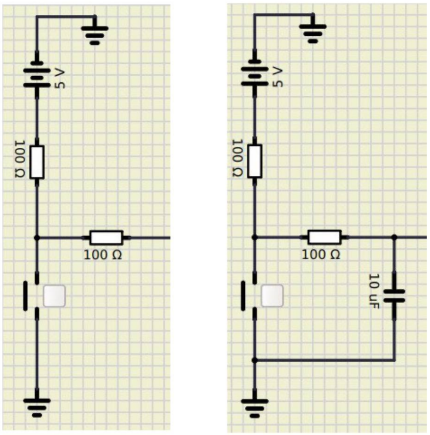

Figure 7: Especificaciones eléctricas

Se escogieron las resistencias con los valores de 100Ω cada una. De igual forma para el capacitor de $150pF$. Si encontramos la constantes de carga, se tiene la expresión 1 que ejemplifica un valor suficiente para el retardo y así evitar en la medida de los posible el efecto rebote del switch.

$$\tau = R * C = (100\Omega + 100\Omega) * 150pF = 30ns \quad (1)$$

Luego, como se quieren mostrar los números generados en dos displays, hay que tomar en cuenta la protección requerida para estos, pues son LEDS. Para esto, se encuentra en la hoja de datos del fabricante en la sección de especificaciones eléctricas que cada pin I/O entrega una corriente de 25 mA como máximo. Al recordar que cada alto significan 5 V y cada bajo 0 V, se puede aplicar la Ley de Ohm para saber cual es el valor mas apropiado para estas resistencias.

$$R = \frac{V}{I} = \frac{5V}{25mA} = 200\Omega R \quad (2)$$

Finalmente, el circuito entero se puede apreciar en la figura 8.

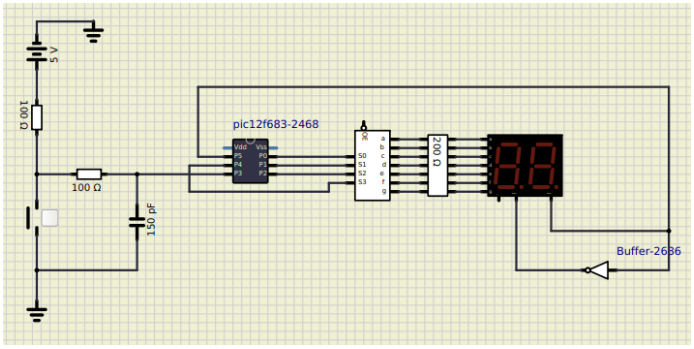

Figure 8: Esquemático entero para el diseño del circuito en el simulIDE

En la parte izquierda se puede apreciar el filtro generado para evitar el rebote del switch con dos resistencias de 100 y un capacitor de 150 pF. Seguidamente, se aprecia el MCU PIC12F683 conectado al decodificador BCD que es el que se utiliza para estos displays de siete segmentos y que en el simulador utilizado viene con el nombre BDCTo7S-2461. Según se observa en la figura **, estos displays tiene LEDs. Es por esta razón que entre las salidas del decodificador y las entradas de display hay siete resistencias de 250 Omega.

De la figura 8 se puede ver que para este laboratorio se utilizó como única entrada el GPIO3 y el GPIO5 como salida hacia los dos displays. Los restantes GPIOS: GPIO0, GPIO1, GPIO2 y el GPIO4 son las salidas hacia el decodificador que va a combinar esas entradas para poner en alto o en bajo los 7 segmentos respectivos y poder con esto encender cada LED necesario. Cabe destacar que poner en alto significa poner con 5 V cada salida, y que se agrega una compuerta not entre la habilitacion de cada uno de los dos displays para inverter o seleccionar la pantalla que se quiere mostrar y que no se observe el mismo número generado en los displays.

Una vez explicado el diseño y los componentes, la tabla de estos con sus precios viene dada por la tabla 1.

Componente	Valor	Cantidad
Resistencia de 100 Omega	$1.5	2
Resistencia de 250 Omega	$7.5	7
Microcontrolador PIC12F683	$2.44	1
Decodificador BCDTO7S	$3.90	1
Compuerta Not	$4	1
Display de siete segmentos	$2.5	2
Total	**$21.84**	

Table 1: Tabla de los componentes para el diseno

III. Desarrollo/Análisis de resultados

Una vez creado el diseño, se procede a explicar el codigo implementado. Para esto, se utilizó el lenguaje C con las bibliotecas stdio.h, pic14/pic12f683.h, math.h, stdbool.h. Como se dijo en la nota teórica, lo primero a realizar es la configuración de los registros TRISIO, GPIO, ANSEL y CMCON. Se indica que tanto ANSEL como CMCON no se van a utilizar para este laboratorio como se indica en la figura 9. Con esto se logra que todos los pines activados funcionen como salidas excepto el GPIO3 que será la única entrada. Adicionalmente, se declaran algunas variables globales que servirán para asignar las entradas del decodificador y que con esto se generen las combinaciones necesarias.

```
typedef unsigned int word;
word __at 0x2007 __CONFIG = (_WDTE_OFF & _WDT_OFF & _MCLRE_OFF);  // WDT y MCLR OFF

void delay (unsigned int tiempo);
unsigned char randomNumberGen();

ANSEL = 0;
CMCON0 = 0x07;
TRISIO=0b00001000;
GPIO = 0x00;

int D0;
int C1;
int B2;
int A3;

int D4;
int C5;
int B6;
int A7;
```

Figure 9: Código implementado

Seguidamente, se inicializan en el main las variables necesarias. Dentro de este también se implementa un while infinito, ya que los microcontroladores necesitan estar corriendo el programa las veces que se necesite. Esto se observa en la figura 10.

```
void main() {

int t;
int u;
int i;
int num;
int counter;

D0 = 0;
C1 = 0;
B2 = 0;
A3 = 0;

D4 = 0;
C5 = 0;
B6 = 0;
A7 = 0;

t = 0;
u = 0;
n = 2;
t = 0;
u = 0;
n = 2;
counter = 0;
```

Figure 10: Código implementado

Ahora, dentro del main se tiene un if para cada vez que el botón del inicio se presione. Una vez que se haya presionado, se van a generar números solamente si el contador es menor a 15 para lograr que slo se muestren los 16 números solicitados en el enunciado. Caso contrario se tiene un else para que se muestre el número 99. Esto se observa en la figura 11.

```
if (GP3==0) {
            delay(100);


      if (counter <= 15) {

            num = randomNumberGen();
            u=num%10;                   // Units
            t=(num/10)%10;              // Tens

            counter = counter+1;
   }

   else {
            delay(100);
            u=9;                  // Units
            t=9;                  // Tens
            counter = 0;

            }
```

Figure 11: Código implementado

En la figura 11 se aprecia que el numero generado se divide en unidades y decenas para que cada uno por separado pueda mostrarse en el display. Se implementa de manera que las decenas se vean en el display izquierdo y las unidades en el display derecho. Luego, cada uno de estos valores entra a un switch units y a un switch tens. Dependiendo del valor, se asignan las entradas del decodificador para que este combine según los segmentos para ese numero dado. Existen 9 casos para cada unidad o decenas para los números de 0 al 9. Se puede ver en la figura 12.

```
switch (u) {
            case 0:

                  D0 = 0;
                  C1 = 0;
                  B2 = 0;
                  A3 = 0;
                  break;

            case 1:

                  D0 = 1;
                  C1 = 0;
                  B2 = 0;
                  A3 = 0;
                  break;
```

Figure 12: Código implementado

Finalmente, se tienen dos funciones. La primera es generadora de los números pseudo randoms, y la segunda un delay de tiempo ya que es necesario esperar algún tiempo para poner observar en los displays, así como de generar un numero cada vez que se presione el botón. Estas se aprecian en la figura 13.

```
262 unsigned char randomNumberGen(){
263
264                         n |= n == 0;
265                         n ^= (n & 0x07) << 5;
266                         n ^= n >> 3;
267
268                         n ^= (n & 0x03) << 6;
269                         n = n & 0xff;
270
271 return n;
272
273 }
274
275 void delay(unsigned int tiempo)
276 {
277         unsigned int i;
278         unsigned int j;
279
280         for(i=0;i<tiempo;i++)
281           for(j=0;j<1275;j++);
282 }
```

Figure 13: Código implementado

En los apéndices se observa ya el circuito simulado y conectado en el simulador SimulIDE. Se generaron tres números para la demostración. Luego de presionar el botón 16 veces, parpadea el numero 99.

IV. Conclusiones y recomendaciones

- Como principal conclusión se tiene que este microcontrolador es bastante pequeño. Solo cuenta con 6 pines I/O, entonces a la hora de poder generar las salidas se tuvo que recurrir a una compuerta Not. Para este laboratorio no fue muy notoria esta problemática ya que según la tabla 1 este llegar a ser muy barato, pero para futuros proyectos es mejor tomarlo en cuenta.

- El PIC12F683 no tiene tanta memoria. Los algoritmos que puede correr son bastante simples. Se quiso agregar un arreglo para poder revisar los números generados y no se permitió su compilación por falta de memoria.

- Como ya se menciono, implementar este experimento en la vida real puede llegar a ser de muy bajo costo, pero como todo hardware, por ser tan barato se vuelve muy limitado.

V. Link a GIT

Este es el link del repositorio que se va a utilizar para el desarrollo de los diferentes laboratorios:

https://github.com/mimiherrera/IE0624-Lab-de-Microcontroladores

VI. Referencias

[1] Microship. *Pic12F683 Datasheet 8-pin FLASH-Based 8-Bit CMOS Microcontrollers*, 2007

APPENDIX A

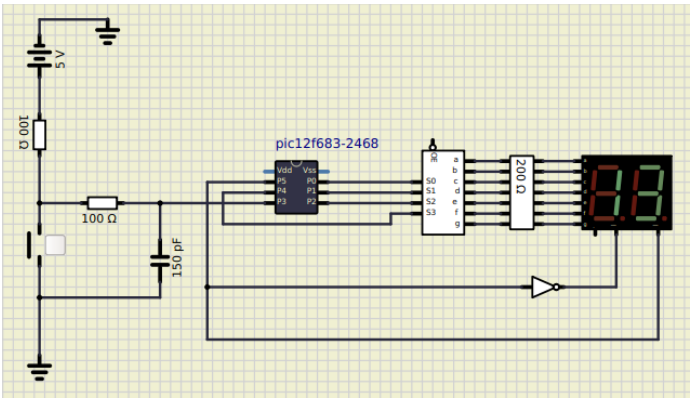

Figure 14: Simulación implementada. Se muestra el valor generado 13.

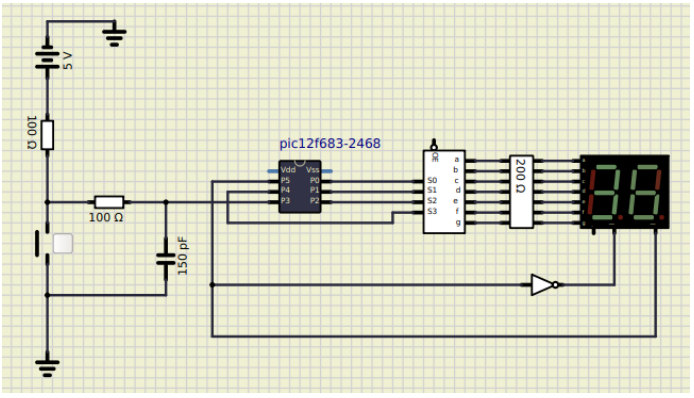

Figure 15: Simulación implementada. Se muestra el valor generado 36.

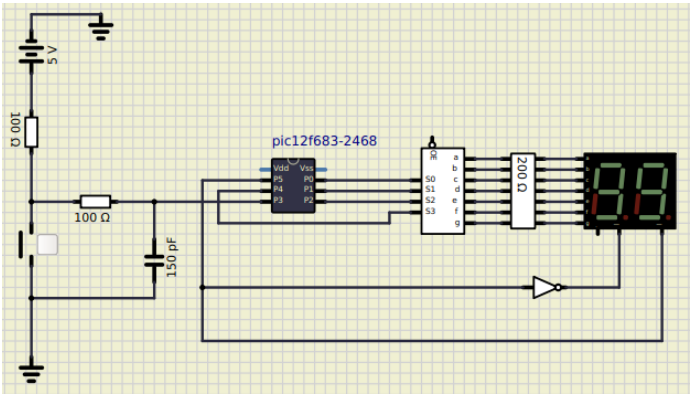

Figure 16: Simulación implementada. Se muestra el valor generado 99.