Microcontroladores

Semestre: 2021-2

Profesor: Kalun José Lau Gan

Semana 9: Lenguajes de alto nivel en microcontroladores

¿Preguntas previas?

- Sobre el TF
  - Se estará publicando mas tarde la “guía del trabajo final” en el AV
- ¿Cómo se trabajarán en los laboratorios siguientes?
  - De manera individual
- ¿Se requerirán mas materiales para los laboratorios?
  - Solo los de la lista publicada a inicios del curso
- El hecho de usar C lo hace mas fácil?
  - El proceso de diseño es el mismo tanto para lenguaje Assembler como en lenguaje C, se tiene que implementar primero el hardware, desarrollar el algoritmo y finalmente aterrizar el algoritmo en un lenguaje de programación.
  - El lenguaje XC se va a permitir el uso de librerías y funciones especializadas, además de manejar mejor las operaciones matemáticas y manipulación de datos

Agenda:

- Resolución del EA
- Lineamientos de TF
- Los lenguajes de alto nivel, en nuestro caso el XC8
- El MPLAB Xpress
- Desarrollo de una plantilla en el MPLAB X para el XC8
- Ejemplos iniciales en XC8

Panorama de los lenguajes de alto nivel para microcontroladores

Cada dispositivo microcontrolador tendrá sus propias plataformas de lenguaje de programación.

Va a depender del desarrollador del lenguaje de alto nivel para determinado microcontrolador (No todos los desarrolladores soportan todos los dispositivos de un fabricante)

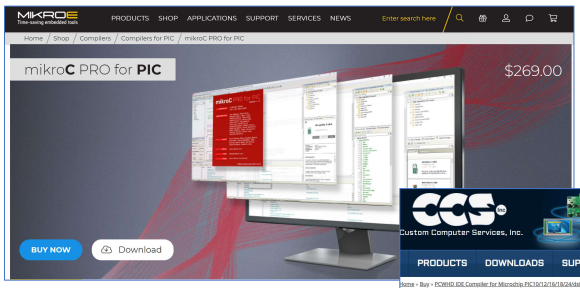
Hay varias compañías que desarrollan lenguajes de alto nivel para un microcontrolador.

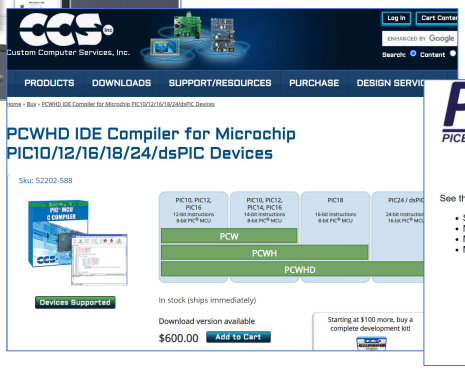
Problema de compatibilidad entre distintas plataformas de desarrollo aún empleando un mismo lenguaje.

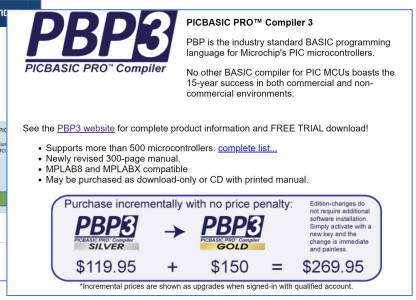
- Basic
- C
- Java (muy poco utilizado y soportado)
- Python

Panorama de los lenguajes de alto nivel para microcontroladores

- Los compiladores de lenguajes de alto nivel (de terceros) mas empleados en la actualidad.
- Estas empresas ofrecen versiones DEMO para que el usuario pueda probar las bondades de cada uno.
- Cada empresa ofrece un pack de librerías especializadas para crear aplicaciones con periféricos especializados en el menor tiempo.



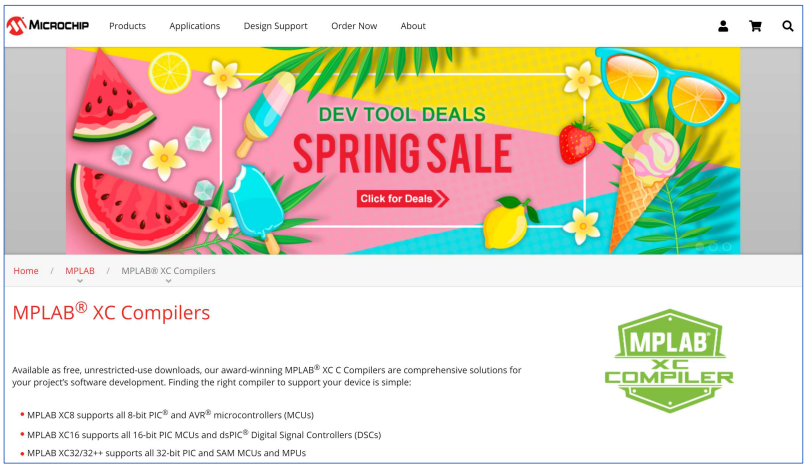




5

El compilador XC de Microchip

- Disponible versión gratis sin límite.
- La versión PRO no es gratis. Provee optimización de código compilado para que ocupe menos espacio.
- Tres versiones:
  - XC8 – PIC10, PIC12, PIC16, PIC18
  - XC16 – PIC24, dsPIC
  - XC32 – PIC32



6

El compilador XC de Microchip

- Los XC8, XC16 y XC32 vienen separados…
- Ej. El bootloader HID para el microcontrolador PIC18F4550 requiere que se compile el código en XC8 PRO para que entre en su memoria.

Part Number: SW006021-DGL - MPLAB XC8 Compiler PRO Dongle License

The MPLAB XC8 is a full-featured, highly-optimized ANSI C compiler for all 8-bit AVR® and PIC® MCUs. This compiler integrates into Microchip's MPLAB(R) X IDE, is compatible with all Microchip debuggers and emulators, and runs on Windows, Linux and Mac OS X.

The dongle license is a USB flash drive that contains a single-user encrypted license. It is a perpetual license and unlocks PRO optimizations for all versions of the MPLAB XC8 compilers, **version 1.41 and later**, and does not include High Priority Access (HPA). The Dongle License allows a user to unlock PRO optimizations on any computer it is plugged into. If lost, the Dongle License can be replaced one time for a processing fee of $200 and the dongle must be registered to the user.

More Info »

Standard Pricing:

Order Quantity	USD per Unit
1+	$1,695.00

In Stock: **9**
Delivery and scheduling options available in the cart

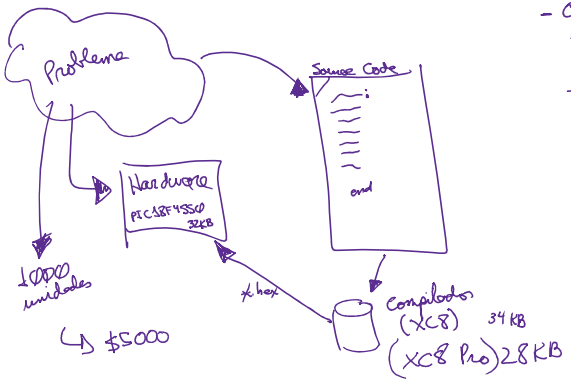
Order now, up to **9** can ship on **20-May-2020**

Lead Time For Additional Quantities
Additional quantities can ship by **11-Aug-2020**

Quantity:

7

Caso:



- Opciones:
  - Quitarle funcionalidad a la aplicación X
  - Cambiar el uC.
    +USD5
  - Comprar licencia XC8 Pro

8

Optimización en el compilador XC

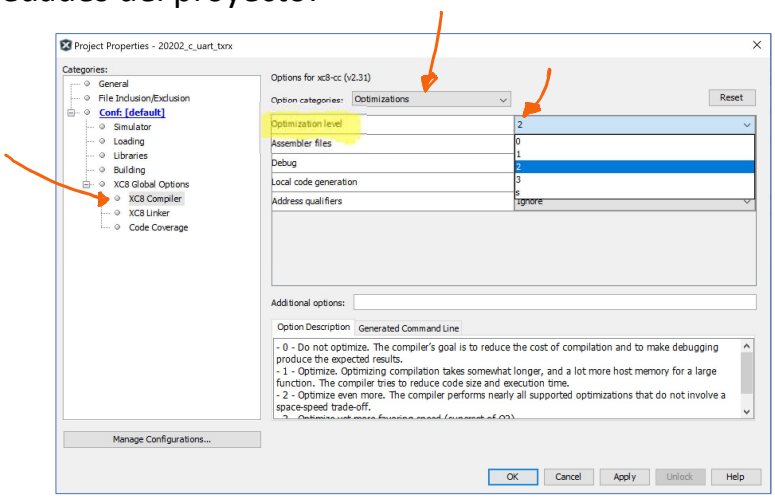
- **Optimization levels:**
  - Level 0: Fastest compilation time, minimal optimizations
  - Level 1: Optimizations with low debugging impact
  - Level 2: All optimizations that give the best balance between speed and size
  - Level 3: Program execution will be as fast as possible
  - Level s: Code size will be as small as possible
- **Where available use:**
  - Use Whole-program and Link-time setting
  - Procedural abstraction

- En la versión gratuita del XC tenemos optimización hasta el nivel 2
- Para acceder al nivel 3 o nivel s requerimos tener la licencia PRO

9

Optimización en el compilador XC

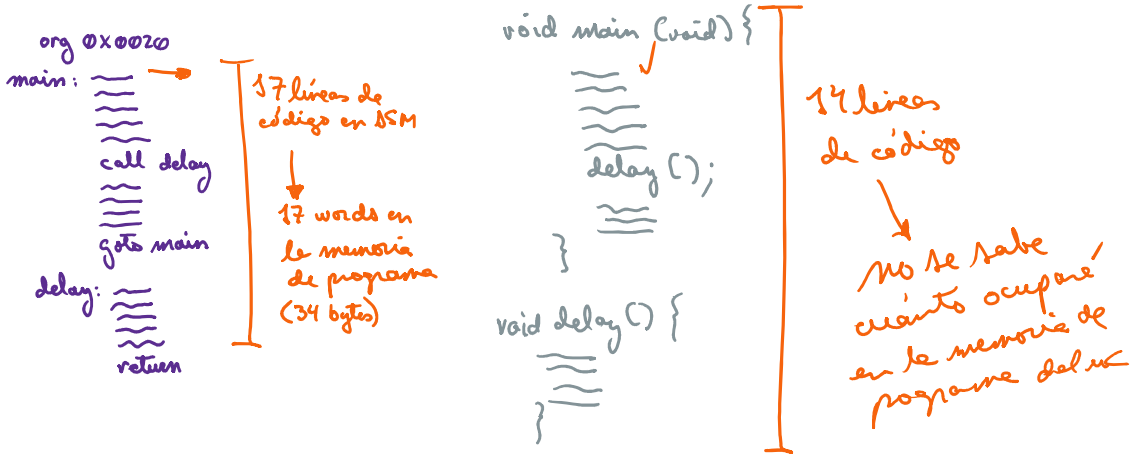
- Para acceder a las opciones de optimización ingresamos a las propiedades del proyecto:



10

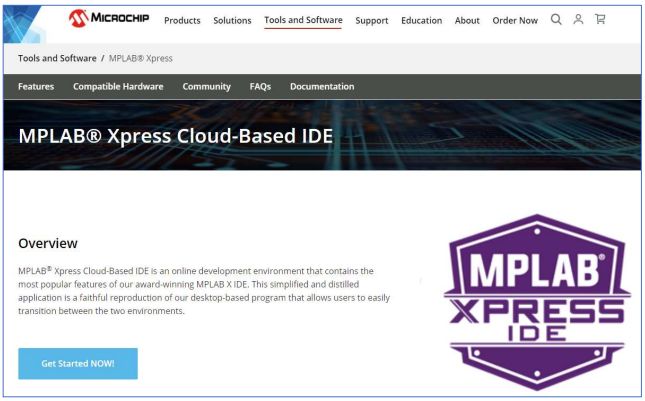
Eficiencia de código en Assembler vs C

¿Por qué Assembler y por qué C?



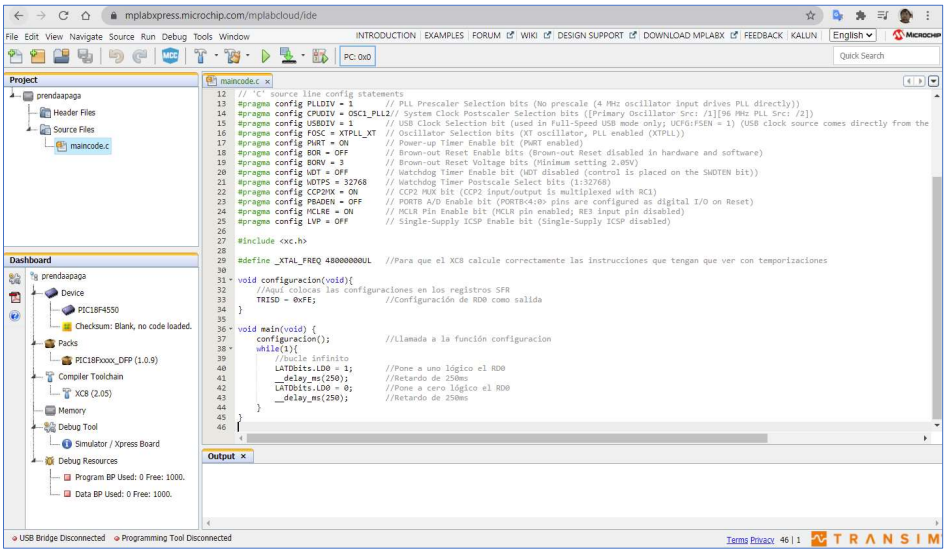
El MPLAB Xpress

- Link: https://www.microchip.com/en-us/development-tools-tools-and-software/mplab-xpress
- Entorno de trabajo online
- Permite realizar proyectos en XC sin tener que instalar el MPLAB X en el computador.



El MPLAB Xpress

- Los proyectos se almacenan en la nube de Microchip
- Soporte solo para algunos modelos de microcontrolador PIC



13

MPLAB X: Creación de un proyecto en XC8

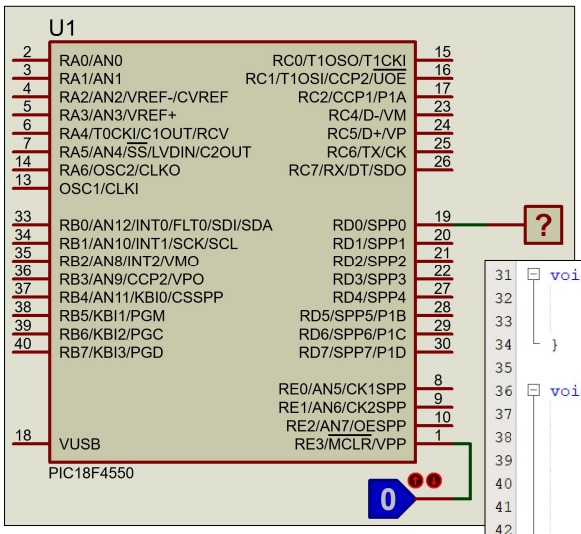
- Link del manual de XC8:
  - http://ww1.microchip.com/downloads/en/devicedoc/50002053g.pdf
- Link de descarga del XC8 v2.32:
  - https://www.microchip.com/mplabxc8windows

14

Plantilla de código en XC8:

```
10  #pragma config PLLDIV = 1        // PLL Prescaler Selection bit
11  #pragma config CPUDIV = OSC1_PLL2// System Clock Postscaler Se
12  #pragma config FOSC = XTPLL_XT   // Oscillator Selection bits (
13  #pragma config PWRT = ON         // Power-up Timer Enable bit (
14  #pragma config BOR = OFF         // Brown-out Reset Enable bits
15  #pragma config BORV = 3          // Brown-out Reset Voltage bit
16  #pragma config WDT = OFF         // Watchdog Timer Enable bit (
17  #pragma config WDTPS = 32768     // Watchdog Timer Postscale Se
18  #pragma config CCP2MX = ON       // CCP2 MUX bit (CCP2 input/ou
19  #pragma config PBADEN = OFF      // PORTB A/D Enable bit (PORTB
20  #pragma config MCLRE = ON        // MCLR Pin Enable bit (MCLR p
21  #pragma config LVP = OFF         // Single-Supply ICSP Enable b
22
23  #include <xc.h>
24
25  #define _XTAL_FREQ 48000000UL   //Frecuencia de trabajo 48MHz
26
27  void configuracion(void) {
28      //Aqui colocas las configuraciones iniciales
29  }
30
31  void main(void) {
32      configuracion();
33      while (1) {
34          //Tu programa de usuario
35      }
36  }
```

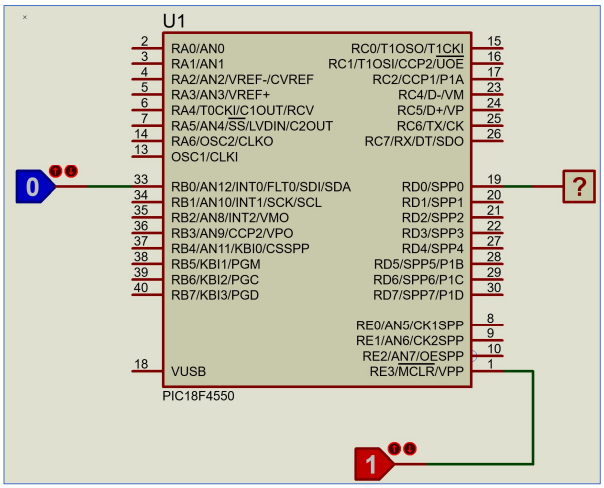

Ejemplo en XC8: Titileo de LED




```
31  void configuracion(void){
32      //Aquí colocas las configuraciones en los registros SFR
33      TRISD = 0xFE;               //Configuración de RD0 como salida
34  }
35
36  void main(void) {
37      configuracion();            //Llamada a la función configuracion
38      while(1){
39          //bucle infinito
40          LATDbits.LD0 = 1;       //Pone a uno lógico el RD0
41          __delay_ms(250);        //Retardo de 250ms
42          LATDbits.LD0 = 0;       //Pone a cero lógico el RD0
43          __delay_ms(250);        //Retardo de 250ms
44      }
45  }
```

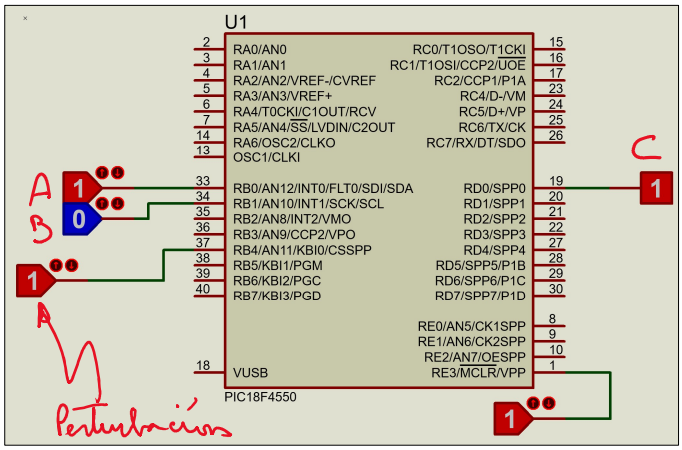
Ejemplo en XC8: Negador lógico de un bit

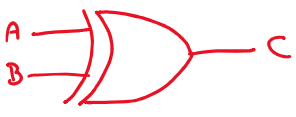
```
#pragma config PLLDIV = 1          // PLL Prescaler Selection bits (No pre
#pragma config CPUDIV = OSC1_PLL2// System Clock Postscaler Selection b
#pragma config FOSC = XTPLL_XT  // Oscillator Selection bits (XT oscill
#pragma config PWRT = ON           // Power-up Timer Enable bit (PWRT enab
#pragma config BOR = OFF           // Brown-out Reset Enable bits (Brown-o
#pragma config BORV = 3            // Brown-out Reset Voltage bits (Minimu
#pragma config WDT = OFF           // Watchdog Timer Enable bit (WDT disab
#pragma config WDTPS = 32768       // Watchdog Timer Postscale Select bits
#pragma config CCP2MX = ON         // CCP2 MUX bit (CCP2 input/output is m
#pragma config PBADEN = OFF        // PORTB A/D Enable bit (PORTB<4:0> pin
#pragma config MCLRE = ON          // MCLR Pin Enable bit (MCLR pin enable
#pragma config LVP = OFF           // Single-Supply ICSP Enable bit (Singl

#include <xc.h>
#define _XTAL_FREQ 48000000UL

void init_conf(void){
    //Aca colocaremos las configuraciones iniciales de la aplicacion
    //TRISDbits.RD0 = 0;            // RD0 como salida
    asm("bcf TRISD, 0");            // Escribiendo instrucciones en mpasm
}

void main(void) {
    init_conf();
    while(1){
        if (PORTBbits.RB0 == 1) {
            LATDbits.LD0 = 0;
        }
        else{
            LATDbits.LD0 = 1;
        }
    }
}
```



17

Ejemplo en XC8: Compuerta XOR






```
void configuracion(void) {
    TRISDbits.RD0 = 0;          //Puerto RD0 como salida
}

void main(void) {
    configuracion();
    while(1) {
        //Bucle infinito
        //Con switch-case
        switch (PORTB & 0x03) {
            case 0x00:
                LATDbits.LD0 = 0;
                break;
            case 0x01:
                LATDbits.LD0 = 1;
                break;
            case 0x02:
                LATDbits.LD0 = 1;
                break;
            case 0x03:
                LATDbits.LD0 = 0;
                break;
        }
    }
}
```

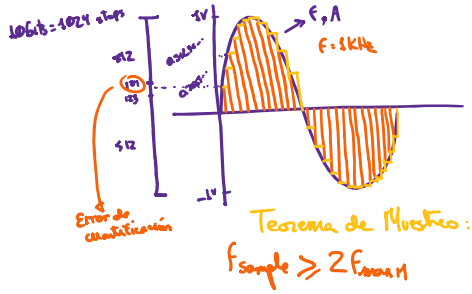
18

Repaso de conocimientos:

¿Tipos de señales? → Continuas / Discretas

¿Por qué digitalizamos señales?

preservar (almacenar) / procesar / transferir } Mejor que en analógica



Teorema de Muestreo:

$f_{sample} \geq 2 f_{max\,M}$

Audio: 20Hz ~ 20kHz → Calidad CD en audio digital: f_s = 44100Hz, 16 bits

Conversor A/D

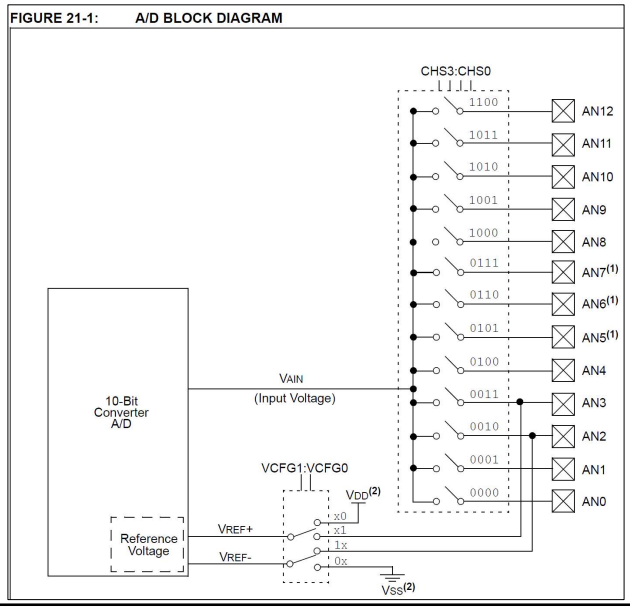
Revisar Capítulo 21 de la hoja técnica del PIC18F4550

- Resolución:
- Cantidad de canales analógicos:
- Tiempo de adquisición:
- Rango de voltaje de entrada:
- ¿Cuáles son los valores límites de Vref+ y Vref-?
- Proceso de adquisición de una señal analógica
- ¿Interviene el teorema de muestreo?
- ¿Hay interrupciones?

Conversor A/D

- Resolución: 10bits (ADRESH:ADRESL)
  - Posee un bit ADFM (justificación del resultado)
- Cantidad de canales analógicos: 13
- Se lee un canal analógico a la vez
- ¿Interviene el teorema de muestreo? Si.
- Tiempo de adquisición: se configura en reg. ADCON2
- Rango de voltaje de entrada: 0-5V? (Vref+ = VDD, Vref- = VSS)
- Proceso de adquisición de una señal analógica (ver datasheet)
- ¿Hay interrupciones? Si. ADIE, ADIF

Conversor A/D: Diagrama de bloques



Registros de configuración para el A/D:

REGISTER 21-1: ADCON0: A/D CONTROL REGISTER 0

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	CHS3	CHS2	CHS1	CHS0	GO/$\overline{\text{DONE}}$	ADON
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 5-2 **CHS3:CHS0:** Analog Channel Select bits
0000 = Channel 0 (AN0)
0001 = Channel 1 (AN1)
0010 = Channel 2 (AN2)
0011 = Channel 3 (AN3)
0100 = Channel 4 (AN4)
0101 = Channel 5 (AN5)(1,2)
0110 = Channel 6 (AN6)(1,2)
0111 = Channel 7 (AN7)(1,2)
1000 = Channel 8 (AN8)
1001 = Channel 9 (AN9)
1010 = Channel 10 (AN10)
1011 = Channel 11 (AN11)
1100 = Channel 12 (AN12)
1101 = Unimplemented(2)
1110 = Unimplemented(2)
1111 = Unimplemented(2)

bit 1 **GO/$\overline{\text{DONE}}$:** A/D Conversion Status bit
When ADON = 1:
1 = A/D conversion in progress
0 = A/D Idle

bit 0 **ADON:** A/D On bit
1 = A/D converter module is enabled
0 = A/D converter module is disabled

REGISTER 21-2: ADCON1: A/D CONTROL REGISTER 1

U-0	U-0	R/W-0	R/W-0	R/W-0(1)	R/W(1)	R/W(1)	R/W(1)
—	—	VCFG1	VCFG0	PCFG3	PCFG2	PCFG1	PCFG0
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 **VCFG1:** Voltage Reference Configuration bit (V_{REF-} source)
1 = V_{REF-} (AN2)
0 = V_{SS}

bit 4 **VCFG0:** Voltage Reference Configuration bit (V_{REF+} source)
1 = V_{REF+} (AN3)
0 = V_{DD}

bit 3-0 **PCFG3:PCFG0:** A/D Port Configuration Control bits:

PCFG3:PCFG0	AN12	AN11	AN10	AN9	AN8	AN7(2)	AN6(2)	AN5(2)	AN4	AN3	AN2	AN1	AN0
0000(1)	A	A	A	A	A	A	A	A	A	A	A	A	A
0001	A	A	A	A	A	A	A	A	A	A	A	A	A
0010	A	A	A	A	A	A	A	A	A	A	A	A	A
0011	D	A	A	A	A	A	A	A	A	A	A	A	A
0100	D	D	A	A	A	A	A	A	A	A	A	A	A
0101	D	D	D	A	A	A	A	A	A	A	A	A	A
0110	D	D	D	D	A	A	A	A	A	A	A	A	A
0111(1)	D	D	D	D	D	A	A	A	A	A	A	A	A
1000	D	D	D	D	D	D	A	A	A	A	A	A	A
1001	D	D	D	D	D	D	D	A	A	A	A	A	A
1010	D	D	D	D	D	D	D	D	A	A	A	A	A
1011	D	D	D	D	D	D	D	D	D	A	A	A	A
1100	D	D	D	D	D	D	D	D	D	D	A	A	A
1101	D	D	D	D	D	D	D	D	D	D	D	A	A
1110	D	D	D	D	D	D	D	D	D	D	D	D	A
1111	D	D	D	D	D	D	D	D	D	D	D	D	D

A = Analog input D = Digital I/O

23

Registros de configuración para el A/D:

REGISTER 21-3: ADCON2: A/D CONTROL REGISTER 2

R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ADFM	—	ACQT2	ACQT1	ACQT0	ADCS2	ADCS1	ADCS0
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 **ADFM:** A/D Result Format Select bit
1 = Right justified
0 = Left justified

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

bit 5-3 **ACQT2:ACQT0:** A/D Acquisition Time Select bits
111 = 20 T_{AD}
110 = 16 T_{AD}
101 = 12 T_{AD}
100 = 8 T_{AD}
011 = 6 T_{AD}
010 = 4 T_{AD}
001 = 2 T_{AD}
000 = 0 T_{AD}(1)

bit 2-0 **ADCS2:ADCS0:** A/D Conversion Clock Select bits
111 = F_{RC} (clock derived from A/D RC oscillator)(1)
110 = F_{OSC}/64
101 = F_{OSC}/16
100 = F_{OSC}/4
011 = F_{RC} (clock derived from A/D RC oscillator)(1)
010 = F_{OSC}/32
001 = F_{OSC}/8
000 = F_{OSC}/2

Procedimiento para adquirir una muestra de una señal analógica:

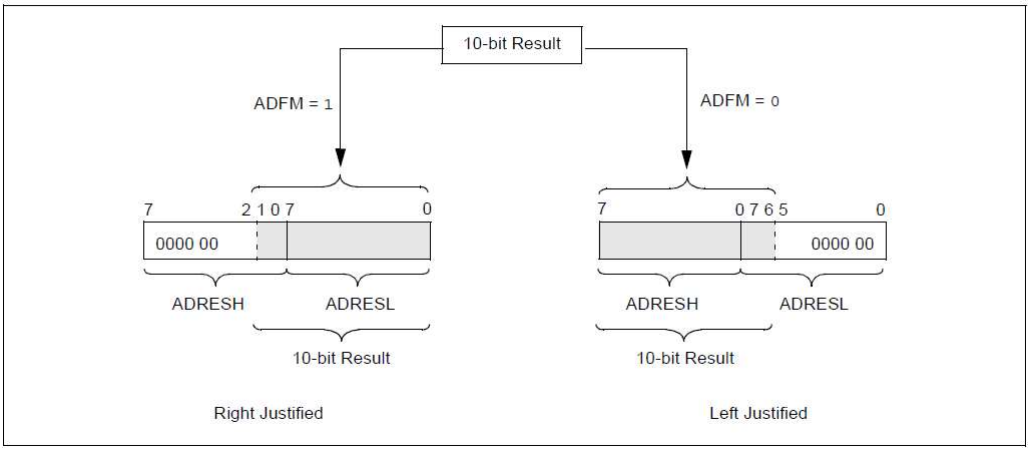
1. Configure the A/D module:
   - Configure analog pins, voltage reference and digital I/O (ADCON1)
   - Select A/D input channel (ADCON0)
   - Select A/D acquisition time (ADCON2)
   - Select A/D conversion clock (ADCON2)
   - Turn on A/D module (ADCON0)
2. Configure A/D interrupt (if desired):
   - Clear ADIF bit
   - Set ADIE bit
   - Set GIE bit
3. Wait the required acquisition time (if required).
4. Start conversion:
   - Set GO/$\overline{\text{DONE}}$ bit (ADCON0 register)
5. Wait for A/D conversion to complete, by either:
   - Polling for the GO/$\overline{\text{DONE}}$ bit to be cleared

   OR
   - Waiting for the A/D interrupt
6. Read A/D Result registers (ADRESH:ADRESL); clear bit ADIF, if required.
7. For next conversion, go to step 1 or step 2, as required. The A/D conversion time per bit is defined as T_{AD}. A minimum wait of 3 T_{AD} is required before the next acquisition starts.

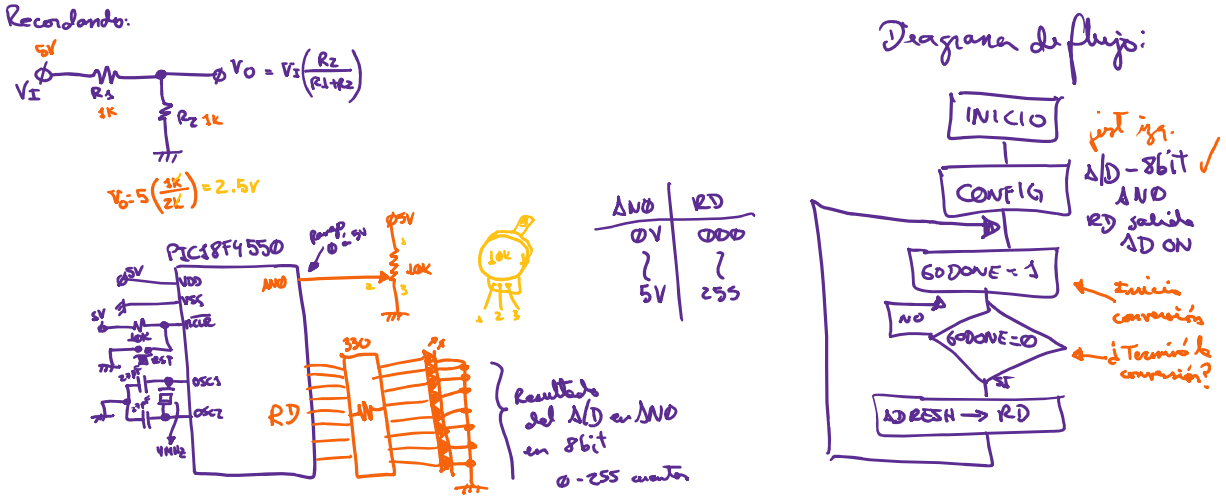
24

Para obtener el resultado de la conversión A/D:

FIGURE 11-4: A/D RESULT JUSTIFICATION



Ejemplo, leer en AN0 el valor de voltaje de un potenciómetro configurado como divisor de tensión y el resultado en 8 bits emitirlo por el puerto D



Configuración para el A/D para AN.0:

REGISTER 21-1: ADCON0: A/D CONTROL REGISTER 0

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	CHS3	CHS2	CHS1	CHS0	GO/DONE	ADON
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 5-2 **CHS3:CHS0:** Analog Channel Select bits
0000 = Channel 0 (AN0)
0001 = Channel 1 (AN1)
0010 = Channel 2 (AN2)
0011 = Channel 3 (AN3)
0100 = Channel 4 (AN4)
0101 = Channel 5 (AN5)(1,2)
0110 = Channel 6 (AN6)(1,2)
0111 = Channel 7 (AN7)(1,2)
1000 = Channel 8 (AN8)
1001 = Channel 9 (AN9)
1010 = Channel 10 (AN10)
1011 = Channel 11 (AN11)
1100 = Channel 12 (AN12)
1101 = Unimplemented(2)
1110 = Unimplemented(2)
1111 = Unimplemented(2)

bit 1 **GO/DONE:** A/D Conversion Status bit
When ADON = 1:
1 = A/D conversion in progress
0 = A/D Idle

bit 0 **ADON:** A/D On bit
1 = A/D converter module is enabled
0 = A/D converter module is disabled

ADCON0 = 0X01

REGISTER 21-2: ADCON1: A/D CONTROL REGISTER 1

U-0	U-0	R/W-0	R/W-0	R/W-0(1)	R/W(1)	R/W(1)	R/W(1)
—	—	VCFG1	VCFG0	PCFG3	PCFG2	PCFG1	PCFG0
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 **VCFG1:** Voltage Reference Configuration bit (VREF- source)
1 = VREF- (AN2)
0 = VSS

bit 4 **VCFG0:** Voltage Reference Configuration bit (VREF+ source)
1 = VREF+ (AN3)
0 = VDD

bit 3-0 **PCFG3:PCFG0:** A/D Port Configuration Control bits:

PCFG3: PCFG0	AN12	AN11	AN10	AN9	AN8	AN7(2)	AN6(2)	AN5(2)	AN4	AN3	AN2	AN1	AN0
0000(1)	A	A	A	A	A	A	A	A	A	A	A	A	A
0001	A	A	A	A	A	A	A	A	A	A	A	A	A
0010	A	A	A	A	A	A	A	A	A	A	A	A	A
0011	D	A	A	A	A	A	A	A	A	A	A	A	A
0100	D	D	A	A	A	A	A	A	A	A	A	A	A
0101	D	D	D	A	A	A	A	A	A	A	A	A	A
0110	D	D	D	D	A	A	A	A	A	A	A	A	A
0111(1)	D	D	D	D	D	A	A	A	A	A	A	A	A
1000	D	D	D	D	D	D	A	A	A	A	A	A	A
1001	D	D	D	D	D	D	D	A	A	A	A	A	A
1010	D	D	D	D	D	D	D	D	A	A	A	A	A
1011	D	D	D	D	D	D	D	D	D	A	A	A	A
1100	D	D	D	D	D	D	D	D	D	D	A	A	A
1101	D	D	D	D	D	D	D	D	D	D	D	A	A
1110	D	D	D	D	D	D	D	D	D	D	D	D	A
1111	D	D	D	D	D	D	D	D	D	D	D	D	D

A = Analog input D = Digital I/O

ADCON1 = 0X0E

27

Conf. del A/D para AN0: (tiempo de adq y just res)

REGISTER 21-3: ADCON2: A/D CONTROL REGISTER 2

R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ADFM	—	ACQT2	ACQT1	ACQT0	ADCS2	ADCS1	ADCS0
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 **ADFM:** A/D Result Format Select bit
1 = Right justified
0 = Left justified

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

bit 5-3 **ACQT2:ACQT0:** A/D Acquisition Time Select bits
111 = 20 TAD
110 = 16 TAD
101 = 12 TAD
100 = 8 TAD
011 = 6 TAD
010 = 4 TAD
001 = 2 TAD
000 = 0 TAD(1)

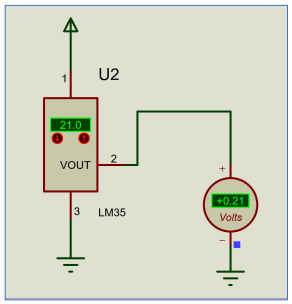
bit 2-0 **ADCS2:ADCS0:** A/D Conversion Clock Select bits
111 = FRC (clock derived from A/D RC oscillator)(1)
110 = FOSC/64
101 = FOSC/16
100 = FOSC/4
011 = FRC (clock derived from A/D RC oscillator)(1)
010 = FOSC/32
001 = FOSC/8
000 = FOSC/2

ADCON2 = 0X24

28

El sensor de temperatura LM35

- Sensor de temperatura de rampa lineal
- Pendiente de 10mv/°C
- Alimentación 5V
- Reacción lenta ante cambios bruscos de temperatura



1 Features

- Calibrated Directly in Celsius (Centigrade)
- Linear + 10-mV/°C Scale Factor
- 0.5°C Ensured Accuracy (at 25°C)
- Rated for Full −55°C to 150°C Range
- Suitable for Remote Applications
- Low-Cost Due to Wafer-Level Trimming
- Operates From 4 V to 30 V
- Less Than 60-μA Current Drain
- Low Self-Heating, 0.08°C in Still Air
- Non-Linearity Only ±¼°C Typical
- Low-Impedance Output, 0.1 Ω for 1-mA Load

2 Applications

- Power Supplies
- Battery Management
- HVAC
- Appliances

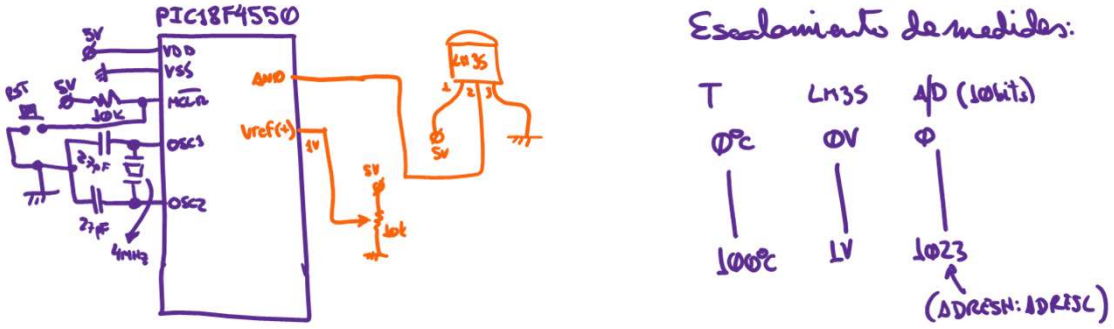
29

Consideraciones para usar el LM35 con el microcontrolador PIC18F4550

- Recortar la escala para una mayor facilidad en el tratamiento numérico. Ej 0°C a 100°C
- Si recortamos a ese rango (recordando 10mv/°C):
  - 0°C = 0V
  - 100°C = 1V
- El conversor A/D se deberá ajustar el Vref+ a un valor de 1V para que la señal adquirida obtenga todo el rango del A/D.

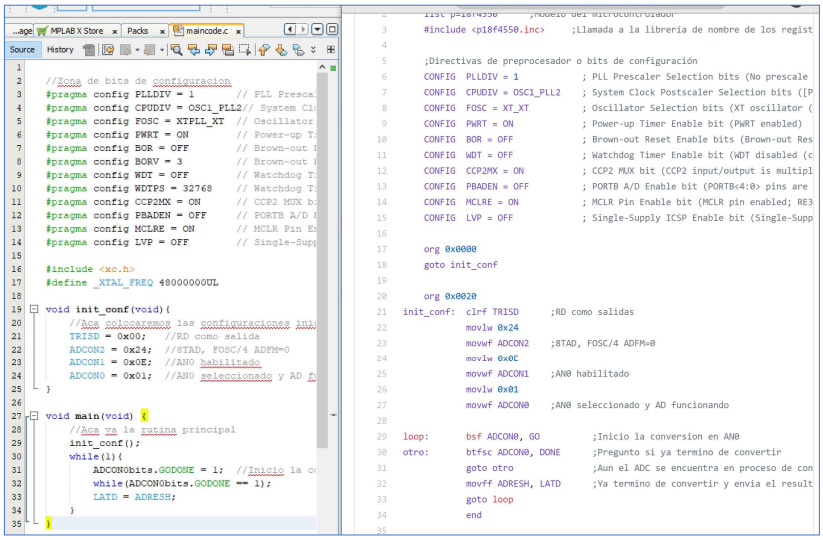
30

Circuito de interface del PIC18F4550 con el LM35



Ejemplo: Uso del A/D en XC8 y MPASM

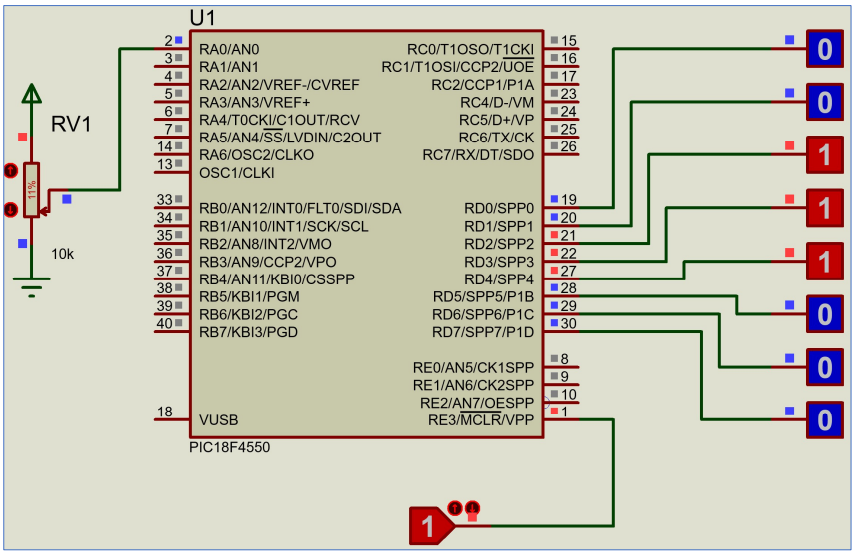
- XC8 vs MPASM
- Se puede apreciar que aparentemente hay menos líneas en XC8
- Al compilar el programa en XC8 se puede ver que ocupa 60bytes en memoria de programa, en MPASM ocupa una cuarta parte



```
//Zona de bits de configuracion
#pragma config PLLDIV = 1          // PLL Prescal
#pragma config CPUDIV = OSC1_PLL2// System Cl
#pragma config FOSC = XTPLL_XT   // Oscillator
#pragma config PWRT = ON         // Power-up T
#pragma config BOR = OFF         // Brown-out
#pragma config BORV = 3          // Brown-out
#pragma config WDT = OFF         // Watchdog T
#pragma config WDTPS = 32768     // Watchdog T
#pragma config CCP2MX = ON       // CCP2 MUX b
#pragma config PBADEN = OFF      // PORTB A/D
#pragma config MCLRE = ON        // MCLR Pin E
#pragma config LVP = OFF         // Single-Sup

#include <xc.h>
#define _XTAL_FREQ 48000000UL

void init_conf(void){
    //Aca colocaremos las configuraciones ini
    TRISD = 0x00;  //RD como salida
    ADCON2 = 0x24;  //8TAD, FOSC/4 ADFM=0
    ADCON1 = 0x0E;  //AN0 habilitado
    ADCON0 = 0x01;  //AN0 seleccionado y AD f
}

void main(void) {
    //Aca va la rutina principal
    init_conf();
    while(1){
        ADCON0bits.GODONE = 1;  //Inicio la c
        while(ADCON0bits.GODONE == 1);
        LATD = ADRESH;
    }
}
```

```
    #include <p18f4550.inc>     ;Llamada a la librería de nombre de los regist

    ;Directivas de preprocesador o bits de configuración
    CONFIG  PLLDIV = 1          ; PLL Prescaler Selection bits (No prescale
    CONFIG  CPUDIV = OSC1_PLL2  ; System Clock Postscaler Selection bits ([P
    CONFIG  FOSC = XT_XT        ; Oscillator Selection bits (XT oscillator (
    CONFIG  PWRT = ON           ; Power-up Timer Enable bit (PWRT enabled)
    CONFIG  BOR = OFF           ; Brown-out Reset Enable bits (Brown-out Res
    CONFIG  WDT = OFF           ; Watchdog Timer Enable bit (WDT disabled (c
    CONFIG  CCP2MX = ON         ; CCP2 MUX bit (CCP2 input/output is multipl
    CONFIG  PBADEN = OFF        ; PORTB A/D Enable bit (PORTB<4:0> pins are
    CONFIG  MCLRE = ON          ; MCLR Pin Enable bit (MCLR pin enabled; RE3
    CONFIG  LVP = OFF           ; Single-Supply ICSP Enable bit (Single-Supp

    org 0x0000
    goto init_conf

    org 0x0020
init_conf:  clrf TRISD      ;RD como salidas
            movlw 0x24
            movwf ADCON2    ;8TAD, FOSC/4 ADFM=0
            movlw 0x0E
            movwf ADCON1    ;AN0 habilitado
            movlw 0x01
            movwf ADCON0    ;AN0 seleccionado y AD funcionando

loop:       bsf ADCON0, GO          ;Inicio la conversion en AN0
otro:       btfsc ADCON0, DONE      ;Pregunto si ya termino de convertir
            goto otro               ;Aun el ADC se encuentra en proceso de con
            movff ADRESH, LATD      ;Ya termino de convertir y envia el result
            goto loop
            end
```

Ejemplo: Uso del A/D en XC8 y MPASM



Cuestionario:

- Ventajas y desventajas entre: XC8, CCS-C, mikroC

Fin de la sesión

- Laboratorio: Potenciómetros 10Kohm, LCD 16x2