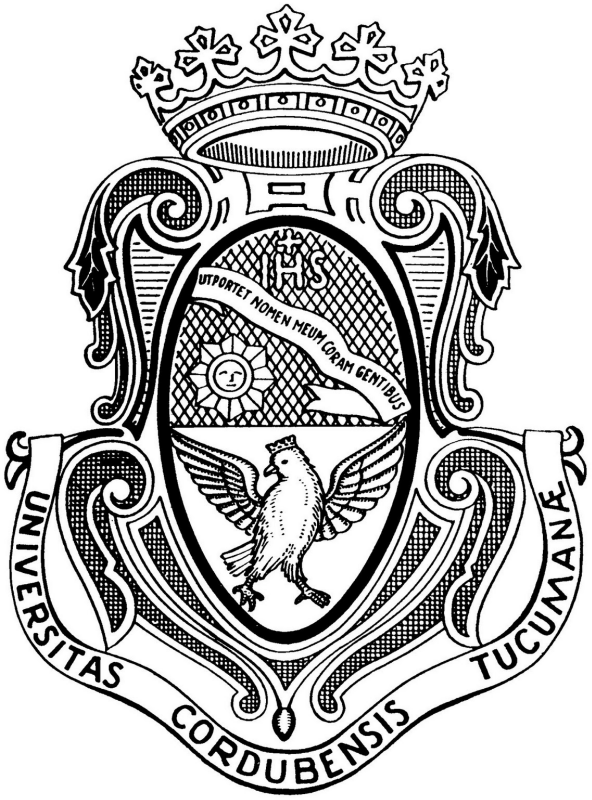
**Universidad Nacional de Córdoba**
**Facultad de Ciencias Exatas, Físicas y Naturales**




# Primer Trabajo Práctico

Electrónica Digital II

Docente: Ing. Martín Del Barco

Losano Quintana, Juan Cruz
Piñero, Tomás Santiago
Ingeniería en Computación

Año 2019

# Índice

# 1. Enunciado

Realizar un programa que permita sumar dos valores de cuatro bits cada uno, ingresados desde puertos configurados como entradas. El resultado debe mostrarse mediante cuatro LEDs conectados a puertos configurados como salidas. En caso de producirse un acarreo del tipo "*Digit Carry*", un quinto LED empezará a parpadear indefinidamente con un periodo de aproximadamente un segundo prendido y un segundo apagado. A partir de ese instante ya no podrán realizarse más sumas, salvo que se realice un reset del microcontrolador.

Armar en la protoboard el circuito totalmente funcional para ser presentado y evaluado en clase.

Adjuntar una foto de la hoja que muestre el diagrama de todas las conexiones realizadas en el diseño con el cálculo del valor de las resistencias limitadoras de los puertos de salida y el cálculo del tiempo de parpadeo de los bucles anidados utilizados para la frecuencia de reloj elegida.

# 2. Introducción

Para la realización del trabajo práctico se utilizaron los siguientes materiales:

- Microcontrolador PIC16F887.
- Cristal de 4 MHz.
- LEDs de color verde, amarillo y rojo.
- Resistencias de 1 kΩ, 330 Ω y 220 Ω.
- Capacitores de 22 pF y 100 nF.
- Dos Dip Switch (DS) de cuatro llaves.
- Un pulsador.

Al ser dos números de cuatro bits, el resultado será de cinco bits en el peor de los casos, por lo tanto se pueden utilizar dos puertos del PIC para implementar el circuito, ya que éste posee cuatro puertos de ocho bits.

Se seleccionó el puerto B como puerto de entrada de ambos números y el puerto A como salida para mostrar el resultado de su suma.

# 3. Desarrollo

## Cálculos de resistencias

Antes de realizar la implementación del circuito se deben calcular las resistencias para las salidas del PIC.

### Pull-ups

Las resistencias *pull-up* son utilizadas en circuitos digitales para asegurar en cualquier circunstancia un nivel lógico seguro y definido en una determinada entrada o pin digital. Los estados lógicos son tres:

1. Alto (*High*): también llamado *uno lógico*, representa la presencia de voltaje.
2. Bajo (*Low*): también llamado *cero lógico*, representa la ausencia de voltaje.
3. Flotante (*Floating*): estado de *alta impedancia*, desconectado del resto del circuito.

Las resistencias pull-up se utilizaron para el pulsador de reset.

**Reset** En este caso las resistencias cumplen una doble función:

1. Limitar la corriente en el pin.
2. Impide la conexión directa del pin a $V_{cc}$ cuando se presiona el pulsador, evitando así un cortocircuito.

Se utilizó el circuito recomendado en hoja de datos del PIC para el *Master Clear* ($\overline{MCLR}$) en la sección 14 *Special Features of the CPU*, subsección 2.1 *Power-on Reset (POR)*, página 213.

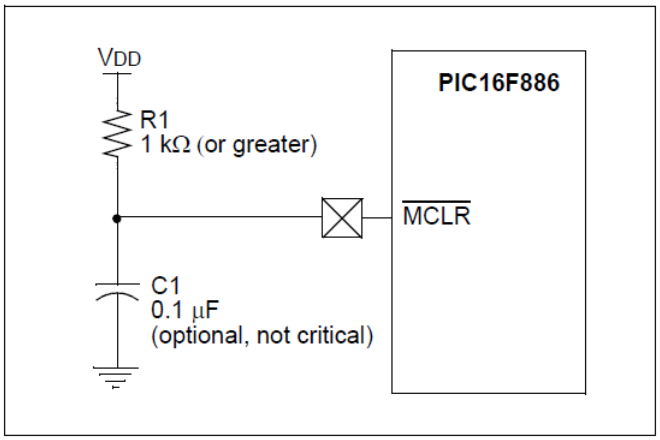



Figura 1: Circuito recomendado para el $\overline{MCLR}$

**Puerto de entrada** Se eligió el puerto B como entrada debido a que es el único de los cinco puertos que tiene pull-up integrado, evitando así el cálculo e implementación de resistencias adicionales.

Estos pull-up internos deben habilitarse vía software desde el bit siete ($\overline{RBPU}$) del registro de opciones ($OPTION\_REG$) y configurarse por medio del registro $WPUB$.

## LEDs

Las resistencias para cada LEDs se calculan según la *Ley de Ohm* y teniendo en cuenta la tensión máxima de salida del puerto en alto $V_{OH}$. Según la hoja de datos, este valor de tensión es de $V_{cc} - 0{,}7V$ (ver sección 17 *Electrical Specifications*, subsección 5 *DC Characteristics*, página 251).

Como el circuito se alimenta con 5 V, el valor será $V = V_{OH} = 4{,}3V$. Por lo tanto el circuito para los LEDs queda de la siguiente manera:

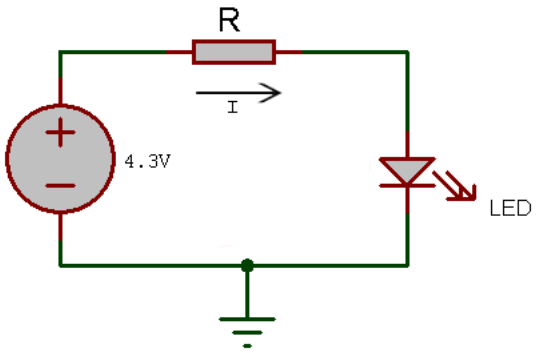



Figura 2: Circuito LED

La corriente I de la Figura **2** deseada para cada LED es de 10 mA, por ende las ecuaciones quedan:

$$\begin{aligned} 4{,}3V &= V_{LED} + V_R \\ 4{,}3V &= V_{LED} + I * R \\ 4{,}3V &= V_{LED} + 10mA * R \end{aligned}$$

$$R = \frac{4{,}3V - V_{LED}}{10mA} \tag{1}$$

El valor de las resistencias que se pondrán como resultado es el normalizado, redondeado, en lugar del calculado.

**LED Rojo** El LED de color rojo se activa con una tensión de 1.7 V, por lo tanto, la ecuación **1** queda:

$$R = \frac{4{,}3V - 1{,}7V}{10mA}$$

$$R = 330\,\Omega \tag{2}$$

**LED Verde** El LED de color verde se activa con una tensión de 2.2 V, por lo tanto, la ecuación **1** queda:

$$R = \frac{4{,}3V - 2{,}2V}{10mA}$$

$$R = 220\,\Omega \tag{3}$$

# 4. Implementación

El PIC tiene como frecuencia de reloj un cristal de 4 MHz, por lo que el ciclo de instrucción se realiza con una frecuencia de 1 MHz debido a que cada 4 ciclos del reloj se realiza una instrucción. Esto se debe a que para ejecutar la instrucción indicada, el PIC debe ejecutar cuatro acciones:

1. Buscar la instrucción en la memoria principal.
2. Decodificar la instrucción.
3. Ejecutar la instrucción.
4. Almacenar los resultados.

Esto es importante para el cálculo de la subrutina de retardo, ya que depende de la frecuencia de reloj que se utilice.

A continuación se muestran los diagramas de flujo del programa utilizado.

## Diagramas de flujo

En esta sección se muestran los diagramas de flujo del programa principal y las subrutinas que utiliza.

### Programa principal

Primero se realiza la configuración de los puertos A y B como salida y entrada digitales, respectivamente.

Una vez configurados los puertos se toman los datos del puerto de entrada y a esa lectura se la invierte, ya que cuando los Dip Switch estén bajos las entradas estarán con un valor de uno lógico debido a la presencia de las resistencias pull-up.

Consecuentemente el programa almacena los últimos cuatro bits en la variable *numero1* y se lo suma a los primeros cuatro bits leídos, mostrando el resultado en los LEDs de salida. Si el resultado es de cinco bits, el LED del *digit carry* parpadeará y no se podrá realizar otra suma hasta que se resetee el PIC.

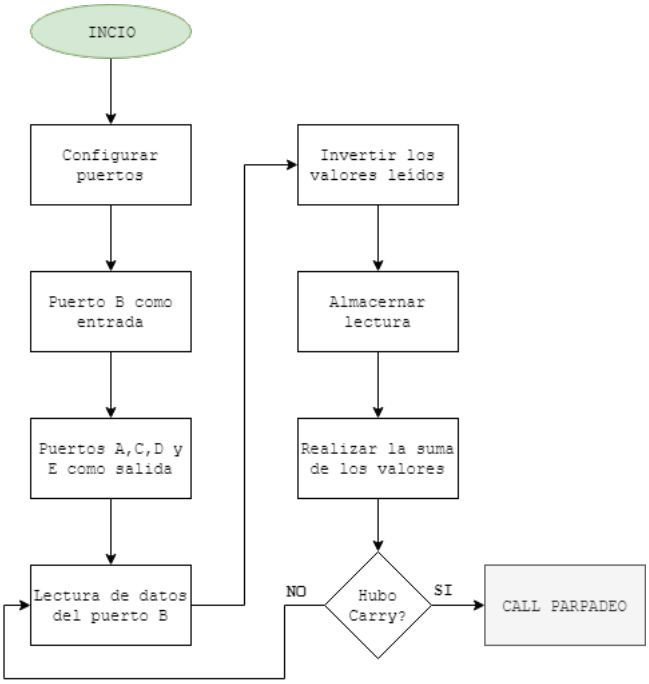



Figura 3: Diagrama de flujo del programa.

## Parpadeo

Es una subrutina muy simple que se encarga solamente de encender y apagar el LED correspondiente al *digit carry* con un retardo de aproximadamente un segundo.

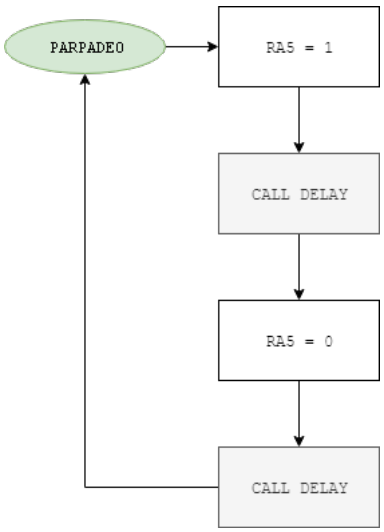



Figura 4: Diagrama de flujo de la subrutina *Parpadeo*.

### Delay

Como se mecionó anteriormente, esta subrutina demora aproximadamente un segundo en ejecutarse, por lo que se utiliza para el parpadeo del *digit carry*.

El cálculo para su realización es el siguiente:

$$4\,\mu s * (cuenta - 1\,\mu s) + 5\,\mu s = 1\,000\,000\,\mu s$$

$$cuenta = \frac{1\,000\,000\,\mu s}{4\,\mu s} - 4\,\mu s$$

$$cuenta = 249996 \tag{4}$$

Al ser registros de ocho bits, un contador solamente puede llegar hasta 256, por lo que para llegar al valor de *cuenta* calculado se utilizarán tres contadores:

1. C1 = 4
2. C2 = 244
3. C3 = 250

Al anidar estos tres contadores se llegará a una cuenta de 244000, llegando aproximadamente al segundo necesario.

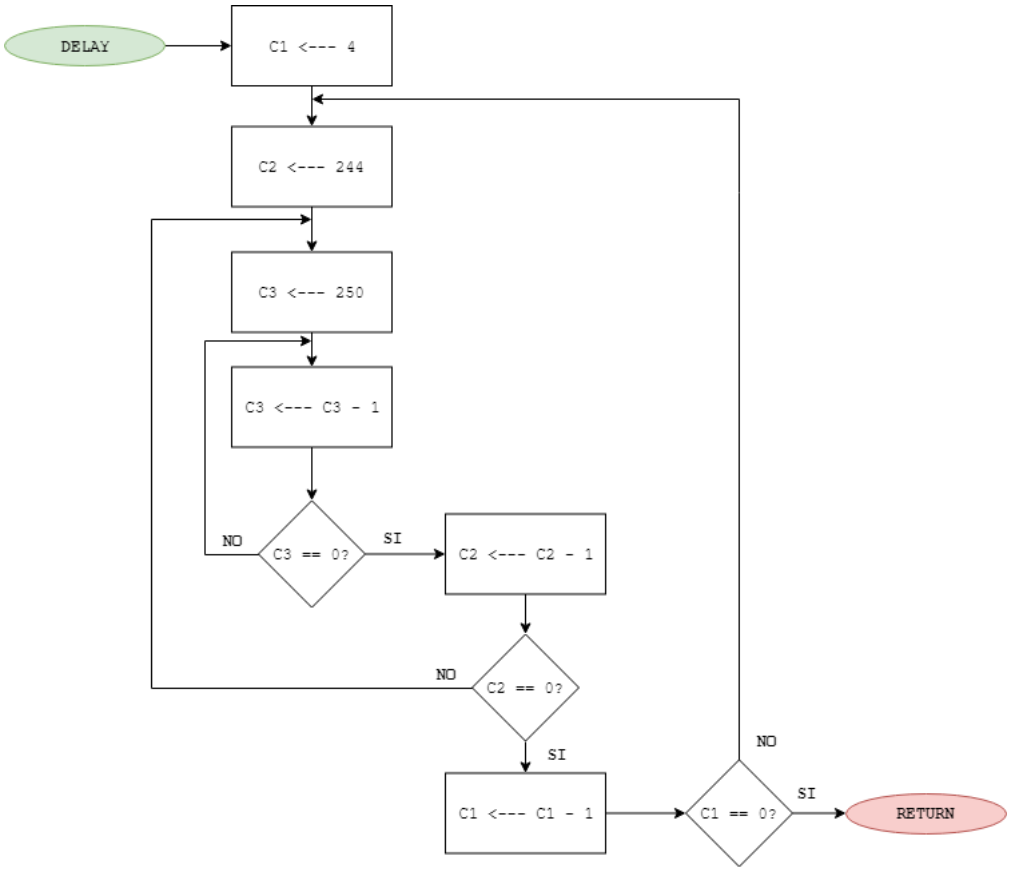



Figura 5: Diagrama de flujo de la subrutina *Delay*.

# 5. Esquema del circuito