

Laboratorio 2

# Sistemas Embebidos

Equipo 10

Integrantes del equipo:

Patricio Zarauz

Sebastián Reynosa

Gastón Salustio

Primera Parte

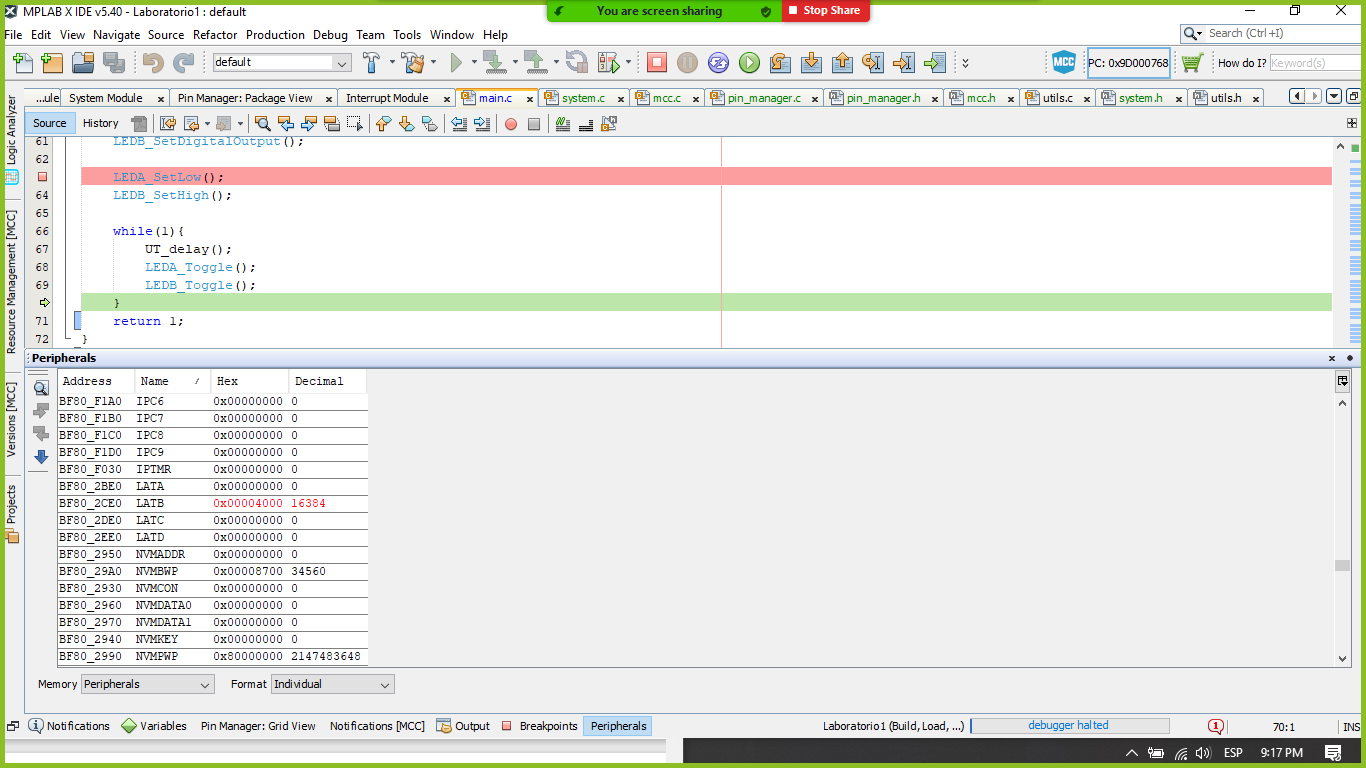
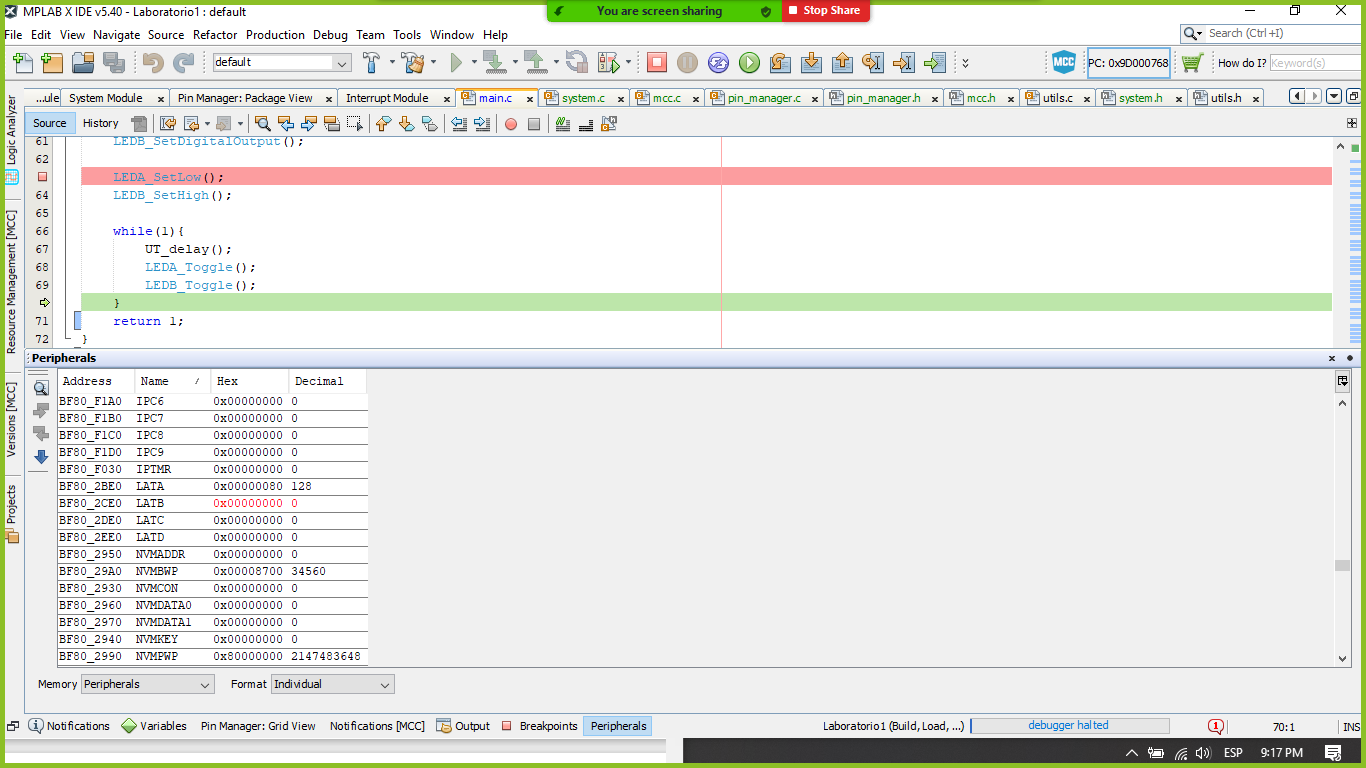
1.g

El archivo pin\_manager.h se puede relacionar con el archivo HardwareProfile.h debido a que en el mismo se declaran y definen las funciones relacionadas a los LEDs.

Como en el pin\_manager.h las funciones tienen los mismos nombres que en HardwareProfile.h, este archivo ya no será necesario incluirlo en el proyecto.

Con respecto a los otros archivos generados en el MCC, estos corresponden a las configuraciones que previamente se realizaban en el main.c, antes de iniciar el programa.

1.h

2.a

Debemos configurar los botones, que se encuentran en los pines RB15 y RA13 (S2 y S3 respectivamente). Los mismos se deben configurar como pines de entrada.

2.b

Por lo visto en el circuito esquemático de la placa de desarrollo, se necesita una resistencia de pull-down para que coincida con el simulador, ya que se verificó, que cuando no está presionado el botón hay una entrada lógica de 0.

2.d

Los macros que se generar son las siguientes:

BTNX\_SetHigh: Para darle el valor de 1 al botón X

BTNX\_SetLow: Para darle el valor de 0 al botón X

BTNX\_Toggle: Cambiar el valor actual del botón X.

BTNX\_GetValue: Indica el valor del botón X.

BTNX\_SetDigitalInput: Configura al botón X como pin de entrada

BTNX\_SetDigitalOutput: Configura al botón X como pin de salida

Mientras que la función que se genera es BTNX\_SetValue. La misma permite setear el valor deseado a un botón, dependiendo del valor booleano pasado por parámetro.

Segunda Parte

1.c y d

Se creó el archivo interrupt\_manager (.c y .h) en el cual se inicializan los interrupts deseados y se habilitan o deshabilitan los interrupts del sistema.

Las funciones generadas fueron las siguientes:

BTNX\_SetInterruptHandler, nos permite asignar un puntero de una función con una dirección de callback al botón deseado.

BTNX\_SetIOCInterruptHandler, nos permite asignar un puntero de una función con una direccion de callback al boton deseado.

\_CHANGE\_NOTICE\_X, realiza el servicio de rutina del interrupt para el registro CNXI (X depende del change notice del vector deseado). Aca se resetean la flag del interrupt después de haberse ejecutado.

Los registros configurados para las interrupciones de los pines de los botones son:

En el registro INTCON se seteo en 1 el pin MVEC, habilitando la configuración de multi vector.

IPC2, en este registro se setean las prioridades de las notificaciones de cambio de estado de los puertos.

Otros registros configurados son CNEN0A, CNEN0B, CNFA, CNFB, CNCONA, CNCONB, los cuales sirven para configurar el interrupt cuando se detecte un cambio de estado en los pines y puertos deseados.

Los registros IFS0 y IEC0 permiten setear el valor de los interrupt flags e interrupt enable, respectivamente.

2.a

Las funciones creadas fueron:

TMR2\_Initialize: Inicializa el valor del timer

TMR2\_Period16BitSet: Define el valor de período del timer

TMR2\_Period16BitGet: Retorna el valor de período del timer

TMR2\_Counter16BitSet: Setea el valor del contador

TMR2\_Counter16BitGet: Retorna el valor del contador

TMR2\_SetInterruptHandler: Asigna un puntero a la función a ejecutar cuando ocurra la interrupción.

TMR2\_Start: Empieza el timer 2

TMR2\_Stop: Termina el timer 2

TMR2\_GetElapsedThenClear: Retorna el estado de si el timer término o no (elapsed status) y resetea la bandera si está seteada.

TMR2\_SoftwareCounterGet: Retorna el valor del contador de software.

TMR2\_SoftwareCounterClear: Resetea en 0 el valor del contador de software.