Ejemplo de utilización del programador universal HI-LO ALL 100 con el microcontrolador Microchip PIC 16F690 y el compilador CCS PCWH 3.249

Versión 1.0

Autor: Marcos Morales Pallarés

Escola Universitària d'Enginyeria Tècnica Industrial de Barcelona
Laboratorio de Proyectos. Unidad de Electrónica
Universitat Politècnica de Catalunya

Barcelona, noviembre del 2006

Índice

1. Objetivo de este documento

El programador universal ALL-100 de la firma HI-LO está disponible en el laboratorio de PFCs para aquellos/as proyectistas que lo necesiten. El gran número de circuitos integrados que permite programar lo convierte en una herramienta muy versátil.

En muchas ocasiones los estudiantes necesitan volcar programas en microcontroladores, grabar datos en ROMs o programar PLDs. Estos y otros dispositivos (FPGAs…) pueden ser programados con el ALL-100. Sólo precisamos seleccionar el integrado a emplear, generar el archivo en formato hexadecimal que queramos volcar (mediante un compilador en el caso de los microcontroladores, editores hexadecimales para las ROMs o herramientas de diseño VHDL para las PLDs y FPGAs) y programar el integrado.

En el caso de no disponer de los drivers necesarios para programar algún chip, siempre podremos consultar la web del fabricante (http://www.hilo-europe.com/html/all_100.htm) y comprobar la existencia de actualizaciones y soporte para nuevos dispositivos.

Para que sirva de pauta y a modo de ejemplo, proporcionamos al lector este manual. En él se recogen los pasos necesarios para llevar a cabo el montaje más básico que existe para todo microcontrolador: el control de encendido de un LED.

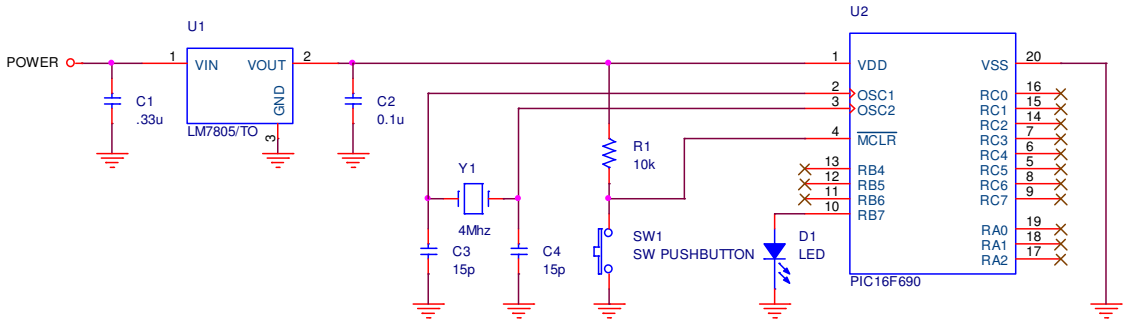
Para ello se ha escogido el PIC16F690, de la casa Microchip. La familia 16FXXX es reconocida por su versatilidad, facilidad de uso y robustez. Nos hemos decantado por el compilador CCS PCWH, pues se trata de una de las herramientas más empleadas al programar código en lenguaje C para los PIC, además de poseer gran cantidad de librerías y ejemplos.

El documento está estructurado a modo de 'guía de pasos' a realizar para llevar a cabo este ejemplo en concreto con éxito. Dada la naturaleza de este documento, no se ha creído conveniente entrar en detalles de funcionamiento.

Tomando este manual como punto de partida, no debería resultar difícil para el lector realizar procesos similares para otros microcontroladores. Obviamente, si lo que el lector precisa es volcar datos a otros circuitos integrados como por ejemplo ROMs o FPGAs, deberá emplear otras herramientas software, aunque podrá guiarse de la sección dedicada exclusivamente al programador ALL-100 y el volcado de datos, teniendo en cuenta que algunos de los parámetros aquí expuestos variarán.

2. Aplicación de ejemplo a montar y programar

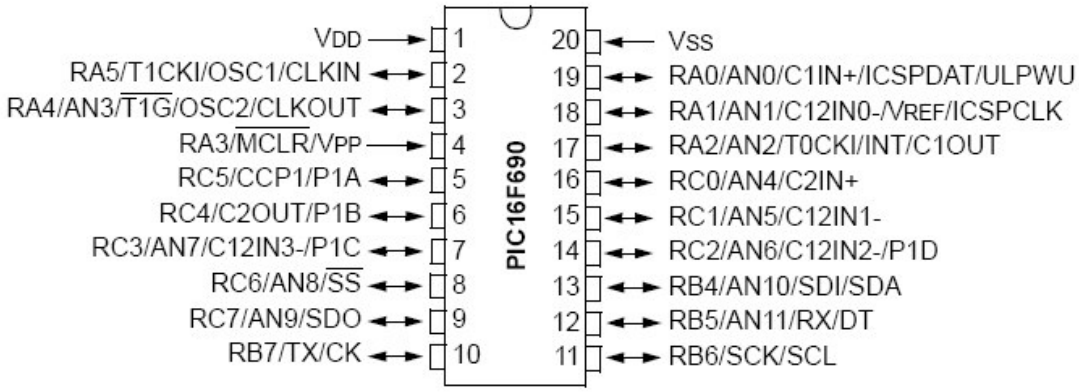
La siguiente figura muestra el circuito que implementaremos en nuestra protoboard.



Aspectos a observar:

- Alimentación regulada por un 7805 a 5 V. Los condensadores de desacoplo de 0.33 uF y 0.1 uF resultan imprescindibles.
- Empleo de un cristal de cuarzo de 4 MHz. En general se trata de la velocidad mínima empleada en todo montaje con microcontrolador.
- Botón de reset.
- LED conectado al pin RB7. Obsérvese la conexión directa al pin del micro, sin emplear ninguna resistencia limitadora de corriente. En general los PIC proporcionan la corriente necesaria para alimentar un diodo LED de forma directa. Esto nos sirve para nuestro ejemplo, aunque en la práctica se recomienda el uso de dichas resistencias.
- No hemos incluído un condensador de desacoplo para el micro, aunque para aplicaciones más complejas puede llegar a ser imprescindible.

A continuación se muestra la distribución de pines proporcionada en el datasheet del fabricante. Consúltese en caso de tener alguna duda.



3. Procedimiento

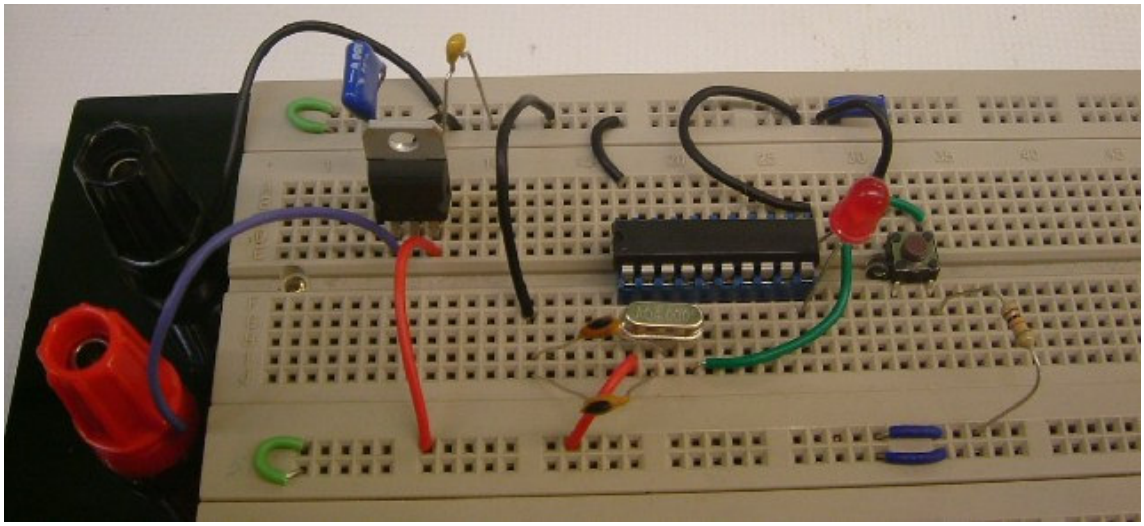
Los subapartados que se exponen a continuación deben seguirse de forma lineal.

No hace falta decir que, en el caso de tener ya instalado el compilador CCS PCWH o el software del programador universal ALL-100 podremos prescindir de los apartados 3.2 y 3.3, respectivamente.

Dentro de cada apartado observaremos que los pasos se encuentran numerados. En cada uno de ellos se incluye una breve explicación de lo que se debe realizar o, en su defecto, una imagen explicativa por sí sola. Los apartados se encuentran separados por barras negras horizontales.

3.1. Montaje del circuito en protoboard

1. El aspecto del circuito una vez montado es el siguiente:

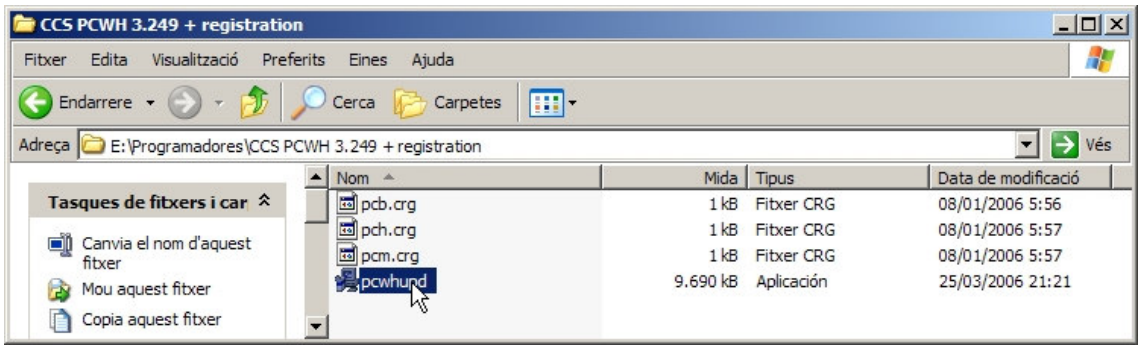


Obsérvese el empleo de un zócalo de 20 pines para nuestro micro. Resulta muy recomendable su uso puesto que al programar varias veces el micro y comprobar su funcionamiento estaremos insertándolo y extrayéndolo de la protoboard continuamente, con lo que podríamos dañar físicamente alguna de sus patitas.

Nuestro objetivo será volcar y ejecutar un programa que encienda y apague el LED de forma intermitente e indefinida. El porqué de realizar un montaje y un programa tan simple es porque nos permite comprobar de un modo sencillo que el proceso de compilación y volcado se ha realizado de forma correcta.

3.2. Instalación de CCS PCWH 3.249

1. Ejecutar 'pcwhund.exe':

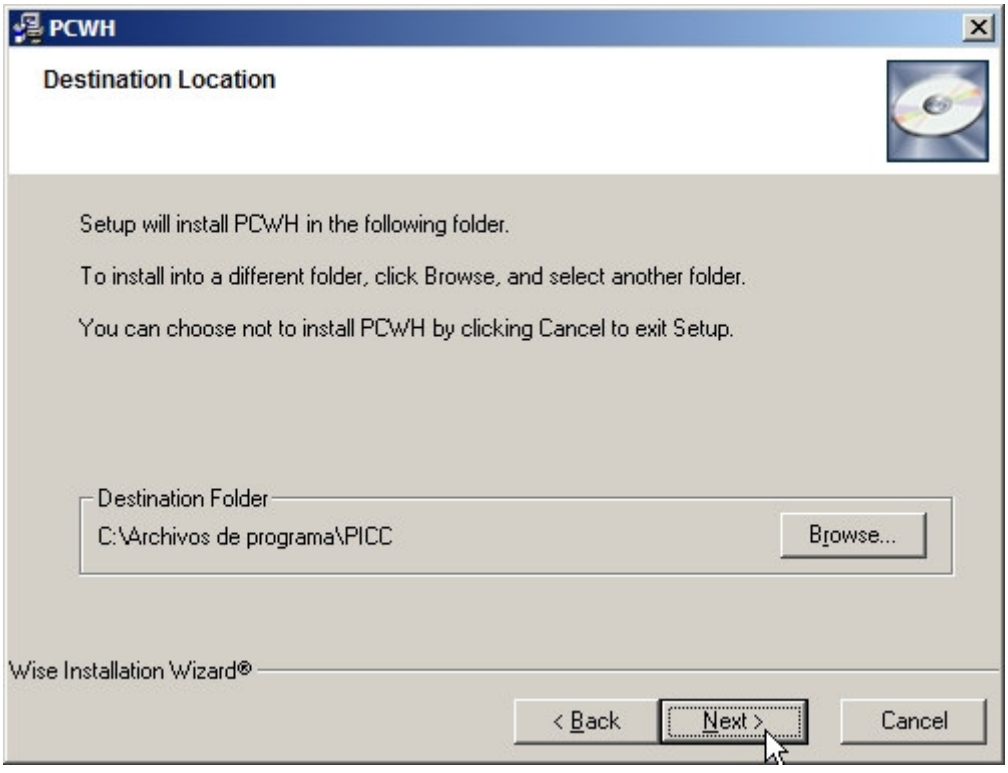


---

2. Clic en 'Next' > Clic en 'Next' > Clic en 'Next'

---

3. Seleccionar la carpeta destino para instalar el programa:

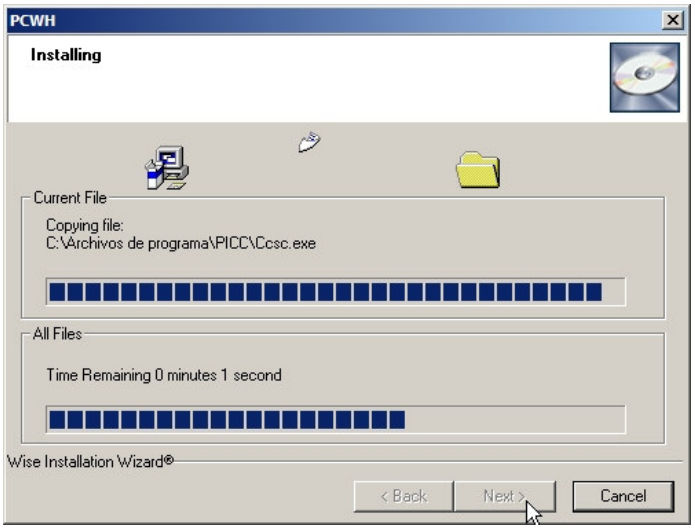


---

4. Clic en 'Next'

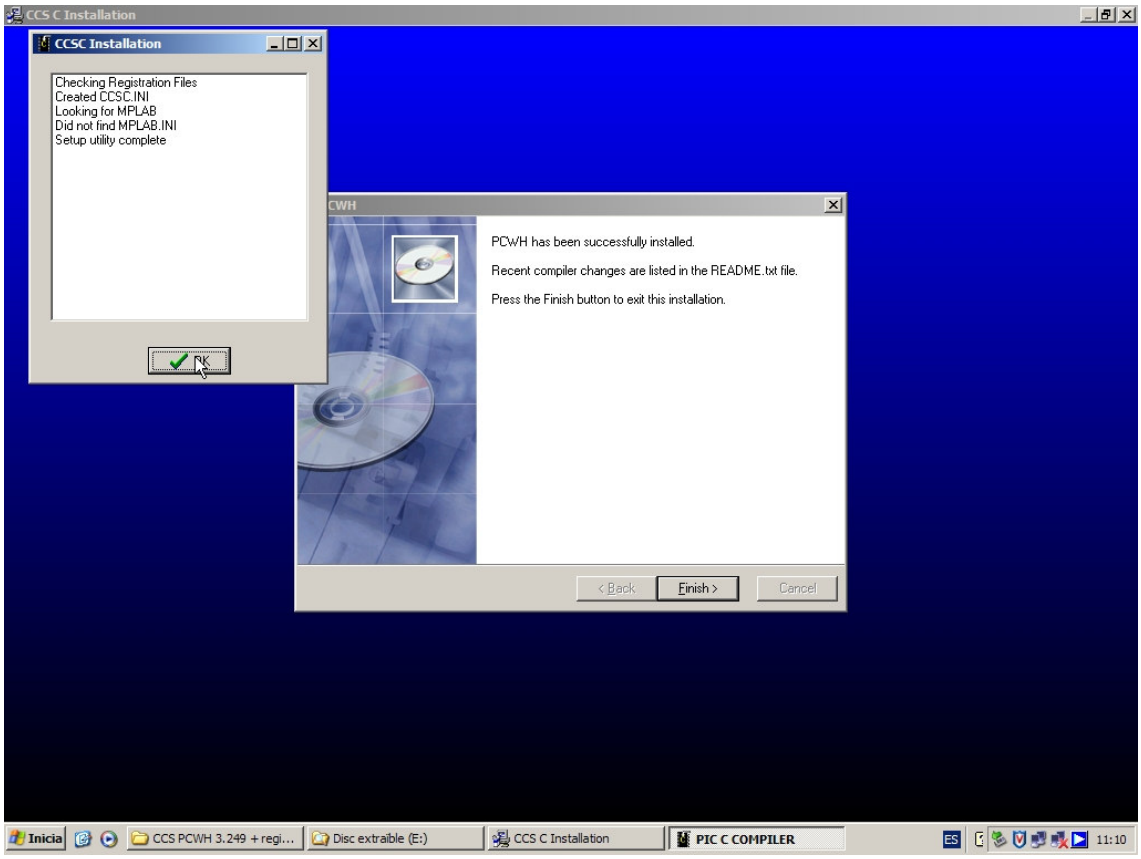
---

5. Esperar a que acabe la copia de archivos:



---

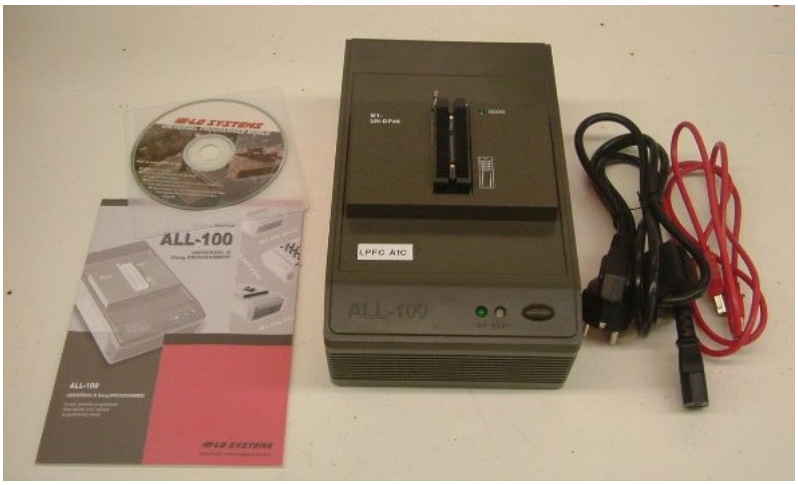
6. Si todo ha ido bien, aparecerán los siguientes cuadros informativos:



Con lo que finalizamos con la instalación de CCS PCWH 3.249.

3.3. Instalación del ALL-100

1. El programador universal ALL-100 se compone de los siguientes elementos:
    - Módulo programador.
    - Manual de instalación/utilización.
    - CD (drivers + software de programación + manuales electrónicos).
    - Cable USB.
    - Cable de alimentación.



En caso de faltar algún elemento será preciso avisar a uno de los responsables del laboratorio.

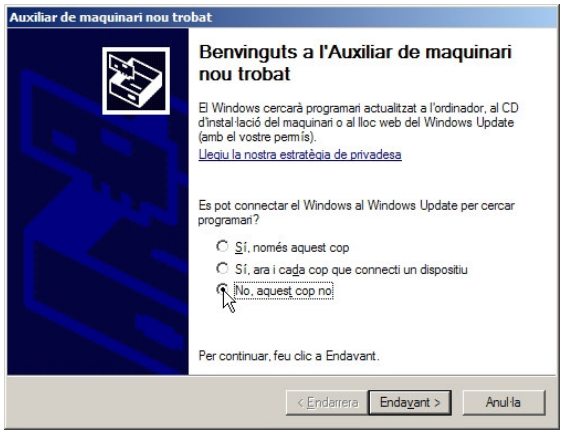
---

2. En primer lugar nos aseguraremos de que el interruptor de encendido del módulo programador está apagado.

3.
- Conectamos el cable USB al dispositivo y al PC.
- Conectamos el cable de alimentación del módulo.
- Finalmente encendemos el aparato.

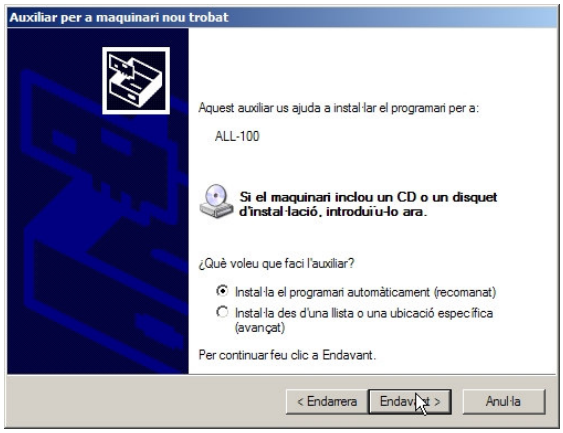
---

4. Aparecerá la siguiente pantalla (Windows XP SP2):



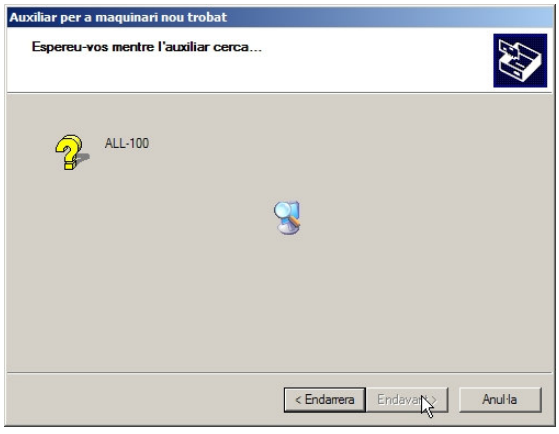
Clicamos en 'No' y continuamos con el asistente de nuevo hardware encontrado.

5.



Introducimos el CD de instalación y continuamos con el asistente.
Es posible que Windows inicie el programa autoarrancable del CD. No lo cerraremos, pues luego instalaremos el software para poder utilizar el ALL-100 (paso 10).

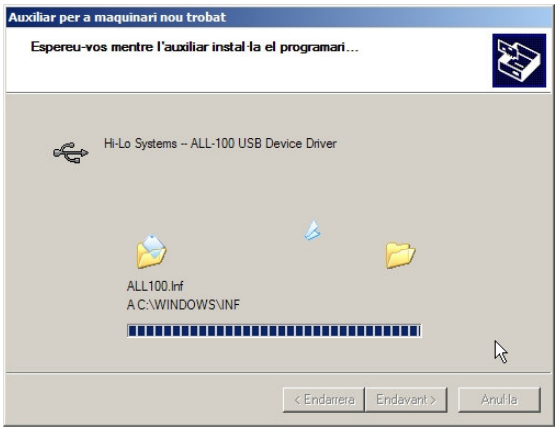
6.



---

7. Clicamos en ‘Continuar igualmente’.

8.

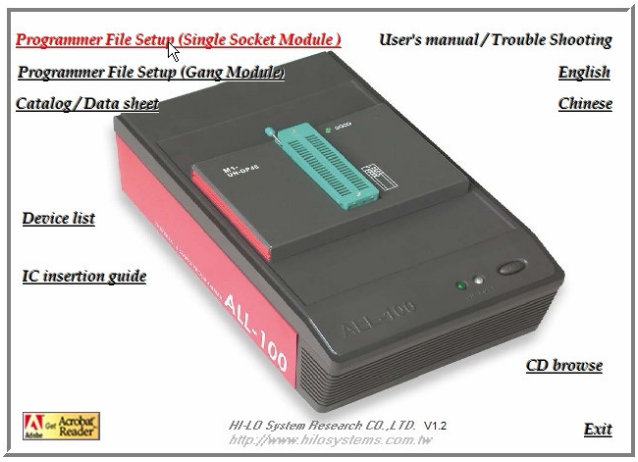


---

9. Finalizamos el asistente.

---

10. Desde la pantalla del menú autoarrancable del CD clicamos en ‘Programmer File Setup (Single Socket Module)’. En caso de no haberse iniciado dicho menú, lo encontraremos en la raíz del disco: autorun.exe.



---

11. Al iniciarse el programa de instalación, clicaremos en 'siguiente' sobre el mensaje de bienvenida.

---

12. Seleccionamos la carpeta destino donde instalaremos el software.

---

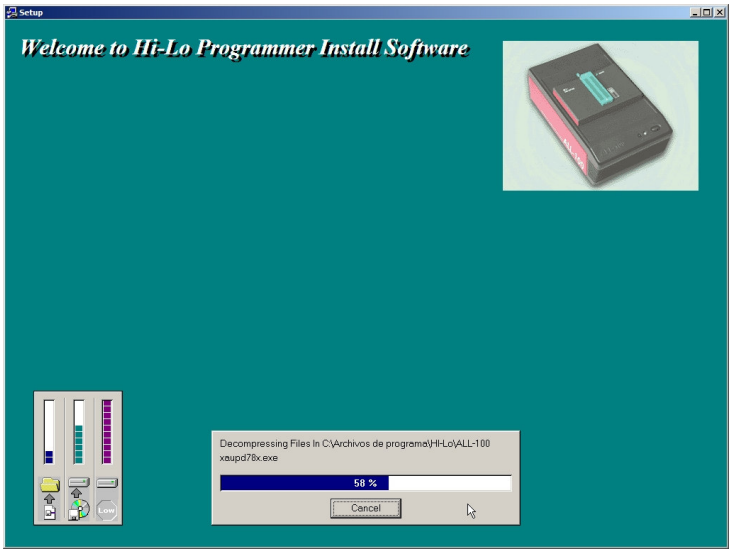
13. Seleccionamos la carpeta del menú inicio donde ubicaremos nuestros accesos directos.

---

14. Clic en siguiente.

---

15. Esperamos a que finalice la copia de archivos.



---

16. Finalizamos el asistente. El ordenador reiniciará.

3.4. Creación del programa a ejecutar en el PIC16F690

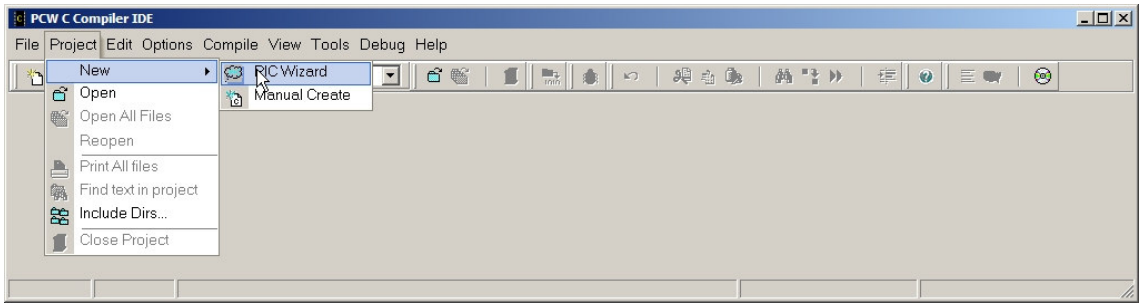
1. Iniciamos el CCS PCW PIC C Compiler:

Inicio > Programas > PIC-C > PIC C Compiler

---

2. Ejecutamos el asistente para la creación de proyectos:

Project > New > PIC Wizard

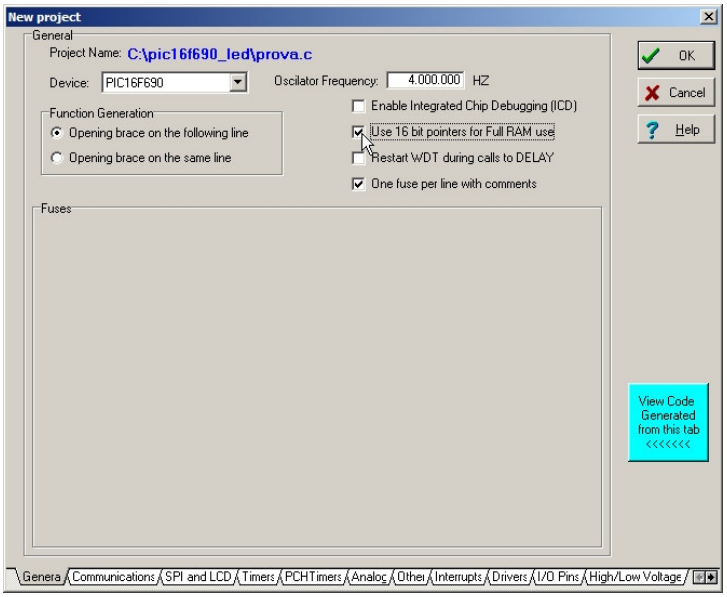


---

3. Se nos preguntará por el nombre y la ubicación de nuestro proyecto. Es recomendable dedicar un directorio exclusivamente al proyecto, pues el número de archivos generados puede ser considerable.

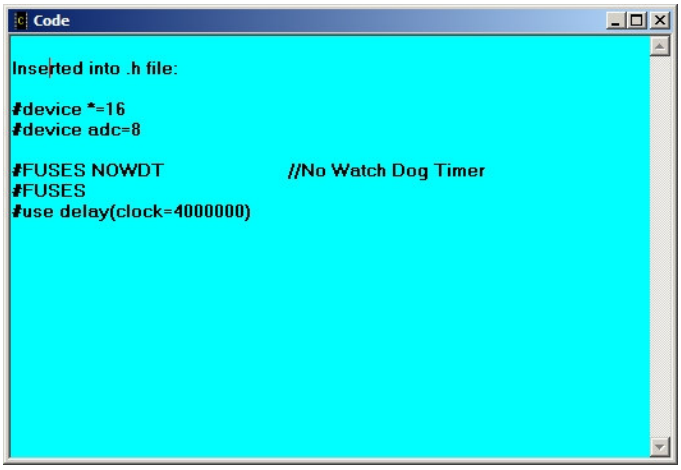
---

4. Aparecerá el asistente abierto por la pestaña 'General':
En 'Device' seleccionamos nuestro modelo de microcontrolador (PIC16F690).
Introducimos la frecuencia de trabajo de nuestro cristal oscilador (4000000).
Seleccionamos el uso de punteros de 16 bits para poder utilizar toda la RAM del micro.



---

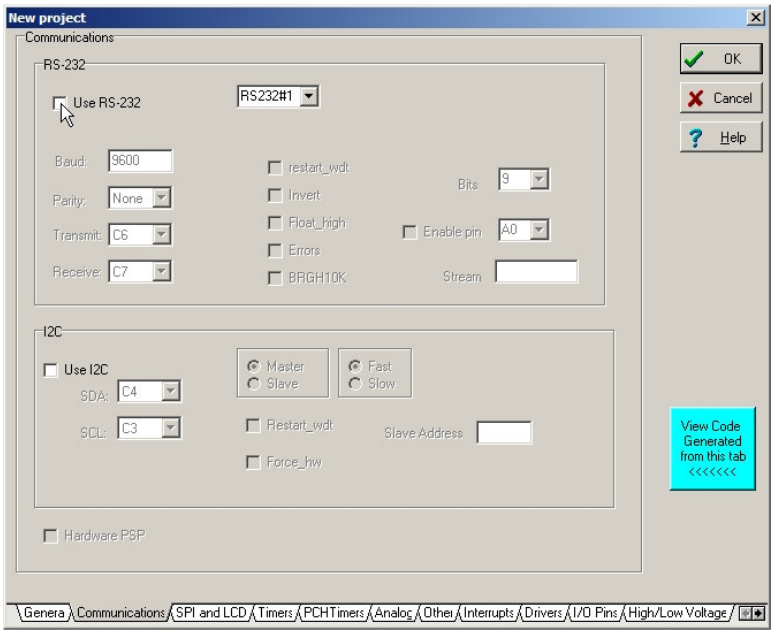
5. Si clicamos en 'View Code Generated from this tab' observaremos la ventana inferior.



En ella podemos observar el código C que el asistente generará (y dónde lo insertará) teniendo en cuenta las opciones seleccionadas de la pestaña del asistente en la que nos encontramos.

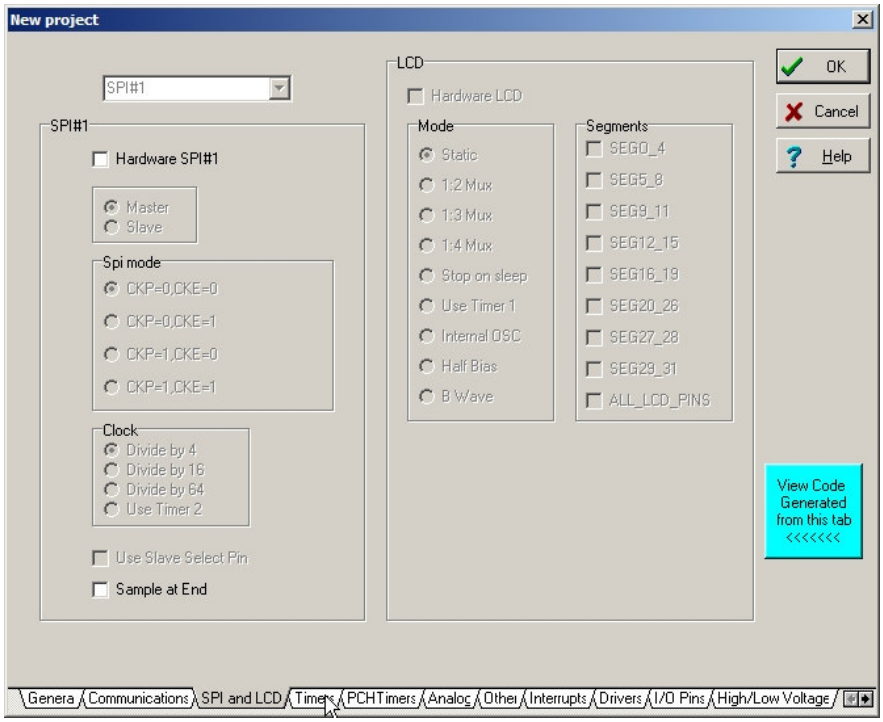
---

6. Pestaña 'Communications':

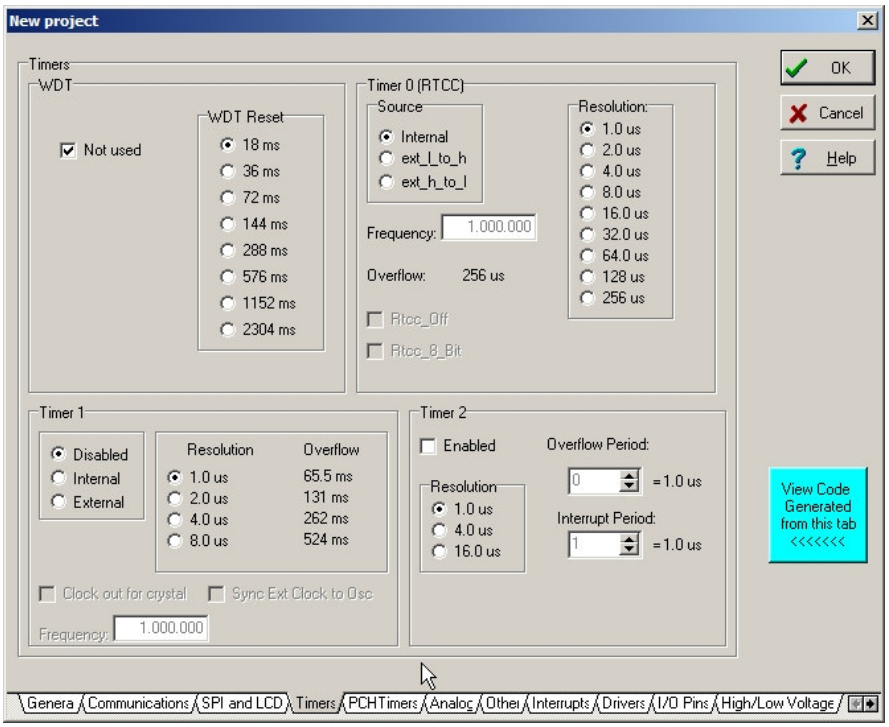


---

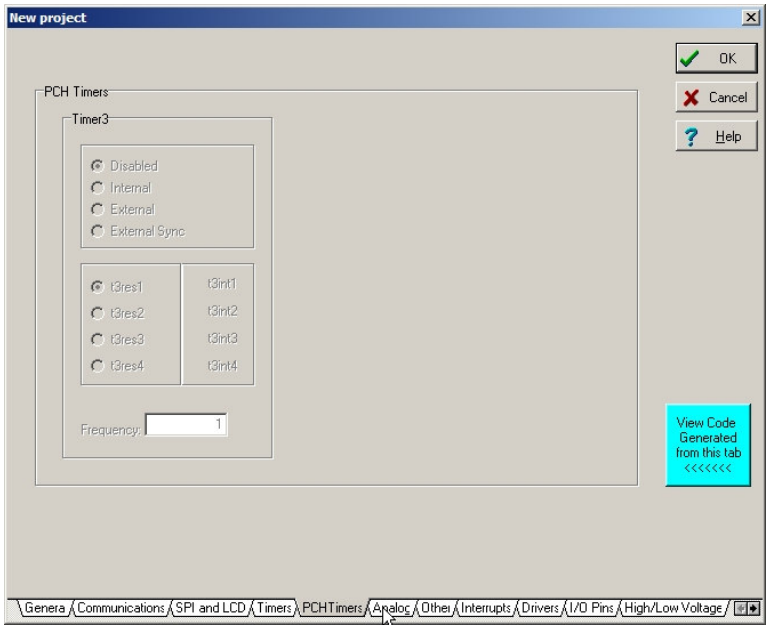
7. Pestaña ‘SPI and LCD’:



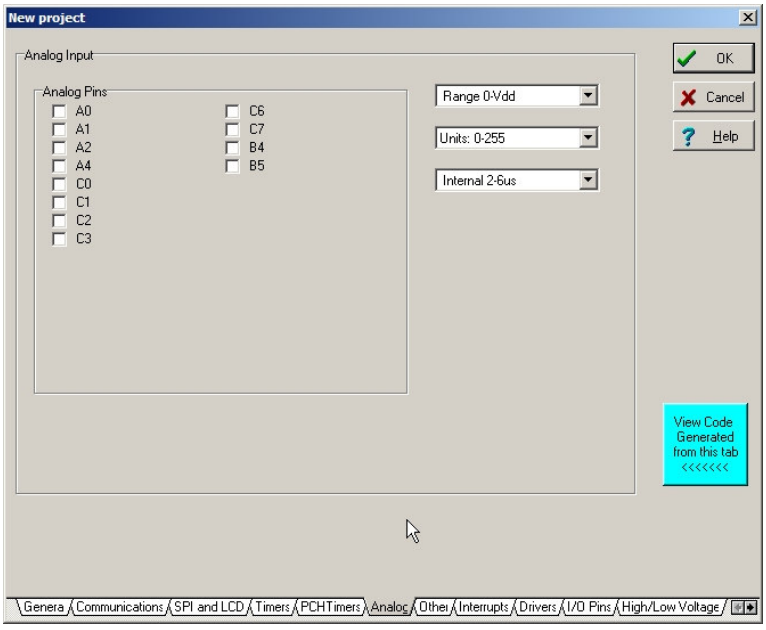
8. Pestaña ‘Timers’:

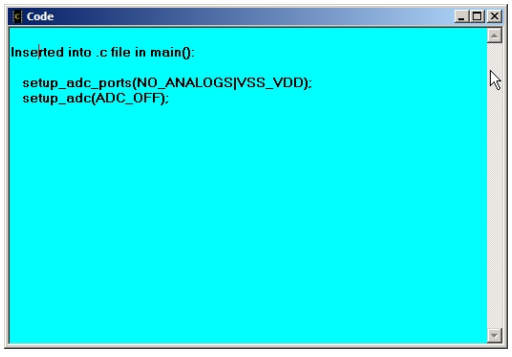


9. Pestaña ‘PCH Timers’:

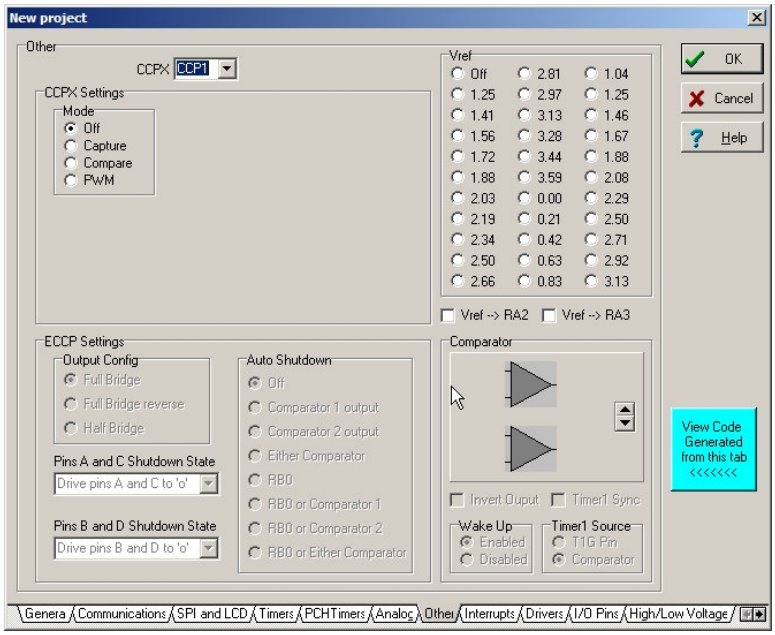


10. Pestaña ‘Analog’:



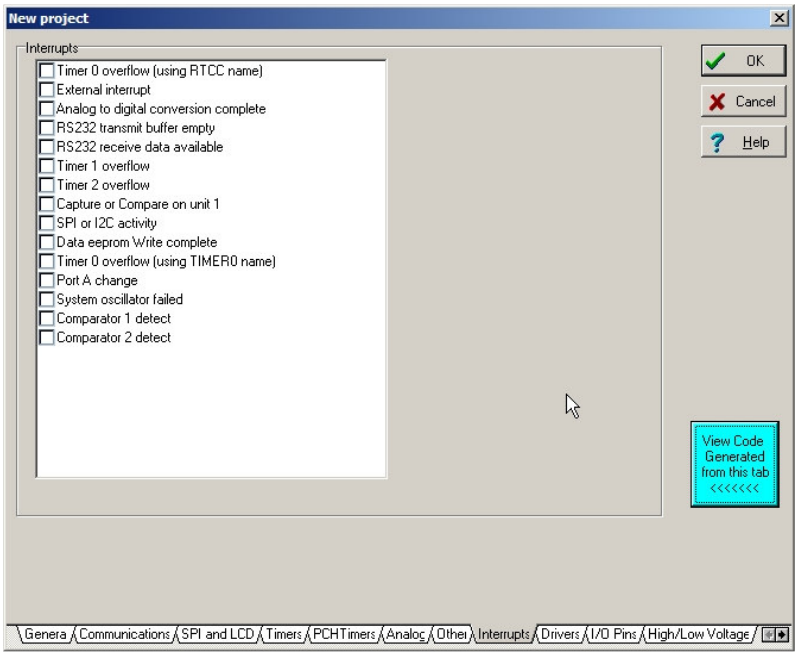


11. Pestaña ‘Other’:

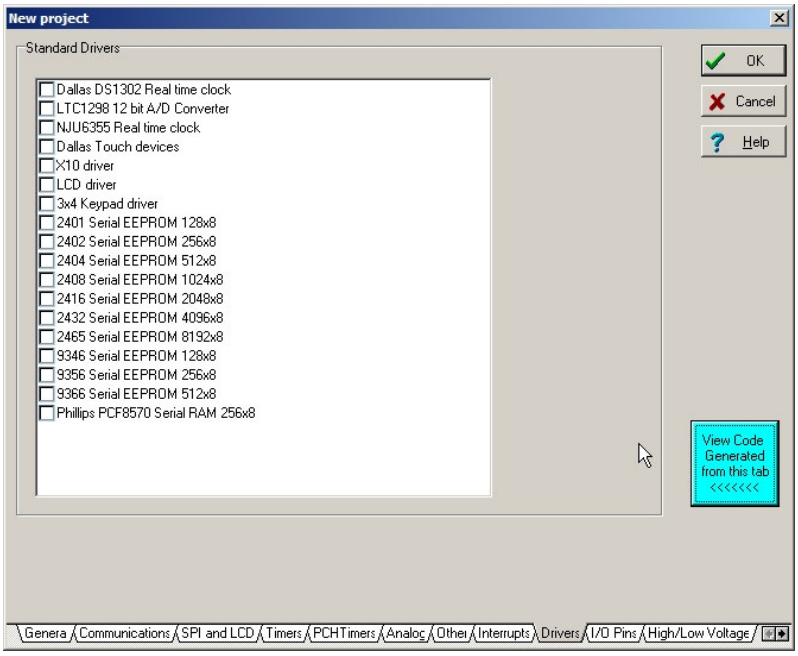


---

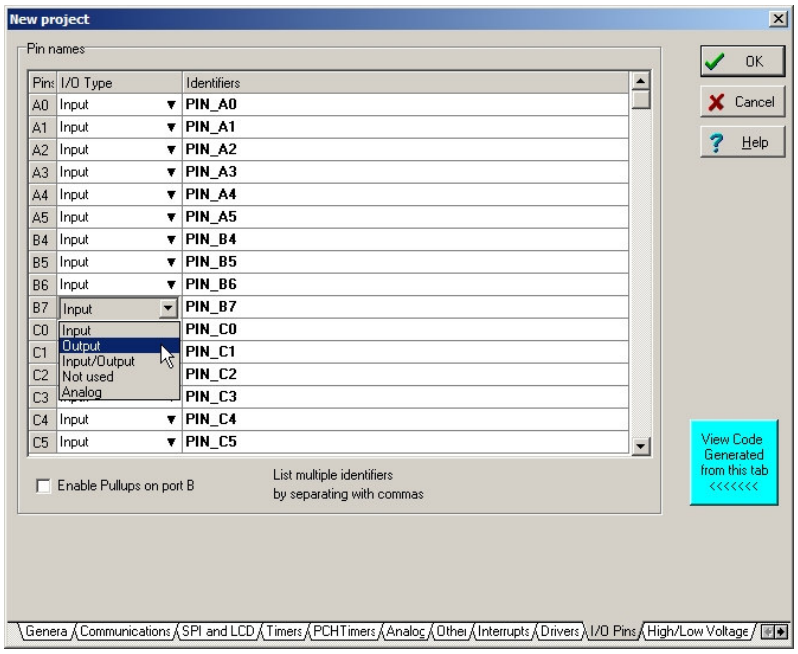
12. Pestaña ‘Interrupts’:



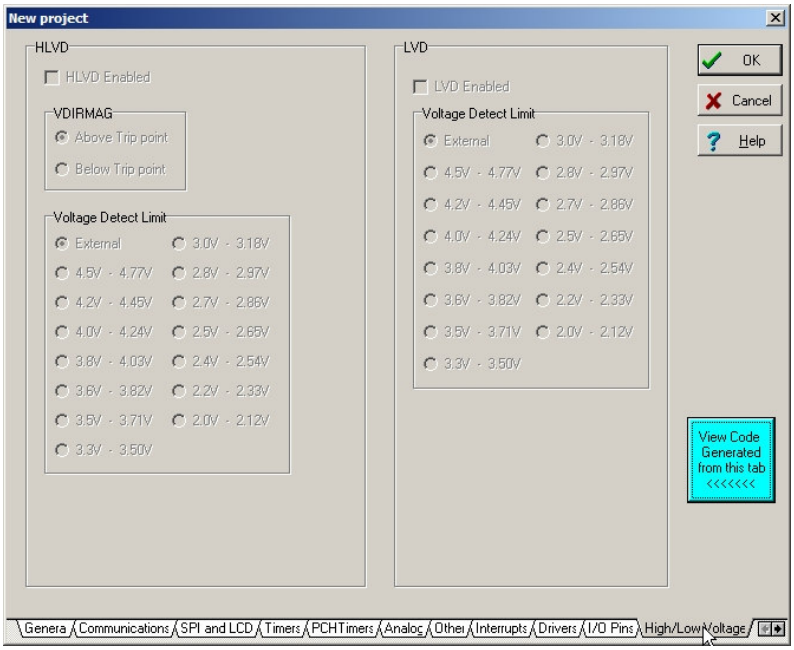
13. Pestaña ‘Drivers’:



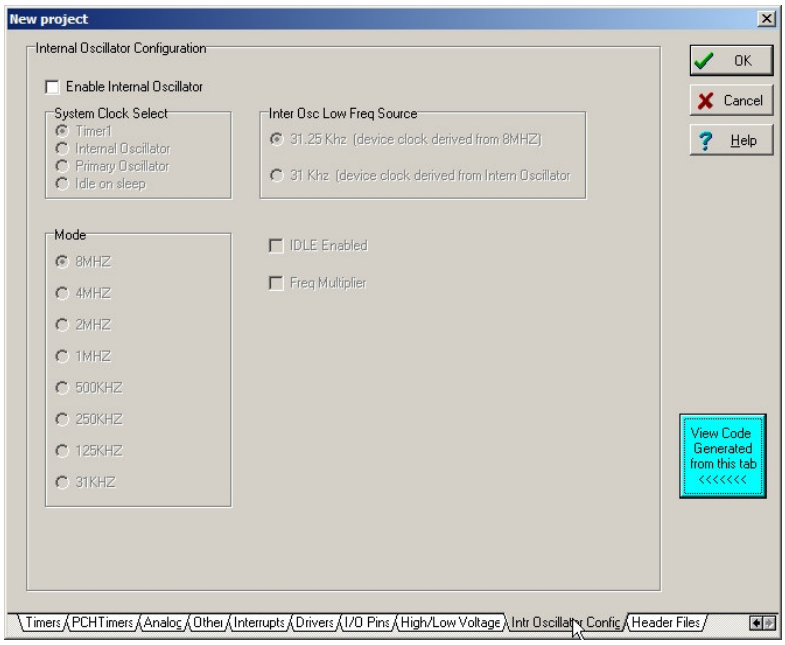
14. Pestaña ‘I/O Pins’:



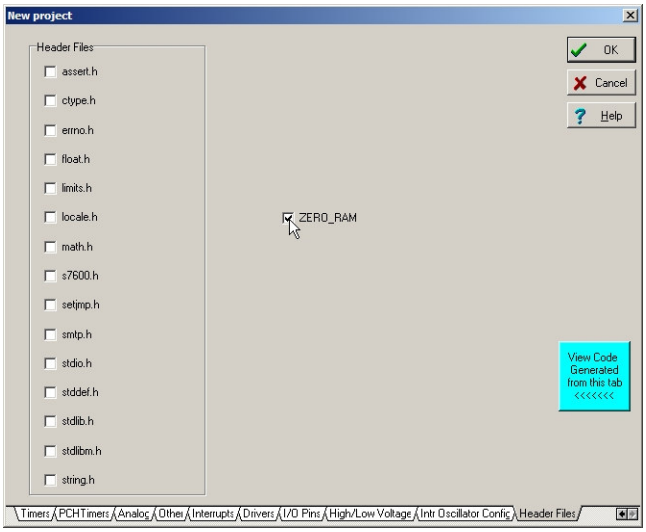
15. Pestaña ‘High/Low Voltage’:

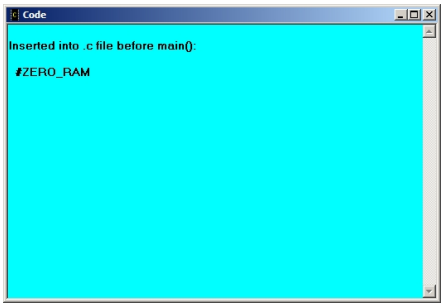


16. Pestaña ‘Intr Oscillator Config’:



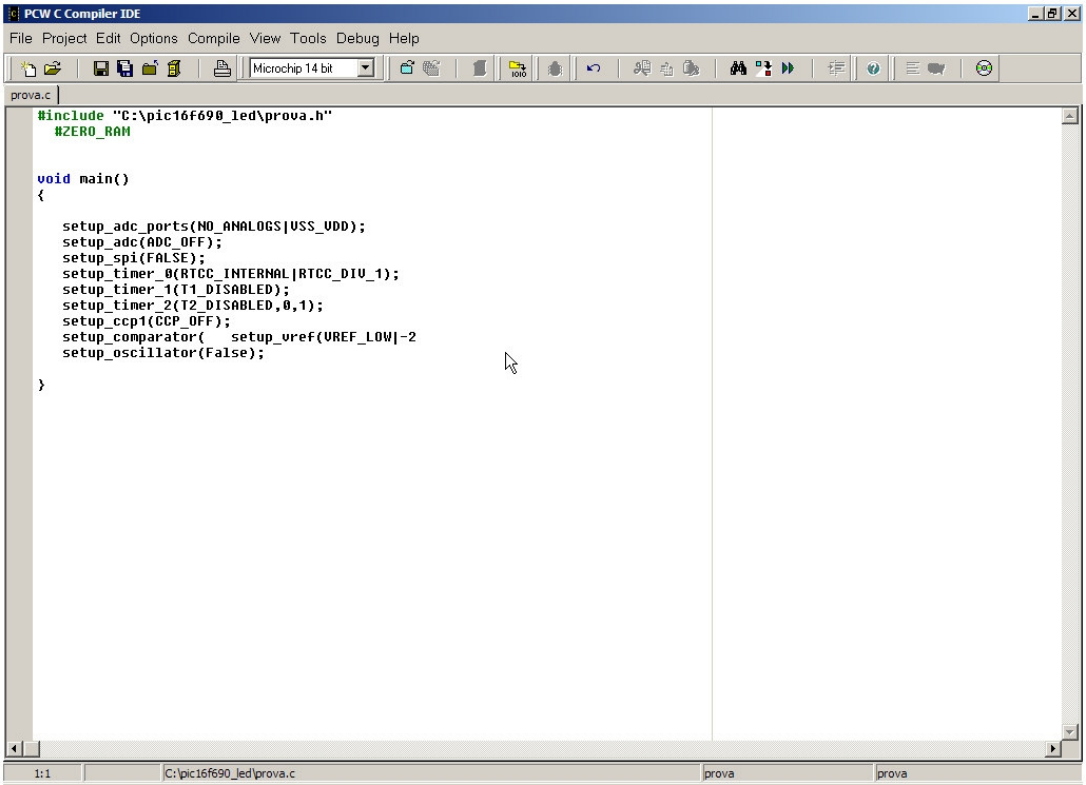
17. Pestaña ‘Header Files’:





---

18. Finalmente clicamos en ‘OK’ y obtendremos el siguiente código:



Podemos realizar varias observaciones:

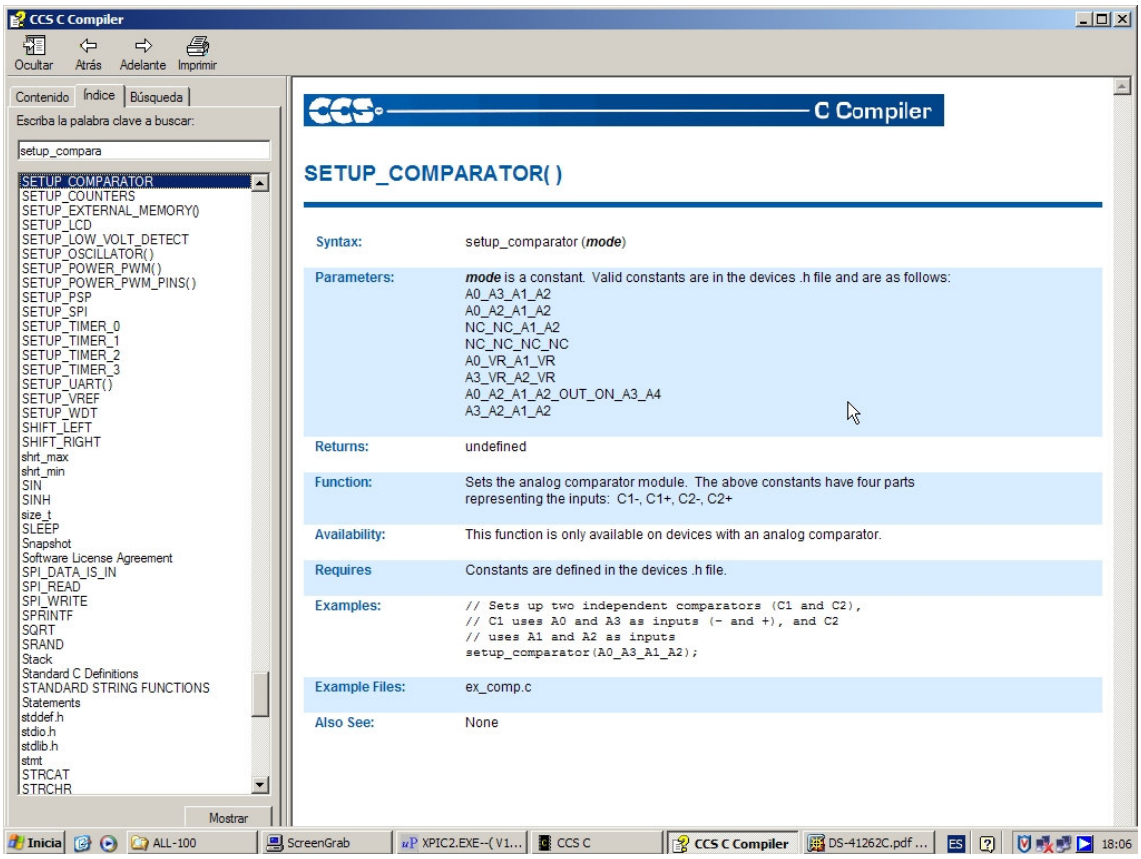
- Nuestro proyecto se ha creado con un archivo de código principal 'prova.c', y un archivo cabecera 'prova.h'.
- En la rutina principal (main) podemos observar una serie de llamadas a funciones de configuración que proporciona el propio compilador: setup_xxx.
- El asistente ha redactado un par de instrucciones con errores de sintaxis: setup_comparator(x) y setup_vref(x). Se trata de un bug de CCS PCWH que deberemos solucionar.

---

19. Consultamos la ayuda para conocer la sintaxis de setup_comparator(x):

Help > Contents

Clicamos en la pestaña 'Índice' y accedemos a la entrada SETUP_COMPARATOR de la lista.



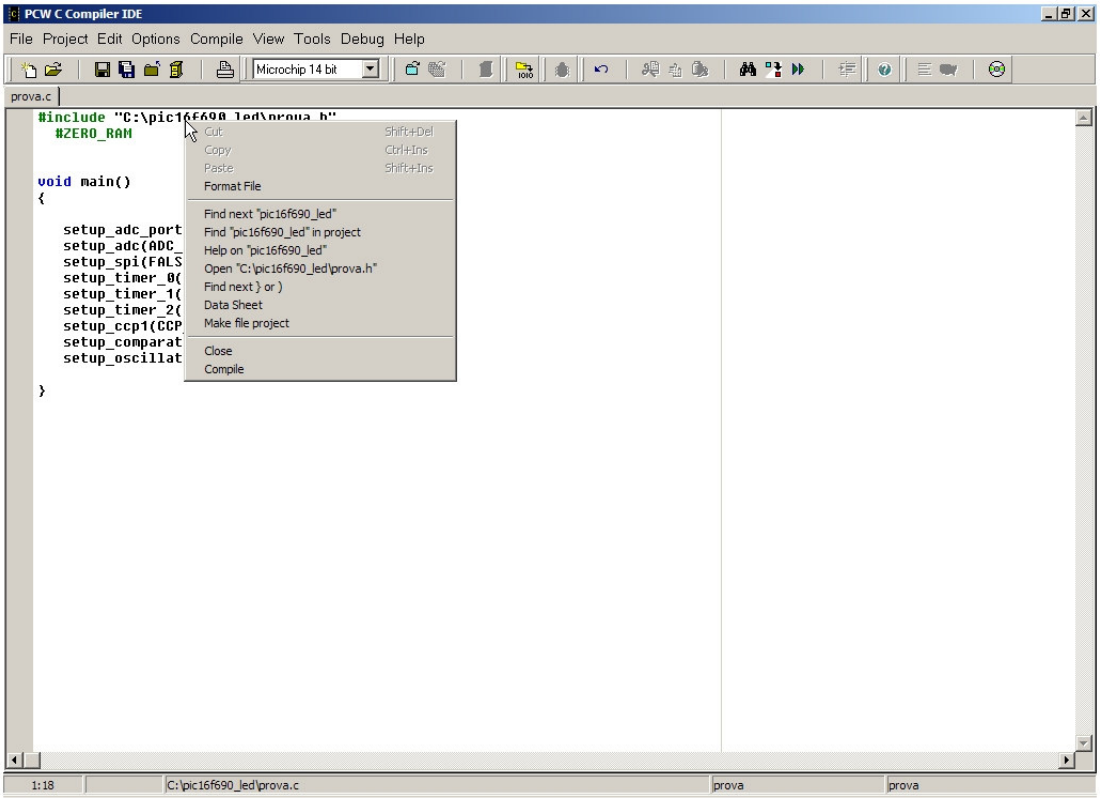
Como se puede observar, la función setup_comparator consta de un único parámetro 'mode', que especifica la configuración de los comparadores del micro. Como no vamos a emplear ninguno, usaremos la constante NC_NC_NC_NC.

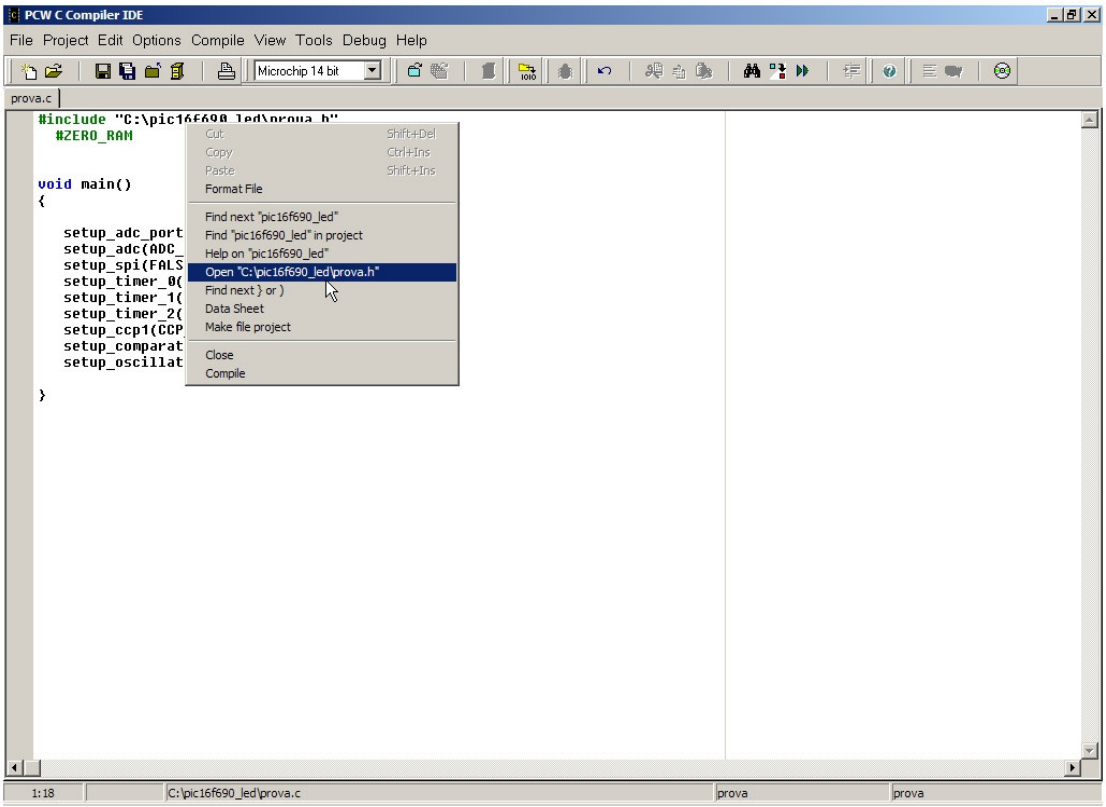
---

20. El paso anterior se puede realizar también accediendo al fichero de cabecera específico del 16F690, y consultando la sección que contiene las definiciones correspondientes a los comparadores, y consultando a su vez en el datasheet el valor de los valores numéricos que las deficiones nos indican.

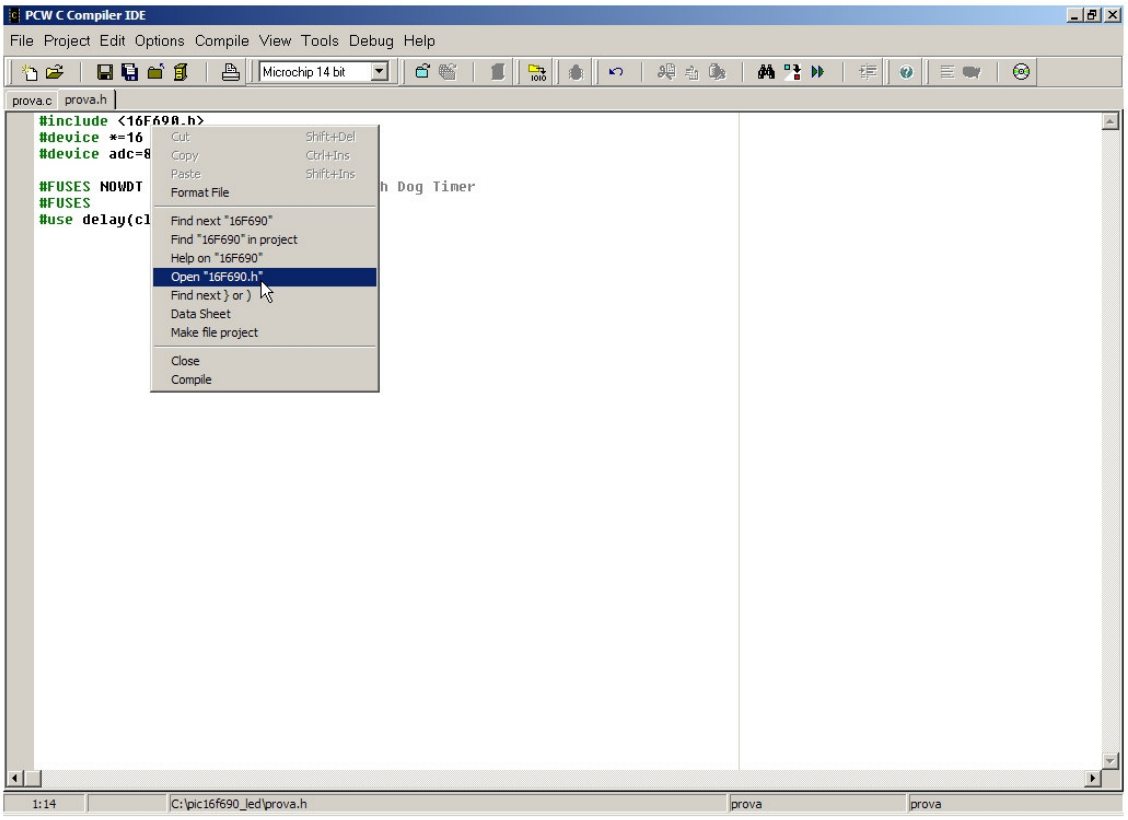
Para acceder al fichero 16F690.h podemos hacer:

- Clic con el botón derecho en el fragmento de código que incluye prova.h.
- Clic en Open '…\prova.h'

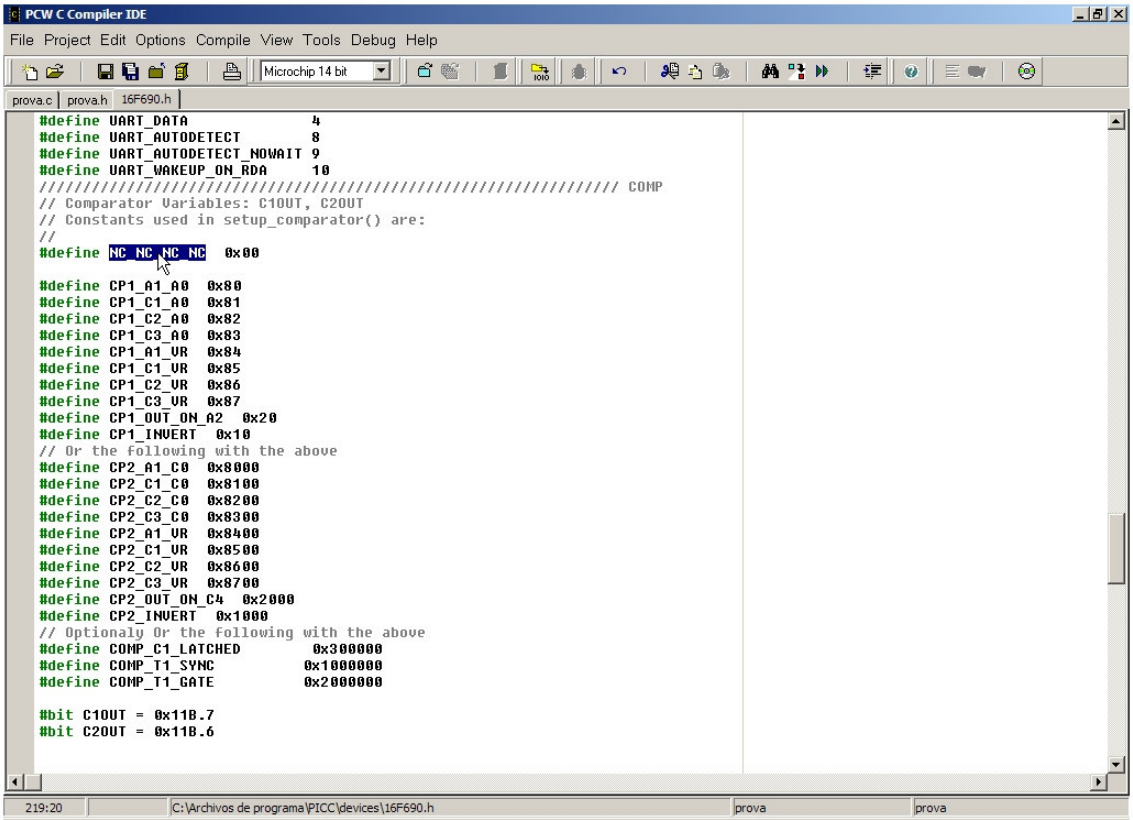




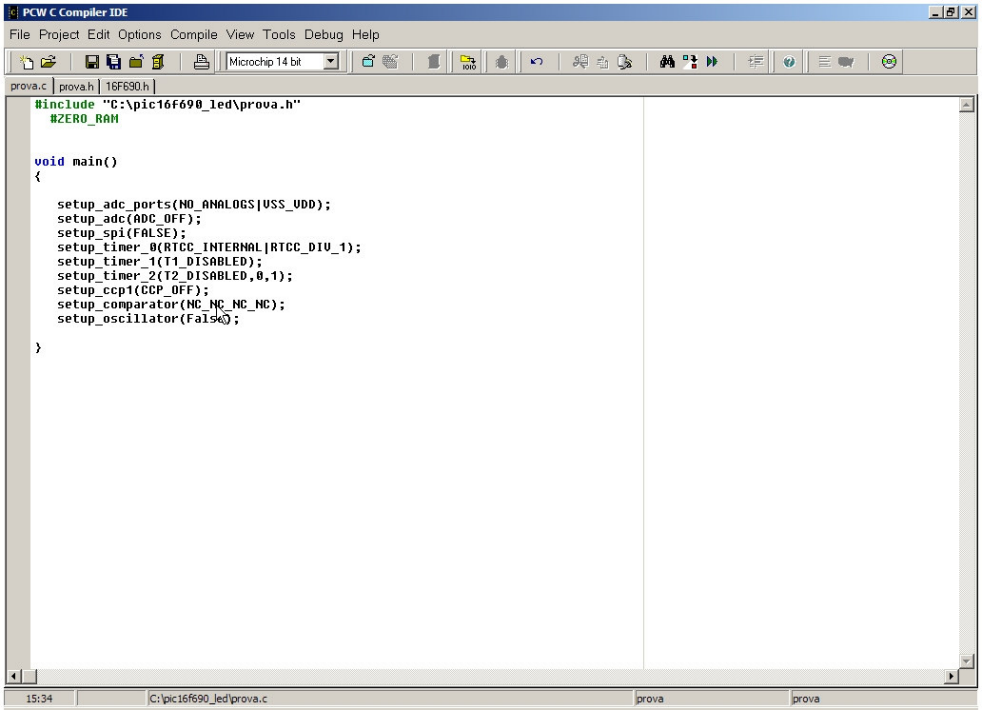
Una vez abierto 'prova.h', repetimos el proceso con '16f690.h', cuya inclusión se realiza desde prova.h.



En la figura inferior se puede observar la sección de comparadores del fichero '16f690.h':



21. En la figura inferior podemos observar el código final después de los arreglos.

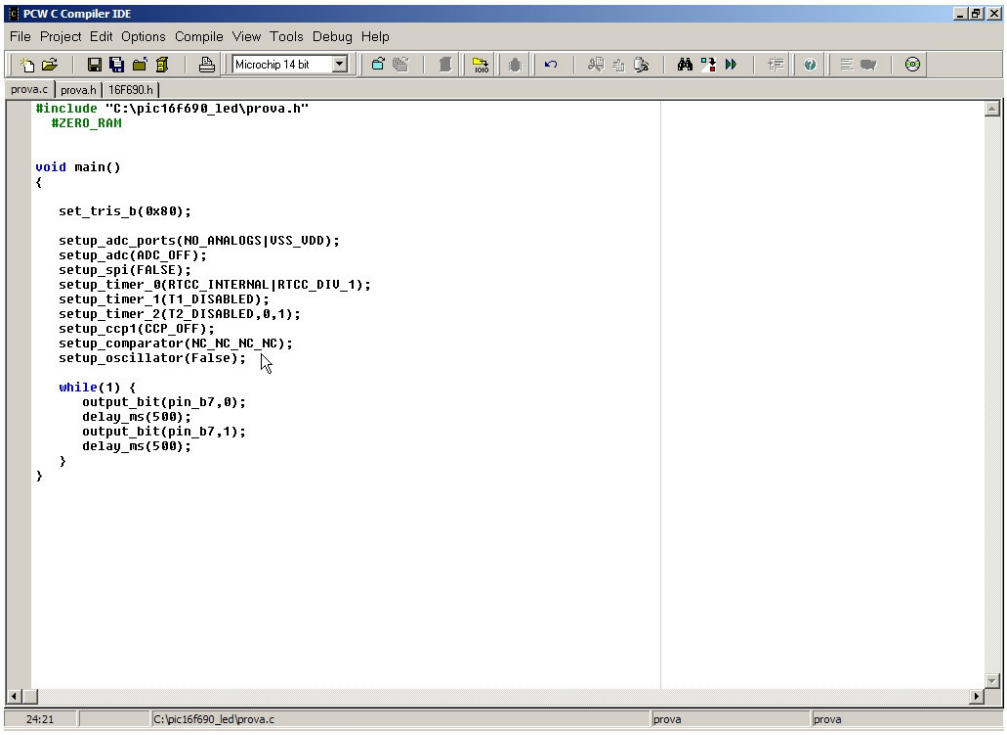


A destacar:

- La llamada a la función setup_comparator, con la constante que deshabilita los comparadores.
- La supresión de la llamada a la función que configura la tensión de referencia del A/D (setup_vref). Al no emplearla no será necesario configurar nada.

---

22. Escribimos el resto del código:

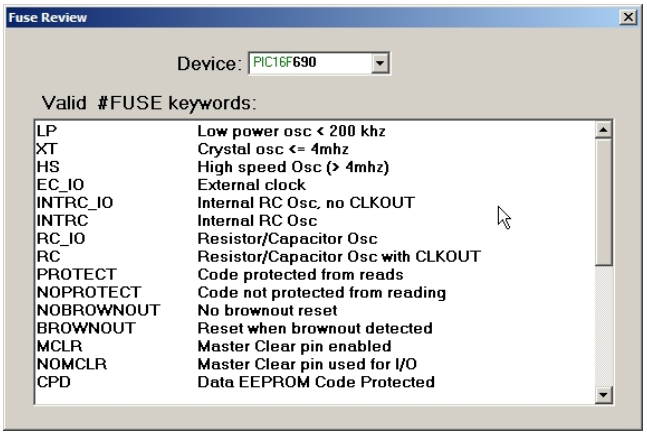


Como se puede ver, se trata de un bucle infinito (while(1)), en cuyo interior se activa y desactiva el bit 7 del puerto b (RB7), con retardo de 0,5 segundos. Así pues, si todo ha ido bien, veremos parpadear al led conectado a dicho pin.

---

23. A continuación vamos a ver qué constantes podemos asignar a la directiva #FUSES.

View > Valid Fuses.



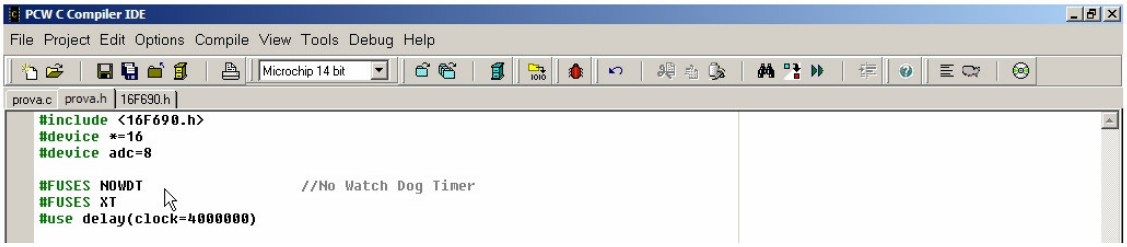
Añadiremos la directiva que indica que nuestro cristal funciona a una frecuencia inferior o igual a 4 MHz.

Como se puede observar, existen otras constantes interesantes. Por ejemplo, si quisiéramos proteger nuestro código de posibles lecturas, emplearíamos la constante PROTECT.

Deberemos tener en cuenta qué directivas empleamos en nuestro código porque deberemos configurar al programador universal en consonancia a los parámetros aquí escogidos.

---

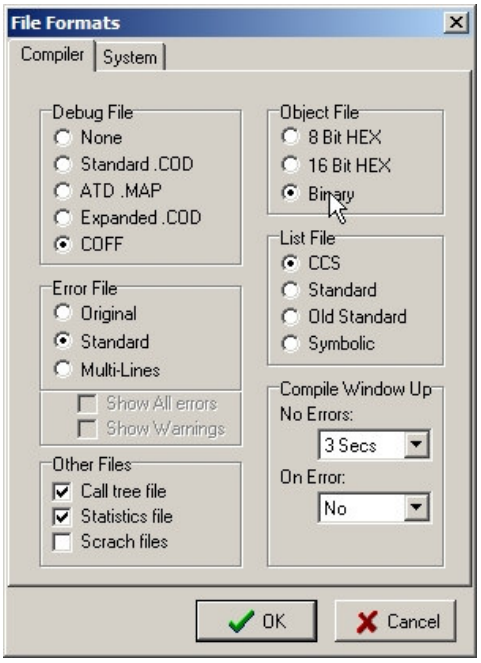
24. Modificamos prova.h añadiendo el parámetro XT a la directiva #FUSES:



---

25. Antes de compilar deberemos especificar el formato del archivo .hex que cargaremos en nuestro micro.

Accedemos al menú Options > File Formats y seleccionamos la opción 'Binary' del marco 'Object File'.

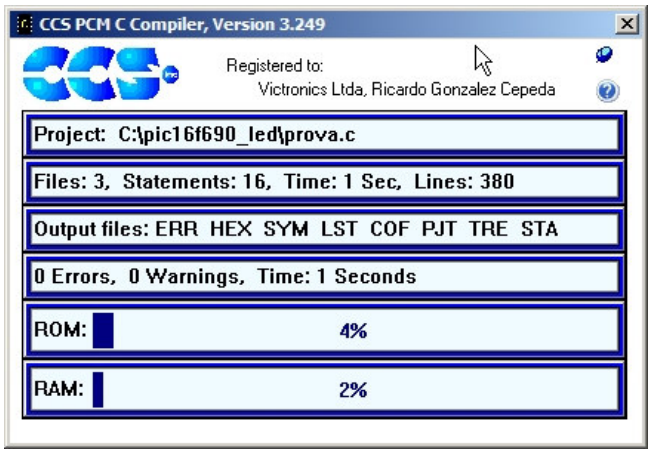


---

26. Recordemos salvar nuestro proyecto de vez en cuando. Una buena costumbre es hacerlo siempre antes de compilar cualquiera que sea el entorno de programación en el que trabajemos.

Ya podemos compilar nuestro proyecto mediante el menú Compile > Compile.

Si todo ha ido bien veremos la siguiente figura:

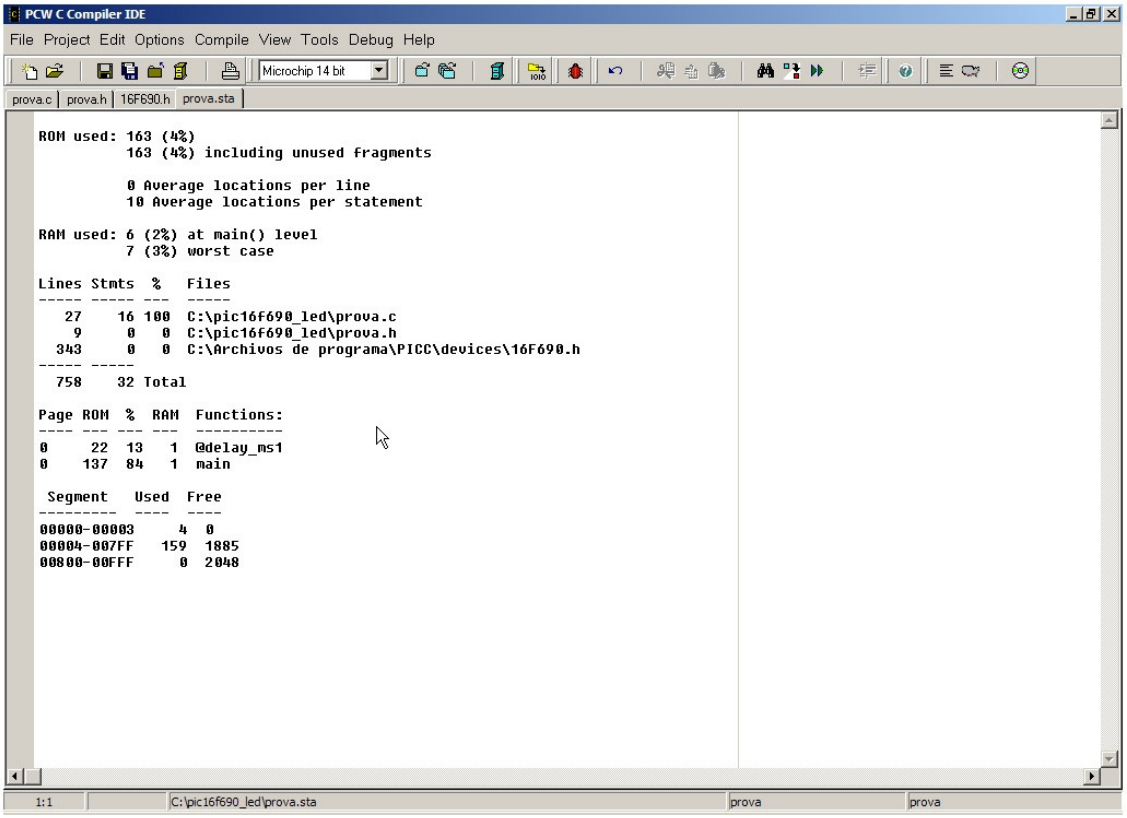


Obsérvese como una aplicación tan simple apenas consume memoria ROM (programa) y RAM (variables y registros de tiempo de ejecución).

Ya tenemos en el directorio de nuestro proyecto el fichero 'prova.hex' con el código máquina de nuestro micro, cuyo contenido volcaremos con el ALL-100.

---

27. Antes de pasar al programador, puede ser útil saber que después de compilar podemos acceder al menú View > Statistics. Se nos mostrará un archivo recopilatorio de datos estadísticos útiles sobre el rendimiento de nuestro programa.



3.5. Programación del microcontrolador

1.

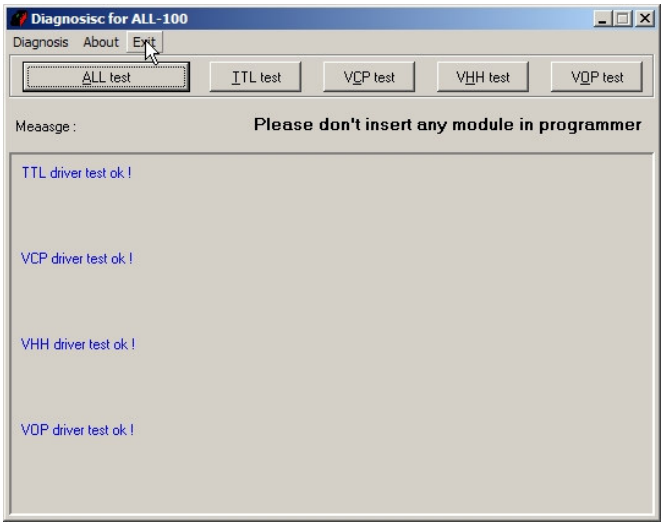
- Conectamos y encendemos el módulo programador.
- Iniciamos XAccess, el entorno que nos permite llamar a cada uno de los drivers programadores de dispositivos. Cada driver consiste en una aplicación .EXE que permite programar una familia de dispositivos de cierta marca (p.ej. Microchip).

---

2. Antes de empezar, comprobaremos el correcto funcionamiento del módulo.
Tester > Diagnostic tester

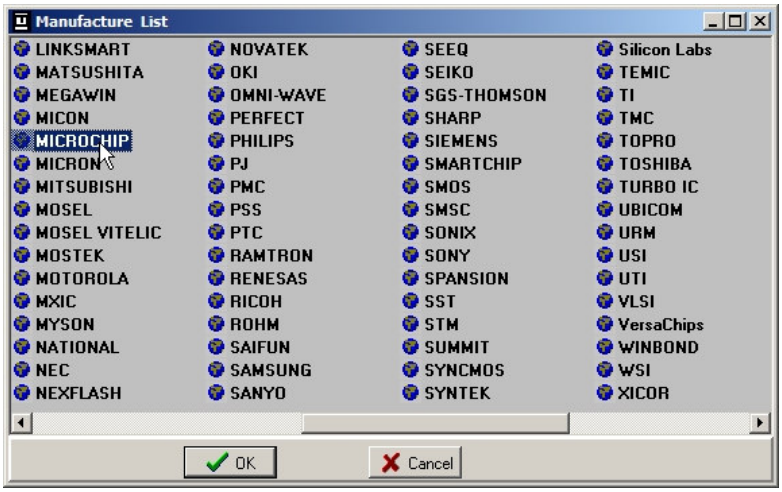
3.

- Comprobamos que no hemos insertado ningún integrado en el zócalo del módulo.
- Click en 'ALL test'.
- En el caso de que todo haya funcionado correctamente, la ventana que se mostrará será la siguiente.
- Click en 'Exit'.

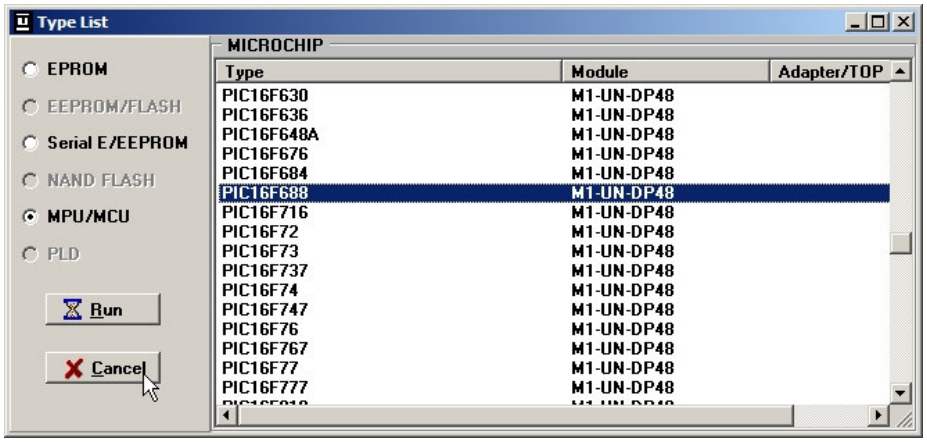


4. Clicamos en 'Device'.

5. Seleccionamos 'Microchip'.

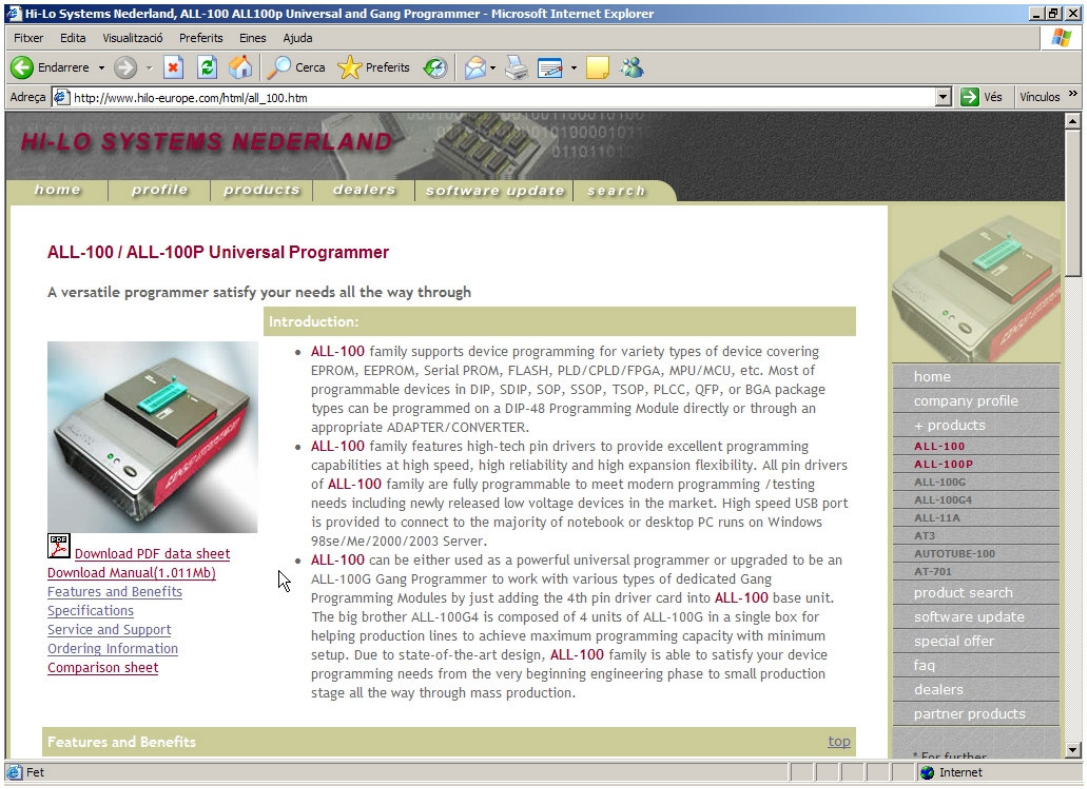


6. Comprobamos la existencia de nuestro dispositivo:
   - Seleccionamos 'MPU/MCU'.
   - Buscamos el PIC16F690. Como se puede observar en la figura, el micro que utilizamos es más reciente que el driver que vamos a emplear, así que deberemos descargarlo de la web del fabricante.
   - Click en 'Cancelar'.



---

7. Accedemos a la web del ALL-100, de Hi-Lo:
   www.hilo-europe.com/html/all_100.htm

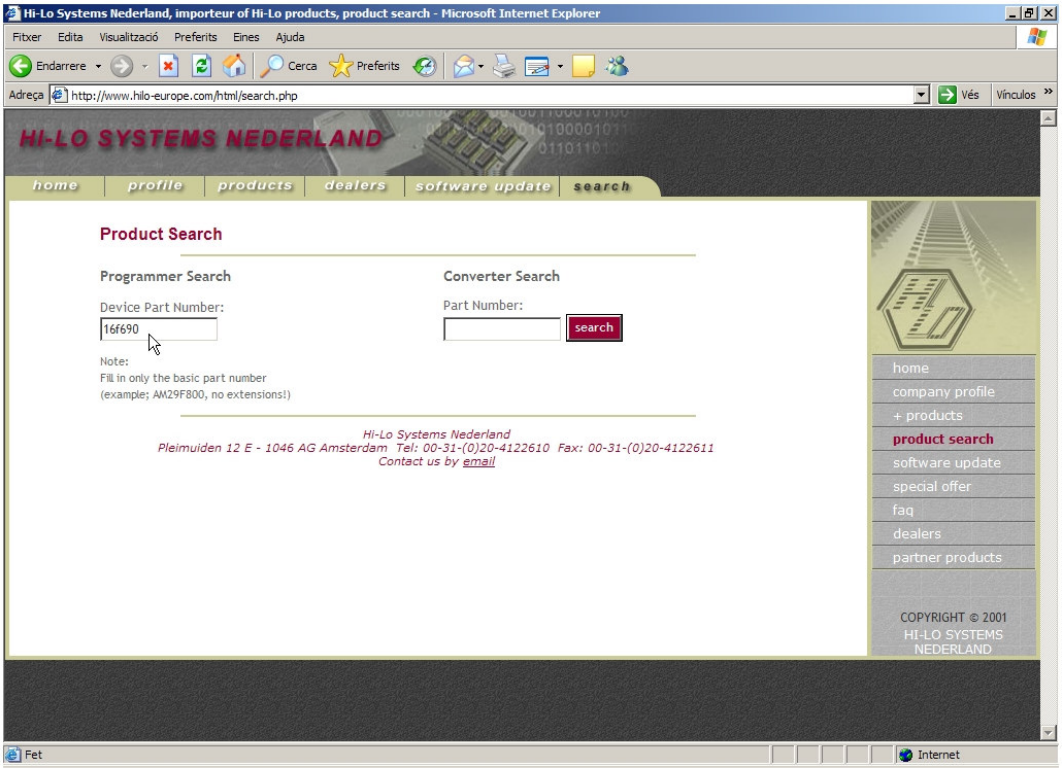


---

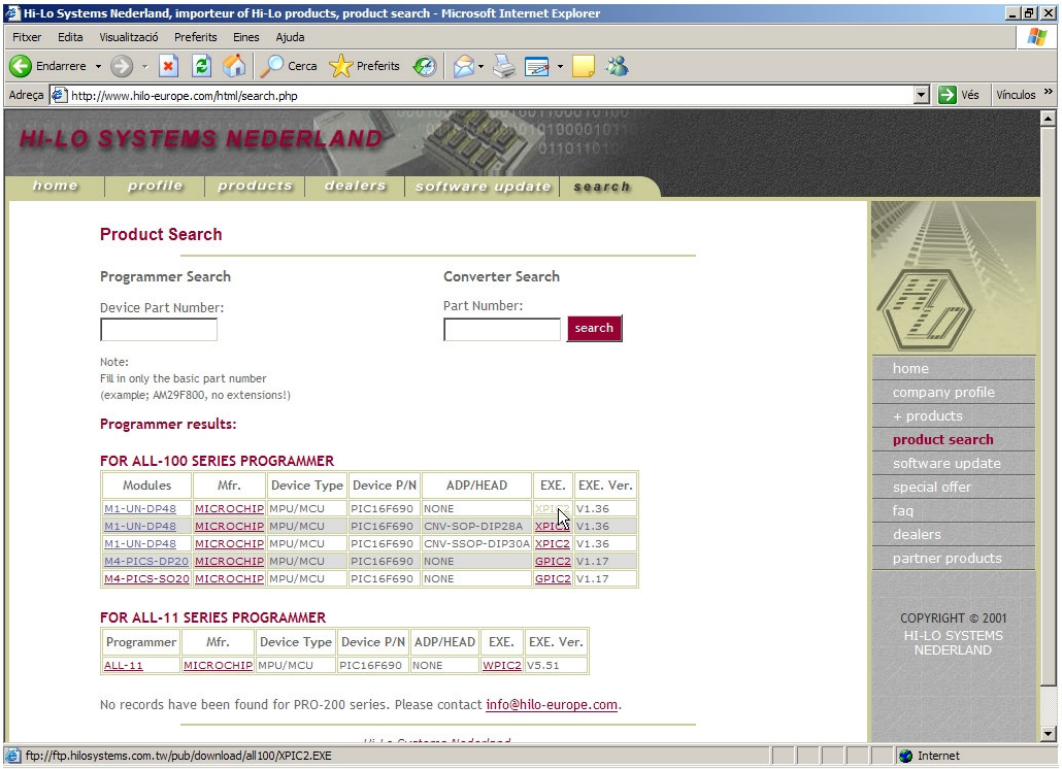
8. Además de poder descargarnos el manual del programador en formato PDF, podremos buscar el soporte para cierto integrado que nos interese.
Click en 'product search'.

---

9. Introducimos la referencia del dispositivo en 'Device Part Numbre': 16f690.



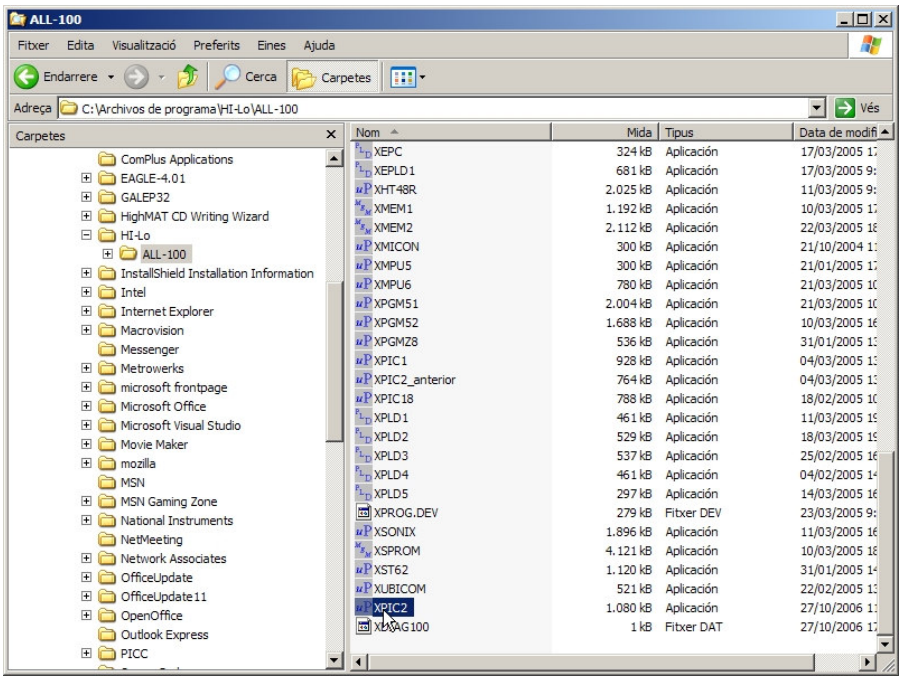
10. En la lista que se nos proporciona hacemos click en el link apuntado. Importante tener en cuenta el módulo que utiliza nuestro programador (para zócalos DIP de 48 pines máximo –DP48- y sin adaptador –NONE-).



11. Al descargarlo se nos pedirá la ruta donde almacenar el archivo XPIC2.EXE. Si le asignamos la misma carpeta que Xaccess (donde se encuentran el resto de drivers), cambiaremos el nombre del antiguo archivo XPIC2.EXE, para poder recuperarlo en caso de necesidad.
Ya podemos empezar con la descarga.

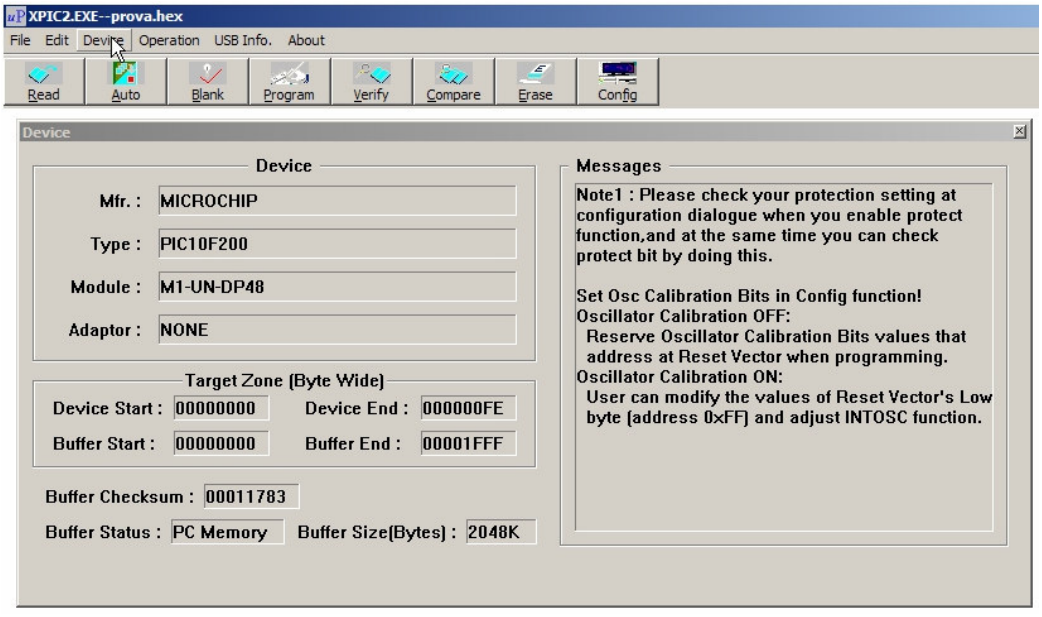
---

12. Ejecutamos el archivo descargado XPIC2.EXE. Obsérvese como cada driver es independiente, si bien XAccess proporciona un método cómodo para acceder a cada uno de ellos.



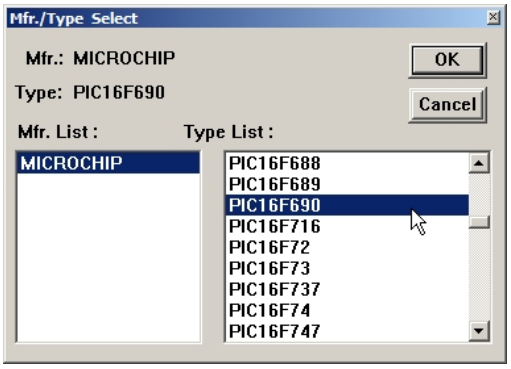
---

13. Click en 'Device'.



---

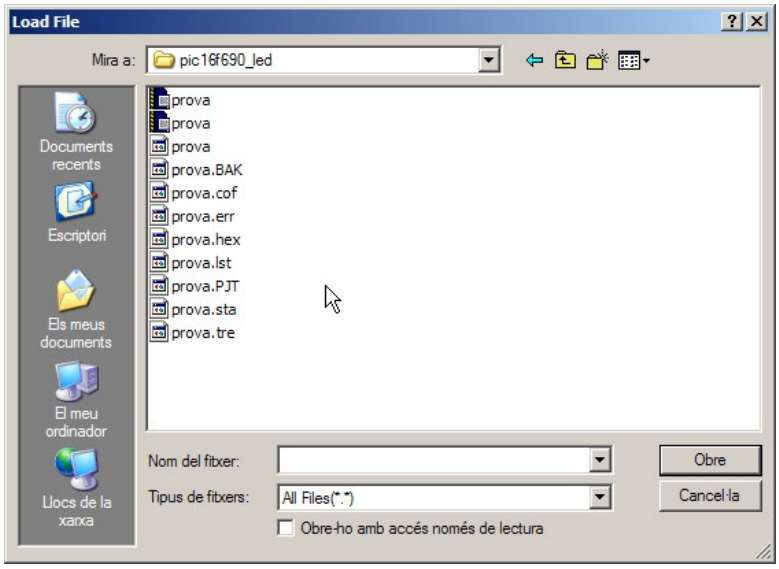
14. Seleccionamos el dispositivo a programar: PIC16F690.



---

15. Cargamos el archivo .hex que hemos creado con el compilador PCWH. Para ello:

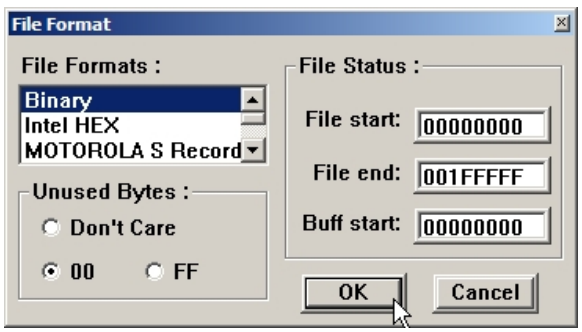
File > Load File to Program Buffer



En nuestro caso se trata de 'prova.hex'.

---

16. Seleccionamos el formato de archivo 'Binary' y la opción que establece que los bytes en ROM no usados queden como '0'.

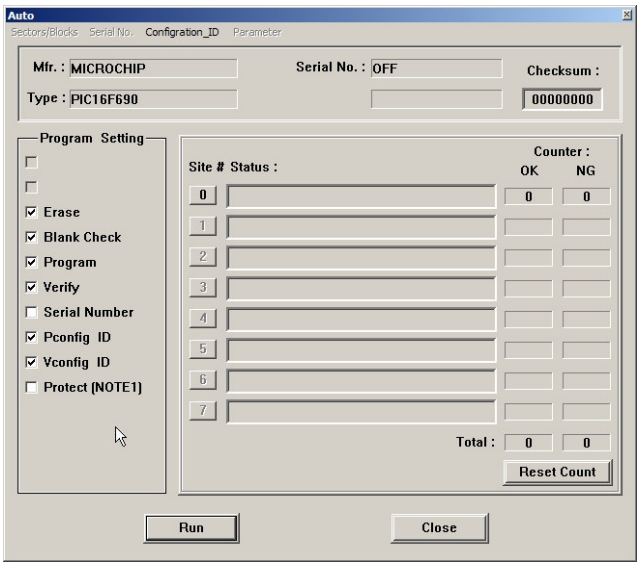


---

17. Click en el botón ‘Auto’.

---

18. Las opciones de programación del chip deben quedar como sigue:



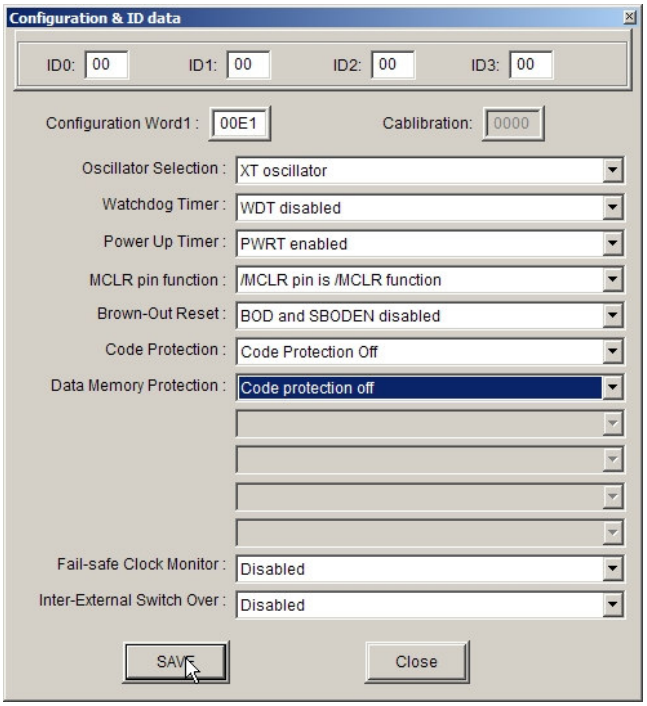
Obsérvese que no vamos a proteger el dispositivo de posibles lecturas del código almacenado en ROM.

---

19. Click en ‘Configuration_ID’.

---

20. Seleccionamos los parámetros adecuados (Config Word) para nuestro montaje.



- En ‘Oscillator Selection’, debemos seleccionar ‘XT oscillator’ pues trabajamos con un cristal de 4 MHz.
- Si queremos habilitar el Reset (nuestro montaje lo incorpora), seleccionamos en ‘MCLR pin function’, ‘/MCLR pin is /MCLR function’.
- En ‘Code Protection’, obviamente seleccionamos ‘Code Protection Off’. Ídem en ‘Data Memory Protection’.

Finalmente click en ‘Save’.

---

21. Cerramos el cuadro de diálogo ‘Auto’, clicando en ‘Close.

---

22. Grabamos la configuración que hemos establecido para este dispositivo y este montaje en concreto. Para ello:

File > Save Programmer Configuration

Y nombramos el archivo de extensión .cfg.

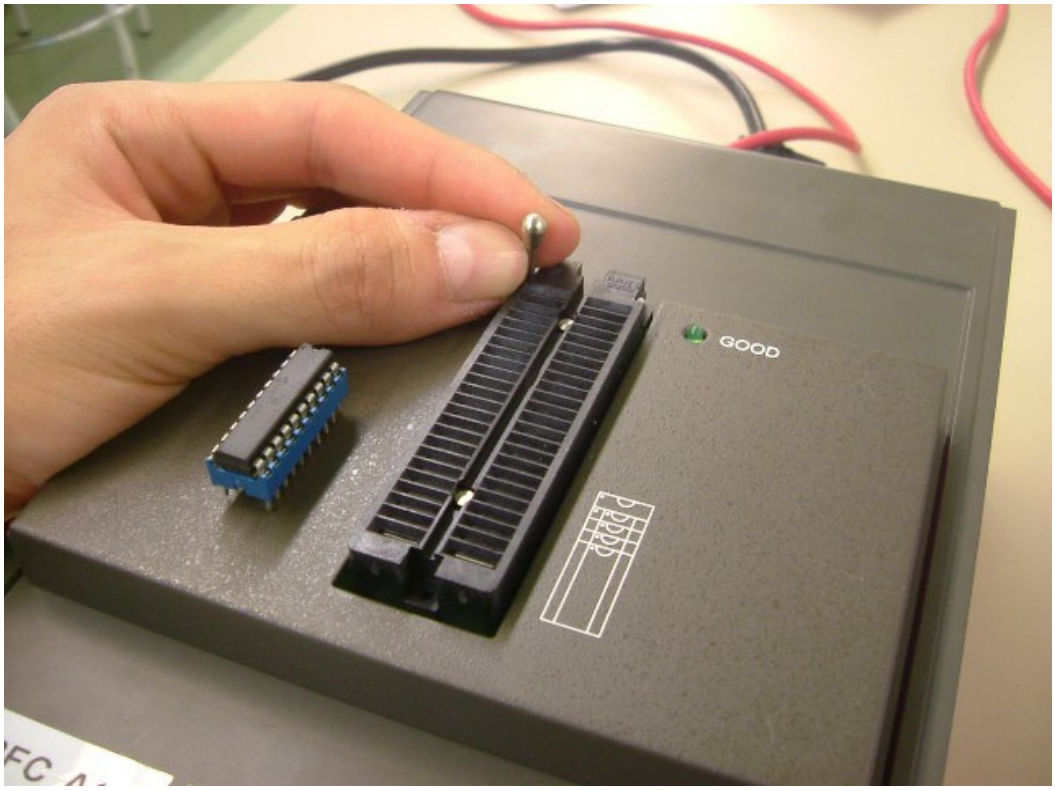
Es recomendable hacerlo pues cada vez que carguemos un archivo .hex para volcarlo, dicha configuración se perderá.

---

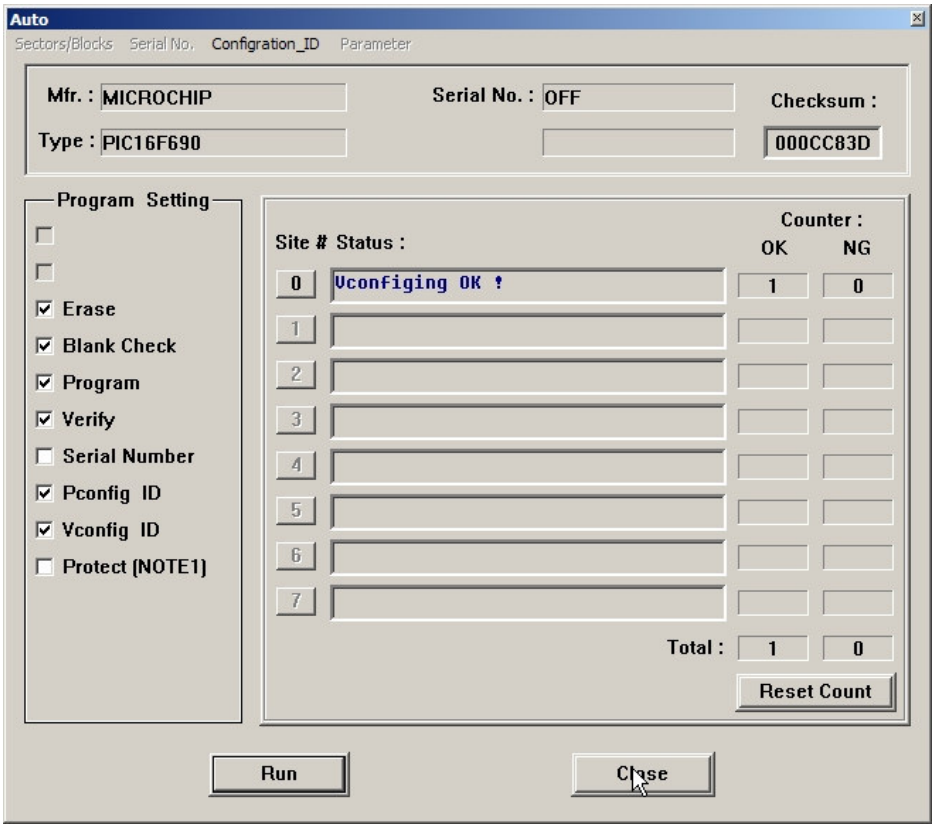
23. Clicamos, de nuevo, en el botón ‘Auto’.

---

24. Procedemos a insertar el chip en el zócalo del módulo programador. Para ello tiramos de la palanquita situada en la parte superior izquierda del zócalo e insertamos el chip. Bajamos la palanquita.

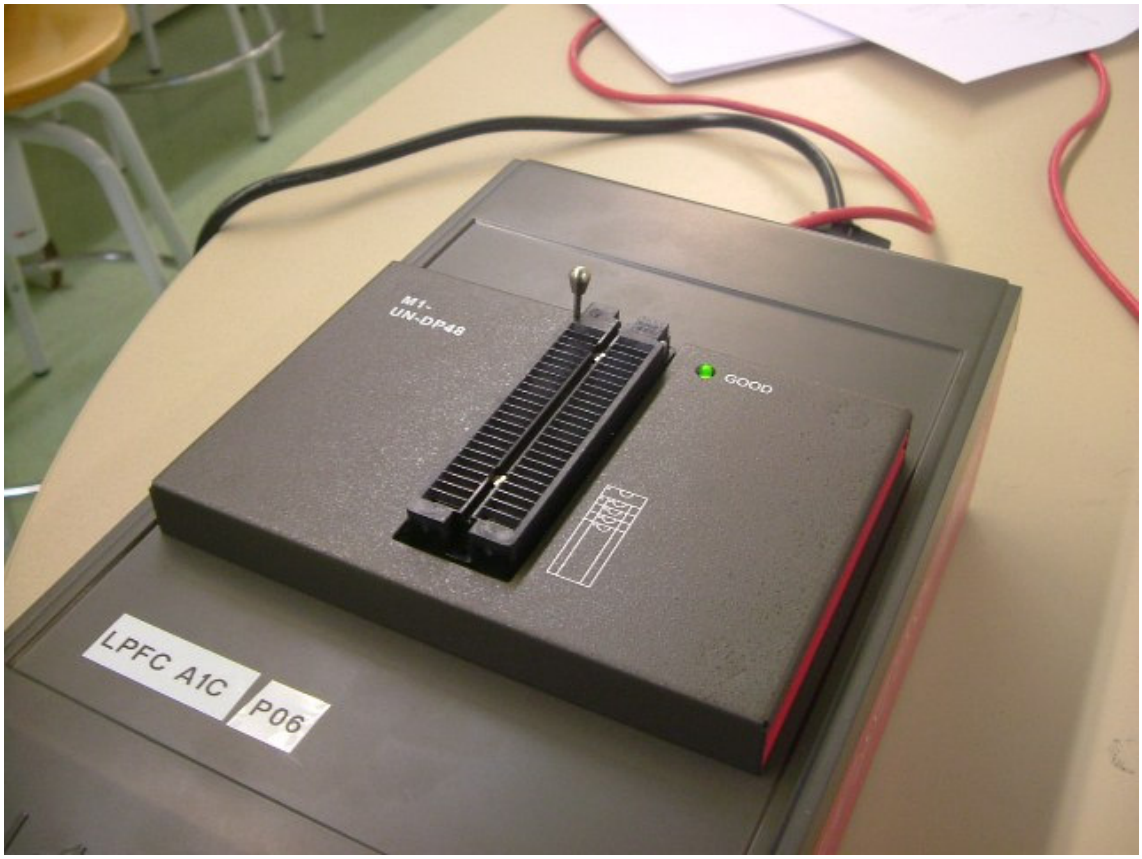


25. Click en el botón 'Run'. Si todo ha ido bien al finalizar el proceso de volcado observaremos el siguiente mensaje:



Click en 'Close'.

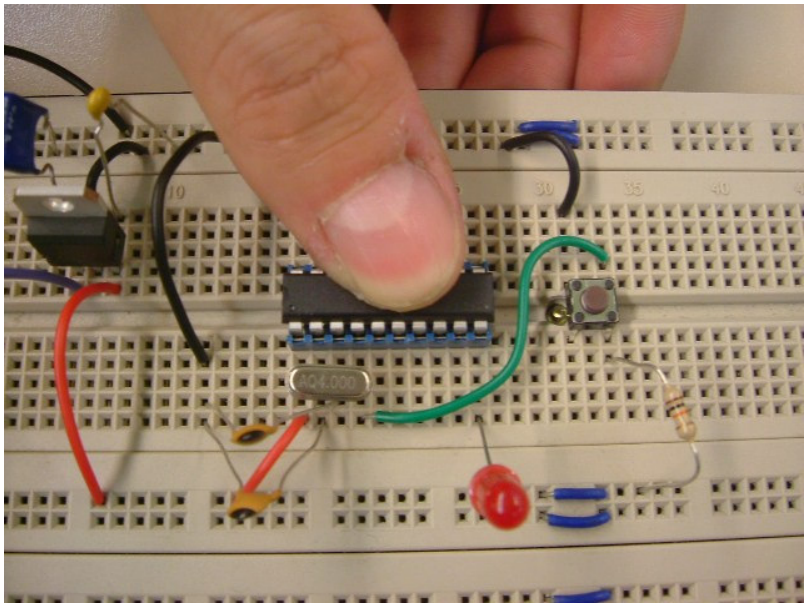
26. Tiramos de la palanquita y retiramos el micro.



El PIC ya está programado y podemos insertarlo en nuestro circuito. Cada vez que queramos realizar alguna modificación al programa deberemos realizar el proceso de compilado y volcado del hexadecimal en la ROM del chip.

3.6. Comprobación del correcto funcionamiento

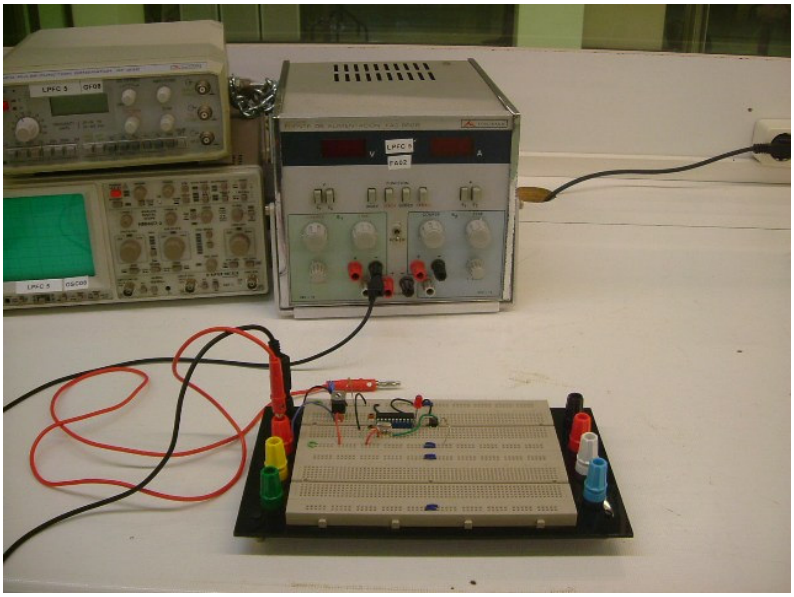
1. Insertamos el micro en la posición que le corresponde de la protoboard, asegurándonos de que queda bien fijado y las patitas hacen contacto con el sustrato metálico de la placa de pruebas.



---

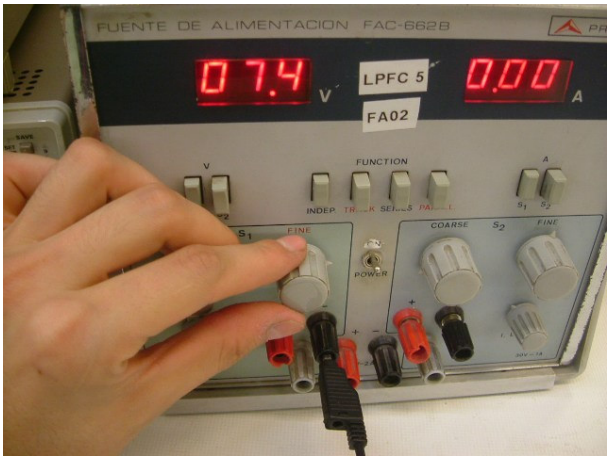
2. Conectamos los cables banana-banana a la protoboard y a la fuente de alimentación, a excepción del borne positivo que corresponde a la fuente. El porqué de proceder así radica en el pico de tensión y corriente que proporciona la fuente al encenderla. Podríamos dañar nuestro micro.

Más conveniente es conectar el positivo de la placa a la fuente una vez esté encendida.



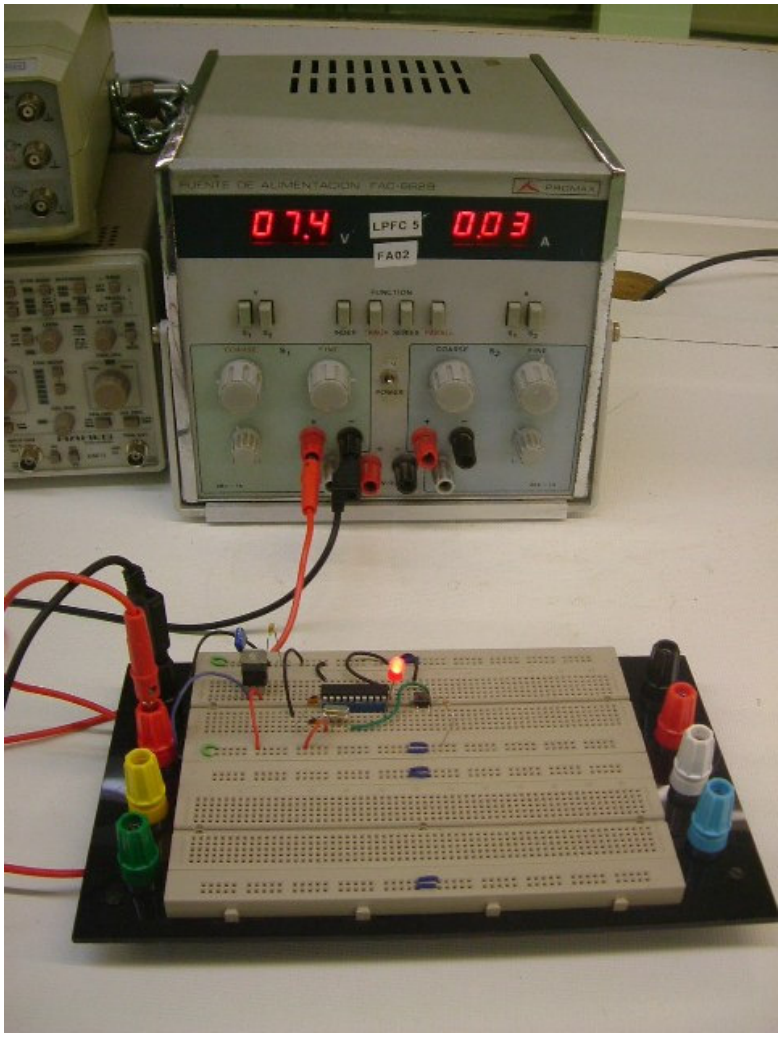
---

3. Encendemos la fuente de alimentación y seleccionamos una tensión superior a los 7 V (requisito del 7805) e inferior a 8 V.



---

4. ¡Llegó el momento de la verdad! Conectamos la placa a la fuente. Si todo ha ido bien, nuestro LED se encenderá y apagará alternativamente cada medio segundo.



---

5. Finalmente comprobamos que el botón de reset realiza su función correctamente: al mantenerlo pulsado el circuito debería permanecer inactivo (el LED deja de encenderse). Al soltarlo vuelve a encenderse de forma intermitente.