EIE
Escuela de
Ingeniería Eléctrica

Universidad de Costa Rica
Facultad de Ingeniería
Escuela de Ingeniería Eléctrica

UNIVERSIDAD DE
COSTA RICA

IE-0624
Laboratorio de Microcontroladores
Marco Villalta Fallas

Laboratorio #1 : PIC12f683

Jesus Jiménez Acosta
B73881

I ciclo 2023

1. Introducción

Para el primer laboratorio se usó el microcontrolador pic12f683 para crear una tómbola de un bingo que genere 16 números aleatorios entre el 00 y el 99,en el cual se usó un codificador de BCD a 7segment,dos displays de 7 segmentos,un pulsador y un filtro RC.El pulsador es necesario ya que el es el que dicta cuando sale un numero aleatorio,la señal entra al MC y genera un numero aleatorio con el método *Linear feedback shift register* también denominado como lfsr ,el cual aplica una retroalimentación lineal y un desplazamiento de bits mediante una compuerta XOR.
El microcontrolador pic12f683 cuenta con una EEPROM (*Electrically Erasable Programmable Read Only Memory*) de 128 bytes y con solo 6 GPIO usables lo cual fue un reto a la hora de establecer un diagrama y crear un programa que no rebasara la memoria disponible,de lo cual se pudo usar el pic12f683 exitosamente para la generación de números aleatorios.

2. Nota Teórica

2.1. PIC12f683

2.1.1. Características generales del MC:

- Es un MC de 8 bits
- Tiene una arquitectura RISC
- Tiene 64 bytes RAM y 128 bytes EEPROM .
- Tiene 6 pines GPIO que pueden configurarse como salidas o entradas digitales.
- Es de bajo consumo de energía ya que su consumo de corriente es de $100\mu A$

2.1.2. Para el diagrama de pines :

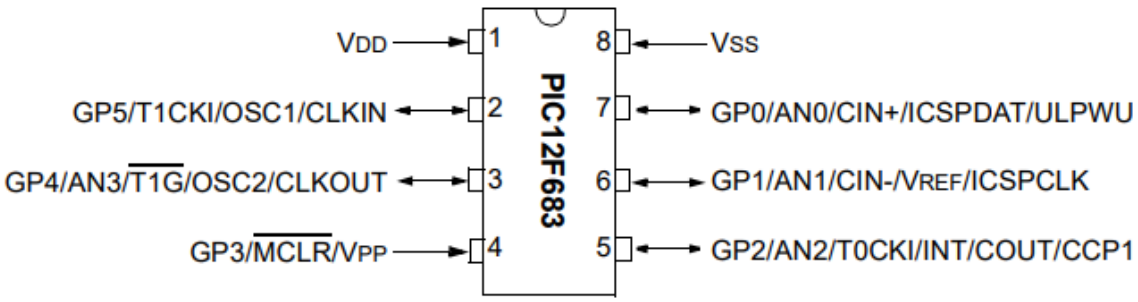


Figura 1: Diagrama de pines PIC12f683. Tomado de [1]

2.1.3. Las características eléctricas:

Absolute Maximum Ratings(†)

Ambient temperature under bias	-40°C 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 Vss	-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	300 mA
Maximum current into V_{DD} pin	250 mA
Input clamp current, I_{IK} (V_I < 0 or V_I > V_{DD})	± 20 mA
Output clamp current, I_{OK} (Vo < 0 or Vo > 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	200 mA
Maximum current sourced GPIO	200 mA

Figura 2: Especificaciones eléctricas. Tomado de [1]

2.1.4. Registros utilizados

Registro TRISIO: Este registro permite modificar si los pines son entradas o salidas.
Registro ANSEL: Con este registro se puede controlar el conversor analógico/digital del microcontrolador.
Registro GPIO: Este registro se utiliza para configurar el estado predeterminado de los GPIO, el cual puede ser alto o bajo .
Registro CMCON: Este registro sirve para controlar el comparador que se encuentra dentro del microcontrolador .
La configuración de todos los pines se tiene en el siguiente código:

```
1    TRISIO = 0b00001000 ; // Se configura gp3 como entrada los demas como salidas
2    ANSEL = 0b00001000; // Set GP3 as digital input
3    CMCON0 = 0x07; //Se apagan los comparadores
4    GPIO = 0x00; //Inicializan todos los pines en 0
```

Para el dimensionamiento de las resistencias de protección a los dos displays,basta con saber la corriente máxima que corre en las salidas de los gpio,según la imagen 2 la corriente máxima es de 25mA pero se configuró para 20mA,de lo cual se forma una malla de R_{protec} y el cada led de los dos displays ,para el valor numérico se tiene la siguiente ecuación:

$$R_{protec} = \frac{5}{20x10 - 3} \approx 250\Omega \tag{1}$$

2.1.5. Switch Bounce Circuit:

Tal y como se vió en clases hay un efecto rebote al haber un cambio de tensión en los pines ,de los cual se implementó un filtro RC de entrada con una constante de tiempo adecuada para que evite los rebotes en cada conmutación del MC,que de acuerdo a la imagen 2 ocurren en el ciclo de reloj 4MHz.El cálculo partido de usar dos resistencias de 100Ω y un capacitor de 100pF:

$$\tau = RC = 200 \cdot 100x10^{-12} = 20x10^{-9}s \tag{2}$$

2.1.6. Precio Componentes:

Componente	Cantidad	Precio Total ($)
PIC12f683	1	1.92
Resistencia 250Ohm	2	0.2
Resistencia 100Ohm	7	0.7
Display 7 segmentos	2	2.56
Decoder BCD to 7s	1	2.02
NOT gate	1	1.09
Total	8.49$	

2.1.7. Esquemático Final Bingo:

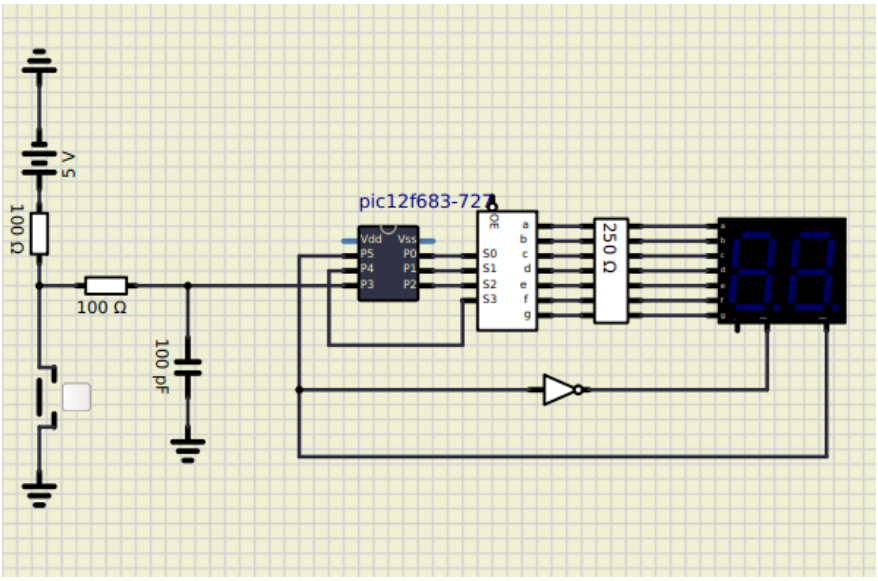


Figura 3: Circuito bingo. Creación propia

El funcionamiento del bingo es de la siguiente forma: el GP3 funciona como la entrada del circuito con un botón,esto se configuró desde el registro TRISIO en el código antes expuesto,los GP0,GP1,GP2,GP4,GP5 son las salidas del circuito,en este caso el GP5 se encarga de switchear el enable a las pantallas,de lo cual se tiene que en un instante de tiempo

GP5 es cero,por tanto el display de las unidades lee el valor mientras que la compuestar not hace que el display de las decenas esté apagado,despues de un pequeño delay el GP5 se invierte y ahora el display de las decenas está prendido y el de las unidades apagado,la idea de esta lógica es aprovechar el reloj rápido del MC para hacer la operación tan rápido que el ojo humano no pueda percibir esos cambios de encendido y apagado entre las decenas y centenas.Una vez hecha la lógica correspondiente para calcular el numero aleatorio se puede crear la lógica para demultiplexar las salidas de los gpio y crear el bingo.

3. Desarrollo y Análisis de resultados

Primero se tiene el diagrama de flujo que se usó para pensar el circuito:

```
/*
Diagrama de flujo
presiono boton:
i<16?
si:
calcular numero aleatorio:
el numero es igual a los guardados?
si1:
no guardar en el registro
pantalla negro
jump: presiono boton
no1:
guardar en el registro
imprimir
jump: presiono boton
no:
vaciar registro
pantalla negro
jump: presiono boton

*/
```

Primero se parte por el alcance al que llegó el circuito,en este caso el circuito es capaz de crear numeros aleatorios con ayuda del siguiente código:

```
struct Numero_total {
  int Unid;
  int Decen;
};
struct Numero_total* numero_aleatorio() {
    struct Numero_total numero;
```

```
7       int aux = 0 ;
8       unsigned int bit;
9       // se cambia el lfsr y se hace el cambio del bit
10      bit = ((lfsr >> 0) ^ (lfsr >> 2) ^ (lfsr >> 3) ^ (lfsr >> 5)) & 1;
11      // se cambia el lfsr con el bit anterior
12      lfsr = (lfsr >> 1) | (bit << 15);
13
14      // se usa el modulo de 100 al lfrs ya que se ocupa un numero entre 00 y 99
15      aux=lfsr % 100;
16      //printf("num1 = %d",aux);
17      numero.Unid = aux % 10;
18      numero.Decen = (aux / 10) % 10;
19      return &numero;
20  }
21
```

Con esto se puede calcular un numero aleatorio y separarlo en unidades y decenas,para desplegarlo en cada pantalla por separado.Este código funcionó perfectamente y desplegaba los números aleatorios.A la hora de realizar este informe se intentó realizar las capturas del funcionamiento sin embargo debido a que la computadora que se usa posee capacidades técnicas limitadas,lo que ocasionó que el simulide no compilara de forma correcta y por tanto el real speed desplegado en pantalla fuera de 1 % o menor,esto se ve en la siguiente imagen:

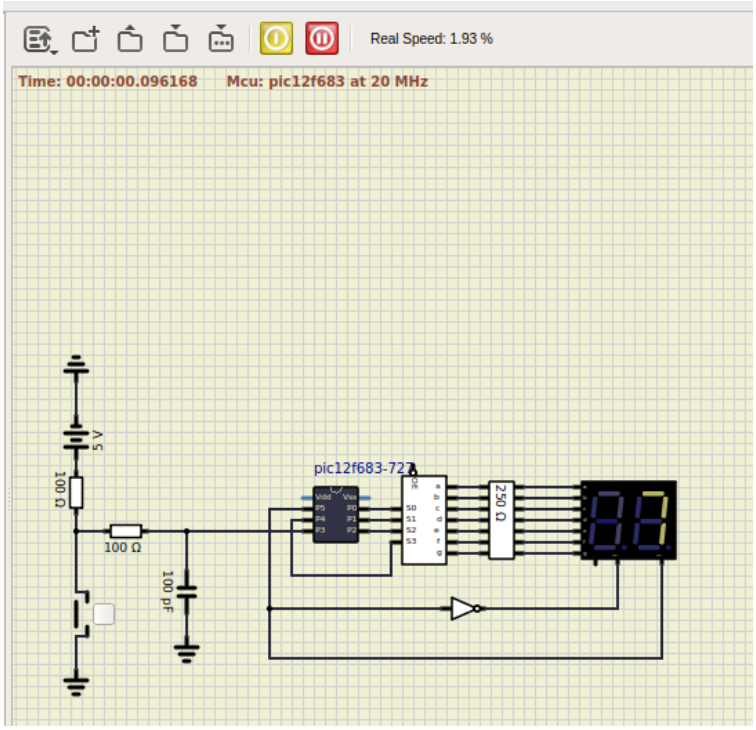


Figura 4: Error de real speed debido a computadora ralentizada. Creación propia

Como se puede ver solo hay un display prendido esto es debido a la lógica que se implemento,sin embargo al correr el circuito en una computadora mas capaz se debería de ver los números aleatorios desplegados correctamente,ya que el circuito se probó la noche anterior y funcionaba correctamente.También el circuito tiene implementado que pare de generar números aleatorios después del 16vo numero generado.
También es necesario hablar sobre los requerimientos del bingo que no se pudieron lograr: El primero es que no se pudieron alojar los números dentro de la memoria EEPROM del MC debido a que no se llegó a la lógica necesaria para lograr implementar esto,pero se sabe que está la función malloc para pic12f683 que permite guardar memoria para el MC,con esto se pudo haber logrado el requerimiento debido a lo anterior tampoco se pudo hacer la comparación para evitar números aleatorios repetidos,sin embargo esto se podía resolver recorriendo un array y comparando los números alojados en la EEPROM.También por cuestión de tiempo no se pudieron crear mediciónes para verificar la corriente que pasa por el display para verificar la Rprotec sin embargo el valor teórico debería funcionar

4. Conclusiones y recomendaciones

Se logró usar el MC PIC12f683 para realizar un bingo que genere 16 numeros pseudo aleatorios mediante la técnica lfsr y se lograron ver las configuraciones basicas de los pines gpio del microcontrolador.Se logró crear una lógica que permitiera crear el bingo a pesar de la poca disponibilidad en pines.

5. Bibliografía

1. Microchip. PIC12F629/675 Data Sheet 8-Pin FLASH-Based 8-Bit CMOS Microcontrollers, February 2003.

6. GIT

Link proyecto: Laboratorio 1

7. Apéndices

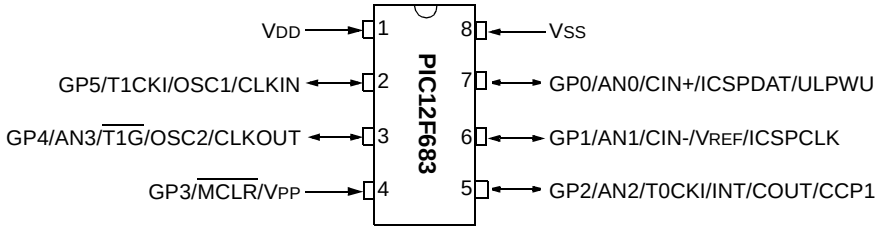
PIC12F683
Data Sheet

8-Pin Flash-Based, 8-Bit CMOS Microcontrollers with nanoWatt Technology

* 8-bit, 8-pin Devices Protected by Microchip's Low Pin Count Patent: U.S. Patent No. 5,847,450. Additional U.S. and foreign patents and applications may be issued or pending.

Pin Diagram

8-pin PDIP, SOIC, DFN-S



1.0 DEVICE OVERVIEW

This document contains device specific information for the PIC12F683. Additional information may be found in the "*PICmicro® Mid-Range MCU Family Reference Manual*" (DS33023), which may be obtained from your local Microchip Sales Representative or downloaded from the Microchip web site. The reference manual should be considered a complementary document to this data sheet and is highly recommended reading for a better understanding of the device architecture and operation of the peripheral modules.

The PIC12F683 is covered by this data sheet. It is available in 8-pin PDIP, SOIC and DFN-S packages. Figure 1-1 shows a block diagram of the PIC12F683 device. Table 1-1 shows the pinout description.

FIGURE 1-1: PIC12F683 BLOCK DIAGRAM

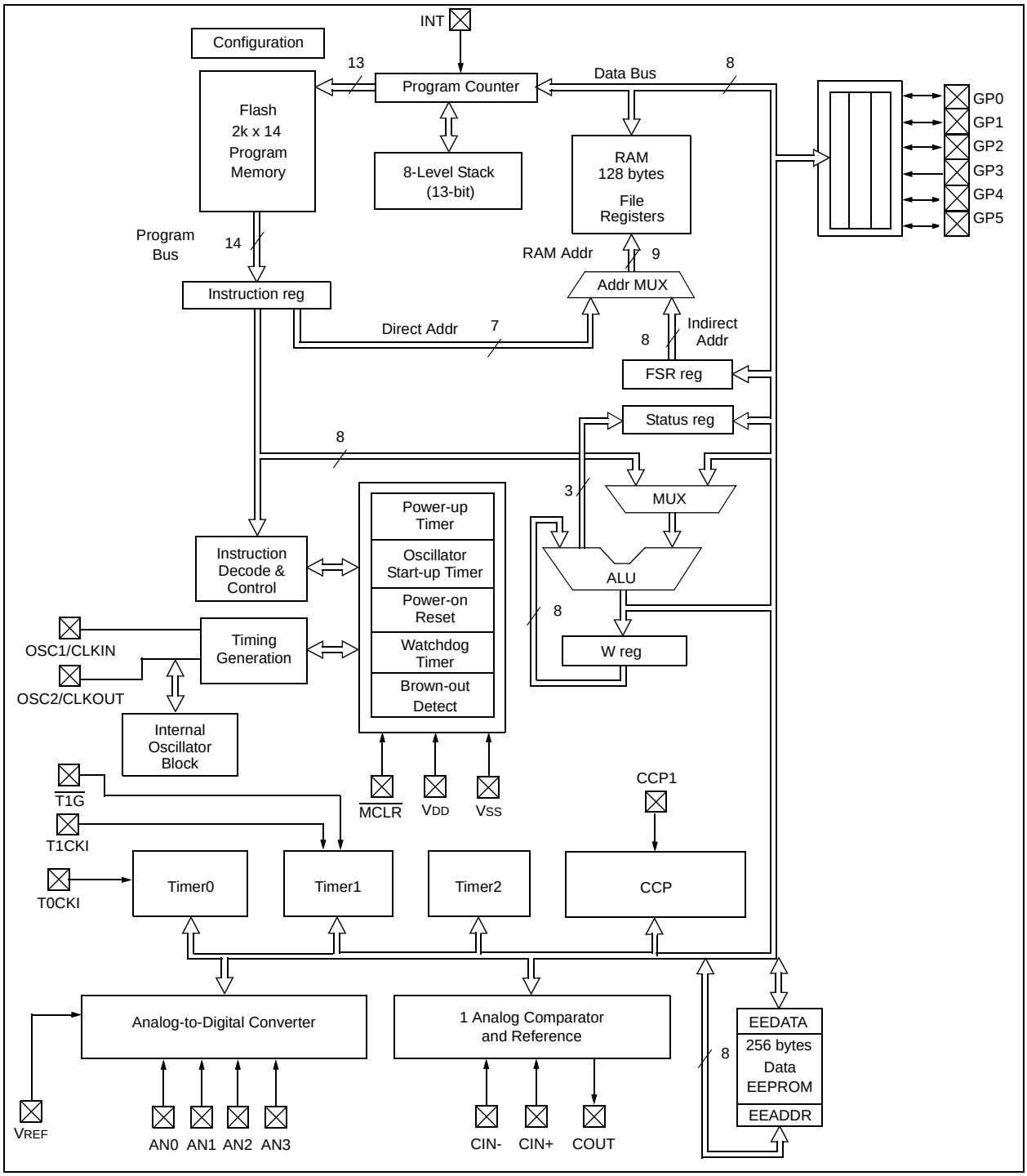


TABLE 1-1: PIC12F683 PINOUT DESCRIPTION

Name	Function	Input Type	Output Type	Description
VDD	VDD	Power	—	Positive supply
GP5/T1CKI/OSC1/CLKIN	GP5	TTL	CMOS	GPIO I/O w/programmable pull-up and interrupt-on-change
	T1CKI	ST	—	Timer1 clock
	OSC1	XTAL	—	Crystal/Resonator
	CLKIN	ST	—	External clock input/RC oscillator connection
GP4/AN3/$\overline{T1G}$/OSC2/CLKOUT	GP4	TTL	CMOS	GPIO I/O w/programmable pull-up and interrupt-on-change
	AN3	AN	—	A/D Channel 3 input
	$\overline{T1G}$	ST	—	Timer1 gate
	OSC2	—	XTAL	Crystal/Resonator
	CLKOUT	—	CMOS	FOSC/4 output
GP3/$\overline{MCLR}$/VPP	GP3	TTL	—	GPIO input with interrupt-on-change
	$\overline{MCLR}$	ST	—	Master Clear w/internal pull-up
	VPP	HV	—	Programming voltage
GP2/AN2/T0CKI/INT/COUT/CCP1	GP2	ST	CMOS	GPIO I/O w/programmable pull-up and interrupt-on-change
	AN2	AN	—	A/D Channel 2 input
	T0CKI	ST	—	Timer0 clock input
	INT	ST	—	External Interrupt
	COUT	—	CMOS	Comparator 1 output
	CCP1	ST	CMOS	Capture input/Compare output/PWM output
GP1/AN1/CIN-/VREF/ICSPCLK	GP1	TTL	CMOS	GPIO I/O w/programmable pull-up and interrupt-on-change
	AN1	AN	—	A/D Channel 1 input
	CIN-	AN	—	Comparator 1 input
	VREF	AN	—	External Voltage Reference for A/D
	ICSPCLK	ST	—	Serial Programming Clock
GP0/AN0/CIN+/ICSPDAT/ULPWU	GP0	TTL	CMOS	GPIO I/O w/programmable pull-up and interrupt-on-change
	AN0	AN	—	A/D Channel 0 input
	CIN+	AN	—	Comparator 1 input
	ICSPDAT	ST	CMOS	Serial Programming Data I/O
	ULPWU	AN	—	Ultra Low-power Wake-up input
VSS	VSS	Power	—	Ground reference

Legend: AN = Analog input or output; CMOS = CMOS compatible input or output; TTL = TTL compatible input; ST = Schmitt Trigger input with CMOS levels; HV = High Voltage; XTAL = Crystal

