:: PIC - Parte II - Índice

No haré ninguna introducción para no aburrirlos de tanto bla, bla, bla, comienza de una vez y deja de estar perdiendo tiempo...

Primeros contactos

Descripción de la barra de menú

Creando un proyecto con MPLAB

El código fuente de CQPIC en varias partes

Simulando CQPIC con MPLAB

Nota

Quiero agradecer el apoyo de todos mis lectores, la verdad que dan ganas de continuar, así es que nos veremos en la próxima actualización, y aquellos que deseen colaborar con la página, serán bienvenidos...

Saludos para todos...!!!

R-Luis

:: Microcontroladores PIC - Segunda Parte - MPLAB

MPLAB

Bienvenido a esta sección, como ves... aquí estamos nuevamente, y esta vez decididos a liarnos un poquito más, y como lo prometido es deuda, volví para presentarles "El Entorno Integrado de Desarrollo" MPLAB, la mejor herramienta para desarrollar proyectos con PIC's, bue... según yo...!!!

El tema es, ya tienes MPLAB...???. Si no es así pues bye, que esto no te servirá de nada, y si quieres continuar, ingresa aquí, lo descargas lo instalas y ya...!!!

Instalación y Configuración

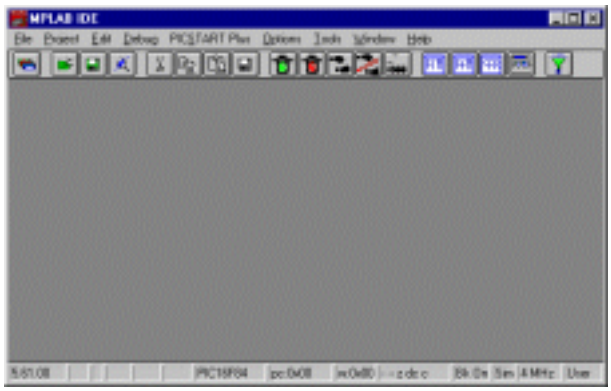
Alguien dijo por ahí que cuando lo instales, le saques los tildes a todo lo referente a PICMASTER, PROMATE, MPLAB-ICE, MPLAB-ICD, ICEPIC, dado que se refieren a herramientas hard de microchip, y que nosotros no tenemos, pero bueno, yo me enteré tarde :oP

Algunas de las cosas más importantes serían:

- Archivos MPLAB IDE
- Archivos MPASM/MPLINK/MPLIB
- Archivos de Protección del Simulador
- MPLAB-SIM

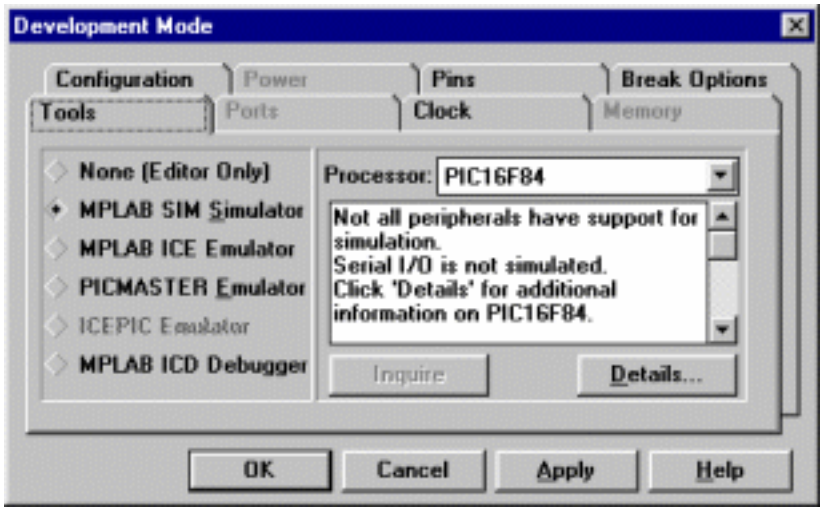
y por supuesto, los Archivos de Ayuda. Luego seleccionas los componentes de lenguaje Microchip que debes instalar (Todos por defecto), bien termines la instalación, lo ejecutas y veras el IDE de MPLAB.

Por las dudas surja algún cuadro de diálogo diciéndote algo de un proyecto, que si quieres crear uno o que no existe ninguno o bueno algo así, pues le das a que no y continúas, y no te preocupes demasiado que luego hablaremos de ese tema, por ahora haremos una inspección de nuestro entorno de trabajo luego entramos en materia...



Ahora lo vamos a preparar para nuestra ardua tarea de programar y simular nuestros proyectos según el Micro del que disponemos, es decir, mi querido y bien amado 16F84...

Te diriges al menú **Options --> Development Mode...** y verás un cuadro de diálogo algo así...



En la pestaña **Tools** seleccionas el procesador **PIC16F84** y del otro lado el modo en que lo vas a utilizar **MPLAB-SIM Simulador** eso te permitirá programar, depurar, simular, etc., el resto lo dejas como está, finalmente le das a **Ok** y verás en la barra de estado los cambios que hiciste...

Hasta aquí estamos listos para comenzar a trabajar, así es que daremos inicio a este tutorial...

Que Desperdicio de Hoja...!!!

:: PIC - Parte II - Capítulo 1

Barra de Herramientas y Menús:

Me imagino que estarás igual que yo cuando lo tuve por primera vez en mis manos, desesperado por comenzar a utilizarlo sin tener la más pálida idea de lo que se trataba, ni nadie que me lo indicara, más que un par de tutoriales que hicieron pié para ese gran desafío, va...!!!, no fue tan crucial así que a ponerle ganas...!!!

Comencemos por la barra de herramientas...

MPLAB pone a tu disposición 4 barras de herramientas personalizables según la tarea que estés realizando, pues no voy a describirlas, ya que puedes acceder a cada una de las opciones desde cualquiera de los menús, los cuales si describiré, pero veamos cuales son esas barras...

La primera es la barra USER, que por defecto es la única que utilizo, bueno tu elegirás la que mas te agrade. La segunda es la barra EDIT, todo lo relacionado a la Edición de texto. La tercera es la barra DEBUG, con todo lo necesario para simular nuestro programa, depurarlo, etc. Finalmente la barra PROJECT, todo lo referido al proyecto que estás realizando

Por cierto puedes cambiar de barra de herramientas simplemente presionando repetidas veces el primer botón, si colocas el cursor del mouse sobre este icono verás en la parte inferior de la ventana la función que cumple, en este caso... "Swap Toolbar" en entendible "Cambiar barra de herramientas", pues es como una ayuda emergente coloca el cursor sobre otros botones y sabrás para que se utilizan...

Ok, vamos por los menús, por ahora solo los más utilizados...

Menú File:

NEW: Crea un nuevo archivo con extensión .asm
OPEN: Abre un archivo que puede ser

.asm/.obj/.c/.err/.h/.lkr/.map/.hex/.lst
VIEW: Abre un archivo en modo solo lectura.

y lo más común...

SAVE (Guardar), **SAVE AS** (guardar como), **SAVE ALL** (guardar todo), **CLOSE** (cerrar un proyecto), **CLOSE ALL** (cerrar todos los proyectos abiertos).

Respecto a **IMPORT** y **EXPORT**, no los utilizaremos por ahora.

PRINT (Imprime), y **PRINT SETUP** te permite configurar el modo de impresión, **EXIT** cierra MPLAB, y finalmente los 5 últimos archivos utilizados.

Menú Project:

Como MPLAB te permite trabajar con varios archivos y todos los elementos y herramientas a la vez es bueno crear un proyecto y tener en él todo lo necesario para realizar tu tarea, como ser; codificar, ensamblar, simular, etc. Pronto veremos como crear un proyecto o comienza ahora si lo deseas, y deja esto para otro momento...

El resto que me acompañe..., veamos que contiene este menu...

NEW PROJECT: Abre un cuadro de diálogo para crear un nuevo proyecto y en él todo lo que necesites para tenerlo más a tu alcance.
OPEN PROJECT: Abrir un proyecto
CLOSE PROJECT: Cerrar un proyecto
SAVE PROYECT: Guardar Proyecto
EDIT PROJECT: similar a New Project, pero referido a un proyecto ya abierto, al que se le puede agregar nodos, librerías y la forma en que desarrollaras el proyecto en cuestión.
MAKE PROJECT: Según los que de todo saben dicen que ensamblara todo, pero teniendo en cuenta la fecha de creación del archivo .HEX la cual compara con el archivo .asm del que ensamblo. Si este tiene una fecha mas reciente que el archivo .HEX asociado, debido digamos a una actualización del código, entonces ensamblará nuevamente el proyecto. En caso de que la fecha sea anterior al archivo .HEX generado (es decir, archivo .ASM mas viejo que el .HEX) este no hará nada.(extraido textualmente de no recuerdo donde, pero igual, yo jamás lo utilicé)
BUILD ALL: Le importa nada la fecha y ensambla todo. (este si, y de tiempo completo...!!!)
BUIL NODE: Ensamblar un nodo que hayamos asociado al proyecto.
INSTALL LANGUAGE TOOL: Es para configurar el lenguaje a utilizar, que en nuestro caso es el ASM de Microchip.

Y por ultimo el acceso directo a los últimos proyecto abiertos.

Menú Edit:

Quizás a algunos les parezca grosero que incluya algunas de estas cosas, pero que va aquí se las mando...

UNDO: Deshacer, y en ese orden Cortar, Copiar y Pegar
SELECT ALL: seleccionar todo
SELECT WORD: selecciona la palabra sobre la que esta el cursor.
DELETE LINE: borra la línea sobre la que esta el cursor.
DELETE EOL: borra desde donde esta el cursor hasta el final de la línea.
GOTO LINE: Abre un cuadro de diálogo para saltar a una determinada línea (Line to go to), te indica además la cantidad total de líneas (Last Line) y la línea en que tienes el cursor (Current Line), es algo así como el "Ir a" de Word.
FIND: es para buscar algún texto dentro de nuestro código, tiene algunas opciones, pruébalas...
REPLACE es para reemplazar un texto o una pequeña frace.
REPEAT FIND: es para repetir la búsqueda si es que hay otra palabra o frace igual.
REPEAT REPLACE es para repetir el reemplazo.
MATCH BRACE Vaya Dios a saber para que es, yo nunca lo utilice, ...si alguien lo sabe, me cuenta...?.
TEMPLATE: Pues como todos los templates, para crear unos archivos personalizados o plantillas, y así no estas reescribiendo todo un código nuevamente.
TEXT: Tambien tiene sus opciones; **Transpose** intercambia los caracteres que están a ambos lados del cursor; **Upper case** cambiar a mayúsculas; **Lower case** cambiar a minúsculas; **Indent** mantiene la tabulación de la línea anterior; **Un-Indent** lo opuesto.

Esta parte está muy extensa, así que vamos a la próxima página...

:: PIC - Parte II - Capítulo 2

Menú Debug:

De lo más importante, desde aquí haremos las simulaciones de nuestro programa, los tres primeros submenús son los de mayor interés, veamos de que se trata cada uno de ellos...

RUN: Aquí tenemos varias opciones, a por ellas...

- **RUN:** Como su nombre lo indica, Runnnnnn... inicia la simulación a una velocidad que la verdad, no se distingue nada de lo que ocurre a lo largo del código, verás la parte inferior de la ventana toda pintada de amarillo (señal de que la simulación está en proceso). Útil cuando colocas algunos puntos de ruptura (breakpoints) para detener la ejecución y así no estar esperando que se ejecute todo aquello que sabes que esta correcto.
- **RESET:** Nada, Resetea el micro y te ubica en la primer línea donde comenzará la simulación (en ese caso verás toda la línea pintada de negro y las letras blancas) "listo para comenzar la simulación"
- **HALT:** Detiene la ejecución.
- **HALT TRACE:** Detiene un traceo que se este haciendo (yo aun no lo utilicé).
- **ANIMATE:** es igual que RUN pero lo hace más lento, para que vayas siguiendo la ejecución, mostrándote línea por línea todo lo que se está ejecutando, es el que más utilizo yo.
- **STEP:** paso, es decir, un paso por cada vez que lo presionas (en la barra de herramientas verás dos huellas de zapato, pues es eso, paso a paso). Simplemente ejecuta una a una cada línea de código por cada vez que lo presionas.
- **STEP OVER:** Igual que el anterior pero con un pequeño detalle, cada vez que se encuentre con un call lo ejecuta de modo tan rápido que ni tu te enteras de que ya pasó, es decir, utilízalo si no quieres perder tiempo con el call.
- **UPDATE ALL REGISTER:** Actualiza el valor de todos los registros.
- **CHANGE PROGRAM COUNTER:** Carga el Programa Counter Strike para jugar un rato, jajajajaja, es broma. La verdad es que cambia el valor del PC y te ubica en la etiqueta del código a la que quieres ir.

EXECUTE: Bueno, execute tiene dos opciones, veamos...

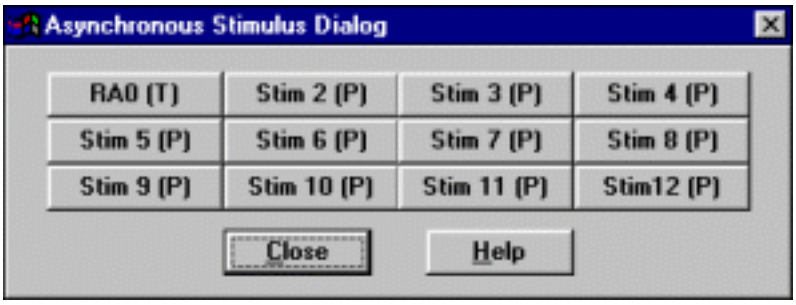
- **EXECUTE AN OPCODE:** te permite ejecutar un código cualquiera desde cualquier sitio en el que te encuentras, por ejemplo ingresas

un **goto inicio** y hacia allí irá. independiente a la secuencia del programa.

- **CONDITIONAL BREAK:** Para ejecutar un breakpoint en forma condicional, Por ejemplo por que cambió el valor de un registro, o por que quieres ver que paso hasta ese punto, o bueno, en fin. (por si las moscas, para quien no tenga idea de que es un Breakpoint, es un punto de ruptura en el código del programa, es decir, si se está ejecutando..., cuando se encuentre con el breakpoint se detendrá y allí se quedará hasta que le des la orden de continuar), habré sido claro...???

SIMULATOR STIMULUS: Con este nos vamos a lucir, desde aquí podrás simular que le envías señales al micro por uno de sus pines, este submenú tiene 4 opciones...

- **ASYNCHRONOUS STIMULUS:** te abrirá un pequeño diálogo con 12 botones como el que se muestra en la imagen, vendrían a ser como interruptores, nota que al primero le asigné el pin RA0 (pin 17) y entre paréntesis una T (de TOGGLE)

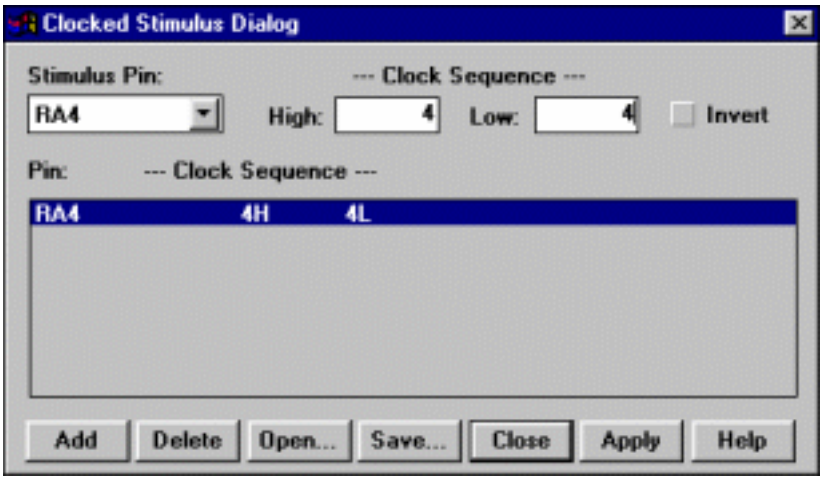


Si presionas con el botón derecho del mouse en uno de esos botones veras las opciones de configuración de estos estímulos desde donde los puedes configurar, es algo así...

- **ASSIGN PIN:** muestra los pines del puerto para que selecciones aquel que llevará un estímulo.
- **PULSE:** un pulso, hasta ahora no hice uso de el...
- **LOW:** Le asigna un nivel bajo permanente
- **HIGH:** Lo contrario, nivel alto permanente

    - **TOGGLE:** Cada vez que lo pulses cambiará de nivel de alto a bajo o de bajo a alto.
    - **HELP:** tampoco lo utilicé así que no preguntes...!!!

- **PIN STIMULUS:** Es para cuando creas un archivo de texto con extensión .sti, desde aquí lo cargas o lo quitas, se trata de utilizar un archivo que creaste con esta extensión y que contiene los estímulos ya asignados para cada uno de los pines.
- **CLOCK STIMULUS:** Desde aquí puedes enviarle pulsos de reloj a un determinado pin, indicando el tiempo que se mantendrá en nivel alto y el tiempo en nivel bajo, tipo (timer).



Tiene un par de opciones como ser; Guardarlo como un archivo .sti, Abrir alguno si es que ya lo tienes creado, Aplicar los cambios, agregarle mas impulsos a otros pines, y el help (por supuesto, la ayuda).

- **REGISTER STIMULUS:** Es exactamente lo mismo que con los pines, solo que llava la extensión .reg y sirve para que en una determinada posición del programa se cargue un registro con el valor que tu le quieras dar.

CENTER DEBUG LOCATION: te ubica en la posición en la cual el programa se está ejecutando, por si perdiste la línea de ejecución, se suele utilizar cuando detienes la ejecución, empiezas a husmear por otro lado y luego no sabes donde andabas, pues bueno ahí te lo acercas...

BREAK SETTINGS, Te muestra un diálogo para que le des nombre a los breakpoints y luego desde aquí los habilitas, los quitas, los guardas etc.

TRIGGER IN/OUT SETTINGS: Son para los emuladores MPLABICE y PICMASTER, aquellos que dijimos no tener...

TRIGGER OUTPUT POINTS: Para cuando consigas PICMASTER.

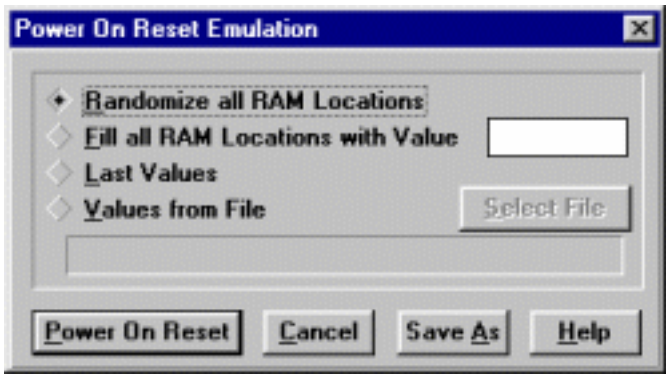
CLEAR ALL POINT limpia todos los breakpoints y los tracepoints que pusiste a lo largo del programa.

COMPLEX TRIGGER SETTING y **ENABLE CODE COVERAGE** Para cuando consigas MPLAB-ICE, jeje...!!!

CLEAR PROGRAM MEMORY: Borra la memoria de programa, yo nunca lo utilicé, pero según leí por ahí... Cuando simulamos un programa, lo que la maquina en realidad hace, es seguir a travez de la memoria de programa. Luego desde allí mismo, puedes grabar el PIC si es que cuentas con el programador PICSTART PLUS, así es que ahí lo dejé...

SYSTEM RESET: Eso mismo, Resetea el sistema.

POWER ON RESET: Para el caso en que se podría producir un reset en el micro, y así saber que ocurre o que es lo que hace el micro si esto llegara a ocurrir.



Sólo le das a Power On Reset y habrás ocasionado un reset en el pin4 del micro (MCLR), luego le das a "cancel" y continuas con la ejecución para saber que hace el pic, y así asegurarte de que todo está en orden...

Nos queda lo último, así que no me aflojes ahora, que luego viene lo mejor...

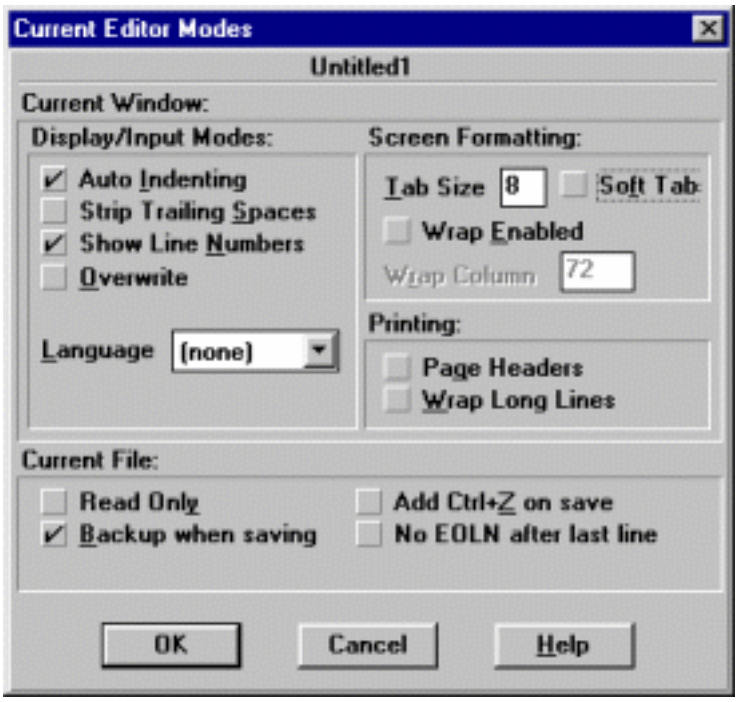
:: PIC - Parte II - Capítulo 3

Menú Options:

DEVELOPMENT MODE: ya hablamos de el en la introducción a este turorial, simplemente es donde hacemos las configuraciones del proyecto.

WINDOW SETUP: como estamos en la sección de personalización, puedes personalizar totalmente la forma en que vas a trabajar, tus preferencias, luego Guardarlas (Save Setup), Cargarlas (Load Setup) si es que ya las tienes guardadas, o utilizar una por defecto (Default Configuration).

CURRENT EDITOR MODES: Es para que personalices el modo de edición, por ejemplo las tabulaciones de 8 espacios, que se mantenga la indentación de la línea anterior, que se muestre el número de líneas al margen del código, hacer un backup cada cierto tiempo (por si olvidas que de vez en cuando es bueno guardaaaaar...), etc. aquí tienes una imagen de esas opciones...



RESET EDITOR MODES: Elimina los cambios que hiciste anteriormente
ENVIRONMENT SETUP: Wowwww...!!! verás un inmenso cuadro de diálogo para una personalización completa.
PROGRAMMER OPTIONS: Son las opciones para configurar el programador que utilizarás con MPLAB, pero como no lo utilizaremos para cargar el pic ya que no disponemos de esas herramientas, ahí queda...

Menú Tools:

Personalmente, nunca lo utilice y no tengo la más pálida idea de para que sirve, como dije anteriormente, si alguien lo sabe, me lo cuenta y luego lo incluimos aquí, si es que fuera necesario por supuesto...!!!

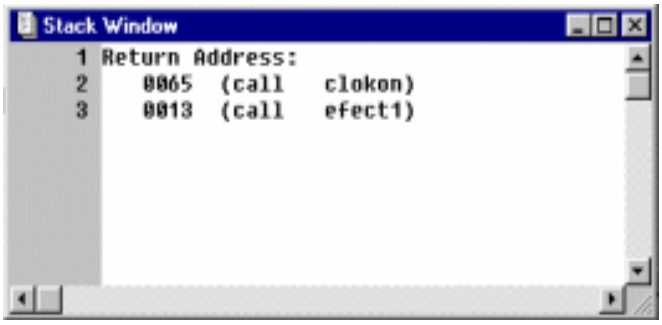
Menú Window:

El más importante de todos, ya que desde aquí nos veremos cara a cara con cada uno de los bits de los registros del micro, el estado que tienen y como se modifican, claro que lo verás cuando lo ejecutes en modo "step" (paso a paso) o "Animate" (de forma animada)...

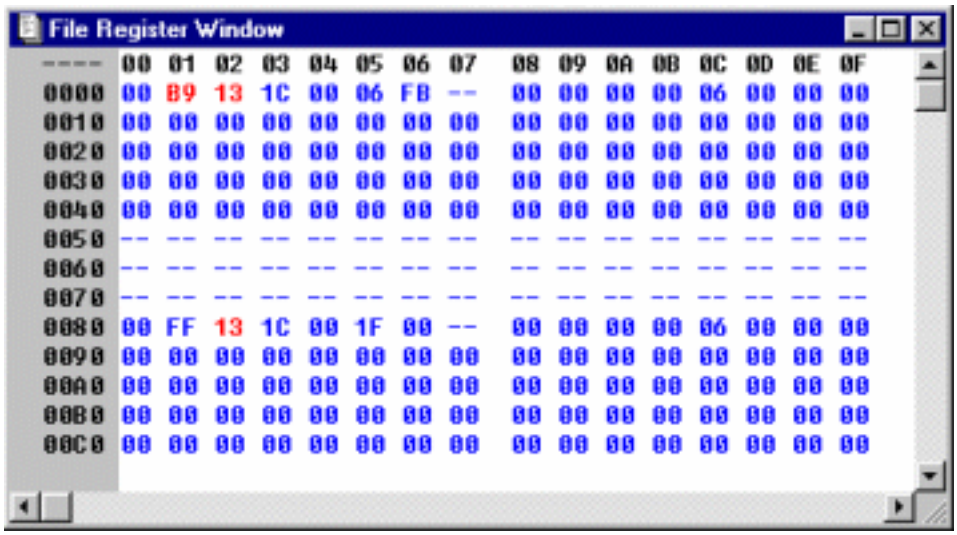
PROGRAM MEMORY: Verás la memoria de programa, y así sabes cuanto espacio te queda disponible, no te preocupes que un ejemplo lo aclarará todo, seguimos...?

Unos cuantos que no mencionaré, y pasemos a lo que mas me interesa

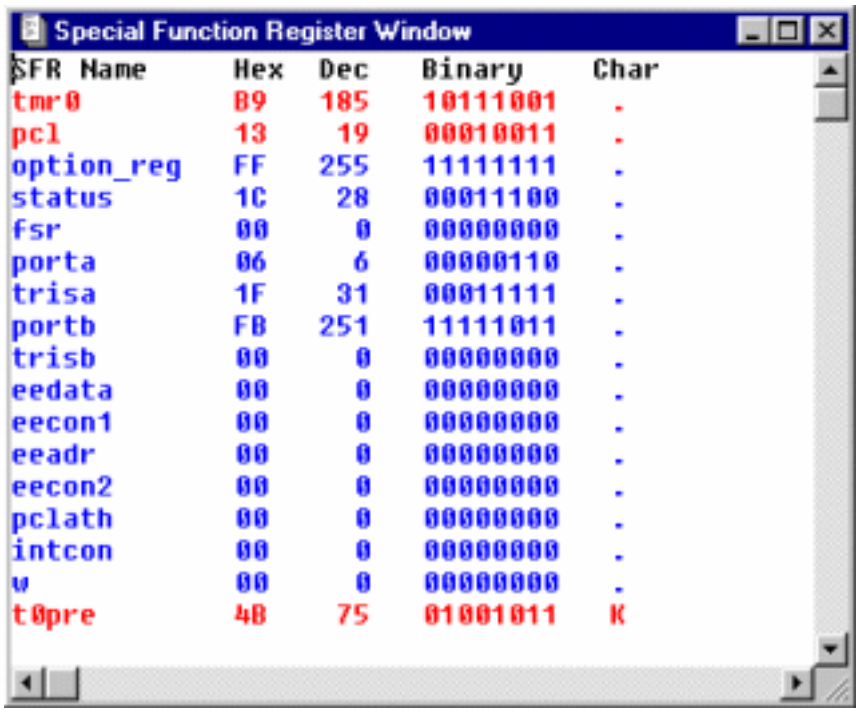
STACK: La pila, recuerdas aquello de la pila de platos, en nuestro primer tutorial, bueno aquí la tienes en vivo, recuerda que son de 8 posiciones y la ultima en ingresar es la primera en salir, aquí la imagen de uno que estoy simulando en este momento...



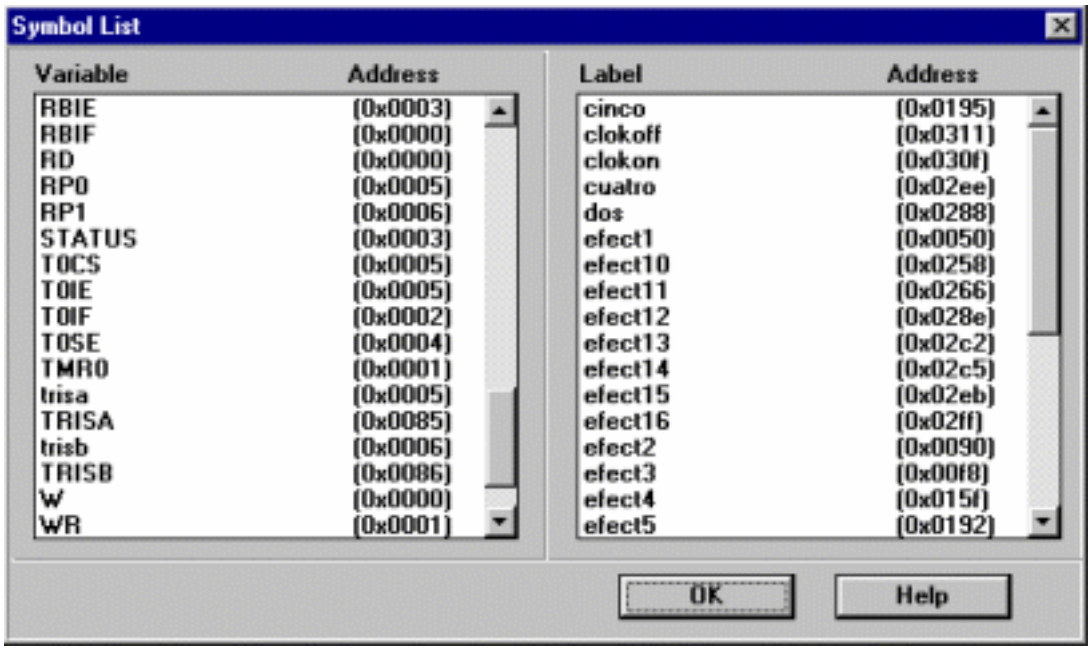
FILE REGISTER: Para que veas como se modifican los registros mientras el programa se está ejecutando, solo en modo "Step" o "Animate".



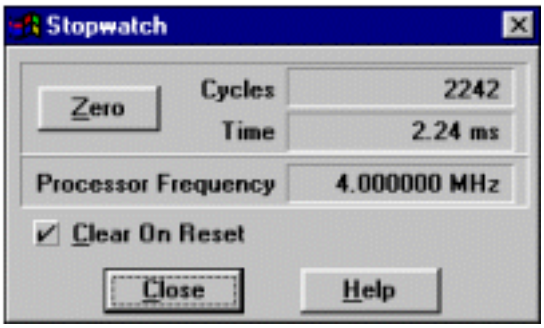
SPECIAL FUNCTION REGISTER: Los Registros de funciones especiales, que más dá, con nombre y todo, trisa/b, porta/b, status, w, creo que lo que mas quieres es ver como se modifican los bits de entrada y salida de los puertos pues aquí los tienes, como antes, sólo los verás cuando lo ejecutas en modo "Step" o "Animate".



SHOW SIMBOL LIST: todos los símbolos utilizados y su respectiva ubicación (dirección de registro), aquello que definimos al iniciar nuestro código, como "status equ 0x03". Del otro lado las etiquetas utilizadas a lo largo del programa, en fin...



STOPWATCH: Para que veas el tiempo en milisegundo consumidos por el micro que cuenta con un XT de 4 MHz...



PROJECT: Los datos del proyecto que estas realizando
WATCH WINDOW: Para crear tu propia ventana de registros, Cuando encaremos un proyecto que la requiera hablaremos de ella...
MODIFY: Para modificar los valores de algún registro determinado, en caso de que no quieras esperar demasiado por ejemplo cuando haces un retardo, pronto se aclararan tus dudas, obvio si practicas...!!! :oP

El resto es de poco interés, asi es que... ni para que mencionarlos

Terminamooooooooosssssss, al fiiiiinnn..., no sientes una gran satisfacción después de tanto lío...???

En fin... En la siguiente página comenzaremos un proyecto desde CERO y que proyecto...!!!

Ahí nos vemos...

Otra hoja al cuete...!!!

:: PIC - Parte II - Capítulo 4

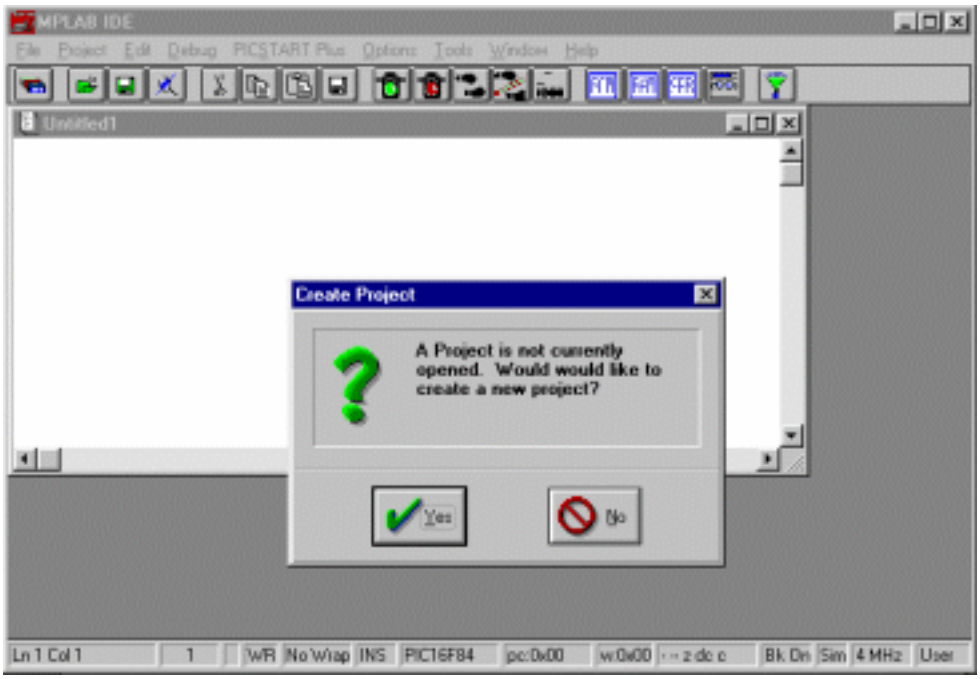
Cómo crear un proyecto con MPLAB:

Bueno, Lo que voy a describir en este apartado es aplicable en general para cualquier proyecto que quieras realizar, por lo que cada uno de los procedimientos serán siempre los mismos.

Comencemos...

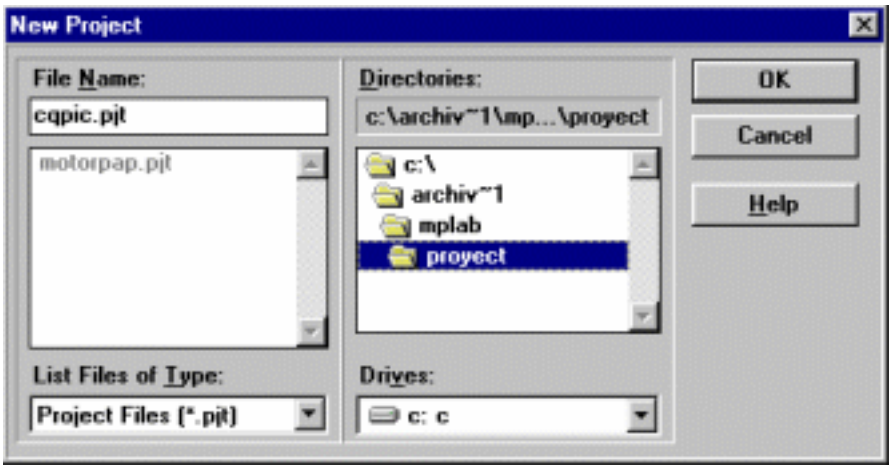
Lo primero es abrir MPLAB, si es que no lo tienes abierto ya...!!!

Selecciona el menú **File-->New** y verás un diálogo como el siguiente...

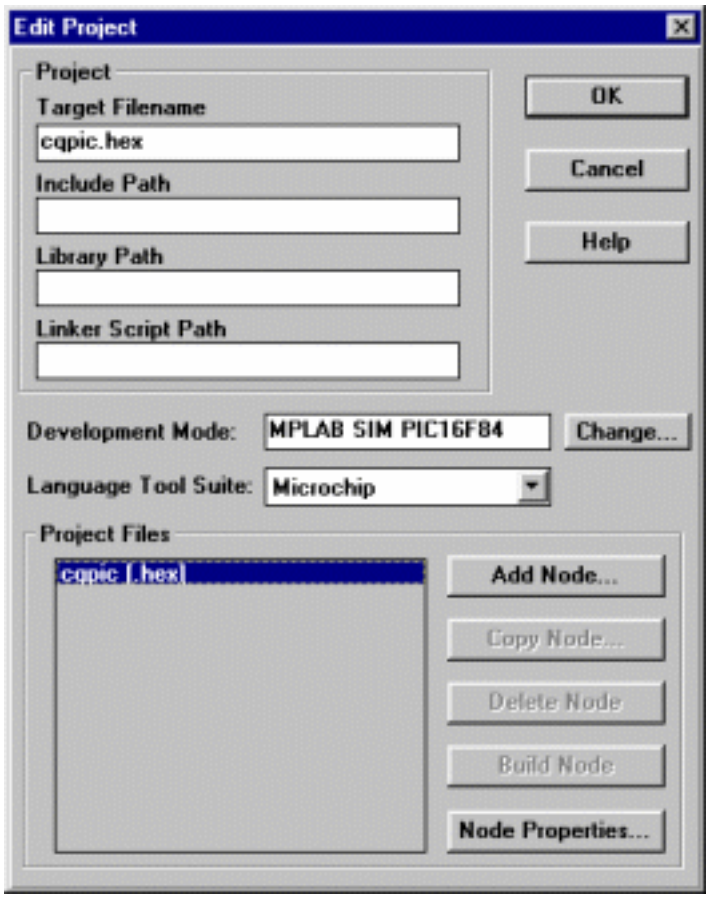


Que según mi inglés dice algo así... "No existe ningún proyecto abierto, desea crear uno?", y como en eso andamos seleccionas "YES", nota que también se abrió una página en blanco "Untitled1", bueno, en ella es que introduciremos nuestro código, pero sigamos...

Luego de darle a Yes, verás una nueva ventana "New Project" en la que nos pide el nombre del proyecto y el directorio en que lo vamos a guardar, de nombre le puse "cqpic.pjt" (.pjt es la extención de project) y lo guardaré en una carpeta que tengo por ahí llamada proyect justamente.



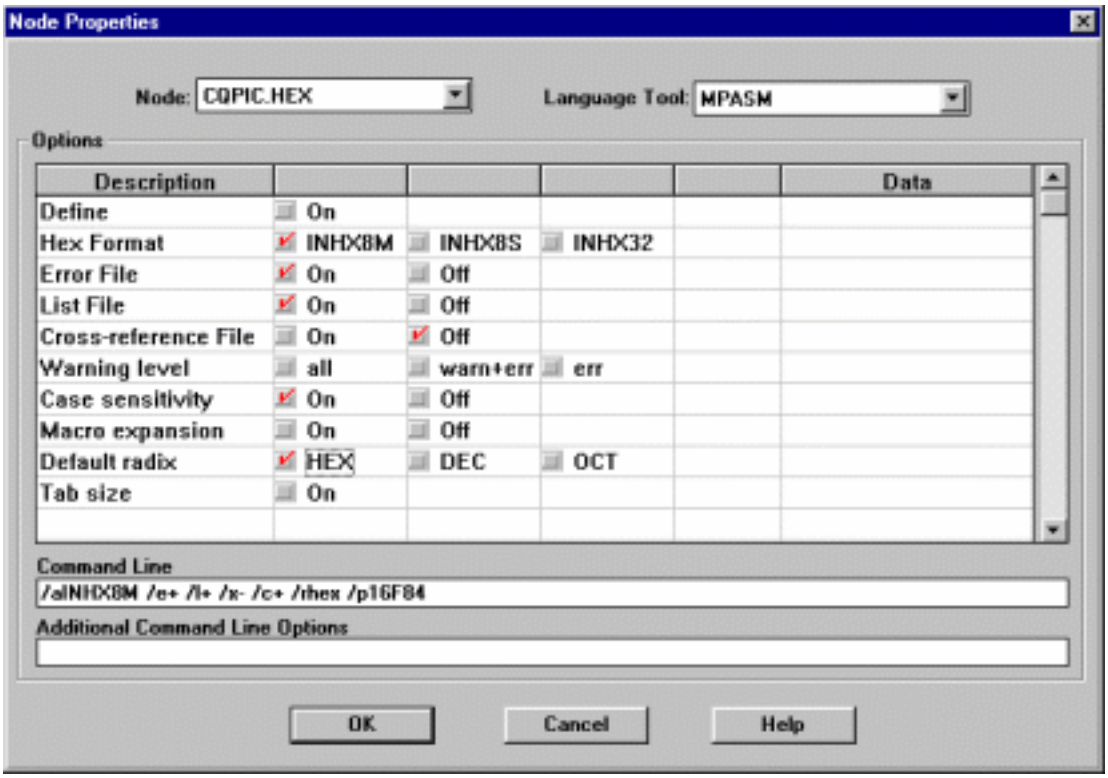
Pues bien, seleccionas **OK** y comenzaremos a crear nuestro proyecto desde la ventana **Edit Project** (Edición de Proyecto).



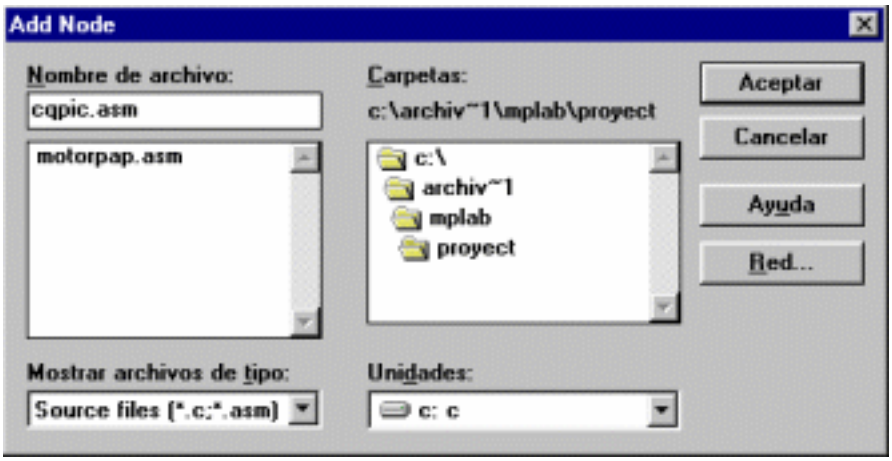
Aquí es donde comenzamos a ajustar todos los detalles para nuestro proyecto, y si estuviste siguiendo el tutorial desde un principio, te habrás dado cuenta que hay algunas cosas que ya nos son familiares, como... "Development Mode" al cual accedimos alguna vez desde el menú "Options" y nos muestra el micro para el que estamos trabajando, "Language Tool Suite" el lenguage utilizado por Microchip.

Un detalle a tener en cuenta es que MPLAB ya le asignó la extensión .hex al proyecto que estamos creando, el cual por defecto lleva el mismo nombre, de hecho la finalidad es esa, crear un archivo con extensión .hex para luego grabarla en la memoria del pic.

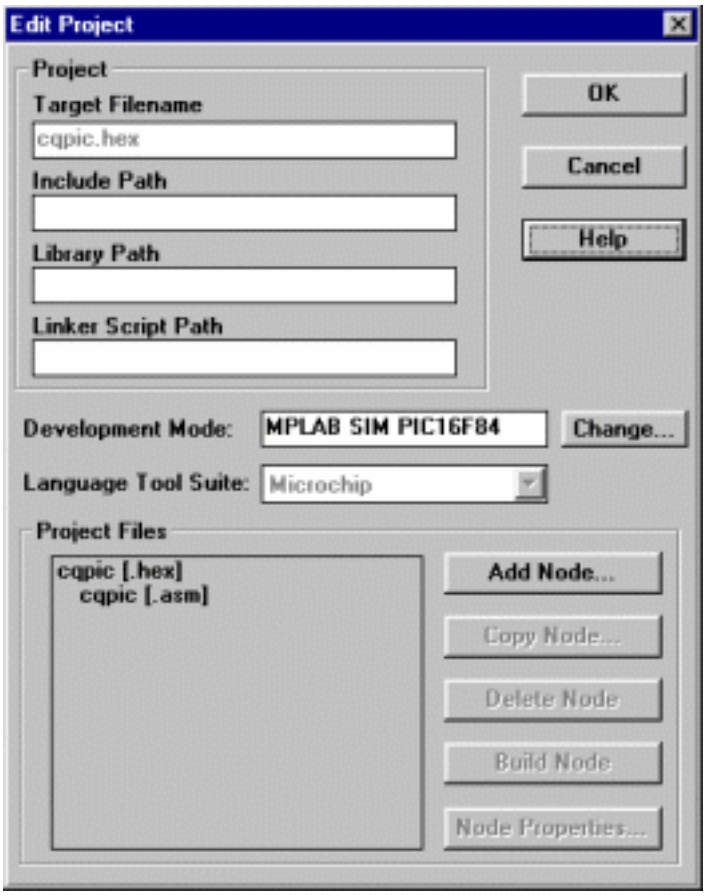
Si seleccionamos **cqpic[.hex]** en **Projects Files** (parte inferior de la ventana), podrás ver que se nos habilita el botón **Node Properties**. como se ve en la imagen anterior, haces un click en el, y verás la ventana desde la cual configuraremos la forma en que MPLAB debe generar el archivo .hex, te debería quedar algo así...



Como verás aquí también hay algunas cositas ya conocidas, como INHX8M (nuestro código objeto con extensión .hex), el fichero de errores (.err), el archivo list (.lst), Default Radix que vendría a ser el formato de los números que vamos a utilizar (hexadecimal), etc., bien, ahora presionamos **OK** y volvemos a la ventana anterior, seleccionas **Add Node** y verás el siguiente diálogo...



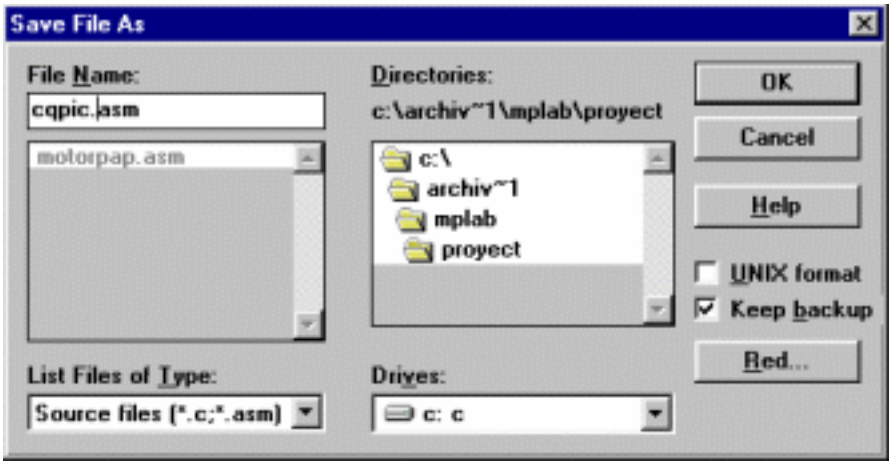
Desde aquí vamos a ligar al proyecto nuestro archivo .asm, que es el que codificaremos, y que todavía no hemos creado, de todos modos sabemos que se llamará **cqpic.asm** otra cosa que debes tener en cuenta es que deberás guardarlo en el mismo directorio en que creaste el proyecto, luego presionas **Aceptar** y regresarás al cuadro **Edit Project** pero de la mano de nuestro **cqpic[.asm]** observa...



Listo, ya está, ahora presionas **OK** y regresas al IDE de MPLAB en el cual tienes esa página en blanco para ingresar el código, el tema es que aún

sigue siendo **Untitled1**.

Veamos... Anteriormente solo creamos un proyecto llamado cqpic.pjt, luego le ligamos un nodo con cqpic.asm (el cual aún no existe) e incluso le dijimos donde lo guardaremos, y nos pusimos de acuerdo con MPLAB la forma en que queremos crear el archivo .hex, pues bueno lo que estamos haciendo ahora, es crear cqpic.asm, así es que te diriges a **File-->Save As** y lo guardas como cqpic.asm.



Ahora sí, ya estamos completos...

Supongo que pensarás lo mismo que yo en mis inicios, "crear un proyecto es toda una azaña", pero bueno, créeme que lo vale...!!!

Bien mis queridos amigos, debo felicitarlos por haber llegado a este punto, la verdad..., que aguante no...!!!

Bueno, lo que viene ahora..., que vá, ve a la siguiente página y lo descubres... :o))

:: PIC - Parte II - Capítulo 5

Antes de continuar con el tutorial nos vamos a tomar un descanso para aclarar un par de cosas, primero necesito un proyecto para que pongas en práctica todo lo que te ofrece MPLAB, así es que hablemos un poco de eso, el proyecto...

CQPIC:

Me imagino que debes estar ya cansado de leer CQPIC, sin saber que demonios significa eso, hice lo que todos, traté de darle al proyecto un nombre que me sea orientativo, se trata de un Secuenciador con PIC, como verás soy un apasionado de esos aparatejos, y aquí en jujuy competimos a gran escala por ellos en épocas de primavera "Fiesta Nacional de los Estudiantes", que nunca escuchaste hablar de eso...???, si quieres saberlo entra a la página oficial www.fne.org.ar y verás a que me dedico todos los años en esta época.

Tenía pensado hacer un proyecto sencillo para que te guíes mejor en el uso de MPLAB, pero luego lo pensé, y me dije un proyecto sencillo no te lleva a descubrir mucho de MPLAB, así que decidí hacer algo mas complejo, no me arrojes piedras por eso ok?, de todos modos trataré de detallarlo lo mejor posible así evacuamos todas las dudas, de acuerdo...?

Siempre que buscaba secuenciadores no me gustaban ya que eran muy comunes, hasta que me di de narices con éste, y dije "Wooooooooowwwww... un secuenciador de 8 salidas y 16 efectos...!!!", cada uno de los efectos se seleccionan con la combinación de 4 interruptores, bueno, como verás no es originalmente mío, y está basado en un proyecto de Jaime Bosch, este fue publicado en la edición nº 21 de la revista "Electrónica Práctica actual" en septiembre del 2000. La verdad JAIME te las mereces todas...!!!, un trabajo espectacular, si andas por estos rumbos, recibe mi mas cordial saludo y felicitaciones por ese gran trabajo.

"Armarlo por primera vez fue toda una anécdota que quizás un día se las cuente..."

Aunque aquella publicación no traía el código fuente, me tocó estudiar el .hex (pero lo abrí con I-CProg ;oP)y hasta el día de hoy no encuentro nada mejor, y lo modifico las veces que se me antoja o a pedido de mis alumnos.

Ok, Fin de la introducción, para que te des una idea de lo que vamos a programar, hice un par de animaciones, sólo tenle un poco de paciencia por si demora en cargarse, ya que son puras imágenes, creo que es mejor eso a tener que explicar algo que es tan complejo, ya que podrás ver los 16

efectos y la posición de los interruptores para cada uno de ellos..., ya puedes ingresar...

--- Ver los 16 efectos ---

La verdad es que en pdf no vas a ver nada, pero bueno, ahi queda...

:: PIC - Parte II - Los Efectos de CQPIC

CQPIC: Secuenciador de 8 Canales y 16 Efectos

Algunos de los efectos fueron conservados del original, bueno, los que más me gustaban, el resto fue hecho en casa, notarás que además incluí la posición de los interruptores escrita en binario y a la vez el número que le corresponde en hexadecimal, eso nos hará falta para cuando comencemos a codificar...

Interruptores en posición **0000**, 0x00 en hexa. Hecho en casa...

Interruptores en posición **0001**, 0x01 en hexa, conservado del original...

Interruptores en posición **0010**, 0x02 en hexa. Conservado del original...

Interruptores en posición **0011**, 0x03 en hexa. Hecho en casa...

Interruptores en posición **0100**, 0x04 en hexa. Hecho en casa...

Interruptores en posición **0101**, 0x05 en hexa. Hecho en casa...

Interruptores en posición **0110**, 0x06 en hexa. Hecho en casa...

Interruptores en posición **0111**, 0x07 en hexa. Conservado del original...

Interruptores en posición **1000**, 0x08 en hexa. Hecho en casa...

Interruptores en posición **1001**, 0x09 en hexa. Hecho en casa...

Interruptores en posición **1010**, 0x0A en hexa. Hecho en casa...

Interruptores en posición **1011**, 0x0B en hexa. Hecho en casa...

Interruptores en posición **1100**, 0x0C en hexa. Hecho en casa...

Interruptores en posición **1101**, 0x0D en hexa. Hecho en casa...

Interruptores en posición **1110**, 0x0E en hexa. Hecho en casa...

Interruptores en posición **1111**, 0x0F en hexa, hecho en casa..., como verás ya no sabía que hacer...!!!

Espero haya sido de tu agrado, ya que a esto nos dedicaremos de ahora en más...

:: PIC - Parte II - Capítulo 6

Estuvo bueno no crees...???

Descripción general del proyecto - Los puertos:

Ahora es que empieza todo. Primero tenemos que saber que es lo que haremos con las entradas y que haremos con las salidas, cuantas salidas son necesarias y cuantas entradas también, debes tener en cuenta que en algún momento querrás tener control en la velocidad del desplazamiento de las luces, lo cual ya te puede llevar a pensar en un potenciómetro, pues si, es así, de todos modos lo aclararemos un poco más...

Salidas:

El secuenciador posee 8 salidas **que tomaremos del puerto B** y las asignaciones serán las siguientes...

- Salida 0 <-- RB0 (pin6)
- Salida 1 <-- RB1 (pin7)
- Salida 2 <-- RB2 (pin8)
- Salida 3 <-- RB3 (pin9)
- Salida 4 <-- RB4 (pin10)
- Salida 5 <-- RB5 (pin11)
- Salida 6 <-- RB6 (pin12)
- Salida 7 <-- RB7 (pin13)

Entradas:

Los 4 interruptores necesariamente deben ser entradas y las conectaremos al **puerto A**, estas serán las asignaciones...

- llave 0 <-- RA0 (pin17)
- llave 1 <-- RA0 (pin18)
- llave 2 <-- RA0 (pin1)
- llave 3 <-- RA0 (pin2)
- timer <-- RA0 (pin3)

Los pulsos de reloj que generan el desplazamiento de los distintos efectos

serán entregados por un típico timer "el 555", eso permitirá el control de la velocidad por medio de un potenciómetro, como ves le asigne el pin RA4

Esto es sólo para que tengas en cuenta como es que vamos a hacer las conexiones al pic, bien, esas serán todas las conexiones que le haremos.

Ahora pasemos a ver como vamos a encarar los inicios de la programación

Descripción general del proyecto - Primer Planteo del Programa:

Piensa en como debe trabajar el PIC desde que reciba la alimentación hasta que se encuentra trabajando a full, piensa en que el usuario es muy torpe y siempre suele meter la pata, ten en cuenta también la velocidad a la cual trabaja el micro, y por último piensa que... bueno, mejor sigamos...

Parte I: Cuando el micro se inicia

Cada vez que el micro reciba corriente de la fuente de alimentación (esto es, cada vez que lo conecten o enciendan el secuenciador) éste, deberá setear los puertos darle a cada pin del micro la asignación que le corresponde (entradas y salidas), así es que eso es lo que haremos en la primera sección del código.

Parte II: Verificando el estado de los interruptores

Luego de lo anterior que es lo que siempre se hace, comienza lo nuestro, lo primero que haremos será verificar los interruptores, y para ello pasaremos todo lo que hay en el **Puerto A** a un registro llamado **switch**, el tema es que al hacer esto también viene incluido el timer, y como este cambia a todo momento debemos quitarlo del registro y quedarnos con los bits 0, 1, 2y 3 que son los que nos interesan, mmmmmmm... entonces haremos lo siguiente, un AND con 00001111 (0x0f en hexa), recuerda que una operación AND da 1 como resultado siempre que los dos bits sean 1 y cualquier otra combinación te dará siempre cero, (pues eso es justamente lo que buscamos para sacarnos de encima al timer). Ahora bien, luego de esta operación tenemos el estado de los interruptores conectados al puerto A, sin el timer, es decir tienes el estado de los interruptores totalmente en limpio, así que hay que cuidarlos, y para no perderlos lo vamos a guardar en un registro llamado **llaves**

Parte III: Seleccionando los efectos

Teniendo el valor de las llaves, lo que nos toca es ver cual de todas las combinaciones de estos 4 bits corresponde al efecto 0, al 1, al 2, etc. y una vez lo descubrimos, hacemos una llamada a ese efecto.

Suponte que la combinación de las llaves es 0000 eso significa que le corresponde al primer efecto que lo llamaré **efect1**, entonces haremos una llamada para ejecutar todo lo que se encuentre en **efect1** y allí, lo primero que haremos será limpiar el **puerto B** (ponerlo a cero) y comenzaremos con ese efecto divino que viste anteriormente. una vez termine el efecto regresaré a revisar nuevamente las llaves para ver si el usuario seleccionó otro, y si lo hizo, pues entonces cambiaré, sino reiniciaré el efecto que tenía anteriormente, y así sucesivamente.

Parte IV: El timer

Cada vez que enviemos un dato al puerto B por cualquiera de los efectos seleccionados haremos una llamada al timer para ver si hay un nuevo pulso, y cuando este pulso termine regresaremos para enviar el siguiente dato, y luego de eso volveremos al timer, y así...

El timer es una de las partes más importantes aunque la más corta de toda la codificación, debes recordar que el timer mantiene niveles altos y niveles bajos de tensión, vamos a suponer lo siguiente... que un pulso comienza cuando se inicia un nivel alto, y termina cuando comienza el nivel bajo, por lo que si la velocidad de timer es lenta el dato que se envíe a la salida, también será lento, pues bien para eso nos sirve verificar el cambio de nivel del timer.

Lo dicho anteriormente, mas claro con una imagen

El código que analiza el comienzo del nivel alto lo llamaré **clockon**, y al que analiza el final de este nivel **clockoff**, que creativo no...?

Parte V: Cómo se codifican los efectos

Esta es la parte que te interesa verdad...?, como la mayoría de los efectos no son más que un desplazamiento de bits, haremos uso de una instrucción de rotación, para la cual utilizaremos el bit de **CARRY**, y para que tengas una idea de lo que estoy hablando se trata del primer bit (el **bit0**) del registro **STATUS**, sí... aquel en que configuras los puertos, siiii, ese mismo, bueno, espero que te acuerdes...!!!, luego lo veremos más en detalle.

Parte VI: Nota

No viene mal hacer algunas aclaraciones... En los comentarios que encuentres en el código verás que puse el valor de los bits del puerto B en binario, eso aclarará un poco de que efecto estamos hablando y nos ayudará para no perdernos, ya que la numeración utilizada en el cógio es hexadecimal.

Bueno mi querido colega, estamos listos para iniciar la codificación, sin armar el circuito, claro...!!!, ya que lo simularemos con MPLAB, de eso se trataba o no...???, calma, calma.... que cuando termines de leer este tutorial tendrás el circuito listo para montar, si es que lo deseas, de todos modos habrás dado un gran salto en la simulación del micro en MPLAB...

:: PIC - Parte II - Capítulo 7

Para los más curiosos

Y sí... Esta sección está dedicada para los más curiosos, aquellos que desean comprender desde cero como se hicieron todos esos efectos, aquellos que son algo así como yo y que no quieren nada servido, sino que buscan aprender a como de lugar, que palo no...???, ok para los otros, que ya tienen conocimiento más avanzado y solo quieren armar el proyecto les dejo aquí el código fuente...

Los demás no descarguen este archivo y no me pregunten por qué, solo síganme que esto aún no termina. Para no liarte tirando todo el código así de una, lo voy a trocear en varias partes, así es que vamos por la primera...

Primera parte - Cuando el micro se inicia

```
;====================== Encabezado ======================
;
;                        Ero-Pic // De Rueda Luis
;                    Secuenciador de 8 Canales y 16 efectos.
;
;=========================================================

          LIST     P=16F84
          include "P16F84.inc"

;=================== Mapa de Memoria ====================

estado  equ    0x03    ; Haciendo asignaciones
trisa   equ    0x05
trisb   equ    0x06
porta   equ    0x05
portb   equ    0x06

llaves  equ    0x0C    ; almacenara el estado de las llaves
```

```
;=============== Configuración de puertos ================

            ORG     0x00
            GOTO    inicio
            ORG     0x05

inicio  BSF     estado,5    ; cambio al banco 1 del pic
            MOVLW   0x1f
            MOVWF   trisa   ; Asigna al puerto A como entrada
            MOVLW   0x00
            MOVWF   trisb   ; y Al puerto B como salida
            BCF     estado,5    ; Regresa al banco 0
            CLRF    porta       ; limpia el puerto A
            CLRF    portb       ; limpia el puerto B
```

Descripción del código

Encabezado:

De nuestro primer tutorial, si es que lo viniste siguiendo, sabemos que el encabezado es sólo un comentario y luego de el, viene el famoso **List P=16f84** que no es otra cosa que la referencia a nuestro micro, para el cual estamos codificando, lo nuevo aquí es **include "P16F84.inc"**, esto es para que cuando MPLAB ensamble nuestro código, haga uso de la librería "P16F84.inc" en la cual tiene todos los datos de este integrado, si olvidas colocarlo, cuando lo ensambles te dará error, y eso no es agradable...!!!

Mapa de memoria

Como antes, aquí damos a conocer los registros que utilizaremos, para que los ponga a nuestra disposición, y si necesitas una variable, simplemente le asignas el nombre a uno de los registros que nos quedan disponibles, como podrás ver todos los nombres que utilicé están escritos en minúsculas (detalle a tener en cuenta), entre todos ellos hay uno raro...

llaves equ 0x0C

Como dice el comentario, es el que utilizaré para almacenar el valor de las llaves (los 4 interruptores que seleccionan el efecto), el resto ya lo conocíamos.

Configuración de Puertos

Aquí haremos las asignaciones a las entradas (interruptores y timer) del

puerto A y las salidas (8 salidas para los LED's) en el puerto B

```
;=============== Configuración de puertos ================

            ORG     0x00
            GOTO    inicio
            ORG     0x05

inicio  BSF     estado,5    ; cambio al banco 1 del pic
            MOVLW   0x1f
            MOVWF   trisa   ; Asigna al puerto A como entrada
            MOVLW   0x00
            MOVWF   trisb   ; y Al puerto B como salida
            BCF     estado,5    ; Regresa al banco 0
            CLRF    porta       ; limpia el puerto A
            CLRF    portb       ; limpia el puerto B
```

ORG 0x00 es el vector de Reset, cada vez que el micro se inicie lo hará desde aquí (ya sea que conectaron el secuenciador directamente o que lo encendieron o que algo le haya ocurrido al micro saltará a este punto) y en la siguiente instrucción irá a la etiqueta **inicio** pasando por encima del vector de interrupción, que está en la posición 0x04.

ORG 0x05 desde aquí comenzaremos a ensamblar nuestro código (significa eso en pocas palabras).

El resto ya nos es familiar, traduciré en entendible línea por línea.

cambio al banco 1.
carga el registro w con 00011111 (en binario), ó 0x1f en hexadecimal, por si dudas, utiliza la calculadora de windows para ver esta equivalencia.
pasa w a trisa y quedan los 5 primeros bits del puerto A como entrada.
carga w con 00000000 (en binario), ó 0x00 en hexa.
lo pasa a trisb y ahora todo el puerto B es salida.
regresa al banco 0.
finalmente limpia (CLRF) todo lo que haya en ambos puerto.

CLRF es como decir... clear a todo lo que hay en el registro F que te voy a especificar (porta y portb), así por ejemplo...

CLRF portb

esto es simplemente para asegurarse de que no hay ninguna cosa rara por ahí...

Y así fue nuestra primera parte... ahora vamos a lo otro...

Una más...!!!

:: PIC - Parte II - Capítulo 8

Lo que viene ahora es verificar el estado de los interruptores, no te asustes por lo extensa que es esta parte del código, no es nada complicado ya que se repite 16 veces para seleccionar uno de los 16 efectos disponibles según la combinación de las llaves, o interruptores.

Parte II: Verificando el estado de los interruptores

```
switch  MOVF    porta,0     ; carga w con el puerto A
        ANDLW   0x0F        ; retiene los 4 bits de interés (las llaves)
        MOVWF   llaves      ; y los guarda en llaves
        XORLW   0x00        ; verifica si es el primer efecto
        BTFSC   estado,2    ; si es así
        CALL    efect1      ; lo llama y lo ejecuta
        MOVF    llaves,0    ; sino, carga llaves en w
        XORLW   0x01        ; y verifica si es el segundo efecto
        BTFSC   estado,2    ; si es así
        CALL    efect2      ; lo llama y lo ejecuta
        MOVF    llaves,0    ; y así con los demás
        XORLW   0x02        ; ya me aburrí
        BTFSC   estado,2    ; como verás el resto es lo mismo
        CALL    efect3
        MOVF    llaves,0
        XORLW   0x03
        BTFSC   estado,2
        CALL    efect4
        MOVF    llaves,0
        XORLW   0x04
        BTFSC   estado,2
        CALL    efect5
        MOVF    llaves,0
        XORLW   0x05
        BTFSC   estado,2
        CALL    efect6
        MOVF    llaves,0
        XORLW   0x06
        BTFSC   estado,2
        CALL    efect7
        MOVF    llaves,0
        XORLW   0x07
        BTFSC   estado,2
        CALL    efect8
        MOVF    llaves,0
        XORLW   0x08
        BTFSC   estado,2
```

```
        CALL    efect9
        MOVF    llaves,0
        XORLW   0x09
        BTFSC   estado,2
        CALL    efect10
        MOVF    llaves,0
        XORLW   0x0A
        BTFSC   estado,2
        CALL    efect11
        MOVF    llaves,0
        XORLW   0x0B
        BTFSC   estado,2
        CALL    efect12
        MOVF    llaves,0
        XORLW   0x0C
        BTFSC   estado,2
        CALL    efect13
        MOVF    llaves,0
        XORLW   0x0D
        BTFSC   estado,2
        CALL    efect14     ; Comienza a revisar de nuevo
        MOVF    llaves,0
        XORLW   0x0E
        BTFSC   estado,2
        CALL    efect15
        MOVF    llaves,0
        XORLW   0x0F
        BTFSC   estado,2
        CALL    efect16
        GOTO    switch
```

Descripción del código

Jaja...!!!, no voy a describir línea por línea hasta el final, y no me lo pidas por que no me vas a convencer... :o))

Lo que haré será describir esta sección que es la mas crucial, y el resto puedes detenerte a pensarlo a verlo, y descubrirás que es lo mismo...

Aquí esta parte del código...

```
switch  MOVF    porta,0     ; carga w con el puerto A
        ANDLW   0x0F        ; retiene los 4 bits de interés (las llaves)
        MOVWF   llaves      ; y los guarda en llaves
        XORLW   0x00        ; verifica si es el primer efecto
        BTFSC   estado,2    ; si es así
        CALL    efect1      ; lo llama y lo ejecuta
        MOVF    llaves,0    ; sino, carga llaves en w
        XORLW   0x01        ; y verifica si es el segundo efecto
        BTFSC   estado,2    ; si es así
        CALL    efect2      ; lo llama y lo ejecuta
        MOVF    llaves,0    ; y así con los demás
        XORLW   0x02        ; ya me aburrí
        BTFSC   estado,2    ; como verás el resto es lo mismo
```

switch es la etiqueta que le corresponde a toda esta sección de código, y le puse ese nombre por que es justamente lo que se hace, revisar todos los switch's.

MOVF porta,0

Repasemos un poco, **MOVF** significa mover todo lo que hay en el registro **F**, el registro F en este caso es **porta** y para especificar que lo debe guardar en **W** le ponemos un **0**, con esto pasamos el **puertoA** a **W**.

ANDLW 0x0F

Esto es, hacer **L AND W** y guardarlo en **W**, ahora bien, **L** es un literal (un valor) en este caso **0x0F** y **W** es el que contiene el valor del puerto A, veamos un ejemplo supongamos que lo que hay en el puerto A es 00010001, eso significa que W=00010001 (gracias a la instrucción anterior) cuando hagas el AND con 0x0F (00001111 en binario) el resultado será el siguiente...

```
  00001111   L
  00010001   W
--------------------
  00000001   ANDLW
```

Ahora es W=00000001, entiendes por que hice uso de la instrucción AND, bueno así es que nos quedamos con los bit's de interés (los bits de interés son los interruptores), ok, sigamos

MOVWF llaves

Es como decir **MOV** todo lo que hay en **W** al registro **F**, F en este caso es **llaves**, y llaves es el nombre que le asignamos al registro 0x0C (en el encabezado... recuerdas...???), muy bien, ahora llaves contiene el estado de los interruptores (en limpio, es decir sin el timer). Si seguimos con el ejemplo

anterior luego de la instrucción, nos quedará **llaves=00000001**, o sea que el primer interruptor está activado...

XORLW 0x00

hacer **W XOR L**, recuerda que W=00000001 y en este caso L=00000000 (0x00 en hexa) recuerda también que la operación XOR entre dos bits dará como resultado **0** si ambos bits contienen el mismo valor, ahora, el resultado de esta operación será..

```
  00000001   W
  00000000   L
--------------------
  00000001   XORLW
```

Es cuestión de segundos para que te des cuenta que lo que estamos haciendo es una comparación, si los valores son los mismos el resultado dará cero.

BTFSC estado,2

Esto es como decirle... mira ve y prueba el Bit **B** del registro **F** y saltéate una línea si es que es **0**.

Ok, aclaremos un poco esto, B es el bit2 del registro STATUS y F ahora es el registro STATUS, te preguntarás que tiene que ver el registro STATUS con todo lo que estamos haciendo, te lo explicaré, el bit2 del registro STATUS es el tercer bit **"Z"** (conocido como bandera de CERO) y esa bandera se pone a uno siempre que una operación aritmética o lógica como la anterior (**XORLW**) de como resultado **0 (CERO)**, en cualquier otro caso se pondrá en cero.

Ahora vamos a repetirlo para que quede mas claro, Piensa que las dos instrucciones que acabamos de ver van de la mano

```
XORLW  0x00      ;XOR entre W y 00000000 (todas las llaves en cero)
BTFSC  estado,2  ;prueba si Z=0 si es así se saltea una línea.
```

del ejemplo anterior la operación dio como resultado 00000001 por lo tanto la bandera no cambio sigue siendo **Z=0** entonces me salteo una línea y voy a esa instrucción

MOVF llaves,0

Vuelvo a cargar **W** con el contenido del registro **llaves**, recuerda que llaves contiene el estado de los interruptores en limpio

Lo que haré ahora será comparar el estado de los interruptores con el segundo efecto ya que el anterior no era, así que haré un nuevo XOR, así...

XORLW 0x01 ;XOR entre W y 00000001 (sólo el primer interruptor en 1)

El tema es que ahora el resultado de esta operación será **00000000** ya que **W** tiene el mismo valor que el literal 0x01, por lo tanto la bandera **Z** se hace **Z=1** y cuando pase a...

BTFSC estado,2 ;saltea una línea si Z=0

Pues no saltare nada ya que Z es 1 y pasaré a un...

CALL efect2

CALL es una llamada, y CALL efect2 es llamar a la etiqueta efect2, esa etiqueta contiene el segundo efecto de luces, ya que el primero no lo pude ejecutar por que no correspondía.

Una vez terminado de ejecutar efect2 volveré a verificar el estado de las llaves, es decir, volveré a...

MOVF llaves,0

Pero llaves sigue siendo 00000001, por lo tanto todas las otras operaciones me darán cualquier cosa y no 00000000, esto ocurrirá hasta que llegue a la instrucción...

GOTO switch

Un espectacular salto (**GOTO**) a la etiqueta **switch** (el inicio de toda esta sección de código), donde me encuentro con...

MOVF porta,0

y cargo W con todo lo que hay en el puerto A (que vendría a ser el nuevo estado de los interruptores). Luego haré

ANDLW 0x0F

Para sacarme de encima al timer, y luego un...

MOVF llaves,0

para cargar el registro **llaves** con el nuevo estado de los interruptores conectados al puerto A, y estoy listo para verificar de que efecto se trata, y cuando lo descubra haré una llamada al efect**X** que le corresponde y lo ejecutaré, y así sucesivamente.

Espero que hayas logrado comprender esta sección y sino analiza el código nuevamente hasta que lo entiendas, te confieso que no es tan complicado

como parece.

Si está todo claro puedes pasar a la siguiente página...

Ahí te espero...

:: PIC - Parte II - Capítulo 9

Los efectos - Primera Sección

Al fin lo que estabas esperando, es hora de codificar los efectos, como dije anteriormente, en los comentarios del código utilicé la numeración en binario para orientarte un poco más y así sabes que salida esta activa.

Bien ahora vamos por el primer efecto que sería este...

El código para obtener este efecto es muy sencillo, y se trata de ir activando cada uno de los bits del puerto B por cada nuevo pulso de reloj. Aquí tienes los detalles...

```
;==================== Efectos ====================

efect1 CLRF    portb    ; limpia el puerto B
       BSF     portb,7  ; (10000000)
       CALL    clokon
       BSF     portb,6  ; (11000000)
       CALL    clokon
       BSF     portb,5  ; (11100000)
       CALL    clokon
       BSF     portb,4  ; (11110000)
       CALL    clokon
       BSF     portb,3  ; (11111000)
       CALL    clokon
       BSF     portb,2  ; (11111100)
       CALL    clokon
       BSF     portb,1  ; (11111110)
       CALL    clokon
       BSF     portb,0  ; (11111111)
       CALL    clokon
       BCF     portb,7  ; (01111111)
       CALL    clokon
       BCF     portb,6  ; (00111111)
       CALL    clokon
       BCF     portb,5  ; (00011111)
       CALL    clokon
       BCF     portb,4  ; (00001111)
       CALL    clokon
```

```
        BCF     portb,4 ; (00000111)
        CALL    clokon
        BCF     portb,3 ; (00000011)
        CALL    clokon
        BCF     portb,2 ; (00000001)
trece   CALL    clokon
        BCF     portb,1 ; (00000000)
        CALL    clokon
        CLRF    portb   ; (00000001)
        CALL    clokon
                        ; (00000011)
        BSF     portb,0
        CALL    clokon  ; (00000111)
        BSF     portb,1
        CALL    clokon  ; (00001111)
        BSF     portb,2
        CALL    clokon  ; (00011111)
        BSF     portb,3
        CALL    clokon  ; (00111111)
        BSF     portb,4
        CALL    clokon  ; (01111111)
        BSF     portb,5
        CALL    clokon  ; (11111111)
        BSF     portb,6
        CALL    clokon  ; (11111110)
        BSF     portb,7
        CALL    clokon  ; (11111100)
        BCF     portb,0
        CALL    clokon  ; (11111000)
        BCF     portb,1
        CALL    clokon  ; (11110000)
        BCF     portb,2
        CALL    clokon  ; (11100000)
        BCF     portb,3
        CALL    clokon  ; (11000000)
        BCF     portb,4
        CALL    clokon  ; (10000000)
        BCF     portb,5 ; a revisar nuevamente las llaves
        CALL    clokon
        BCF     portb,6
        CALL    clokon
        RETURN
```

No se si es necesario explicarlo pero ahi va, **efect1** es la etiqueta para este efecto, y en la primera línea me aseguro de que no hay señal en la salida haciendo un CLRF portb, es decir dejo todo el puerto B en (00000000).

```
;==================== Efectos ====================

efect1 CLRF        portb    ; limpia el puerto B
       BSF         portb,7  ; (10000000)
       CALL        clokon
       BSF         portb,6  ; (11000000)
       CALL        clokon
       BSF         portb,5  ; (11100000)
       CALL        clokon
       BSF         portb,4  ; (11110000)
```

Ahora paso a la siguiente instrucción...

BSF portb,7

BSF es poner a 1 el bit del registro F, es decir... poner a 1 el bit7 del registro portb. el resultado es (10000000)

CALL clokon

Esto es fácil, sólo llama a la etiqueta clokon, la ejecuta y regresa, ya se que debería ser clockon, pero es muy largo, imagínate como quedaría clockoff, peor aún...!!!, así que lo dejemos así.

clokon es la rutina que verifica el estado del timer. Entonces si hay un pulso regresa y hace...

BSF portb,6

Pone a 1 el bit6 de portb, y el resultado es (11000000), (fíjate que el bit7 aún esta activo, bueno, estará así hasta que le pongas un **0**), luego de eso va a verificar si hay un nuevo pulso...

CALL clokon

si lo hay regresa y activa el bit5 de portb, y así hasta que estén todos encendidos o sea (11111111) observa el código más arriba...

Cuando eso ocurra haré un...

BCF portb,7

Es decir poner a 0 el bit7 de portb, y el resultado es (01111111), luego de eso va a verificar si hay un nuevo pulso, y si lo hay pondré a cero el siguiente bit y quedará (00111111), y así hasta que portb sea (00000000) y luego haré lo mismo, pero al revés.

Como ves es muy sencillo... En cada instrucción no apunto a todo el puerto B sino a uno de los bits de ese puerto, activándolo a desactivándolo.

Pasemos a lo siguiente...

Perdón, pero antes de que lo olvide... En cierto lugar de este código incluí una etiqueta llamada **trece**, cuando llegue el momento hablaremos de ella, por ahora haz de cuenta que no existe de acuerdo...??

Ahora si sigamos...

:: PIC - Parte II - Capítulo 10

Los efectos - Sección II

Si en el primer efecto ibas activando los bits uno a uno con cada pulso de reloj y dejando el anterior activo y al final llegabas con todos llenos, lo que haremos aquí será llevar uno activo pero quitando el anterior y una vez lleguemos al extremo lo dejaremos a ese en nivel alto y comenzaremos de nuevo para obtener este efecto...

Aquí tienes el código para lograrlo...

```
efect2  CLRF    portb       ; limpia el puerto B
        BSF     portb,7     ; (10000000)
        CALL    clokon
        BCF     portb,7     ; (00000000)
        BSF     portb,6     ; (01000000)
        CALL    clokon
        BCF     portb,6     ; (00000000)
        BSF     portb,5     ; (00100000)
        CALL    clokon
        BCF     portb,5     ; (00000000)
        BSF     portb,4     ; (00010000)
        CALL    clokon
        BCF     portb,4     ; (00000000)
        BSF     portb,3     ; (00001000)
        CALL    clokon
        BCF     portb,3     ; (00000000)
        BSF     portb,2     ; (00000100)
        CALL    clokon
        BCF     portb,2     ; (00000000)
        BSF     portb,1     ; (00000010)
        CALL    clokon
        BCF     portb,1     ; (00000000)
        BSF     portb,0     ; (00000001)
        CALL    clokon
        BSF     portb,7     ; (10000001)
        CALL    clokon
        BCF     portb,7     ; (00000001)
                            ; (01000001)
                            .
```

```
        BSF         portb,6     .
        .           .           .
        .           .
        .           .
        RETURN
```

Perdona, pasa que es muy extenso, pero no te preocupes que luego te daré el código completo, por ahora sólo trata de comprender como se hicieron todos estos efectos...

Bien, veamos parte de las instrucciones...

```
efect2  CLRF        portb       ; limpia el puerto B
        BSF         portb,7     ; (10000000)
        CALL        clokon
        BCF         portb,7     ; (00000000)
        BSF         portb,6     ; (01000000)
        CALL        clokon
        BCF         portb,6     ; (00000000)
        BSF         portb,5     ; (00100000)
        CALL        clokon
```

Como siempre, lo primero que hacemos es borrar el puerto para comenzar desde cero así que...

CLRF portb

luego ponemos un 1 en el bit7 del puerto B y vamos a verificar si hay un nuevo pulso...

BSF portb,7
CALL clokon

Ahora viene lo nuevo ya que son como dos instrucciones en una...

BCF portb,7 ; (00000000)
BSF portb,6 ; (01000000)

En la primer línea borro el bit que activé anteriormente y en la segunda pongo a 1 el bit que sigue (6) (eso es lo que genera el desplazamiento...!!!), luego llama al timer para recién hacer otro cambio, esta es una de las formas de mover un bit de un extremo a otro.

Saltemos un par de líneas más abajo y supongamos que logramos llegar al otro extremo de portb, entonces verás este trozo de código...

BSF portb,0 ; (00000001)

```
CALL   clokon
BSF    portb,7    ; (10000001)
CALL   clokon
BCF    portb,7    ; (00000001)
BSF    portb,6    ; (01000001)
CALL   clokon
```

En la primer línea llegamos al bit0 de portb el cual ya no tocaremos, y comenzaremos nuestro recorrido nuevamente desde el bit7 al bit1, luego será del 7 al 2, luego del 7 al 3, etc. Una vez esté todo lleno haremos un RETURN.

El efecto 3, es lo mismo, pero al revés... veamos el que sigue...

:: PIC - Parte II - Capítulo 11

Los efectos - Sección III

Si te ubicaste con los anteriores este ya te será más fácil de comprender, quizás hasta con solo verlo ya sabes como se hace...

Aquí una parte del código...

```
efect4  CLRF    portb     ; limpia el puerto B
        BSF     portb,7   ; (10000000)
        BSF     portb,0   ; (10000001)
        CALL    clokon
        BSF     portb,6   ; (11000001)
        BSF     portb,1   ; (11000011)
        CALL    clokon
        BSF     portb,5   ; (11100011)
        BSF     portb,2   ; (11100111)
        CALL    clokon
        BSF     portb,4   ; (11110111)
        BSF     portb,3   ; (11111111)
        CALL    clokon
        .         .         .
        .         .         .
        .         .
        RETURN
```

No hay mucho que explicar, sólo activar los bits de ambos extremos y luego ir llenando hacia el centro, Luego los desactivas desde ambos extremos hacia el centro, y repites todo pero al revés, desde el centro hacia afuera los vas activando y una vez lleno los desactivas desde el centro hacia afuera...

Eso es todo, recuerda que siempre se encuentra RETURN al final de cada efecto ya que cada uno de los 16 es llamado desde los interruptores...

Es importante que recuerdes que la posición de los interruptores son revisadas después de que cada efecto termina, así que no te sorprendas

que si cambiaste los interruptores y no hay cambio de efecto es porque aún no terminó de ejecutarse el efecto activo.

Bien, veamos otro...

:: PIC - Parte II - Capítulo 12

Los efectos - Sección IV

Pasemos al efecto 16 que sería el último, bueno elegí este, por que es el más sensillo de explicar, y luego cuando lo entiendas, podrás hacerle frente a los demás, la secuencia sería la siguiente...

Y este es el código completo...

```
efect16  CLRF    portb      ; limpia el puerto B
         BSF     portb,7    ; (10000000)
         BCF     estado,0   ; pone a 0 el bit C de status (el 1º bit)
seis     CALL    clokon
         RRF     portb,1    ; rota a la derecha
         BTFSS   portb,0    ; ve si terminó de rotar
         GOTO    seis
         CLRF    portb      ; (00000000)
         BSF     portb,0    ; (00000001)
siete    BCF     estado,0   ; pone el carry a 0
         CALL    clokon
         RLF     portb,1    ; rota a la izquierda
         BTFSS   portb,7    ; ve si terminó de rotar
         GOTO    siete
         CALL    clokon
         RETURN
```

Lo que hice en este efecto es una rotación, primero en un sentido, luego en otro y para ello hice uso del CARRY (acarreo), el CARRY (C) es el primer bit (bit0) del Registro STATUS, aquel que vimos en nuestro primer tutorial cuando hacíamos la configuración de puertos, y que siempre utilizamos para hacer el cambio de banco de memoria.

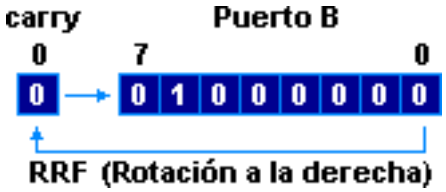
Ahora voy a tratar de explicarte como hacer un acarreo utilizando este bit. Por ahora desconozco si está a 0 ó a 1, y en la primera instrucción limpio el puerto B, de tal modo que las cosas están así...

En la segunda instrucción pongo a 1 el bit7, y en la tercera el CARRY a 0, ahora la situación cambió a esto...

Lo que hice hasta ahora sólo fue preparar las cosas para hacer una rotación, pero como el efecto ya comenzó y se activó el bit7 de portb, llamo al timer, y cuando regrese, comenzaré a rotar.

RRF portb,1 ; rota a la derecha

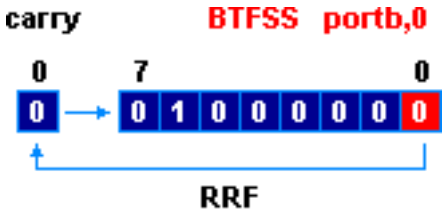
Esta es la primera rotación, antes de explicar nada, quiero que veas como queda luego de la primera rotación



La verdad creo que... no se si es necesario explicar, pero para los más duritos, lo único que ocurrió es algo así como que el bit0 pasó al CARRY (C). (aunque lo cierto es que el carry se pondrá a 1 cuando haya un desbordamiento del registro que se esta rotando, en este caso portb)

BTFSS portb,0 ; ve si terminó de rotar

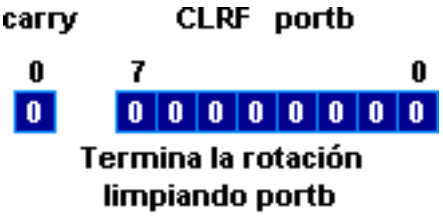
Lo que se hace aquí es una inspección en el bit0 de portb (que también podrías hacerla en el carry, pero no es este caso por ahora)



La instrucción en sí es un condicional algo así "prueba el bit0 de portb y salta una línea si es **1**", bien, ahora contesta la pregunta...!!!, es uno...?, no verdad..., pues entonces hacemos un...

GOTO seis

Es decir, vamos a la etiqueta **seis** para seguir con la rotación, y allí nos encontramos con la llamada al timer y luego bla, bla, bla ..., ahora lo vamos a ver completo, ahí va...



Bueno, creo que no lo vas a ver si no actualizas la página :o))

Observa como se desplaza el bit activo, y como luego de cada desplazamiento se hace la inspección, y una vez la inspección detecta el uno en el bit0 de portb, termina limpiando el puerto (poniéndolo a cero)

Ahora que ya terminó la rotación hacemos...

BSF portb,0 ; (00000001)

Esto es poner a 1 el bit0 de portb, y luego hacemos la rotación al revés, con **RLF**, sensillo no crees...???

Bien, ahora nos toca complicarnos un poco más por que sino esto se pone aburrido...

:: PIC - Parte II - Capítulo 13

Los efectos - Sección V

En este caso también haremos uso de la rotación mediante el carry, pero guardaremos el resultado de la rotación en W para no perderlo y luego lo enviaremos repetidas veces a portb...

El código completo te debería quedar así...

```
efect5  CLRF    portb       ; limpia el puerto B
        MOVLW   0x01        ; comienza con (00000001)
        MOVWF   portb       ; lo envía a la salida
cinco   BSF     estado,0    ; pone a 1 el bit C de status (carry)
        CALL    clokon
        MOVWF   portb       ; lo envía a la salida
        RLF     portb,0     ; rota a la izquierda y pasa el valor a W
        MOVWF   portb       ; lo envía a la salida
        CALL    clokon
        CLRF    portb       ; (00000000)
        CALL    clokon
        MOVWF   portb       ; repite
        CALL    clokon
        CLRF    portb       ; (00000000)
        CALL    clokon
        MOVWF   portb       ; repite
        BTFSS   portb,7     ; ve si terminó de rotar
        GOTO    cinco
        CALL    clokon      ; (01111111)
        BCF     portb,7
        CALL    clokon      ; (00111111)
        BCF     portb,6
        CALL    clokon      ; (00011111)
        BCF     portb,5
        CALL    clokon      ; (00001111)
        BCF     portb,4
        CALL    clokon      ; (00000111)
        BCF     portb,3
        CALL    clokon      ; (00000011)
```

```
        BCF     portb,2     ; (00000001)
        CALL    clokon
        BCF     portb,1
        CALL    clokon
        RETURN
```

Aquí hay una pequeña diferencia, ya que ponemos un 1 en el carry y no un 0 como lo hicimos anteriormente, y desde allí vamos cargando el puerto B hasta que se active el último bit de **portb**.

Veamos, hacemos una rotación, lo enviamos a portb, llamamos al timer, borramos portb, llamamos al timer, hacemos una repetición , llamamos nuevamente al timer, y así, luego de las repeticiones, verificamos si se terminó de rotar, sino hacemos una nueva rotación, finalmente terminaremos con el portb lleno.

Luego de terminada la rotación iremos poniendo a 0 todos los bits, desde el bit7 al bit0...(es como hacer un barrido limpiando todo el puerto)

Analiza el código, que no es tan complicado como parece, si lograste comprender como se hace la rotación esto ya te resultará más fácil.

:: PIC - Parte II - Capítulo 14

Los últimos efectos y el Timer

El resto de los efectos ya no es más complicado que lo que vimos hasta ahora, es más creo que son los más sencillos, aunque, ahora que lo recuerdo, para no liar entre tanto despiole que me hice en el efecto 8 combiné dos de los que ya estaban hechos para obtener este...

Demasiado corto no...???

```
efect8  CALL    efect3      ; combinan el efecto 3
        CALL    efect2      ; con el efecto 2
        RETURN
```

Mira... más fácil que esto, imposible...!!!

Ahora que recuerdo nos quedó pendiente aquello de **trece** en efect1, bien, el efecto 13 no es otra cosa que una llamada a parte del efecto uno.

```
efect13 CLRF    portb       ; limpia el puertoB
        CALL    trece       ; ejecuta parte de efect1
        RETURN
```

Ahora si podemos seguir...

Hay algo que me gustaría mostrarte que aún no lo mencioné y es el control de pulsos del timer, el código es este...

```
;============ control de pulsos de clock =============

clokon  BTFSS     porta,4     ; prueba si es 1
        GOTO      clokon      ; sino espera
clokoff BTFSC     porta,4     ; prueba si termina el pulso
        GOTO      clokoff     ; sino espera que termine
        RETURN                ; regresa y continúa

;==================== final ====================

        END
```

En la primer línea controlamos si el 555 envía señal al pin RA4, y nos quedamos esperando hasta que eso ocurra, y cuando así sea saltamos una línea más abajo a **clokoff** y esperamos a que termine el pulso, y recién entonces regresamos al sitio de donde fue llamado.

Queda recalcar que lo que intente mostrarte aquí es sólo a modo descriptivo, por lo que sólo incluí aquello que no habíamos tocado en el tutorial anterior (eso respecto al código), por lo demás solo son técnicas caseras para facilitar un poco la tarea, a demás hay algunos de los efectos que pueden ser optimizados para así tener menor cantidad de líneas de código, eso lo dejo en tus manos, yo sólo hice pié para que puedas comprender un poco más y así lanzarte a realizar otros proyectos por tu propia cuenta, como dije antes, ya se te despertaran las neuronas...

Bien mis queridos amigos, no queda más que vernos en la próxima donde comenzaremos la simulación de todo este código...

Saludos para todos...!!!

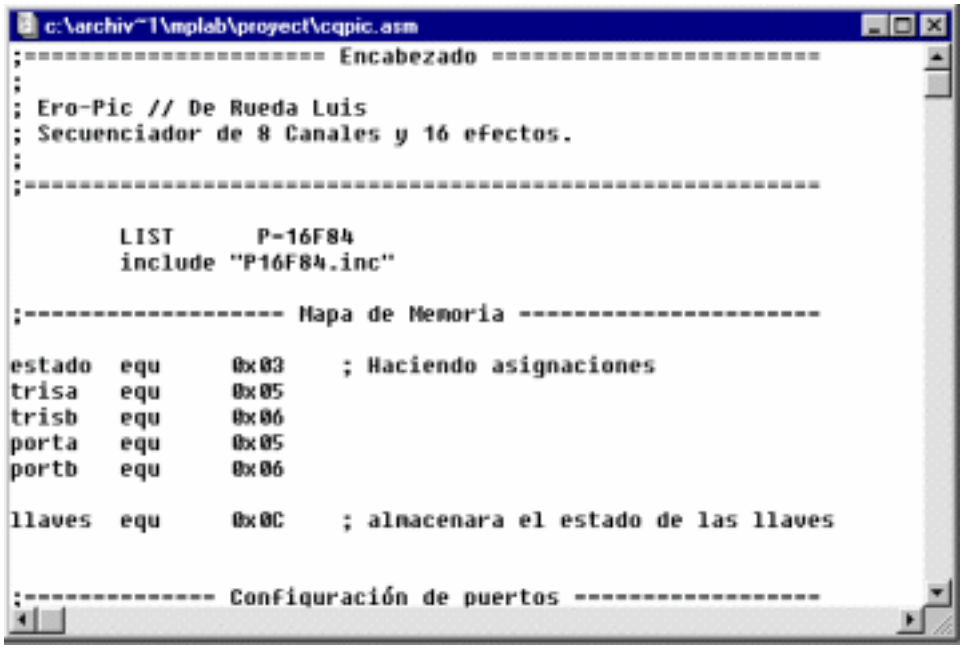
:: PIC - Parte II - Capítulo 15

Ensamblando el Código

Seguramente debes haber cerrado MPLAB, así que vamos nuevamente por el y comencemos...

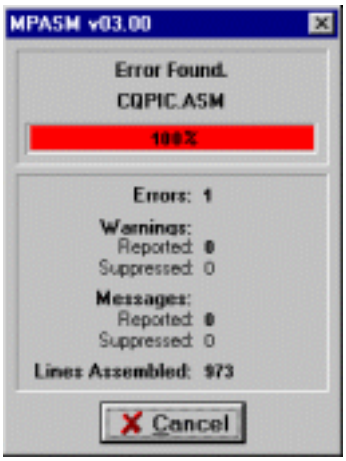
Cuando lo inicies verás un cuadro de diálogo que dice algo así **Open CQPIC.PJT?** seleccionas yes y ya estamos adentro, bueno, si por las dudas aparece otro malicioso mensaje diciendo "**No hex file has been built for this project**" según mi inglés... "No existe un archivo hex para este proyecto" pues le das a Aceptar y que sea lo que Dios quiera..., bueno por suerte apareció nuestra hoja en blanco, ahora sí...

Mira no voy a ponerte a escribir todo el código, así que te lo bajas de aquí, lo copias y lo pegas, una vez hecho esto lo tendrás así...



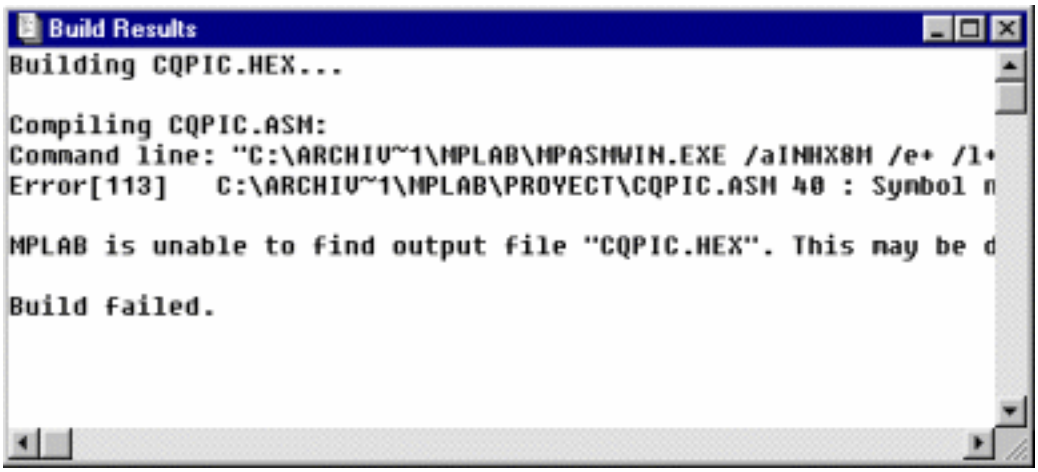
Este vínculo está inactivo, copia el codigo que está al final de este archivo, este no contiene errores

Bien, ya que lo tienes listo lo guardas **Save**, te diriges al menú **Project --> Build All** o presionas **Ctrl+F10** da lo mismo, y nuestro código comenzará a ensamblarse y una vez finalizado tendrás algo como esto...



Diablos...!!!, como odio estos mensajitos, :oP

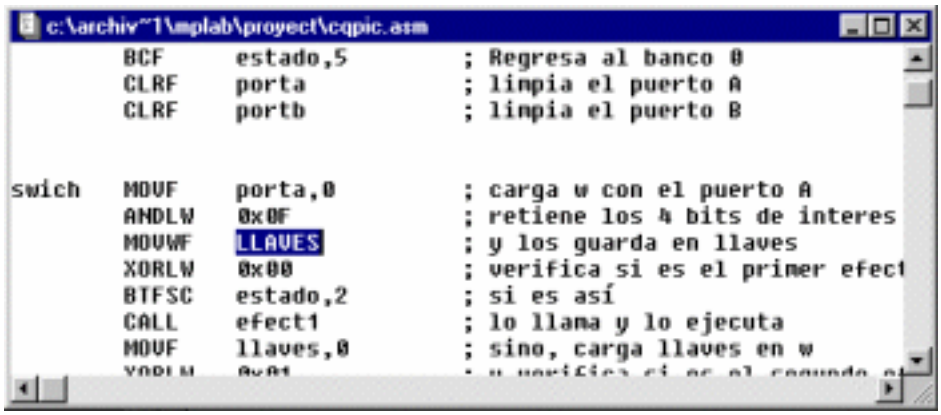
la verdad es que lo vamos a aprovechar para aprender algo más, luego de ese odioso mensaje de Error nos queda la ventana de resultado **Build Results**



El cual nos da los detalles de la metida de pata, y si sigues el mensaje verás que dice algo así...

Error[113] C:\ARCHIV~1\MPLAB\PROYECT\CQPIC.ASM 40 : Symbol not previously defined (LLAVES)

Es decir, el error está en la línea **40** en donde el símbolo **(LLAVES)** no está definido. Cómo es posible...??? si yo definí llaves al iniciar todo el código...!!! AHHHHHHHH, lo que pasa es que lo definí en minúsculas no en mayúsculas, Pues vamos a arreglarlo, no necesitas ir a buscar esa línea, sólo haz doble click sobre el error y automáticamente te llevará a el, y aquí está...



Bien, la cambiamos, guardamos el proyecto, cerramos la ventana **Build Results** y ensamblamos de nuevo, allá vamos...

Ahora siii...

Apareció lo que estábamos esperando...

Build completed successfully.

El resultado fue satisfactorio, ahora sigamos, ya que todo está bárbaro, comienza lo más interesante de MPLAB...

Simularemos que tenemos el 555, que están conectados los 4 interruptores, y los 8 LED's de las salidas del PIC.

Me acompañas...???

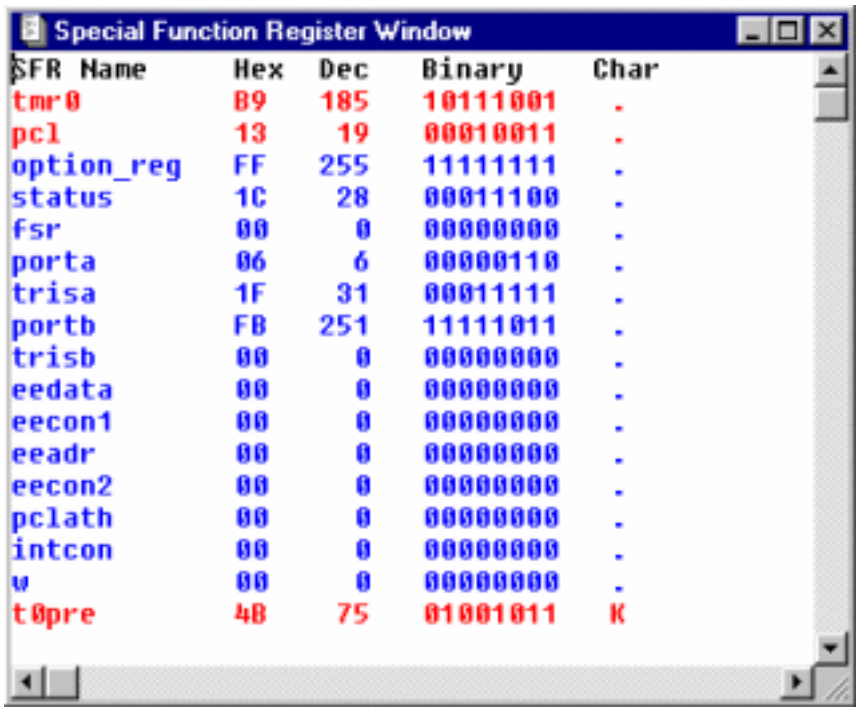
:: PIC - Parte II - Capítulo 16

Preparando el entorno para la simulación

Aquí comienza la diversión, lo que haremos será preparar todo lo que necesitemos, primero vamos por la ventana que nos muestra los registros de las funciones especiales, entonces ve al menú...

Window --> Special Function Registers

y esto es lo que verás...

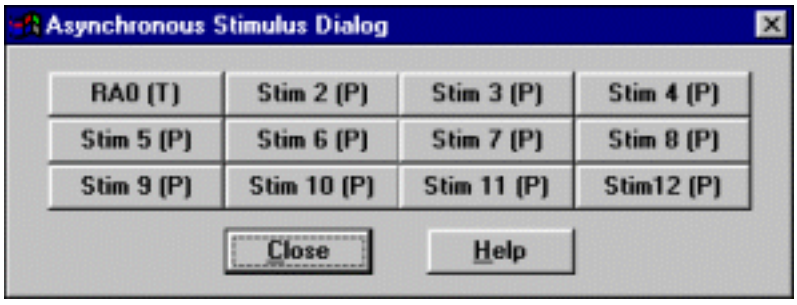


Puede que a ti te aparezcan otros valores en los registros, no te preocupes por eso, pasa que yo estuve jorobando antes, jejeje

Sigamos, en esta ventana verás como se modifican los registros cuando comencemos a simular.

Ahora vamos por otra, dirígete al menú...

Debug --> Simulator Stimulus --> Asynchronous Stimulus...



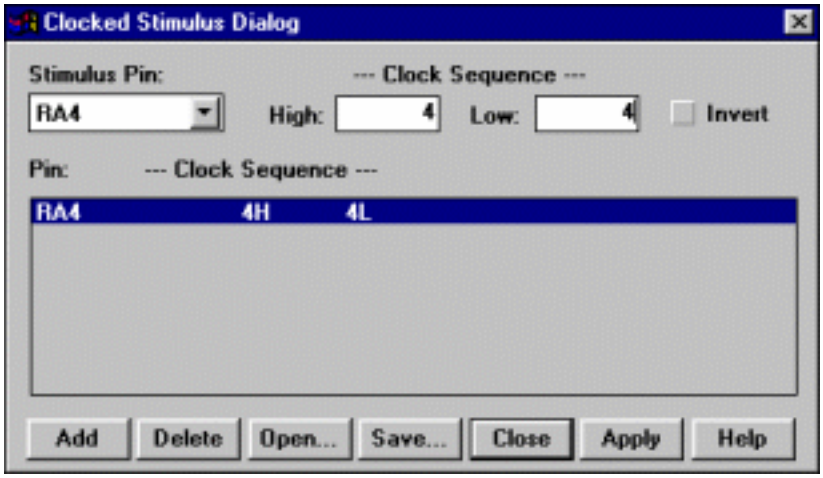
Programa RA0 como Toggle, ok de acuerdo, cuando lo abres todos los botones están sin configurar con la leyenda Stim 1(P), Stim2(P), etc. con el botón derecho del mouse haz clic en **Stim 1(P)** y verás un menu emergente, selecciónа **Assign pin.. --> RA0** ahora Stim 1(P) es RA0 (P), nuevamente haz click con el botón derecho sobre el mismo botón pero esta vez selecciona **Toggle**

Bien, yo lo hice para RA0, debes hacer lo mismo para RA1, RA2 y RA3, con esto, habrás creado los cuatro interruptores que seleccionan los distintos efectos para el secuenciador.

Lo que necesitamos ahora es el 555 o timer, como no disponemos de eso nos la arreglaremos y crearemos un timer que envíe pulsos a la patilla RA4, de acuerdo...???, entonces vamos al menú...

Debug --> Simulator Stimulus --> Clock Stimulus...

Verás el diálogo para configurar pulsos de reloj en uno de los pines, comencemos, en **Stimulus Pin:**, despliega la lista y selecciona **RA4**, en **High** y en **Low** escribe **4**, esto es el tiempo que permanecerá en nivel alto (High), y el tiempo en nivel bajo (Low), ahora presiona el botón **Add** (parte inferior de la misma ventana), y por último lo seleccionas y presionas Apply, para que se aplique a este proyecto, ya está, ya lo configuraste asi que debe estar así...



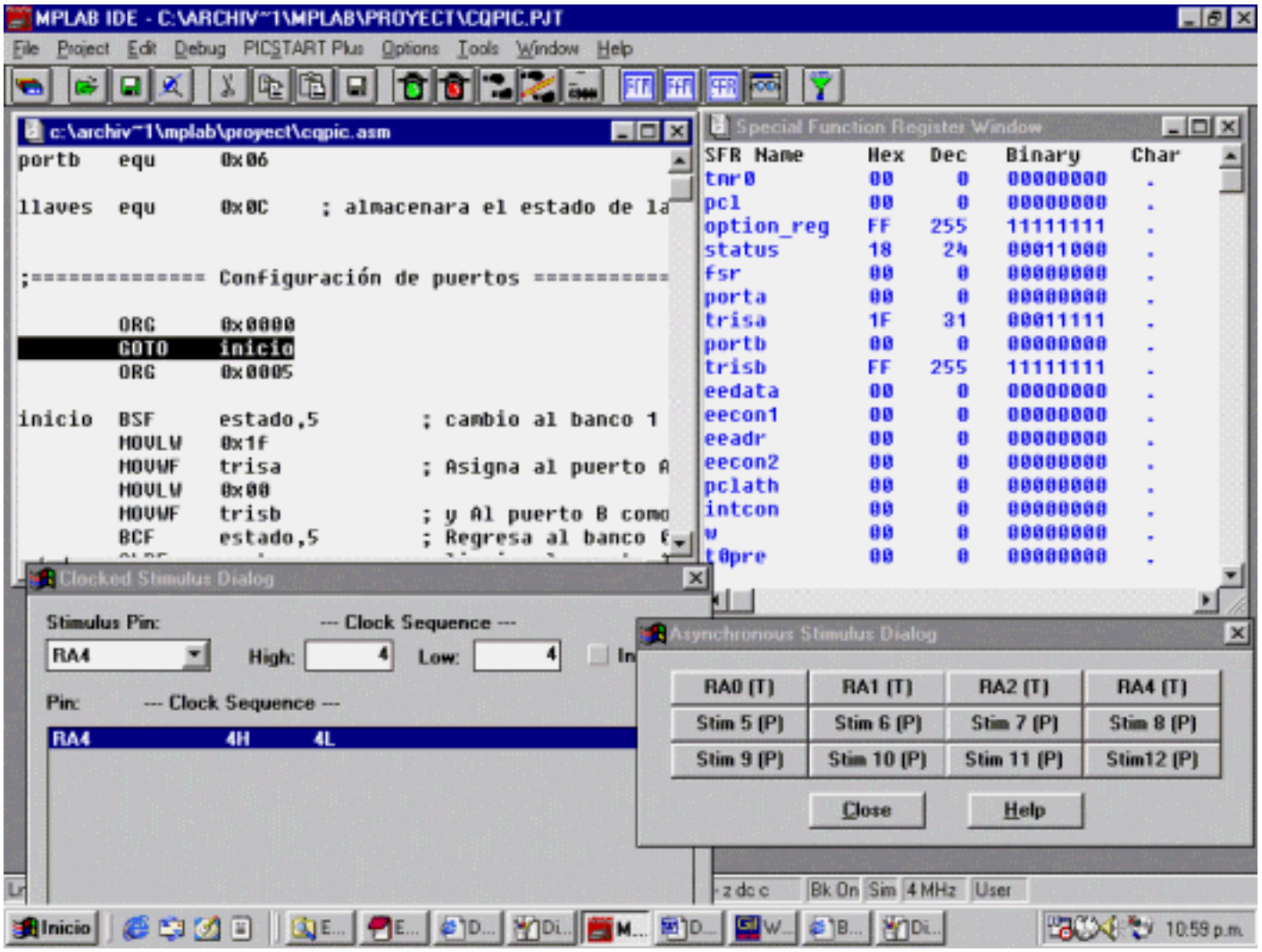
Bueno, estamos listos para comenzar la simulación, pero organiza un poco las ventanas para que puedas ver todo con mayor comodidad.

Ahora presiona el botón de Reset del micro, dirígete al menú...

Debug --> Run --> Reset

o presiona **F6** o bien, presiona este botón...

Ahora estamos en esta situación...



Observa que el micro se ubicó en la posición donde se inicia cada vez que el secuenciador se encienda, aquello que venimos diciendo desde hace tiempo

Ok, en la próxima página comenzamos a simular...

Esta es la última...!!!

Pero Quiero aclarar que esta queda así
por que yo quise, de acuerdo...!!!

:: PIC - Parte II - Capítulo 17

Comenzando con la simulación

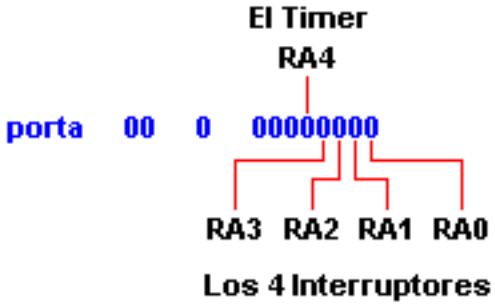
Aclaremos un poco las cosas, tienes 4 ventanas abiertas

una con el **código**
otra con el **timer**
otra más con los **registros** del micro
y una última con los 4 **interruptores**

La primera es para que sigas al código mientras este se ejecuta y no la tocaremos mientras se esté ejecutando.

La segunda sólo tiene la configuración del timer y ésta enviará señales al micro como si fuese un 555. Tampoco la tocaremos

La ventana de Registros del micro nos mostrará los cambios en todos los registros del micro, en especial los que queremos ver, como ser los pulsos en RA4 (que hace de timer), el estado de los interruptores RA0, RA1, RA2 y RA3, que son los que se modificarán cada vez que presiones alguno de los 4 botones que configuramos anteriormente como Toggle, te mostraré donde están...



Ese es el registro del puerto A. Pero también verás los cambios en el Puerto B.

portb 00 0 00000000

Estas son las salidas, es decir los LED's, claro que en forma simulada :oP

La cuarta ventana, la de los pulsadores es la única que tocaremos, presionando los botones para cambiar los datos del puerto A (es decir, para cambiar de efecto).

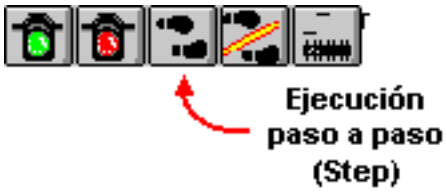
Perfecto...!!!, ahora comenzamos a toquetear, Ve al menú...

Debug --> Run --> Run

Jaja, hizo runrun, ves lo rápido que se ejecuta todo el código, ahora me crees...???, bueno, esto fue sólo para que lo veas, ahora presiona el botón del semáforo rojo, para detener la simulación, vaya Dios a saber por donde está la línea de ejecución de código pero comencemos de nuevo, resetea el micro como lo hiciste anteriormente, y ahora ve al menú...

Debug --> Run --> Step

o presiona el botón de los zapatitos, este...

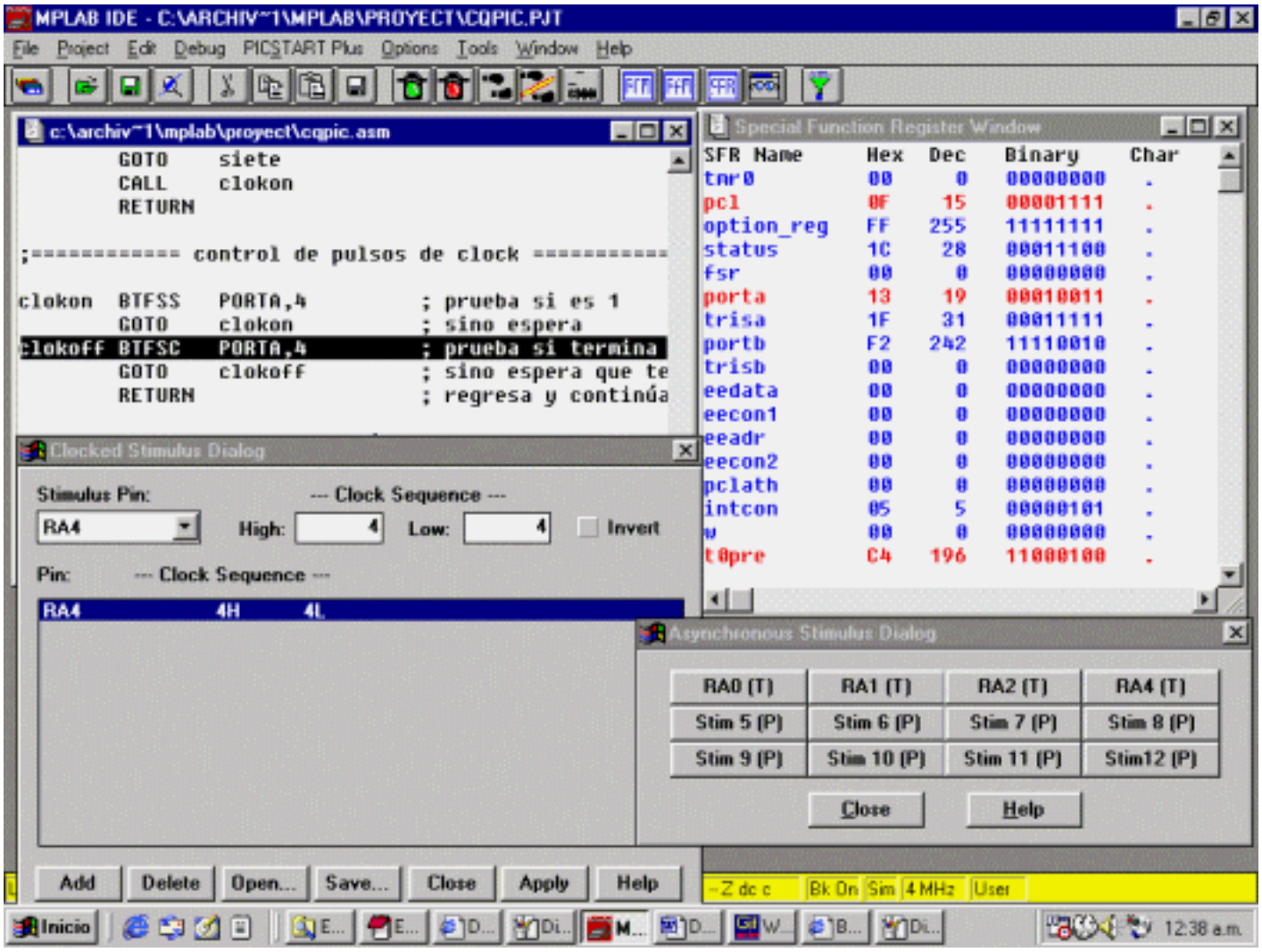


por cada vez que presiones este botón verás como avanza la ejecución del código línea por línea, ahora observa como se inicia la configuración de los puertos haciendo uso del registro **w**, en **trisa** para el Puerto A con los 5 primeros bits como entrada y **trisb** con los 8 bits como salida, bueno, eso fue para que veas esta forma de ejecución, si continúas así podrás ver también el quinto bit de porta que cambia con cada pulso del timer que programamos, a demás al estar los interruptores en cero, se ejecutará el efecto 1.

Bien, ahora que ya lo viste lo ejecutaremos de la forma que más me gusta, Resetea el micro y apunta al menú...

Debug --> Run --> Animate

Esto es un espectáculo, observa esta toma de pantalla...



En la ventana de código se está analizando el timer (RA4) en clokoff, en la ventana de Registros, aquello que está en rojo es lo último que se analizó y se ejecutó, observa...

porta 13 19 00010011

porta es el puerto A, **13** es el valor de los bits de este registro en hexadecimal, **19** es lo mismo pero en decimal, y finalmente **00010011** que es lo mismo pero en binario, bien, fíjate que los interruptores que ahora te los marcaré en azul **00010011** están en 0011 y el clock **00010011** en 1, por lo tanto se está ejecutando el cuarto efecto (efect4), recuerda que el primer efecto es con los interruptores en 0000 el segundo en 0001, el tercero en 0010 y el cuarto en 0011. y el timer está en nivel alto **(1)**

Por otro lado...

portb F2 242 11110010

es el estado de los bits del puerto B, esto quiere decir que están encendidos los LED's 1, 4, 5, 6 y 7 estos que te marco en rojo **11110010**

En fin, continúa simulando si deseas ver todos los efectos, recuerda que cada vez que presiones uno de los interruptores éste cambiará de estado de 0 a 1 o bien de 1 a 0.

Ahora experimentemos un poquito más...

:: PIC - Parte II - Capítulo 18

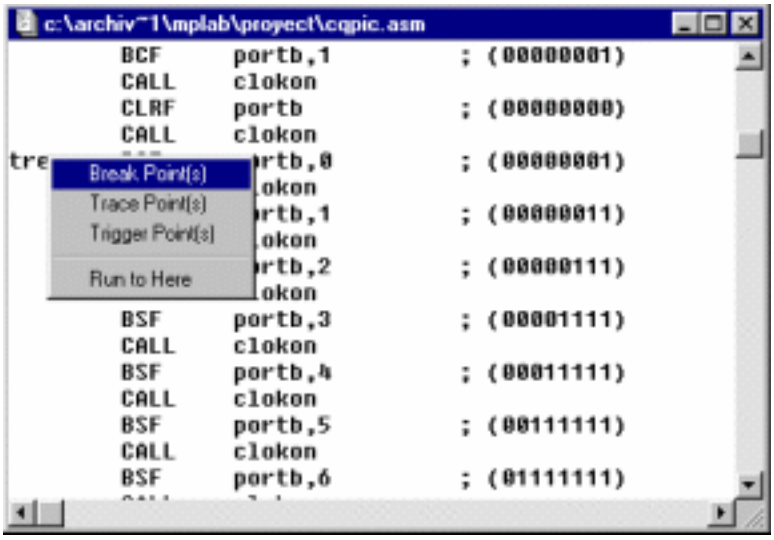
Break Points, La pila y la ventana File Register

Resetea el micro y luego ejecútalo en forma animada coloca los interruptores en 0000, así...

porta 10 16 00010000

No importa el estado del timer (ese 1 que apareció ahí), esto es sólo para asegurarnos que se ejecutará efect1 (es decir el primer efecto). Ahora detén la ejecución.

En la ventana de código ve a la etiqueta **trece** que se encuentra en **efect1**, casi en la mitad del código que corresponde a este efecto, coloca el cursor en esa línea y click con el botón derecho, verás un menú emergente alto así...



pues bien, selecciona **Break Point(s)** y toda esa línea se te pintará de rojo, lo que hicimos es colocar un punto de ruptura en esa línea, esto quiere decir que cuando se ejecute el código, este se detendrá en esa línea, ahora lo veremos en vivo, ve al menú...

Debug --> Run --> Run

esto es ejecutar el código a todo lo que da, y Oh sorpresa...!!!, la ejecución terminó en el punto de ruptura, bien, veamos que pasó en la ventana de Registros observa esto...

portb 00 0 00000000

las cosas están así, la instrucción...

CLRF portb ; (00000000)

es la que envió 00000000 a portb.

CALL clokon

fue la llamada al timer y cuando regresó se encontró con el punto de ruptura en ...

trece BSF portb,0 ; (00000001)

por lo tanto esta línea no se ejecutó, es por eso que el estado de portb es...

portb 00 0 00000000

ocurrirá lo mismo si lo ejecutas en forma animada, prueba y verás...

Bueno, ahora quitaremos este **Break Point**, ve al menú...

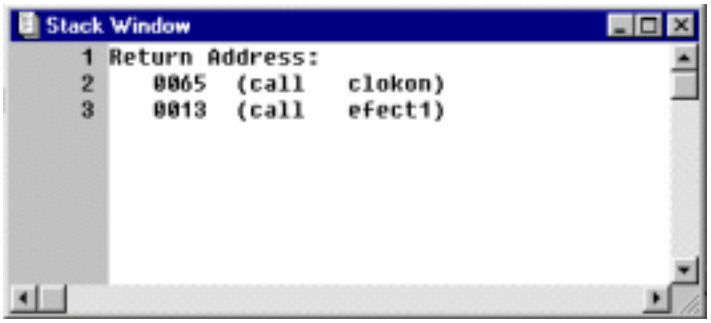
Debug --> Clear All Points

y verás un mensaje que dice **"Do you want to Clear all Break, trace, and trigger points?"** que en entendible seria, **"estas por quitar todos los puntos de rupturas, bla, bla, bla..."**, ...que inglés más raro el mío...!!!, pero bueno, le das a **si** y listo, ya lo quitamos.

otra cosa que puedes ver es como se van apilando los datos en la pila (**Stack**) (aquella de las 8 posiciones), vamos por ella, ve al menú...

Window --> Stack

y aquí la tienes ejecutándose...

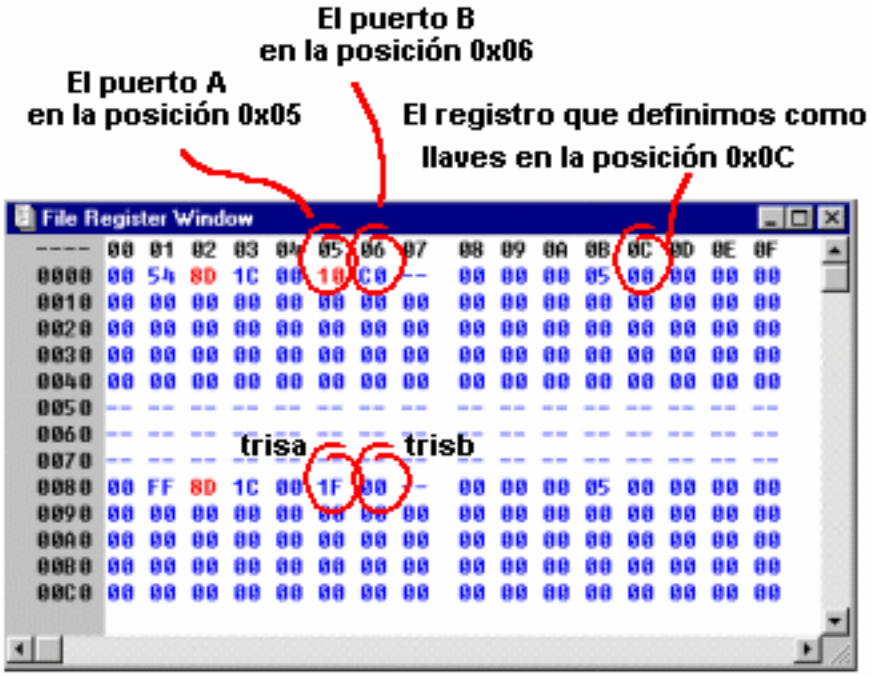


Observa como se carga y se descarga en plena ejecución...

Otra de las ventanas que podemos ver es File register, ve al menú...

Window --> File Register

aquí está...



Ejecuta el código en modo Animate y verás como cambian los registros.

Bueno ahora si, sólo quería mostrarte algunas de las opciones que tienes cuando ejecutas tu programa.

Espero que lo hayas disfrutado, y lo más importante, que hayas comprendido como hacer la simulación con MPLAB, ahora es tu turno, experimenta con MPLAB, tiene varias opciones las cuales te mostré al comenzar el tutorial, se que podrás hacer cosas más interesantes que estas y mucho más complejas por supuesto, lo que yo siempre hago es hacer una copia del proyecto y experimentar para aprender algo más, y preguntar sólo si no le encuentro solución a mi problema, lo más típico es errarle hasta que salga todo bien, creo que es la mejor forma de aprender, no crees...???

:: PIC - Parte II - Capítulo 19

Últimos Comentarios

Sólo resta cargar nuestro cqpic.hex en el pic, para hacerlo ve a la carpeta donde guardaste el proyecto, que seguramente debe estar en...

C --> Archivo de Programas --> MPLAB --> *Tu carpeta de proyectos*

Allí verás este archivo junto a otros más, pues es hora de cargar cqpic.hex en el micro, abre el programa que utilizas siempre y lo cargas, respecto a este tema ya lo describí en el tutorial anterior, no lo voy a repetir ahora.

Como dije anteriormente MPLAB es para codificar, simular, etc., etc., pero menos para cargar el programa en el pic, ya que todos tenemos nuestro propio grabador, y bueno... que si no tienes uno ya es hora que lo armes no crees...???

Lo que necesitas ahora es montar el circuito, así es que ve a la sección de proyectos, que ahí están todos los detalles, o haz click aquí, que más da...

Bien mis queridos amigos... Aquí damos por finalizado este tutorial, espero haberles sido de ayuda para aquello que estén a punto de emprender, dime si no valió la pena...???, viste que si...!!!

Nos veremos en la próxima y a ver si hacemos algo mejor, Suerte, y que Dios acompañe vuestros códigos, y las metidas de pata también, porque no? :o))

```
;====================== Encabezado ======================
;
; Ero-Pic // De Rueda Luis
; Secuenciador de 8 Canales y 16 efectos.
;
;========================================================

        LIST      P=16F84
        include "P16F84.inc"

;================== Mapa de Memoria =====================

estado  equ     0x03    ; Haciendo asignaciones
trisa   equ     0x05
trisb   equ     0x06
porta   equ     0x05
portb   equ     0x06

llaves  equ     0x0C    ; almacenara el estado de las llaves


;============== Configuración de puertos ================

        ORG     0x0000
        GOTO    inicio
        ORG     0x0005

inicio  BSF     estado,5        ; cambio al banco 1 del pic
        MOVLW   0x1f
        MOVWF   trisa           ; Asigna al puerto A como entrada
        MOVLW   0x00
        MOVWF   trisb           ; y Al puerto B como salida
        BCF     estado,5        ; Regresa al banco 0
        CLRF    porta           ; limpia el puerto A
        CLRF    portb           ; limpia el puerto B


swich   MOVF    porta,0         ; carga w con el puerto A
        ANDLW   0x0F            ; retiene los 4 bits de interes (las llaves)
        MOVWF   llaves          ; y los guarda en llaves
        XORLW   0x00            ; verifica si es el primer efecto
        BTFSC   estado,2        ; si es así
        CALL    efect1          ; lo llama y lo ejecuta
        MOVF    llaves,0        ; sino, carga llaves en w
        XORLW   0x01            ; y verifica si es el segundo efecto
        BTFSC   estado,2        ; si es así
        CALL    efect2          ; lo llama y lo ejecuta
        MOVF    llaves,0        ; y así con los demás
        XORLW   0x02            ; ya me aburrí
        BTFSC   estado,2        ; como verás el resto es lo mismo
        CALL    efect3
        MOVF    llaves,0
        XORLW   0x03
        BTFSC   estado,2
        CALL    efect4
        MOVF    llaves,0
        XORLW   0x04
        BTFSC   estado,2
        CALL    efect5
```

```
        MOVF    llaves,0
        XORLW   0x05
        BTFSC   estado,2
        CALL    efect6
        MOVF    llaves,0
        XORLW   0x06
        BTFSC   estado,2
        CALL    efect7
        MOVF    llaves,0
        XORLW   0x07
        BTFSC   estado,2
        CALL    efect8
        MOVF    llaves,0
        XORLW   0x08
        BTFSC   estado,2
        CALL    efect9
        MOVF    llaves,0
        XORLW   0x09
        BTFSC   estado,2
        CALL    efect10
        MOVF    llaves,0
        XORLW   0x0A
        BTFSC   estado,2
        CALL    efect11
        MOVF    llaves,0
        XORLW   0x0B
        BTFSC   estado,2
        CALL    efect12
        MOVF    llaves,0
        XORLW   0x0C
        BTFSC   estado,2
        CALL    efect13
        MOVF    llaves,0
        XORLW   0x0D
        BTFSC   estado,2
        CALL    efect14
        MOVF    llaves,0
        XORLW   0x0E
        BTFSC   estado,2
        CALL    efect15
        MOVF    llaves,0
        XORLW   0x0F
        BTFSC   estado,2
        CALL    efect16
        GOTO    swich           ; Comienza a revisar de nuevo

;======================== Efectos ========================

efect1  CLRF    portb           ; limpia el puerto B
        BSF     portb,7         ; (10000000)
        CALL    clokon
        BSF     portb,6         ; (11000000)
        CALL    clokon
        BSF     portb,5         ; (11100000)
        CALL    clokon
        BSF     portb,4         ; (11110000)
        CALL    clokon
```

```
        BSF     portb,3         ; (11111000)
        CALL    clokon
        BSF     portb,2         ; (11111100)
        CALL    clokon
        BSF     portb,1         ; (11111110)
        CALL    clokon
        BSF     portb,0         ; (11111111)
        CALL    clokon
        BCF     portb,7         ; (01111111)
        CALL    clokon
        BCF     portb,6         ; (00111111)
        CALL    clokon
        BCF     portb,5         ; (00011111)
        CALL    clokon
        BCF     portb,4         ; (00001111)
        CALL    clokon
        BCF     portb,3         ; (00000111)
        CALL    clokon
        BCF     portb,2         ; (00000011)
        CALL    clokon
        BCF     portb,1         ; (00000001)
        CALL    clokon
        CLRF    portb           ; (00000000)
        CALL    clokon
trece   BSF     portb,0         ; (00000001)
        CALL    clokon
        BSF     portb,1         ; (00000011)
        CALL    clokon
        BSF     portb,2         ; (00000111)
        CALL    clokon
        BSF     portb,3         ; (00001111)
        CALL    clokon
        BSF     portb,4         ; (00011111)
        CALL    clokon
        BSF     portb,5         ; (00111111)
        CALL    clokon
        BSF     portb,6         ; (01111111)
        CALL    clokon
        BSF     portb,7         ; (11111111)
        CALL    clokon
        BCF     portb,0         ; (11111110)
        CALL    clokon
        BCF     portb,1         ; (11111100)
        CALL    clokon
        BCF     portb,2         ; (11111000)
        CALL    clokon
        BCF     portb,3         ; (11110000)
        CALL    clokon
        BCF     portb,4         ; (11100000)
        CALL    clokon
        BCF     portb,5         ; (11000000)
        CALL    clokon
        BCF     portb,6         ; (10000000)
        CALL    clokon
        RETURN                  ; a revisar nuevamente las llaves

efect2  CLRF    portb           ; limpia el puerto B
```

```
BSF     portb,7         ; (10000000)            EFECTO ACUMULATIVO EN "0"
CALL    clokon
BCF     portb,7         ; (00000000)
BSF     portb,6         ; (01000000)
CALL    clokon
BCF     portb,6         ; (00000000)
BSF     portb,5         ; (00100000)
CALL    clokon
BCF     portb,5         ; (00000000)
BSF     portb,4         ; (00010000)
CALL    clokon
BCF     portb,4         ; (00000000)
BSF     portb,3         ; (00001000)
CALL    clokon
BCF     portb,3         ; (00000000)
BSF     portb,2         ; (00000100)
CALL    clokon
BCF     portb,2         ; (00000000)
BSF     portb,1         ; (00000010)
CALL    clokon
BCF     portb,1         ; (00000000)
BSF     portb,0         ; (00000001)
CALL    clokon
BSF     portb,7         ; (10000001)
CALL    clokon
BCF     portb,7         ; (00000001)
BSF     portb,6         ; (01000001)
CALL    clokon
BCF     portb,6         ; (00000001)
BSF     portb,5         ; (00100001)
CALL    clokon
BCF     portb,5         ; (00000001)
BSF     portb,4         ; (00010001)
CALL    clokon
BCF     portb,4         ; (00000001)
BSF     portb,3         ; (00001001)
CALL    clokon
BCF     portb,3         ; (00000001)
BSF     portb,2         ; (00000101)
CALL    clokon
BCF     portb,2         ; (00000001)
BSF     portb,1         ; (00000011)
CALL    clokon
BSF     portb,7         ; (10000011)
CALL    clokon
BCF     portb,7         ; (00000011)
BSF     portb,6         ; (01000011)
CALL    clokon
BCF     portb,6         ; (00000011)
BSF     portb,5         ; (00100011)
CALL    clokon
BCF     portb,5         ; (00000011)
BSF     portb,4         ; (00010011)
CALL    clokon
BCF     portb,4         ; (00000011)
BSF     portb,3         ; (00001011)
CALL    clokon
```

```
        BCF     portb,3         ; (00000011)
        BSF     portb,2         ; (00000111)
        CALL    clokon
        BSF     portb,7         ; (10000111)
        CALL    clokon
        BCF     portb,7         ; (00000111)
        BSF     portb,6         ; (01000111)
        CALL    clokon
        BCF     portb,6         ; (00000111)
        BSF     portb,5         ; (00100111)
        CALL    clokon
        BCF     portb,5         ; (00000111)
        BSF     portb,4         ; (00010111)
        CALL    clokon
        BCF     portb,4         ; (00000111)
        BSF     portb,3         ; (00001111)
        CALL    clokon
        BSF     portb,7         ; (10001111)
        CALL    clokon
        BCF     portb,7         ; (00001111)
        BSF     portb,6         ; (01001111)
        CALL    clokon
        BCF     portb,6         ; (00001111)
        BSF     portb,5         ; (00101111)
        CALL    clokon
        BCF     portb,5         ; (00001111)
        BSF     portb,4         ; (00011111)
        CALL    clokon
        BSF     portb,7         ; (10011111)
        CALL    clokon
        BCF     portb,7         ; (00011111)
        BSF     portb,6         ; (01011111)
        CALL    clokon
        BCF     portb,6         ; (00011111)
        BSF     portb,5         ; (00111111)
        CALL    clokon
        BSF     portb,7         ; (10111111)
        CALL    clokon
        BCF     portb,7         ; (00111111)
        BSF     portb,6         ; (01111111)
        CALL    clokon
        BSF     portb,7         ; (11111111)
        CALL    clokon
        RETURN

efect3  CLRF    portb           ; limpia el puerto B
        CALL    clokon
        BSF     portb,0         ; (00000001)
        CALL    clokon
        BCF     portb,0         ; (00000000)
        BSF     portb,1         ; (00000010)
        CALL    clokon
        BCF     portb,1         ; (00000000)
        BSF     portb,2         ; (00000100)
        CALL    clokon
        BCF     portb,2         ; (00000000)
        BSF     portb,3         ; (00001000)
```

```
CALL    clokon
BCF     portb,3         ; (00000000)
BSF     portb,4         ; (00010000)
CALL    clokon
BCF     portb,4         ; (00000000)
BSF     portb,5         ; (00100000)
CALL    clokon
BCF     portb,5         ; (00000000)
BSF     portb,6         ; (01000000)
CALL    clokon
BCF     portb,6         ; (00000000)
BSF     portb,7         ; (10000000)
CALL    clokon
BSF     portb,0         ; (10000001)
CALL    clokon
BCF     portb,0         ; (10000000)
BSF     portb,1         ; (10000010)
CALL    clokon
BCF     portb,1         ; (10000000)
BSF     portb,2         ; (10000100)
CALL    clokon
BCF     portb,2         ; (10000000)
BSF     portb,3         ; (10001000)
CALL    clokon
BCF     portb,3         ; (10000000)
BSF     portb,4         ; (10010000)
CALL    clokon
BCF     portb,4         ; (10000000)
BSF     portb,5         ; (10100000)
CALL    clokon
BCF     portb,5         ; (10000000)
BSF     portb,6         ; (11000000)
CALL    clokon
BSF     portb,0         ; (11000001)
CALL    clokon
BCF     portb,0         ; (11000000)
BSF     portb,1         ; (11000010)
CALL    clokon
BCF     portb,1         ; (11000000)
BSF     portb,2         ; (11000100)
CALL    clokon
BCF     portb,2         ; (11000000)
BSF     portb,3         ; (11001000)
CALL    clokon
BCF     portb,3         ; (11000000)
BSF     portb,4         ; (11010000)
CALL    clokon
BCF     portb,4         ; (11000000)
BSF     portb,5         ; (11100000)
CALL    clokon
BSF     portb,0         ; (11100001)
CALL    clokon
BCF     portb,0         ; (11100000)
BSF     portb,1         ; (11100010)
CALL    clokon
BCF     portb,1         ; (11100000)
BSF     portb,2         ; (11100100)
```

```
        CALL    clokon
        BCF     portb,2         ; (11100000)
        BSF     portb,3         ; (11101000)
        CALL    clokon
        BCF     portb,3         ; (11100000)
        BSF     portb,4         ; (11110000)
        CALL    clokon
        BSF     portb,0         ; (11110001)
        CALL    clokon
        BCF     portb,0         ; (11110000)
        BSF     portb,1         ; (11110010)
        CALL    clokon
        BCF     portb,1         ; (11110000)
        BSF     portb,2         ; (11110100)
        CALL    clokon
        BCF     portb,2         ; (11110000)
        BSF     portb,3         ; (11111000)
        CALL    clokon
        BSF     portb,0         ; (11111001)
        CALL    clokon
        BCF     portb,0         ; (11111000)
        BSF     portb,1         ; (11111010)
        CALL    clokon
        BCF     portb,1         ; (11111000)
        BSF     portb,2         ; (11111100)
        CALL    clokon
        BSF     portb,0         ; (11111101)
        CALL    clokon
        BCF     portb,0         ; (11111100)
        BSF     portb,1         ; (11111110)
        CALL    clokon
        BSF     portb,0         ; (11111111)
        CALL    clokon
        RETURN

efect4  CLRF    portb           ; limpia el puerto B
        BSF     portb,7         ; (10000000)
        BSF     portb,0         ; (10000001)
        CALL    clokon
        BSF     portb,6         ; (11000001)
        BSF     portb,1         ; (11000011)
        CALL    clokon
        BSF     portb,5         ; (11100011)
        BSF     portb,2         ; (11100111)
        CALL    clokon
        BSF     portb,4         ; (11110111)
        BSF     portb,3         ; (11111111)
        CALL    clokon
        BCF     portb,7         ; (01111111)
        BCF     portb,0         ; (01111110)
        CALL    clokon
        BCF     portb,6         ; (00111110)
        BCF     portb,1         ; (00111100)
        CALL    clokon
        BCF     portb,5         ; (00011100)
        BCF     portb,2         ; (00011000)
        CALL    clokon
```

```
        BCF     portb,4         ; (00001000)
        BCF     portb,3         ; (00000000)
        CALL    clokon
        CALL    clokon
        BSF     portb,3         ; (00001000)
        BSF     portb,4         ; (00011000)
        CALL    clokon
        BSF     portb,5         ; (00111000)
        BSF     portb,2         ; (00111100)
        CALL    clokon
        BSF     portb,6         ; (01111100)
        BSF     portb,1         ; (01111110)
        CALL    clokon
        BSF     portb,7         ; (11111110)
        BSF     portb,0         ; (11111111)
        CALL    clokon
        BCF     portb,3         ; (11110111)
        BCF     portb,4         ; (11100111)
        CALL    clokon
        BCF     portb,5         ; (11000111)
        BCF     portb,2         ; (11000011)
        CALL    clokon
        BCF     portb,6         ; (10000011)
        BCF     portb,1         ; (10000001)
        CALL    clokon
        BCF     portb,7         ; (00000001)
        BCF     portb,0         ; (00000000)
        CALL    clokon
        RETURN

efect5  CLRF    portb           ; limpia el puerto B
        MOVLW   0x01            ; comienza con (00000001)
        MOVWF   portb           ; lo envía a la salida
cinco   BSF     estado,0        ; pone a 0 el bit C de status (carry)
        CALL    clokon
        MOVWF   portb           ; lo envía a la salida
        RLF     portb,0         ; rota a la izquierda y pasa el valor a W
        MOVWF   portb           ; lo envía a la salida
        CALL    clokon
        CLRF    portb           ; (00000000)
        CALL    clokon
        MOVWF   portb           ; repite
        CALL    clokon
        CLRF    portb           ; (00000000)
        CALL    clokon
        MOVWF   portb           ; lo envía a la salida
        BTFSS   portb,7         ; ve si terminó de rotar
        GOTO    cinco
        CALL    clokon
        BCF     portb,7         ; (01111111)
        CALL    clokon
        BCF     portb,6         ; (00111111)
        CALL    clokon
        BCF     portb,5         ; (00011111)
        CALL    clokon
        BCF     portb,4         ; (00001111)
        CALL    clokon
```

```
        BCF     portb,3         ; (00000111)
        CALL    clokon
        BCF     portb,2         ; (00000011)
        CALL    clokon
        BCF     portb,1         ; (00000001)
        CALL    clokon
        RETURN


efect6  CLRF    portb           ; limpia el puerto B
        MOVLW   0x01            ; comienza con (00000001)
        MOVWF   portb           ; lo envía a la salida
tres    BCF     estado,0        ; pone a 0 el bit C de status (carry)
        CALL    clokon
        MOVWF   portb           ; lo envía a la salida
        RLF     portb,0         ; rota a la derecha y pasa el valor a W
        MOVWF   portb           ; lo envía a la salida
        CALL    clokon
        CLRF    portb           ; (00000000)
        CALL    clokon
        MOVWF   portb           ; repite
        CALL    clokon
        CLRF    portb           ; (00000000)
        CALL    clokon
        MOVWF   portb           ; lo envía a la salida
        BTFSS   portb,7         ; ve si terminó de rotar
        GOTO    tres
        CALL    clokon
        RETURN

efect7  CLRF    portb           ; limpia el puerto B
        CALL    clokon
        BSF     portb,0         ; (00000001)
        CALL    clokon
        BCF     portb,0         ; (00000000)
        BSF     portb,1         ; (00000010)
        CALL    clokon
        BCF     portb,1         ; (00000000)
        BSF     portb,2         ; (00000100)
        CALL    clokon
        BCF     portb,2         ; (00000000)
        BSF     portb,3         ; (00001000)
        CALL    clokon
        BCF     portb,3         ; (00000000)
        BSF     portb,4         ; (00010000)
        CALL    clokon
        BCF     portb,4         ; (00000000)
        BSF     portb,5         ; (00100000)
        CALL    clokon
        BCF     portb,5         ; (00000000)
        BSF     portb,6         ; (01000000)
        CALL    clokon
        BCF     portb,6         ; (00000000)
        BSF     portb,7         ; (10000000)
        CALL    clokon
        BSF     portb,6         ; (11000000)
        CALL    clokon
```

```
BCF     portb,6         ; (10000000)
BSF     portb,5         ; (10100000)
CALL    clokon
BCF     portb,5         ; (10000000)
BSF     portb,4         ; (10010000)
CALL    clokon
BCF     portb,4         ; (10000000)
BSF     portb,3         ; (10001000)
CALL    clokon
BCF     portb,3         ; (10000000)
BSF     portb,2         ; (10000100)
CALL    clokon
BCF     portb,2         ; (10000000)
BSF     portb,1         ; (10000010)
CALL    clokon
BCF     portb,1         ; (10000000)
BSF     portb,0         ; (10000001)
CALL    clokon
BSF     portb,1         ; (10000011)
CALL    clokon
BCF     portb,1         ; (10000001)
BSF     portb,2         ; (10000101)
CALL    clokon
BCF     portb,2         ; (10000001)
BSF     portb,3         ; (10001001)
CALL    clokon
BCF     portb,3         ; (10000001)
BSF     portb,4         ; (10010001)
CALL    clokon
BCF     portb,4         ; (10000001)
BSF     portb,5         ; (10100001)
CALL    clokon
BCF     portb,5         ; (10000001)
BSF     portb,6         ; (11000001)
CALL    clokon
BSF     portb,5         ; (11100001)
CALL    clokon
BCF     portb,5         ; (11000001)
BSF     portb,4         ; (11010001)
CALL    clokon
BCF     portb,4         ; (11000001)
BSF     portb,3         ; (11001001)
CALL    clokon
BCF     portb,3         ; (11000001)
BSF     portb,2         ; (11000101)
CALL    clokon
BCF     portb,2         ; (11000001)
BSF     portb,1         ; (11000011)
CALL    clokon
BSF     portb,2         ; (11000111)
CALL    clokon
BCF     portb,2         ; (11000011)
BSF     portb,3         ; (11001011)
CALL    clokon
BCF     portb,3         ; (11000011)
BSF     portb,4         ; (11010011)
CALL    clokon
```

```
        BCF     portb,4         ; (11000011)
        BSF     portb,5         ; (11100011)
        CALL    clokon
        BSF     portb,4         ; (11110011)
        CALL    clokon
        BCF     portb,4         ; (11100011)
        BSF     portb,3         ; (11101011)
        CALL    clokon
        BCF     portb,3         ; (11100011)
        BSF     portb,2         ; (11100111)
        CALL    clokon
        BSF     portb,3         ; (11101111)
        CALL    clokon
        BCF     portb,3         ; (11100111)
        BSF     portb,4         ; (11110111)
        CALL    clokon
        BSF     portb,3         ; (11111111)
        CALL    clokon
        BCF     portb,2         ; (11111011)
        CALL    clokon
        BSF     portb,2         ; (11111111)
        CALL    clokon
        BCF     portb,1         ; (11111101)
        CALL    clokon
        BSF     portb,1         ; (11111111)
        CALL    clokon
        BCF     portb,0         ; (11111110)
        CALL    clokon
        BSF     portb,0         ; (11111111)
        CALL    clokon
        BCF     portb,0         ; (11111110)
        CALL    clokon
        BCF     portb,1         ; (11111100)
        CALL    clokon
        BCF     portb,2         ; (11111000)
        CALL    clokon
        BCF     portb,3         ; (11110000)
        CALL    clokon
        BCF     portb,4         ; (11100000)
        CALL    clokon
        BCF     portb,5         ; (11000000)
        CALL    clokon
        BCF     portb,6         ; (10000000)
        CALL    clokon
        BCF     portb,7         ; (00000000)
        CALL    clokon
        RETURN

efect8  CALL    efect3          ; combinan el efecto 3
        CALL    efect2          ; con el efecto 2
        RETURN

efect9  CLRF    portb           ; limpia el puerto B
        MOVLW   0xEE            ; comienza con (11101110)
        MOVWF   portb           ; lo pasa a portb
        BSF     estado,0        ; pone el carry a 1
rotar   CALL    clokon
```

```
        RLF     portb,1         ; inicia la rotación
        BTFSC   portb,7         ; ve si terminó de rotar
        GOTO    rotar           ; sino continúa
        CALL    clokon
        RETURN                  ; terminó, ver si cambió efecto

efect10 CLRF    portb           ; limpia el puerto B
        MOVLW   0xFE            ; comienza con (11111110)
        MOVWF   portb           ; lo pasa a portb
        BSF     estado,0        ; pone el carry a 1
rotar1  CALL    clokon
        RLF     portb,1         ; inicia la rotación
        BTFSC   portb,7         ; ve si terminó de rotar
        GOTO    rotar1          ; sino continúa
rotar2  CALL    clokon
        RRF     portb,1         ; ahora rota al revés
        BTFSC   portb,0         ; ve si terminó de rotar
        GOTO    rotar2          ; sino continúa
        CALL    clokon
        RETURN                  ; terminó, ver si cambió efecto

efect11 CLRF    portb           ; limpia el puerto B
        BSF     portb,0         ; (00000001)
        CALL    clokon
        BSF     portb,1         ; (00000011)
        CALL    clokon
        BSF     portb,2         ; (00000111)
        CALL    clokon
        BSF     portb,3         ; (00001111)
        CALL    clokon
        BSF     portb,4         ; (00011111)
        CALL    clokon
        BSF     portb,5         ; (00111111)
        CALL    clokon
        BSF     portb,6         ; (01111111)
        CALL    clokon
        BSF     portb,7         ; (11111111)
        BCF     estado,0        ; pone el carry a 0
uno     CALL    clokon
        RRF     portb,1         ; rotará uno apagado
        BTFSC   portb,0         ; ve si es (11111110)
        GOTO    uno             ; sino continúa
        CALL    clokon
        BCF     portb,1         ; (11111100)
        CALL    clokon
        BCF     portb,2         ; (11111000)
        CALL    clokon
        BCF     portb,3         ; (11110000)
        CALL    clokon
        BCF     portb,4         ; (11100000)
        CALL    clokon
        BCF     portb,5         ; (11000000)
        CALL    clokon
        BCF     portb,6         ; (10000000)
        BCF     estado,0        ; pone el carry a 0
dos     CALL    clokon
        RRF     portb,1         ; rotará uno encendido
```

```
        BTFSS   portb,0         ; ve si es (00000001)
        GOTO    dos             ; sino continúa
        CALL    clokon
        RETURN                  ; terminó, ver si cambió efecto

efect12 CLRF    portb           ; limpia el puerto B
        BSF     portb,0         ; (00000001)
        CALL    clokon
        BSF     portb,1         ; (00000011)
        CALL    clokon
        BSF     portb,2         ; (00000111)
        CALL    clokon
        BSF     portb,3         ; (00001111)
        CALL    clokon
        BSF     portb,4         ; (00011111)
        CALL    clokon
        BSF     portb,5         ; (00111111)
        CALL    clokon
        BSF     portb,6         ; (01111111)
        CALL    clokon
        BSF     portb,7         ; (11111111)
        CALL    clokon
        CLRF    portb           ; (00000000)
        CALL    clokon
        MOVLW   0xFF
        MOVWF   portb           ; (11111111) enciendo todo
        CALL    clokon
        CLRF    portb           ; (00000000)
        CALL    clokon
        MOVLW   0xFF
        MOVWF   portb           ; (11111111) enciendo todo
        CALL    clokon
        CLRF    portb           ; (00000000)
        CALL    clokon
        MOVLW   0xFF
        MOVWF   portb           ; (11111111) enciendo todo
        CALL    clokon
        CLRF    portb           ; (00000000)
        CALL    clokon
        MOVLW   0xFF
        MOVWF   portb           ; (11111111) enciendo todo
        CALL    clokon
        BCF     portb,7         ; (01111111)
        CALL    clokon
        BCF     portb,6         ; (00111111)
        CALL    clokon
        BCF     portb,5         ; (00011111)
        CALL    clokon
        BCF     portb,4         ; (00001111)
        CALL    clokon
        BCF     portb,3         ; (00000111)
        CALL    clokon
        BCF     portb,2         ; (00000011)
        CALL    clokon
        BCF     portb,1         ; (00000001)
        CALL    clokon
        RETURN
```

```
efect13 CLRF    portb           ; limpia el puerto B
        CALL    trece           ; ejecuta parte del efecto 1
        RETURN

efect14 CLRF    portb           ; limpia el puerto B
        BSF     portb,7         ; (10000000)    EFECTO ENCIENDE DE AFUERA AL CENTRO
        BSF     portb,0         ; (10000001)-----
        CALL    clokon
        CLRF    portb
        BSF     portb,6         ; (01000000)
        BSF     portb,1         ; (01000010)-----
        CALL    clokon
        CLRF    portb
        BSF     portb,5         ; (00100000)
        BSF     portb,2         ; (00100100)-----
        CALL    clokon
        CLRF    portb
        BSF     portb,4         ; (00010000)
        BSF     portb,3         ; (00011000)-----
        CALL    clokon
        BSF     portb,5         ; (00111000)
        BSF     portb,2         ; (00111100)
        CALL    clokon
        BSF     portb,6         ; (01111100)
        BSF     portb,1         ; (01111110)
        CALL    clokon
        BSF     portb,7         ; (11111110)
        BSF     portb,0         ; (11111111)
        CALL    clokon
        BCF     portb,3         ; (11110111)
        BCF     portb,4         ; (11100111)
        CALL    clokon
        BCF     portb,5         ; (11000111)
        BCF     portb,2         ; (11000011)
        CALL    clokon
        BCF     portb,6         ; (10000011)
        BCF     portb,1         ; (10000001)
        CALL    clokon
        BCF     portb,7         ; (00000001)
        BCF     portb,0         ; (00000000)
        CALL    clokon
        RETURN

efect15 CLRF    portb           ; limpia el puerto B
        MOVLW   0x80            ; comienza con (10000000)
        MOVWF   portb           ; lo envía a la salida
cuatro  BCF     estado,0        ; pone a 0 el bit C de status (carry)
        CALL    clokon
        MOVWF   portb           ; lo envía a la salida
        RRF     portb,0         ; rota a la izquierda y pasa el valor a W
        MOVWF   portb           ; lo envía a la salida
        CALL    clokon
        CLRF    portb           ; (00000000)
        CALL    clokon
        MOVWF   portb           ; repite
        CALL    clokon
```

```
        CLRF    portb           ; (00000000)
        CALL    clokon
        MOVWF   portb           ; lo envía a la salida
        BTFSS   portb,0         ; ve si terminó de rotar
        GOTO    cuatro
        CALL    clokon
        RETURN

efect16 CLRF    portb           ; limpia el puerto B
        BSF     portb,7         ; (10000000)
        BCF     estado,0        ; pone a 0 el bit C de status (el 1º bit)
seis    CALL    clokon
        RRF     portb,1         ; rota a la derecha
        BTFSS   portb,0         ; ve si terminó de rotar
        GOTO    seis
        CLRF    portb           ; (00000000)
        BSF     portb,0         ; (00000001)
        BCF     estado,0        ; pone el carry a 0
siete   CALL    clokon
        RLF     portb,1         ; rota a la izquierda
        BTFSS   portb,7         ; ve si terminó de rotar
        GOTO    siete
        CALL    clokon
        RETURN

;============ control de pulsos de clock ==================

clokon  BTFSS   PORTA,4         ; prueba si es 1
        GOTO    clokon          ; sino espera
clokoff BTFSC   PORTA,4         ; prueba si termina el pulso
        GOTO    clokoff         ; sino espera que termine
        RETURN                  ; regresa y continúa

;======================== final ==========================

        END
```