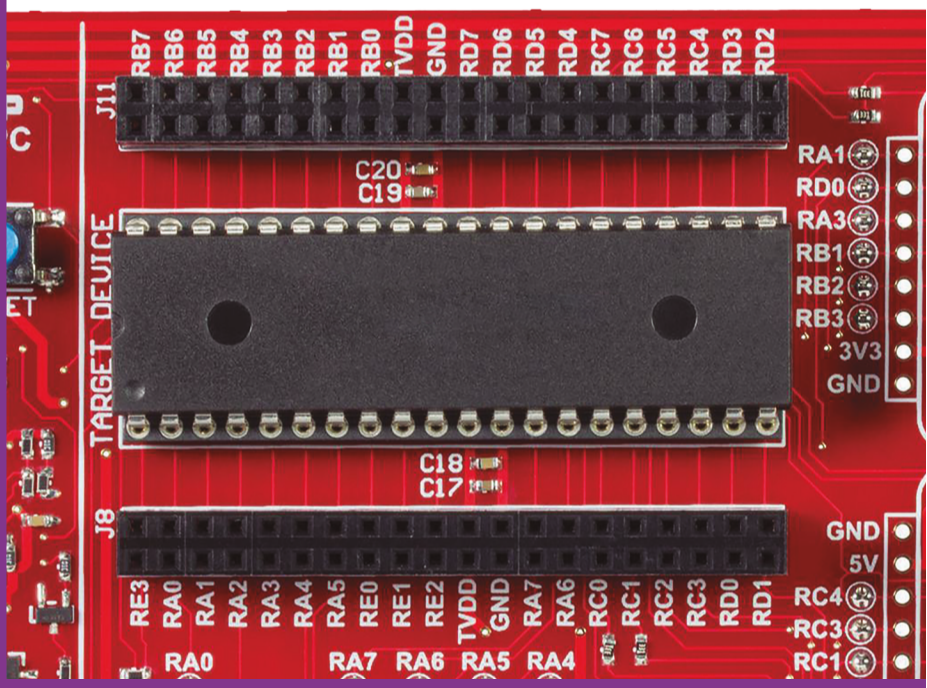
UNIVERSIDAD AUTÓNOMA METROPOLITANA
Casa abierta al tiempo
Unidad Lerma

Dirección
Ciencias Básicas
e Ingeniería

MANUAL DE PRÁCTICAS



Laboratorio para el curso Sistemas Basados en Microcontroladores

Departamento de Recursos de la Tierra

Departamento de Procesos Productivos

Departamento de Sistemas de Información y Comunicaciones

UNIVERSIDAD AUTÓNOMA METROPOLITANA
Casa abierta al tiempo

Primera edición 2020

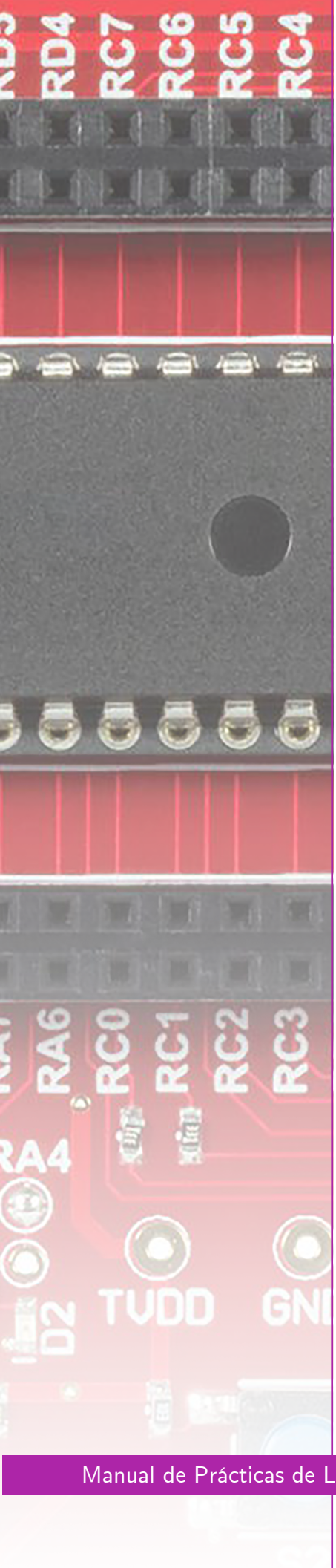
Av. de las Garzas 10, Col. El Panteón, CP 52005,
Lerma de Villada, México.

Diseño editorial: LDG José Uriel Hernández Pérez

Manual de prácticas de Laboratorio para el curso Sistemas Basados en Microcontroladores

Gerardo A. Laguna Sánchez
Celso Márquez Sánchez

CONTENIDO



I. INTRODUCCIÓN

La noción de microcontrolador se refiere a todo dispositivo microprocesador que, además de la unidad de procesamiento central, cuenta con cierta cantidad de dispositivos periféricos dentro del mismo circuito integrado.

En este manual de prácticas de laboratorio para la Unidad de Enseñanza Aprendizaje (UEA) Sistemas Basados en Microcontroladores, o cualquier otra UEA equivalente, se presenta el desarrollo, paso a paso, para la realización de 7 programas de aplicación usando la tarjeta de desarrollo *Curiosity HPC*, de la empresa Microchip, que incluye un microcontrolador de propósito general.

El desarrollo de estos 7 programas se distribuye en igual número de prácticas de laboratorio, presentando los conceptos básicos y desarrollando habilidades prácticas de manera incremental para que, al terminar las prácticas, el alumno se encuentre en condiciones de desarrollar de su propio proyecto durante el tiempo que se le asigne dentro del curso.

Cada una de las prácticas presentadas incluye los antecedentes, el objetivo de la práctica, el material necesario y el desarrollo paso a paso. Al finalizar la práctica, se indica la realización de alguna actividad o experimento adicional a fin de fomentar el análisis técnico del alumno respecto de los códigos presentados en la práctica. Por último, toda práctica debe reportarse con un documento que considere

las siguientes secciones y recomendaciones:

- Título de la práctica.
- Objetivo de la práctica.
- Introducción con marco teórico y conceptual. No "copiar y pegar" ningún contenido o recurso digital. Parafrasear los contenidos e indicar la fuente consultada.
- Materiales y procedimiento (o metodología) sintético. No repetir textualmente el procedimiento de la práctica, parafrasear y sólo indicar los pasos a groso modo.
- Resultados y discusión de los resultados. Aquí van las imágenes (Incluyan, al menos, una imagen fotográfica de su experimento) y las tablas con las observaciones y mediciones, además de los comentarios al respecto.
- Conclusiones. En esta sección se indica si se cumplió o no el objetivo de la práctica, así como sus impresiones generales sobre la actividad.
- Referencias de fuentes consultadas.

Aunque este manual de prácticas permite experimentar en el laboratorio con sistemas soportados con microcontroladores en forma auto-contenida, es muy conveniente acompañar el desarrollo de las mismas con la teoría indispensable.

Por ello, adicionalmente y con independencia de que los alumnos incluyan en su reporte de cada práctica el marco conceptual con las ideas clave, el profesor responsable de la conducción de las prácticas tiene, al menos, tres opciones para cubrir la parte de teoría necesaria. Con base en lo que se solicita en el reporte de cada práctica, el profesor puede: 1) Presentar una exposición al respecto antes del

desarrollo de la práctica; 2) Dejar tarea para que los alumnos investiguen el tema antes de la sesión de laboratorio; o 3) Dejar que los alumnos investiguen el tema después de la sesión de laboratorio para que su experiencia práctica les permita enfocarse en los temas importantes.

No se debe olvidar que el desarrollo de las prácticas es inseparable de un trabajo de investigación mínimo por parte de los alumnos. Es justo este trabajo de investigación el que deben reflejarse en el marco teórico conceptual dentro del reporte. Por lo demás, si el profesor así lo considerara pertinente, puede solicitar que los alumnos lleven una bitácora para sustentar el desarrollo de cada práctica.

En cada práctica se presenta, al final, la Referencias donde el alumno puede encontrar las fuentes necesarias para cubrir los aspectos teóricos solicitados en el reporte, así como para profundizar en el conocimiento y manejo de los temas trabajados durante la práctica.

Los autores de este manual de prácticas confían en que el material proporcionado sea de gran ayuda, tanto para los profesores como para los alumnos, a fin de contribuir con el cabal cumplimiento de los objetivos académicos señalados en el programa de estudio correspondiente. Adicional a lo anterior, este manual de prácticas también puede ser de gran ayuda para cualquier persona que esté interesada en el desarrollo de aplicaciones de procesamiento con microcontroladores y que haya tomado, al menos, un curso teórico de programación básica y uno de diseño lógico u otro equivalente.

Breve descripción de la arquitectura del microcontrolador Pic16F18875

La familia de microcontroladores PIC16(L) F1885x/7x, del fabricante Microchip, se caracteriza por ofrecer bloques de periféricos independientes que incluyen diferentes funciones, desde el ámbito de lo analógico hasta el de las comunicaciones digitales. Esta familia se caracteriza por su bajo consumo de energía, lo que la convierte en una alternativa ideal para aplicaciones de propósito general que requieren maximizar el aprovechamiento de la fuente de alimentación.

Los microcontroladores de esta familia cuentan con diferentes alternativas para la detección de fallas y su correspondiente re-inicialización, a fin de garantizar una operación segura de las aplicaciones que se desarrollen con ellos. Dos características notables de estos microcontroladores son que pueden contar con hasta 56 KB de memoria Flash, para el almacenamiento del código, y que sus conversores analógicos-digitales (ADC, por sus siglas en inglés) ofrecen una resolución de 10 bits, incluyendo algunas facilidades de pre-procesamiento de las señales que disminuyen la complejidad de las aplicaciones.

En general, las características de esta familia incluyen lo siguiente:

- Núcleo de procesamiento con 49 instrucciones y una pila (stack) de 16 niveles.
- Oscilador interno configurable con una velocidad máxima de 32 MHz.
- 2 módulos de modulación de ancho de pulso (PWM, por sus siglas en inglés) de 10 bits.
- 5 módulos de comparación y captura PWM.
- Generador de señales complementario.

- Oscilador controlado numéricamente (NCO, por sus siglas en inglés).
- 4 celdas lógicas configurables.
- Módulo ADC de 10 bits con pre-procesamiento de señales.
- 5 módulos de conversión digital a analógico (DAC, por sus siglas en inglés).
- Bloque modulador de señales a partir de datos.
- Detección de cruce por ceros.
- Bloque de verificación de código de redundancia cíclica (CRC, por sus siglas en inglés).
- Temporizador para garantizar la integridad del código ("perro guardián") con ventaneo (WWDT, por sus siglas en inglés).
- Reinicio automático por variaciones en la fuente de alimentación.
- Circuito integrado para la programación serial del dispositivo.
- Alimentaciones en el intervalo 1.8V-3.6V, para la serie LF, y de 2.3 a 5.5, para la serie F.

Para mayor información sobre la arquitectura y funciones de esta familia, el lector puede remitirse a la documentación completa que se encuentra disponible, en línea, en la página del fabricante:

https://www.microchip.com/wwwproducts/en/PIC16F18875

Descripción gráfica de la tarjeta de desarrollo *Curiosity HPC*

En la figura A se proporciona una imagen comentada sobre la tarjeta de desarrollo *Curiosity HPC*, a fin de que los usuarios tengan una referencia rápida sobre sus componentes más usados en las prácticas de este documento. Para mayor información sobre la tarjeta y su documentación completa, el lector puede remitirse a la página del fabricante:

https://www.microchip.com/Developmenttools/ProductDetails/DM164136

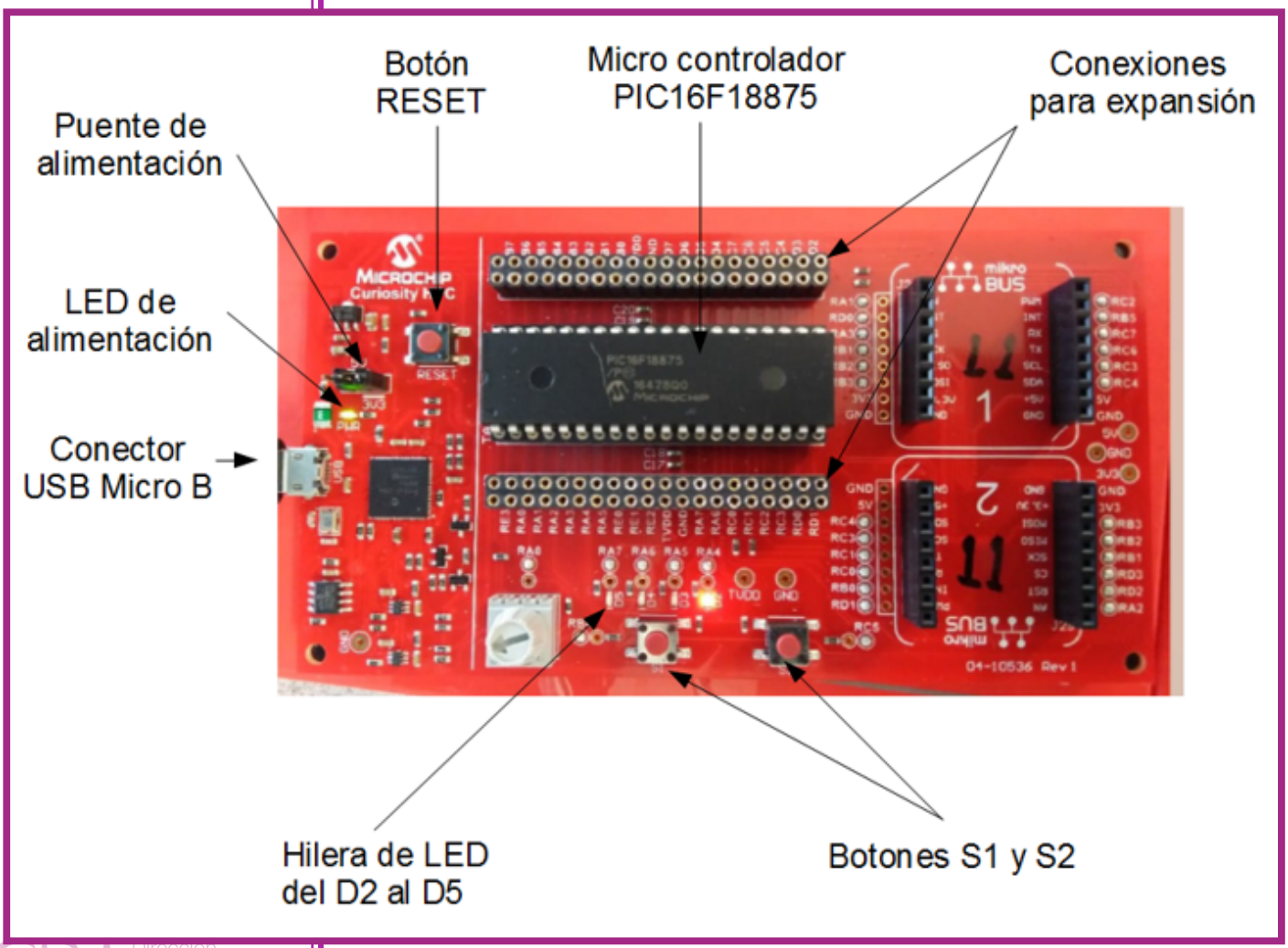
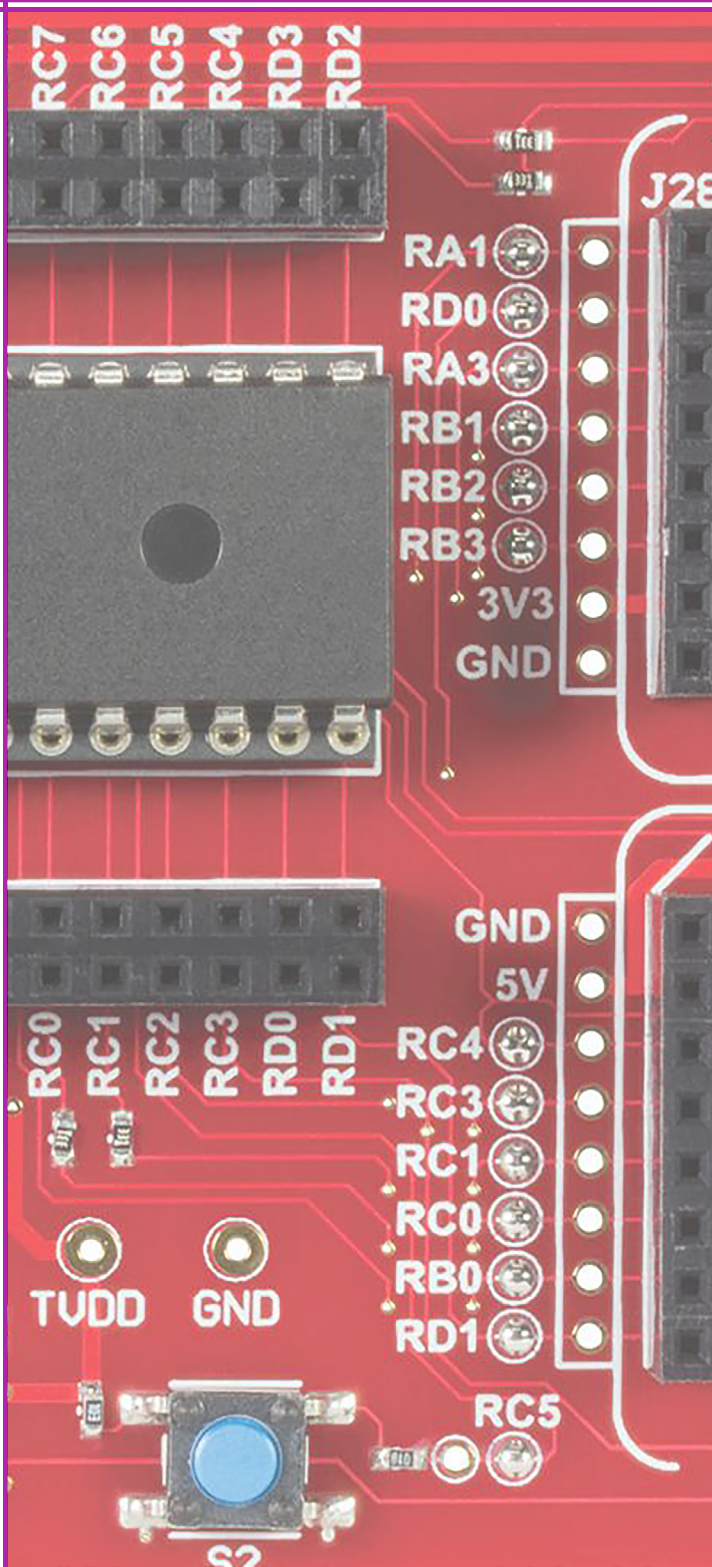


Figura A. La tarjeta de desarrollo *Curiosity HPC.*

PRÁCTICA 1

El entorno de desarrollo



Dirección
Ciencias Básicas
e Ingeniería

OBJETIVOS

Conocer el entorno de desarrollo MPLAB IDE de Microchip e identificar los pasos básicos en la construcción de un proyecto con microcontroladores PIC.

ANTECEDENTES

De forma muy general, un microcontrolador es un circuito integrado programable que ejecuta ciertas instrucciones programadas previamente. Para poder programar dichos microcontroladores se requiere de software y hardware especializado. Dentro del software se encuentran los entornos de desarrollo integrado (IDE, por sus siglas en inglés), con los cuales es posible, entre otras cosas, escribir, editar y ensamblar los programas que se desea ejecutar en el microcontrolador. Este software puede ser proporcionado por los mismos fabricantes del microcontrolador. Por ejemplo, Microchip proporciona el software IDE MPALAB de forma gratuita.

De acuerdo con Microchip, MPLAB es un software modular y altamente configurable que incorpora poderosas herramientas que ayudan a configurar, desarrollar, depurar y evaluar los diseños de sistemas que emplean la mayoría de los microcontroladores y controladores de señales digitales fabricados por Microchip. Además, el entorno IDE MPLAB permite la descarga y programación de los códigos en los microcontroladores de Microchip.

Por otro lado, el dispositivo de programación es el hardware especializado que se conecta a la computadora y que permite grabar el código, previamente ensamblado o compilado, en la memoria del microcontrolador. Algunos fabricantes como Microchip tienen disponibles tarjetas

Práctica 1
El entorno
de desarrollo

de desarrollo, como la *Curiosity HPC*, que ya incluyen el programador, con lo cual se facilita mucho su uso.

Específicamente, la tarjeta de desarrollo *Curiosity HPC*, es una tarjeta que está dirigida a usuarios que desean experimentar, incluso fabricantes, con los microcontroladores de la marca Microchip mediante una tarjeta de prototipado rápido. Esta tarjeta permite aprovechar completamente el entorno de desarrollo MPLAB de Microchip e incluye un programador/depurador, integrado en la misma tarjeta, por lo cual no requiere ningún hardware adicional para funcionar.

MATERIAL

- Tarjeta modelo *Curiosity HPC* de marca Microchip.
- Computadora con software MPLAB X IDE v4.05.

PROCEDIMIENTO

Antes de conectar la tarjeta *Curiosity*, asegúrese de que cuenta con el puente de alimentación, ya sea en la posición de 3.3V (3V3), cuando se alimenta mediante el cable USB, o la de 5V, si se emplea una fuente de alimentación externa. Independientemente de lo anterior, la tarjeta Curiosity siempre debe programarse con bajo voltaje (*Low-voltage programming* o LVP), por ello hay que asegurarse que se encuentre seleccionada esta opción mediante el bit LVP en las palabras de configuración ($CONFIG4$). [1] Conecte la tarjeta a la computadora de mediante un cable USB A-USB Micro B y compruebe que el led PWR (verde) enciende.

[1] Respecto de este último punto, el alumno no debe preocuparse de esto cuando emplee el código proporcionado con las prácticas, en tanto que el bit LPV en todas ellas está activo. Sólo debe tener en cuenta esta recomendación si va a emplear la tarjeta para un proyecto y código propio.

1.- Iniciar la aplicación MPLAB IDE.

Una vez iniciado el programa, aparecerá la ventana de la figura 1.1, que es la vista principal del entorno de desarrollo.

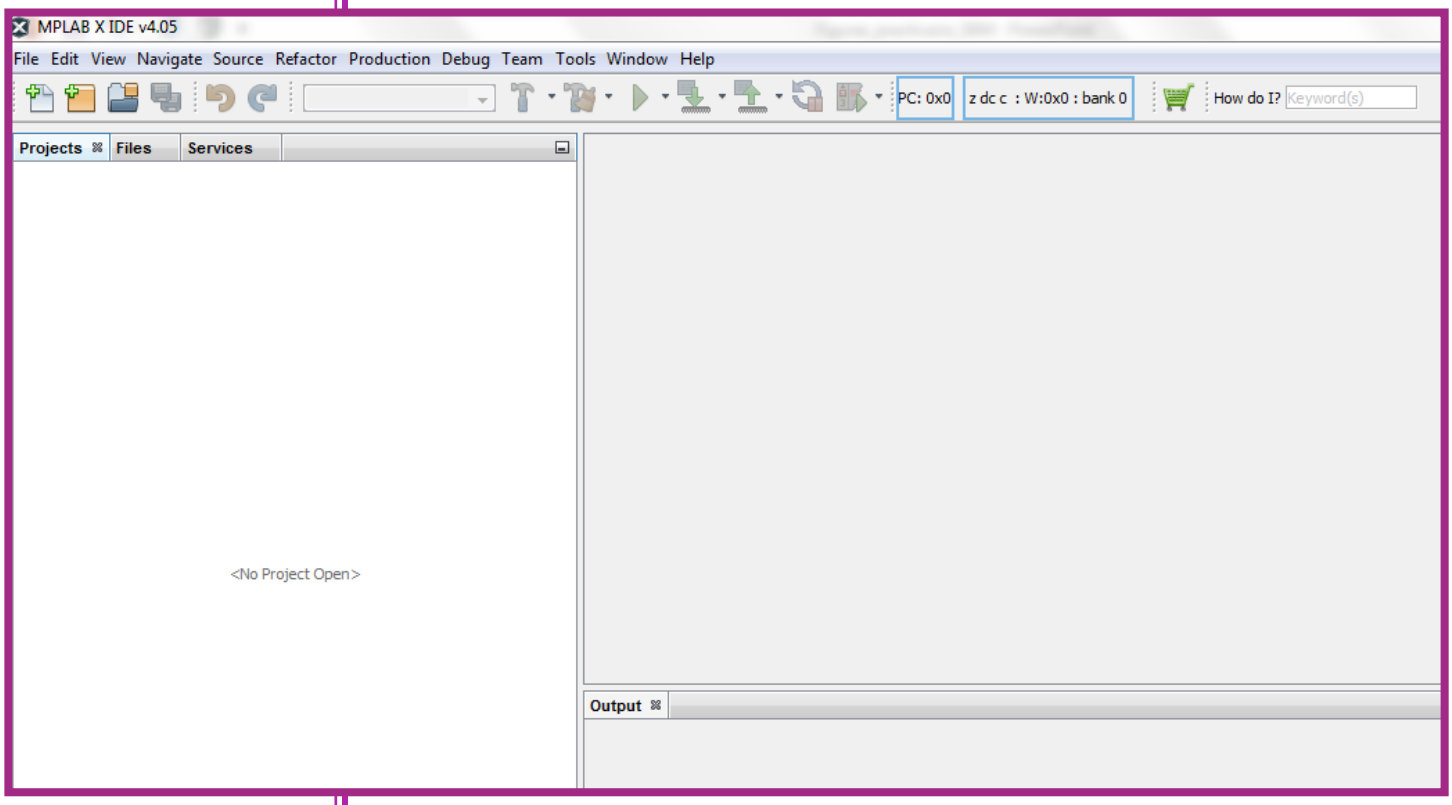


Figura 1.1. Ventana principal del entorno de programación MPLAB.

2.- Crear un nuevo proyecto.

2.1. Inicio del nuevo proyecto.

En el entorno de desarrollo, vaya al menú File y elija la opción “New Project...”.

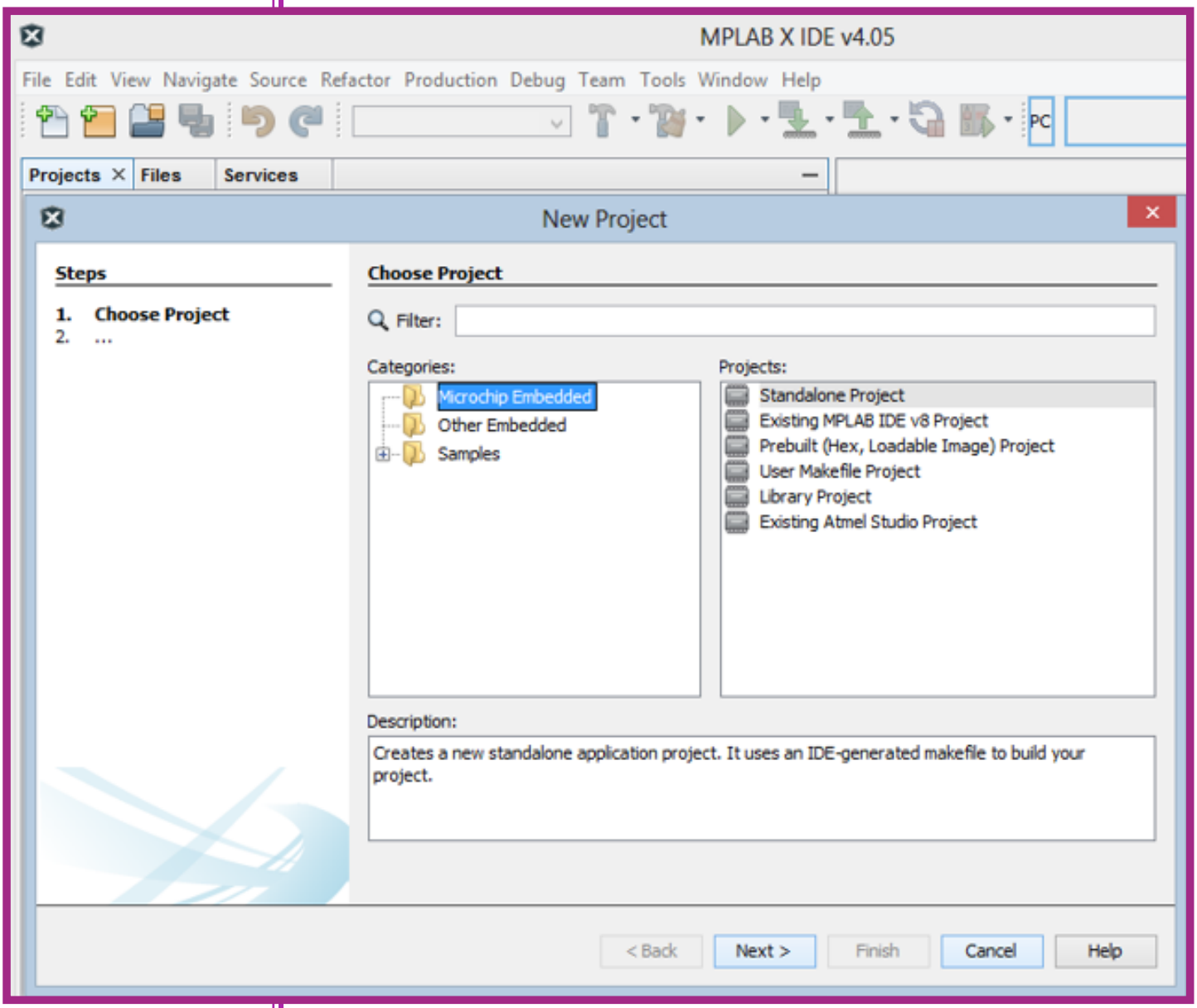


Figura 1.2. Ventana para un nuevo proyecto.

2.2. Elección del tipo de proyecto.

Entonces, se abre la ventana wizard "New Project" (ver la figura 1.2). Elija la opción por defecto "Standalone Project" de la categoría "Microchip Embedded", oprimiendo el botón <Next>. A continuación, en la ventana para seleccionar el dispositivo (ver la figura 1.3), coloque el número de parte PIC16F18875 en la ventana Device. Oprima <Next> para continuar.

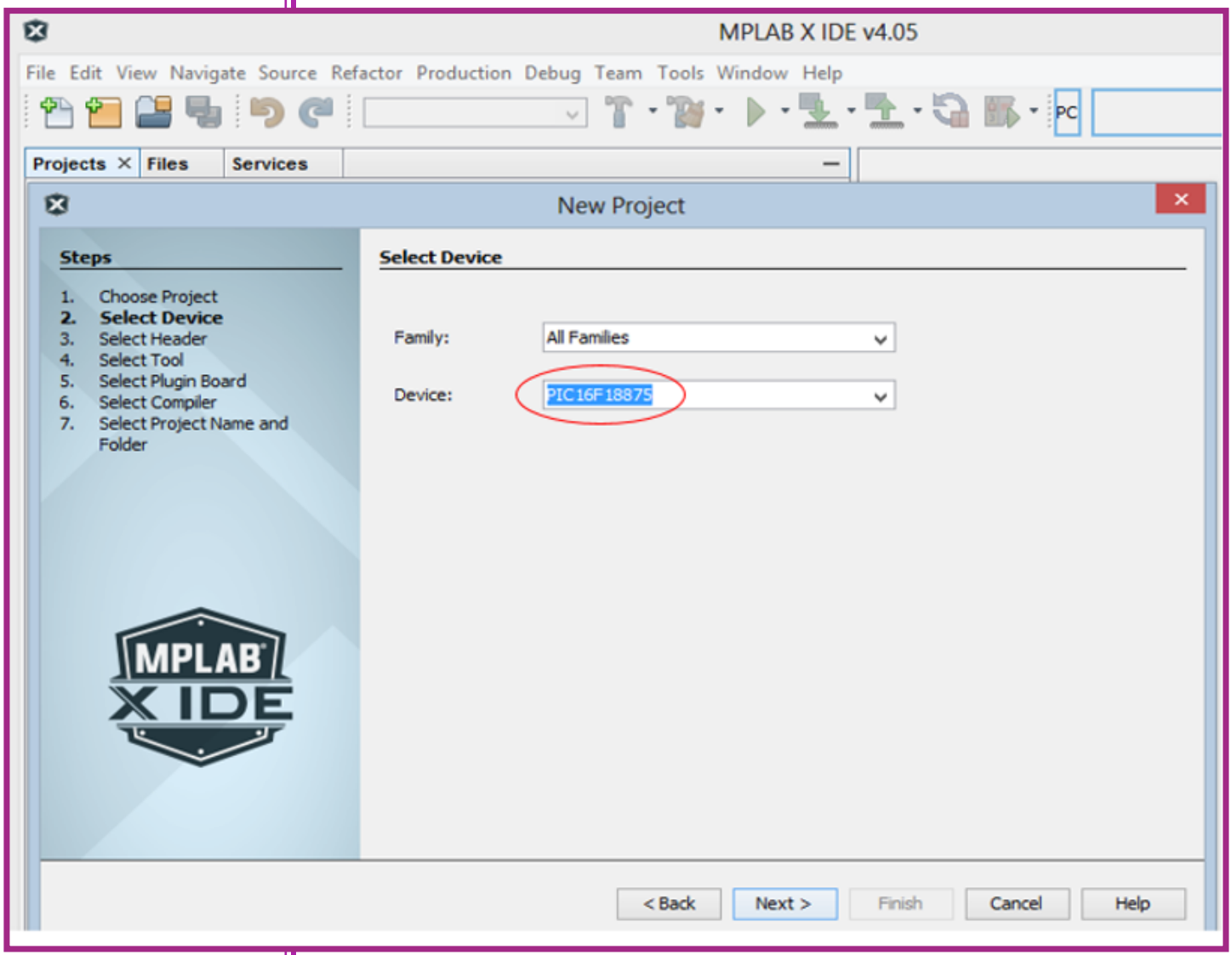


Figura 1.3. Ventana para elegir al dispositivo.

2.3. Elección del dispositivo.

En la nueva ventana se muestran las herramientas de trabajo (ver la figura 1.4), debe aparecer la rama Starter Kits (PKOB) y, dentro de ella, la rama Curiosity. Oprima <Next> para continuar.

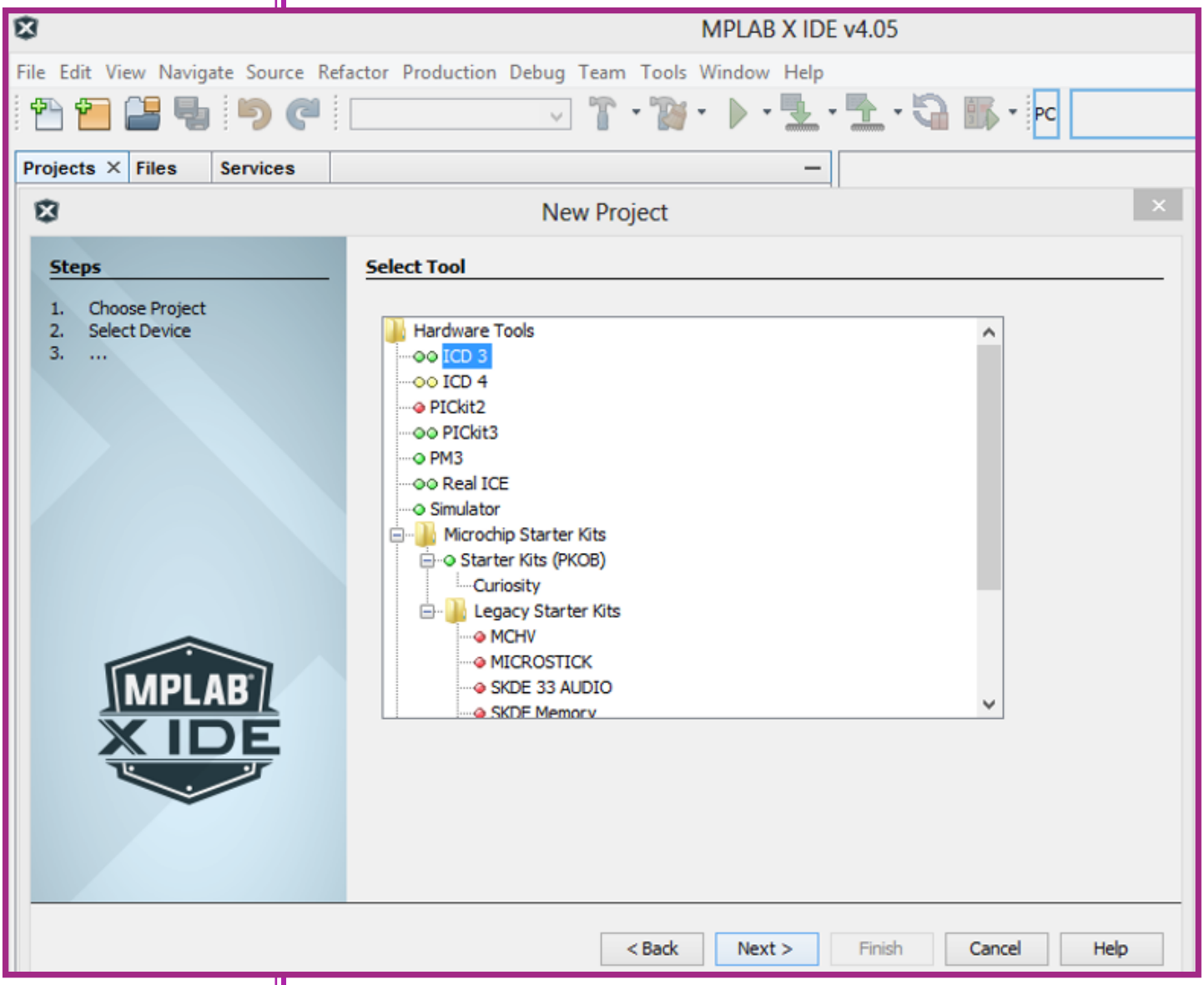


Figura 1.4. Ventana elegir la herramienta de trabajo.

2.4. Elección del compilador/ensamblador.

Ahora debe seleccionar el compilador/ensamblador que se va a emplear (ver la figura 1.5). Marque la opción mpasm para emplear el ensamblador. Oprima <Next> para continuar.

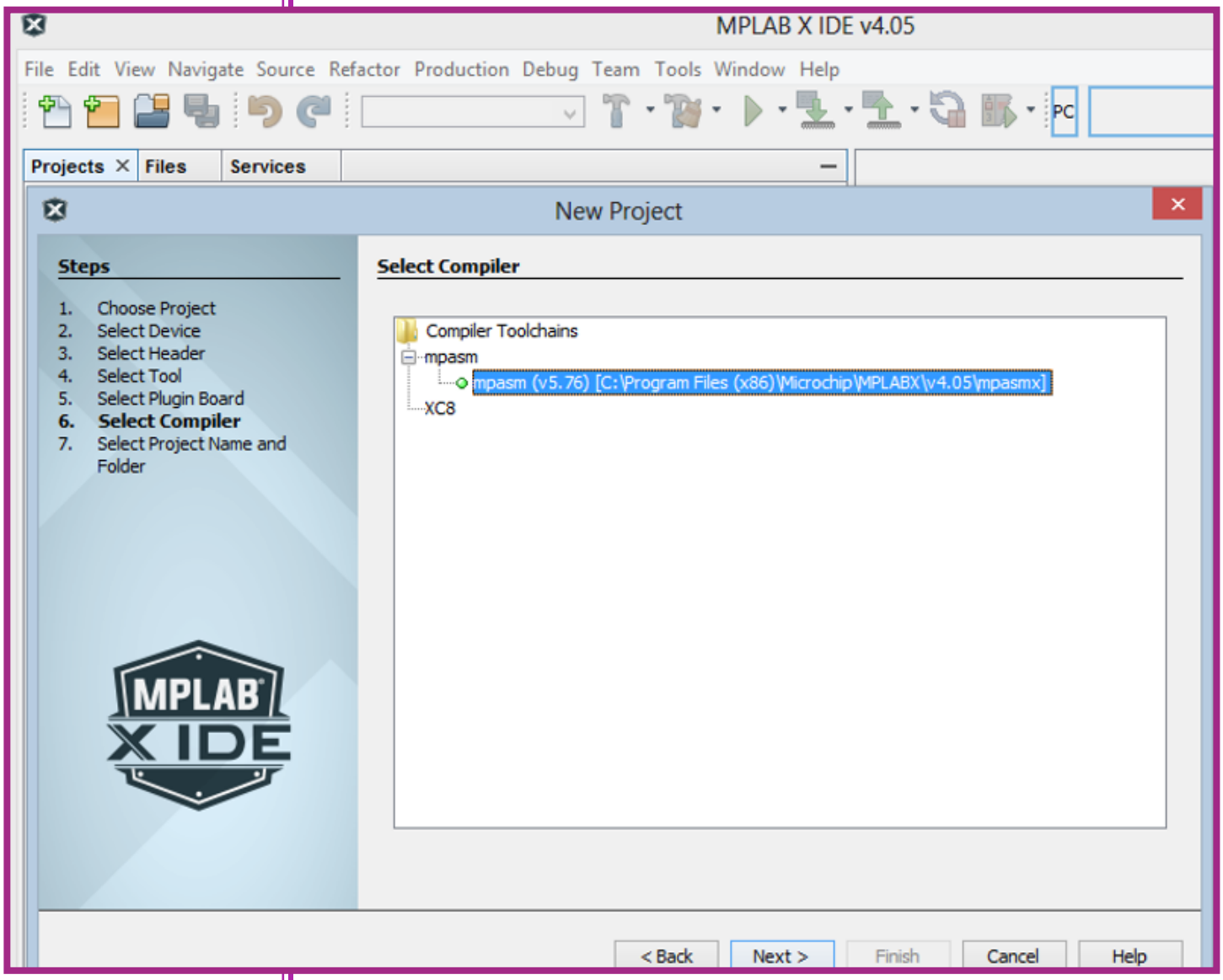


Figura 1.5. Ventana para elegir el compilador/ensamblador.

2.5. Nombre y ruta del proyecto. Entonces puede introducir el nombre del proyecto, en este caso "Practica01", y definir la ruta del proyecto (ver la figura 1.6). La ruta no debe contener espacios. Dejar las demás opciones con los valores por defecto y oprima el botón <Finish>.

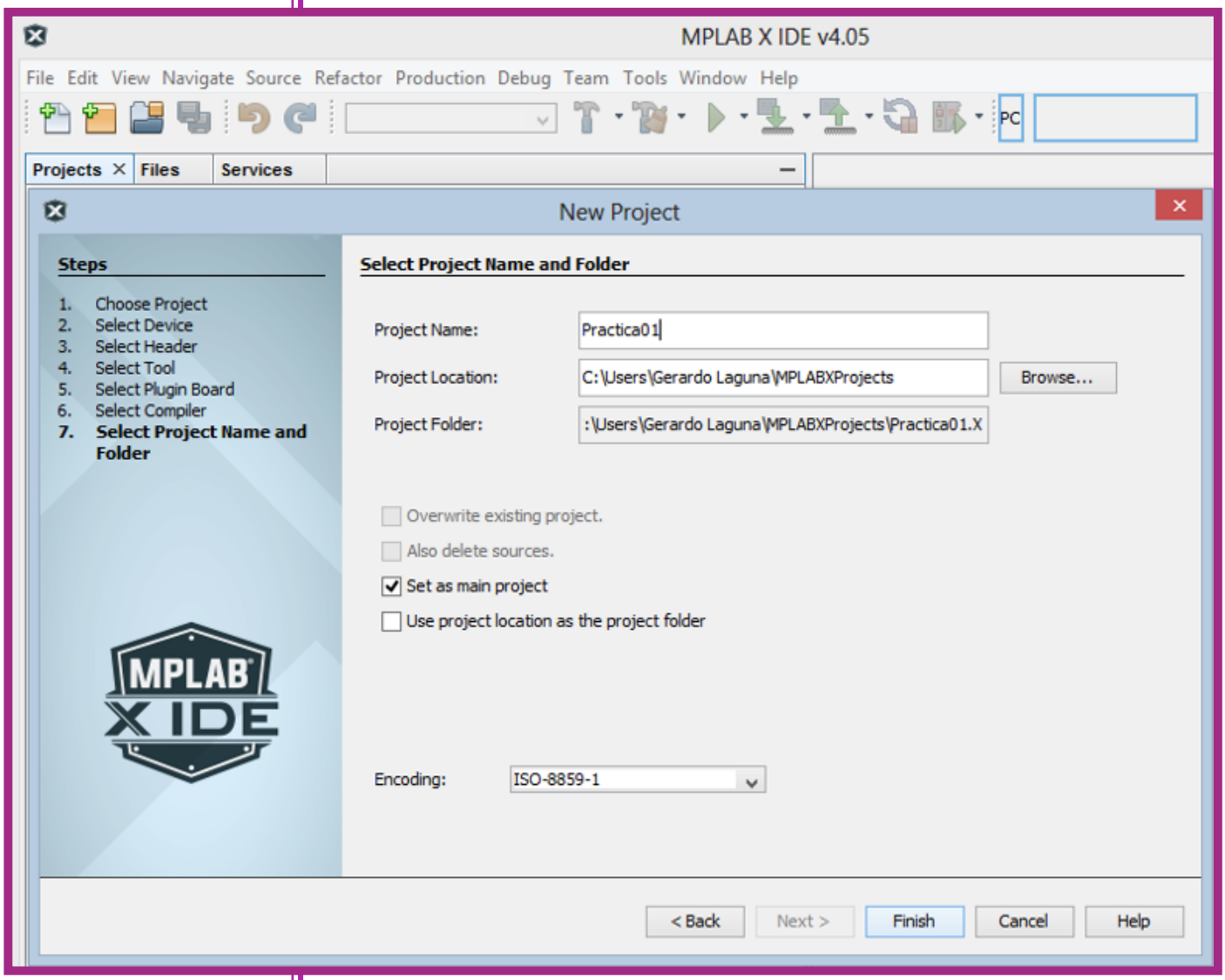


Figura 1.6. Ventana especificar el nombre y la ruta del proyecto.

2.6. Agregar un nuevo archivo para el código fuente.

En la ventana principal del entorno MPLAB, del lado izquierdo (ver la figura 1.7) aparece el árbol de archivos del proyecto. Oprima el botón derecho del ratón sobre la rama Source Files. Elija New y "pic_8b_general.asm...". En su defecto, para agregar un Nuevo archivo, puede oprimir la combinación de teclas [Ctrl+N].

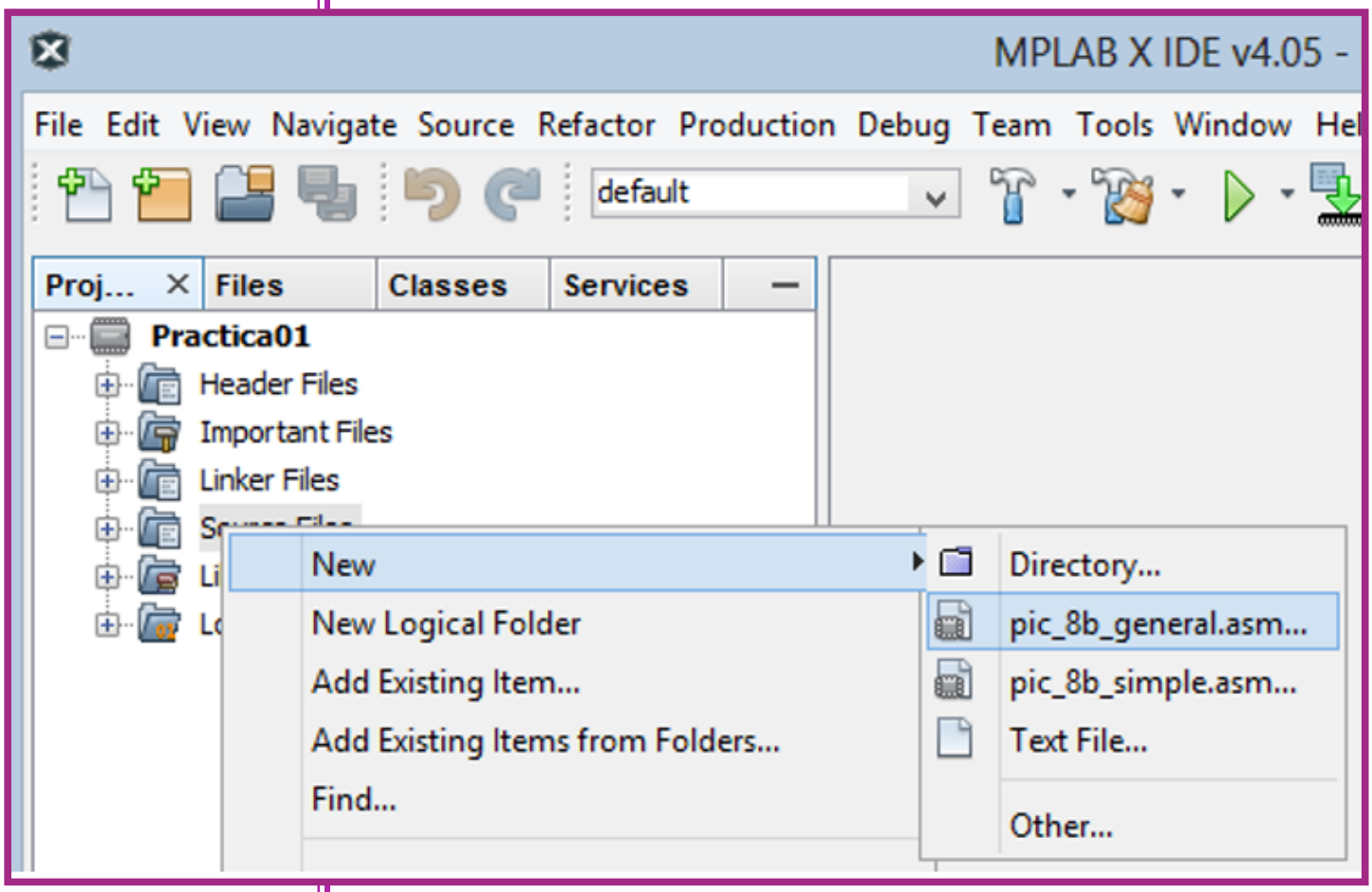


Figura 1.7. Pasos para agregar un nuevo archivo de código fuente.

2.5. Nombre y ruta del nuevo archivo.

Ahora, aparece la ventana de la figura 1.8, coloque el nombre del nuevo archivo asm, por ejemplo, main01.asm. Finalmente oprima <Finish>:

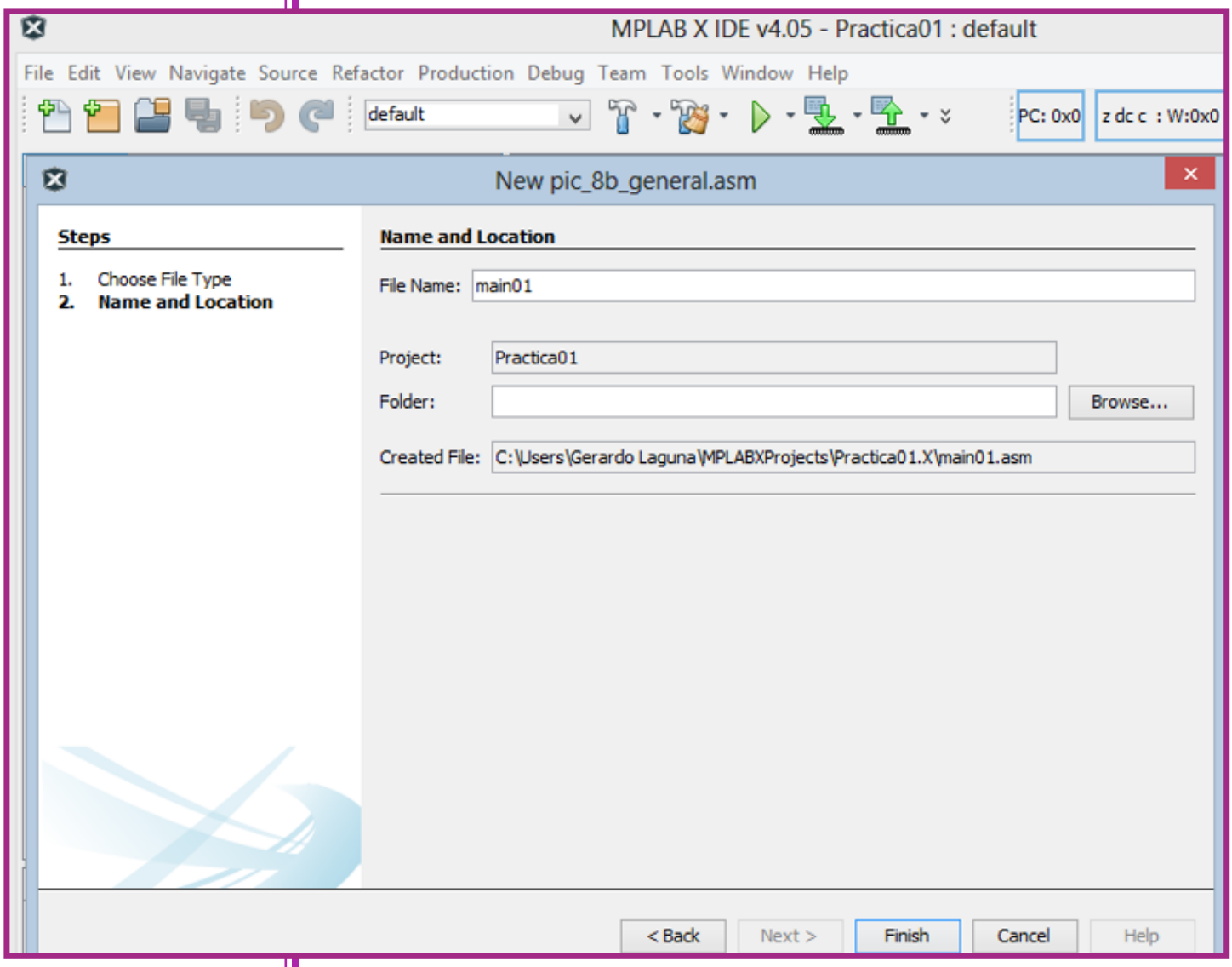


Figura 1.8. Ventana especificar el nombre y la ruta del archivo.

En forma automática, se genera un código de ejemplo, mayormente comentado, como se puede apreciar en la figura 1.9. Nótese que los comentarios inician con el carácter ';' (punto y coma). En particular y por defecto, en el entorno de desarrollo MPLAB, los comentarios se identifican con el color verde.

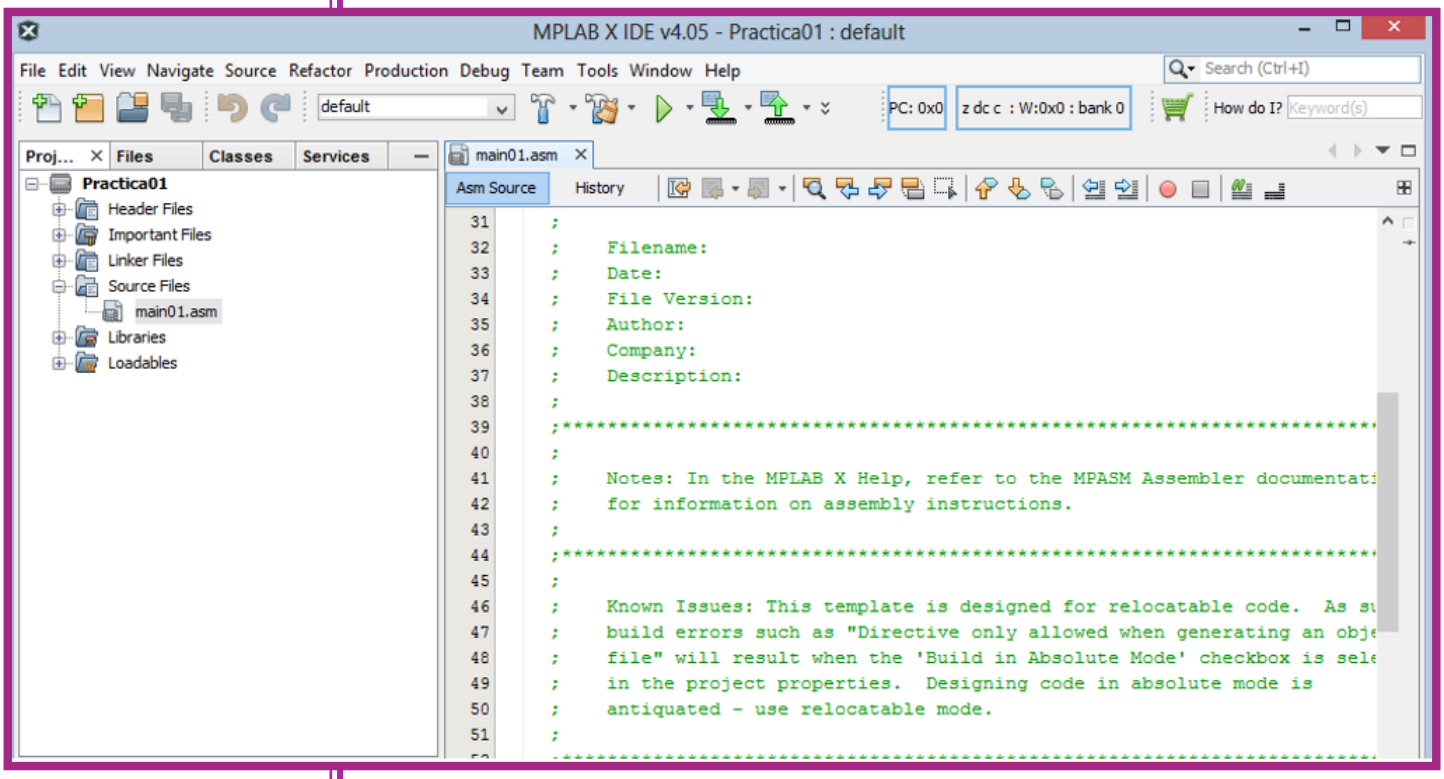


Figura 1.9. Nuevo archivo de código, con un ejemplo pre-llenado y comendado.

3.- Agregar el código fuente propio.

Sustituya el código de ejemplo por el que aparece en el anexo 1 (SBM_source_01.asm), para que aparezca en el entorno de programación, tal y como se muestra en la figura 1.10. Se trata de un programa que en su cuerpo contiene un lazo infinito que invoca a una subrutina de retardo a fin de prender y apagar intermitentemente al LED D2. Los retardos asumen cierta frecuencia del reloj de sistema (1 MHz). El programa se mantendrá operando mientras la tarjeta no se apague. Si la tarjeta se desconecta, el programa no se pierde porque ha quedado almacenado en su memoria Flash.

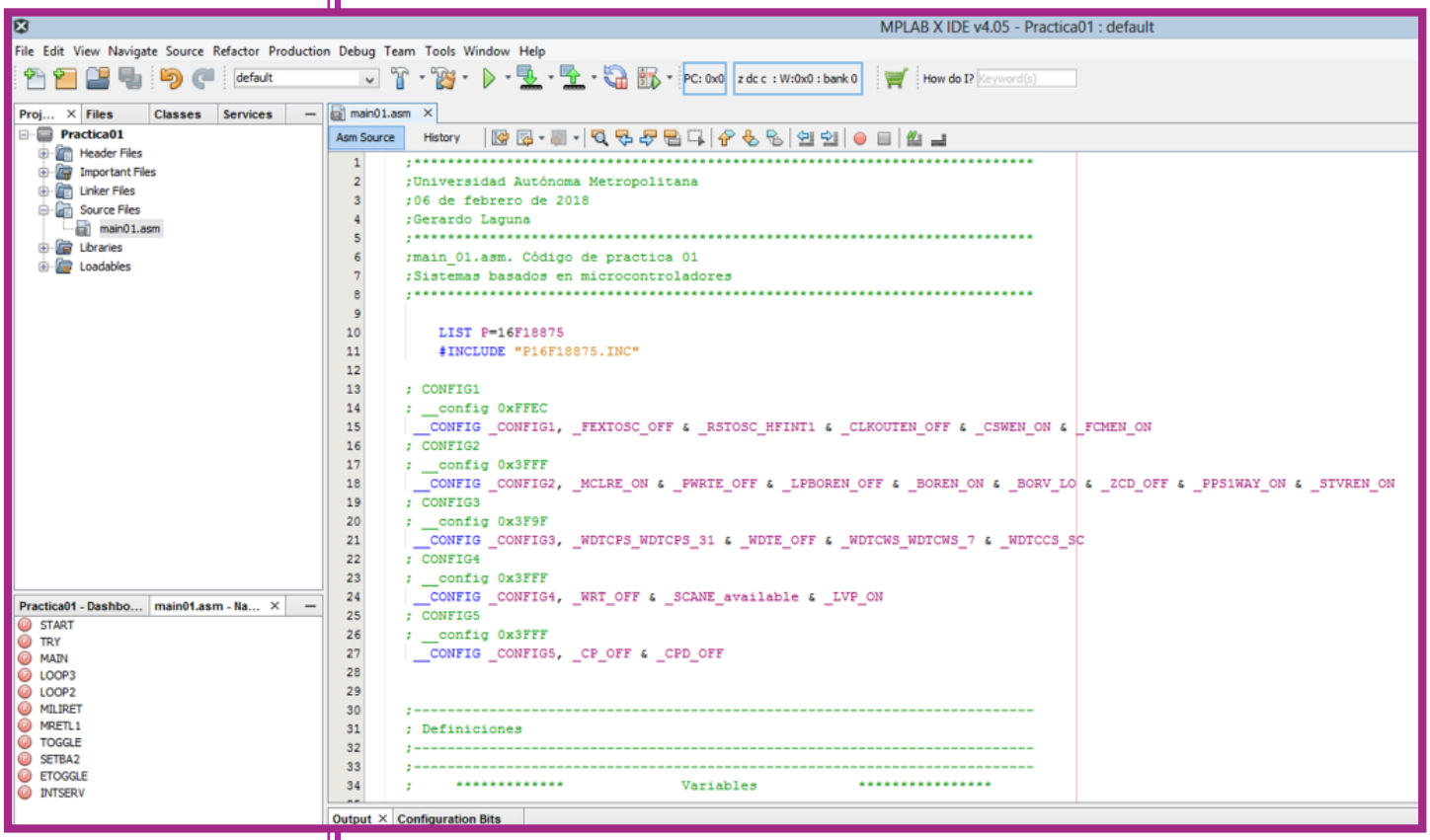


Figura 1.10. Sustitución del código de ejemplo por el de la práctica.

4.- Ensamblar el código.

Para ensamblar el código y verificar que no tiene errores, marque el archivo main01.asm en el árbol de códigos y oprima el botón derecho. Luego, seleccione la opción Assemble File, como se muestra en la figura 1.11.

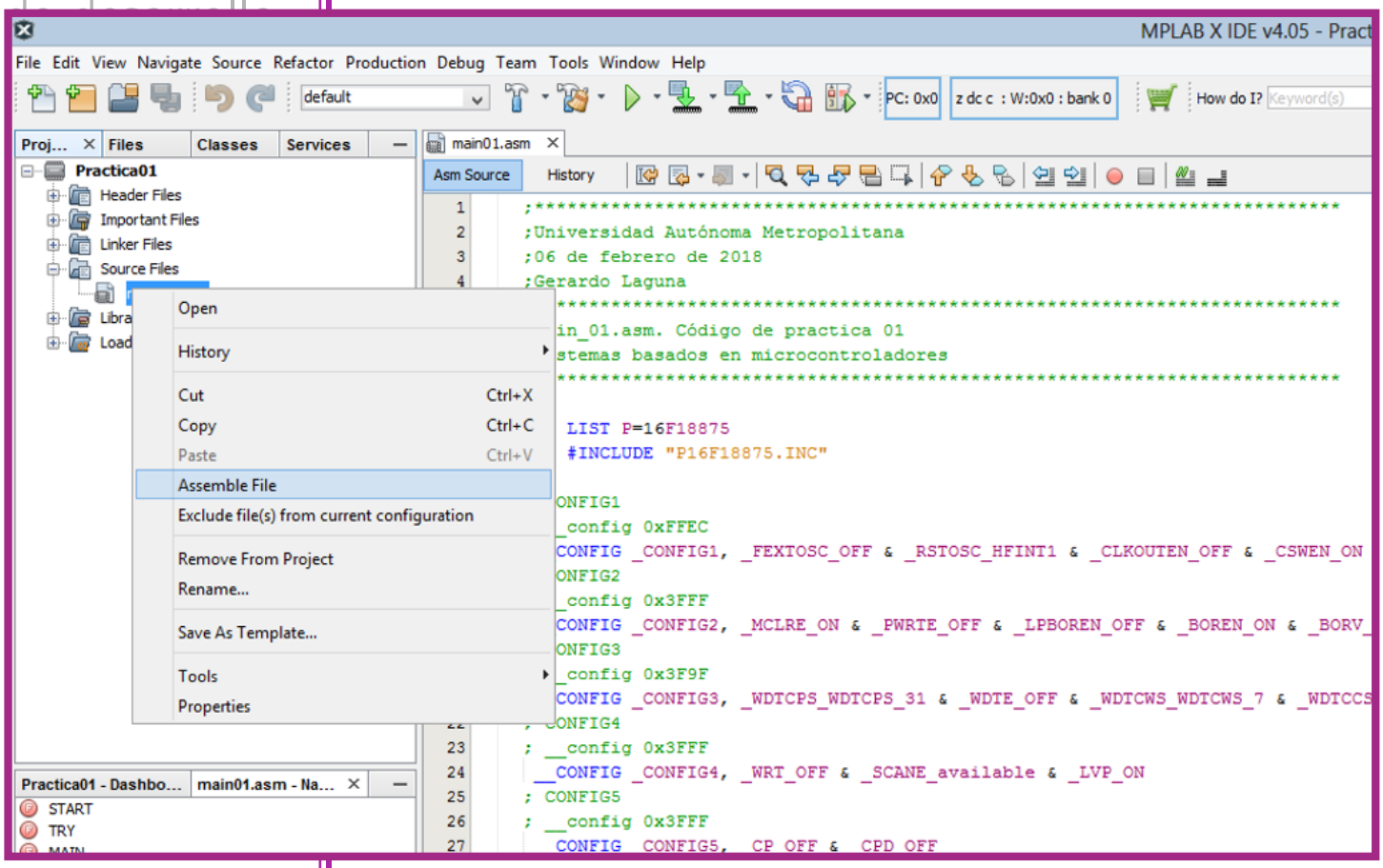


Figura 1.11. Pasos para ensamblar el código fuente.

Si no existen errores, en la ventana se salida, en la parte inferior del entorno IDE, aparece la leyenda "BUILD SUCCESSFUL", como se muestra en la figura 1.12.

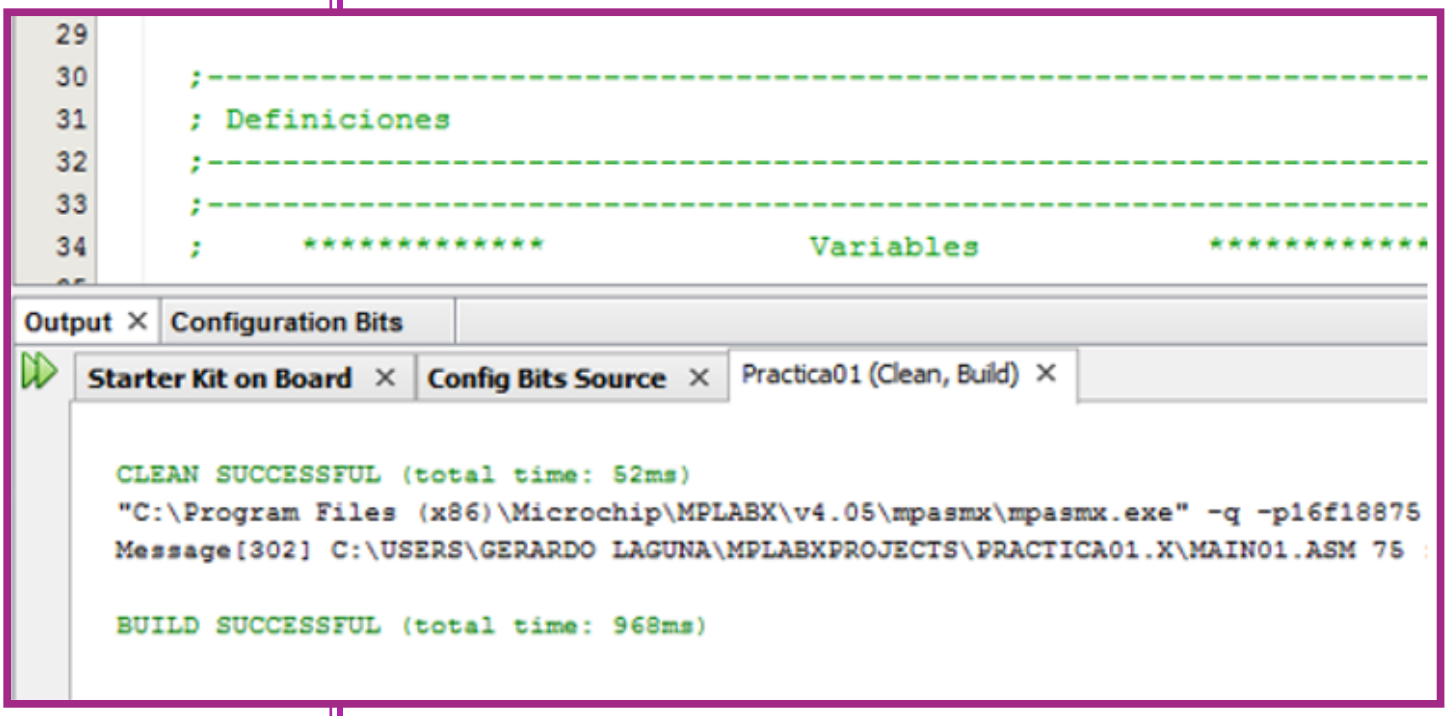


Figura 1.12. Sección de mensajes del entorno de programación.

5.- Programar el dispositivo con el código.

Asegúrese que la tarjeta está conectada a la computadora y que el LED PWR está encendido. En el árbol de códigos (ver figura 1.13), marcar el proyecto (en este caso, Practica01) y oprimir el botón derecho del ratón. Elegir la opción Make and Program Device:

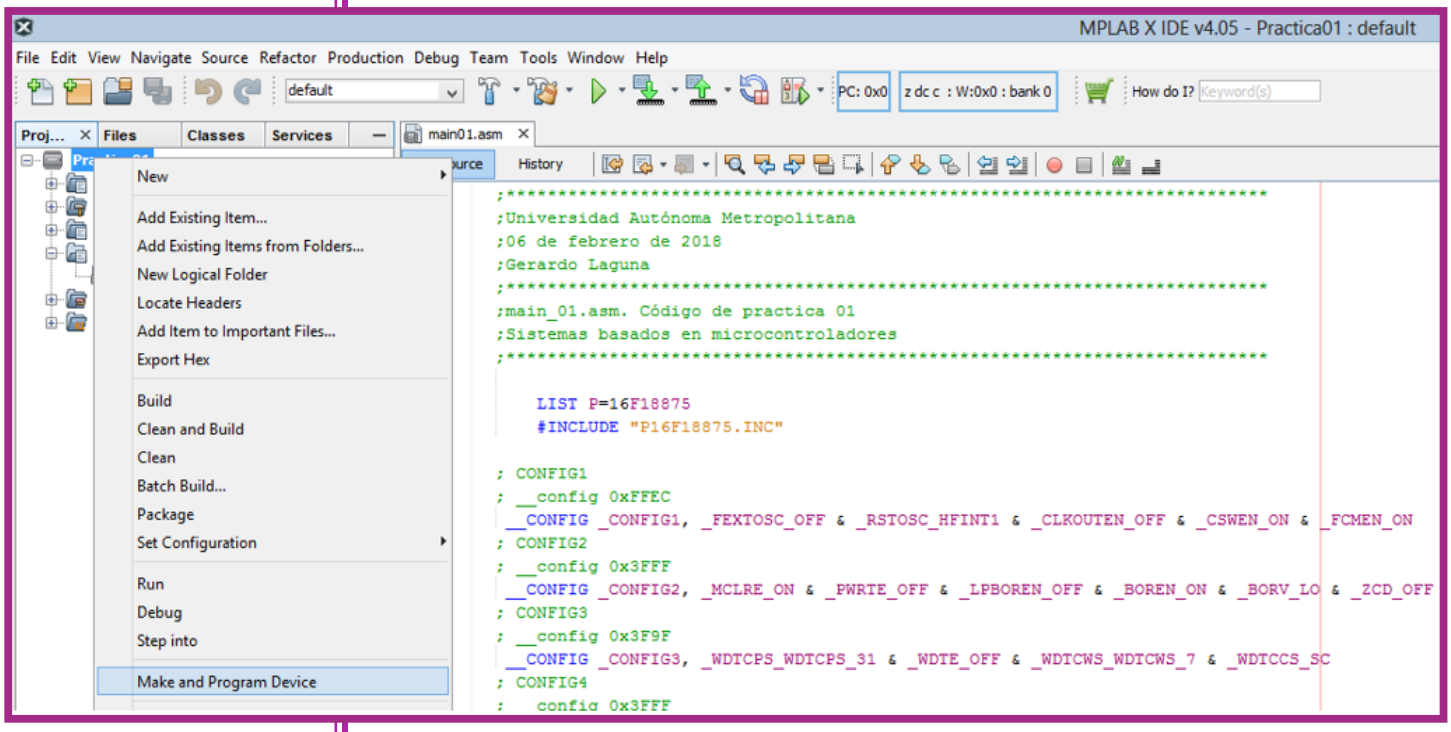


Figura 1.13. Pasos para generar y programar el código binario.

En la ventana de salida de la parte inferior se puede observar el avance del proceso. Si todo sale bien y la tarjeta se programa exitosamente, al final aparece el mensaje “Programming/Verify complete”, como se muestra en la figura 1.14.

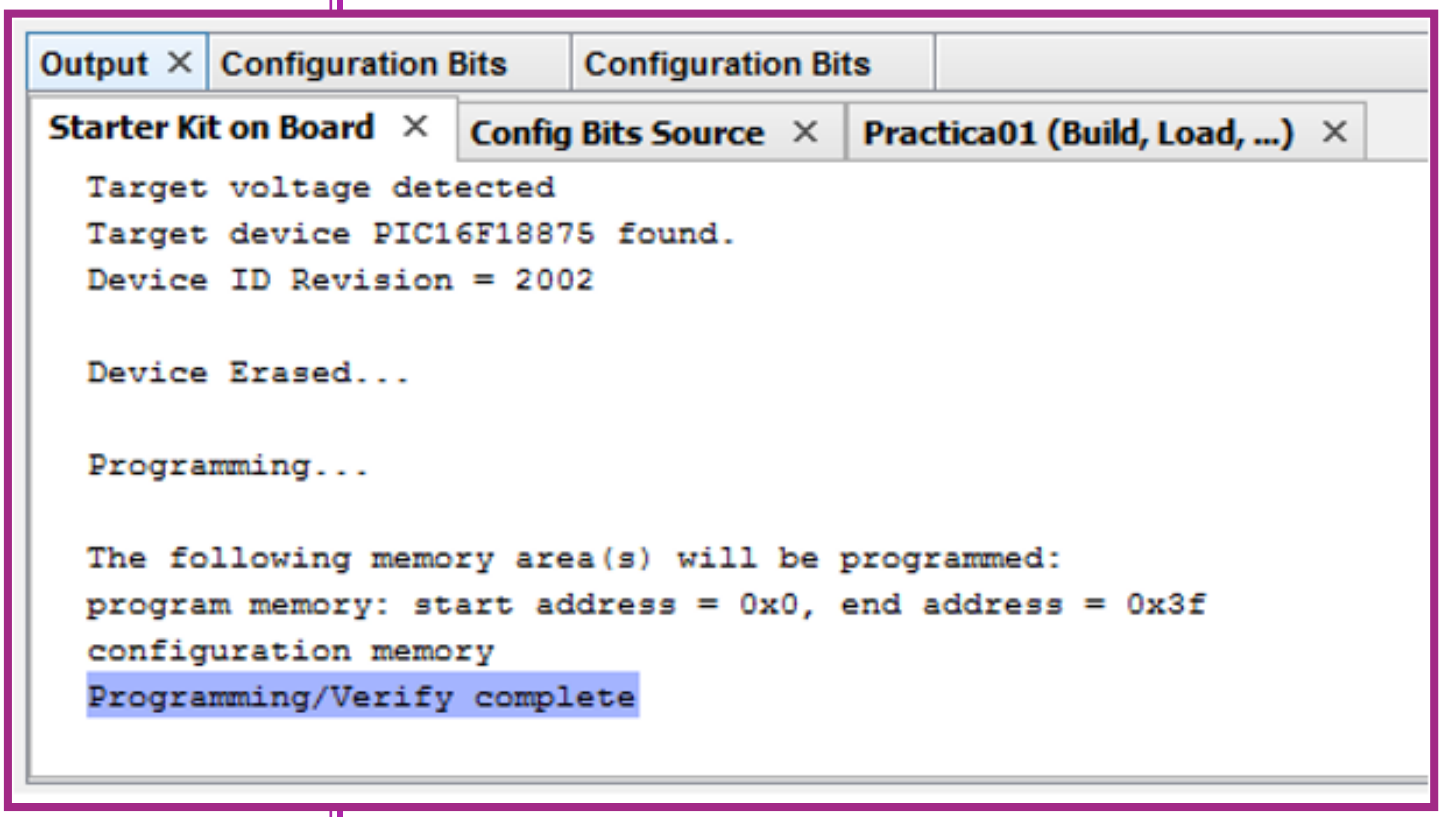


Figura 1.14. Mensajes de programación y verificación exitosa.

Si todo salió bien, el dispositivo se comportará de acuerdo al código programado, prendiendo y apagando el LED D2, como se muestra en la figura 1.15.

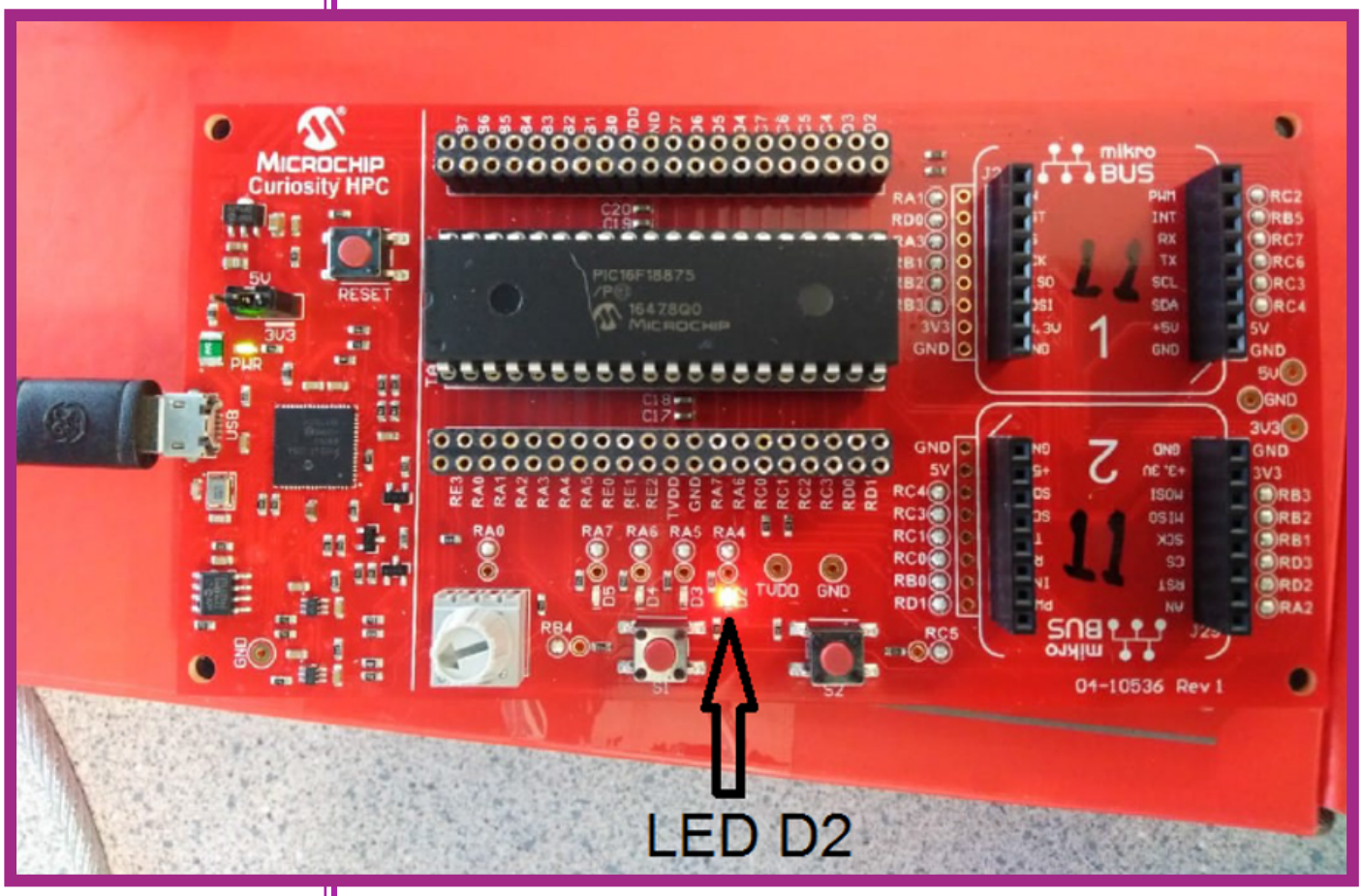


Figura 1.15. Programa corriendo en la tarjeta.

6. Actividad y reporte.

Identifique las secciones del código. Localice la sección con la configuración para las config words (CONFIG1, VONFIG2, CONFIG3, CONFIG4 y CONFIG5). En este punto es importante recordar que en la programación de sistemas se emplea el concepto de "máscara" para hacer referencia a una secuencia de unos y ceros que, mediante una operación de lógica booleana, AND u OR, permite

apagar o encender ciertos bits de una palabra. En particular, la operación AND de un bit con un cero equivale a apagarlo, mientras que la operación AND, del mismo bit, con un uno equivale a dejarlo con su valor original. Identifique la máscara _LVP_ON con la que se especifica que se programará al dispositivo con bajo voltaje. Luego, localice la sección donde se declaran las variables R0 a R3 e I0 a I3. Identifique la región de memoria donde se encuentran alojadas. Finalmente, localice la sección con el programa principal (ver figura 1.16), donde se invoca a la rutina RETARDO.

```
80  ;-------------------------------------------------------------------------
81  ; Programa principal
82  ;-------------------------------------------------------------------------
83  ;SEÑAL DE ARRANQUE:
84  ;
85                  MOVLW   D'10'   ;CUENTA PARA R3
86                  MOVWF   R3
87  TRY:
88                  MOVLW   D'1'    ;VALOR PARA I2 QUE PRODUCE UN RETARDO DE 0.1 SEG
89                  MOVWF   I2      ;SE CARGA PARAMETRO
90                  CALL    RETARDO ;I2 VECES RETARDO DE 0.1 SEG.
91                  CALL    TOGGLE  ;CONMUTAMOS LED
92                  DECFSZ  R3,F    ; ES CERO?
93                  GOTO    TRY
94  MAIN:
95                  MOVLW   D'5'    ;VALOR PARA I2 QUE PRODUCE UN RETARDO DE 0.5 SEG
96                  MOVWF   I2      ;SE CARGA PARAMETRO
97                  CALL    RETARDO ;I2 VECES RETARDO DE 0.1 SEG.
98                  CALL    TOGGLE  ;CONMUTAMOS LED
99                  GOTO    MAIN
100
```

Figura 1.16. Vista del código fuente principal.

Modifique el valor que se le asigna a la variable I2 y con ello el argumento que se pasa a la rutina RETARDO. Salve, ensamble y programe el dispositivo para observar el efecto de su cambio en el comportamiento de la tarjeta.

Particularmente, en esta primera práctica, dentro del marco teórico del reporte, haga una reseña de en las características y recursos de la tarjeta Curiosity, del dispositivo PIC16F18875 y una breve descripción del proceso de generación de ejecutables mediante códigos en lenguaje ensamblador. En el reporte incluya el código generado en un anexo.

REFERENCIAS

- **Gaonkar, R. S. (2007).** Fundamentals of Microcontrollers and Applications in Embedded Systems with PIC microcontrollers. USA: Cengage Learning.

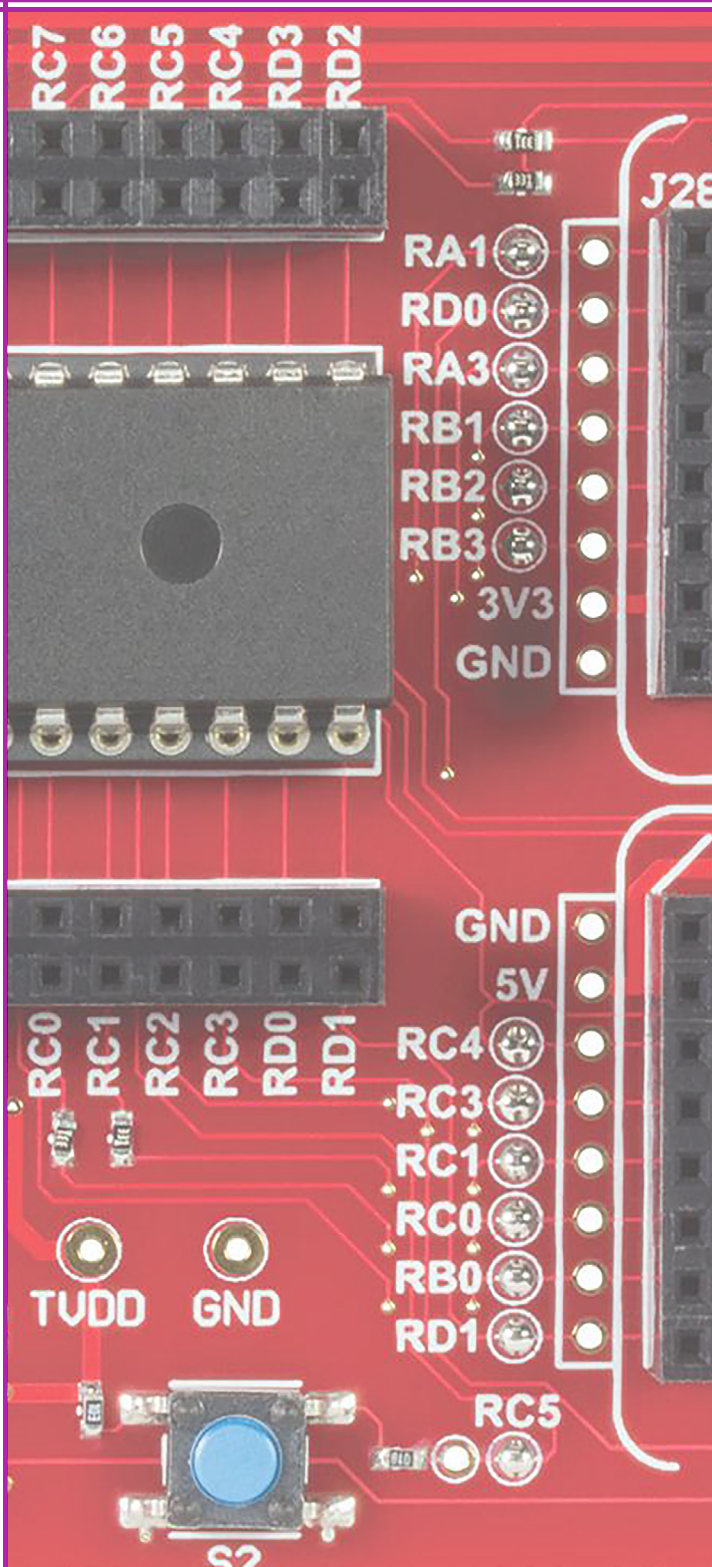
- **Microchip. (2005)**. MPASM/MPLINK PICmicro MCU Quick Chart (DS30400G). Consultado el 24 de abril de 2020, de Microchip Sitio web: http://ww1.microchip.com/downloads/en/devicedoc/30400g.pdf

- **Microchip. (2016)**. Curiosity High Pin Count (HPC) Development Board User's Guide (DS40001856A). Consultado el 24 de abril de 2020, de Microchip Sitio web: http://ww1.microchip.com/downloads/en/devicedoc/40001856a.pdf

- **Microchip. (2018).** PIC16(L)F18855/75 Data Sheet (DS40001802E). Consultado el 24 de abril 2020, de Microchip Sitio web: http://ww1.microchip.com/downloads/en/DeviceDoc/PIC16(L)F18855_75%20Data%20Sheet_DS40001802E.pdf

PRÁCTICA 2

El reloj del sistema



Dirección
Ciencias Básicas
e Ingeniería

OBJETIVOS

Identificar y manipular la configuración de la fuente de reloj para un microcontrolador PIC16F18875.

ANTECEDENTES

Los microcontroladores, al igual que la mayoría de circuitos integrados complejos, requieren una señal de reloj para funcionar. La señal de reloj suele ofrecerse por medio de un reloj generador, el cual puede ser interno o externo al microcontrolador. En las hojas técnicas y diagramas, la señal de reloj se suele denotar como CLK. La señal de reloj es una señal binaria que sirve para sincronizar las diferentes acciones de un microcontrolador. Además, esta señal se utiliza para generar la base de tiempo del microcontrolador y se caracteriza por tener una frecuencia y un ciclo de trabajo.

La frecuencia del reloj del sistema, también medida en ciclos por segundo o Hz, afecta directamente a la velocidad máxima del microcontrolador para ejecutar instrucciones por segundo, de tal forma que, a mayor frecuencia, el microcontrolador ejecutara las instrucciones más rápido. Por ejemplo, si un microcontrolador tiene un reloj con una frecuencia de 1 Hz esto significa que nunca podrá ejecutar más de una instrucción por segundo y si tiene una frecuencia de 1 MHz esto significa que, cuando mucho, podría ejecutar un millón de instrucciones por segundo.

Por otro lado, los ciclos de reloj marcan el ritmo para la realización de cada paso que involucra la ejecución de una instrucción. Así cada instrucción necesitara de uno o varios ciclos de reloj para su ejecución, el total de ciclos de reloj necesarios para ejecutar una instrucción se le llama ciclo

Práctica 2
El reloj
del sistema

de instrucción o ciclo de máquina. En las hojas técnicas se puede encontrar el tiempo necesario para la ejecución de cada una de las instrucciones y, con esta información, se puede calcular el tiempo necesario para ejecutar el total de instrucciones de un programa. Finalmente, es importante mencionar que cada microcontrolador tiene una frecuencia o velocidad máxima a la que puede funcionar y esta frecuencia o velocidad máxima también puede ser consultada en la hoja técnica del microcontrolador.

MATERIAL

- Tarjeta modelo *Curiosity HPC* de marca Microchip.
- Computadora con software MPLAB X IDE v4.05.

PROCEDIMIENTO

Antes de conectar la tarjeta *Curiosity*, asegúrese de que cuenta con el puente de alimentación, en la posición de 3.3V (3V3). Conecte la tarjeta a la computadora de mediante un cable USB A-USB Micro B y compruebe que el led PWR (verde) enciende.

En esta práctica, el dispositivo se comportará esencialmente como en la práctica anterior, conmutando intermitentemente al LED D2. Se trata de un programa que en su cuerpo contiene un lazo infinito que invoca a una subrutina de retardo y otra subrutina para conmutar al LED D2. En esta práctica vamos a conocer cómo se configura la fuente del reloj del sistema y algunas funciones del microprocesador están íntimamente relacionadas con ello.

1.- Iniciar la aplicación MPLAB IDE.

2.- Crear un nuevo proyecto.

Vaya al menú *File* y elija la opción "New Project...". Elija la opción por defecto "*Standalone Project*" de la categoría *"Microchip Embedded"*, seleccione el dispositivo PIC16F18875, verifique que aparece la herramienta de trabajo *Starter Kits (PKOB)* para la tarjeta *Curiosity*, seleccione el compilador **mpasm** y, finalmente, introduzca el nombre y la ruta del proyecto, en este caso "Practica02".

3.- Agregar el código fuente propio.

En la ventana del lado izquierdo, donde aparece el árbol de archivos del proyecto, oprima el botón derecho del ratón sobre la rama *Source Files* y elija *New* con "*pic_8b_general.asm...*". Coloque un nombre apropiado al nuevo archivo asm, por ejemplo, main02.asm. Finalmente, sustituya el código de ejemplo por el código proporcionado en el anexo 2 *(SBM_source_02.asm)*. Como ya se mencionó, este código se comportará esencialmente como en la práctica anterior, conmutando intermitentemente al LED D2.

4.- Ensamblar el código.

Para ensamblar el código y verificar que no tiene errores, marque el archivo *main02.asm* en el árbol de códigos, oprima el botón derecho y seleccione la opción *Assemble File*. Si no existen errores, en la ventana se salida, en la parte inferior del entorno IDE, aparece la leyenda *"BUILD SUCCESSFUL"*.

5.- Programar el dispositivo con el código.

Asegúrese que la tarjeta está conectada a la computadora y que el led PWR está encendido. En el árbol de códigos, marcar el proyecto (en este caso, Practica02), oprimir el botón derecho del ratón y elegir la opción *Make and Program Device*.

Si todo sale bien y la tarjeta se programa exitosamente, al final aparece el mensaje *"Programming/Verify complete"*.

6. Actividad y reporte.

Localice la sección con la configuración para las config words (ver figura 2.1) y revise los ajustes para la palabra $CONFIG1$. Identifique las máscaras que se emplean actualmente (la línea que no está comentada).

```
10          LIST P=16F18875
11          #INCLUDE "P16F18875.INC"
12
13      ; CONFIG1
14      ; __config 0x____
15       __CONFIG _CONFIG1, _FEXTOSC_OFF & _RSTOSC_HFINT1 & _CLKOUTEN_OFF & _CSWEN_ON & _FCMEN_OFF
16      ; __CONFIG _CONFIG1, _FEXTOSC_ECH & _RSTOSC_EXT1X & _CLKOUTEN_OFF & _CSWEN_ON & _FCMEN_ON
```


Figura 2.1. Sección del código fuente con los valores para los registros de configuración.

Localice el archivo P16F18875.INC (también se encuentra disponible en diversos repositorios de la Internet) y busque dentro del mismo las definiciones para cada una de las máscaras empleadas en la línea de configuración para la palabra *CONFIG1*, a saber:

_FEXTOSC_ON y _FEXTOSC_OFF
_RSTOSC_HFINT1
_CLKOUTEN_ON y _CLKOUTEN_OFF
_CSWEN_ON y _CSWEN_OFF
_FCMEN_ON y _FCMEN_OFF

Consulte el documento DS40001802E, *"PIC16(L) F18855/75 Data Sheet"*, del fabricante Microchip y determine la configuración actual para la fuente del reloj del sistema. Reporte sus hallazgos y comentarios.

Luego, comente la línea con la configuración vigente y des-comente la línea inferior. Observe las diferencias y determine la nueva configuración para la fuente de reloj del sistema. Salve los cambios, ensamble el nuevo código y programe al dispositivo para observar los efectos en el hardware. En esta ocasión, ¿Con qué reloj está operando el sistema?

¿Se trata de la misma frecuencia? Reporte sus hallazgos y comentarios.

Finalmente, remplace la máscara $\_FCMEN\_ON$ por $\_FCMEN\_OFF$. Ensamble el código y programe al dispositivo. ¿Qué efecto observa? Explique el comportamiento observado.

REFERENCIAS

- Gaonkar, R. S. (2007). Fundamentals of Microcontrollers and Applications in Embedded Systems with PIC microcontrollers. USA: Cengage Learning.

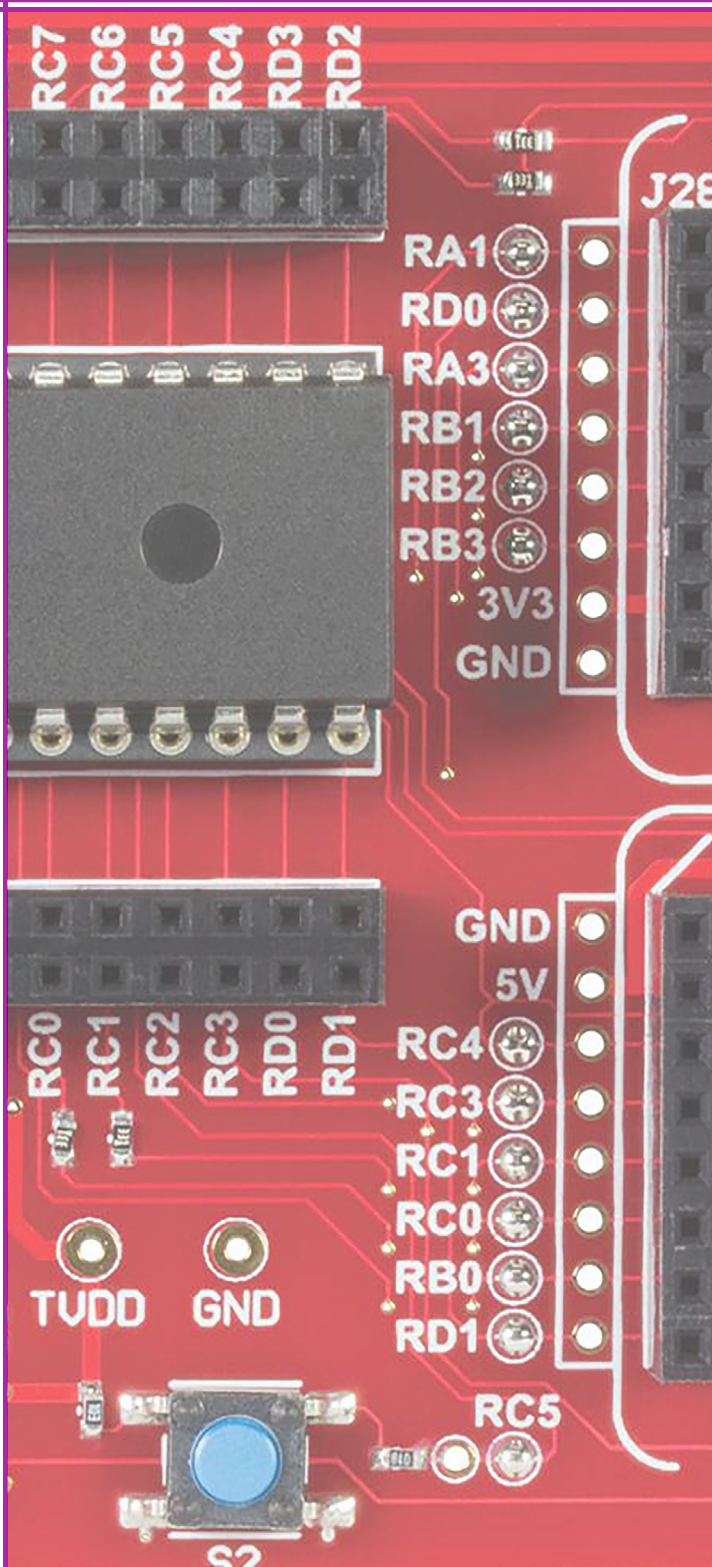
- Microchip. (2005). MPASM/MPLINK PICmicro MCU Quick Chart (DS30400G). Consultado el 24 de abril de 2020, de Microchip Sitio web: http://ww1.microchip.com/downloads/en/devicedoc/30400g.pdf

- Microchip. (2016). Curiosity High Pin Count (HPC) Development Board User's Guide (DS40001856A). Consultado el 24 de abril de 2020, de Microchip Sitio web: http://ww1.microchip.com/downloads/en/devicedoc/40001856a.pdf

- Microchip. (2018). PIC16(L)F18855/75 Data Sheet (DS40001802E). Consultado el 24 de abril 2020, de Microchip Sitio web: http://ww1.microchip.com/downloads/en/DeviceDoc/PIC16(L)F18855_75%20Data%20Sheet_DS40001802E.pdf

PRÁCTICA 3

La iniciación y el reinicio del sistema



OBJETIVOS

Identificar y manipular la señal reset del sistema, así como la configuración del sistema watchdog para un microcontrolador PIC16F18875.

ANTECEDENTES

Los microcontroladores pueden llegar a presentar comportamientos no deseados. Este tipo de falla puede deberse a diferentes causas, dentro de los más comunes se encuentran las interferencias electromagnéticas de gran intensidad, desbordamiento de la pila *(stack)* y variaciones en el voltaje de alimentación.

Este tipo de fallas provocan un comportamiento impredecible del microcontrolador y no se ejecutan de forma normal las instrucciones que tiene almacenadas. Para prevenir y solucionar este tipo de falla, los microcontroladores incorporan los recursos del *reset* y el *watchdog*. El *reset* permite poner el microcontrolador en un estado inicial conocido, sin importar el estado anterior, ya que borra e inicializa muchos de sus registros. El usuario solicita el reinicio del microcontrolador por medio de una señal conocida como *reset*, la que se denota generalmente como *MCLR*. La señal de *reset* también sirve para retardar el inicio del microcontrolador y, de esta forma, evitar que el microcontrolador empiece a funcionar antes de que la alimentación de energía se encuentre estable.

En cuanto al concepto de *watchdog*, este mecanismo se encargar de vigilar el correcto funcionamiento del microcontrolador. De forma simplificada, el sistema *watchdog* es un contador/temporizador, que cuenta permanentemente los ciclos de reloj

del sistema y que debe reiniciarse a cero antes de que alcance la cuenta máxima y se desborde. Cuando este contador/temporizador se desborda provoca el reinicio del microcontrolador. Esta es la razón por la que el programador debe evitar el desbordamiento del *watchdog* y, para ello, insertar en su programa las instrucciones que mandan a cero a su contador/temporizador. Así, si el microcontrolador llega a fallar y se entra en un estado donde la secuencia original de instrucciones se corrompe, el *watchdog* se desbordará y producirá el reinicio del microcontrolador, con lo cual se retorna a la operación normal y, por lo tanto, a su correcto funcionamiento.

MATERIAL

- Tarjeta modelo *Curiosity HPC* de marca Microchip.
- Computadora con software MPLAB X IDE v4.05.

PROCEDIMIENTO

Antes de conectar la tarjeta *Curiosity*, asegúrese de que cuenta con el puente de alimentación, en la posición de 3.3V (3V3). Conecte la tarjeta a la computadora de mediante un cable USB A-USB Micro B y compruebe que el led PWR (verde) enciende.

En esta práctica, el dispositivo se comportará esencialmente como en la primera práctica, conmutando al LED D2 de forma intermitente, pero vamos a aprender cómo se configura la señal *reset* del sistema y algunas funciones del microprocesador están íntimamente relacionadas con el reinicio del sistema.

1.- Iniciar la aplicación MPLAB IDE.

2.- Crear un nuevo proyecto.

Vaya al menú *File* y elija la opción *"New Project..."*. Elija la opción por defecto *"Standalone Project"* de la categoría *"Microchip Embedded"*, seleccione el dispositivo PIC16F18875, verifique que aparece la herramienta de trabajo *Starter Kits (PKOB)* para la tarjeta *Curiosity,* seleccione el compilador **mpasm** y, finalmente, introduzca el nombre y la ruta del proyecto, en este caso "Practica03".

3.- Agregar el código fuente propio.

En la ventana del lado izquierdo, donde aparece el árbol de archivos del proyecto, oprima el botón derecho del ratón sobre la rama *Source Files* y elija *New* con

"pic_8b_general.asm...". Coloque un nombre apropiado al nuevo archivo asm, por ejemplo, *main03.asm*. Finalmente, sustituya el código de ejemplo por el código proporcionado en el anexo 3 *(SBM_source_03.asm)*.

4.- Ensamblar el código.

Para ensamblar el código y verificar que no tiene errores, marque el archivo *main03.asm* en el árbol de códigos, oprima el botón derecho y seleccione la opción *Assemble File*. Si no existen errores, en la ventana se salida, en la parte inferior del entorno IDE, aparece la leyenda *"BUILD SUCCESSFUL"*.

5.- Programar el dispositivo con el código.

Asegúrese que la tarjeta está conectada a la computadora y que el led PWR está encendido. En el árbol de códigos, marcar el proyecto (en este caso, Practica03), oprimir el botón derecho del ratón y elegir la opción *Make and Program Device.*

Si todo sale bien y la tarjeta se programa exitosamente, al final aparece el mensaje *"Programming/Verify complete"*.

6. Actividad y reporte.

En la tarjeta Curiosity localice el botón de RESET. Oprímalo mientras se ejecuta el programa normalmente. ¿Qué ocurre? Reporte sus observaciones y comentarios. Localice, en la sección de configuración (ver figura 3.1), los ajustes para la palabra $CONFIG2$. Identifique la máscara $\_MCLRE\_ON$ que se emplea actualmente para habilitar la terminal $MCLR$ que funciona como *reset*.

```
16      ; CONFIG2
17      ; __config 0x3FFF
18       __CONFIG _CONFIG2, _MCLRE_ON & _PWRTE_OFF & _LPBOREN_OFF & _BOREN_ON & _BORV_LO & _ZCD_OFF & _PPS1WAY_ON & _STVREN_ON
19      ; CONFIG3
20      ; __config 0x3F9F
21       __CONFIG _CONFIG3, _WDTCPS_WDTCPS_31 & _WDTE_OFF & _WDTCWS_WDTCWS_7 & _WDTCCS_SC
22      ;__CONFIG _CONFIG3, _WDTCPS_WDTCPS_31 & _WDTE_ON & _WDTCWS_WDTCWS_7 & _WDTCCS_SC
```


Figura 3.1. Sección del código fuente con los valores para los registros de configuración.

Aunque en teoría es posible deshabilitar la terminal $MCLR$ mediante la máscara $\_MCLRE\_OFF$, esto no será posible por ahora dado que estamos empleando el modo de programación en bajo voltaje y en este modo de operación la terminal $MCLR$ siempre funciona como *reset*. Para más detalles, ver lo relativo a la palabra $CONFIG2$ en el documento DS40001802E, "PIC16(L)F18855/75 Data Sheet", de Microchip

Ahora, localice y revise los ajustes para la palabra $CONFIG3$. Identifique la máscara $\_WDTE\_OFF$ que se emplea actualmente (la línea que no está comentada) para mantener deshabilitada la función de *watchdog timer*. Comente la línea con la configuración vigente y des-comente la línea inferior. La nueva línea emplea la máscara $\_WDTE\_ON$ para habilitar al *watchdog timer*. Salve los cambios, ensamble y programe el dispositivo. ¿Qué fenómeno se presenta? Reporte sus observaciones y comentarios.

Luego, localice la sección con el lazo del código principal (ver figura 3.2) e identifique la línea comentada con la instrucción $CLRWDT$ que sirve para limpiar el temporizador del *watchdog*.

```
81  ;-------------------------------------------------------------------------
82  ; Programa principal
83  ;-------------------------------------------------------------------------
84  ;SEÑAL DE ARRANQUE:
85  ;
86              MOVLW   D'10'   ;CUENTA PARA R3
87              MOVWF   R3
88  TRY:
89              MOVLW   D'1'    ;VALOR PARA I2 QUE PRODUCE UN RETARDO DE 0.1 SEG
90              MOVWF   I2      ;SE CARGA PARAMETRO
91              CALL    RETARDO ;I2 VECES RETARDO DE 0.1 SEG.
92              CALL    TOGGLE  ;CONMUTAMOS LED
93              DECFSZ  R3,F    ; ES CERO?
94              GOTO    TRY
95  MAIN:
96  ;           CLRWDT          :LIMPIA WATCHDOG TIMER
97              MOVLW   D'5'    ;VALOR PARA I2 QUE PRODUCE UN RETARDO DE 0.5 SEG
98              MOVWF   I2      ;SE CARGA PARAMETRO
99              CALL    RETARDO ;I2 VECES RETARDO DE 0.1 SEG.
100             CALL    TOGGLE  ;CONMUTAMOS LED
101             GOTO    MAIN
102
```

Figura 3.2. Sección del código fuente con el lazo principal del programa.

Des-comente la línea con la instrucción *CLRWDT*, ensamble el código y programe al dispositivo. ¿Cómo se comporta ahora el dispositivo? Explique el fenómeno observado. Consulte el documento DS40001802E, *"PIC16(L)F18855/75 Data Sheet"*, de Microchip y reseñe brevemente, en el marco teórico conceptual de su reporte, las funciones de *reset* y de *watchdog* así como las opciones relacionadas.

REFERENCIAS

- **Gaonkar, R. S. (2007)**. Fundamentals of Microcontrollers and Applications in Embedded Systems with PIC microcontrollers. USA: Cengage Learning.

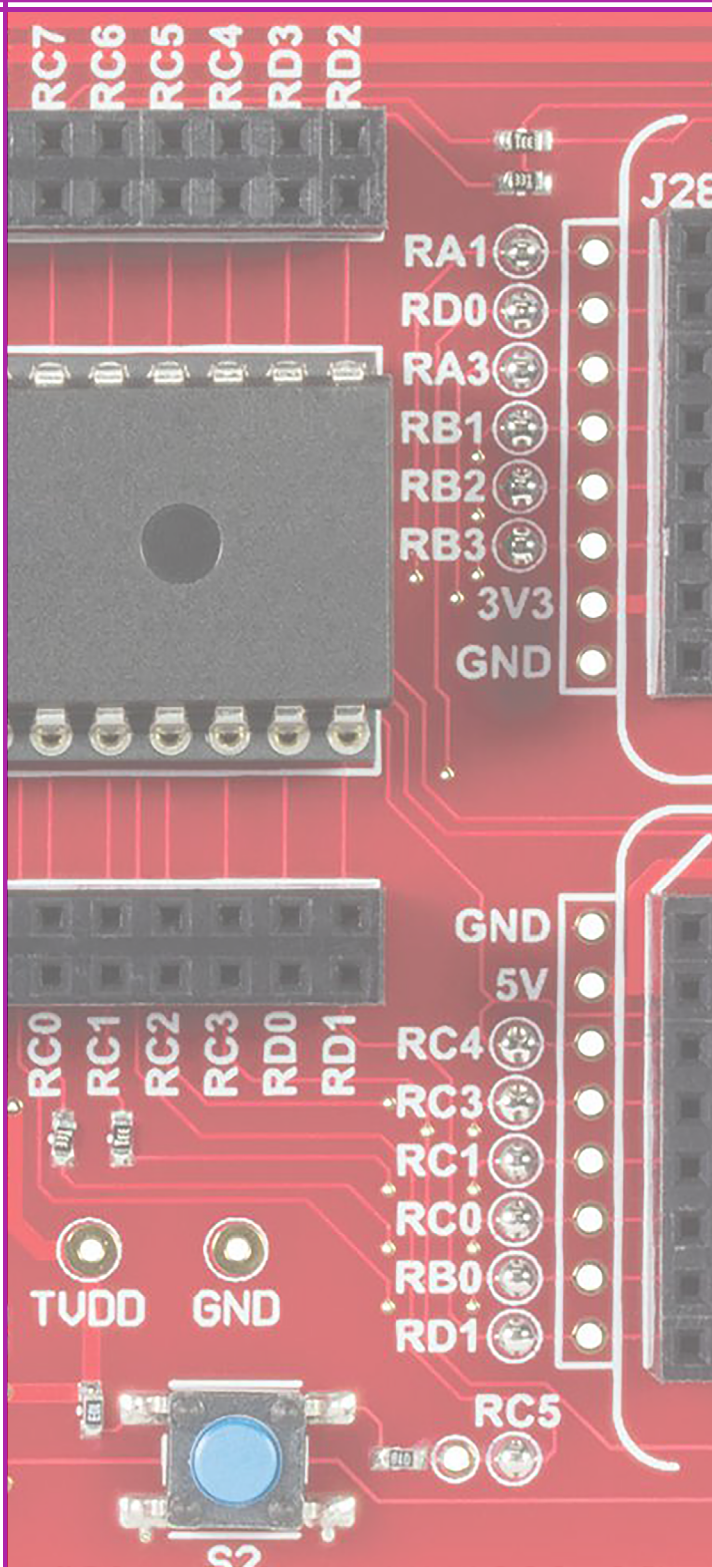
- **Microchip. (2005).** MPASM/MPLINK PICmicro MCU Quick Chart (DS30400G). Consultado el 24 de abril de 2020, de Microchip Sitio web: http://ww1.microchip.com/downloads/en/devicedoc/30400g.pdf

- **Microchip. (2016).** Curiosity High Pin Count (HPC) Development Board User's Guide (DS40001856A). Consultado el 24 de abril de 2020, de Microchip Sitio web: http://ww1.microchip.com/downloads/en/devicedoc/40001856a.pdf

- **Microchip. (2018).** PIC16(L)F18855/75 Data Sheet (DS40001802E). Consultado el 24 de abril 2020, de Microchip Sitio web: http://ww1.microchip.com/downloads/en/DeviceDoc/PIC16(L)F18855_75%20Data%20Sheet_DS40001802E.pdf

PRÁCTICA 4

El control de flujo de los programas



OBJETIVOS

Reconocer y aplicar instrucciones para control de flujo y asociarlas a la configuración y estado digital de un puerto de entrada en un microcontrolador.

ANTECEDENTES

Una de las características que hace muy atractivos y útiles a los microcontroladores son sus puertos de E/S, los cuales permiten a la unidad central de procesamiento comunicarse con dispositivos externos. De esta forma, el microcontrolador puede recibir/enviar información desde/hacia el exterior y comunicarse con otros dispositivos.

Los puertos del microcontrolador pueden estar disponibles hacia el exterior mediante un subconjunto de las terminales del microcontrolador que son, en muchos de los casos, completamente configurables. Aunque la mayoría de las terminales del microcontrolador pueden ser configuradas como entradas o como salidas, nunca podrán ser configurados como ambos al mismo tiempo.

La configuración de algunas terminales del microcontrolador, como entradas o como salidas, dependerá de las necesidades del proyecto. Por ejemplo, las terminales que sean configurados como entradas digitales podrán interpretar el valor de voltaje que reciban como un valor lógico 0 o 1. De forma equivalente, las terminales que sean configurados como salidas digitales, podrán generar los niveles de voltaje que corresponda a los valores lógicos 0 y 1. Es importante saber que existe un intervalo de voltajes que son aceptados por el microcontrolador como 0 o 1 y que estos intervalos de voltaje son determinados por cada fabricante, de tal suerte que pueden

Práctica 4
El control de flujo de los programas

ser consultados en la hoja técnica del microcontrolador en cuestión.
Las entradas y salidas, tanto las digitales como las analógicas, se pueden emplear para interactuar, según su naturaleza, con elementos externos tales como leds, sensores, circuitos integrados, etc.

MATERIALES

- Tarjeta modelo *Curiosity HPC* de marca Microchip.
- Computadora con software MPLAB X IDE v4.05.

PROCEDIMIENTO

Antes de conectar la tarjeta *Curiosity*, asegúrese de que cuenta con el puente de alimentación, en la posición de 3.3V (3V3). Conecte la tarjeta a la computadora de mediante un cable USB A-USB Micro B y compruebe que el led PWR (verde) enciende.

En esta práctica, el dispositivo se comportará esencialmente como en la primera práctica, conmutando intermitentemente al LED D2, pero en esta práctica vamos a aprender cómo se pueden tomar decisiones y alterar el curso del programa con base en el estado de un puerto de entrada.

1.- Iniciar la aplicación MPLAB IDE.

2.- Crear un nuevo proyecto.

Vaya al menú *File* y elija la opción *"New Project..."*. Elija la opción por defecto *"Standalone Project"* de la categoría *"Microchip Embedded"*, seleccione el dispositivo PIC16F18875, verifique que aparece la herramienta de trabajo *Starter Kits (PKOB)* para la tarjeta *Curiosity*, seleccione el compilador **mpasm** y, finalmente, introduzca el nombre y la ruta del proyecto, en este caso "Practica04".

3.- Agregar el código fuente propio.

En la ventana del lado izquierdo, donde aparece el árbol de archivos del proyecto, oprima el botón derecho del ratón sobre la rama *Source Files* y elija *New* con *"pic_8b_general.asm..."*. Coloque un nombre apropiado al nuevo archivo asm, por ejemplo, *main04.asm*. Finalmente, sustituya el código de ejemplo por el código proporcionado en el anexo 4 *(SBM_source_04.asm)*.

4.- Ensamblar el código.

Para ensamblar el código y verificar que no tiene errores, marque el archivo *main04.asm* en el árbol de códigos, oprima el botón derecho y seleccione la opción Assemble File. Si no existen errores, en la ventana se salida, en la parte inferior del entorno IDE, aparece la leyenda *"BUILD SUCCESSFUL"*.

5.- Programar el dispositivo con el código.

Asegúrese que la tarjeta está conectada a la computadora y que el led PWR está encendido. En el árbol de códigos, marcar el proyecto (en este caso, Practica04), oprimir el botón derecho del ratón y elegir la opción *Make and Program Device.*

Si todo sale bien y la tarjeta se programa exitosamente, al final aparece el mensaje *"Programming/Verify complete".*

6. Actividad y reporte.

Estudie el código, particularmente la etapa de configuración de puertos así como el programa principal (ver la figura 4.1), e infiera el funcionamiento del mismo. Revise el diagrama esquemático de la tarjeta Curiosity y determine cuáles son los componentes, las señales y las terminales involucradas. Dibuje un diagrama de flujo para expresar la configuración y el funcionamiento del programa.

```
;-------------------------------------------------------------------------
; Programa principal
;-------------------------------------------------------------------------
;SEÑAL DE ARRANQUE:
;
                MOVLW   D'10'   ;CUENTA PARA R3
                MOVWF   R3
TRY:
                MOVLW   D'1'    ;VALOR PARA I2 QUE PRODUCE UN RETARDO DE 0.1 SEG
                MOVWF   I2      ;SE CARGA PARAMETRO
                CALL    RETARDO ;I2 VECES RETARDO DE 0.1 SEG.
                CALL    TOGGLE  ;CONMUTAMOS LED
                DECFSZ  R3,F    ; ES CERO?
                GOTO    TRY
;Rápido o lento?
MAIN:
                BTFSS   PORTB,RB4   ;Está oprimido SW1 (lógica negativa)?
                GOTO    SPEEDY      ;Si SW1 está oprimido: LED parpadea rápido
                MOVLW   D'5'    ;VALOR PARA I2 QUE PRODUCE UN RETARDO DE 0.5 SEG
                MOVWF   I2      ;SE CARGA PARAMETRO
                GOTO    SUBR
SPEEDY:
                MOVLW   D'1'    ;VALOR PARA I2 QUE PRODUCE UN RETARDO DE 0.1 SEG
                MOVWF   I2      ;SE CARGA PARAMETRO
SUBR:
                CALL    RETARDO ;I2 VECES RETARDO DE 0.1 SEG.
                CALL    TOGGLE  ;CONMUTAMOS LED
                GOTO    MAIN

;-------------------------------------------------------------------------
```

Figura 4.1. Sección del código fuente con el lazo del programa principal.

Modifique el código para emplear el botón S2, en vez del botón S1, a fin de modificar la velocidad de parpadeo del LED. Para más detalles sobre la configuración de los puertos digitales de entrada/salida, ver lo conducente en el documento DS40001802E, *"PIC16(L)F18855/75 Data Sheet"*, de Microchip. Incluya el código resultante como anexo en su reporte.

REFERENCIAS

- Gaonkar, R. S. (2007). Fundamentals of Microcontrollers and Applications in Embedded Systems with PIC microcontrollers. USA: Cengage Learning.

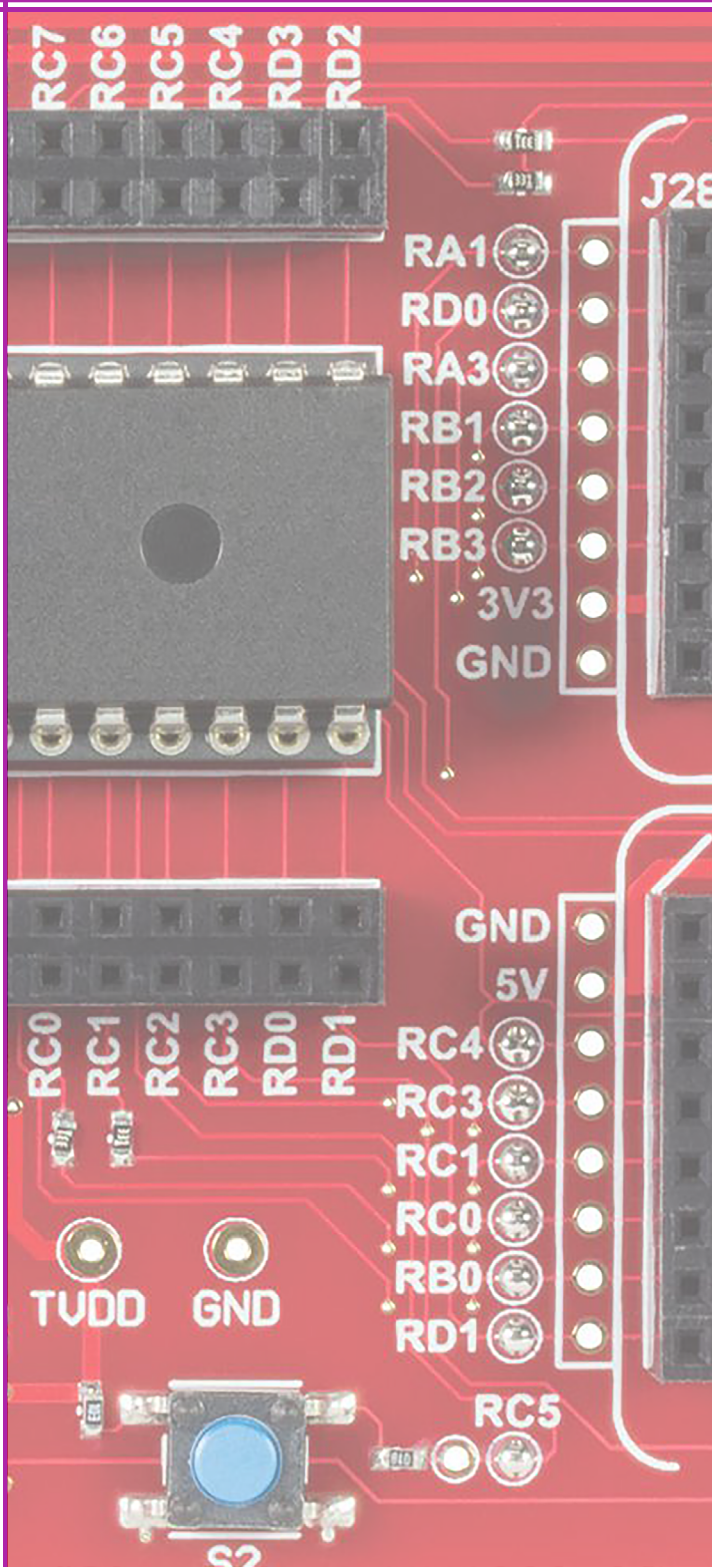
- Microchip. (2005). MPASM/MPLINK PICmicro MCU Quick Chart (DS30400G). Consultado el 24 de abril de 2020, de Microchip Sitio web: http://ww1.microchip.com/downloads/en/devicedoc/30400g.pdf

- Microchip. (2016). Curiosity High Pin Count (HPC) Development Board User's Guide (DS40001856A). Consultado el 24 de abril de 2020, de Microchip Sitio web: http://ww1.microchip.com/downloads/en/devicedoc/40001856a.pdf

- Microchip. (2018). PIC16(L)F18855/75 Data Sheet (DS40001802E). Consultado el 24 de abril 2020, de Microchip Sitio web: http://ww1.microchip.com/downloads/en/DeviceDoc/PIC16(L)F18855_75%20Data%20Sheet_DS40001802E.pdf

PRÁCTICA 5

El procesamiento de las interrupciones del sistema



Dirección
Ciencias Básicas
e Ingeniería

OBJETIVOS

Aplicar los conceptos de interrupción y de rutina de atención a interrupción, así como el procedimiento de configuración y escritura de código para las mismas en un microcontrolador.

ANTECEDENTES

Debido a la versatilidad de los microcontroladores, estos pueden interactuar con diferentes dispositivos. Algunos de estos dispositivos requieren de una atención inmediata como, por ejemplo, la detección de señales externas, la recepción y transmisión de datos, la activación de un temporizador, etc. En ese sentido, los microcontroladores cuentan con un mecanismo rápido y eficaz que permite el tratamiento de estos eventos. Este mecanismo se conoce como atención de *interrupciones*.

Como ya se dijo, las *interrupciones* permiten reaccionar a eventos externos al microcontrolador de forma inmediata. La atención de las interrupciones implica, por un lado, ejecutar códigos cortos para el procesamiento requerido y, por otro lado, detener momentáneamente la ejecución del código principal.

Algunos microprocesadores permiten que una interrupción se genere a partir de un evento en hardware o uno por software. En el caso particular del microcontrolador que estamos usando, únicamente se generan interrupciones por hardware. Cuando alguno de estos eventos sucede la ejecución del microcontrolador se suspende, se guarda el estado de algunos registros del sistema y se produce un salto hacia la *rutina de atención a la interrupción*. Dicha rutina realiza el procesamiento requerido y, una vez terminado, el microcontrolador retoma el programa principal que fue interrumpido, restableciendo el estado original de los registros que fueron salvados al entrar en la interrupción.

MATERIAL

- Tarjeta modelo *Curiosity HPC* de marca Microchip.
- Computadora con software MPLAB X IDE v4.05.

PROCEDIMIENTO

Antes de conectar la tarjeta Curiosity, asegúrese de que cuenta con el puente de alimentación, en la posición de 3.3V (3V3). Conecte la tarjeta a la computadora de mediante un cable USB A-USB Micro B y compruebe que el led PWR (verde) enciende.

En esta práctica, el dispositivo se comportará esencialmente como en la primera práctica, conmutando al LED D2 en forma intermitente, pero en esta práctica vamos a aplicar el concepto de interrupción a fin de que nuestro programa responda a eventos en el hardware.

1.- Iniciar la aplicación MPLAB IDE.

2.- Crear un nuevo proyecto.

Vaya al menú File y elija la opción "New Project...". Elija la opción por defecto "Standalone Project" de la categoría "Microchip Embedded", seleccione el dispositivo PIC16F18875, verifique que aparece la herramienta de trabajo Starter Kits (PKOB) para la tarjeta Curiosity, seleccione el compilador mpasm y, finalmente, introduzca el nombre y la ruta del proyecto, en este caso "Practica05".

3.- Agregar el código fuente propio.

En la ventana del lado izquierdo, donde aparece el árbol de archivos del proyecto, oprima el botón derecho del ratón sobre la rama Source Files y elija New con "pic_8b_general.asm...". Coloque un

nombre apropiado al nuevo archivo asm, por ejemplo, main05.asm. Finalmente, sustituya el código de ejemplo por el código proporcionado en el anexo 5 (SBM_source_05.asm).

4.- Ensamblar el código.

Para ensamblar el código y verificar que no tiene errores, marque el archivo main05.asm en el árbol de códigos, oprima el botón derecho y seleccione la opción Assemble File. Si no existen errores, en la ventana se salida, en la parte inferior del entorno IDE, aparece la leyenda "BUILD SUCCESSFUL".

5.- Programar el dispositivo con el código.

Asegúrese que la tarjeta está conectada a la computadora y que el led PWR está encendido. En el árbol de códigos, marcar el proyecto (en este caso, Practica05), oprimir el botón derecho del ratón y elegir la opción Make and Program Device. Si todo sale bien y la tarjeta se programa exitosamente, al final aparece el mensaje "Programming/Verify complete".

6. Actividad y reporte.

Estudie el código, particularmente la etapa de configuración del sistema, el programa principal y la rutina de atención a interrupciones (ver la figura 5.1). Explique cada una de las líneas de código de la sección donde se inicializan las interrupciones:

```
;SE INICIALIZAN INTERRUPCIONES:
        BANKSEL INTPPS
        MOVLW     0x0C  ;ELIGE RB4 COMO FUENTE PARA
                          INTERRUPCION EXTERNA
        MOVWF     INTPPS
        BANKSEL PIE0
        BSF   PIE0,INTE   ;HABILITA INTERRUPCION EXTERNA
        BCF   INTCON,INTEDG  ;ACTIVACION POR FLANCO DE
                             BAJADA
        BSF   INTCON,GIE  ;HABILITA INTERRUPCIONES
```

Revise el diagrama esquemático de la tarjeta *Curiosity* y determine cuáles son los componentes, las señales y las terminales involucradas. Infiera el funcionamiento del mismo y corrobore su hipótesis al oprimir el botón S1 y observar el efecto producido.

```
193     ;-------------------------------------------------------------------------
194     ;-------------------------------------------------------------------------
195     ; RUTINAS DE ATENCION A INTERRUPCIONES
196     ;-------------------------------------------------------------------------
197     ;
198     ;-------------------------------------------------------------------------
199     ;PROFUNDIDAD DE STACK=1
200     ;-------------------------------------------------------------------------
201     ;RUTINA INTSERV() SERVICIO A INTERRUPCION
202     ;
203     INTSERV:
204                     BANKSEL PIR0
205                     BTFSS   PIR0,INTF   ;ES INTERUPCION EXTERNA?
206                     GOTO    EINTSR      ;SI NO ES INTERUPCION EXTERNA, SALIR
207                     BCF     PIR0,INTF   ;LIMPIA FUENTE DE INTERRUPCION
208     ;CONMUTA BANDERA:
209                     BTFSS   FLAGS,SPEEDYF
210                     GOTO    SETFLAG
211     ;LIMPIA BANDERA:
212                     BCF     FLAGS,SPEEDYF
213                     GOTO    EINTSR
214     ;PRENDE BANDERA:
215     SETFLAG:
216                     BSF     FLAGS,SPEEDYF
217     EINTSR:
218
219                     RETFIE
220     ;FIN RUTINA INTSERV
221     ;
```

Figura 5.1. Sección del código fuente con la rutina de atención a interrupciones.

Modifique el código, realizando y probando un cambio a la vez, para corroborar los efectos de deshabilitar el bit *GIE*, la habilitación del bit *INTEDG* y la omisión del reinicio de la bandera *INTF*. Investigue la función de cada uno de estos bits y explique el comportamiento observado después de cada modificación. Para más detalles sobre la configuración y operación de la interrupción externa (INT), ver lo conducente en el documento DS40001802E, *"PIC16(L)F18855/75 Data Sheet"*, de Microchip.

REFERENCIAS

- **Gaonkar, R. S. (2007).** Fundamentals of Microcontrollers and Applications in Embedded Systems with PIC microcontrollers. USA: Cengage Learning.

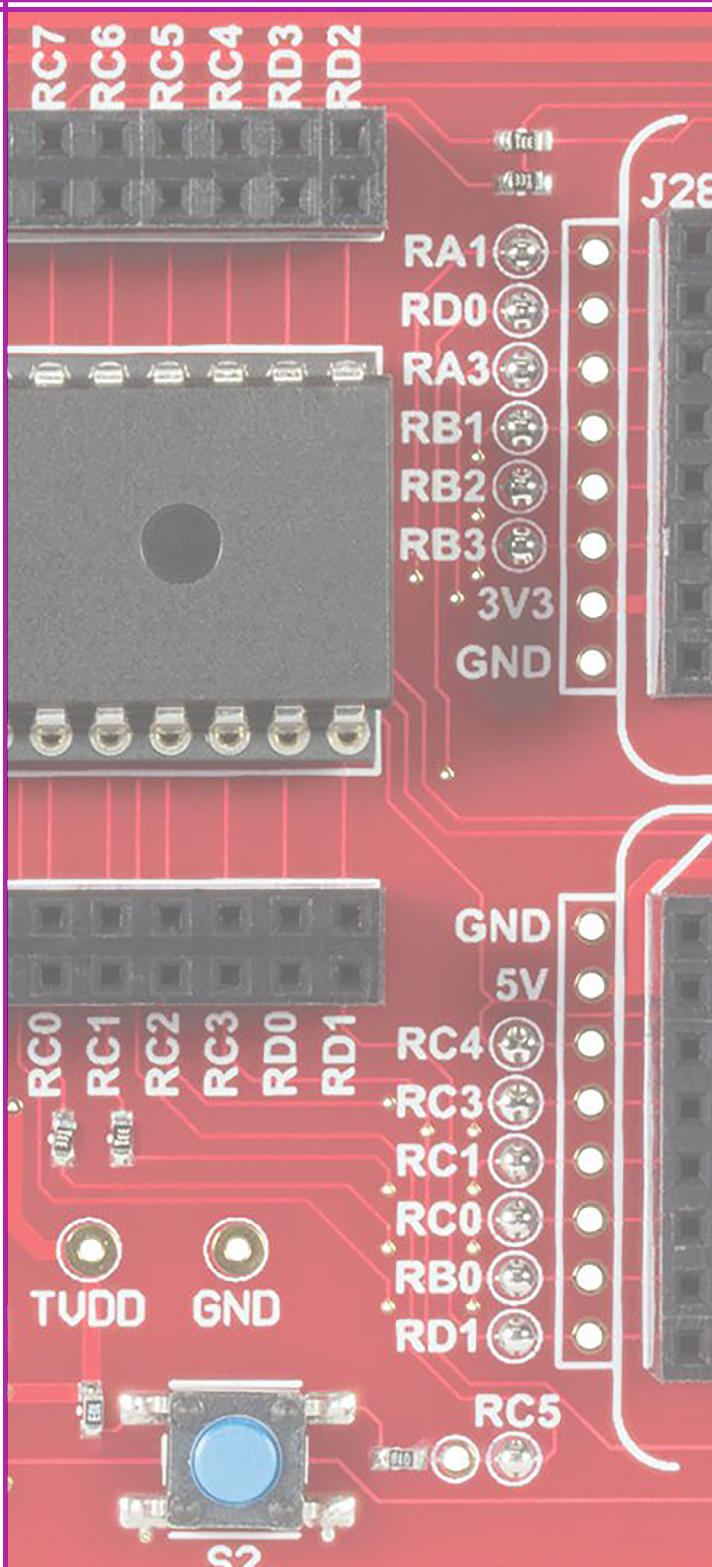
- **Microchip. (2005).** MPASM/MPLINK PICmicro MCU Quick Chart (DS30400G). Consultado el 24 de abril de 2020, de Microchip Sitio web: http://ww1.microchip.com/downloads/en/devicedoc/30400g.pdf

- **Microchip. (2016).** Curiosity High Pin Count (HPC) Development Board User's Guide (DS40001856A). Consultado el 24 de abril de 2020, de Microchip Sitio web: http://ww1.microchip.com/downloads/en/devicedoc/40001856a.pdf

- **Microchip. (2018).** PIC16(L)F18855/75 Data Sheet (DS40001802E). Consultado el 24 de abril 2020, de Microchip Sitio web: http://ww1.microchip.com/downloads/en/DeviceDoc/PIC16(L)F18855_75%20Data%20Sheet_DS40001802E.pdf

PRÁCTICA 6

El empleo de los temporizadores



OBJETIVOS

Aplicar el concepto de temporizador, así como el procedimiento de configuración y escritura de código para el uso del mismo en un microcontrolador.

ANTECEDENTES

Dos de las funciones más útiles al realizar proyectos con microcontroladores son la contabilización de eventos y la generación de marcas de tiempo. Para este tipo de tareas, los microcontroladores cuentan con circuitos especializados para dicho fin. Estos circuitos son los contadores y temporizadores. A veces, las funciones de contador y temporizador se realizan por el mismo periférico.

En particular, a los temporizadores se les puede encontrar en la hoja técnica del microcontrolador con las siglas TMR. El número de temporizadores con lo que cuenta un microcontrolador dependen del modelo de este. Los temporizadores son esencialmente un registro contador. Dicho registro aumenta su valor en una unidad, cada ciertos ciclos del reloj del sistema y puede indicar, mediante una bandera o una interrupción, cuando ha alcanzado un valor específico definido por el usuario.

En principio y conceptualmente, los temporizadores pueden funcionar de dos formas: contado los ciclos de un reloj externo o contando los ciclos de un reloj interno. En el caso de contador de ciclos externos, el temporizador se incrementará cada que se detecta cierto flanco (de subida o de bajada) de una señal externa. Ya sea que la cuenta del contador/temporizador se incrementa por un reloj interno o externo, en ambos casos el mecanismo se puede emplea para la generación de señales o

interrupciones periódicas. El tiempo de expiración del temporizador se puede configurar modificando la el valor de desbordamiento o el valor inicial de la cuenta y ello depende de las características del microcontrolador empleado.
Otra característica de los temporizadores, es que pueden ser de diferente número de bits, por ejemplo de 8 o 16 bits. Así, un temporizador de 8 bits tendrá un rango de valores de 0 a 255, mientras que un temporizador de 16 bits tendrá un rango de valores de 0 a 65535. En la hoja técnica del microcontrolador se indica el tipo y el número de temporizadores disponibles.

MATERIAL

- Tarjeta modelo *Curiosity HPC* de marca Microchip.
- Computadora con software MPLAB X IDE v4.05.

PROCEDIMIENTO

Antes de conectar la tarjeta *Curiosity*, asegúrese de que cuenta con el puente de alimentación, en la posición de 3.3V (3V3). Conecte la tarjeta a la computadora de mediante un cable USB A-USB Micro B y compruebe que el led PWR (verde) enciende.

En esta práctica, el dispositivo se comportará esencialmente como en la primera práctica. También conmuta al LED D2, cada cierto tiempo, pero en esta ocasión se emplea uno de los temporizadores disponibles para generar la base de tiempo y su interrupción para conmutar al LED. En esta práctica vamos a aprender cómo programar y configurar la funcionalidad descrita.

1.- Iniciar la aplicación MPLAB IDE.

2.- Crear un nuevo proyecto.

Vaya al menú *File* y elija la opción *"New Project..."*. Elija la opción por defecto *"Standalone Project"* de la categoría *"Microchip Embedded"*, seleccione el dispositivo PIC16F18875, verifique que aparece la herramienta de trabajo *Starter Kits (PKOB)* para la tarjeta *Curiosity*, seleccione el compilador **mpasm** y, finalmente, introduzca el nombre y la ruta del proyecto, en este caso "Practica06".

3.- Agregar el código fuente propio.

En la ventana del lado izquierdo, donde aparece el árbol de archivos del proyecto, oprima el botón derecho del ratón sobre la

rama *Source Files* y elija *New* con *"pic_8b_general.asm..."*. Coloque un nombre apropiado al nuevo archivo asm, por ejemplo, *main06.asm*. Finalmente, sustituya el código de ejemplo por el código que proporciona en el anexo 6 *(SBM_source_06.asm).*

4.- Ensamblar el código.

Para ensamblar el código y verificar que no tiene errores, marque el archivo *main06.asm* en el árbol de códigos, oprima el botón derecho y seleccione la opción *Assemble File*. Si no existen errores, en la ventana se salida, en la parte inferior del entorno IDE, aparece la leyenda *"BUILD SUCCESSFUL"*.

5.- Programar el dispositivo con el código.

Asegúrese que la tarjeta está conectada a la computadora y que el led PWR está encendido. En el árbol de códigos, marcar el proyecto (en este caso, Practica06), oprimir el botón derecho del ratón y elegir la opción *Make and Program Device*.

Si todo sale bien y la tarjeta se programa exitosamente, al final aparece el mensaje *"Programming/Verify complete"*.

6. Actividad y reporte.

Estudie el código, particularmente la etapa de configuración del sistema, el programa principal y la rutina de atención a interrupciones (ver figura 6.1). Revise el diagrama esquemático de la tarjeta *Curiosity* y determine cuáles son los componentes, las señales y las terminales involucradas. Infiera el funcionamiento del mismo. Como referencia, tome en cuenta que en el código de la primera práctica la base de tiempo para la conmutación del LED eran las rutinas de retardo con base en lazos de instrucciones. En este caso ya no se emplean dichas

rutinas para definir el tiempo que el LED permanece prendido o pagado. Entonces el punto de interés es que se entienda cómo se logra el efecto de la conmutación del LED, pero esta vez empleando la base de tiempo de un temporizador y la interrupción involucrada.

```
166     ;
167     ;-------------------------------------------------------------------------
168     ;-------------------------------------------------------------------------
169     ; RUTINAS DE ATENCION A INTERRUPCIONES
170     ;-------------------------------------------------------------------------
171     ;
172     ;-------------------------------------------------------------------------
173     ;PROFUNDIDAD DE STACK=2
174     ;-------------------------------------------------------------------------
175     ;RUTINA INTSERV() SERVICIO A INTERRUPCION
176     ;
177     INTSERV:
178                     BANKSEL PIR0
179                     BTFSS   PIR0,TMR0IF   ;ES INTERUPCION de TIMR0?
180                     GOTO    EINTSR      ;SI NO ES INTERUPCION TIMR0, SALIR
181                     BCF     PIR0,TMR0IF   ;LIMPIA FUENTE DE INTERRUPCION
182     ;CONMUTA LED:
183                     CALL    TOGGLE
184     EINTSR:
185
186                     RETFIE
187     ;FIN RUTINA INTSERV
188     ;
```

Figura 6.1. Sección del código fuente con la rutina de atención a interrupciones.

Modifique el código, realizando y probando un cambio a la vez, para corroborar los efectos de deshabilitar el bit *GIE*, limpiar el bit *T0EN*, la omisión del reinicio de la bandera *TMR0IF*. Investigue la función de cada uno de estos bits y explique el comportamiento observado después de cada modificación. Luego, modifique los bits *T0CKPS<3:0>* del registro *T0CON1* y observe el efecto que produce cada valor de pre-escala. Para más detalles sobre la configuración y operación del temporizador TMR0, ver lo conducente en el documento DS40001802E, "PIC16(L)F18855/75 Data Sheet", de Microchip.

REFERENCIAS

- **Gaonkar, R. S. (2007).** Fundamentals of Microcontrollers and Applications in Embedded Systems with PIC microcontrollers. USA: Cengage Learning.

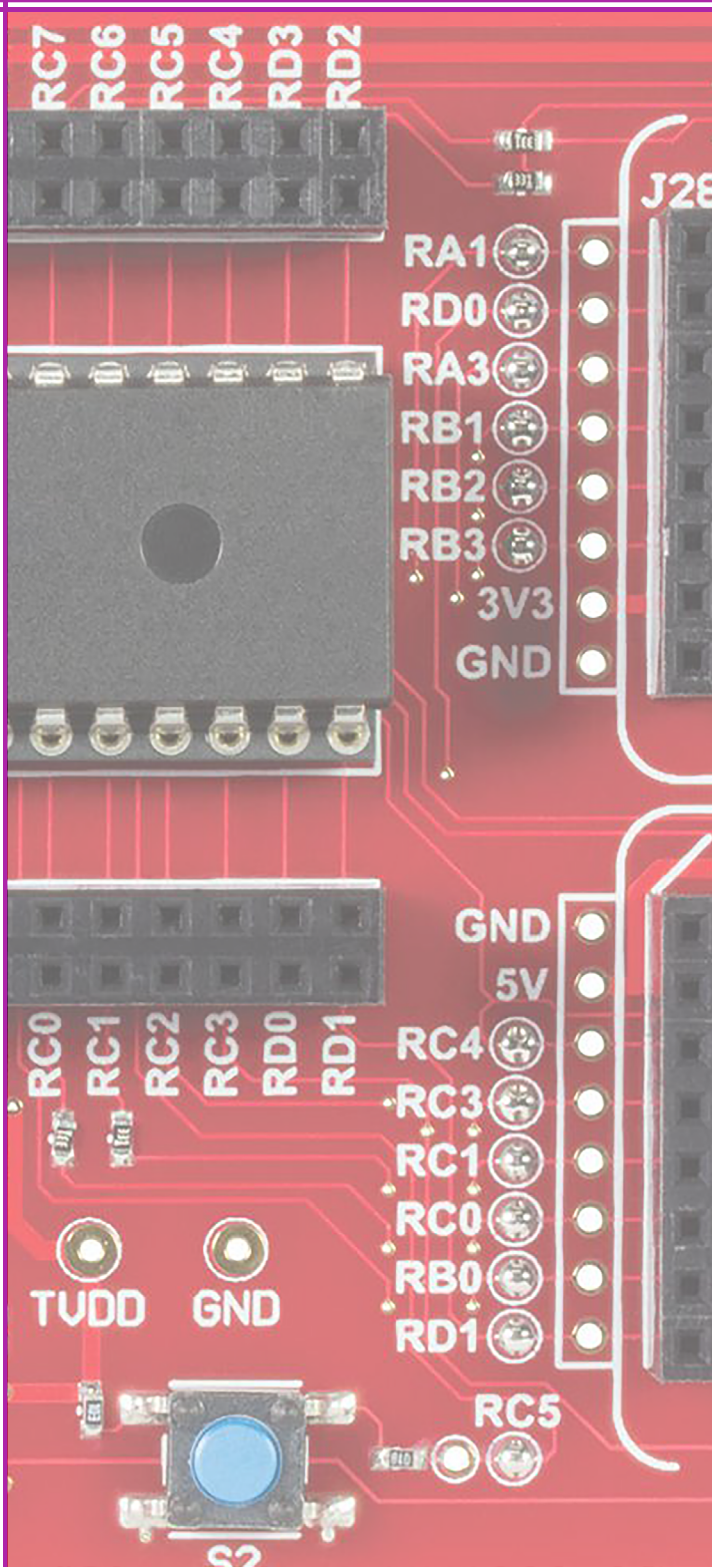
- **Microchip. (2005).** MPASM/MPLINK PICmicro MCU Quick Chart (DS30400G). Consultado el 24 de abril de 2020, de Microchip Sitio web: http://ww1.microchip.com/downloads/en/devicedoc/30400g.pdf

- **Microchip. (2016).** Curiosity High Pin Count (HPC) Development Board User's Guide (DS40001856A). Consultado el 24 de abril de 2020, de Microchip Sitio web: http://ww1.microchip.com/downloads/en/devicedoc/40001856a.pdf

- **Microchip. (2018).** PIC16(L)F18855/75 Data Sheet (DS40001802E). Consultado el 24 de abril 2020, de Microchip Sitio web: http://ww1.microchip.com/downloads/en/DeviceDoc/PIC16(L)F18855_75%20Data%20Sheet_DS40001802E.pdf

PRÁCTICA 7

Aplicación con interfaz de comunicación serial



Dirección
Ciencias Básicas
e Ingeniería

OBJETIVO

Aplicar el concepto de comunicación serial con un microcontrolador, así como el procedimiento de configuración y escritura de código para el uso del periférico de interfaz serial con el estándar RS-232.

ANTECEDENTES

Existen diferentes tipos de tareas que podemos realizar con el apoyo de microcontroladores. Una de estas tareas es la comunicación de datos. Dentro de la comunicación de datos se puede dar el caso en el que sólo se requiera enviar información hacia otro dispositivo, o que sólo se requiera recibir información o, incluso, que se requiera enviar y recibir información.

Para lograr la comunicación de datos existen diferentes técnicas y una de ellas es la que se conoce como comunicación serial. La comunicación serial tiene la característica de que envía la información bit por bit y, por lo tanto, requiere pocos cables para la comunicación. En contraparte, la comunicación paralela puede transmitir varios bits al mismo tiempo, pero también ocupa mayor cantidad de cables. Actualmente, la comunicación serial es la técnica más socorrida. En un enlace de comunicación de datos, a la cantidad de bits por segundo que se transmiten se puede medir en baudios o bits por segundo.

Por su parte, una comunicación serial puede ser *síncrona* o *asíncrona*. La comunicación es síncrona cuando dentro de las señales de control se encuentra una señal de reloj que determina los instantes en que se encuentran disponibles los bits de datos. La comunicación es asíncrona si no se emplea una señal de reloj y, entonces, se recurre a un patrón estandarizado para reconocer el inicio de la secuencia de bits. En particular,

Práctica 7 Aplicación con interfaz de comunicación serial

en esta práctica, emplearemos la comunicación serial asíncrona y un *estándar de referencia* conocido como RS-232. El estándar RS-232, además del formato de la secuencia de pulsos para lograr la sincronización, indica las posibles velocidades de transmisión, las señales de control disponibles y los niveles de voltaje empleados.

El microcontrolador que se empleará en esta práctica cuenta con un módulo *receptor transmisor síncrono asíncrono universal* (USART, por sus siglas en inglés) que soporta tanto comunicaciones síncronas como asíncronas. Específicamente, para propósitos de comunicación serial, el microcontrolador cuenta con algunas terminales especiales, tales como la terminal receptora, que se denota como RX, y la terminal transmisora, que se denota como TX.

MATERIAL

- Base para prototipado (Protoboard).
- Circuito MAX232, de Texas Instruments, o equivalente.
- Conector DB-9 hembra.
- 4 capacitores de 1uF a 16V.
- Cable, soldadura y cautín.
- Multímetro con función para prueba de continuidad.
- Cable de conversión USB-RS232
- Tarjeta modelo Curiosity HPC de marca Microchip.
- Computadora con software MPLAB X IDE v4.05 y programa terminal (TeraTerm o PuTTY).

PROCEDIMIENTO

Antes de conectar la tarjeta Curiosity, asegúrese de que cuenta con el puente de alimentación, en la posición de 3.3V (3V3). Conecte la tarjeta a la computadora de mediante un cable USB A-USB Micro B y compruebe que el led PWR (verde) enciende. En esta práctica, la tarjeta conmutará el LED D2 a intervalos regulares, igual que hasta ahora, pero en esta ocasión se trata de un programa que en su cuerpo principal contiene un lazo infinito y una subrutina para la transmisión de un mensaje a través de la interfaz de comunicación serial (SCI) asignada a la terminal RC6 (Puerto C, terminal 6). Para probar el funcionamiento del programa se empleará un circuito con el estándar RS-232, a fin de transmitir caracteres ASCII y poder observarlos en el programa terminal de una computadora.

1.- Iniciar la aplicación MPLAB IDE.

2.- Crear un nuevo proyecto.

Vaya al menú *File* y elija la opción *"New Project..."*. Elija la opción por defecto *"Standalone Project"* de la categoría *"Microchip Embedded"*, seleccione el dispositivo PIC16F18875, verifique que aparece la herramienta de trabajo *Starter Kits (PKOB)* para la tarjeta *Curiosity*, seleccione el compilador **mpasm** y, finalmente, introduzca el nombre y la ruta del proyecto, en este caso "Practica07".

3.- Agregar el código fuente propio.

En la ventana del lado izquierdo, donde aparece el árbol de archivos del proyecto, oprima el botón derecho del ratón sobre la rama *Source Files* y elija *New* con *"pic_8b_general.asm..."*. Coloque un nombre apropiado al nuevo archivo asm, por ejemplo, *main07.asm*. Finalmente, sustituya el código de ejemplo por el código que proporcionado en el anexo 7 *(SBM_source_07.asm).*

4.- Ensamblar el código.

Para ensamblar el código y verificar que no tiene errores, marque el archivo *main07.asm* en el árbol de códigos, oprima el botón derecho y seleccione la opción *Assemble File*. Si no existen errores, en la ventana se salida, en la parte inferior del entorno IDE, aparece la leyenda *"BUILD SUCCESSFUL"*.

5.- Programar el dispositivo con el código.

Asegúrese que la tarjeta está conectada a la computadora y que el led PWR está encendido. En el árbol de códigos, marcar el proyecto (en este caso, Practica07), oprimir el botón derecho del ratón y elegir la opción *Make and Program Device.*

Si todo sale bien y la tarjeta se programa exitosamente, al final aparece el mensaje *"Programming/Verify complete".*

6. Actividad y reporte.

Estudie el código, particularmente la etapa de configuración del sistema, el programa principal y las rutinas para el puerto serial (ver la figura 7.1). Revise el diagrama esquemático de la tarjeta Curiosity y determine cuáles son los componentes, las señales y las terminales involucradas. Infiera el funcionamiento del mismo. En esta práctica, la rutina clave es la identificada como SENDMSG. Estudie

el código para entender cómo es que esta rutina permite transmitir mensajes a través del puerto de comunicación serial.

```
119     ;-------------------------------------------------------------------------
120     ; Programa principal
121     ;-------------------------------------------------------------------------
122     ;SEÑAL DE ARRANQUE:
123     ;
124                     MOVLW   D'10'   ;CUENTA PARA R3
125                     MOVWF   R3
126     TRY:
127                     MOVLW   D'1'    ;VALOR PARA I2 QUE PRODUCE UN RETARDO DE 0.1 SEG
128                     MOVWF   I2      ;SE CARGA PARAMETRO
129                     CALL    RETARDO ;I2 VECES RETARDO DE 0.1 SEG.
130                     CALL    TOGGLE  ;CONMUTAMOS LED
131                     DECFSZ  R3,F    ; ES CERO?
132                     GOTO    TRY
133     MAIN:
134                     CALL    SENDMSG ;ENVIA MENSAJE POR PUERTO SERIAL
135                     GOTO    MAIN
136
137     ;-------------------------------------------------------------------------
138     ;******************** RUTINAS PARA PUERTO SERIAL ***************************
139     ;-------------------------------------------------------------------------
140     ;PROFUNDIDAD DE STACK=1
141     ;-------------------------------------------------------------------------
142     ;RUTINA SEND(W)
143     ;EN W EL CARACTER A TRANSMITIR
144     SEND:
145             BANKSEL TX1STA
146     TRYSND:
147             BTFSS   TX1STA,TRMT ;ESTA VACIO EL REGISTRO DE CORRIMIENTO DE TX (TSR)?
148             GOTO    TRYSND  ;SI NO ESTA VACIO: VULVE A INTENTAR
149             BANKSEL TXREG
150             MOVWF   TXREG   ;TRANSMITE EL VALOR EN W
151             RETURN
152     ;FIN RUTINA SEND
```

Figura 7.1. Sección del código fuente con la rutina para la transmisión serial de caracteres.

Para más detalles sobre la configuración y operación de la interfaz de comunicación serial (módulo EUSART), ver lo conducente en el documento DS40001802E, *"PIC16(L)F18855/75 Data Sheet", de Microchip*.

Para probar la comunicación serial, siga los siguientes pasos. Primero, desconecte la alimentación de la tarjeta *Curiosity*. Luego, arme el circuito mostrado en la figura 7.2 y realice las conexiones indicadas para acoplar el puerto de comunicación serial del Pic16F18875 (asignado a la terminal RC6) con el dispositivo MAX232 y, con ello, hacer posible su conexión a un puerto RS-232 de la computadora (en nuestro caso, proporcionado a través del cable de conversión USB-RS232). En la figura 7.3 se muestra una vista del circuito armado en una tarjeta de prototipado. Es muy importante que **NO CONECTE NADA SI LA TARJETA ESTÁ ALIMENTADA**. Para conectar cualquier cable o componente, la tarjeta *Curiosity* debe estar apagada por completo.

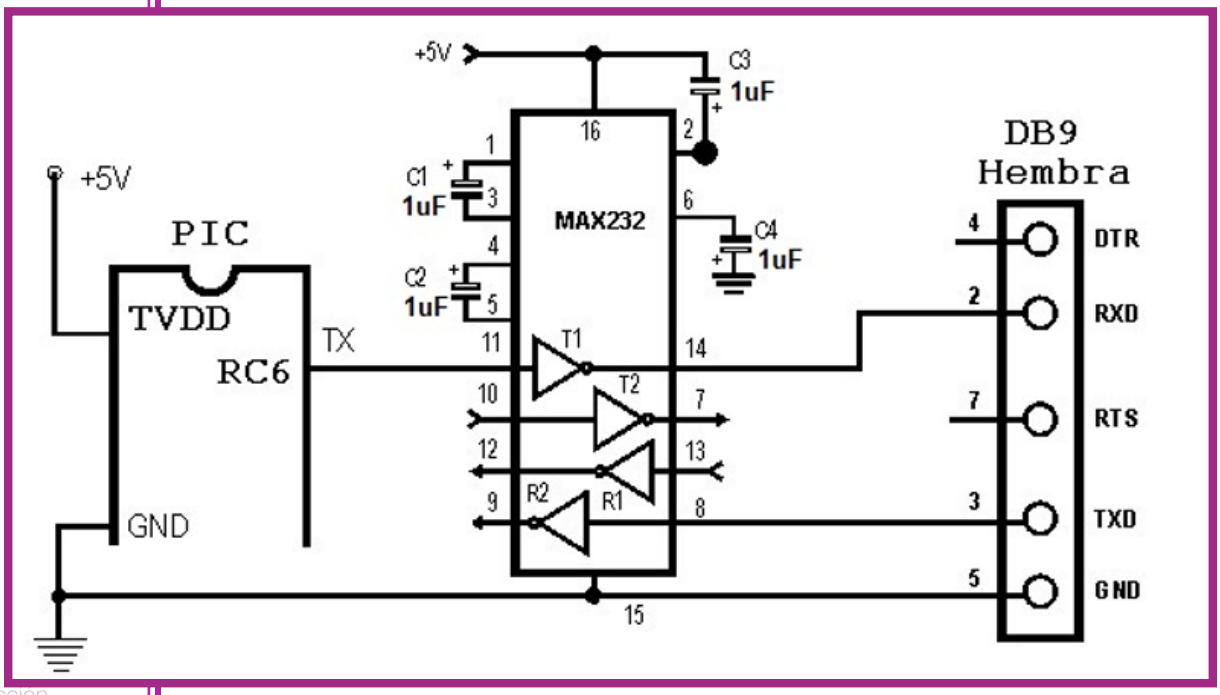


Figura 7.2. Diagrama con el circuito para la interfaz de comunicación serial RS-232.

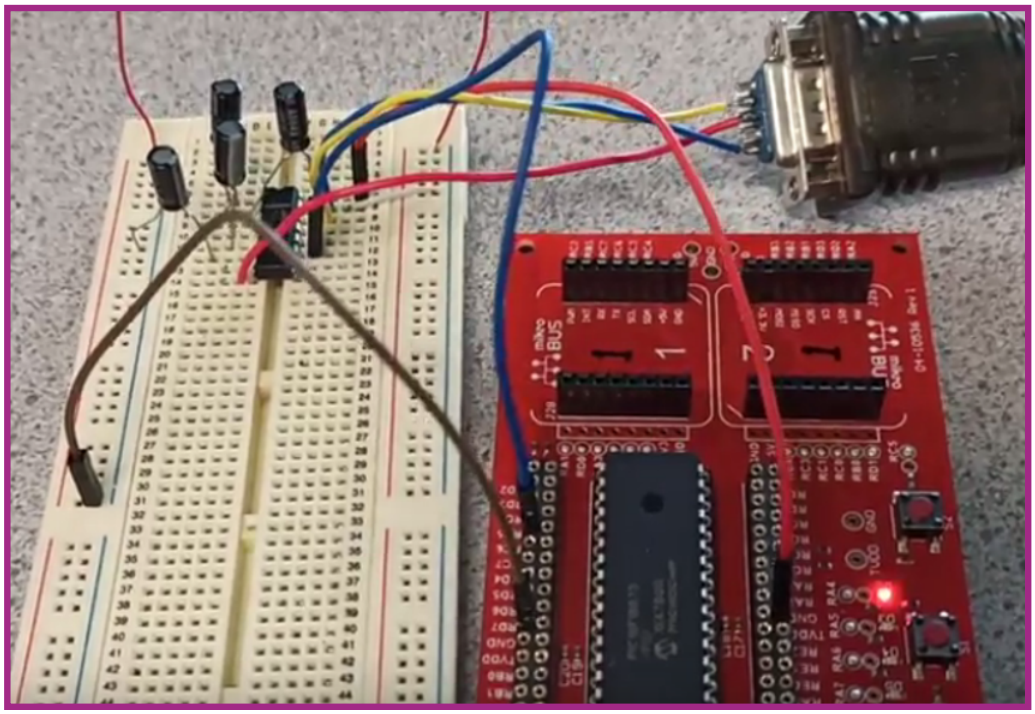


Figura 7.3. Protoboard con el circuito para la interfaz de comunicación serial RS-232.

Revise muy bien todas las conexiones, asegurándose que son correctas, que tienen continuidad y que las polaridades son las especificadas en el diagrama (sobre todo la alimentación y cada uno de los capacitores).

Finalmente, ejecute el programa terminal y configure el puerto serial disponible con los siguientes parámetros (ver figura 7.4):

- 9600 bps
- 8 bits
- Sin paridad
- Sin control de flujo

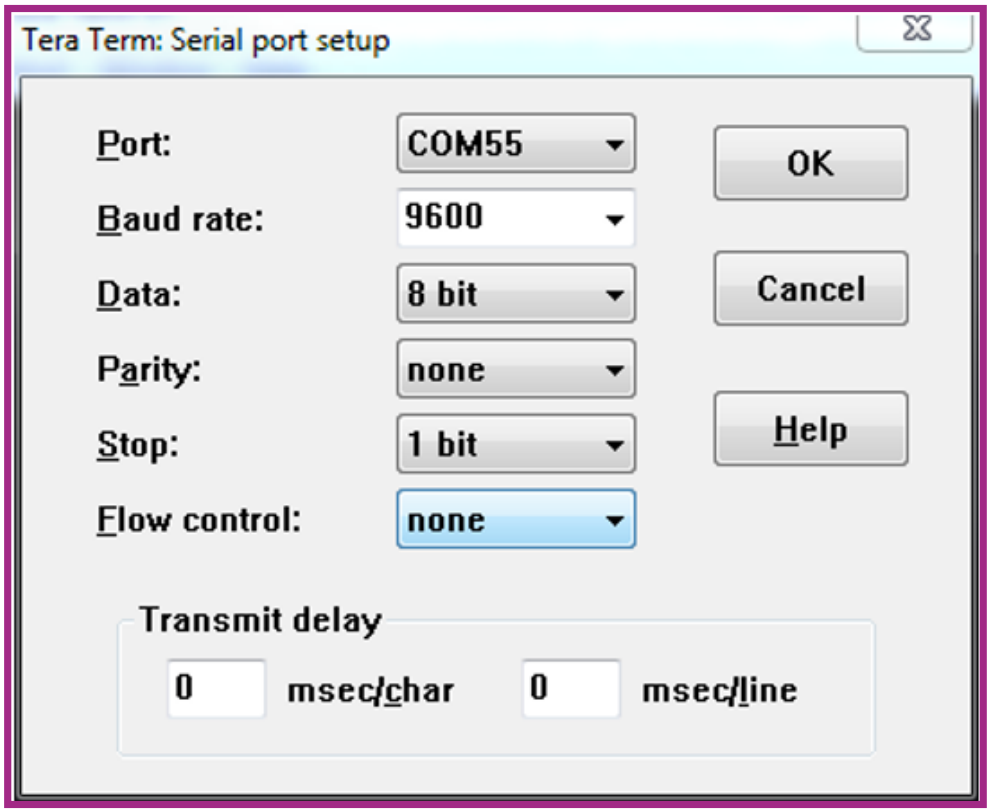


Figura 7.4. Ventana de configuración de la comunicación serial en la terminal Tera Term.

Ahora, conecte el adaptador USB-RS232 a la computadora y al conector DB9 hembra del circuito que armamos. Finalmente, alimente la tarjeta *Curiosity* mediante el cable USB A-USB Micro B. Si todo es correcto, en la ventana del programa terminal deberá observar el mensaje "HOLA" repetido en forma persistente, como se muestra en la figura 7.5.

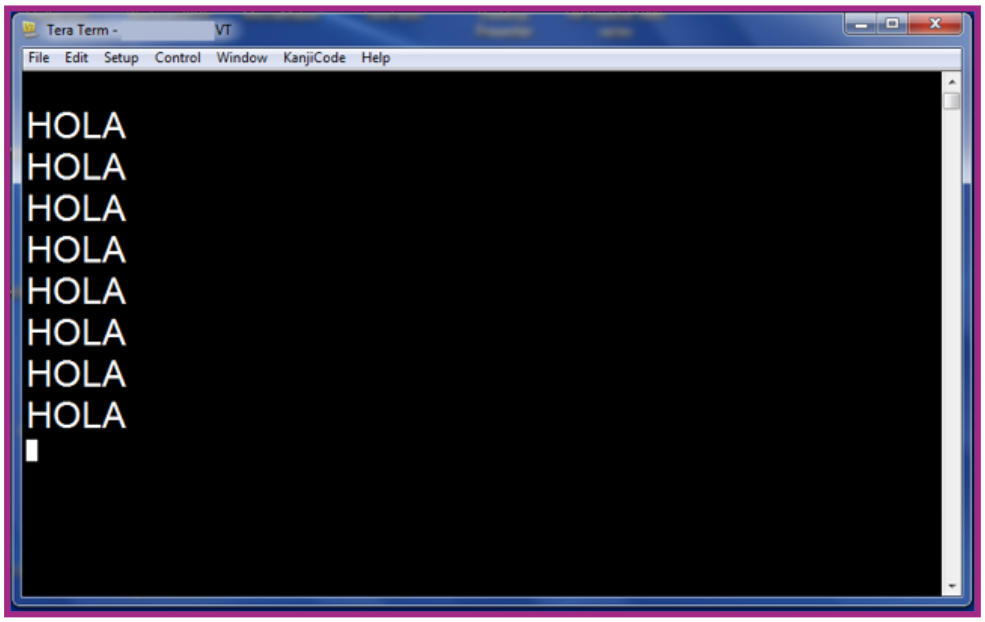


Figura 7.5. Mensaje recibido en el programa terminal.

Modifique el programa para enviar cualquier otro mensaje por la interfaz serial y corrobore que los cambios surten el efecto deseado.
Dentro del reporte, incluya un anexo con el esquemático de sistema conectado a la interfaz de comunicación serial y el dispositivo MAX232. No puede usar la imagen de la figura 7.1 de esta práctica, realice su propio diagrama esquemático.

REFERENCIAS

- **Gaonkar, R. S. (2007).** Fundamentals of Microcontrollers and Applications in Embedded Systems with PIC microcontrollers. USA: Cengage Learning.

- **Microchip. (2005).** MPASM/MPLINK PICmicro MCU Quick Chart (DS30400G). Consultado el 24 de abril de 2020, de Microchip Sitio web: http://ww1.microchip.com/downloads/en/devicedoc/30400g.pdf

- **Microchip. (2016).** Curiosity High Pin Count (HPC) Development Board User's Guide (DS40001856A). Consultado el 24 de abril de 2020, de Microchip Sitio web: http://ww1.microchip.com/downloads/en/devicedoc/40001856a.pdf

- **Microchip. (2018).** PIC16(L)F18855/75 Data Sheet (DS40001802E). Consultado el 24 de abril 2020, de Microchip Sitio web: http://ww1.microchip.com/downloads/en/DeviceDoc/PIC16(L)F18855_75%20Data%20Sheet_DS40001802E.pdf

Anexo 1

Anexo. 1. Código SBM_source_01.asm

```
;*******************************************************************************
;Universidad Autonoma Metropolitana
;24 de abril de 2020
;Gerardo Laguna
;*******************************************************************************
;SBM_source_01.asm. Codigo de practica 01
;Sistemas basados en microcontroladores
;*******************************************************************************

   LIST P=16F18875
;El sistema de representacion por defecto (radix) es hexadecimal.
   #INCLUDE "P16F18875.INC"

; CONFIG1
; __config 0x____
 __CONFIG _CONFIG1, _FEXTOSC_OFF & _RSTOSC_HFINT1 & _
    CLKOUTEN_OFF & _CSWEN_ON & _FCMEN_OFF
; __CONFIG _CONFIG1, _FEXTOSC_ECH & _RSTOSC_EXT1X & _
    CLKOUTEN_OFF & _CSWEN_ON & _FCMEN_ON
; CONFIG2
; __config 0x3FFF
 __CONFIG _CONFIG2, _MCLRE_ON & _PWRTE_OFF & _LPBOREN_
    OFF & _BOREN_ON & _BORV_LO & _ZCD_OFF & _PPS1WAY_ON &
_STVREN_ON
; CONFIG3
; __config 0x3F9F
 __CONFIG _CONFIG3, _WDTCPS_WDTCPS_31 & _WDTE_OFF & _
    WDTCWS_WDTCWS_7 & _WDTCCS_SC
; CONFIG4
; __config 0x3FFF
 __CONFIG _CONFIG4, _WRT_OFF & _SCANE_available & _LVP_ON
; CONFIG5
; __config 0x3FFF
 __                    CONFIG _CONFIG5, _CP_OFF & _CPD_OFF
```

Anexo 1

```
;-----------------------------------------------------------------------------------------------------------
; Definiciones
;-----------------------------------------------------------------------------------------------------------
;-----------------------------------------------------------------------------------------------------------
;     ********************          Variables          ********************
;-----------------------------------------------------------------------------------------------------------
;
R0      EQU   0x70
R1      EQU   0x71
R2      EQU   0x72
R3      EQU   0x73
I0      EQU   0x74
I1      EQU   0x75
I2      EQU   0x76
I3      EQU   0x77
R0BAK       EQU   0x78
;-----------------------------------------------------------------------------------------------------------
; Codigo
;-----------------------------------------------------------------------------------------------------------
;-----------------------------------------------------------------------------------------------------------
; Vector para reset
;-----------------------------------------------------------------------------------------------------------
        ORG     0x00
        GOTO    START
;-----------------------------------------------------------------------------------------------------------
; Vector para interrupciones
;-----------------------------------------------------------------------------------------------------------
        ORG     0x04
        GOTO    INTSERV
;-----------------------------------------------------------------------------------------------------------
; Inicializacion
;-----------------------------------------------------------------------------------------------------------
        ORG     0x05
START:
;SE INICIALIZA PUERTO A:
                BANKSEL     PORTA
                CLRF  PORTA
                BANKSEL     LATA
                CLRF  LATA  ;CLEAR LATCH
                BANKSEL     ANSELA
                CLRF  ANSELA        ;DIGITAL I/O
```

```
                BANKSEL     TRISA
                CLRF  TRISA;ALL OUTPUTS
;
;-------------------------------------------------------------------------------------------------------
; Programa principal
;-------------------------------------------------------------------------------------------------------
;SENAL DE ARRANQUE:
;
                MOVLW       D'10'   ;CUENTA PARA R3
                MOVWF       R3
TRY:
                MOVLW       D'1'    ;VALOR PARA I2 QUE PRODUCE UN
RETARDO DE 0.1 SEG
                MOVWF       I2      ;SE CARGA PARAMETRO
                CALL  RETARDO       ;I2 VECES RETARDO DE 0.1 SEG.
                CALL  TOGGLE        ;CONMUTAMOS LED
                DECFSZ  R3,F        ; ES CERO?
                GOTO    TRY
MAIN:
                MOVLW       D'5'    ;VALOR PARA I2 QUE PRODUCE UN
RETARDO DE 0.5 SEG
                MOVWF       I2      ;SE CARGA PARAMETRO
                CALL  RETARDO       ;I2 VECES RETARDO DE 0.1 SEG.
                CALL  TOGGLE        ;CONMUTAMOS LED
                GOTOMAIN


;-------------------------------------------------------------------------------------------------------
;************************ RUTINAS DE RETARDO ***************************
;-------------------------------------------------------------------------------------------------------
;PROFUNDIDAD DE STACK=1
;-------------------------------------------------------------------------------------------------------
;RUTINA RETARDO(I2) CON BASE EN 0.1 SEG (100 X 1mSEG) CON UNA
FREC DE 1 MHz
;EN I2 NUMERO DE ITERACIONES DE 0.1 SEG
;MODIFICA DIRECTAMENTE I1 E INDIRECTAMENTE A I0
;
RETARDO:
LOOP3:
                MOVLW       D'100' ;CUENTA PARA R1
                MOVWF       I1
```

```
LOOP2:
        CALL MILIRET
        DECFSZ  I1,F    ; LOOP 2.
        GOTOLOOP2
        DECFSZ  I2,F    ; LOOP 3.
        GOTOLOOP3
        RETURN
;FIN RUTINA RETARDO
;
;-------------------------------------------------------------------------------------------------------
;RUTINA MILIRET() PARA UN RETARDO DE 1 MILISEGUNDO.
;QUE CORRESPONDE A 62*16uSEG = 62*(NOP+DECFSZ+GOTO) CON UNA
FREC DE 1 MHz.
;MODIFICA I0
;
MILIRET:
        MOVLW   D'62'
        MOVWF   I0      ;CARGAMOS CONTADOR
MRETL1:
      NOP
        DECFSZ  I0,F    ; loop 1.
      GOTO   MRETL1
        RETURN
;FIN RUTINA MILIRET
;
;-------------------------------------------------------------------------------------------------------
;********************** RUTINAS DE SENALIZACION ******************
;-------------------------------------------------------------------------------------------------------
;PROFUNDIDAD DE STACK=1
;-------------------------------------------------------------------------------------------------------
;RUTINA PARA CONMUTAR RA4
;
TOGGLE:
        BANKSEL  PORTA
        BTFSS    PORTA,RA4
        GOTOSETBA2
;LIMPIA BIT:
        BCF   PORTA,RA4
        GOTOETOGGLE
;PRENDE BIT:
```

```
SETBA2:
        BSF     PORTA,RA4
ETOGGLE:
        RETURN
;FIN RUTINA TOGGLE
;
;-----------------------------------------------------------------------------
;-----------------------------------------------------------------------------
; RUTINAS DE ATENCION A INTERRUPCIONES
;-----------------------------------------------------------------------------
;
;-----------------------------------------------------------------------------
;PROFUNDIDAD DE STACK=
;-----------------------------------------------------------------------------
;RUTINA INTSERV() SERVICIO A INTERRUPCION
;
INTSERV:
    RETFIE
;FIN RUTINA RUTEXT
;
;
DR          DW  'G','A','L','S'
VER         DW  'V','e','r','0','.','0'
FIN         DW  'F','I','N'
;-----------------------------------------------------------------------------

        END
```

Anexo 2

Anexo. 2. Código SBM_source_02.asm

```
;*******************************************************************************
;Universidad Autonoma Metropolitana
;24 de abril de 2020
;Gerardo Laguna
;*******************************************************************************
;SBM_source_02.asm. Codigo de practica 02
;Sistemas basados en microcontroladores
;*******************************************************************************
    LIST P=16F18875
;El sistema de representacion por defecto (radix) es hexadecimal.
    #INCLUDE "P16F18875.INC"

; CONFIG1
; __config 0x____
 __CONFIG _CONFIG1, _FEXTOSC_OFF & _RSTOSC_HFINT1 & _
     CLKOUTEN_OFF
     & _CSWEN_ON & _FCMEN_OFF
; __CONFIG _CONFIG1, _FEXTOSC_ECH & _RSTOSC_EXT1X & _
     CLKOUTEN_OFF
     & _CSWEN_ON & _FCMEN_ON
; CONFIG2
; __config 0x3FFF
 __CONFIG _CONFIG2, _MCLRE_ON & _PWRTE_OFF & _LPBOREN_OFF
& _BO
     REN_ON & _BORV_LO & _ZCD_OFF & _PPS1WAY_ON & _STVREN_ON
; CONFIG3
; __config 0x3F9F
 __CONFIG _CONFIG3, _WDTCPS_WDTCPS_31 & _WDTE_OFF & _
     WDTCWS_WDTCWS_7 & _WDTCCS_SC
; CONFIG4
; __config 0x3FFF
 __CONFIG _CONFIG4, _WRT_OFF & _SCANE_available & _LVP_ON
; CONFIG5
; __config 0x3FFF
 __CONFIG _CONFIG5, _CP_OFF & _CPD_OFF
```

Anexo 2

```
;-------------------------------------------------------------------------------
; Definiciones
;-------------------------------------------------------------------------------
;-------------------------------------------------------------------------------
;  ************************        Variables        ************************
;-------------------------------------------------------------------------------
;
R0      EQU   0x70
R1      EQU   0x71
R2      EQU   0x72
R3      EQU   0x73
I0      EQU   0x74
I1      EQU   0x75
I2      EQU   0x76
I3      EQU   0x77
R0BAK       EQU   0x78
;-------------------------------------------------------------------------------
; Código
;-------------------------------------------------------------------------------
;-------------------------------------------------------------------------------
; Vector para reset
;-------------------------------------------------------------------------------
        ORG     0x00
        GOTO    START
;-------------------------------------------------------------------------------
; Vector para interrupciones
;-------------------------------------------------------------------------------
        ORG     0x04
        GOTO    INTSERV
;-------------------------------------------------------------------------------
; Inicialización
;-------------------------------------------------------------------------------
        ORG     0x05
START:
;SE INICIALIZA PUERTO A:
                BANKSEL     PORTA
                CLRF  PORTA
                BANKSEL     LATA
                CLRF  LATA  ;CLEAR LATCH
                BANKSEL     ANSELA
                CLRF  ANSELA        ;DIGITAL I/O
```

```
                BANKSEL     TRISA
                CLRF  TRISA;ALL OUTPUTS
;
;-------------------------------------------------------------------------------------------------------
; Programa principal
;-------------------------------------------------------------------------------------------------------
;SENAL DE ARRANQUE:
;
                MOVLW       D'10'  ;CUENTA PARA R3
                MOVWF       R3
TRY:
                MOVLW       D'1'   ;VALOR PARA I2 QUE PRODUCE UN
                                    RETARDO DE 0.1 SEG
                MOVWF       I2     ;SE CARGA PARAMETRO
                CALL  RETARDO      ;I2 VECES RETARDO DE 0.1 SEG.
                CALL  TOGGLE       ;CONMUTAMOS LED
                DECFSZ  R3,F     ; ES CERO?
                GOTO    TRY
MAIN:
                MOVLW       D'5'   ;VALOR PARA I2 QUE PRODUCE UN
                                    RETARDO DE 0.5 SEG
                MOVWF       I2     ;SE CARGA PARAMETRO
                CALL  RETARDO      ;I2 VECES RETARDO DE 0.1 SEG.
                CALL  TOGGLE       ;CONMUTAMOS LED
                GOTOMAIN
;-------------------------------------------------------------------------------------------------------
;************************ RUTINAS DE RETARDO ************************
;-------------------------------------------------------------------------------------------------------
;PROFUNDIDAD DE STACK=1
;-------------------------------------------------------------------------------------------------------
;RUTINA RETARDO(I2) CON BASE EN 0.1 SEG (100 X 1mSEG) CON UNA
FREC DE 1 MHz
;EN I2 NUMERO DE ITERACIONES DE 0.1 SEG
;MODIFICA DIRECTAMENTE I1 E INDIRECTAMENTE A I0
;
RETARDO:
LOOP3:
                MOVLW       D'100' ;CUENTA PARA R1
                MOVWF       I1
```

```
LOOP2:
        CALL MILIRET
        DECFSZ  I1,F    ; LOOP 2.
        GOTOLOOP2
        DECFSZ  I2,F    ; LOOP 3.
        GOTOLOOP3
        RETURN
;FIN RUTINA RETARDO
;
;-------------------------------------------------------------------------------------------------------
;RUTINA MILIRET() PARA UN RETARDO DE 1 MILISEGUNDO.
;QUE CORRESPONDE A 62*16uSEG = 62*(NOP+DECFSZ+GOTO) CON UNA
FREC DE 1 MHz.
;MODIFICA I0
;
MILIRET:
        MOVLW   D'62'
        MOVWF   I0      ;CARGAMOS CONTADOR
MRETL1:
      NOP
        DECFSZ  I0,F    ; loop 1.
      GOTO  MRETL1
        RETURN
;FIN RUTINA MILIRET
;
;-------------------------------------------------------------------------------------------------------
;********************** RUTINAS DE SENALIZACION **********************
;-------------------------------------------------------------------------------------------------------
;PROFUNDIDAD DE STACK=1
;-------------------------------------------------------------------------------------------------------
;RUTINA PARA CONMUTAR RA4
;
TOGGLE:
        BANKSEL PORTA
        BTFSS   PORTA,RA4
        GOTOSETBA2
;LIMPIA BIT:
        BCF PORTA,RA4
        GOTOETOGGLE
;PRENDE BIT:
```

Anexo 2

```
SETBA2:
                BSF    PORTA,RA4
ETOGGLE:
                RETURN
;FIN RUTINA TOGGLE
;
;-----------------------------------------------------------------------------------------------
;-----------------------------------------------------------------------------------------------
; RUTINAS DE ATENCION A INTERRUPCIONES
;-----------------------------------------------------------------------------------------------
;
;-----------------------------------------------------------------------------------------------
;PROFUNDIDAD DE STACK=
;-----------------------------------------------------------------------------------------------
;RUTINA INTSERV() SERVICIO A INTERRUPCION
;
INTSERV:
        RETFIE
;FIN RUTINA RUTEXT
;
;
DR              DW    'G','A','L','S'
VER             DW    'V','e','r','0','.','0'
FIN             DW    'F','I','N'
;-----------------------------------------------------------------------------------------------

                END
```

Anexo 3

Anexo. 3. Código SBM_source_03.asm

```
;*******************************************************************************
;Universidad Autonoma Metropolitana
;24 de abril de 2020
;Gerardo Laguna
;*******************************************************************************
;SBM_source_03.asm. Codigo de practica 03
;Sistemas basados en microcontroladores
;*******************************************************************************
    LIST P=16F18875
;El sistema de representacion por defecto (radix) es hexadecimal.
    #INCLUDE "P16F18875.INC"

; CONFIG1
; __config 0x1FEC
 __CONFIG _CONFIG1, _FEXTOSC_OFF & _RSTOSC_HFINT1 & _
     CLKOUTEN_OFF
     & _CSWEN_ON & _FCMEN_OFF
; CONFIG2
; __config 0x3FFF
 __CONFIG _CONFIG2, _MCLRE_ON & _PWRTE_OFF & _LPBOREN_
     OFF & _BO
     REN_ON & _BORV_LO & _ZCD_OFF & _PPS1WAY_ON & _STVREN_
     ON
; CONFIG3
; __config 0x3F9F
 __CONFIG _CONFIG3, _WDTCPS_WDTCPS_31 & _WDTE_OFF & _
     WDTCWS_WDTCWS_7 & _WDTCCS_SC
;__CONFIG _CONFIG3, _WDTCPS_WDTCPS_31 & _WDTE_ON & _
     WDTCWS_WDTCWS_7 & _WDTCCS_SC
; CONFIG4
; __config 0x3FFF
 __CONFIG _CONFIG4, _WRT_OFF & _SCANE_available & _LVP_ON
; CONFIG5
; __config 0x3FFF
 __CONFIG _CONFIG5, _CP_OFF & _CPD_OFF
```

Anexo 3

```
;-------------------------------------------------------------------------------
; Definiciones
;-------------------------------------------------------------------------------
;-------------------------------------------------------------------------------
;     ********************          Variables          **********************
;-------------------------------------------------------------------------------
;
R0      EQU   0x70
R1      EQU   0x71
R2      EQU   0x72
R3      EQU   0x73
I0      EQU   0x74
I1      EQU   0x75
I2      EQU   0x76
I3      EQU   0x77
R0BAK       EQU   0x78
;-------------------------------------------------------------------------------
; Codigo
;-------------------------------------------------------------------------------
;-------------------------------------------------------------------------------
; Vector para reset
;-------------------------------------------------------------------------------
        ORG     0x00
        GOTO    START
;-------------------------------------------------------------------------------
; Vector para interrupciones
;-------------------------------------------------------------------------------
        ORG     0x04
        GOTO    INTSERV
;-------------------------------------------------------------------------------
; Inicialización
;-------------------------------------------------------------------------------
        ORG     0x05
START:
;SE INICIALIZA PUERTO A:
                BANKSEL     PORTA
                CLRF  PORTA
                BANKSEL     LATA
                CLRF  LATA  ;CLEAR LATCH
                BANKSEL     ANSELA
                CLRF  ANSELA        ;DIGITAL I/O
```

```
                BANKSEL     TRISA
                CLRF  TRISA;ALL OUTPUTS
;
;-------------------------------------------------------------------------------------------------
; Programa principal
;-------------------------------------------------------------------------------------------------
;SENAL DE ARRANQUE:
;
                MOVLW       D'10'  ;CUENTA PARA R3
                MOVWF       R3
TRY:
                MOVLW       D'1'   ;VALOR PARA I2 QUE PRODUCE UN
                                    RETARDO DE 0.1 SEG
                MOVWF       I2     ;SE CARGA PARAMETRO
                CALL  RETARDO      ;I2 VECES RETARDO DE 0.1 SEG.
                CALL  TOGGLE       ;CONMUTAMOS LED
                DECFSZ  R3,F     ; ES CERO?
                GOTO    TRY
MAIN:
;               CLRWDT             :LIMPIA WATCHDOG TIMER
                MOVLW       D'5'   ;VALOR PARA I2 QUE PRODUCE UN
                                    RETARDO DE 0.5 SEG
                MOVWF       I2     ;SE CARGA PARAMETRO
                CALL  RETARDO      ;I2 VECES RETARDO DE 0.1 SEG.
                CALL  TOGGLE       ;CONMUTAMOS LED
                GOTOMAIN


;-------------------------------------------------------------------------------------------------
;************************ RUTINAS DE RETARDO ************************
;-------------------------------------------------------------------------------------------------
;PROFUNDIDAD DE STACK=1
;-------------------------------------------------------------------------------------------------
;RUTINA RETARDO(I2) CON BASE EN 0.1 SEG (100 X 1mSEG) CON UNA
FREC DE 1 MHz
;EN I2 NUMERO DE ITERACIONES DE 0.1 SEG
;MODIFICA DIRECTAMENTE I1 E INDIRECTAMENTE A I0
;
RETARDO:
LOOP3:
                MOVLW       D'100' ;CUENTA PARA R1
                MOVWF       I1
```

```
LOOP2:
        CALL MILIRET
        DECFSZ  I1,F    ; LOOP 2.
        GOTOLOOP2
        DECFSZ  I2,F    ; LOOP 3.
        GOTOLOOP3
        RETURN
;FIN RUTINA RETARDO
;
;-------------------------------------------------------------------------------------------------------
;RUTINA MILIRET() PARA UN RETARDO DE 1 MILISEGUNDO.
;QUE CORRESPONDE A 62*16uSEG = 62*(NOP+DECFSZ+GOTO) CON UNA
FREC DE 1 MHz.
;MODIFICA I0
;
MILIRET:
        MOVLW   D'62'
        MOVWF   I0      ;CARGAMOS CONTADOR
MRETL1:
      NOP
        DECFSZ  I0,F    ; loop 1.
      GOTO   MRETL1
        RETURN
;FIN RUTINA MILIRET
;
;-------------------------------------------------------------------------------------------------------
;********************** RUTINAS DE SENALIZACION ******************
;-------------------------------------------------------------------------------------------------------
;PROFUNDIDAD DE STACK=1
;-------------------------------------------------------------------------------------------------------
;RUTINA PARA CONMUTAR RA4
;
TOGGLE:
        BANKSEL  PORTA
        BTFSS    PORTA,RA4
        GOTOSETBA2
;LIMPIA BIT:
        BCF   PORTA,RA4
        GOTOETOGGLE
;PRENDE BIT:
```

Anexo 3

```
SETBA2:
                BSF    PORTA,RA4
ETOGGLE:
                RETURN
;FIN RUTINA TOGGLE
;
;-----------------------------------------------------------------------------------------------
;-----------------------------------------------------------------------------------------------
; RUTINAS DE ATENCION A INTERRUPCIONES
;-----------------------------------------------------------------------------------------------
;
;-----------------------------------------------------------------------------------------------
;PROFUNDIDAD DE STACK=
;-----------------------------------------------------------------------------------------------
;RUTINA INTSERV() SERVICIO A INTERRUPCION
;
INTSERV:
        RETFIE
;FIN RUTINA RUTEXT
;
;
DR              DW    'G','A','L','S'
VER             DW    'V','e','r','0','.','0'
FIN             DW    'F','I','N'
;-------------------------------------------------------

            END
```

Anexo 4

Anexo. 4. Código SBM_source_04.asm

```
;*******************************************************************************
;Universidad Autonoma Metropolitana
;24 de abril de 2020
;Gerardo Laguna
;*******************************************************************************
;SBM_source_04.asm. Codigo de practica 04
;Sistemas basados en microcontroladores
;*******************************************************************************
    LIST P=16F18875
;El sistema de representacion por defecto (radix) es hexadecimal.
    #INCLUDE "P16F18875.INC"

; CONFIG1
; __config 0xFFEC
 __CONFIG _CONFIG1, _FEXTOSC_OFF & _RSTOSC_HFINT1 & _
     CLKOUTEN_OFF & _CSWEN_ON & _FCMEN_ON
; CONFIG2
; __config 0x3FFF
 __CONFIG _CONFIG2, _MCLRE_ON & _PWRTE_OFF & _LPBOREN_
     OFF & _BOREN_ON & _BORV_LO & _ZCD_OFF & _PPS1WAY_ON &
     _STVREN_ON
; CONFIG3
; __config 0x3F9F
 __CONFIG _CONFIG3, _WDTCPS_WDTCPS_31 & _WDTE_OFF & _
     WDTCWS_WDTCWS_7 & _WDTCCS_SC
; CONFIG4
; __config 0x3FFF
 __CONFIG _CONFIG4, _WRT_OFF & _SCANE_available & _LVP_ON
; CONFIG5
; __config 0x3FFF
 __CONFIG _CONFIG5, _CP_OFF & _CPD_OFF
```

Anexo 4

```
;-------------------------------------------------------------------------------
; Definiciones
;-------------------------------------------------------------------------------
;-------------------------------------------------------------------------------
;     **********************        Variables        **********************
;-------------------------------------------------------------------------------
;
R0      EQU   0x70
R1      EQU   0x71
R2      EQU   0x72
R3      EQU   0x73
I0      EQU   0x74
I1      EQU   0x75
I2      EQU   0x76
I3      EQU   0x77
R0BAK       EQU   0x78
;-------------------------------------------------------------------------------
; Codigo
;-------------------------------------------------------------------------------
;-------------------------------------------------------------------------------
; Vector para reset
;-------------------------------------------------------------------------------
        ORG     0x00
        GOTO    START
;-------------------------------------------------------------------------------
; Vector para interrupciones
;-------------------------------------------------------------------------------
        ORG     0x04
        GOTO    INTSERV
;-------------------------------------------------------------------------------
; Inicializacion
;-------------------------------------------------------------------------------
        ORG     0x05
START:
;SE INICIALIZA PUERTO A:
                BANKSEL     PORTA
                CLRF  PORTA
                BANKSEL     LATA
                CLRF  LATA  ;CLEAR LATCH
                BANKSEL     ANSELA
                CLRF  ANSELA        ;DIGITAL I/O
```

```
                BANKSEL     TRISA
                CLRF  TRISA;ALL OUTPUTS
;SE INICIALIZA PUERTO B:
                BANKSEL     PORTB
                CLRF  PORTB
                BANKSEL     LATB
                CLRF  LATB ;CLEAR LATCH
                BANKSEL     ANSELB
                CLRF  ANSELB       ;DIGITAL I/O
                BANKSEL     TRISB
                MOVLW       0xFF
                MOVWF       TRISB;ALL INPUTS
;
;-------------------------------------------------------------------------------------------------------------
; Programa principal
;-------------------------------------------------------------------------------------------------------------
;SENAL DE ARRANQUE:
;
                MOVLW       D'10'   ;CUENTA PARA R3
                MOVWF       R3
TRY:
                MOVLW       D'1'    ;VALOR PARA I2 QUE PRODUCE UN
                                     RETARDO
                                     DE 0.1 SEG
                MOVWF       I2      ;SE CARGA PARAMETRO
                CALL  RETARDO   ;I2 VECES RETARDO DE 0.1 SEG.
                CALL  TOGGLE    ;CONMUTAMOS LED
                DECFSZ  R3,F    ; ES CERO?
                GOTO    TRY
;Rapido o lento?
MAIN:
                BTFSS       PORTB,RB4   ;Esta oprimido SW1 (logica negativa)?
                GOTOSPEEDY          ;Si SW1 esta oprimido: LED parpadea rapido
                MOVLW       D'5'    ;VALOR PARA I2 QUE PRODUCE UN
RETARDO
                                      DE 0.5 SEG
                MOVWF       I2        ;SE CARGA PARAMETRO
                GOTOSUBR
```

```
SPEEDY:
        MOVLW   D'1'    ;VALOR PARA I2 QUE PRODUCE UN
                         RETARDO
                        DE 0.1 SEG
        MOVWF   I2      ;SE CARGA PARAMETRO
SUBR:
        CALL RETARDO    ;I2 VECES RETARDO DE 0.1 SEG.
        CALL TOGGLE     ;CONMUTAMOS LED
        GOTOMAIN

;-------------------------------------------------------------------------------
;********************** RUTINAS DE RETARDO ****************************
;-------------------------------------------------------------------------------
;PROFUNDIDAD DE STACK=1
;-------------------------------------------------------------------------------
;RUTINA RETARDO(I2) CON BASE EN 0.1 SEG (100 X 1mSEG) CON UNA
FREC DE 1 MHz
;EN I2 NUMERO DE ITERACIONES DE 0.1 SEG
;MODIFICA DIRECTAMENTE I1 E INDIRECTAMENTE A I0
;
RETARDO:
LOOP3:
        MOVLW   D'100' ;CUENTA PARA R1
        MOVWF   I1
LOOP2:
        CALL MILIRET
        DECFSZ  I1,F    ; LOOP 2.
        GOTOLOOP2
        DECFSZ  I2,F    ; LOOP 3.
        GOTOLOOP3
        RETURN
;FIN RUTINA RETARDO
;
;-------------------------------------------------------------------------------
```

```
;RUTINA MILIRET() PARA UN RETARDO DE 1 MILISEGUNDO.
;QUE CORRESPONDE A 62*16uSEG = 62*(NOP+DECFSZ+GOTO) CON UNA
FREC DE 1 MHz.
;MODIFICA I0
;
MILIRET:
            MOVLW     D'62'
            MOVWF     I0      ;CARGAMOS CONTADOR
MRETL1:
          NOP
            DECFSZ  I0,F    ; loop 1.
          GOTO    MRETL1
            RETURN
;FIN RUTINA MILIRET
;
;-------------------------------------------------------------------------------
;********************** RUTINAS DE SENALIZACION **********************
;-------------------------------------------------------------------------------
;PROFUNDIDAD DE STACK=1
;-------------------------------------------------------------------------------
;RUTINA PARA CONMUTAR RA4
;
TOGGLE:
            BANKSEL   PORTA
            BTFSS     PORTA,RA4
            GOTOSETBA2
;LIMPIA BIT:
            BCF   PORTA,RA4
            GOTOETOGGLE
;PRENDE BIT:
SETBA2:
            BSF   PORTA,RA4
ETOGGLE:
          RETURN
;FIN RUTINA TOGGLE
;
```

Anexo 4

```
;-------------------------------------------------------------------------------
;-------------------------------------------------------------------------------
; RUTINAS DE ATENCION A INTERRUPCIONES
;-------------------------------------------------------------------------------
;
;-------------------------------------------------------------------------------
;PROFUNDIDAD DE STACK=
;-------------------------------------------------------------------------------
;RUTINA INTSERV() SERVICIO A INTERRUPCION
;
INTSERV:
        RETFIE
;FIN RUTINA RUTEXT
;
;
DR              DW      'G','A','L','S'
VER             DW      'V','e','r','0','.','0'
FIN             DW      'F','I','N'
;-------------------------------------------------------------------------------
            END
```

Anexo 5

Anexo. 5. Código SBM_source_05.asm

```
;*******************************************************************************
;Universidad Autonoma Metropolitana
;24 de abril de 2020
;Gerardo Laguna
;*******************************************************************************
;SBM_source_05.asm. Codigo de practica 05
;Sistemas basados en microcontroladores
;*******************************************************************************
    LIST P=16F18875
;El sistema de representacion por defecto (radix) es hexadecimal.
    #INCLUDE "P16F18875.INC"

; CONFIG1
; __config 0xFFEC
 __CONFIG _CONFIG1, _FEXTOSC_OFF & _RSTOSC_HFINT1 & _
     CLKOUTEN_OFF & _CSWEN_ON & _FCMEN_ON
; CONFIG2
; __config 0x3FFF
 __CONFIG _CONFIG2, _MCLRE_ON & _PWRTE_OFF & _LPBOREN_
     OFF & _BOREN_ON & _BORV_LO & _ZCD_OFF & _PPS1WAY_ON &
     _STVREN_ON
; CONFIG3
; __config 0x3F9F
 __CONFIG _CONFIG3, _WDTCPS_WDTCPS_31 & _WDTE_OFF & _
     WDTCWS_WDTCWS_7 & _WDTCCS_SC
; CONFIG4
; __config 0x3FFF
 __CONFIG _CONFIG4, _WRT_OFF & _SCANE_available & _LVP_ON
; CONFIG5
; __config 0x3FFF
 __CONFIG _CONFIG5, _CP_OFF & _CPD_OFF
```

Anexo 5

```
;-------------------------------------------------------------------------------
; Definiciones
;-------------------------------------------------------------------------------
;-------------------------------------------------------------------------------
;     **********************        Variables        **********************
;-------------------------------------------------------------------------------
;
R0      EQU   0x70
R1      EQU   0x71
R2      EQU   0x72
R3      EQU   0x73
I0      EQU   0x74
I1      EQU   0x75
I2      EQU   0x76
I3      EQU   0x77
FLAGS       EQU   0x78
R0BAK       EQU   0x79
;-------------------------------------------------------------------------------
;     **********************        Constantes        **********************
;-------------------------------------------------------------------------------
;
;FLAGS       EQU   0x78
SPEEDYF     EQU   0
;-------------------------------------------------------------------------------
; Codigo
;-------------------------------------------------------------------------------
;-------------------------------------------------------------------------------
; Vector para reset
;-------------------------------------------------------------------------------
        ORG     0x00
        GOTO    START
;-------------------------------------------------------------------------------
; Vector para interrupciones
;-------------------------------------------------------------------------------
        ORG     0x04
        GOTO    INTSERV
```

Anexo 5

```
;-------------------------------------------------------------------------------------------------------------
; Inicializacion
;-------------------------------------------------------------------------------------------------------------
        ORG     0x05
START:
;SE INICIALIZA PUERTO A:
                BANKSEL     PORTA
                CLRF PORTA
                BANKSEL     LATA
                CLRF LATA ;CLEAR LATCH
                BANKSEL     ANSELA
                CLRF ANSELA       ;DIGITAL I/O
                BANKSEL     TRISA
                CLRF TRISA;ALL OUTPUTS
;SE INICIALIZA PUERTO B:
                BANKSEL     PORTB
                CLRF PORTB
                BANKSEL     LATB
                CLRF LATB ;CLEAR LATCH
                BANKSEL     ANSELB
                CLRF ANSELB       ;DIGITAL I/O
                BANKSEL     TRISB
                MOVLW       0xFF
                MOVWF       TRISB;ALL INPUTS
;SE INICIALIZAN INTERRUPCIONES:
                BANKSEL INTPPS
                MOVLW       0x0C   ;ELIGE RB4 COMO FUENTE PARA
INTERRUPCION EXTERNA
                MOVWF       INTPPS
                BANKSEL PIE0
                BSF    PIE0,INTE   ;HABILITA INTERRUPCION EXTERNA
                BCF    INTCON,INTEDG   ;ACTIVACION POR FLANCO DE
BAJADA
                BSF    INTCON,GIE  ;HABILITA INTERRUPCIONES
;
;-------------------------------------------------------------------------------------------------
; Programa principal
;-------------------------------------------------------------------------------------------------
;SENAL DE ARRANQUE:
;
                MOVLW       D'10'   ;CUENTA PARA R3
```

```
                MOVWF     R3
TRY:
                MOVLW     D'1'    ;VALOR PARA I2 QUE PRODUCE UN
                                   RETARDO DE 0.1 SEG
                MOVWF     I2      ;SE CARGA PARAMETRO
                CALL RETARDO      ;I2 VECES RETARDO DE 0.1 SEG.
                CALL TOGGLE       ;CONMUTAMOS LED
                DECFSZ  R3,F      ; ES CERO?
                GOTO    TRY
;Rapido o lento?
MAIN:
                BTFSS     FLAGS,SPEEDYF   ;Esta activa la bandera?
                GOTOSLOW          ;Si la bandera no esta activa: LED parpadea lento
                MOVLW     D'1'    ;VALOR PARA I2 QUE PRODUCE UN
                                   RETARDO DE 0.1 SEG
                MOVWF     I2      ;SE CARGA PARAMETRO
                GOTOSUBR
SLOW:
                MOVLW     D'5'    ;VALOR PARA I2 QUE PRODUCE UN
                                   RETARDO DE 0.5 SEG
                MOVWF     I2      ;SE CARGA PARAMETRO
SUBR:
                CALL RETARDO      ;I2 VECES RETARDO DE 0.1 SEG.
                CALL TOGGLE       ;CONMUTAMOS LED
                GOTOMAIN


;-------------------------------------------------------------------------------
;**********************    RUTINAS DE RETARDO    ************************
;-------------------------------------------------------------------------------
;PROFUNDIDAD DE STACK=1
;-------------------------------------------------------------------------------
;RUTINA RETARDO(I2) CON BASE EN 0.1 SEG (100 X 1mSEG) CON UNA
FREC DE 1 MHz
;EN I2 NUMERO DE ITERACIONES DE 0.1 SEG
;MODIFICA DIRECTAMENTE I1 E INDIRECTAMENTE A I0
;
RETARDO:
LOOP3:
                MOVLW     D'100' ;CUENTA PARA R1
                MOVWF     I1
LOOP2:
```

```
                CALL MILIRET
                DECFSZ  I1,F    ; LOOP 2.
                GOTOLOOP2
                DECFSZ  I2,F    ; LOOP 3.
                GOTOLOOP3
                RETURN
;FIN RUTINA RETARDO
;
;-------------------------------------------------------------------------------------------------------
;RUTINA MILIRET() PARA UN RETARDO DE 1 MILISEGUNDO.
;QUE CORRESPONDE A 62*16uSEG = 62*(NOP+DECFSZ+GOTO) CON UNA
FREC DE 1 MHz.
;MODIFICA I0
;
MILIRET:
                MOVLW       D'62'
                MOVWF       I0      ;CARGAMOS CONTADOR
MRETL1:
            NOP
                DECFSZ  I0,F    ; loop 1.
            GOTO    MRETL1
                RETURN
;FIN RUTINA MILIRET
;
;-------------------------------------------------------------------------------------------------------
;******************** RUTINAS DE SENALIZACION ************************
;-------------------------------------------------------------------------------------------------------
;PROFUNDIDAD DE STACK=1
;-------------------------------------------------------------------------------------------------------
;RUTINA PARA CONMUTAR RA4
;
TOGGLE:
                BANKSEL     PORTA
                BTFSS       PORTA,RA4
                GOTOSETBA2
;LIMPIA BIT:
                BCF    PORTA,RA4
                GOTOETOGGLE
;PRENDE BIT:
SETBA2:
                BSF    PORTA,RA4
```

```
ETOGGLE:
                RETURN
;FIN RUTINA TOGGLE
;
;-----------------------------------------------------------------------------
;-----------------------------------------------------------------------------
; RUTINAS DE ATENCION A INTERRUPCIONES
;-----------------------------------------------------------------------------
;
;-----------------------------------------------------------------------------
;PROFUNDIDAD DE STACK=1
;-----------------------------------------------------------------------------
;RUTINA INTSERV() SERVICIO A INTERRUPCION
;
INTSERV:
                BANKSEL     PIR0
                BTFSS       PIR0,INTF   ;ES INTERUPCION EXTERNA?
                GOTO        EINTSR      ;SI NO ES INTERUPCION EXTERNA,
                                         SALIR
                BCF   PIR0,INTF   ;LIMPIA FUENTE DE INTERRUPCION
;CONMUTA BANDERA:
                BTFSS       FLAGS,SPEEDYF
                GOTOSETFLAG
;LIMPIA BANDERA:
                BCF       FLAGS,SPEEDYF
                GOTO   EINTSR
;PRENDE BANDERA:
SETFLAG:
                BSF   FLAGS,SPEEDYF
EINTSR:

                RETFIE
;FIN RUTINA INTSERV
;
;
DR              DW    'G','A','L','S'
VER             DW    'V','e','r','0','.','0'
FIN             DW    'F','I','N'
;-------------------------------------------------------------------------------

            END
```

Anexo 6

Anexo. 6. Código SBM_source_06.asm

```
;*******************************************************************************
;Universidad Autonoma Metropolitana
;24 de abril de 2020
;Gerardo Laguna
;*******************************************************************************
;SBM_source_06.asm. Codigo de practica 06
;Sistemas basados en microcontroladores
;*******************************************************************************

    LIST P=16F18875
;El sistema de representacion por defecto (radix) es hexadecimal.
    #INCLUDE "P16F18875.INC"

; CONFIG1
; __config 0xFFEC
 __CONFIG _CONFIG1, _FEXTOSC_OFF & _RSTOSC_HFINT1 & _
    CLKOUTEN_OFF & _CSWEN_ON & _FCMEN_ON
; CONFIG2
; __config 0x3FFF
 __CONFIG _CONFIG2, _MCLRE_ON & _PWRTE_OFF & _LPBOREN_
    OFF & _BOREN_ON & _BORV_LO & _ZCD_OFF & _PPS1WAY_ON &
_STVREN_ON
; CONFIG3
; __config 0x3F9F
 __CONFIG _CONFIG3, _WDTCPS_WDTCPS_31 & _WDTE_OFF & _
    WDTCWS_WDTCWS_7 & _WDTCCS_SC
; CONFIG4
; __config 0x3FFF
 __CONFIG _CONFIG4, _WRT_OFF & _SCANE_available & _LVP_ON
; CONFIG5
; __config 0x3FFF
 __CONFIG _CONFIG5, _CP_OFF & _CPD_OFF
```

Anexo 6

```
;-------------------------------------------------------------------------------
; Definiciones
;-------------------------------------------------------------------------------
;-------------------------------------------------------------------------------
;  ***********************        Variables        ***************************
;-------------------------------------------------------------------------------
;
R0      EQU   0x70
R1      EQU   0x71
R2      EQU   0x72
R3      EQU   0x73
I0      EQU   0x74
I1      EQU   0x75
I2      EQU   0x76
I3      EQU   0x77
R0BAK       EQU   0x79
;-------------------------------------------------------------------------------
; Codigo
;-------------------------------------------------------------------------------
;-------------------------------------------------------------------------------
; Vector para reset
;-------------------------------------------------------------------------------
        ORG     0x00
        GOTO    START
;-------------------------------------------------------------------------------
; Vector para interrupciones
;-------------------------------------------------------------------------------
        ORG     0x04
        GOTO    INTSERV
;-------------------------------------------------------------------------------
; Inicializacion
;-------------------------------------------------------------------------------
        ORG     0x05
START:
;SE INICIALIZA PUERTO A:
                BANKSEL     PORTA
                CLRF  PORTA
                BANKSEL     LATA
                CLRF  LATA  ;CLEAR LATCH
                BANKSEL     ANSELA
                CLRF  ANSELA        ;DIGITAL I/O
```

```
                BANKSEL     TRISA
                CLRF  TRISA;ALL OUTPUTS
;SE INICIALIZA TIMER:
                BANKSEL     T0CON0
                BSF    T0CON0,T0EN       ;HABILITA TIMER
                BSF    T0CON0,T016BIT    ;DE 16 BITS
                BANKSEL     T0CON1
                MOVLW       0x50    ;FUENTE=Fosc/4, ASINCRONO
                MOVWF       T0CON1
;SE INICIALIZAN INTERRUPCIONES:
                BANKSEL PIE0
                BSF    PIE0,TMR0IE   ;HABILITA INTERRUPCION DE TIMER 0
                BSF    INTCON,GIE  ;HABILITA INTERRUPCIONES


;
;-------------------------------------------------------------------------------------------------------
; Programa principal
;-------------------------------------------------------------------------------------------------------
;SENAL DE ARRANQUE:
;
                MOVLW       D'10'   ;CUENTA PARA R3
                MOVWF       R3
TRY:
                MOVLW       D'1'    ;VALOR PARA I2 QUE PRODUCE UN
                                     RETARDO
                                     DE 0.1 SEG
                MOVWF       I2      ;SE CARGA PARAMETRO
                CALL  RETARDO    ;I2 VECES RETARDO DE 0.1 SEG.
                CALL  TOGGLE     ;CONMUTAMOS LED
                DECFSZ  R3,F    ; ES CERO?
                GOTO    TRY
MAIN:
                GOTOMAIN
```

```
;-------------------------------------------------------------------------------
;     ********************* RUTINAS DE RETARDO ************************
;-------------------------------------------------------------------------------
;PROFUNDIDAD DE STACK=1
;-------------------------------------------------------------------------------
;RUTINA RETARDO(I2) CON BASE EN 0.1 SEG (100 X 1mSEG) CON UNA
FREC DE 1 MHz
;EN I2 NUMERO DE ITERACIONES DE 0.1 SEG
;MODIFICA DIRECTAMENTE I1 E INDIRECTAMENTE A I0
;
RETARDO:
LOOP3:
                MOVLW       D'100' ;CUENTA PARA R1
                MOVWF       I1
LOOP2:
                CALL MILIRET
                DECFSZ  I1,F     ; LOOP 2.
                GOTOLOOP2
                DECFSZ  I2,F     ; LOOP 3.
                GOTOLOOP3
                RETURN
;FIN RUTINA RETARDO
;
;-------------------------------------------------------------------------------
;RUTINA MILIRET() PARA UN RETARDO DE 1 MILISEGUNDO.
;QUE CORRESPONDE A 62*16uSEG = 62*(NOP+DECFSZ+GOTO) CON UNA
FREC DE 1 MHz.
;MODIFICA I0
;
MILIRET:
                MOVLW       D'62'
                MOVWF       I0       ;CARGAMOS CONTADOR
MRETL1:
              NOP
                DECFSZ  I0,F     ; loop 1.
              GOTO    MRETL1
                RETURN
;FIN RUTINA MILIRET
;
```

```
;-------------------------------------------------------------------------------
;********************* RUTINAS DE SENALIZACION ************************
;-------------------------------------------------------------------------------
;PROFUNDIDAD DE STACK=1
;-------------------------------------------------------------------------------
;RUTINA PARA CONMUTAR RA4
;
TOGGLE:
        BANKSEL   PORTA
        BTFSS     PORTA,RA4
        GOTOSETBA2
;LIMPIA BIT:
        BCF   PORTA,RA4
        GOTOETOGGLE
;PRENDE BIT:
SETBA2:
        BSF   PORTA,RA4
ETOGGLE:
        RETURN
;FIN RUTINA TOGGLE
;
;-------------------------------------------------------------------------------
;-------------------------------------------------------------------------------
; RUTINAS DE ATENCION A INTERRUPCIONES
;-------------------------------------------------------------------------------
;
;-------------------------------------------------------------------------------
;PROFUNDIDAD DE STACK=2
;-------------------------------------------------------------------------------
;RUTINA INTSERV() SERVICIO A INTERRUPCION
;
INTSERV:
        BANKSEL   PIR0
        BTFSS     PIR0,TMR0IF   ;ES INTERUPCION de TIMR0?
        GOTOEINTSR          ;SI NO ES INTERUPCION TIMR0, SALIR
        BCF   PIR0,TMR0IF   ;LIMPIA FUENTE DE INTERRUPCION
```

```
;CONMUTA LED:
                CALL  TOGGLE
EINTSR:

                RETFIE
;FIN RUTINA INTSERV
;
;
DR              DW    'G','A','L','S'
VER             DW    'V','e','r','0','.','0'
FIN             DW    'F','I','N'
;---------------------------------------------------------------

            END
```

Anexo 7

Anexo. 7. Código SBM_source_07.asm

```
;*****************************************************************
;Universidad Autonoma Metropolitana
;24 de abril de 2020
;Gerardo Laguna
;*****************************************************************
;SBM_source_07.asm. Codigo de practica 07
;Sistemas basados en microcontroladores
;*****************************************************************

    LIST P=16F18875
;El sistema de representacion por defecto (radix) es hexadecimal.
    #INCLUDE "P16F18875.INC"

; CONFIG1
; __config 0xFFEC
 __CONFIG _CONFIG1, _FEXTOSC_OFF & _RSTOSC_HFINT1 & _
     CLKOUTEN_OFF & _CSWEN_ON & _FCMEN_ON
; CONFIG2
; __config 0x3FFF
 __CONFIG _CONFIG2, _MCLRE_ON & _PWRTE_OFF & _LPBOREN_
     OFF & _BOREN_ON & _BORV_LO & _ZCD_OFF & _PPS1WAY_ON &
_STVREN_ON
; CONFIG3
; __config 0x3F9F
 __CONFIG _CONFIG3, _WDTCPS_WDTCPS_31 & _WDTE_OFF & _
     WDTCWS_WDTCWS_7 & _WDTCCS_SC
; CONFIG4
; __config 0x3FFF
 __CONFIG _CONFIG4, _WRT_OFF & _SCANE_available & _LVP_ON
; CONFIG5
; __config 0x3FFF
 __CONFIG _CONFIG5, _CP_OFF & _CPD_OFF
```

Anexo 7

```
;-------------------------------------------------------------------------------
; Definiciones
;-------------------------------------------------------------------------------
;-------------------------------------------------------------------------------
;     ************************      Variables      ************************
;-------------------------------------------------------------------------------
;
R0      EQU   0x70
R1      EQU   0x71
R2      EQU   0x72
R3      EQU   0x73
I0      EQU   0x74
I1      EQU   0x75
I2      EQU   0x76
I3      EQU   0x77
R0BAK       EQU   0x79
;-------------------------------------------------------------------------------
; Codigo
;-------------------------------------------------------------------------------
;-------------------------------------------------------------------------------
; Vector para reset
;-------------------------------------------------------------------------------
        ORG     0x00
        GOTO    START
;-------------------------------------------------------------------------------
; Vector para interrupciones
;-------------------------------------------------------------------------------
        ORG     0x04
        GOTO    INTSERV
;-------------------------------------------------------------------------------
; Inicializacion
;-------------------------------------------------------------------------------
        ORG     0x05
START:
;SE INICIALIZA PUERTO A:
                BANKSEL     PORTA
                CLRF PORTA
                BANKSEL     LATA
                CLRF LATA ;CLEAR LATCH
                BANKSEL     ANSELA
                CLRF ANSELA       ;DIGITAL I/O
```

```
        BANKSEL   TRISA
        CLRF TRISA;ALL OUTPUTS
;SE INICIALIZA PUERTO C:
        BANKSEL   PORTC
        CLRF PORTC
        BANKSEL   LATC
        CLRF LATC   ;CLEAR LATCH
        BANKSEL   ANSELC
        CLRF ANSELC     ;DIGITAL I/O
        BANKSEL   TRISC
        BCF   TRISC,RC6   ;RC6 OUTPUT (SE VA ASIGNAR A TX)
;SE INICIALIZA TIMER:
        BANKSEL   T0CON0
        BSF   T0CON0,T0EN       ;HABILITA TIMER
        BSF   T0CON0,T016BIT    ;DE 16 BITS
        BANKSEL   T0CON1
        MOVLW     0x50    ;FUENTE=Fosc/4, ASINCRONO
        MOVWF     T0CON1
;SE INICIALIZAN INTERRUPCIONES:
        BANKSEL PIE0
        BSF   PIE0,TMR0IE   ;HABILITA INTERRUPCION DE TIMER 0
        BSF   INTCON,GIE  ;HABILITA INTERRUPCIONES
;SE INICIALIZA INTERFAZ DE COMUNICACION SERIAL:
        BANKSEL   RC6PPS
        MOVLW     0x10    ;ASIGNA LA SALIDA DEL PERIFERICO
SERIAL (TX) a RC6
        MOVWF     RC6PPS
        BANKSEL   BAUD1CON
        BCF   BAUD1CON,SCKP     ;IDLE TX ASOCIADO A NIVEL
ALTO (PREVIENDO EL USO DE UN MAX232)
        BSF   BAUD1CON,BRG16    ;16 BITS BAUD RATE GENERATOR
        BANKSEL   TX1STA
        BSF   TX1STA,BRGH       ;HIGH SPEED
        BSF   TX1STA,TXEN       ;TRANSMISION HABILITADA
        BCF   TX1STA,SYNC_TX1STA      ;EN MODO ASINCRONO
        BANKSEL   SP1BRGH
        MOVLW     0x00
        MOVWF     SP1BRGH
```

```
                BANKSEL     SP1BRGL
                MOVLW       D'25'
                MOVWF       SP1BRGL         ;N=25 PARA BAUDS=Fosc/
(4*(N+1))=1000000/(4*(25+1))=9600
                BANKSEL     RC1STA
                BSF     RC1STA,SPEN         ;HABILITACIoN DEL PUERTO
SERIAL
;
;-------------------------------------------------------------------------------------------------------
; Programa principal
;-------------------------------------------------------------------------------------------------------
;SENAL DE ARRANQUE:
;
                MOVLW       D'10'   ;CUENTA PARA R3
                MOVWF       R3
TRY:
                MOVLW       D'1'    ;VALOR PARA I2 QUE PRODUCE UN
RETARDO DE 0.1 SEG
                MOVWF       I2      ;SE CARGA PARAMETRO
                CALL RETARDO    ;I2 VECES RETARDO DE 0.1 SEG.
                CALL TOGGLE     ;CONMUTAMOS LED
                DECFSZ  R3,F    ; ES CERO?
                GOTO    TRY
MAIN:
                CALL SENDMSG ;ENVIA MENSAJE POR PUERTO SERIAL
                GOTOMAIN


;-------------------------------------------------------------------------------------------------------
;********************* RUTINAS PARA PUERTO SERIAL *********************
;-------------------------------------------------------------------------------------------------------
;PROFUNDIDAD DE STACK=1
;-------------------------------------------------------------------------------------------------------
;RUTINA SEND(W)
;EN W EL CARACTER A TRANSMITIR
SEND:
        BANKSEL     TX1STA
TRYSND:
        BTFSS       TX1STA,TRMT ;ESTA VACIO EL REGISTRO DE
CORRIMIENTO DE TX (TSR)?
        GOTOTRYSND      ;SI NO ESTA VACIO: VULVE A INTENTAR
        BANKSEL     TXREG
```

```
        MOVWF   TXREG       ;TRANSMITE EL VALOR EN W
        RETURN
;FIN RUTINA SEND
;
;-------------------------------------------------------------------------------
;PROFUNDIDAD DE STACK=2
;-------------------------------------------------------------------------------
;RUTINA SENDMSG
SENDMSG:
        MOVLW   'H'
        CALL SEND
        MOVLW   'O'
        CALL SEND
        MOVLW   'L'
        CALL SEND
        MOVLW   'A'
        CALL SEND
        MOVLW   D'10'
        CALL SEND
        MOVLW   D'13'
        CALL SEND
        RETURN
;FIN RUTINA SENDMSG
;
;-------------------------------------------------------------------------------
;********************** RUTINAS DE RETARDO **************************
;-------------------------------------------------------------------------------
;PROFUNDIDAD DE STACK=1
;-------------------------------------------------------------------------------
;RUTINA RETARDO(I2) CON BASE EN 0.1 SEG (100 X 1mSEG) CON UNA
FREC DE 1 MHz
;EN I2 NUMERO DE ITERACIONES DE 0.1 SEG
;MODIFICA DIRECTAMENTE I1 E INDIRECTAMENTE A I0
;
RETARDO:
LOOP3:
            MOVLW   D'100' ;CUENTA PARA R1
            MOVWF   I1
LOOP2:
            CALL MILIRET
            DECFSZ  I1,F    ; LOOP 2.
```

```
                GOTOLOOP2
                DECFSZ  I2,F     ; LOOP 3.
                GOTOLOOP3
                RETURN
;FIN RUTINA RETARDO
;
;-------------------------------------------------------------------------------------------------
;RUTINA MILIRET() PARA UN RETARDO DE 1 MILISEGUNDO.
;QUE CORRESPONDE A 62*16uSEG = 62*(NOP+DECFSZ+GOTO) CON UNA
FREC DE 1 MHz.
;MODIFICA I0
;
MILIRET:
                MOVLW       D'62'
                MOVWF       I0       ;CARGAMOS CONTADOR
MRETL1:
            NOP
                DECFSZ  I0,F     ; loop 1.
            GOTO    MRETL1
                RETURN
;FIN RUTINA MILIRET
;



;-------------------------------------------------------------------------------------------------
;********************* RUTINAS DE SENALIZACION ************************
;-------------------------------------------------------------------------------------------------
;PROFUNDIDAD DE STACK=1
;-------------------------------------------------------------------------------------------------
;RUTINA PARA CONMUTAR RA4
;
TOGGLE:
                BANKSEL     PORTA
                BTFSS       PORTA,RA4
                GOTOSETBA2
;LIMPIA BIT:
                BCF    PORTA,RA4
                GOTOETOGGLE
;PRENDE BIT:
SETBA2:
                BSF    PORTA,RA4
```

```
ETOGGLE:
            RETURN
;FIN RUTINA TOGGLE
;
;-----------------------------------------------------------------------------------------------------
;-----------------------------------------------------------------------------------------------------
; RUTINAS DE ATENCION A INTERRUPCIONES
;-----------------------------------------------------------------------------------------------------
;
;-----------------------------------------------------------------------------------------------------
;PROFUNDIDAD DE STACK=2
;-----------------------------------------------------------------------------------------------------
;RUTINA INTSERV() SERVICIO A INTERRUPCION
;
INTSERV:
            BANKSEL     PIR0
            BTFSS       PIR0,TMR0IF   ;ES INTERUPCION de TIMR0?
            GOTOEINTSR          ;SI NO ES INTERUPCION TIMR0, SALIR
            BCF   PIR0,TMR0IF   ;LIMPIA FUENTE DE INTERRUPCION
;CONMUTA LED:
            CALL  TOGGLE
EINTSR:

            RETFIE
;FIN RUTINA INTSERV
;
;
DR          DW    'G','A','L','S'
VER         DW    'V','e','r','0','.','0'
FIN         DW    'F','I','N'
;-------------------------------------------------------

        END
```

Casa abierta al tiempo
UNIVERSIDAD AUTÓNOMA METROPOLITANA
Unidad Lerma

Dirección
Ciencias Básicas
e Ingeniería