

Microcontroladores

Semestre: 2022-1

Profesor: Kalun José Lau Gan

Semana 9: Lenguajes de alto nivel en microcontroladores

1

¿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

2

Preguntas previas:

- Consulta sobre el A/D:
 - Se lee un canal a la vez
 - ADCON2: Configuras base de tiempo de adquisición
 - ADCON1: Estableces cuáles van a ser canales analógicos
 - ADCON0: Escoges el canal ANx que vas a leer
- Sobre las interrupciones externas:
 - Si tienes mas de una fuente de interrupción externa, se evalua la bandera que se levantó, no el contenido del registro de puerto.
- Cuando estaba migrando programas del MPASM y XC8 PIC Assembler:
 - Los labels deben de tener al final ":"
 - El parámetro #bit de las instrucciones deben de ser numéricos:
 - Ejemplo: btfss STATUS, 2

3

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
- MCC: MPLAB Code Configurator

4

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.

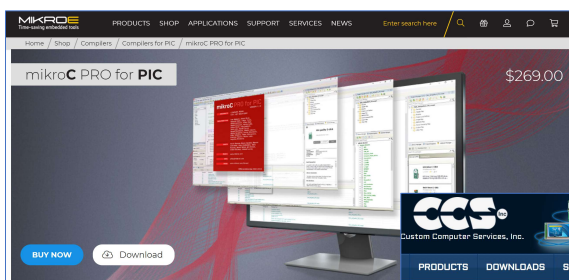
Problema de compatibilidad entre distintas plataformas de desarrollo aún empleando un mismo lenguaje.

- Basic
- C
- Java (muy poco utilizado y soportado)
- Python (Rpi Pico y el ESP32)

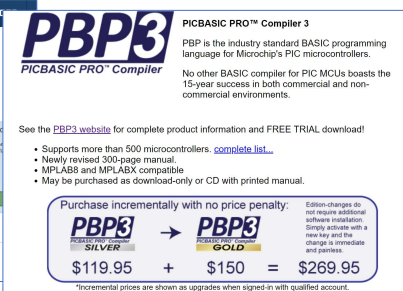
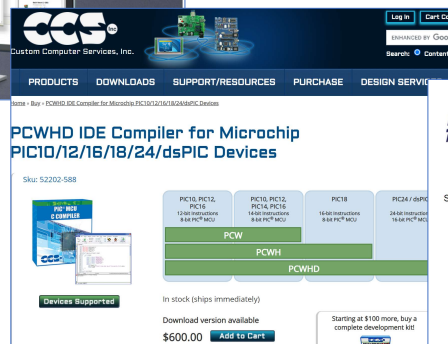


5

Panorama de los lenguajes de alto nivel para microcontroladores



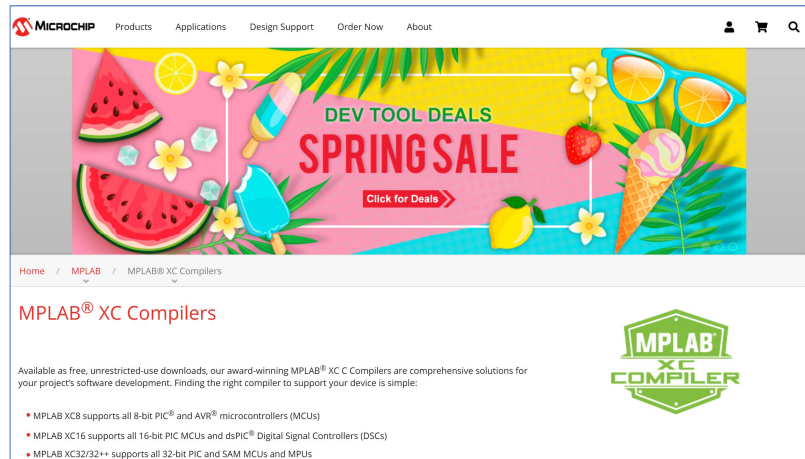
- Los compiladores de lenguajes de alto nivel (de terceros) mas empleados en la actualidad.
- Estas empresas ofrecen versiones DEMO para que el usuario pueda probar las bondades de cada uno.
- Cada empresa ofrece un pack de librerías especializadas para crear aplicaciones con periféricos especializados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



6

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



7

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
1+

USD per Unit
\$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

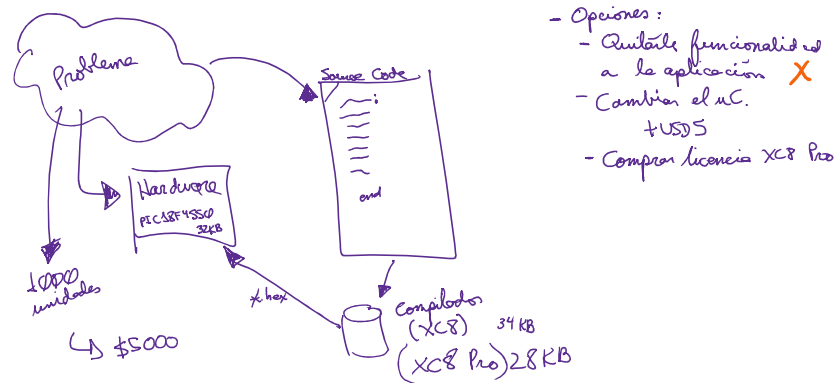
Additional quantities can ship by **11-Aug-2020**

Quantity:



8

Caso:



9

Optimización en el compilador XC

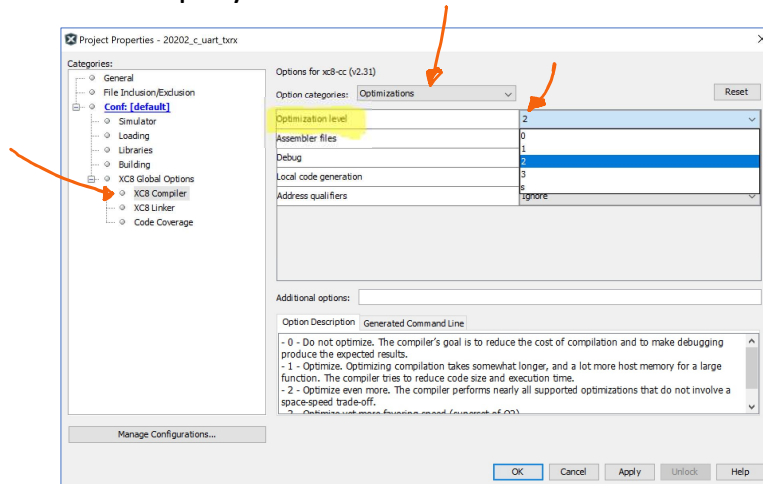
- **Optimization levels:**
 - Level 0: Fastest compilation time, minimal optimizations
 - Level 1: Optimizations with low debugging impact
 - Level 2: All optimizations that give the best balance between speed and size
 - Level 3: Program execution will be as fast as possible
 - Level s: Code size will 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
- Por defecto el nivel de optimización esta en 0.
- Para acceder al nivel 3 o nivel s requerimos tener la licencia PRO

10

Optimización en el compilador XC

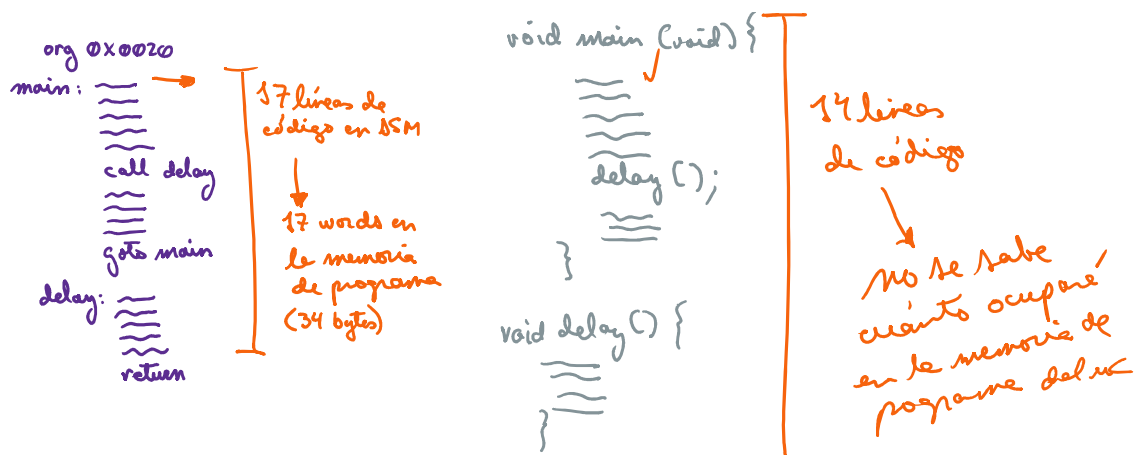
- Para acceder a las opciones de optimización ingresamos a las propiedades del proyecto:



11

Eficiencia de código en Assembler vs C

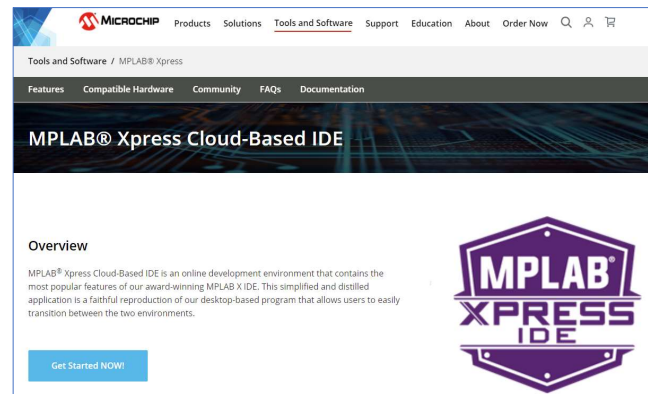
¿Por qué Assembler y por qué C?



12

El MPLAB Xpress

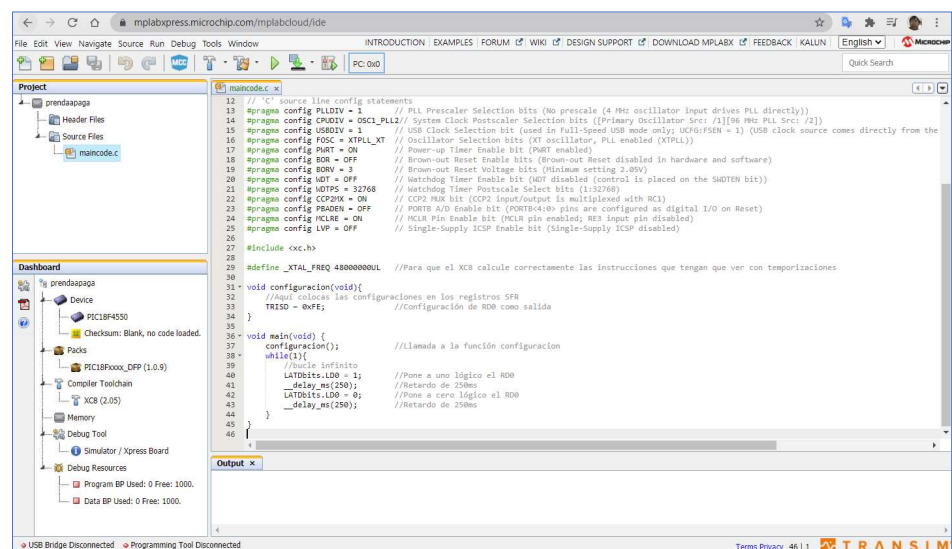
- Link:
<https://www.microchip.com/en-us/development-tools-tools-and-software/mplab-xpress>
- Entorno de trabajo online
- Permite realizar proyectos en XC sin tener que instalar el MPLAB X en el computador.



13

El MPLAB Xpress

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



14

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 v2.36:
 - <https://www.microchip.com/mplabxc8windows>

15

Plantilla de código en XC8:

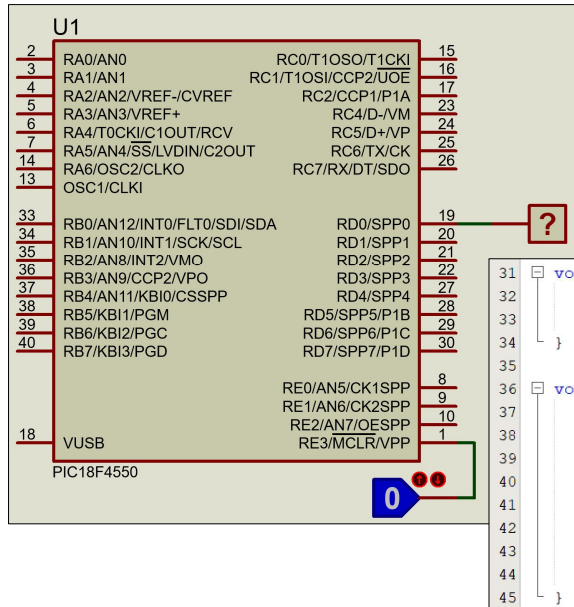
```

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 (PORTE
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      //Aquí colocas las configuraciones iniciales
29  }
30
31  void main(void) {
32      configuracion();
33      while (1) {
34          //Tu programa de usuario
35      }
36  }

```

16

Ejemplo en XC8: Titileo de LED



```

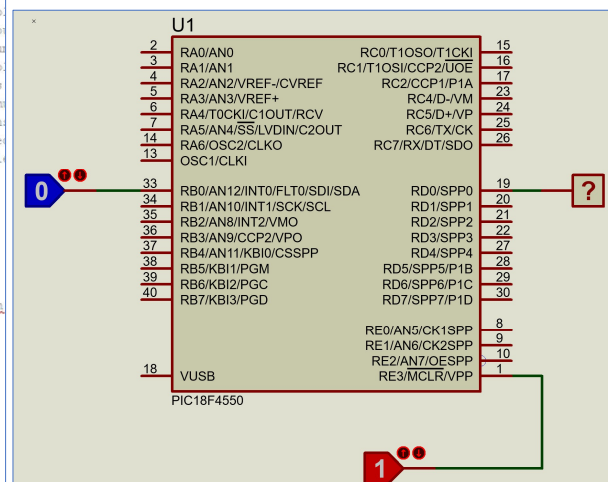
1  void configuracion(void) {
2      //Aquí colocas las configuraciones en los registros SFR
3      TRISD = 0xFE;           //Configuración de RD0 como salida
4  }
5
6  void main(void) {
7      configuracion();        //Llamada a la función configuracion
8      while(1){
9          //bucle infinito
10         LATDbits.LD0 = 1;    //Pone a uno lógico el RD0
11         __delay_ms(250);     //Retardo de 250ms
12         LATDbits.LD0 = 0;    //Pone a cero lógico el RD0
13         __delay_ms(250);     //Retardo de 250ms
14     }
15 }
  
```

17

Ejemplo en XC8: Negador lógico de un bit

```

1  #pragma config PLLDIV = 1      // PLL Prescaler Selection bits (No pre
2  #pragma config CPUDIV = OSC1_PLL2 // System Clock Postscaler Selection b
3  #pragma config FOSC = XTPLL_XT  // Oscillator Selection bits (XT oscill
4  #pragma config FWRT = ON        // Power-up Timer Enable bit (PWRT enab
5  #pragma config BOR = OFF        // Brown-out Reset Enable bits (Brown-o
6  #pragma config BORV = 3         // Brown-out Reset Voltage bits (Minimu
7  #pragma config WDT = OFF        // Watchdog Timer Enable bit (WDT disab
8  #pragma config WDTPS = 32768    // Watchdog Timer Postscale Select bits
9  #pragma config CCP2MX = ON      // CCP2 MUX bit (CCP2 input/output is m
10 #pragma config PBADEN = OFF      // PORTB A/D Enable bit (PORTB<4:0> pin
11 #pragma config MCLRE = ON        // MCLR Pin Enable bit (MCLR pin enable
12 #pragma config LVP = OFF        // Single-Supply ICSP Enable bit (Singl
13
14 #include <xc.h>
15 #define _XTAL_FREQ 4800000UL
16
17 void init_conf(void) {
18     //Aquí colocaremos las configuraciones iniciales de la aplicacion
19     //TRISDbits.RD0 = 0;           // RD0 como salida
20     asm("bcf TRISD, 0");         // Escribiendo instrucciones en mpsas
21 }
22
23 void main(void) {
24     init_conf();
25     while(1){
26         if (PORTBbits.RB0 == 1) {
27             LATDbits.LD0 = 0;
28         }
29         else{
30             LATDbits.LD0 = 1;
31         }
32     }
33 }
34
  
```



18

Ejemplo en XC8: Compuerta XOR

Handwritten red notes: *perturbaciones* (with arrow to VUSB) and a red '1' in a box next to output C.

```

28 void configuracion(void) {
29     TRISDbits.RD0 = 0;           //Puerto RD0 como salida
30 }
31
32 void main(void) {
33     configuracion();
34     while(1) {
35         //Bucle infinito
36         //Con switch-case
37         switch (PORTB & 0x03) {
38             case 0x00:
39                 LATDbits.LD0 = 0;
40                 break;
41             case 0x01:
42                 LATDbits.LD0 = 1;
43                 break;
44             case 0x02:
45                 LATDbits.LD0 = 1;
46                 break;
47             case 0x03:
48                 LATDbits.LD0 = 0;
49                 break;
50         }
51     }
52 }

```

19

Cuestionario:

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

20

Fin de la sesión

- Laboratorio: Potenciómetros 10Kohm, LCD 16x2