# Multiplicador 4x4 en PIC16F628A

Cabe mencionar que el resultado de multiplicar dos números de 4 bits cada uno, requiere al menos 8 bits para el caso máximo que se daría en $15_{10}$ ($1111_2$) x $15_{10}$ ($1111_2$) = $225_{10}$ ($11100001_2$).
El PIC16F628 tiene 2 puertos de 8 bits cada uno, con la restricción de que en el caso del puerto A sólo se puede utilizar el RA5 (MCLR) como entrada y no como salida; los restantes pines de pueden utilizar de manera bidireccional. Para el PIC16F84, sólo se dispone del puerto B completo y de únicamente 5 bits del puerto A.
Buscando que el siguiente código sea compatible con ambos microcontroladores, se utilizan los 4 bits menos significativos del puerto A (RA3 a RA0) para introducir los números a multiplicar, uno detrás del otro. Con el bit número 4 del puerto A (RA4) se da un *Enter* para introducir cada número.

El multiplicador binario por sumas sucesivas tiene la particularidad de que un número sirve para las sumas continuas y el otro se comporta como un contador decremental; por ejemplo, supongamos que multiplicaremos 5 x 4.
El primer número es 5 y se asigna en una localidad de memoria. El segundo número es 4 y se asigna a otra localidad. El algoritmo básico consiste en sumar 5 + 5 y depositar el resultado nuevamente en la misma localidad, a la vez que se decrementa el 4 en 1. El resultado se obtiene hasta que el contador sea cero. Así, es posible sumar 5+5+5+5 (4 veces) para obtener 20.

## ¿Cómo funciona el diseño?

Observa el siguiente diagrama de conexiones. Éste se diseñó para un PIC16F628 configurando su oscilador interno. En el caso del PIC16F84, no se dispone de un oscilador interno, por lo que tienes que conectar tu cristal oscilador de 4MHz de forma externa. Otra diferencia importante es que el PIC16F84 no tiene los comparadores analógicos del PIC16F628 por lo que no debes incluir en tu código, las líneas que deshabilitan los comparadores:

```
clrf   pa
movlw  7
movwf  cmcon
```

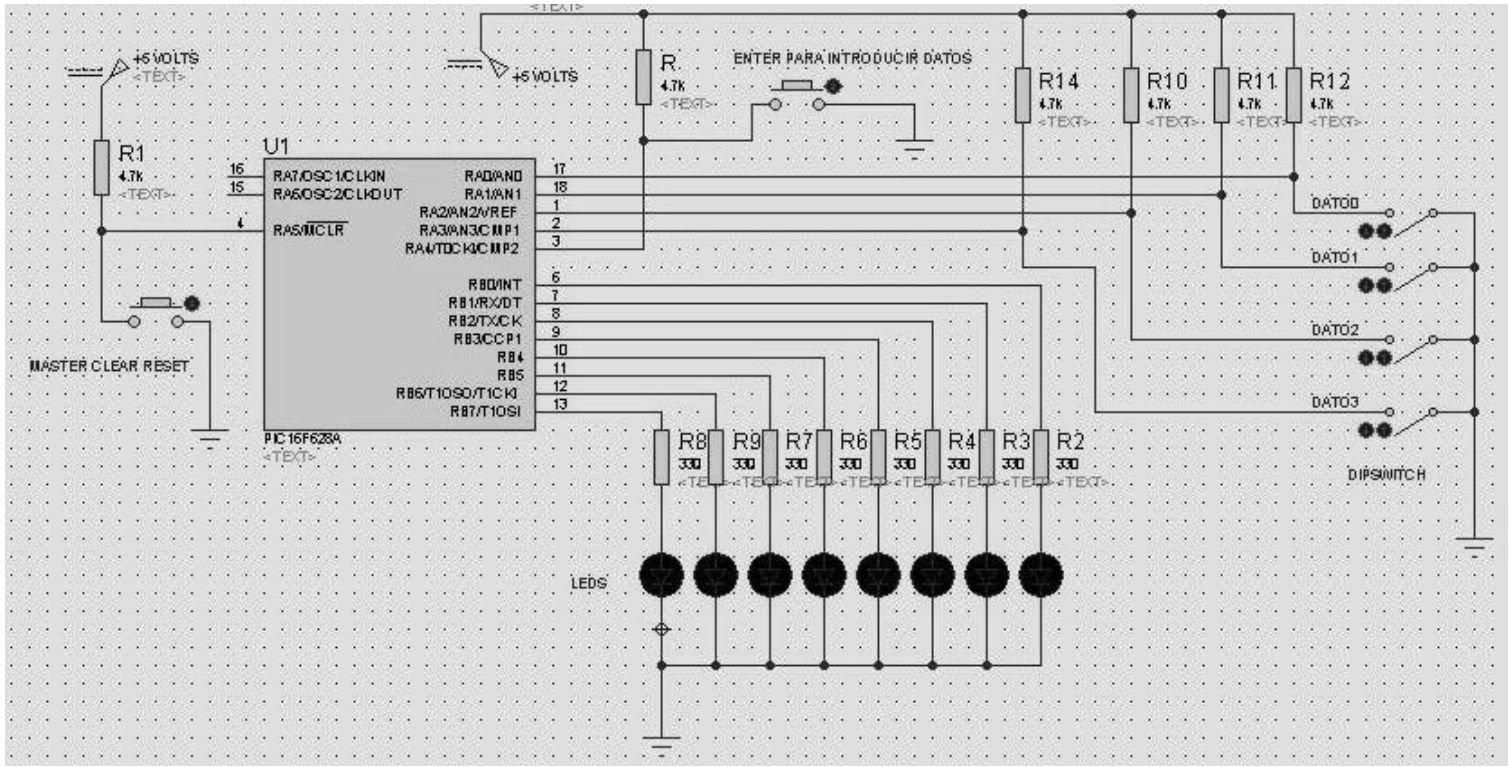
Las resistencias a los LEDs son de 330 Ohms y las propias para los push buttons y el dipswitch son de 4.7KOhms. Recuerda que la alimentación del PIC no está indicada (pin 5, GND y pin 14, Vcc; revisa la hoja de especificaciones). Al momento de programarlo, revisa que el programador universal esté habilitando el oscilador interno y el master clear.

El pin 4 del PIC es RA5 que funciona como Reset general. Los datos se introducen por RA3 (bit 3) hasta RA0 (bit 0).

Al iniciar la aplicación, los 8 leds del puerto B deben estar apagados. Con el dipswitch introducirás el primer número a multiplicar, recordando que RA3 es el bit más significativo y RA0 el menos significativo, de tu número. Tienes que *presionar en una sola ocasión* el push button de *Enter* conectado fisicamente a RA4 (pin 3 del PIC).
Ahora debes introducir el segundo número, moviendo los interruptores de tu dipswitch; *presiona en una sola ocasión* *Enter* para validar el segundo dato y automáticamente obtener el resultado.
El multiplicador, entonces, requiere solamente dos presiones de botón para obtener el resultado; al segundo botonazo se introduce el segundo dato y a la vez se da el resultado..
La aplicación vuelve a su estado inicial, en espera de que introduzcas nuevos números.

```
;**********multipic.asm*******
;Diseño de un contador ascendente-descendente, por jcrls.
;Colocar LEDs es los 8 bits del puerto B (RB7..RB0)
;Conectar un push button en el bit 0 del puerto A (RA0).
;Recuerda que si vas a utilizar el oscilador interno, debes configurarlo en
;el menú CONFIGURE -> CONFIGURATION BITS ->settings (INTRC I/O) de MPLAB.
;*****************************************************************
;*****************************************************************************
list p=16f628a
__CONFIG _CP_OFF & _WDT_OFF & _PWRTE_ON & _INTRC_OSC_NOCLKOUT & _LVP_OFF & _MCLRE_ON
#include <p16f628a.inc>
;*****************************************************************************
estado  equ     0x03  ;Dirección del registro de estado en el banco 0
pb      equ     0x06  ;Dirección del puerto en ambos bancos
pa      equ     0x05
rp0     equ     0x05  ;bit 5 en STATUS
rp1     equ     0x06  ;bit 6 en STATUS
cmcon   equ     0x1F  ;Dirección del registro CMCON en el banco 0
i               equ     0x20
```

```
i1      equ     0x21
num1    equ     0x22
num2    equ     0x23
        org     0           ;Origen
        goto    ini          ;Salto incondicional, obligatorio
        org     5
ini     clrf    pa
        movlw   7
        movwf   cmcon   ;Deshabilitación de comparadores, Puerto A. No válido para el 16F84.
        bcf     estado, rp1     ;inicia configuración
        bsf     estado, rp0
        clrf    pb          ;Puerto B como salidas
        movlw   0x1F
        movwf   pa  ;Los 5 bits menos significativos de A como entrada, RA4 es "enter"
        bcf     estado, rp0     ;Regreso a trabajo normal
        clrf    pb          ;Inicialización del PB con "00000000"
boton   btfsc   pa,4    ;¿RA4 = 0? Entonces se valida que se guarde operando1, si no me voy a boton
        goto    boton           ;Espero por push botton
        call    retardo
        movfw   pa                      ;Leo dato del puerto A
        movwf   num1            ;Asigno el dato a la localidad num1
        movlw   0x0F            ;Cargo en el registro W = 00001111 para hacer una AND
        andwf   num1, 1 ;con num1 y sólo respetar los pines menos significativos.
                        ;el resultado lo dejo en num1
boton2  btfsc   pa,4    ;¿RA4 = 0?  Se valida que se guarde operando2, si no me voy a boton2
        goto    boton2
        call    retardo
        movfw   pa      ;Leo puerto para asignar num2
        movwf   num2
        movlw   0x0F
        andwf   num2, 1
        movlw   0               ;Comienza multiplicador por sumas sucesivas
opera   addwf   num1,0          ;Comienzo num1=num1+num1 hasta que
        decfsz  num2            ;num2 sea 0, decrementándose
        goto    opera
        movwf   pb
        goto    boton           ;Regreso para multiplicar otros datos
retardo clrw                    ;Rutina de retardo
        movlw   0xFF
        movwf   i1
otro1   movwf   i
otro    decfsz  i, 1
        goto    otro
        decfsz  i1, 1
        goto    otro1
        return
        end
```


+5 VOLTS
+5 VOLTS
R
4.7k
ENTER PARA INTRODUCIR DATOS
R14
4.7k
R10
4.7k
R11
4.7k
R12
4.7k
R1
4.7k
U1
RA7/OSC1/CLKIN
RA6/OSC2/CLKOUT
RA5/MCLR
RA0/AN0
RA1/AN1
RA2/AN2/VREF
RA3/AN3/CMP1
RA4/T0CKI/CMP2
RB0/INT
RB1/RX/DT
RB2/TX/CK
RB3/CCP1
RB4
RB5
RB6/T1OSO/T1CKI
RB7/T1OSI
PIC16F628A
DATO0
DATO1
DATO2
DATO3
DIPSWITCH
MASTER CLEAR RESET
R8
330
R9
330
R7
330
R6
330
R5
330
R4
330
R3
330
R2
330
LEDS