Laboratorio de Microcontroladores
Curso UCR

Profesor: Marco Villalta Fallas

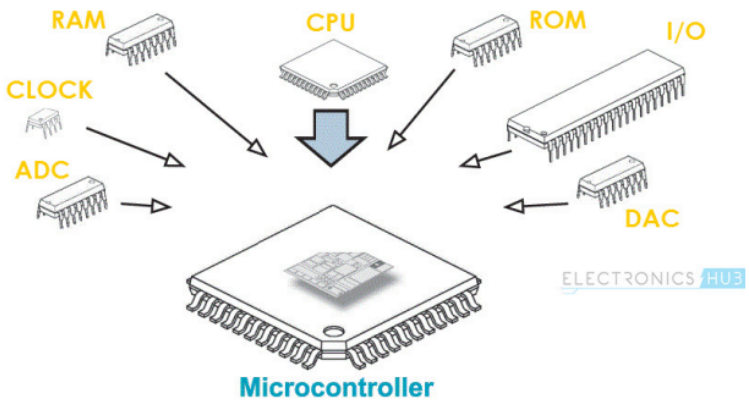
Dualok Fonseca Monge

1. Introducción:

¿Qué es un microcontrolador?

Dispositivo computacional en un solo chip tipo (VLSI-C), conocidos también como Computer on a Chip, también se les llama controladores embebidos porque microcontroladores y circuitos de soporte están integrados.

Tienen memoria RAM, ROM, circuitos de interrupción, puertos generales de entrada y salida (GPIO), comparadores analógicos, contadores, convertidores Analógicos/Digitales, watchdog timers, interfaces de comunicación. Disponibles en diferentes arquitecturas, tamaños de palabra y capacidades.



Estructura básica

- **CPU**: Central processing unit, consta de al menos una ALU y una CU, lee, decodifica y ejecuta instrucciones para realizar operaciones lógicas, aritméticas y de transferencia de datos.
- **Memoria:** -
    - *de programa* (Contiene instrucciones) .
    - *de datos:* guarda la info.
- **I/O ports**: Puertos de E/S, interfaz con el mundo exterior.
- **Buses:** Grupo de conexiones agrupadas entre el CPU y otros periféricos.
- **Contadores/Temporizadores:** Proveen operaciones para contar eventos y tiempo, generaciones de funciones PWM, relojes de control.
- **Puertos seriales:** Interfaces para comunicación con el mundo exterior.
- **Interrupciones:** Proveen mecanismos para atender cambios externos, internos, de hardware y software.
- **Convertidores AC/DC**

Clasificación:

- **Por bits:** Existen microcontroladores de **8 bits** (utilizado para labores básicas); **16 bits**(tiene mayor precisión y desempeño); y **32 bits**(utilizado para aplicaciones avanzadas de control, como aplicaciones médicas, de telecomunicaciones, automatización, etc).
- **Por memoria:** Existen microcontroladores de memoria externa (no tan comunes) y de memoria interna
- **Por conjunto de instrucciones:** RISC (reduced instruction set computer) y CISC (complex instruction set computer)

Diferencias entre microprocesador y microcontrolador

Microcontrolador	Microprocesador
Más lento	Más rápido
Elemento completo y funcional	Requiere de otros periféricos
Diseñado para cumplir tareas puntuales	Tareas de gran capacidad computacional

ARM: (Advanced RISC machines), es más una arquitectura que un microcontrolador o microprocesador. Si tiene RAM y ROM externa se puede considerar un microprocesador, si tiene RAM y ROM interna; un microcontrolador.

Usos de los microcontroladores:

Sensado de luz, temperatura, humedad, velocidad, etc

Dispositivos para controlar procesos, Instrumentación industrial, Robótica, Industria automotriz. IoT.

Tarjetas de desarrollo:

Creadas para facilitar el prototipado, algunos incluyen periféricos y circuitos de alimentación.

Como evitar daños en el MCU:

- ➔ Cortos: Evitar contactos antes de energizar
- ➔ Sobre corriente: Revisar que el consumo de corriente no supere las especificaciones, se puede usar protectores o drivers de corriente
- ➔ Sobre voltaje: Revisar que el adaptador entregue tensión especificada por la placa
- ➔ Voltaje inverso: Revisar la polaridad de adaptador y conexiones
- ➔ Componentes externos: Revisar funcionamiento y conexión
- ➔ Malas prácticas: Tener lugar de trabajo limpio, fuentes de alimentación estables

Microcontroladores en IE-0624:

Microcontroladores: PIC, AVR, Arduino, STM32, ESP32.

Lenguajes: C, C++, Python.

Plataformas: SimulIDE(Simulador), Wokwi(Simulador), STM32F429 Discovery kit(real), Arduino Nano BLE33(real).

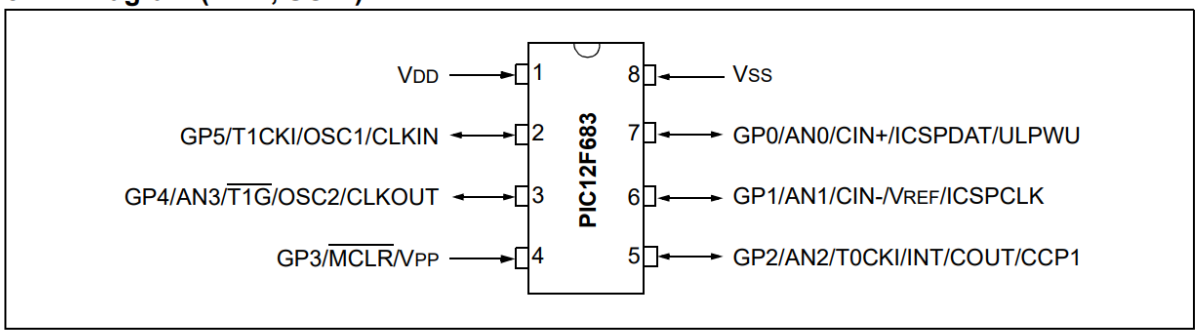
2. PIC12F683 y GPIOs:

GPIO: General purpose input/output

La gran mayoría de los pines en un MCU son GPIO, los pines que no son GPIO son VDD, VS, CLK, algunos pines de GPIO comparten otras funciones, se agrupan en puertos, en el caso del PIC12F683 utiliza un único puerto.

PIC12F683

8-Pin Diagram (PDIP, SOIC)



Cómo funcionan los GPIO:

Generalmente se configura el funcionamiento del GPIO a través de un registro

Si se configura como entrada se realiza la lectura con polling o por medio de interrupciones, si se configura como salida, se maneja el estado en el programa: 1 Alto, 0 bajo.

Registros importantes:

En este microcontrolador hay 6 GPIO y varios registros para dictar cómo funcionan estos pines.

- GPIO: Es un puerto de 6 bits bidireccional, este registro 1 es alto, 0 es bajo.

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

bit 7-6 **Unimplemented**: Read as '0'

bit 5-0 **GP<5:0>**: GPIO I/O Pin bit
1 = Port pin is > VIH
0 = Port pin is < VIL

- TRISIO: Configura el flujo de datos de los GPIO, el tercer bit de GPIO siempre se lee como un input, por lo que el TRISIO(3) siempre lee un 1. 1 es un input y 0 es un output

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

bit 7-6 **Unimplemented**: Read as '0'

bit 5:4 **TRISIO<5:4>:** GPIO Tri-State Control bit
1 = GPIO pin configured as an input (tri-stated)
0 = GPIO pin configured as an output

bit 3 **TRISIO<3>:** GPIO Tri-State Control bit
Input only

bit 2:0 **TRISIO<2:0>:** GPIO Tri-State Control bit
1 = GPIO pin configured as an input (tri-stated)
0 = GPIO pin configured as an output

- CMCON: Registro donde se configura el comparador analógico, se debe modificar si se quiere configurar los pines GPIO0, GPIO1, GPIO2 como entradas.
    - NOTA: ANSEL y CMCON0 deben ser configurados si se quiere tratar un canal analógico como una entrada digital. Los pines configurados como analogicos leen "0"

- ANSEL: Registro donde se configura la conversión y seleccion analogica, se debe modificar si se quiere utilizar 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

bit 7 **Unimplemented**: Read as '0'

bit 6-4 **ADCS<2:0>:** A/D Conversion Clock Select bits
000 = FOSC/2
001 = FOSC/8
010 = FOSC/32
x11 = FRC (clock derived from a dedicated internal oscillator = 500 kHz max)
100 = FOSC/4
101 = FOSC/16
110 = FOSC/64

bit 3-0 **ANS<3:0>**: Analog Select bits
Analog select between analog or digital function on pins AN<3:0>, respectively.
1 = Analog input. Pin is assigned as analog input(1).
0 = Digital I/O. Pin is assigned to port or special function.

Note 1: Setting a pin to an analog input automatically disables the digital input circuitry, weak pull-ups and interrupt-on-change, if available. The corresponding TRIS bit must be set to Input mode in order to allow external control of the voltage on the pin.

- CONFIG

Especificaciones eléctricas:

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 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 ..... 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

Note 1: Power dissipation is calculated as follows: $P_{DIS} = V_{DD} \times \{I_{DD} - \sum I_{OH}\} + \sum \{(V_{DD} - V_{OH}) \times I_{OH}\} + \sum (V_{OL} \times I_{OL})$.

† NOTICE: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at those or any other conditions above those indicated in the operation listings of this specification is not implied. Exposure above maximum rating conditions for extended periods may affect device reliability.

Hola PIC código:

Hola PIC

```
#include <pic14/pic12f683.h>
typedef unsigned int word;
word __at 0x2007 __CONFIG = (_BODEN_OFF);
void delay (unsigned int tiempo);
void main(void)
{
    TRISIO = 0x00;//Poner todos los pines como salidas
    GPIO = 0x00;//Poner todos en bajo
    unsigned int time = 100;

    //Loop forever
    while ( 1 ){
                GPI00 = 0x00;//Esto se puede hacer tmb con GP0=0x00(no recomendado) o enmascarando
                delay(time);
                GPIO0 = ~GPIO0;//Esto es con el metodo de campo de bit
                delay(time);
    }
}

void delay(unsigned int tiempo)
{
        unsigned int i;
        unsigned int j;

        for(i=0;i<tiempo;i++)
          for(j=0;j<1275;j++);
}
```

Configuración de los simuladores y ambiente de trabajo:

Repositorio de Git: https://github.com/Dualock/Microcontroladores-Labs

Se va a trabajar con el simulador SimulIDE, al inicio no podía compilar el ejecutable, hubo que instalar libfuse con el comando:

```
sudo apt install libfuse2
```

También se instaló sdcc con los comandos:

```
cd ~

mkdir tmp

cd tmp,
```

extrayendo el archivo sdcc. Luego ejecutando

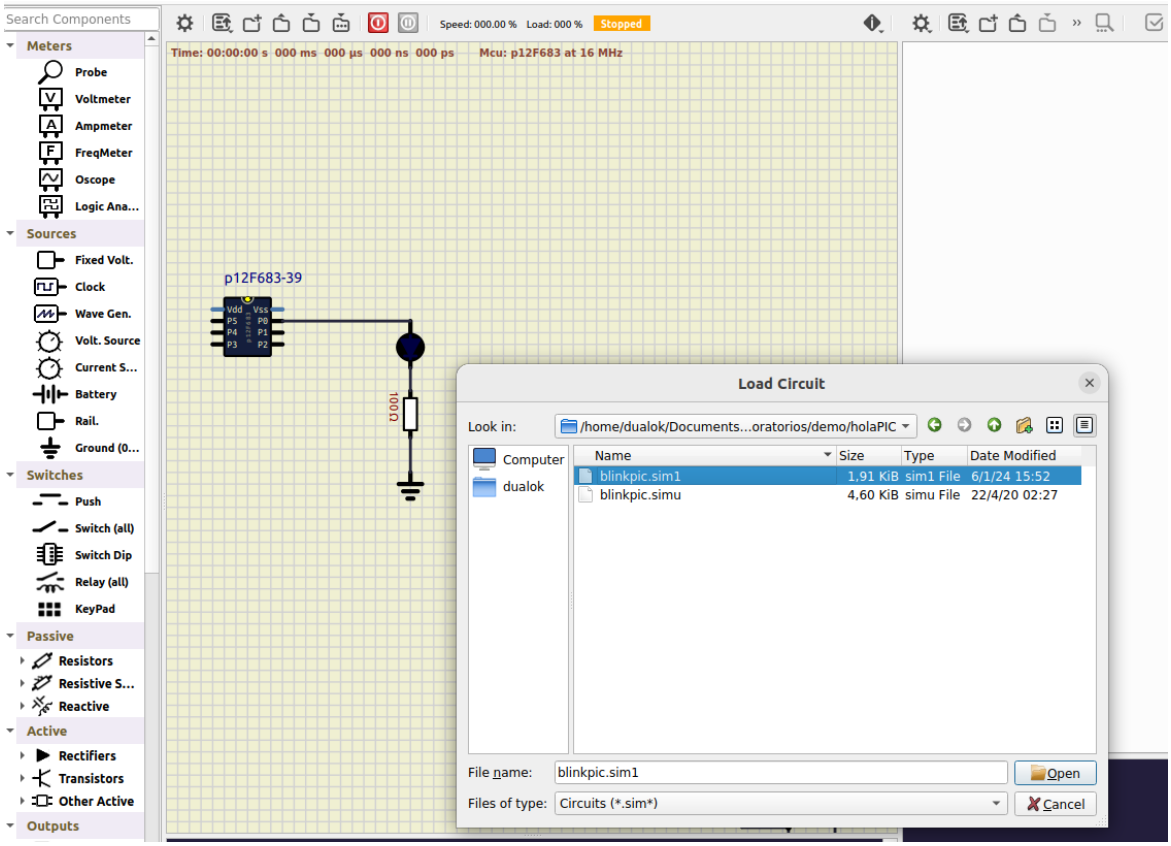
```
cd sdcc-4.0.0

cp -r * /usr/local
```

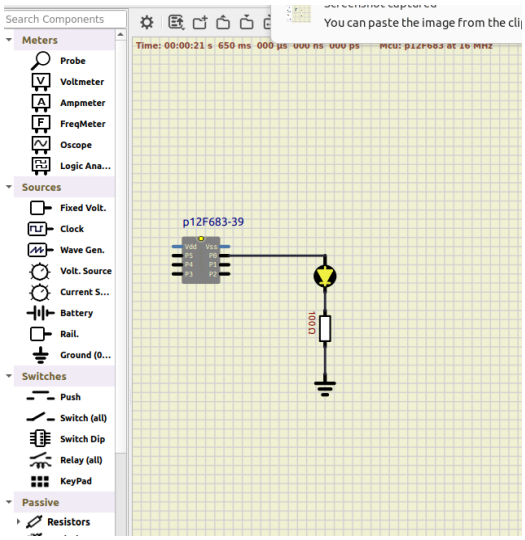
Para verificar que se instalaron correctamente, se ejecuta:

```
sdcc -v
```

Cómo utilizar el simulador: Se ejecuta el archivo ejecutable de simulIDE, en la opción open circuit, se abre el archivo.simu correspondiente, en este caso se abrió el blink.simu



Luego se selecciona el microcontrolador y se hace click en la opción load firmware y se carga el archivo .hex. Antes de esto es necesario haber programado el firmware en C y utilizar el makefile para generar el archivo.hex. Una vez cargado el firmware se comienza la simulación.



Evaluaciones para los laboratorios:

Evaluacion propuesta

- Introducción 5
  - Resumen
  - Conclusiones importantes
- Nota teórica 20
  - Información general MCU
  - Periféricos
  - Registros
  - Diseño de circuito
  - Lista de componentes y precios
  - Conceptos/temas del laboratorio
- Desarrollo/análisis 50
  - Análisis programa
  - Análisis electrónico
- Conclusiones y recomendaciones 10
- Bibliografia 5
- Git/avances 10

2.1 Laboratorio 1 - GPIOs:

Figura 2: Pantalla de 2 digitos

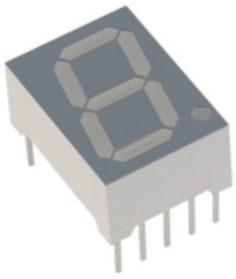
Desarrollara un simulador de tombola simplificada de bingo como el de la figura 1 utilizando dos displays de 7 segmentos para formar la pantalla de la figura 2, un botón, el microcontrolador PIC12F675/PIC12F683 y cualquier otro componente que considere necesario. El objetivo de este laboratorio es introducir al estudiante al manejo de GPIOs (*General Purpose Input Outputs*), generación de números aleatorios y el flujo de desarrollo que se pide para las prácticas dirigidas. Cada vez que se presiona el boton debe desplegarse en el display un número que represente a una de las bolas de la tombola 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 considere que la tombola solo tendra 10 bolas, despues de sacar 10 bolas/numeros debera parpadear tres veces la pantalla con el número 99 y reiniciar el juego.

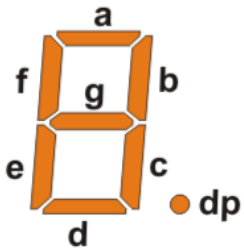
Solución:

Para realizar este laboratorio, se propone dividir el problema en varios pasos, los cuales incluyen: investigación, diseño del circuito y del software.

- ☑ ~~**LED de 7 segmentos**~~

  Se trata de varios LEDS en la misma carcasa de plástico, donde de acuerdo a la configuración, se pueden escribir números (0 al 9) y letras. Para este laboratorio nos interesa con que configuracion de 8 bits se genera cada número





Digits to display	Display Segments							
	dp	a	b	c	d	e	f	g
0	0	1	1	1	1	1	1	0
1	0	1	1	0	0	0	0	0
2	0	1	1	0	1	1	0	1
3	0	1	1	1	1	0	0	1
4	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1
6	0	1	0	1	1	1	1	1
7	0	1	1	1	0	0	0	0
8	0	1	1	1	1	1	1	1
9	0	1	1	1	1	0	1	1

☐ Cantidad de GPIO disponibles como salidas

Según el registro GPIO, se tienen 6 puertos disponibles, pero solo 5 son salidas, ya que el puerto GP3 solo se puede configurar como entrada. Es decir, hay que idear una forma de encender ambos LEDs de 7 segmentos con esas 5 salidas, dígase GP0, GP1, GP2, GP4 y GP5, esto se discute en la siguiente sección.

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

bit 7-6 **Unimplemented**: Read as '0'

bit 5-0 **GP<5:0>**: GPIO I/O Pin bit
1 = Port pin is > VIH
0 = Port pin is < VIL

☐ Técnicas a utilizar

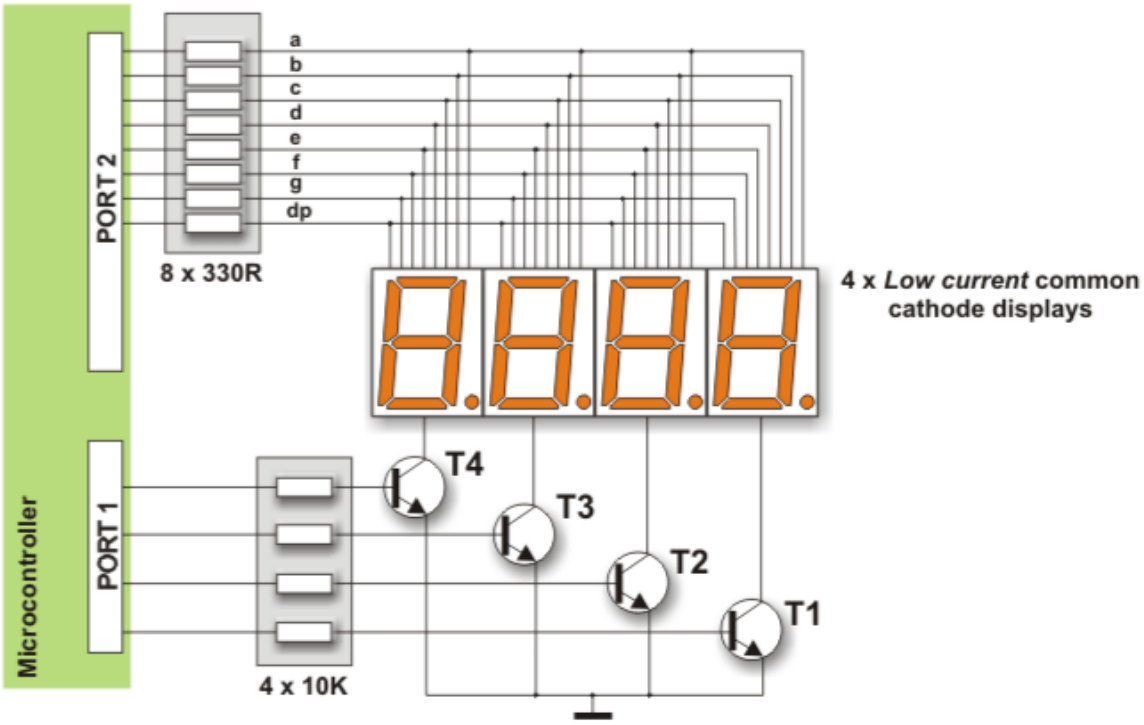
Convertidor BCD a 7 Segmentos:

Se va a utilizar un convertidor de BCD a 7 Segmentos, esto permite mapear valores en BCD (4 bits) a sus correspondientes configuraciones de LED de 7 segmentos. De esta forma se tendrán 4 salidas para encender los LEDs, 1 entrada para el botón de push y sobra un puerto por si lo llegara a necesitar. Los números decimales codificados en binario son:

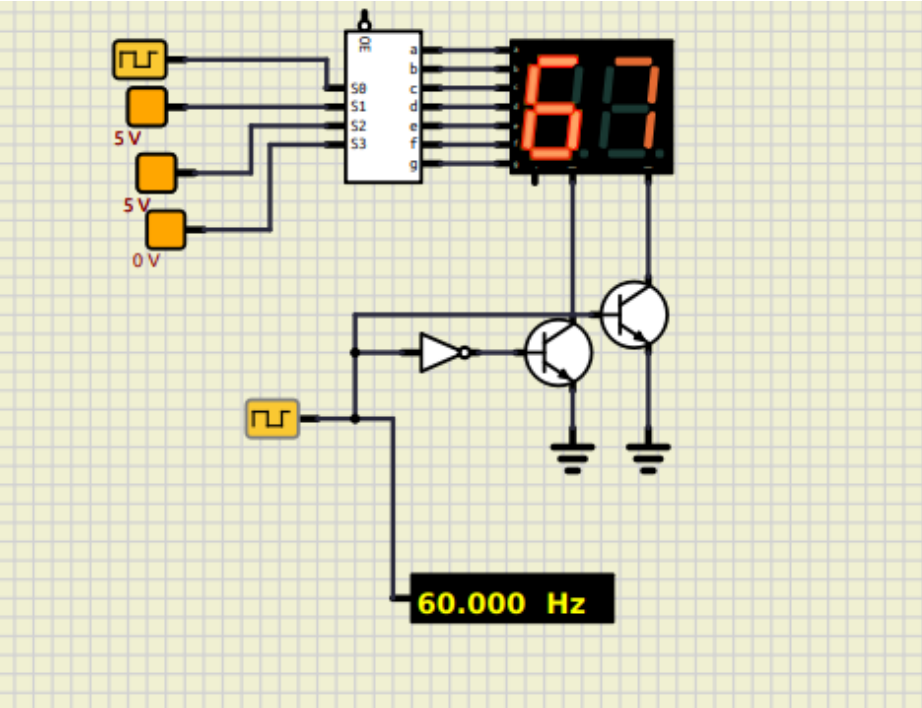
0000: 1	0011: 3	0110: 6	1001: 9
0001: 1	0100: 4	0111: 7	
0010: 2	0101: 5	1000: 8	

Multiplexación:

Se utilizará la técnica de multiplexación para dar la ilusión de que ambos displays de LED están encendidos al mismo tiempo, pero están conmutando con una frecuencia de 60hz haciendo uso de dos transistores BJT.



Como forma de probar este funcionamiento, se creó una simulación con un valor cambiante con la misma frecuencia de conmutación como se muestra en la figura



☐ **Generacion de numeros aleatorios**

Para la generación de números aleatorios se hace uso de un linear feedback shift register

☐ **Comprobación de números repetidos**

ss

☐ **Modificación de registros y macros necesarios del PIC**

☐ **Lectura del estado del botón:**

☐ **Diagrama de flujo del programa:**

☐ **Codificación completa del firmware**

☐ **Circuito completo**

Microcontroladores

Curso Online Edx

Link: https://learning.edx.org/course/course-v1:UTAustinX+UT.6.10x+3T2022/home

Dualok Fonseca Monge

What's an embedded system:

an embedded system is a combination of electrical, mechanical, optical, chemical, computer software, all combined for a dedicated purpose, it has a computer embedded inside

Examples of embedded systems:

pacemakers: helps heart beat at a regular rate, it purpose is to help people live longer

hard drive: it has mechanical parts that rotate, electrical parts which control the motor and optical parts to read and write data from the disk

Voltage detector: allows to monitor the conditions of the operation of any system that you're testing, it has a rich interface that allows you to track information.

Motor controller: it has an interface to control the motor so it can be used in larger applications.

Game controller: it uses mechanical switches aka buttons to interface the game with the player.

Ipad: Music player

Numbers representation: (I already know this shit)

Base 2 system (Binary):

01101010 = 2^6+2^5+2^3+2^1 = 106

Hexadecimal system:

0xhhhhh

0x2a9f = 2x16^3 + 10x16^2 + 9x16^1 + 15x16^0 = 10911

Converting to binary: 0010 1010 1001 1111

hex numbers are used to represent addresses, contents of reg memory

A nibble is defined as a 4 binary digits or 1 hex number, hexadecimal digits are represented by adding $ or 0x before the number. In C we will add the 0x

Hex Digit	Decimal Value	Binary Value
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
A or a	10	1010
B or b	11	1011
C or c	12	1100
D or d	13	1101
E or e	14	1110
F or f	15	1111

Table 2.1. There are 16 hexadecimal digits.

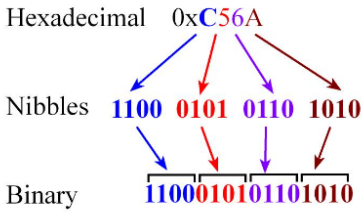


Figure 2.2. Notice that each hex digit maps into 4 binary bits.

Signed vs unsigned numbers

Signed: Min ------ 0 -------- Max

Unsigned: 0 --------- Max

Example:

Unsigned 10001101: 128 + 8 + 4 + 1 = 141

Signed: 10001101: The MSB represents the sign of the number, 0 positive, 1 negative.

Precision and bytes

A byte consist of 8 bits

b7|b6|b5|b4|b3|b2|b1|b0

= 128*b7 + 64*b6 + 32*b5 + 16*b4 + 8*b3 + 4*b2 + 2*b1 + b0

Precision is the number of distinct or different values, we express precision in alternatives, bytes or bits. Alternatives is the total number of possibilities. For ex; an 8 bit ADC can generate 256 different analog outputs

Example	Precision in alternatives	Precision in bits
Fingers	10	a little more than 3
Months	12	between 4 and 5
Human vertebrae	33	a little more than 5
Cards in a deck	52	between 5 and 6
Days in a year	365-366	between 8 and 9
Minutes in a day	1440	between 10 and 11
Men in a legion	5000	a little more than 12

2s complement

flip every bit and then add 1, its the same as left every bit as it is now, from LSB to MSB until we find the first 1, then we flip the rest excluding that first 1

For ex:

What's the complement and the signed and unsigned value of 10011000?

Complement = (10011000)* =01100111 + 1 = 01101000

Unsigned Value = 128+16+8 = 152

Signed Value = 64+32+8 = -104

Another way to calculate it is = -128+16+8 = -104 -> its **way easier** and faster for god sake!

Exercises:

1. Give the representations of the decimal **45** in 8-bit binary and hexadecimal

   45/2 = 22/2 = 11/2 = 5/2 = 2/2 = 1

   1 0 1 1 0 1

   ans = bin = 0010 1101 hex = 0x2D

2. Convert the signed binary number **11011010** to decimal.

-128+64+16+8+2 = -38

Words and halfwords

A word on the ARM cortex M will have 32 bits. There is 2^32 different unsigned bit numbers, the smallest is 0 and the largest is about 4 million

A halfword or double byte contains 16bits. Similar to the unsigned algorithm we can use the basis to convert a decimal number into a signed binary

Number	Basis	Need it	bit	Operation
-100	-128	**yes**	bit 7=**1**	subtract -100 - -128
28	64	no	bit 6=0	none
28	32	no	bit 5=0	none
28	16	**yes**	bit 4=**1**	subtract 28-16
12	8	**yes**	bit 3=**1**	subtract 12-8
4	4	**yes**	bit 2=**1**	subtract 4-4
0	2	no	bit **1**=0	none
0	**1**	no	bit 0=0	none

Table 2.2. Example conversion from decimal to signed 8-bit binary.

so basically, if we have a negative number we will always use the MSB as 1, so will need -128+16+8+4 in order to generate a 100

-128 64 32 16 8 4 2 1

1 0 0 1 1 1 0 0

When dealing with numbers in a computer it's convenient to memorize some powers of 2

exponent	decimal
2^0	1
2^1	2
2^2	4
2^3	8
2^4	16
2^5	32
2^6	64
2^7	128
2^8	256
2^9	512
2^{10}	1024 about a thousand
2^{11}	2048
2^{12}	4096
2^{13}	8192
2^{14}	16384
2^{15}	32768
2^{16}	65536
2^{20}	about a million
2^{30}	about a billion
2^{40}	about a trillion

Here's the way to specify them in a C language

size	unsigned	signed
8 bits	`unsigned char`	`signed char`
16 bits	`unsigned short`	`short`
32 bits	`unsigned long`	`long`
64 bits	`unsigned long long`	`long long`

In C99, the number of bits has been standardized. In C99, we have these data types

size	unsigned	signed
8 bits	`uint_8`	`int_8`
16 bits	`uint_16`	`int_16`
32 bits	`uint_32`	`int_32`
64 bits	`uint_64`	`int_64`

Fields of usage of embedded systems

Cars, military, industry, people, phones, house, consumer electronics,

Components of an embedded system

Computer x86, ARM, memory, I/O(Input output) Interface -> has a lot of components - Hardware - Electrical - Software

Abilities that are extremely important

Testing, Power management, profit, sizing

The idea of time is extremely important, we need to have the correct answer at the right time.

Microcontrollers

to understand the term embedded microcomputer system consider each word separately

Micro = Small - Embedded = hidden inside - Computer: has a processor and memory and means to exchange data with the real world.