

# Sistemas Digitales: Introducción a los microcontroladores

Ingeniería Electrónica  
UPC 2018

Por Kalun Lau



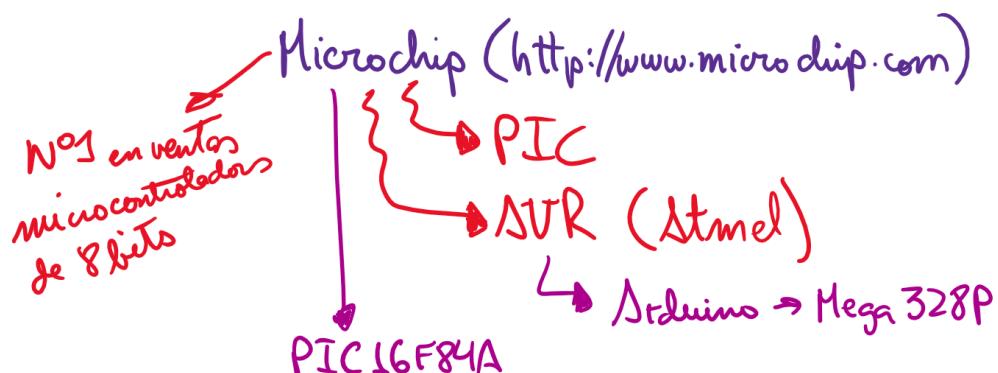
- Microcontroladores
- Dispositivos programable "todo en uno"
- para desarrollar aplicaciones en electrónica,
- Es "la solución en un chip"  
↓  
muy específicos

Ej:



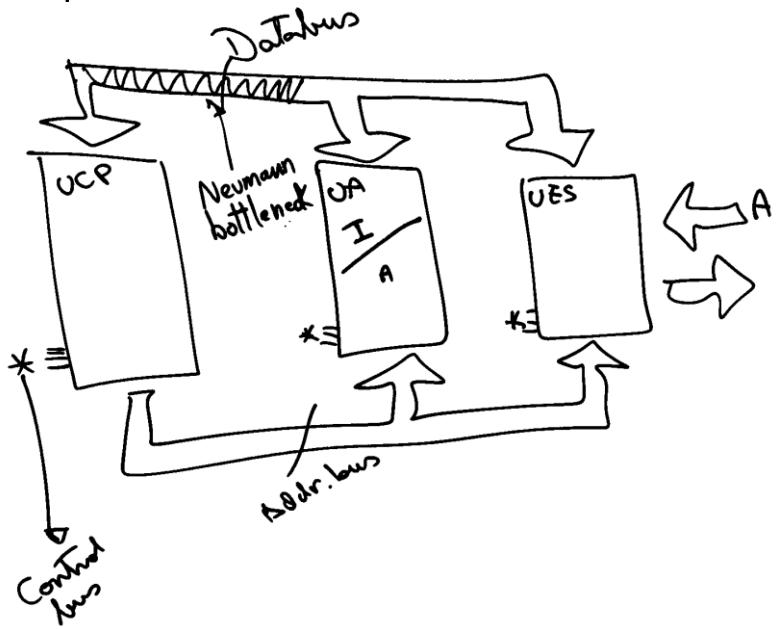
- Tiene un montón de periféricos:
  - Mouse - Sound card
  - Keyboard - Video card
  - HDD - DVD
  - Ethernet - USB

Fabricantes: (Microchip, Renesas, NXP, ST, etc)

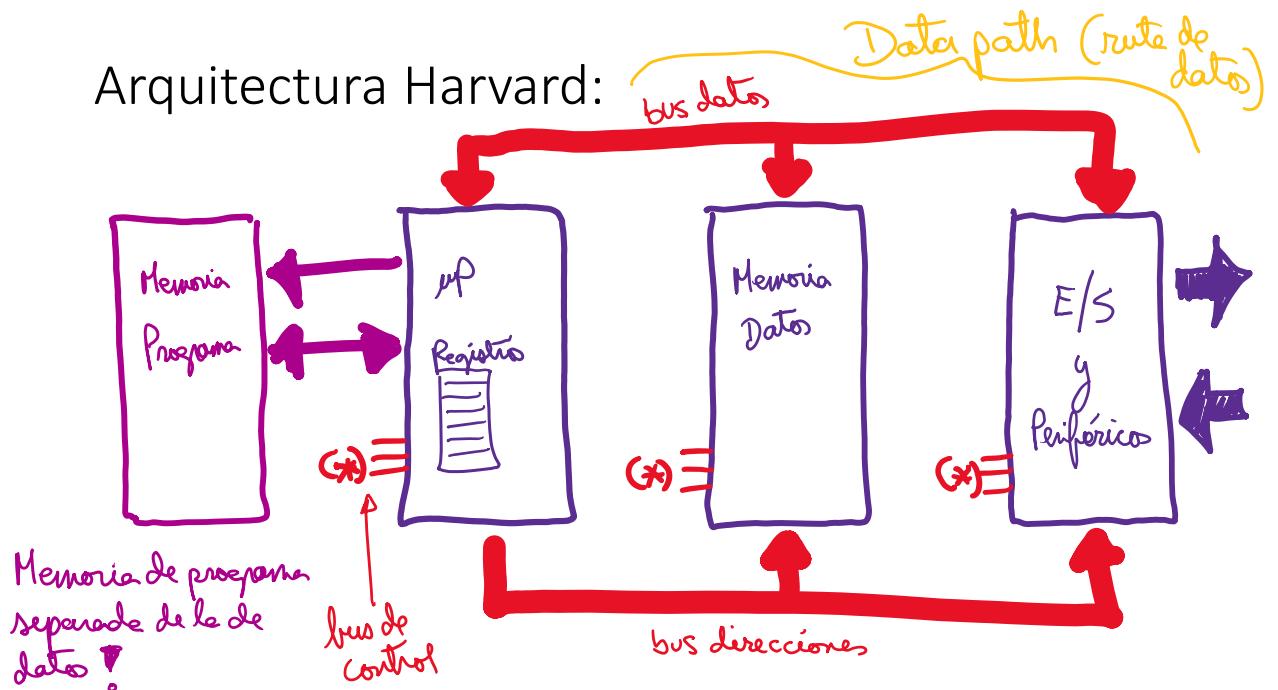


- ↳ se desarrollaron muchas aplicaciones (AN)
- ↳ bajo costo
- ↳ muchos desarrolladores lo emplearon para aplicaciones de control de procesos industriales y comerciales

## Arquitectura Von Neumann

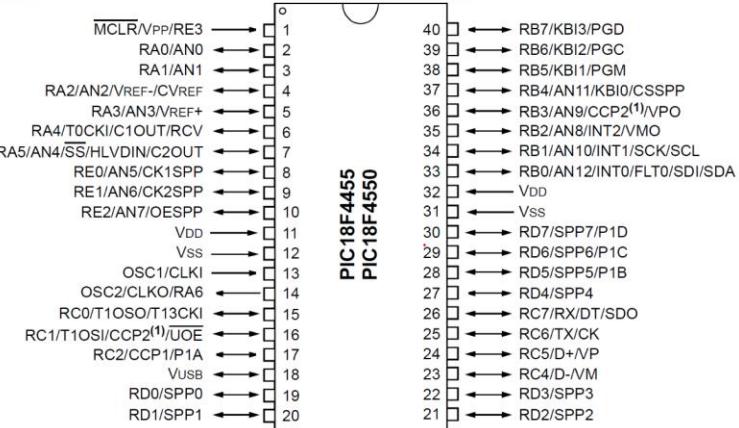


## Arquitectura Harvard:



Nosotros usaremos el PIC18F4550!

40-Pin PDIP



Tecnología  
CMOS

$$\begin{aligned} V_{DD} &= 5V \\ V_{SS} &= 0V \end{aligned}$$

$$\begin{aligned} V_{DD} \text{ mínimo:} \\ 3.0V \text{ (F)} \\ 2.0V \text{ (LF)} \end{aligned}$$

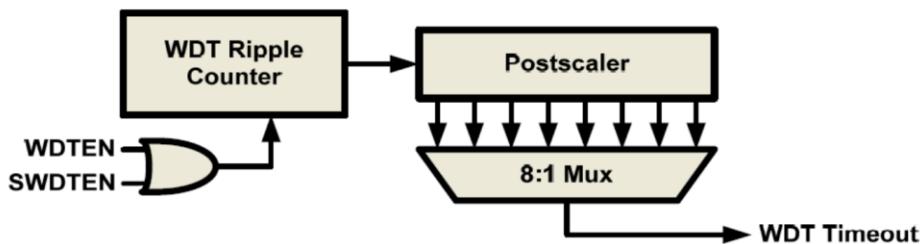
$$\begin{aligned} V_{DD} \text{ máximo:} \\ 5.5V \text{ (F y LF)} \end{aligned}$$

## Características Especiales PIC18F

- Amplio Voltaje de operación: 2.0V a 5.5V
- Memoria de Programa Flash Mejorada con 100,000 ciclos de borrado/escritura
- Memoria de Datos EEPROM con 1,000,000 de ciclos de borrado/escritura
- Retención de Datos en Memoria EEPROM Flash/Data : 100 años típico

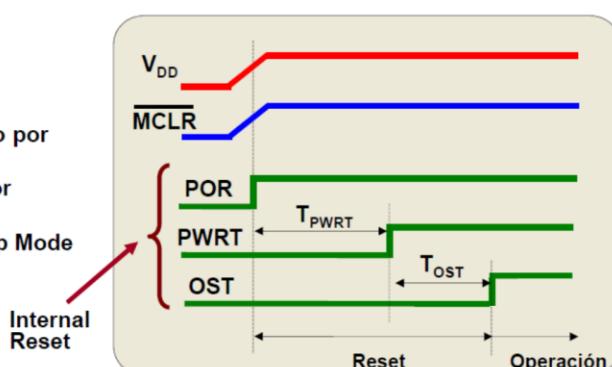
## Watchdog Timer

- Ayuda al software a recuperarse de un mal funcionamiento
- Usa un oscilador libre RC en el chip
- WDT es borrado por la instrucción CLRWDT
- WDT Habilitable (WDTEN) no puede ser borrado por soft
- el overflow (desborde) del WDT reactiva al chip
- Período del timeout programable : 18ms a 3.0s típico
- Opera en modo SLEEP; sobre el time out, despierta la CPU.



## Generador de Reset Interno

- POR: Power On Reset
  - con MCLR atado a  $V_{DD}$ , es generado un pulso de RESET cuando el flanco de subida a  $V_{DD}$  es detectado
- PWRT: Power Up Timer
  - 72 ms (nominal)
  - Desacoplado del BOR
- OST: Oscillator Startup Timer
  - Mantiene en RESET al dispositivo por 1024 cyclos de maquina (TCYs)
  - Le permite al cristal o al resonador estabilizarse
  - Bypaseado en: Two Speed Startup Mode
    - INTOSC usa un clock para el procesador no estable



## BOR –Brown Out Reset

- Cuando el voltaje cae por debajo de un umbral particular, el dispositivo se pone en RESET
- Impide el funcionamiento irregular o inesperado
- Elimina la necesidad de un circuito externo BOR

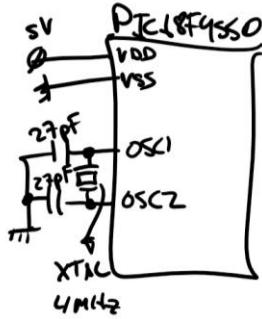
## PBOR – Programmable Brown Out Reset

- Opción de configuración (fijado en tiempo de programa)
  - No puede ser activado/desactivado por software
- Cuatro puntos de disparo BVDD seleccionables
  - 2.5V – Mini V<sub>DD</sub> para OTP MCUs PICmicro®
  - 2.7V
  - 4.2V
  - 4.5V
- Para otros umbrales use un supervisor externo (MCP1xx, MCP8xx/TCM8xx, or TC12xx)

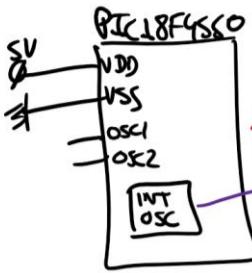
## Fuentes de reloj:

- Cristal externo:

4 MHz, 8 MHz, 10 MHz  
20 MHz hasta 48 MHz



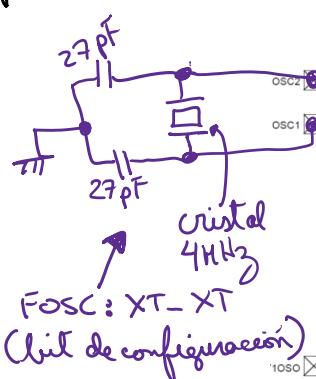
- Oscilador interno: 8 MHz configurable



oscilador RC

varía su precisión con la temperatura

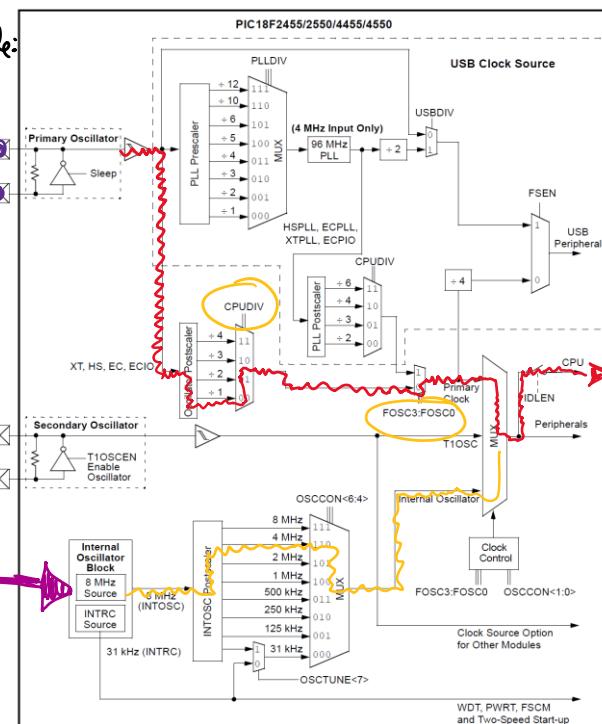
Configuración inicial a detalle:



FOSC: XT - XT  
(bit de configuración)

Opción para usar el oscilador interno:

FOSC: INTOSC - EC  
(bit de configuración)



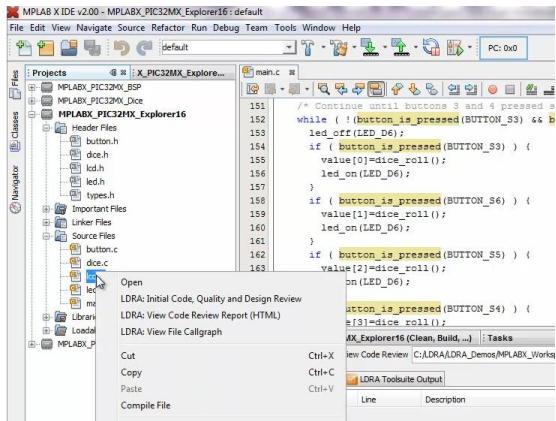
PLL: Sistema que permite multiplicar la frecuencia del cristal externo.

Ojo: Solo se puede usar una fuente de reloj

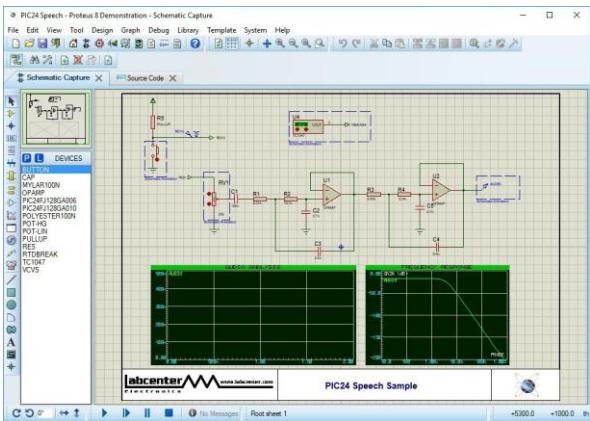
¿Cómo se programa?

- Lenguaje ensamblador (MPASM) ✓
- PBasic compiler (melabs.com, USD 249.99)
- C → XC8 (microchips.com)
  - CCS C Compiler
  - mikroC compiler (mikroe.com)

Herramientas de desarrollo para el microcontrolador PIC18F4550

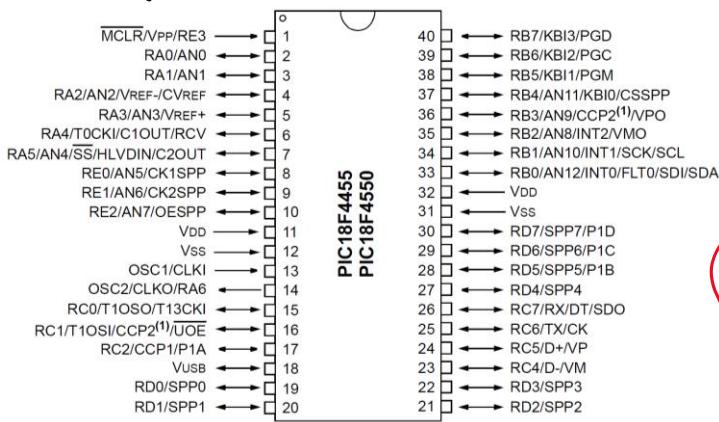


Microchip:  
MPLAB X  
(última versión: 5.05)  
([www.microchip.com/mplab-x-ide/](http://www.microchip.com/mplab-x-ide/))

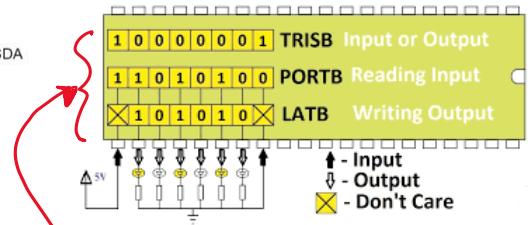


Labcenter:  
Proteus VSM (virtual system modelling)  
se desarrollaría el código  
se harán las simulaciones

## Manejo de puertos de E/S en el microcontrolador PIC18F4550:

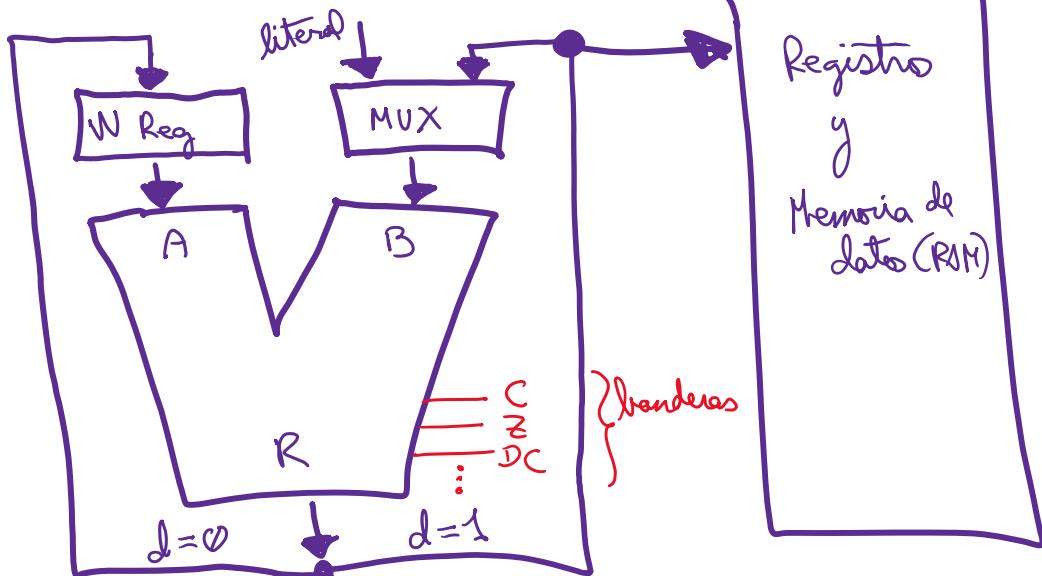


TIP: Declarar si es E o S antes de usar el puerto.



- En modo digital E/S: R<sub>X</sub>Y<sub>Y</sub> (where X is the port letter and Y is the port number)
- Algunos puertos son o solo "E" o solo "S".

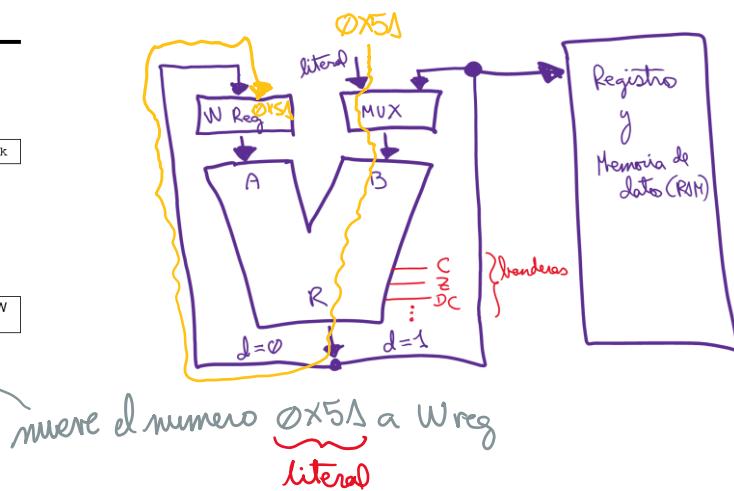
## Ruta de datos del PIC18F4550



# MPASM: Instrucciones de movimiento de datos

<b>MOVLW</b>		Move literal to W
Syntax:	[label]	MOVLW k
Operands:	0 ≤ k ≤ 255	
Operation:	k → W	
Status Affected:	None	
Encoding:	0000 1110 kkkk kkkk	
Description:	The eight-bit literal 'k' is loaded into W.	
Words:	1	
Cycles:	1	
Q Cycle Activity:		
	Q1 Q2 Q3 Q4	
	Decode Read literal 'k' Process Data Write to W	

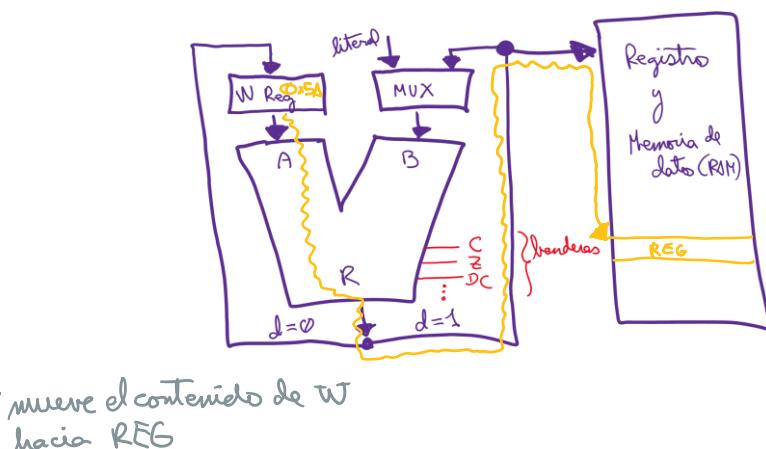
Example: MOVLW 0x5A  
After Instruction  
W = 0x5A



# MPASM: Instrucciones de movimiento de datos

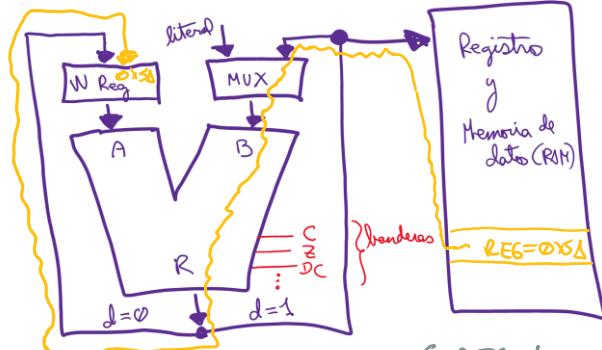
<b>MOVWF</b>		Move W to f
Syntax:	[label]	MOVWF f[,a]
Operands:	0 ≤ f ≤ 255	
	a ∈ {0,1}	
Operation:	(W) → f	
Status Affected:	None	
Encoding:	0110 111a erre erre	
Description:	Move data from W to register f. Location 'f' can be anywhere in the 256 byte bank. If 'a' is 0, the Access Bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).	
Words:	1	
Cycles:	1	
Q Cycle Activity:		
	Q1 Q2 Q3 Q4	
	Decode Read register f Process Data Write register f	

Example: MOVWF REG, 0  
Before Instruction  
W = 0x4F  
REG = 0xFF  
After Instruction  
W = 0x4F  
REG = 0x4F



# MPASM: Instrucciones de movimiento de datos

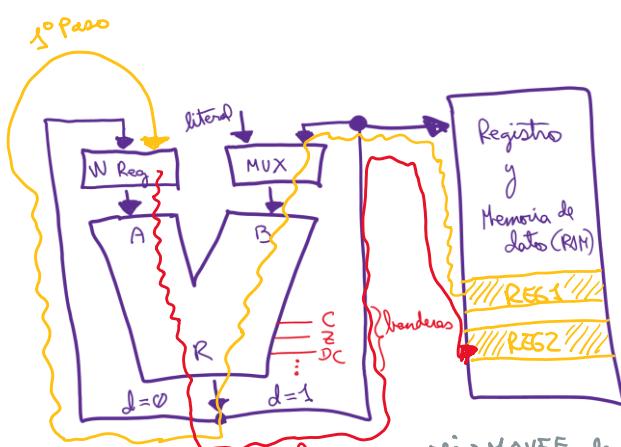
MOVF	Move f
Syntax:	[label] MOVF f,[d,a]
Operands:	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]
Operation:	f → dest
Status Affected:	N, Z
Encoding:	0101 00da ffff ffff
Description:	The contents of register 'f' are moved to a destination dependent upon the status of 'd'. If 'd' is 0, the result is placed in W. If 'd' is 1, the result is placed back in register 'f' (default). Location 'a' can be any where in the 256 byte bank. If 'a' is 0, the Access Bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).
Words:	1
Cycles:	1
Q Cycle Activity:	Q1 Q2 Q3 Q4 Decode Read register f' Process Data Write W
Example:	MOVF REG1, 0, 0
Before Instruction	REG = 0x22 W = 0xFF
After Instruction	REG = 0x22 W = 0x22



Mueve el contenido de REG1 hacia W (d=0) / si movf REG,1: moverá el contenido hacia si mismo!  
Actualizará el estado de los banderas!

# MPASM: Instrucciones de movimiento de datos

MOVFF	Move f to f
Syntax:	[label] MOVFF f <sub>s</sub> ,f <sub>d</sub>
Operands:	0 ≤ f <sub>s</sub> ≤ 4095 0 ≤ f <sub>d</sub> ≤ 4095
Operation:	(f <sub>s</sub> ) → f <sub>d</sub>
Status Affected:	None
Encoding:	1100 1111 ffff ffff ffff f <sub>s</sub>
Description:	The contents of source register f <sub>s</sub> are moved to destination register f <sub>d</sub> . Location of source f <sub>s</sub> can be anywhere in the 4096 byte data space (000h to FFFFh), and location of destination f <sub>d</sub> can also be anywhere from 000h to FFFFh. Either source or destination can be W (a useful special situation). MOVFF is particularly useful for transferring a data memory location to a peripheral register (such as the transmit buffer or an I/O port). The MOVFF instruction cannot use the PCL, TOSU, TOSH or TOSL as the destination register.
Note:	The MOVFF instruction should not be used to modify information while any interrupt is enabled. See Section 8.0 for more information.
Words:	2
Cycles:	2 (3)
Q Cycle Activity:	Q1 Q2 Q3 Q4 Decode Read register f' (src) Process Data No operation Decode No operation No operation Write register f' (dest) No dummy read
Example:	MOVFF REG1, REG2
Before Instruction	REG1 = 0x33 REG2 = 0x11
After Instruction	REG1 = 0x33 REG2 = 0x33

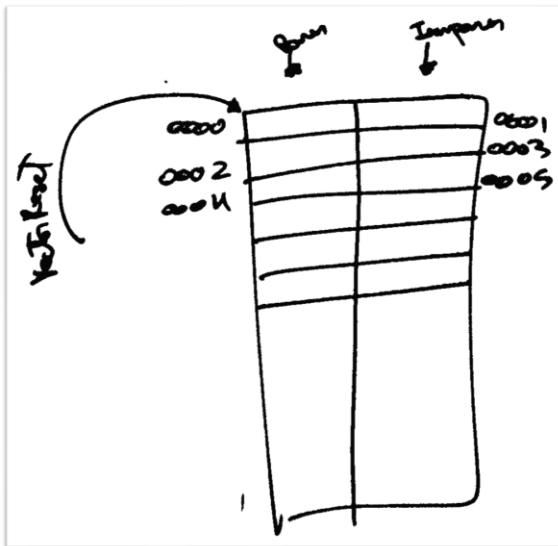


Mueve el contenido de REG1 hacia REG2

ojo: MOVFF demora dos ciclos en ejecutarse

Example:	MOVFF REG1, REG2
Before Instruction	REG1 = 0x33 REG2 = 0x11
After Instruction	REG1 = 0x33 REG2 = 0x33

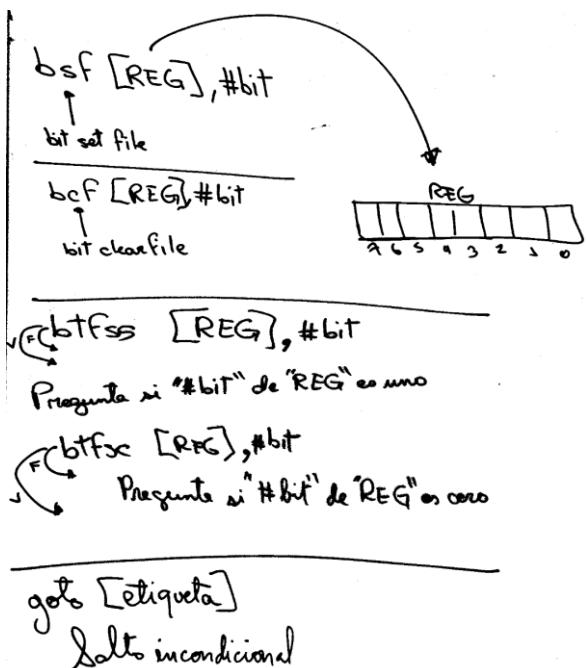
## Estructura de la memoria de programa



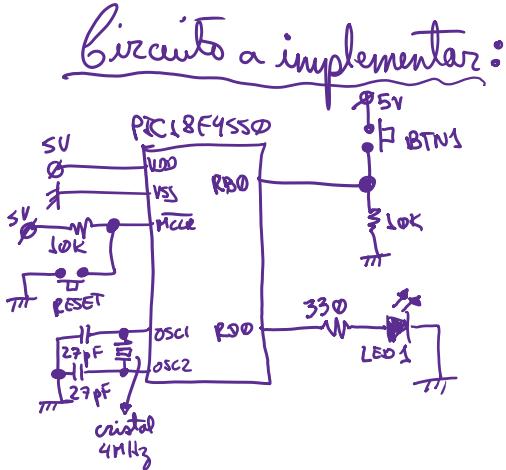
- Memoria de programa de 8 bits de ancho, 32K de capacidad.
- Las instrucciones ocupan 2 bytes (cada una)
- El contador de programa (PC) se incrementa de 2 en 2.
- Se puede usar la memoria de programa para almacenar datos (8 bits)

MPASM - Instrucciones

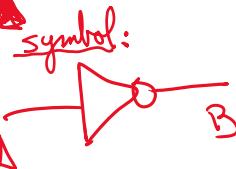
- Manipulación de bit en un registro (bsf, bcf)
- Pregunta de bit en un registro (btfs, btfsc)
- Salto incondicional (goto)



## Ejemplo: Negador lógico

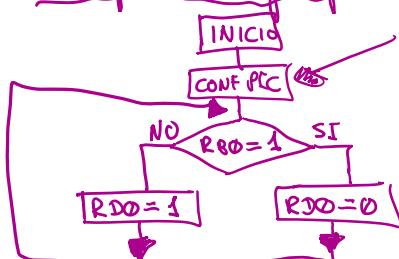


truth table:



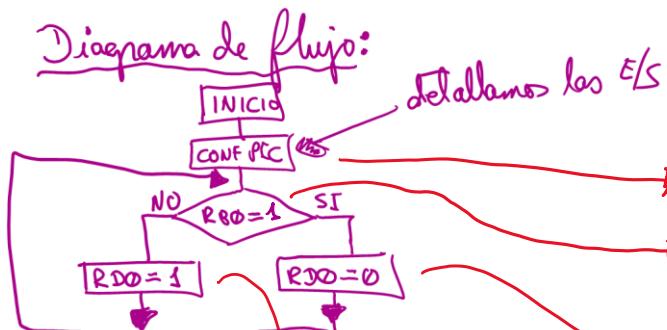
A	B
0	1
1	0

Diagrama de flujo:



detallamos los E/S

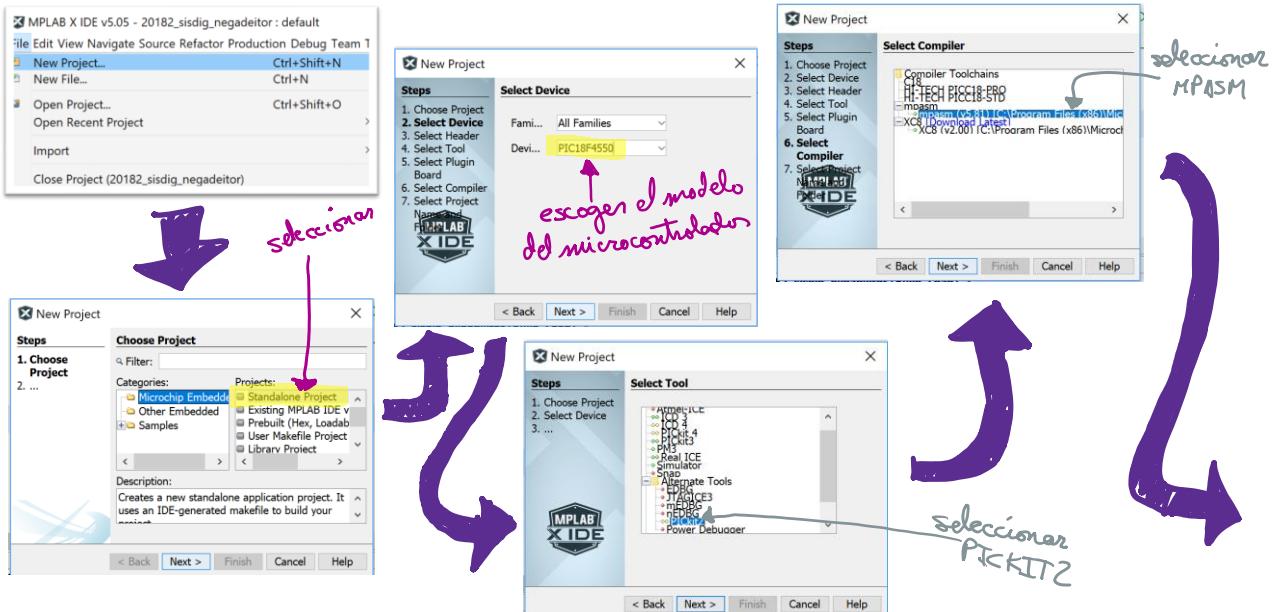
## Ejemplo: Negador lógico



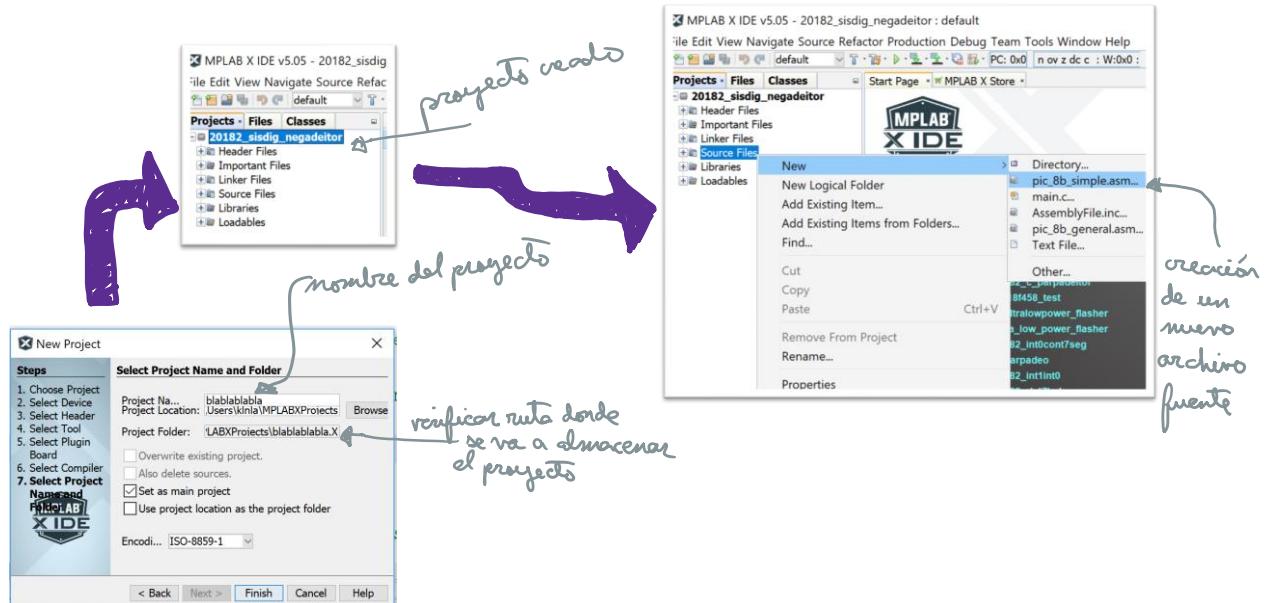
Código en MPASM:

- configuro: bcf TRISD, 0
- inicio: btfss PORTB, 0  
goto falso
- verdadero: bcf LATD, 0  
goto inicio
- Falso: bsf LATD, 0  
goto inicio
- ;
- ;

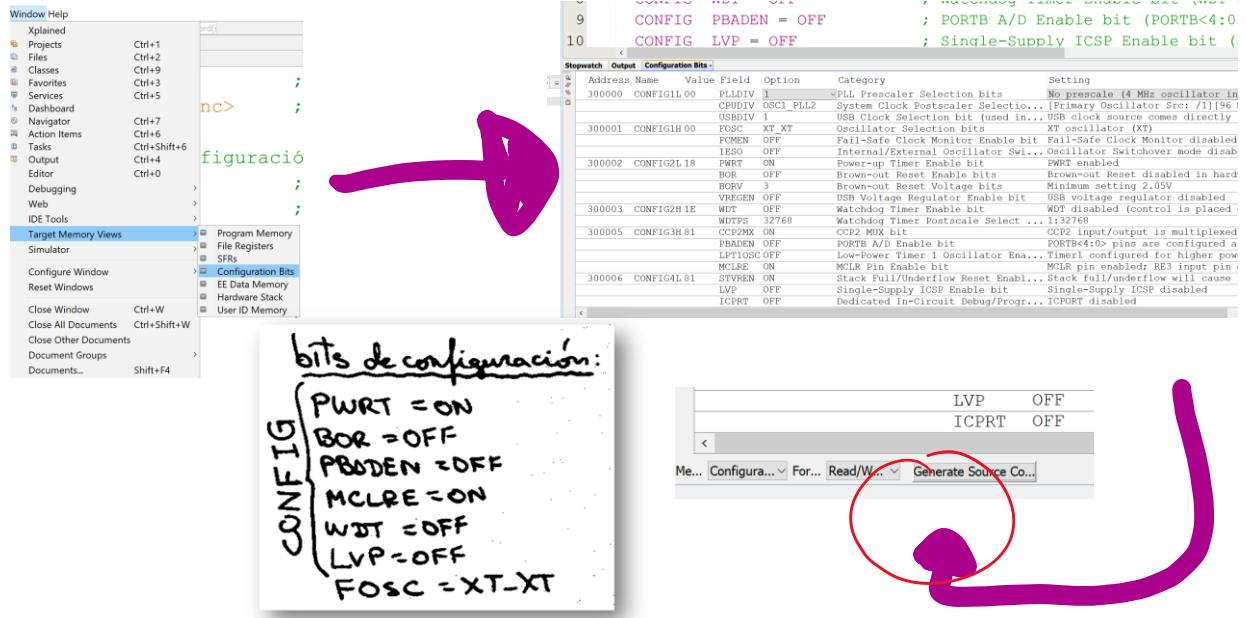
# Creación del proyecto en el MPLAB X



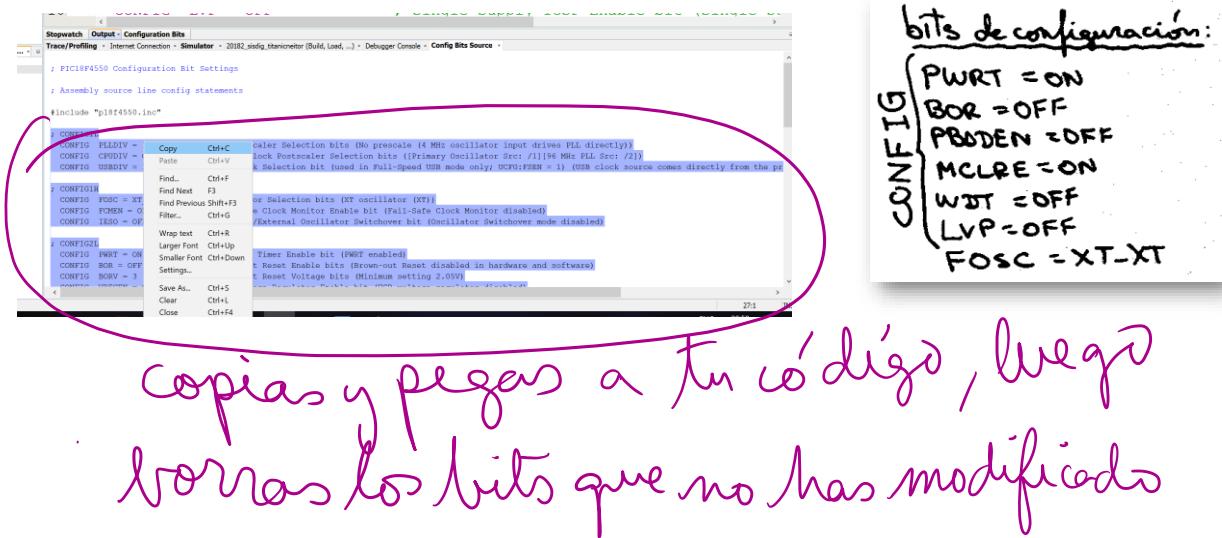
# Creación del proyecto en el MPLAB X



# Obtención de los bits de configuración



# Obtención de los bits de configuración

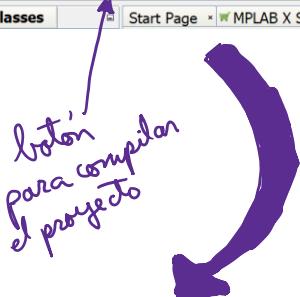
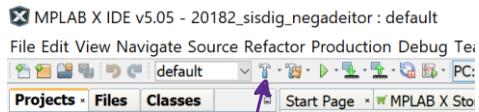


## Plantilla y código del programa

```

1      list p=18f4550          ;Modelo del microcontrolador
2      #include <p18f4550.inc>    ;librería de nombres
3
4      ;Zona de los bits de configuración del microcontrolador
5      CONFIG FOSC = XT_XT       ; Oscillator Selection bits (XT oscillator (XT))
6      CONFIG PWRT = ON          ; Power-up Timer Enable bit (PWRT enabled)
7      CONFIG BOR = OFF           ; Brown-out Reset Enable bits (Brown-out Reset disabled in hardware and software)
8      CONFIG WDT = OFF           ; Watchdog Timer Enable bit (WDT disabled (control is placed on the SWDTEN bit))
9      CONFIG PBADEN = OFF        ; PORTB A/D Enable bit (PORTB<4:0> pins are configured as digital I/O on Reset)
10     CONFIG LVP = OFF           ; Single-Supply ICSP Enable bit (Single-Supply ICSP disabled)
11
12     org 0x0000                ;Vector de reset
13     goto configura
14
15     org 0x0020                ;Zona de programa de usuario
16 configura:
17     bcf TRISD, 0              ;Puerto D0 como salida
18     bsf TRISB, 0              ;Puerto B0 como entrada
19 inicio:
20     btfss PORTB, 0            ;Verificar si el puerto B0 es 0
21     goto falson
22 verdaderon:
23     bcf LATD, 0
24     goto inicio
25 falson:
26     bsf LATD, 0
27     goto inicio
28
29 end

```



Ventana de salida (output)

```

MLINK 5.09, LINKER
wvicio Database Version 1.44
copyright (c) 1998-2011 Microchip Technology Inc.
errors : 0

MPASM 5.09, COFF to HEX File Converter
copyright (c) 1998-2011 Microchip Technology Inc.
errors : 0

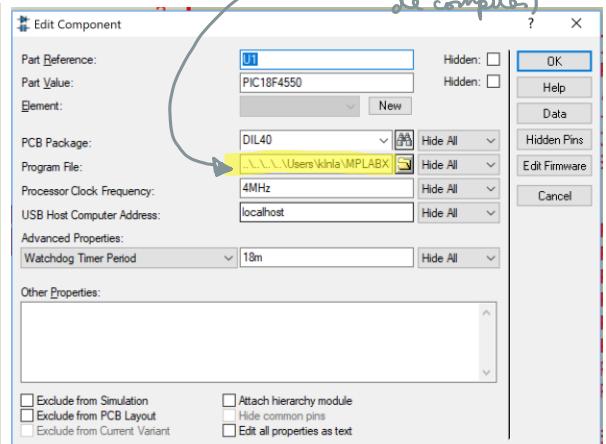
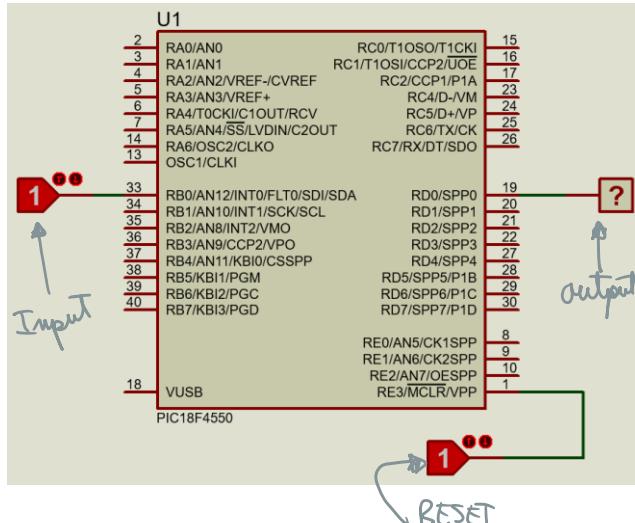
make[2]: Leaving directory 'C:/Users/kinla/MPLABXProjects/20182_sisdig_negadeitor.X'
make[1]: Leaving directory 'C:/Users/kinla/MPLABXProjects/20182_sisdig_negadeitor.X'
GUILD SUCCESSFUL (total time: 1s)
Loading code from C:/Users/kinla/MPLABXProjects/20182_sisdig_negadeitor.X/dist/default/production/20182_sisdig_negadeitor.X.production.
Loading completed

```

MPASM → \*.asm  
bin → \*.hex  
lo que se usa  
para grabar al PIC18F4550

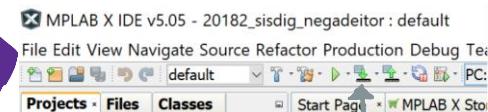
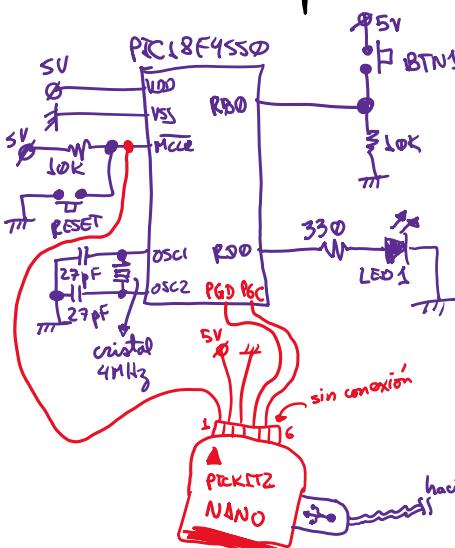
si "build successful" ⇒  
se creará el archivo  
\*.hex

## Simulación en Proteus VSM

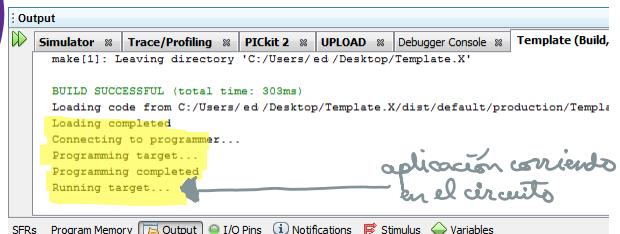


Ventana de propiedades  
del microcontrolador dentro  
del diseño en Proteus (doble clic)

Conexión del PICKIT2 con el circuito de la aplicación:

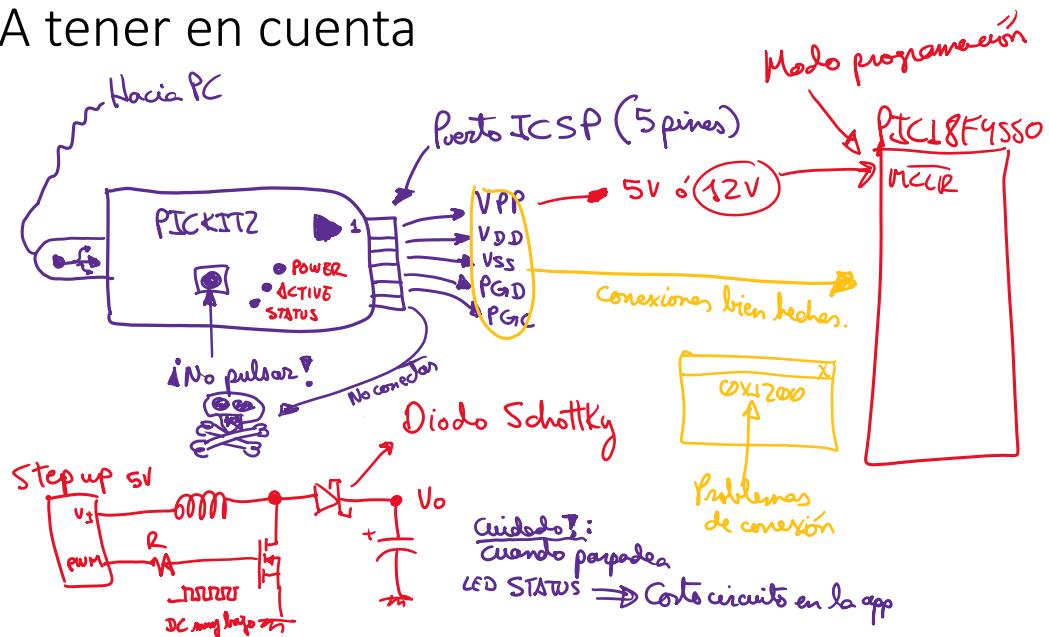


boton para grabar el  
t. hex en el microcon-  
trolero usando el PICKIT2

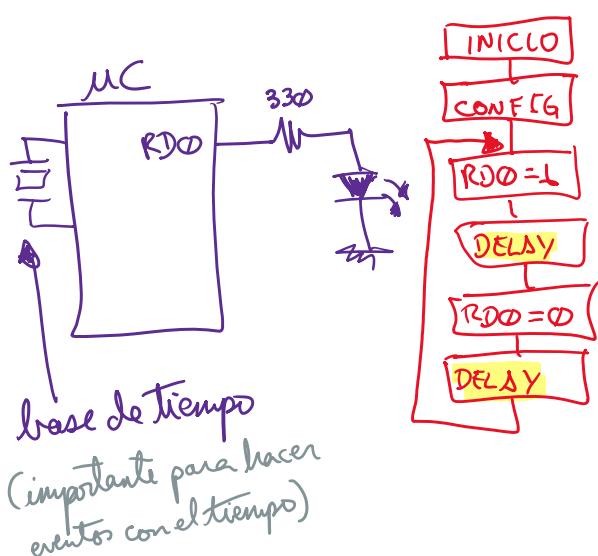


hacia la PC (revisar que el PICKIT2  
se encuentre reconocido por el S.O. Windows)

## A tener en cuenta



## Titanicneitor: Titilar un LED



Jávea de instrucción?

→ Recanto se demora en ejecutar una instrucción

$$F_{osc}/4$$

$$4MHz/4 = 1MHz$$

$$T = \frac{1}{f} = 1\mu s$$

⇒ "Mop" demora 1μs cuando cristal = 4MHz

## Titanicneitor: Titilar un LED

Requerimos generar un DELAY ...

~~mop~~ { Requerimos  
500000 mops  
para un retraso  
de 500ms ...

No alcanza en la  
memoria de programa!



Restricción: el uc  
PIC 18F4550 no puedes  
colocar \$000000, solo  
tiene registros de 8 bits  
(max 255 da)

En alto nivel:

```

For (int i=0; i<250; i++){
    For (int j=0; j<250; j++){
        {mop} → 625000 veces.
    }
}
```

Instrucciones de decremento e incremento de un registro

decremento [reg] y pregunta si llegó a cero (Z=1)

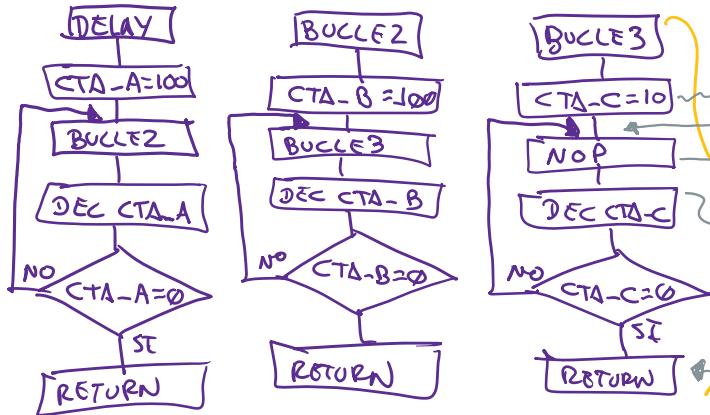
✓ decfsz [reg], d  
    incfsz

instrucción complemento a decfsz  
(incremento y pregunta si llegó  
a cero)

✓ decf [reg], f  
    btss STATUS, Z  
    incf ← instrucción complemento a decf  
(decremento)

# Titanicneitor: Titilar un LED

⇒ Necesitarán 3 anillos de repetición para alcanzar DELAY ~ 500ms.



Ejemplo de codificación en base al diagrama de flujo

```

bucle3:
    movlw .10
    movwf cta_c
otro3:
    nop
    decfsz cta_c, f
    goto otro3
    return

```

## Codificación en Assembler del Titanicneitor

```

1  list p=18f4550          ;Modelo del microcontrolador
2  #include <p18f4550.inc> ;librería de nombres
3
4 ;Zona de los bits de configuración del microcontrolador
5  CONFIG FOSC = XT_XT      ; Oscillator Selection bits (XT oscillator (XT))
6  CONFIG PWRT = ON         ; Power-up Timer Enable bit (PWRT enabled)
7  CONFIG BOR = OFF         ; Brown-out Reset Enable bits (Brown-out Reset disabled (BOR))
8  CONFIG WDT = OFF         ; Watchdog Timer Enable bit (WDT disabled (controlled by CS1 bit))
9  CONFIG PBADEN = OFF      ; PORTB A/D Enable bit (PORTB<4:0> pins are configured as digital inputs)
10 CONFIG LVPEE = OFF        ; Single-Supply ICSP Enable bit (Single-Supply ICSP disabled)
11
12 cblock 0x0020             ;Zona de declaración de etiquetas a los
13     cta_a                  ;registros GPR (variables)
14     cta_b
15     cta_c
16 endc
17
18 org 0x0000                ;Vector de reset
19 goto configurar
20
21 org 0x0020                ;Zona de programa de usuario
22 configurar:
23     bcf TRISD, 0           ;Para hacer que el puerto D0 sea salida
24 inicio:
25     bsf LATD, 0
26     call delaymon
27     bcf LATD, 0
28     call delaymon
29     goto inicio
} rutina principal

```

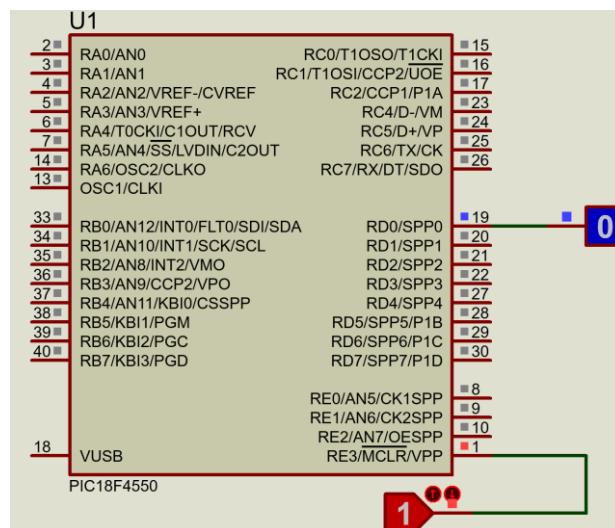
```

30
31 delaymon:
32     movlw .100
33     movwf cta_a
34 otro1:
35     call bucle2
36     decfsz cta_a, f
37     goto otro1
38     return
39
40 bucle2:
41     movlw .100
42     movwf cta_b
43 otro2:
44     call bucle3
45     decfsz cta_b, f
46     goto otro2
47     return
48
49 bucle3:
50     movlw .10
51     movwf cta_c
52 otro3:
53     nop
54     decfsz cta_c, f
55     goto otro3
56     return
57
58 end

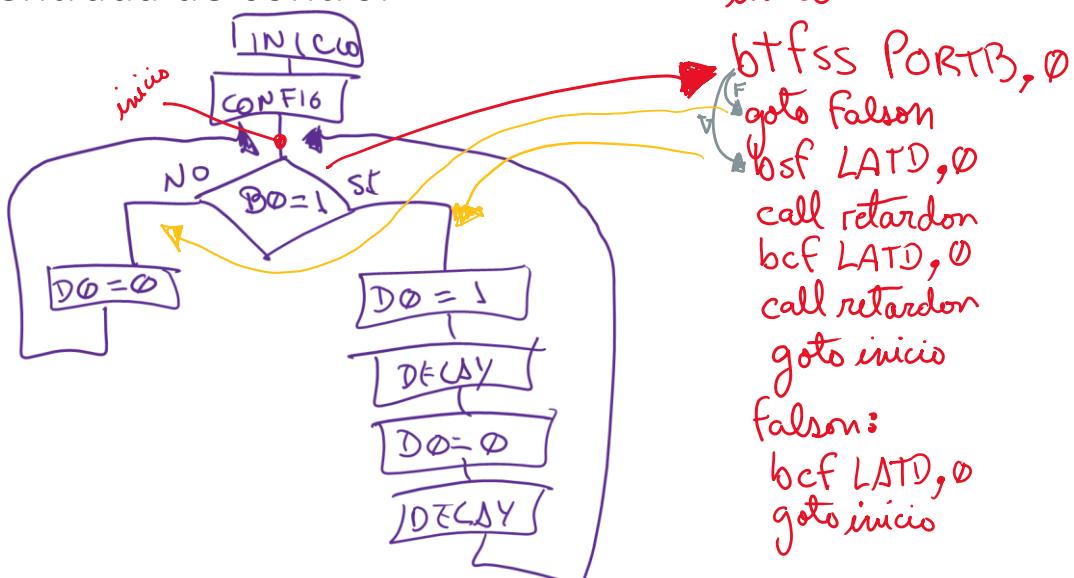
```

Subrutina de retraso

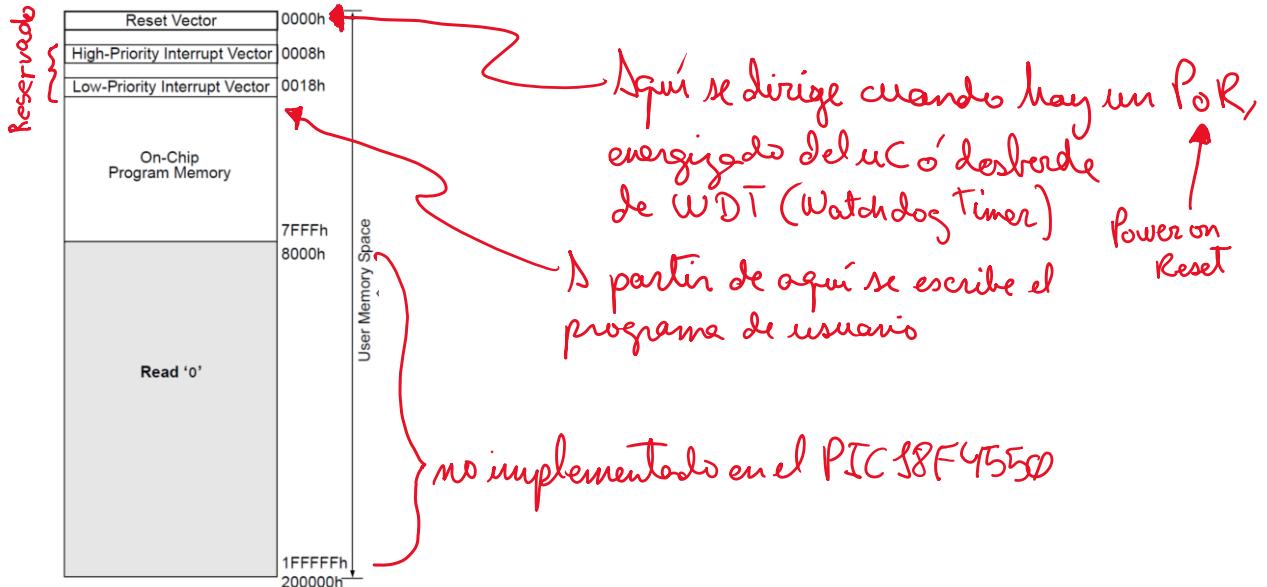
# Simulación en Proteus



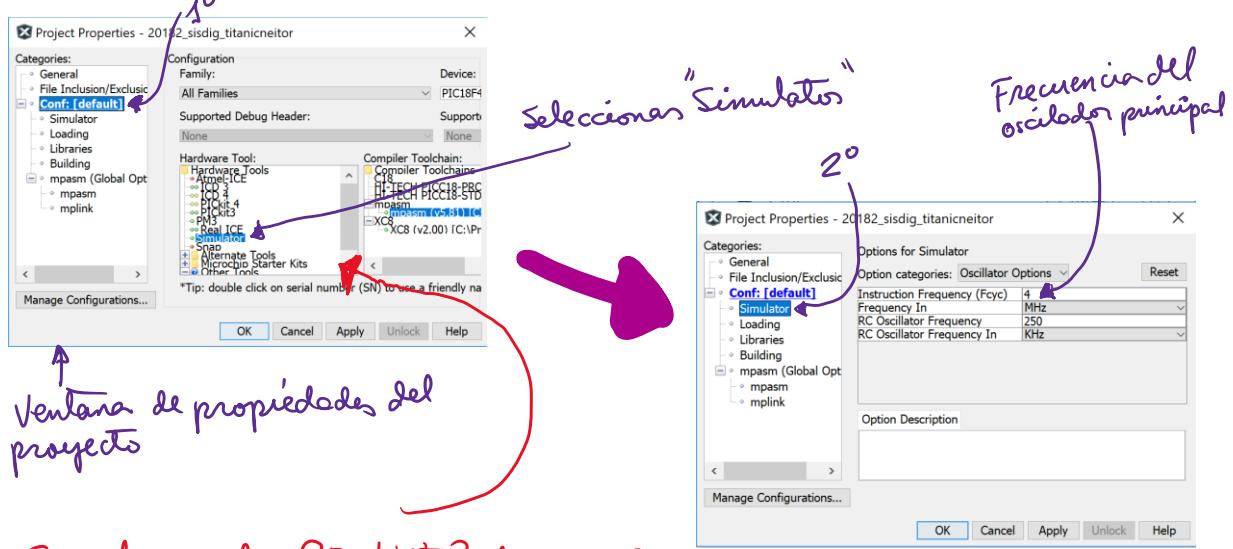
Modificando rutina inicial del titilador: Añadiendo entrada de control



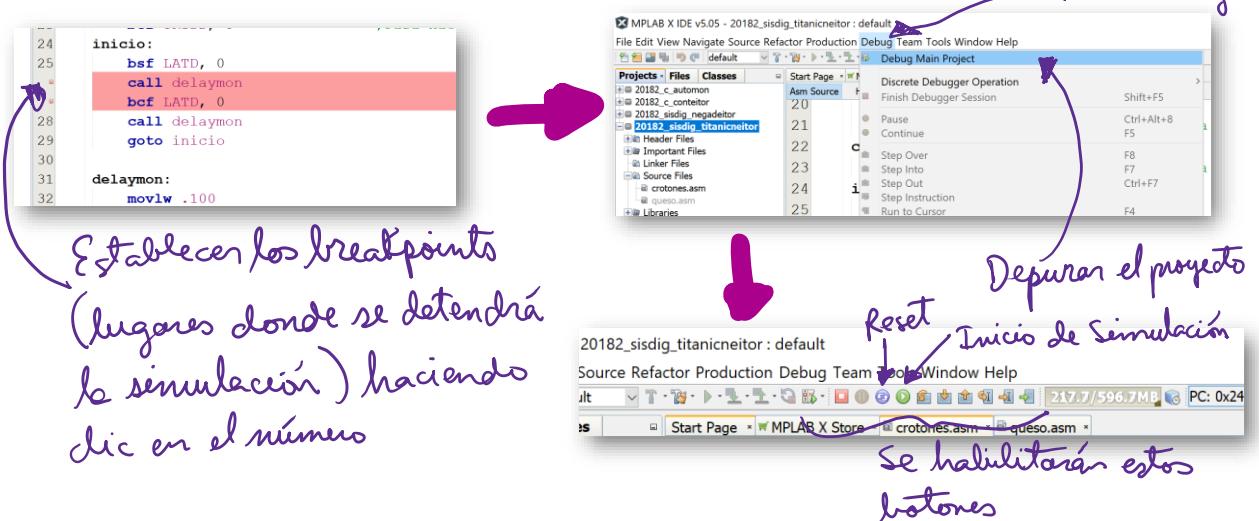
## Recordando la memoria de programa:



Usando el “stopwatch” del modo simulador del MPLAB X para obtener la duración de la subrutina de retardo



## Usando el “stopwatch” del modo simulador del MPLAB X para obtener la duración de la subrutina de retardo



## Usando el “stopwatch” del modo simulador del MPLAB X para obtener la duración de la subrutina de retardo

