

PROTEUS DESIGN SUITE

Primeros pasos con la pestaña Taller VSM

labcenter Electronics
www.labcenter.com

COPYRIGHT NOTICE

© Labcenter Electronics Ltd 1990-2019. All Rights Reserved.

Los programas de software PROTEUS (Proteus Capture, PROSPICE Simulation, Schematic Capture and PCB Layout y sus archivos de librerías asociadas, archivos de datos y toda la documentación son copyright de ©Labcenter Electronics Ltd. Todos los derechos reservados. Si usted ha comprado una licencia para usar el software en una máquina en un momento dado, usted no es dueño del software. La copia no autorizada, el préstamo y la redistribución del software o de la documentación realizada por cualquier medio, constituye una violación de los derechos de autor. La piratería de software es un delito.

PROSPICE incorpora el código fuente de Berkeley Spice3f5 que es copyright de © Regentes de la Universidad de Berkeley. Los modelos SPICE de los fabricantes incluidos con el software son propiedad de sus respectivos autores.

La herramienta Qt GUI es propiedad de © 2012 Digia Plc y/o sus subsidiarias y se usa bajo licencia LGPL versión 2.1. Algunos iconos son propiedad de 2010 The Eclipse Foundation utilizadas bajo la licencia publica versión 1.0 de Eclipse. Algunos ejecutables son utilidades en formato binario utlizadas bajo la licencia © 2010 The GNU Project, versión GPL 2.

AVISO
PROTEUS también contiene código especial que, mediante la utilización de la clave de licencia de usuario, evitará que se pueda utilizar más de una copia en una red al mismo tiempo. Por lo tanto, usted debe comprar una clave de licencia por cada copia que desea ejecutar de forma simultánea.

RENUNCIA
No se garantiza de ninguna manera el contenido de este paquete de software, ni se garantiza su idoneidad para cualquier propósito en particular. Ni Labcenter Electronics Ltd, ni ninguno de sus empleados, subcontratistas o distribuidores serán responsables por los errores que puedan existir en el software, librerías de componentes, modelos de simulación o en la documentación de

usuario. Ni tampoco por cualquier daño directo, indirecto o consecuente ni de las pérdidas financieras derivadas de la utilización del paquete.

Se recuerda en particular a todos los usuarios que la simulación con éxito de un diseño utilizando el simulador PROSPICE no demuestra de manera concluyente que vaya a trabajar cuando se pasan los datos al proceso de fabricación. Siempre es conveniente hacer un prototipo de pruebas antes de mandar fabricar una tirada de cantidades grandes.

Los modelos SPICE de los fabricantes incluidos con PROSPICE se suministran de acuerdo con el concepto "tal y como son" y tampoco Labcenter ni sus autores ofrecen ningún tipo de garantía en cuanto a su precisión o su funcionalidad.

1.-Introducción.

1.1.-Objetivo.

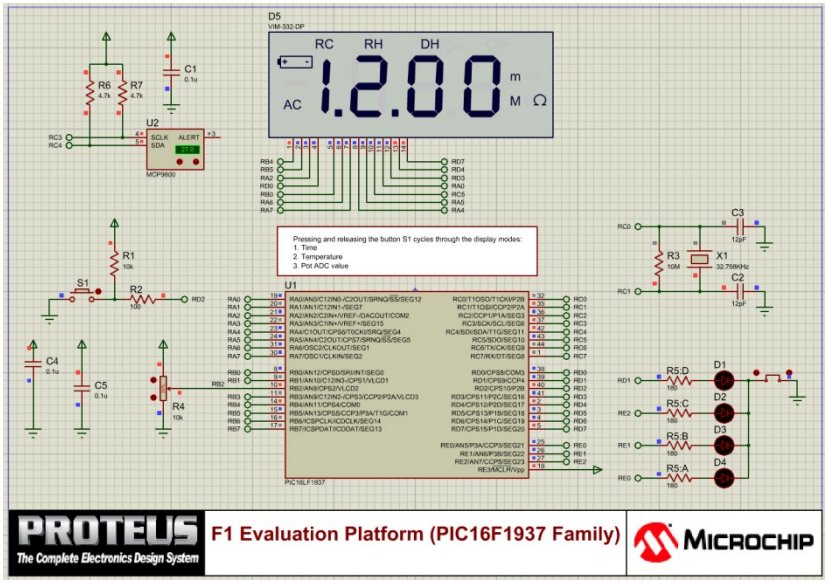
El propósito de este tutorial es introducir al lector en la utilización de PROTEUS VSM y VSM STUDIO IDE, para simular de forma interactiva el funcionamiento de un diseño electrónico, tanto cuando está basado en un microprocesador como cuando no los incluye. Como en todos los tutoriales nuestra intención es, por encima de todo, práctica.

El objetivo que nos hemos fijado es adquirir, en el mínimo tiempo posible, el conocimiento suficiente de las técnicas que pone a nuestra disposición el simulador PROTEUS y el entorno de desarrollo integrado VSM STUDIO IDE. A lo largo de este tutorial nos remitiremos cuando sea necesario a la ayuda en línea suministrada con los diferentes módulos de PROTEUS para profundizar en los diferentes aspectos que vayamos descubriendo.

Este tutorial no cubre la utilización de la pestaña 'esquema electrónico' para realizar el esquema electrónico que sirve de base para nuestra simulación. Por el contrario, damos por sentado que el lector ya sabe utilizarla como herramienta de dibujo de diseños electrónicos antes de introducirse en el mundo de la simulación. Si éste no fuera nuestro caso, debemos primero leer el tutorial correspondiente.

Para llevar a cabo las prácticas, en este tutorial vamos a utilizar un diseño electrónico de una de las tarjetas de evaluación de la firma Microchip. En ella se utiliza como microprocesador el modelo PIC16F1937 de dicha casa.

El esquema electrónico completo que vamos a utilizar (el mismo que la tarjeta de evaluación de Microchip) se muestra en la siguiente imagen.



Como en todos los tutoriales, arrancaremos nuestro trabajo de simulación desde el principio. Utilizaremos el “VSM Studio IDE” para escribir el programa e iremos profundizando de forma progresiva en las técnicas para depurar nuestro software. También iremos conociendo los diferentes equipos de medida virtuales que nos brinda PROTEUS VSM.

1.2.-Requisitos.

Para poder seguir los pasos de este tutorial necesitamos contar con los siguientes elementos:

- Una copia de PROTEUS versión 8 o posterior. Puede descargarse una versión demo del software gratuitamente desde la web de Labcenter: https://www.labcenter.com/downloads/

- Una copia del compilador Hi-Tech PIC16 versión 9.8 o posterior. La descarga e instalación de este compilador se puede gestionar desde el propio software PROTEUS (concretamente desde la pestaña ‘Taller VSM’). Más adelante

en ese tutorial explicaremos el procedimiento para llevarlo a cabo

Aunque no dispongamos de esta segunda herramienta en nuestro ordenador o aunque utilicemos habitualmente en nuestros diseños cualquier otra familia de microprocesadores y compiladores, sigue siendo muy recomendable leer este tutorial y seguir sus ejemplos adaptándolos a nuestros diseños. La mayoría de las técnicas descritas en él son genéricas para todas las familias de microprocesadores soportados por PROTEUS y nos resultará muy útil su conocimiento en el desempeño de nuestro trabajo.

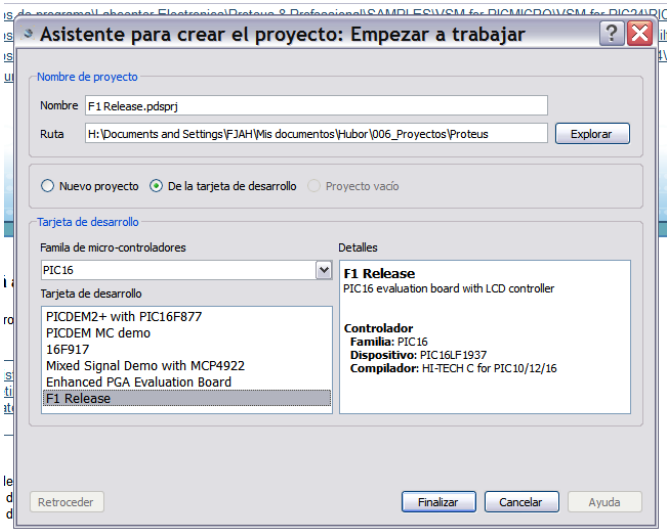
2.-Configuración de nuestro proyecto.

Nuestra primera tarea consiste en configurar nuestro proyecto en PROTEUS. Puesto que vamos a utilizar una tarjeta de desarrollo virtual, el procedimiento es muy sencillo.

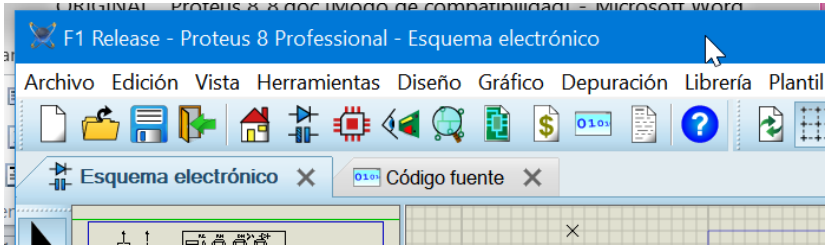
1. Desde la ventana de inicio de Proteus ejecutamos el asistente para la creación de proyectos utilizando el botón 'Nuevo proyecto' de la sección 'Empezar a trabajar'.



2. Lo primero que tenemos que hacer es seleccionar la opción 'De la tarjeta de desarrollo'. A continuación seleccionaremos como familia de microprocesadores 'PIC16'. Y, por último, como tarjeta de desarrollo escogeremos 'F1 release' que se corresponde con la tarjeta de evaluación del PIC16 con controlador LCD.



3. Aceptaremos nuestra selección pulsando el botón ‘Finalizar’ para seguir adelante, si todo está correcto ya tendremos las tres pestañas disponibles, tal y como se muestra en la siguiente pantalla.

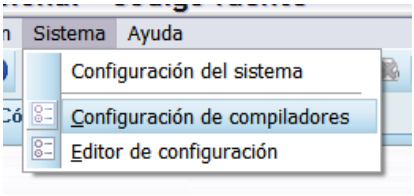


3.-Configuración del compilador.

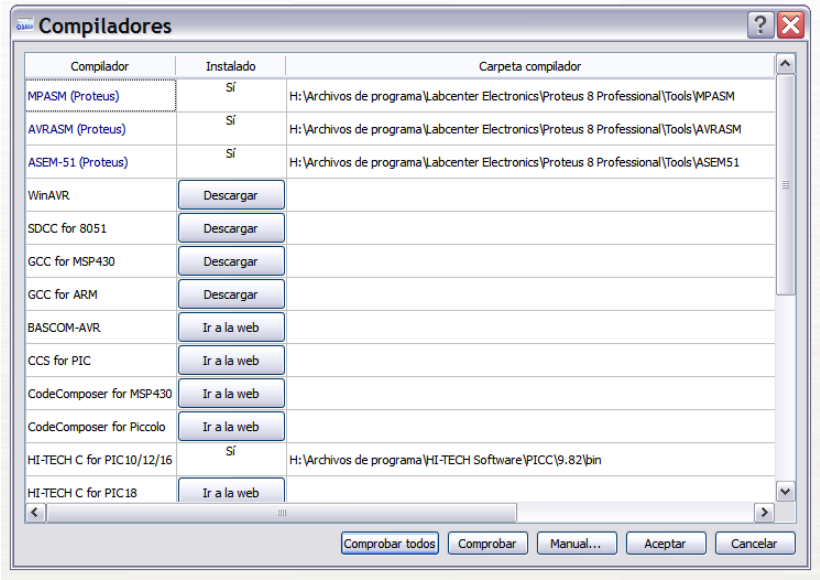
Nuestra siguiente tarea será compilar el código fuente y generar el programa que vamos a utilizar para llevar a cabo nuestra simulación.

El código fuente de la tarjeta de demostración de Microchip se escribió utilizando el compilador HI-Tech PIC16, así que necesitamos instalar esta herramienta para ser capaces de compilar el programa.

Seleccionaremos la pestaña 'Taller VSM' y abriremos la ventana de diálogo de configuración del compilador desde la opción 'Configuración de compiladores' del menú 'Sistema'.



La ventana de diálogo muestra una lista con todos los compiladores soportados por Proteus y nos informa si está o no instalado y configurado.



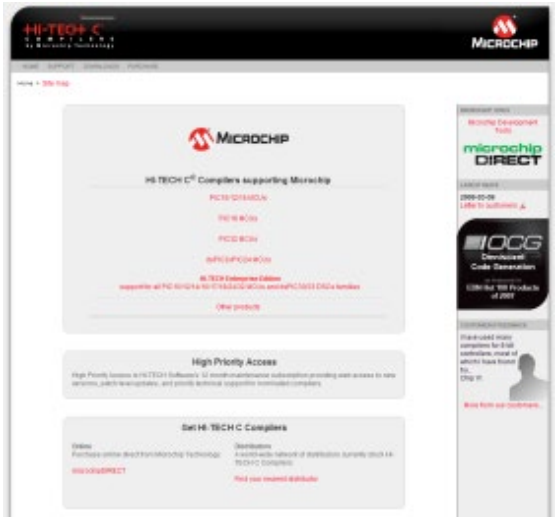
Si utilizamos el botón ‘Comprobar todos’ situado en la zona inferior de la ventana, Proteus realizará una búsqueda en nuestro ordenador para localizar los compiladores ya instalados en nuestro equipo y configurará aquellos que encuentre para poder utilizarlos directamente desde Proteus.

Los compiladores de uso gratuito sin necesidad de adquirir una licencia profesional pueden ser descargados e instalados directamente desde el servidor de Labcenter. Para los compiladores que necesiten de una licencia profesional se muestra un enlace a la página de descarga de la web del fabricante.

El compilador HI-Tech que necesitamos para el proyecto de ejemplo de este tutorial puede ser descargado desde la web de Microchip. Si no lo tenemos instalado previamente en nuestro ordenador debemos seguir los pasos indicados a continuación.

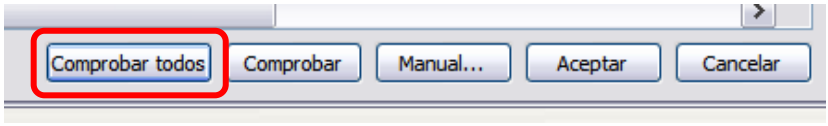
- Los botones rotulados ‘Ir a la web’ nos conducen a la web del proveedor desde donde se puede realizar la descarga.

CodeComposer for MSP430	Ir a la web
CodeComposer for Piccolo	Ir a la web
HI-TECH C for PIC10/12/16	Ir a la web
HI-TECH C for PIC18	Ir a la web

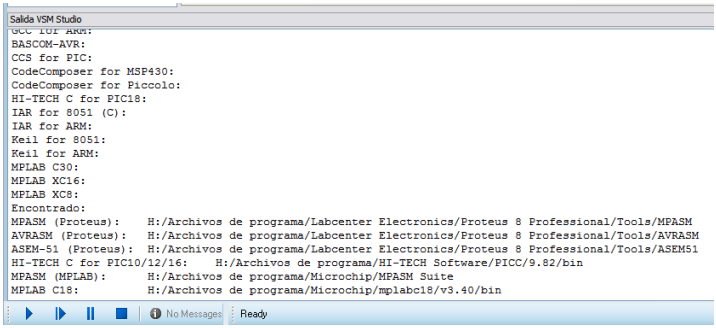


- Descargamos e instalamos en nuestro ordenador el compilador.

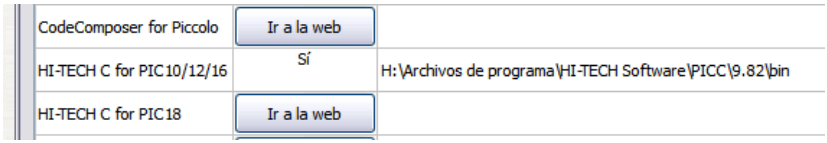
- Utilizamos el botón ‘Comprobar todos’ situado en la zona inferior de la pantalla para configurar el compilador y adaptarlo a Proteus.



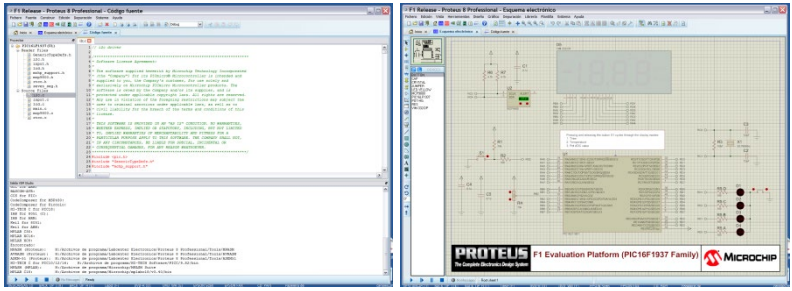
- En la sección ‘Salida VSM Studio’ podemos comprobar el resultado de la operación.



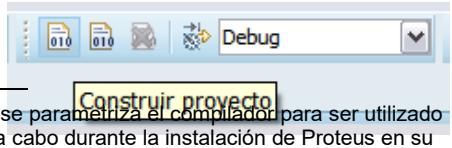
Si todo ha funcionado con normalidad en la línea del listado de compiladores donde aparece HI-Tech para PIC10/12/16 tiene que aparecer que ya está instalado y la ruta donde se encuentra[1].



A estas Alturas ya tenemos el compilador instalado en nuestro ordenador y listo para utilizarlo desde Proteus, el código fuente de nuestro programa (en la pestaña código fuente) y el esquema electrónico de nuestro circuito (en la pestaña esquema electrónico).

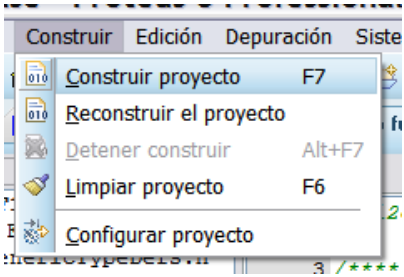


Ya podemos utilizar la opción ‘Construir proyecto’

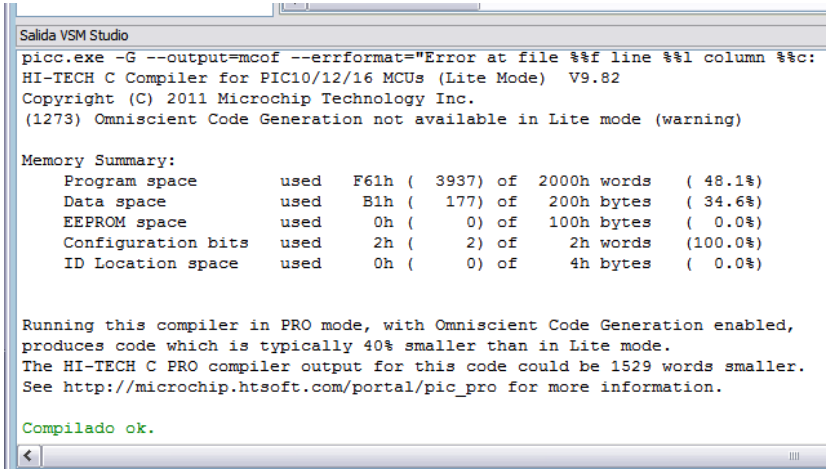


1 Es posible que este proceso donde se parametriza el compilador para ser utilizado desde Proteus ya se hubiera llevado a cabo durante la instalación de Proteus en su ordenador. En este caso, cuando ejecute la opción ‘configuración de compiladores‘, en la ventana de diálogo ya aparecerá el compilador como instalado y listo para usar sin que tengamos que realizar el proceso que acabamos de ver.

del menú ‘Construir’ para compilar nuestro programa. También podemos utilizar el botón ‘Construir proyecto’ de la barra de herramientas superior o el atajo de teclado ‘F7’.



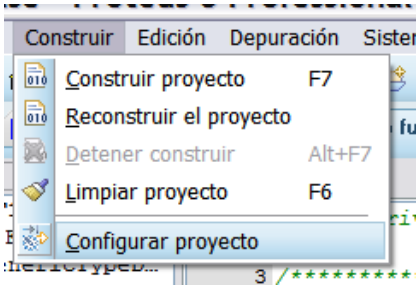
Con cualquiera de estos métodos que elijamos podremos observar que las salidas que genere el compilador al ejecutar el proceso de compilación se visualizarán en el panel inferior titulado ‘Salida VSM Studio’. Si todo ha ido bien, obtendremos un mensaje de ‘compilado ok’.



Nuestro próximo paso será ejecutar la simulación y comprobar que la ejecución de nuestro código fuente en el simulador de nuestra placa de entrenamiento virtual genera los resultados esperados.

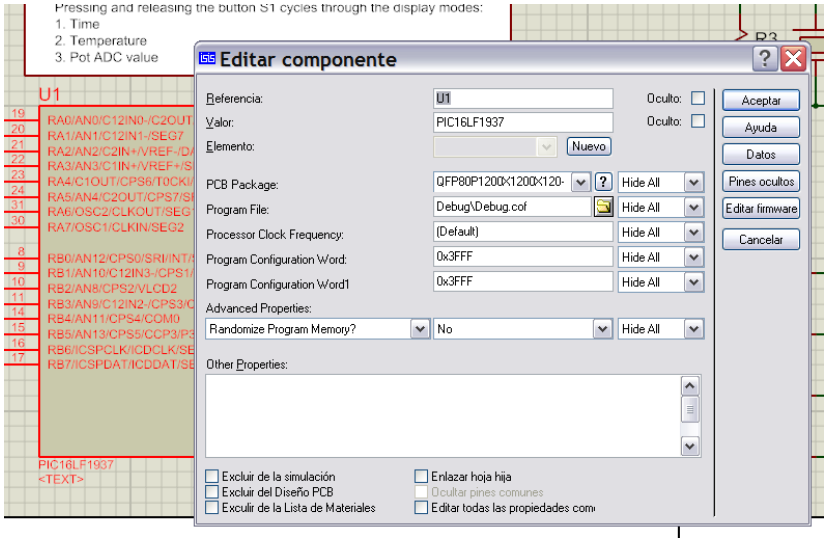
VSM Studio (que se ejecuta en la pestaña código fuente) configura de forma automática las opciones más habituales del compilador para construir correctamente nuestros programas y poder realizar la simulación utilizando Proteus. Si necesitáramos modificar la configuración (v.g. para indicarle la ubicación de

alguna librería externa necesaria para la fase de enlazado) podemos realizarlo utilizando la opción ‘configurar proyecto’ del menú ‘Construir’.



Llegados a este punto tenemos que realizar ciertas consideraciones.

En primer lugar, si no deseamos utilizar PROTEUS VSM Studio como nuestro IDE de programación habitual, indicarle a PROTEUS que deseamos simular el funcionamiento de un programa desarrollado y compilado por nosotros con otra herramienta, es tan sencillo como pulsar con el botón derecho del ratón sobre el microprocesador y seleccionar la opción ‘Editar las propiedades’ en el menú contextual. En la pantalla de diálogo que se abre podemos indicar en el campo ‘Program File’ el fichero que contiene el programa a simular. En el caso que nos ocupa (un microprocesador tipo PIC) podemos utilizar ficheros tipo hex o cof. Si utilizamos el segundo formato (que incluye los nombres de las variables) tendremos disponibles todas las capacidades de depuración utilizables desde PROTEUS.



En segundo lugar, si deseamos saber cómo utilizar otro compilador diferente al propuesto en este tutorial con el VSM Studio IDE, podemos consultar la forma de hacerlo en la ayuda en línea.

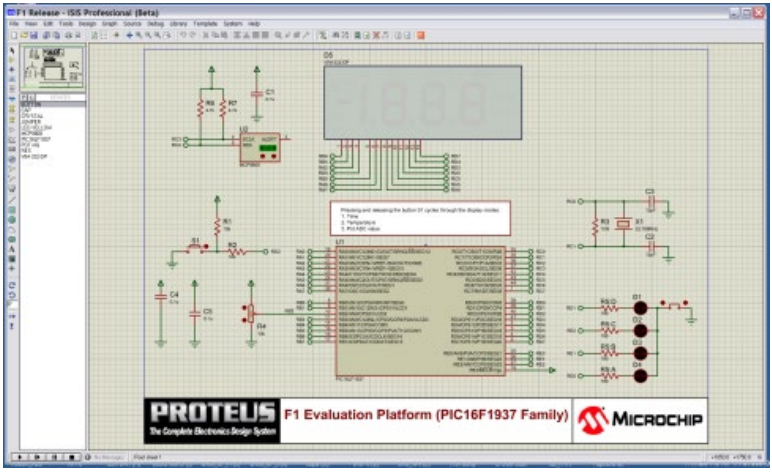
En tercer y último lugar, si deseamos importar uno de nuestros proyectos para utilizarlos desde VSM Studio IDE, podemos consultar el tutorial ‘Primeros pasos con Proteus 8’ donde se explica el proceso para migrar ficheros realizados con versiones anteriores de Proteus (archivos heredados).

A partir de ahora, en este tutorial seguiremos utilizando las herramientas propuestas con objeto de facilitar a todo el mundo, independiente de su conocimiento sobre unas u otras herramientas, el seguimiento de los ejercicios. Puesto que todas ellas son de libre disposición cualquiera puede tener acceso a ellas sin incurrir en un gasto extra de dinero.

Para quien no disponga de experiencia en el uso de herramientas de programación y simulación, les resultará sencillo seguir los pasos propuestos en este tutorial. Para los usuarios expertos en el manejo de otras herramientas o microprocesadores, no le constará nada realizar las prácticas adaptándolas para utilizar aquellos entornos de programación con las que se sientan más familiarizados y a gusto.

4.-Ejecución de la simulación del programa.

Una vez compilado el proyecto, activaremos la pestaña ‘Esquema electrónico’. Tendremos de esta forma a nuestra disposición el esquema electrónico con los distintos componentes enlazados automáticamente con el fichero de salida generado por el compilador para ser ejecutado en el microprocesador con la frecuencia de reloj especificada en la configuración de nuestro proyecto.

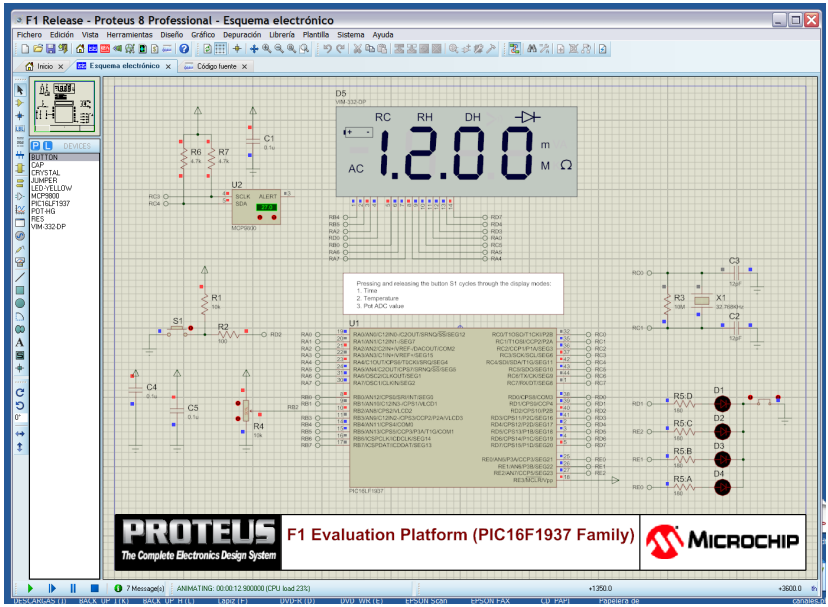


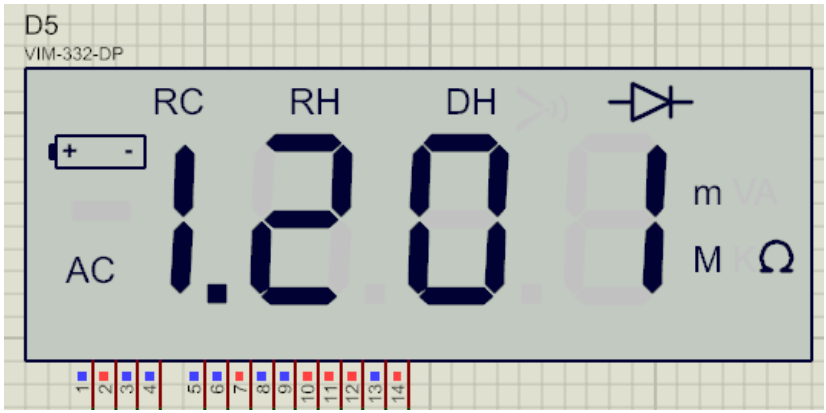
Para arrancar la simulación utilizaremos los controles situados en la zona inferior. Pulsaremos el botón de ‘Play’



Deberíamos poder ver que el esquema electrónico se activa. Lo más fácil de comprobar es que en el display LCD se visualiza un valor y que en los pines de los distintos componentes se puede ver un cuadradito de color rojo o azul (según esté el pin en estado lógico 1 o 0).

En la imagen siguiente recogemos la simulación en ejecución.

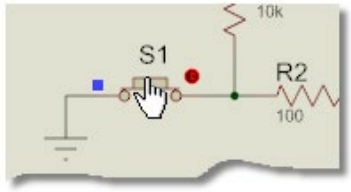




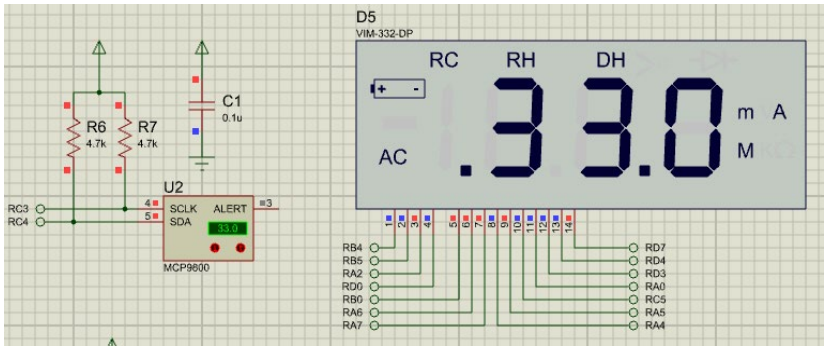
Ahora, una vez arrancada la simulación, podemos llevar a cabo los mismos pasos que podríamos realizar utilizando el kit con la placa de evaluación de Microchip. Podemos utilizar los pulsadores y el potenciómetro disponibles.

Así, por ejemplo, si utilizamos el pulsador S1 arrancaremos la ejecución de la sección del programa que nos sitúa en el modo

temperatura. Observaremos que en la pantalla LCD se visualiza el valor de temperatura medido. Si ajustamos el valor utilizando los controles del dispositivo MCP9800 para cambiar la temperatura, tendremos que observar que los nuevos valores leídos por el microprocesador se visualizan correctamente en el LCD.





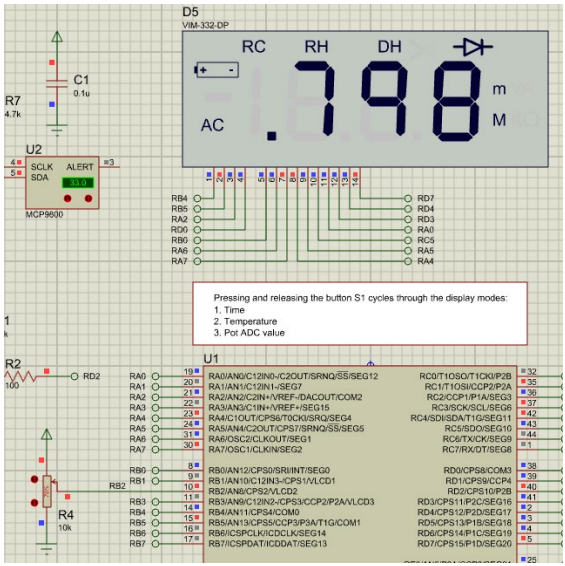


Podemos volver a presionar el pulsador S1 de nuevo para ejecutar la sección del programa del modo convertidor analógico digital para medir la resistencia y ajustar el valor del potenciómetro R4.



De nuevo podremos ver que en la pantalla LCD se muestran los nuevos valores leídos por el microprocesador a medida que vayamos cambiando el estado del potenciómetro. Como estamos utilizando el convertidor analógico digital de 10bits disponible en el

PIC, visualizaremos valores comprendidos entre 0 y 1023 correspondientes a los rangos de voltaje desde 0 a 3,3V.



Terminaremos nuestra simulación de momento pulsando sobre el botón “STOP”



4.1.-Aspectos a tener en cuenta durante la simulación.

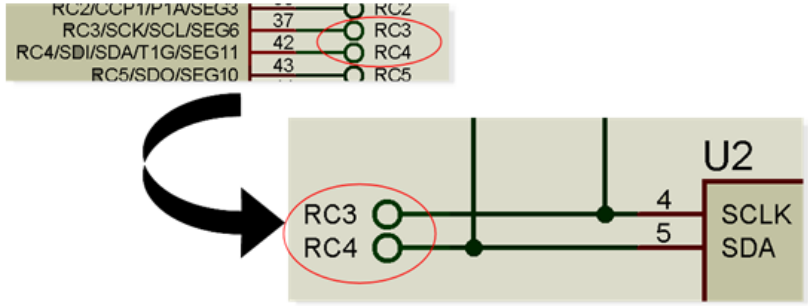
Durante una simulación se llevan a cabo muchas tareas en Proteus que pueden resultar no del todo evidentes para quien la ejecuta, pero que es importante conocer y tener en cuenta.

En primer lugar, el microprocesador que hemos colocado en nuestro esquema electrónico está ejecutando el mismo fichero, resultado de la compilación, y exactamente de la misma forma que un controlador PIC lo ejecutaría en la vida real. Trabajar con Proteus únicamente añade, de forma automática, las tareas necesarias para reflejar en

nuestro esquema electrónico los resultados de la ejecución del programa.

Podemos comprobar que la mayoría de los cables que aparecen en nuestro esquema electrónico acaban con un terminal etiquetado con un nombre. Para aquellos que no estén familiarizados con el funcionamiento de los terminales y su utilización para conectar distintos puntos de nuestro esquema, es un buen momento ahora para dedicar un poco de tiempo a este aspecto.

Siempre que haya dos terminales con el mismo nombre situados en cualquier parte de nuestro esquema electrónico tenemos que entender que están eléctricamente conectados (cómo si existiera un cable virtual que los uniera). Por ejemplo, los terminales RC3 y RC4 unidos al sensor de temperatura (U2) se conectan con los terminales del RC3 y RC4, respectivamente, unidos al procesador PIC (U1). De esta manera el pin SCLK del sensor está eléctricamente unido al pin RC3/SCK/SCL/SEG6 del PIC16LF1937. De la misma forma el pin SDA del sensor está eléctricamente unido al pin RC4/SDI/SDA/T1G/SEG11 del PIC16LF1937.

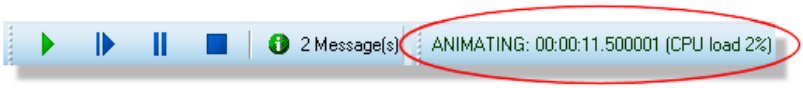


Lo único que hace Proteus es ‘saltar’ y ‘transportar’ la señal entre los dos finales con terminal de forma idéntica a si ellos estuvieran unidos por un cable.

Es bueno acostumbrase a utilizar esta técnica para evitar los típico esquemas ‘espagueti’ donde los cables se entrecruzan dificultando la comprensión del funcionamiento de nuestro diseño. Además, de esta forma, podemos lograr dividir el esquema en pequeños bloques funcionales que facilitan la comprensión de su lógica de trabajo.

Otro aspecto importante a entender es que las señales viajan a través de los cables visibles. Esto significa que podemos colocar una sonda de tensión, una sonda de corriente o un instrumento de medida virtual en cualquier lugar donde veamos un cable para conocer los valores eléctricos en ese punto. Lo veremos con más detalle más adelante en este mismo Tutorial.

Finalmente, conviene fijar nuestra atención en la barra de estado situada en la zona inferior de la pantalla. En la zona derecha se muestra el tiempo que ha transcurrido durante nuestra simulación. Dependiendo de las capacidades de procesamiento de nuestro ordenador personal, la velocidad del reloj que hayamos fijado para el microprocesador que estamos simulando y de la complejidad de nuestro esquema electrónico, es posible que la simulación no pueda realizarse en tiempo real. En estos casos se realizará utilizando una técnica que denominaremos de 'cámara lenta'. Cuando esto ocurre, la sincronización de todos los procesos está asegurada y en la barra de estado se muestra el tiempo de ejecución real en cada instante.



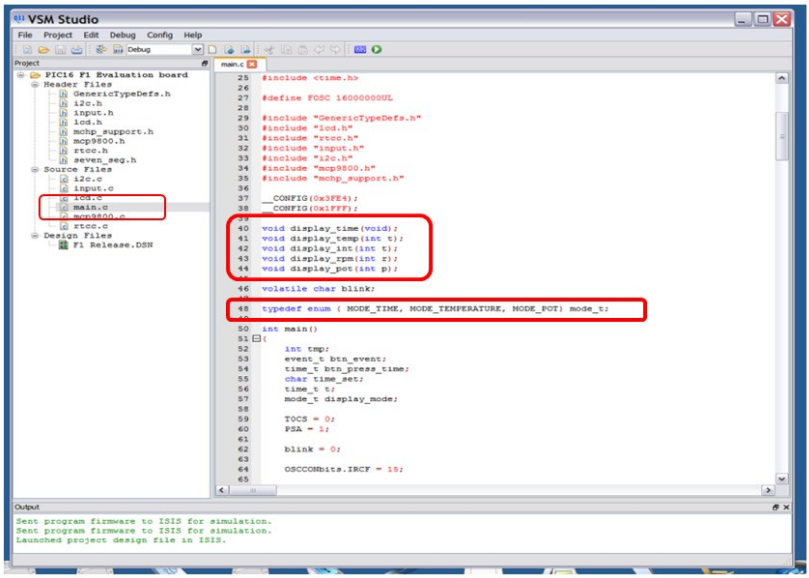
Como ejemplo, si ejecutamos la simulación en un ordenador personal lento podemos observar que el reloj indicador del tiempo de simulación en la barra de estado avanza mucho más despacio que nuestro reloj de pulsera. Podemos, incluso, encontrarnos que simular un segundo de ejecución del procesador puede llevarnos diez segundos del tiempo de nuestro reloj de pulsera (si configuramos la cámara lenta de esta manera). Lo que siempre ocurrirá es que el tiempo reflejado en la barra de estado concuerda exactamente con el tiempo simulado en el microprocesador si éste estuviera funcionando en tiempo real.

5.-Escritura y modificación de los programas.

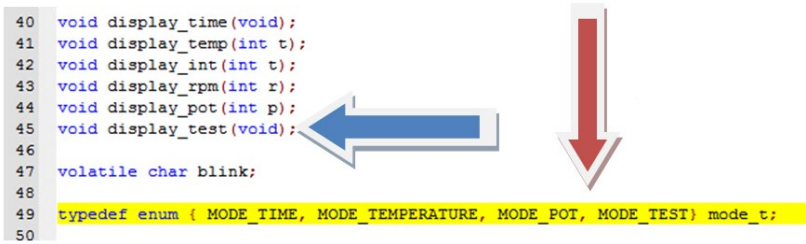
Ahora que ya hemos descubierto los rudimentos de la técnica para llevar a cabo la simulación desde PROTEUS, estamos en condiciones de realizar algunos cambios en el programa.

Con objeto de profundizar en la utilización de las técnicas que pone a nuestra disposición Proteus, vamos a añadir alguna funcionalidad a nuestro programa. Ya hemos visto que presionando el pulsador S1 saltamos a las diferentes secciones del programa que se encargan de visualizar la hora, la temperatura o el valor de la señal generada desde el potenciómetro. Por lo tanto, no nos resultará difícil añadir un nuevo modo de operación.

Abriremos el fichero main.c para editarlo, haciendo una doble pulsación sobre este elemento de esta línea en el panel con el árbol del proyecto. En la zona superior del código, después de la sección de comentarios, podemos ver los prototipos de las funciones y una constante de enumeración con los diferentes modos de operación disponibles.



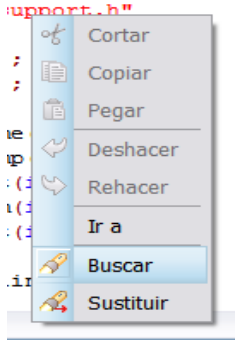
Necesitamos añadir un nuevo prototipo de función para incorporar la nueva funcionalidad a nuestro programa (vamos a llamarla display_test) y el correspondiente nuevo valor (MODE_TEST) en la constante de enumeración.



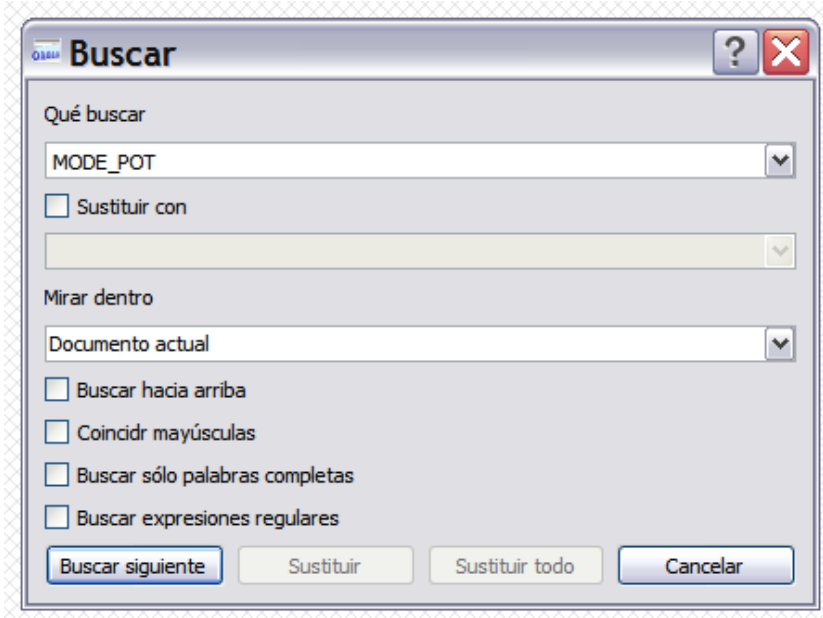
Ahora, tenemos que encontrar el lugar donde se encuentra el código que se encarga de controlar la lógica de funcionamiento del pulsador para cambiar entre los distintos modos de operación disponibles. La forma más sencilla de hacerlo es utilizar la búsqueda para encontrar cualquiera de los modos ya existentes.

Podemos acceder a la búsqueda colocando el cursor en cualquier área de la ventana donde se visualiza el código y pulsando con el

botón derecho del ratón para visualizar el menú contextual. Cuando aparezca seleccionaremos la opción ‘Buscar’. También podemos hacer uso de la opción del mismo nombre desde el menú ‘Edición’.



En la ventana de diálogo que aparece, escribiremos ‘MODE_POT’ en el cuadro de texto ‘Qué buscar’ y pulsaremos el botón ‘Buscar siguiente’.



La primera aparición de la cadena de texto que hemos introducido se encuentra en el trozo de código que nos interesa[2]. El cursor se habrá situado sobre la línea de código 172 (que es la línea donde aparece la cadena de búsqueda introducida) que se enmarca dentro del código de la instrucción “switch” con los tres “case” disponibles (MODE_TEMPERATURE, MODE_POT y MODE_TIME) y donde se define como modo por defecto (default) el MODE_TIME.

```
switch(display_mode)
{
     case MODE_TIME: display_mode = MODE_TEMPERATURE;
          break;
     case MODE_TEMPERATURE: display_mode = MODE_POT;
          break;
     case MODE_POT: display_mode = MODE_TIME;
          break;
     default:
          display_mode = MODE_TIME;
          break;
 }
```

Si analizamos el código con detenimiento, podremos ver que lo que hace es comprobar en qué modo nos encontramos y seleccionar en función de ello el siguiente disponible. Así, con cada nueva presión sobre el pulsador cambiamos al modo de funcionamiento siguiente. Por lo tanto, todo lo que tenemos que hacer es modificar el “case” del “mode_pot” para que apunte al nuevo modo que deseamos crear y añadir una nueva sección “case” con este nuevo modo de funcionamiento apuntando al siguiente “mode” de la cadena. Para ello escribiremos el código que aparece en la imagen recuadrado en rojo y modificaremos el recuadrado en verde.

2 Esto es verdad si acabamos de introducir el código de programa según se indica en este tutorial. Si no aparecemos en este punto, porque nos hemos desplazado a lo largo del código para estudiarlo, podemos utilizar la opción ‘buscar siguiente’ del menú ‘edición’ o el atajo de teclado ‘F3’ hasta que logremos encontrar la coincidencia que deseamos.

```
switch(display_mode)
{
    case MODE_TIME: display_mode = MODE_TEMPERATURE;
        break;
    case MODE_TEMPERATURE: display_mode = MODE_POT;
        break;
    case MODE_POT: display_mode = MODE_TEST;
        break;
    case MODE_TEST: display_mode = MODE_TIME;
        break;
    default:
        display_mode = MODE_TIME;
        break;
}
```

modificar

Justo un poco más abajo de este código, nos encontraremos otro trozo de código con una nueva instrucción “switch”. En él se controla la ejecución de la función correspondiente según el modo de funcionamiento actualmente activo.

```
switch(display_mode)
{
    case MODE_TIME:
        display_time();
        break;
    case MODE_TEMPERATURE:
        display_temp(mcp9800_get_temp());
        break;
    case MODE_POT:
        display_pot(input_pot());
        break;
    default: display_mode = MODE_TIME; break;
}
```

En este punto debemos añadir una nueva sección “case” para que se ejecute nuestra nueva función cuando nos encontremos en el modo que deseamos añadir. Para esto incluiremos el código remarcado en rojo en la siguiente imagen. Más tarde añadiremos el código de esta nueva función al final del fichero.

```
switch(display_mode)
{
    case MODE_TIME:
        display_time();
        break;
    case MODE_TEMPERATURE:
        display_temp(mcp9800_get_temp());
        break;
    case MODE_POT:
        display_pot(input_pot());
        break;
    case MODE_TEST;
        display_test();
        break;
    default: display_mode = MODE_TIME; break;
}
```

El último paso consiste en escribir la función “display_test” con el código que se deberá ejecutar cada vez que entremos en el nuevo modo de operación que hemos creado. La añadiremos al final del fichero. Evidentemente, aquí podemos escribir la función qué deseemos. A los efectos de este tutorial nos servirá escribir un sencillo código que escriba el valor ‘1111’ en la pantalla LCD, que cree un contador binario que se incremente cada nuevo bucle del programa y que visualice el valor de este contador con ayuda de los LEDs D2, D3 y D4. El código que debemos introducir al final de nuestro fichero para la función display_test, es el siguiente.

```
void display_test(void)
{
    static int i,d;

    display_int(1111);
    if( ++d > 400 )
    {
        d = 0;
        PORTE = i;
        i = ( i+1 ) % 8;
    }
}
```

Es fácil observar que estamos utilizando el puerto E de nuestro microprocesador PIC (PORTE) para mostrar el valor del contador en forma binaria utilizando los leds conectados a los pines de este puerto.

Por esta razón, también necesitamos configurar en nuestro software los pines de este puerto. Lo haremos asignado el valor 0 al registro ANSELE en la sección dedicada a la configuración de los periféricos, situadas en la zona superior de la función main().

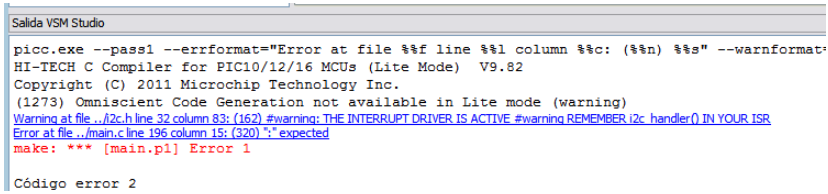
```
67      // peripheral configuration
68      PORTC = 0;
69      PORTD = 0;
70      PORTE = 0;
71      TRISC2 = 0;
72      TRISE0 = 0;
73      TRISE1 = 0;
74      TRISE2 = 0;
75      TRISD5 = 0;
76      TRISD6 = 0;
77
78      ANSA1 = 1;
79      ANSA3 = 1;
80      ANSB3 = 1;
81      ANSB1 = 1;
82      ANSELE = 0;
83
84      rtcc_init();
85      lcd_init();
86      input_init();
87
88      i2c_init();
89
90      PEIE = 1;
91      GIE = 1;
```

La última parte de nuestro trabajo consistirá en compilar todo el proyecto y ejecutar ISIS como ya describimos con anterioridad en esta misma guía. Recordemos que debemos usar el botón ‘Construir proyecto’ situado en la barra de herramientas o la opción ‘Construir proyecto’ del menú ‘Construir’ para compilar el proyecto.

Si hemos cometido algún error el compilador nos mostrará los mensajes de error correspondientes en el panel ‘salida VSM Studio’.

También nos mostrará mensajes de aviso (warning) en el caso de detectar algún problema que no se considere crítico.

Nosotros hemos forzado un mensaje de error y otro de aviso para tener un ejemplo. Así que al compilar debemos recibir un mensaje en la ventana de salida similar al que se muestra en la imagen siguiente.



Si pulsamos con el botón izquierdo del ratón con el cursor situado sobre el error[3], automáticamente VSM Studio nos situará sobre la sección del código donde se encuentra el origen del error.

En nuestro caso, hemos escrito un punto y coma (;) al final de la línea de código de una clausula CASE en lugar de los preceptivos dos puntos (:).

```
switch(display_mode)
{
    case MODE_TIME:
        display_time();
        break;
    case MODE_TEMPERATURE:
        display_temp(mcp9800_get_temp());
        break;
    case MODE_POT:
        display_pot(input_pot());
        break;
    case MODE_TEST;
        display_test();
        break;
    default: display_mode = MODE_TIME; break;
}
```

Algunos errores, como por ejemplo los errores generados por el enlazador (linker), no son susceptibles de que VSM Studio nos sitúe sobre ellos cuando pulsamos con el botón izquierdo del ratón sobre el mensaje de error.

3 Observar que la línea que muestra el error comienza por 'error at file…' mientras que la línea donde se muestra el aviso comienza por 'warning at file...'.

En estos casos tendremos que cambiar las opciones utilizando los comandos de configuración del proyecto o la información almacenada en los ficheros de cabecera.

Modifiquemos ahora, la línea de código errónea y volvamos a compilar nuestro proyecto. En la ventana de salida nos tendrá que aparecer un mensaje de que la compilación se ha llevado a cabo correctamente.

```
Microchip MPLAB XC8 C Compiler (Free Mode) V2.00
Build date: May 27 2018
Part Support Version: 2.00
Copyright (C) 2018 Microchip Technology Inc.
License type: Node Configuration

(1273) Omniscient Code Generation not available in Free mode (warning)
C:\Program Files (x86)\Microchip\xc8\v2.00\pic\sources\c90\common\Umul16.c: 15: non-reentrant funct
C:\Program Files (x86)\Microchip\xc8\v2.00\pic\sources\c90\common\Umul8.c: 4: non-reentrant functio

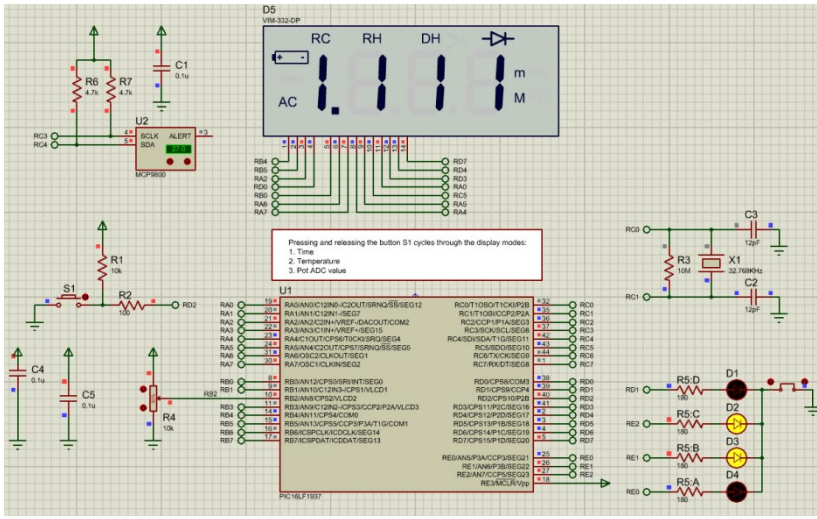
Memory Summary:
    Program space        used   D40h (  3392) of  2000h words   ( 41.4%)
    Data space           used    B2h (   178) of   200h bytes   ( 34.8%)
    EEPROM space         used     0h (     0) of   100h bytes   (  0.0%)
    Data stack space     used     0h (     0) of   135h bytes   (  0.0%)
    Configuration bits   used     2h (     2) of     2h words   (100.0%)
    ID Location space    used     0h (     0) of     4h bytes   (  0.0%)

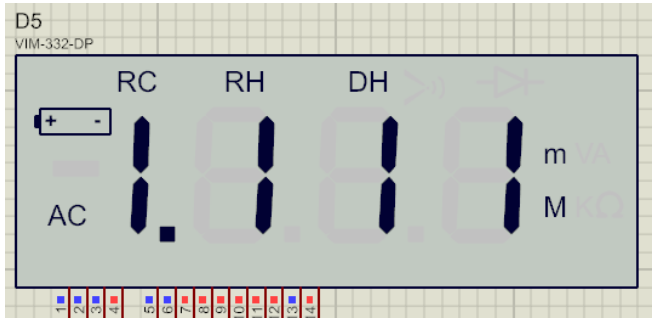
rm startup.as
rm Debug.as
Compilado ok.
```

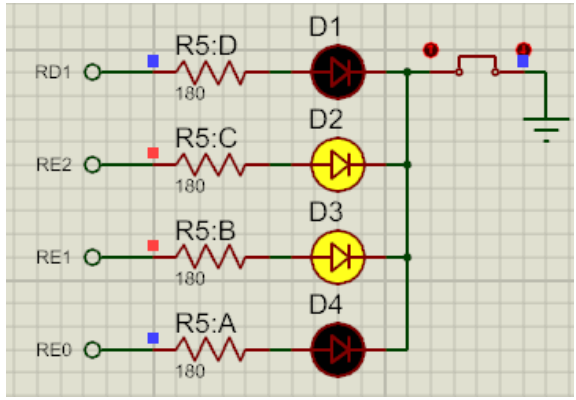
Podemos observar que el mensaje de aviso permanece, pero eso no impide que el compilador termine su labor satisfactoriamente. El compilador no generará código con errores, pero sí lo hará con avisos.

Ahora ya estamos en situación de volver a llevar a cabo la simulación con PROTEUS utilizando el botón de la herramienta correspondiente situado en la barra de herramientas.

Si pulsamos varias veces sobre el pulsador S1 iremos pasando por los diferentes modos de operación disponibles. Cuando lleguemos al modo que acabamos de crear tenemos que ver el valor 1111 en la pantalla LCD y podemos observar como los leds D2, D3 y D4 van iluminándose y apagándose para visualizar los distintos valores de nuestro contador en código binario.


D5
VIM-332-DP
RC
RH
DH
AC
m
M
U1
PIC16LF1937
Pressing and releasing the button S1 cycles through the display modes:
1. Time
2. Temperature
3. Pot ADC value
R1
10k
R2
100
R3
10M
X1
32.768KHz
C1
0.1u
C2
12pF
C3
12pF
C4
0.1u
C5
0.1u
R4
10k
R5:D
R5:C
R5:B
R5:A
180
D1
D2
D3
D4
S1
U2
MCP9800


D5
VIM-332-DP
RC
RH
DH
AC
m
M


RD1
RE2
RE1
RE0
R5:D
R5:C
R5:B
R5:A
180
D1
D2
D3
D4

Pulsemos ahora el botón STOP de los controles de la simulación situados en la zona inferior izquierda de la ventana para detener la simulación. Evidentemente, esto solo ha sido un pequeño ejemplo (bastante simple y no demasiado emocionante) para mostrar las posibilidades que nos brinda PROTEUS de escribir y simular código en nuestro hardware virtual. Pero, es evidente, que podemos escribir, utilizando las mismas técnicas que acabamos de aprender, programas tan complejos como necesitemos y nos permita la potencia de nuestro microprocesador.

6.-Las ventanas emergentes activas.

Hasta este momento hemos sido capaces de escribir nuestro código utilizando la pestaña ‘Código fuente’ y ejecutar la simulación en la pestaña ’Esquema electrónico’. No hay nada malo en trabajar de esta manera. Incluso, si disponemos de dos monitores, podemos ‘coger y arrastar’ una de las pestañas al segundo monitor, con lo que podemos visualizar simultáneamente el código fuente y la simulación.

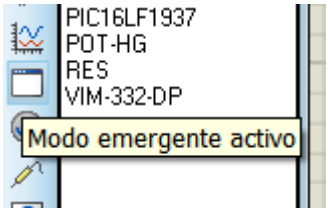
Sin embargo, cuando estamos depurando el código de nuestro proyecto nos encontraremos muchas veces que nos resulta muy útil poder ir depurando paso a paso nuestro programa y fijar nuestra atención en la simulación de pequeñas secciones de nuestro esquema electrónico para verificar el resultado. La técnica de ventanas emergentes activas que nos proporciona Proteus está diseñada específicamente para posibilitarnos utilizar la visualización de pequeñas porciones del esquema electrónico dentro de la pestaña ‘Código fuente’ mientras ejecutamos la simulación.

En el diseño que estamos utilizando en este tutorial, el sensor de temperatura puede ser un buen ejemplo de una porción del esquema electrónico que nosotros podríamos querer ver e interactuar sobre él durante la depuración. Así que vamos a crear una ventana emergente activa y aprenderemos a utilizarlas.

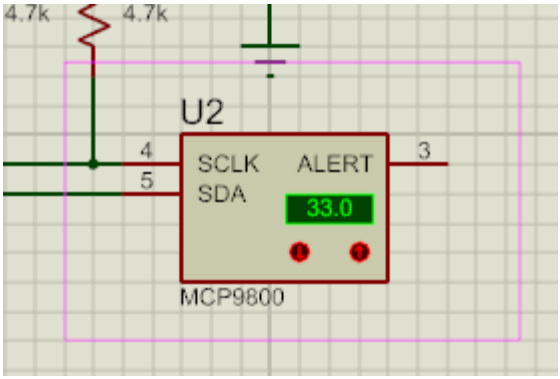
Detengamos la simulación, si no lo hicimos todavía, con los controles de la zona inferior de la ventana.



Primero seleccionaremos el botón ‘Modo emergente activo’ de la barra de herramientas lateral.

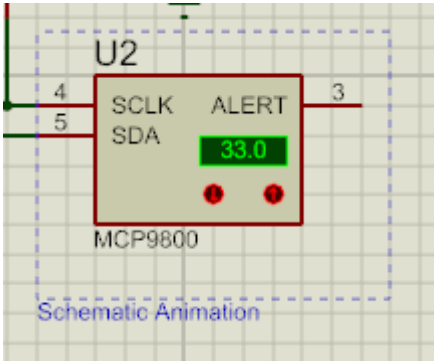


A continuación trazaremos un recuadro donde seleccionaremos nuestro dispositivo de lectura de temperatura.

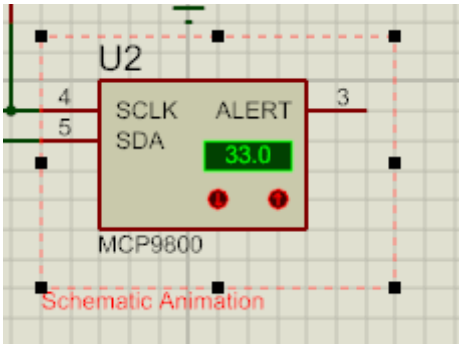


Cuando lo tengamos hecho y contenga la parte de pantalla que nos interesa, pulsaremos de nuevo con el botón del ratón izquierdo para terminar la selección.

Veremos que la zona seleccionada se indica con un recuadro azul de trazos.

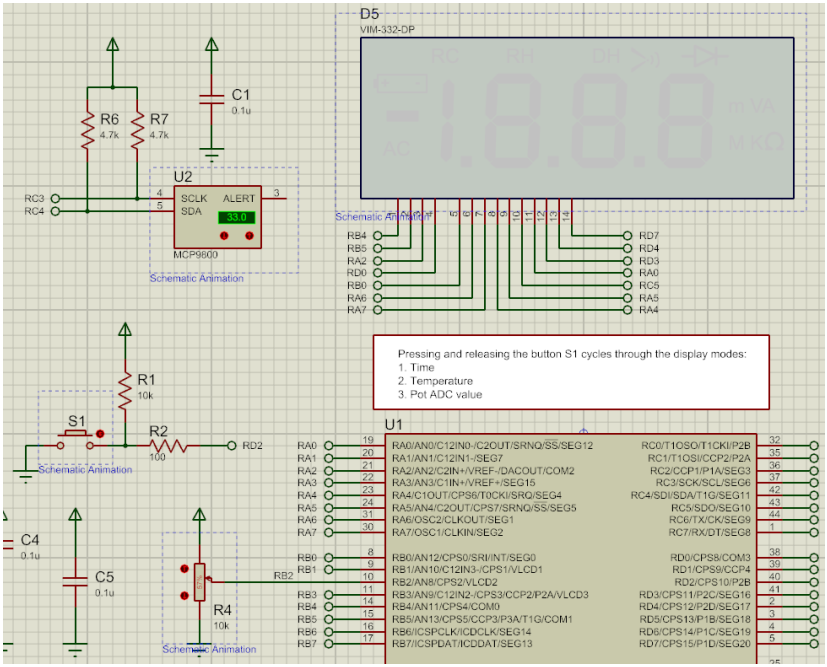


Si nos hubiéramos equivocado podemos volver a seleccionar con el ratón este recuadro y cambiar su tamaño y posición con ayuda de los ‘selectores’ que aparecen.

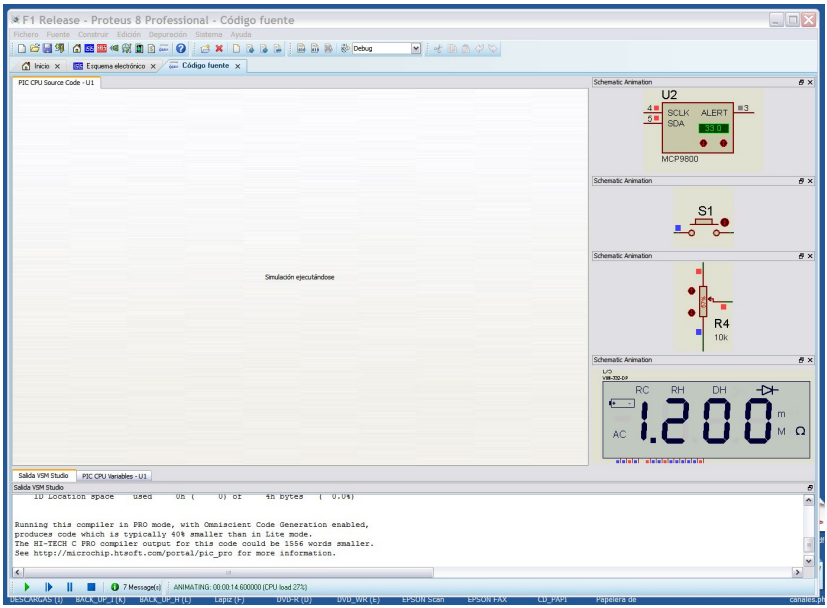


Otros buenos candidatos para estar en ventanas activas emergentes son el pulsador de selección de modo de trabajo, el potenciómetro y el propio display LCD.

Así que podemos practicar nuestras destrezas con esta técnica trazando cajas adicionales alrededor de estos elementos. Al final nuestro esquema electrónico debería tener un aspecto similar al que se muestra en la siguiente imagen.



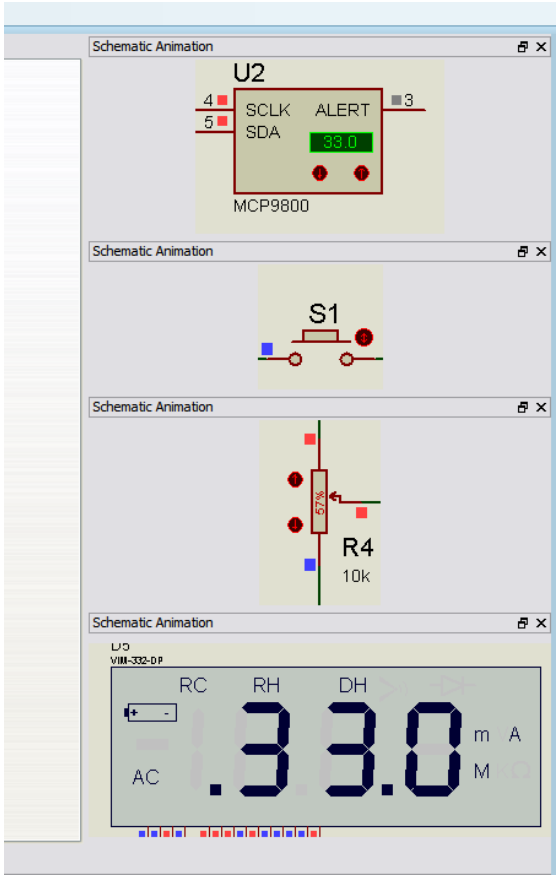
Una vez que hemos seleccionado ventanas emergentes activas con las que supuestamente deseamos trabajar, Desearemos trabajar desde la pestaña ‘Código fuente’. Así que, iremos a la pestaña ‘Código fuente’ y pulsaremos de nuevo el botón ‘Play’ de los controles de animación situados en la zona inferior de la ventana. Veremos que ha arrancado la simulación y que en la zona derecha de la ventana se muestran tantas ventanas emergentes activas como hemos definido sobre nuestro esquema electrónico.



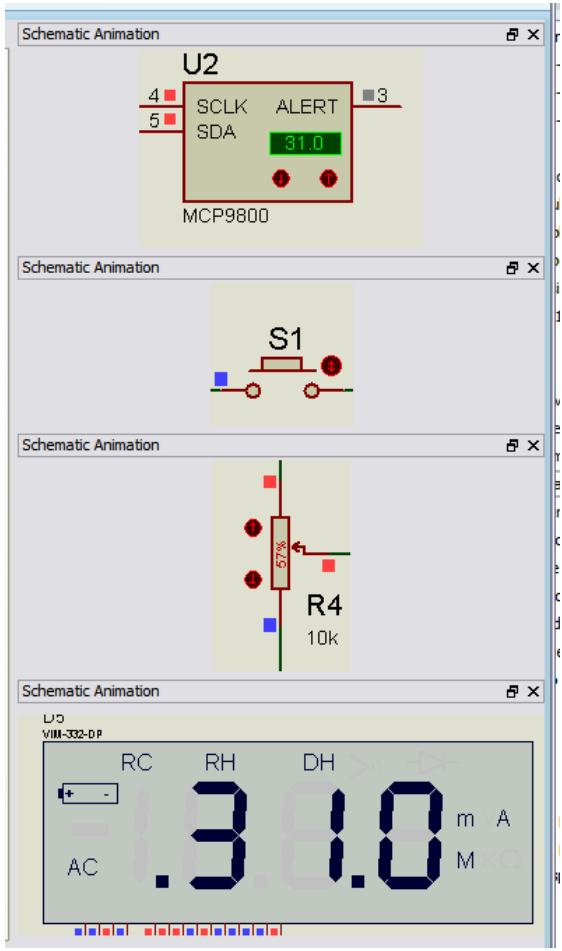
En el panel central de la pestaña sólo se puede leer un mensaje que nos informa que la simulación está ejecutándose. Este decepcionante resultado no debe desanimarnos. En el próximo capítulo veremos las enormes posibilidades que Proteus nos brinda para la depuración y cómo podemos ir viendo los cambios que se generan en nuestro código mientras este se va ejecutándose.

En este punto, solo queremos centrar nuestra atención en lo útiles que resultan las ventanas activas emergentes porque nos posibilitan que seamos capaces de ver trozos de nuestro esquema electrónico en tiempo real durante la simulación en el mismo entorno donde estamos llevando a cabo la depuración.

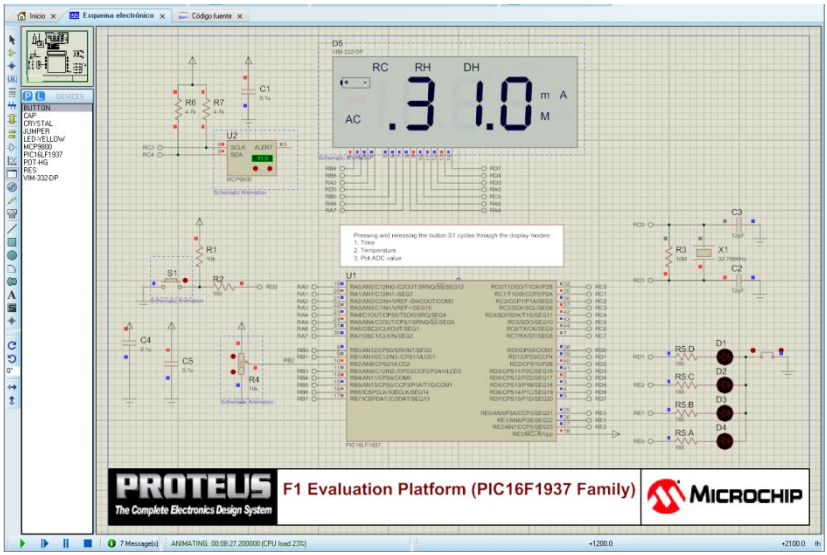
Pero, es que además, las ventanas activas emergentes son intenligentes y también podemos interactuar activamente con ellas. Por ejemplo, podemos actuar sobre el pulsador S1 para cambiar de modo de funcionamiento y pasar a la presentación de la lectura de temperatura.



Como es lógico, ahora podemos modificar el valor de temperatura en el dispositivo MCP9800 y comprobar que los cambios se llevan a cabo en tiempo real.



Además, si volvemos en cualquier momento a la pestaña ‘Esquema electrónico’ podremos comprobar que nos presenta exactamente el mismo estado de la simulación que muestran las ventanas emergentes activas.



Cuando detengamos la simulación utilizando el botón ‘STOP’ de los controles de animación, las ventanas emergentes activas desaparecerán de la pestaña ‘código fuente’ y el panel donde se muestra la depuración del código de esta pestaña se sustituirá por los dos donde se muestra el árbol de archivos de nuestro proyecto de programa y la ventana de edición del código donde podemos volver a modificarlo a nuestro gusto.

Sólo se puede crear ventanas activas emergentes desde la pestaña ‘esquema electrónico’ y mientras la simulación no se esté ejecutando.

Lo que sí se puede hacer mientras se ejecuta la simulación es modificar el tamaño de las ventanas activas emergentes para que muestren más o menos elementos de nuestro esquema electrónico[4].

Ahora que ya disponemos de nuestras ventanas activas emergentes correctamente configuradas, podemos empezar con la depuración de nuestro programa.

4 Esta operación es posible realizarla cuando nuestro ordenador personal es suficientemente potente para permitir ejecutar la simulación y modificar el tamaño de las ventanas activas emergentes. Nosotros aconsejamos, de todas formas, realizar los ajustes de tamaño de las ventanas activas emergentes con la depuración detenida.

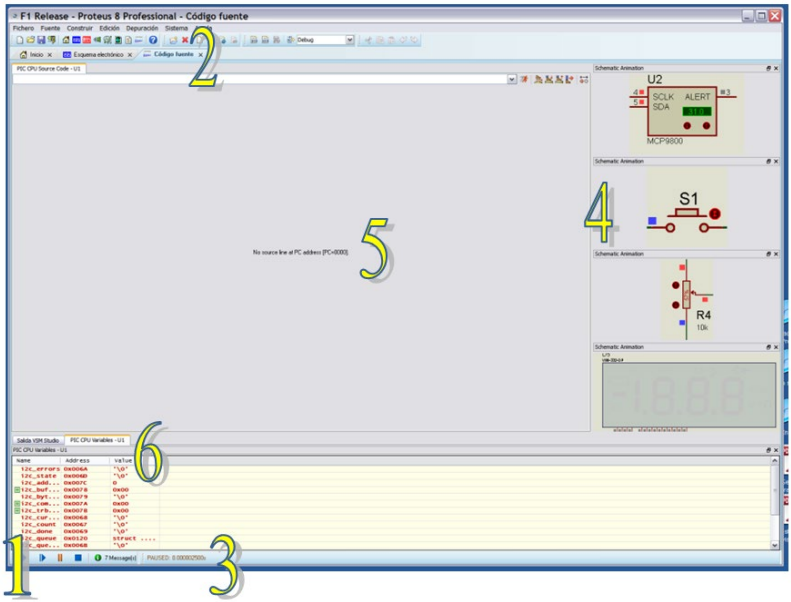
En la siguiente sección vamos a descubrir las posibilidades que nos brinda PROTEUS para los casos en los que las cosas no funcionan como nos gustaría a la primera.

7.-Principios básicos de la depuración de programas.

Gran parte de la potencia de usar PROTEUS VSM la vamos a descubrir cuando empecemos a utilizar sus herramientas para la depuración de nuestros programas. Ya vimos en secciones anteriores cómo podemos escribir el código de nuestro proyecto y cómo podemos comprobar su funcionamiento mediante la simulación sobre el equipo virtual construido con PROTEUS. En esta sección vamos a profundizar en el uso de las técnicas que nos brinda PROTEUS VSM para ir navegando a través de nuestro código mientras se está ejecutando la simulación.

1. Comenzaremos la simulación utilizando el botón “PAUSE” de los mandos de control de la simulación en lugar de seguir el proceso visto hasta ahora de utilizar el botón “PLAY”.



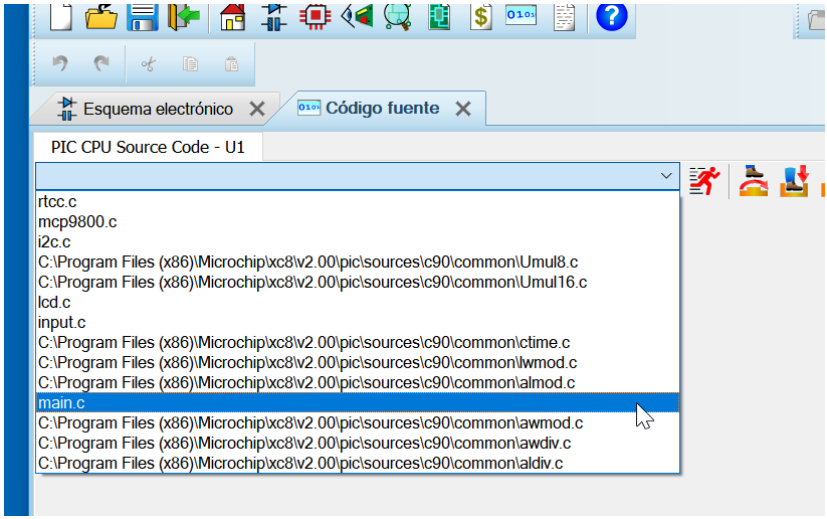


2. El programa activará automáticamente la pestaña ‘código fuente’.
3. Podemos fijarnos que en la barra de estado se dice que la simulación está detenida (PAUSED) en el momento prácticamente cero de la simulación.
4. En la zona derecha de la pantalla se pueden ver las ventanas activas emergentes mostrando la situación de los elementos de nuestro esquema electrónico en su estado actual de simulación.
5. En el panel principal, donde se visualiza el código fuente, se puede ver un mensaje que dice ‘no source líne at PC address (PC=000)’ que quiere decir que no hay ninguna línea de código en la dirección 0 del contador de programa.
6. En la zona inferior de la ventana hay un panel con dos pestañas. Si activamos la pestaña ‘PIC CPU variables – U1’ podemos ver una lista con todas las variables definidas en nuestro proyecto y su actual valor.

En este momento de la simulación sólo se ha producido el arranque hasta un punto estable del sistema pero todavía no se ha ejecutado ninguna instrucción de nuestro programa y no se ha producido ninguna variación en el contador del tiempo real.

En la ventana donde se visualiza el código fuente no se mostrará ningún fichero todavía puesto que el valor del contador de línea de programa no apunta todavía a ninguna línea de programa válida y contiene el valor 0.

En el panel principal donde se muestra el código fuente podemos encontrar un cuadro de diálogo desplegable. Desde él podemos seleccionar cualquier fichero de los que contiene el código fuente de nuestro proyecto.



Vamos a seleccionar el fichero “main.c” que es el que contiene el programa principal con el bucle de control.

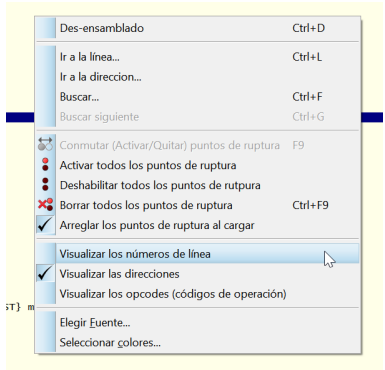
PIC CPU Source Code - U1

main.c

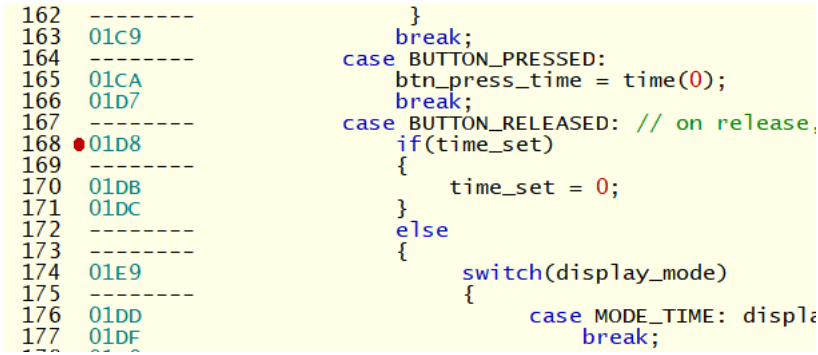
```
---- /*****************************************************************************
---- * Software License Agreement:
---- *
---- * The software supplied herewith by Microchip Technology Incorporated
---- * (the "Company") for its PICmicro® Microcontroller is intended and
---- * supplied to you, the Company's customer, for use solely and
---- * exclusively on Microchip PICmicro Microcontroller products. The
---- * software is owned by the Company and/or its supplier, and is
---- * protected under applicable copyright laws. All rights are reserved.
---- * Any use in violation of the foregoing restrictions may subject the
---- * user to criminal sanctions under applicable laws, as well as to
---- * civil liability for the breach of the terms and conditions of this
---- * license.
---- *
---- * THIS SOFTWARE IS PROVIDED IN AN "AS IS" CONDITION. NO WARRANTIES,
---- * WHETHER EXPRESS, IMPLIED OR STATUTORY, INCLUDING, BUT NOT LIMITED
---- * TO, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
---- * PARTICULAR PURPOSE APPLY TO THIS SOFTWARE. THE COMPANY SHALL NOT,
---- * IN ANY CIRCUMSTANCES, BE LIABLE FOR SPECIAL, INCIDENTAL OR
---- * CONSEQUENTIAL DAMAGES, FOR ANY REASON WHATSOEVER.
---- *****************************************************************************/
----
---- #include <pic.h>
---- #include <stdlib.h>
---- #include <time.h>
----
---- #define FOSC 16000000UL
----
---- #include "GenericTypeDefs.h"
---- #include "lcd.h"
---- #include "rtcc.h"
---- #include "input.h"
---- #include "i2c.h"
---- #include "mcp9800.h"
---- #include "mchp_support.h"
----
---- __CONFIG(0x3FE4);
---- __CONFIG(0x1FFF);
----
---- void display_time(void);
---- void display_temp(int t);
---- void display_int(int t);
---- void display_rpm(int r);
---- void display_pot(int p);
---- void display_test(void);
----
```

Vamos a empezar con una técnica sencilla de depuración, colocando un punto de ruptura en el bucle principal del programa cuando se controla si se ha dejado de actuar sobre el pulsador.

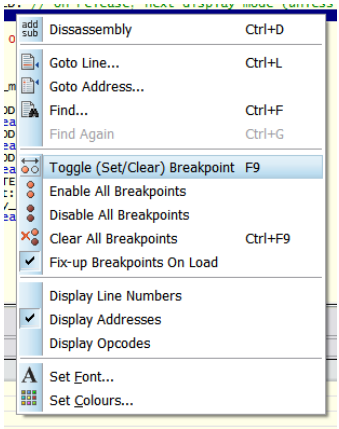
Si no ve los números de línea, puede habilitar esta opción desde el menú contextual.



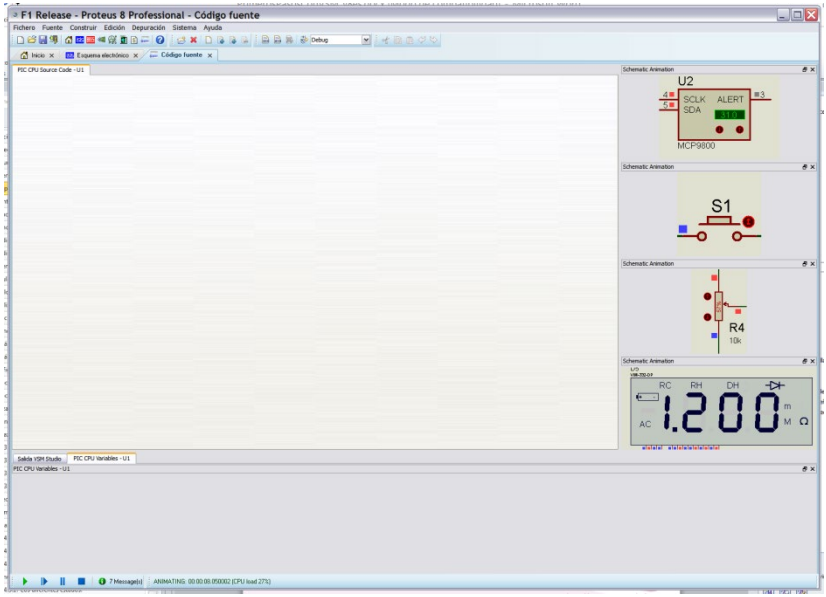
La forma más sencilla de colocar o de quitar un punto de ruptura es hacer una doble pulsación con el botón izquierdo del ratón sobre la línea de código del fichero fuente en la que deseamos hacer la pausa en la ejecución de nuestro proyecto. Al hacerlo podemos observar un indicador de la situación del punto de ruptura (un punto rojo) a la izquierda del código junto a la línea de programa en cuestión. Si nos equivocamos al colocar el punto de ruptura podemos eliminarlo volviendo hacer una doble pulsación con el botón izquierdo del ratón sobre la línea de código donde está situado el punto de ruptura. La primera doble pulsación coloca un punto de ruptura. La segunda lo desactiva y la tercera lo elimina.



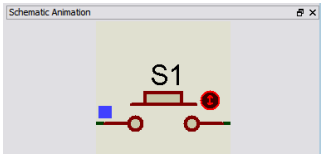
También podemos utilizar el menú contextual pulsando con el botón derecho del ratón sobre la línea de código y eligiendo la opción 'Toggle (Set/Clear) Breakpoint).



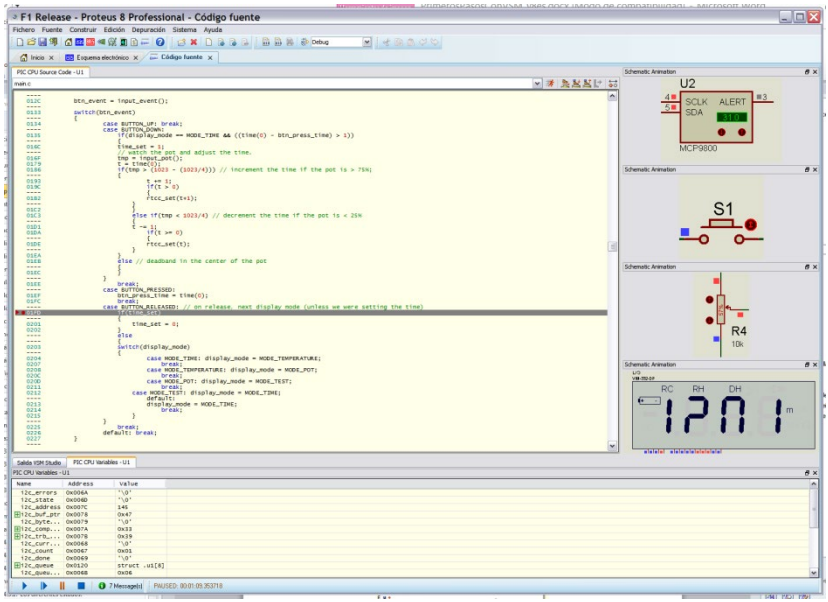
Si ahora ponemos en marcha la simulación (utilizando el botón de PLAY) observaremos que la simulación corre normalmente y que las ventanas con el código fuente y los valores de las variables desaparecen.



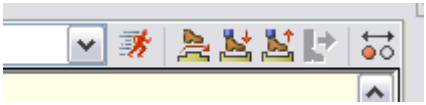
La simulación seguirá corriendo hasta que actuemos sobre el pulsador S1 y lo liberemos. Podemos hacerlo desde la ventana activa emergente.

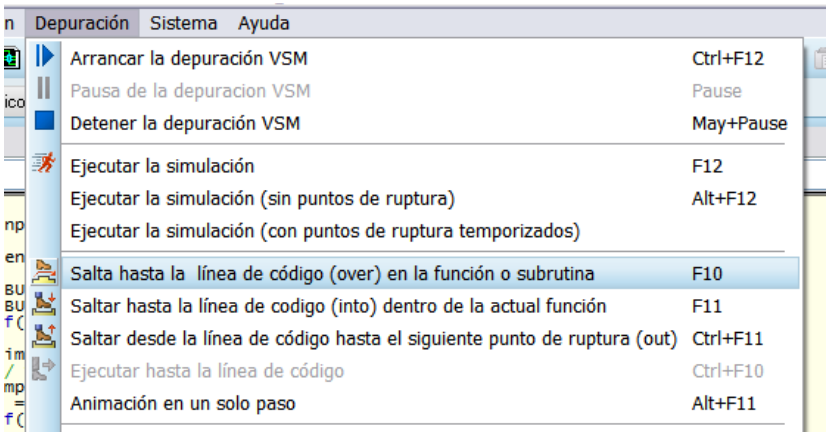


Justo en el momento en que liberemos el pulsador, la simulación se detendrá y volverán a visualizarse el panel con el código del programa y el panel donde se muestran las variables con los datos actualizados al momento en que se ha detenido la ejecución.

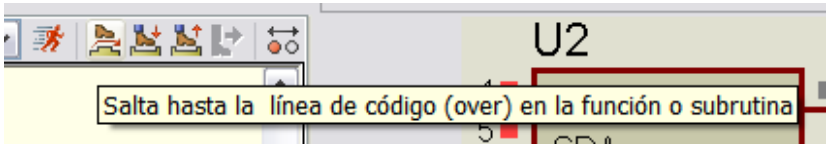


También es posible ejecutar el código por saltos simples de una sola línea de código utilizando los comandos que se encuentran situados en la zona superior de la ventana de código fuente, las opciones del menú ‘Depuración’ o los atajos de teclado “F10” y “F11”

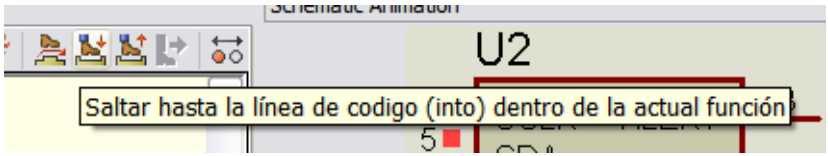




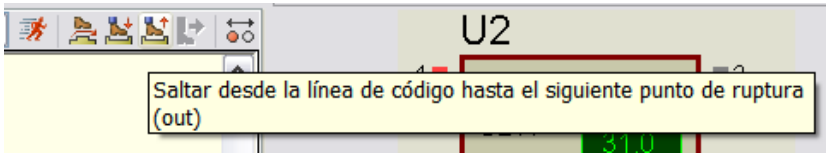
'Saltar hasta la línea de código (over) en la función o subrutina', atajo de teclado 'F10', genera la ejecución del programa hasta la siguiente línea del fichero que estemos visualizando en la ventana de código.



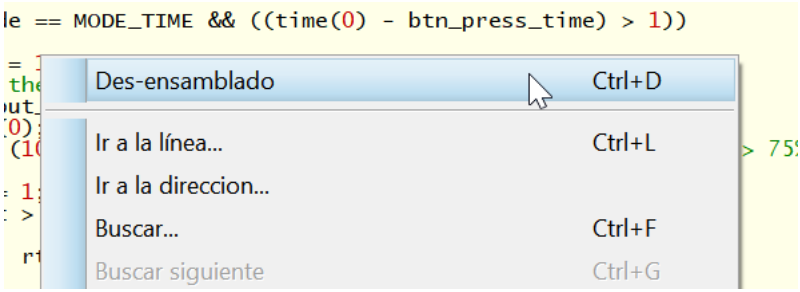
'Saltar hasta la línea de código (into) dentro de la actual función', atajo de teclado 'F11', genera la ejecución del programa hasta la siguiente línea de código pertenezca a ese fichero o a otro de los que componen el proyecto.



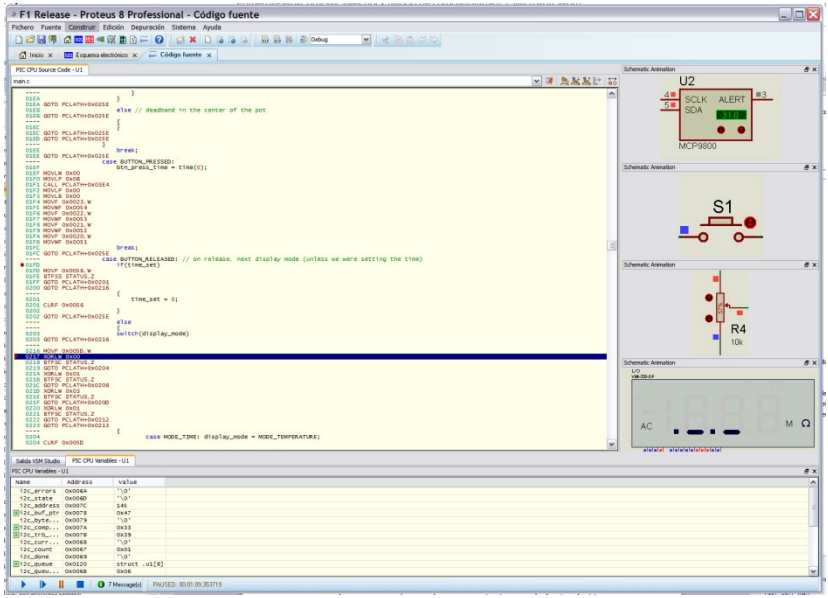
'Saltar desde la línea de código hasta el siguiente punto de ruptura (out)', atajo de teclado 'CTRL-F11', genera la ejecución del programa hasta que se encuentre un nuevo punto de ruptura.



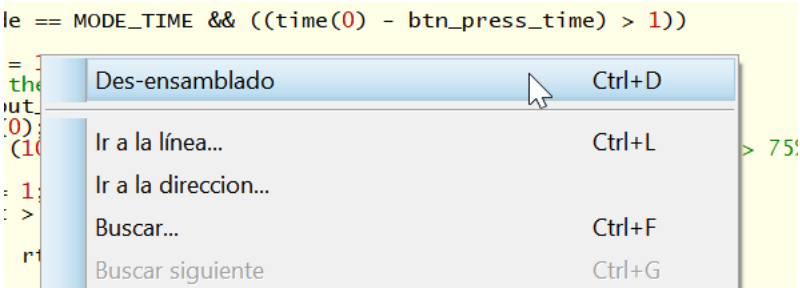
Si deseamos examinar el comportamiento de nuestro programa con el máximo detalle, podemos ejecutar el programa haciéndolo a saltos donde cada paso se ejecuta una línea de código a nivel ensamblador. Para ello detendremos la ejecución de la simulación con el botón de “PAUSE” y pulsaremos con el botón derecho del ratón sobre la ventana del código fuente. En el menú contextual que nos aparece seleccionaremos la opción ‘Des-ensamblado’ y podremos utilizar los mismos comandos que acabamos de estudiar.



Podemos comprobar que durante la depuración mediante la ejecución a saltos no se obtendrá una salida legible en la pantalla LCD. Este comportamiento es correcto y es consecuencia de que la pantalla está siendo utilizada mediante multiplexado, el cual no se lleva a cabo correctamente al ejecutar el código paso a paso.



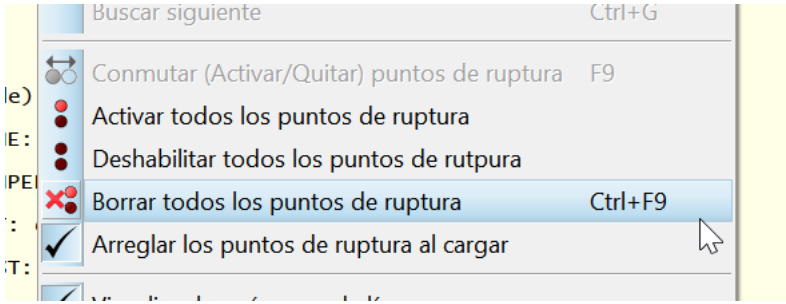
En este punto, podemos volver a activar el menú contextual con el botón derecho del ratón y volver a seleccionar la opción ‘Des-ensamblado’ para volver al código fuente escrito en lenguaje C.



Para volver a detener el programa en el lugar donde habíamos establecido nuestro punto de ruptura, sólo tenemos que actuar y soltar sobre el pulsador S1 en la ventana activa emergente.

Si pulsamos con el botón derecho sobre la ventana de código y seleccionamos ‘Borrar todos los puntos de ruptura’ en el menú contextual que nos aparece, liberaremos todos los puntos de ruptura

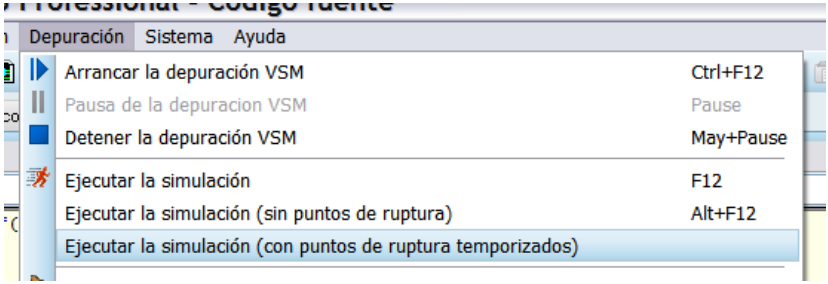
que hayamos creado y podremos seguir ejecutando la simulación con normalidad.



Al terminar la simulación podemos utilizar el botón 'STOP' de los controles de simulación para finalizar nuestro trabajo.

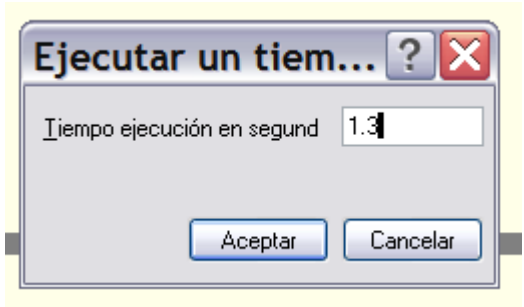
También es muy interesante la posibilidad que nos permite utilizar PROTEUS VSM de ejecutar la simulación durante un espacio de tiempo determinado y poder seguir a continuación ejecutando la simulación paso a paso o con cualquier otro de los métodos que ya hemos visto hasta ahora.

Para simular la ejecución de nuestro programa durante un tiempo determinado utilizaremos la opción 'Ejecutar la simulación (con puntos de ruptura temporizados)' del menú 'Depuración'.

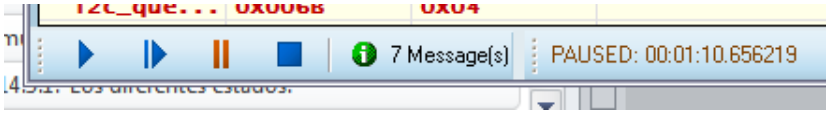


En la ventana de diálogo que aparece introduciremos el tiempo que deseemos simular expresado en segundos, disponiendo de la posibilidad de utilizar decimales[5].

5 Debemos recordar que hay que usar el punto como separador decimal.



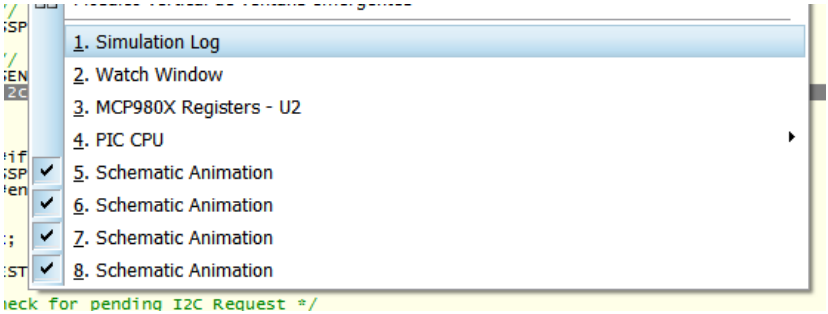
Cuando utilicemos esta opción será muy útil consultar la información del tiempo que se lleva de ejecución que se muestra en la barra de estado. Podemos repetir este comando tantas veces como deseemos.



Durante el proceso de depuración hay algunos puntos importantes que tenemos que tener en cuenta.

- Cuando situamos un punto de ruptura o realizamos la simulación paso a paso en PROTEUS es importante comprender que tenemos el sistema entero bajo el control temporal. Esto significa, por ejemplo, que los condensadores no se descargarán o que los motores no pierden impulso cuando el sistema está en pausa. Cuando ejecutemos un salto en la simulación, las instrucciones son ejecutadas y los efectos de esas líneas de programa se propagan a través de todo el sistema refrescándose el estado de todos los componentes hasta volver a detenerse la simulación de nuevo. Podremos comprobar que el tiempo ha avanzado consecuentemente en la barra de estado.
- PROTEUS pone a nuestra disposición otras muchas ventanas de depuración diferentes que no son objeto de estudio en este tutorial. Todas ellas pueden ser activadas desde el menú 'Depuración' siempre que la simulación esté en pausa. Es posible cerrar cada una de las ventanas de depuración en

cualquier momento y volver a activarlas de nuevo utilizando los comandos del menú ‘Depuración’.

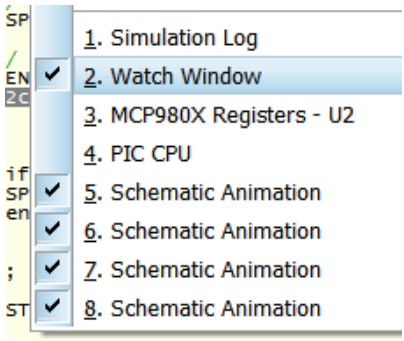


8.-La ventana de inspección.

La ventana de inspección (watch window) es la única que permanece visible mientras la simulación se está ejecutando y nos facilita otra forma de trabajar diferente a la de los puntos de ruptura. Para introducirnos en el uso de esta técnica vamos a utilizar la ventana de inspección para monitorizar el funcionamiento del convertidor analógico digital durante la lectura de los valores del potenciómetro.

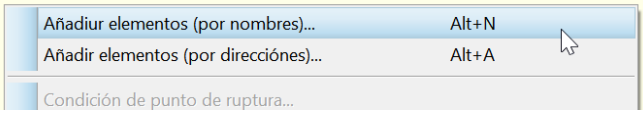
Pondremos en marcha la simulación utilizando el botón PLAY de los controles de simulación.

A continuación abriremos la ventana de inspección utilizando la opción '2. Watch Window' del menú 'Depuración'.



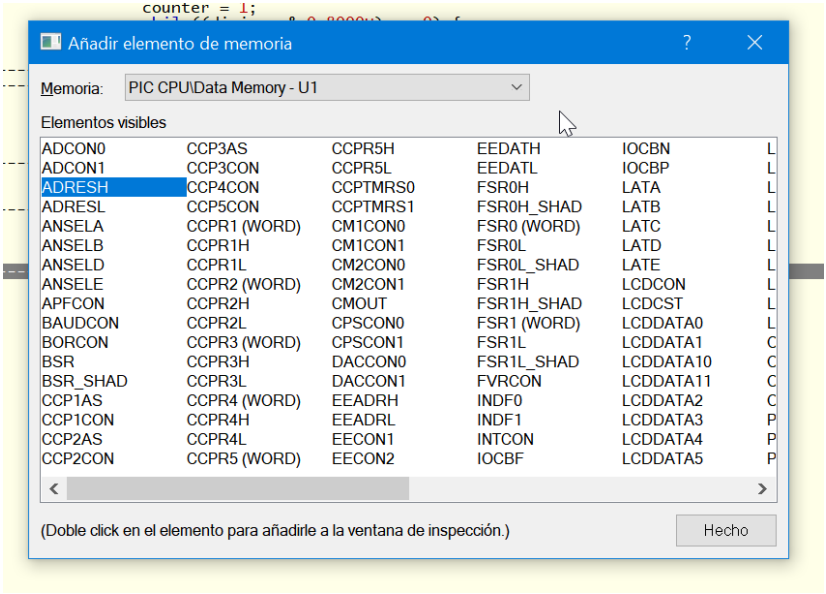
Situaremos el cursor del ratón sobre la ventana de inspección y pulsaremos el botón derecho del ratón.

Seleccionaremos la opción "Añadir elementos (por nombres)…" del menú contextual que nos aparece o el atajo de teclado 'ALT-N'.

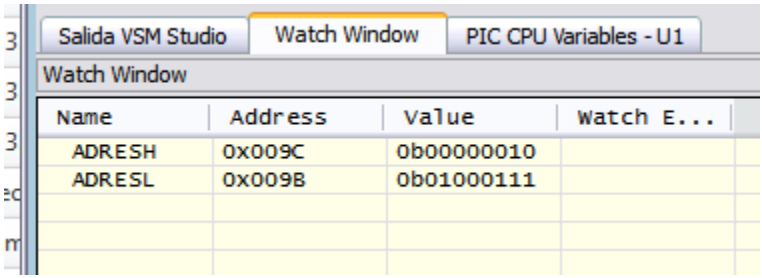


Puesto que queremos añadir a la ventana de inspección los registros donde se almacenan los resultados del convertidor analógico digital, seleccionaremos los registros ADRESH y ADRESL.

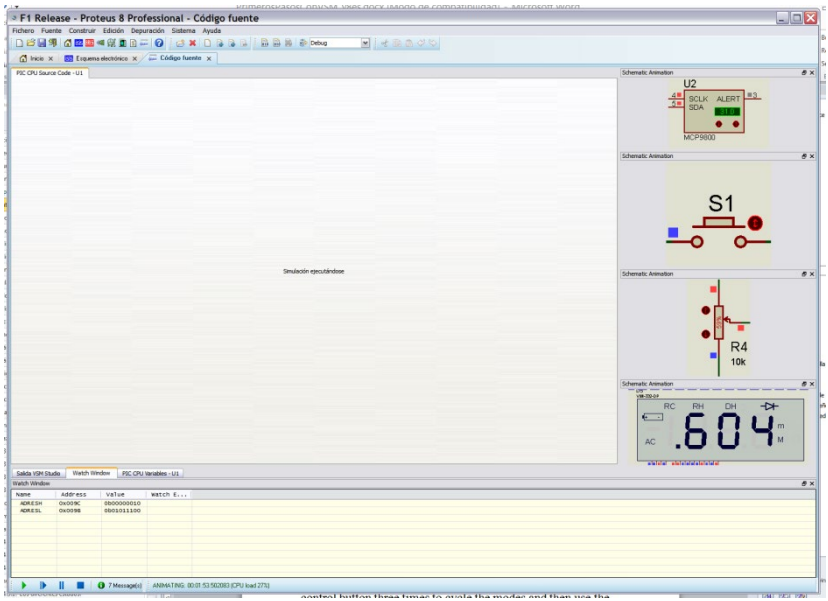
Para añadirlos a la lista de variables de la ventana de inspección sólo tenemos que hacer una doble pulsación con el botón izquierdo del ratón sobre sus respectivos nombres.



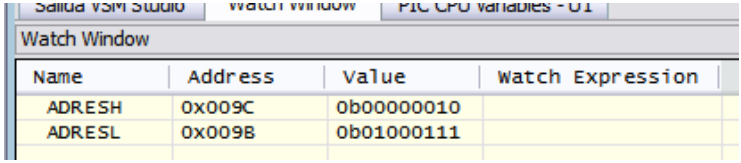
Al terminar de seleccionar los registros que nos interesan cerraremos la ventana de diálogo pulsando sobre el botón “Hecho”. En la ventana de inspección se visualizaran dos líneas con los valores de los registros seleccionados.



Ahora necesitamos ayudarnos de las ventanas activas emergentes (o ir a la pestaña ‘esquema electronico’) para utilizar el pulsador S1 para ir al modo de lectura del valor del potenciómetro R4. Actuaremos tres veces sobre el pulsador y luego utilizaremos los actuadores situados a la izquierda del potenciómetro para modificar su valor.

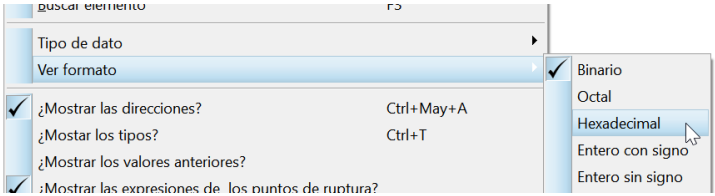


Ya tenemos que ser capaces de ver el cambio que se experimenta en los valores presentados en la ventana de inspección cuando modificamos el ajuste del valor del potenciómetro.



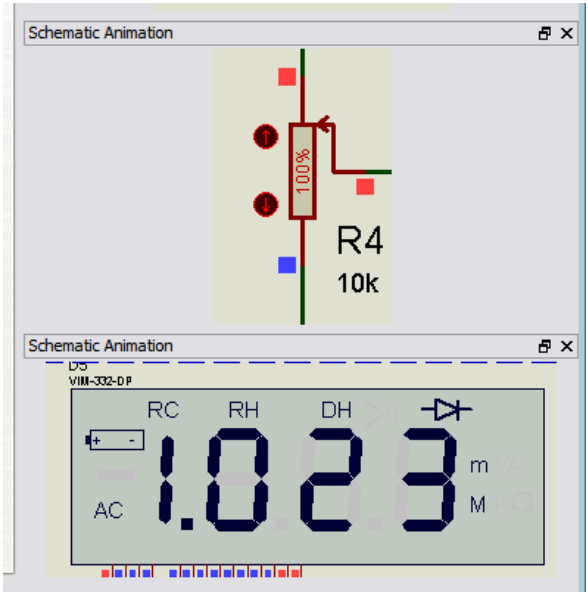
Posiblemente nos resulte más sencillo de entender los valores mostrados si cambiamos el formato de presentación. Coloquemos el cursor del ratón sobre cada uno de los dos datos visualizados en la ventana de inspección y pulsemos el botón derecho del ratón. En el menú contextual que aparece seleccionaremos la opción ‘Ver

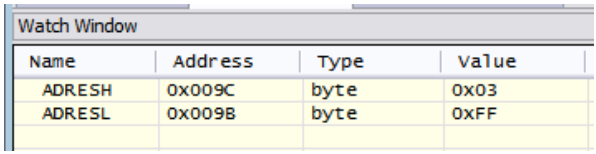
formato’ y en el nuevo menú que se despliega, la opción ‘hexadecimal’.



Puesto que estamos utilizando un convertidor analógico digital de 10bits tenemos que obtener el mayor valor posible, 1023 en decimal o 0x03FF en hexadecimal, cuando hayamos llevado el potenciómetro a su extremo superior.

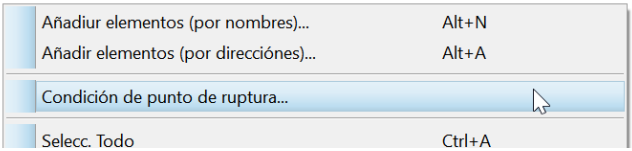
Llevemos el potenciómetro R4 a su valor más alto y observemos que en el display se muestra el valor 1023 y que en la ventana de inspección, el byte alto se muestra en el registro ADRESH y el byte bajo en el registro ADRESL. El valor total se obtendrá combinado ambos registros.





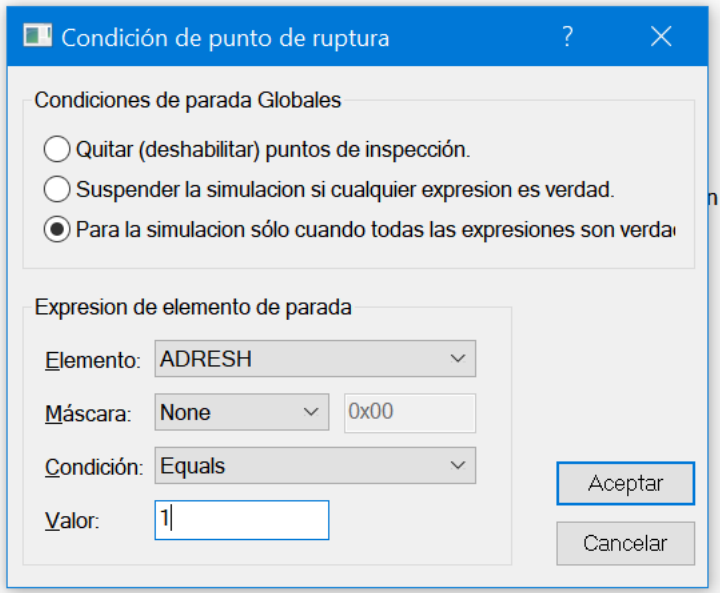
Supongamos ahora que necesitamos capturar el punto medio en el recorrido del potenciómetro. No es difícil calcular que el punto medio cae en el valor de 0x01FF, así que podemos fijar un conjunto de condiciones sobre los elementos que estamos inspeccionando para fijar un punto de ruptura cuando se alcance este valor. Para llevarlo a cabo, pulsaremos con el botón derecho del ratón cuando el cursor esté sobre uno de los elementos visualizados en la ventana de inspección para abrir el menú contextual.

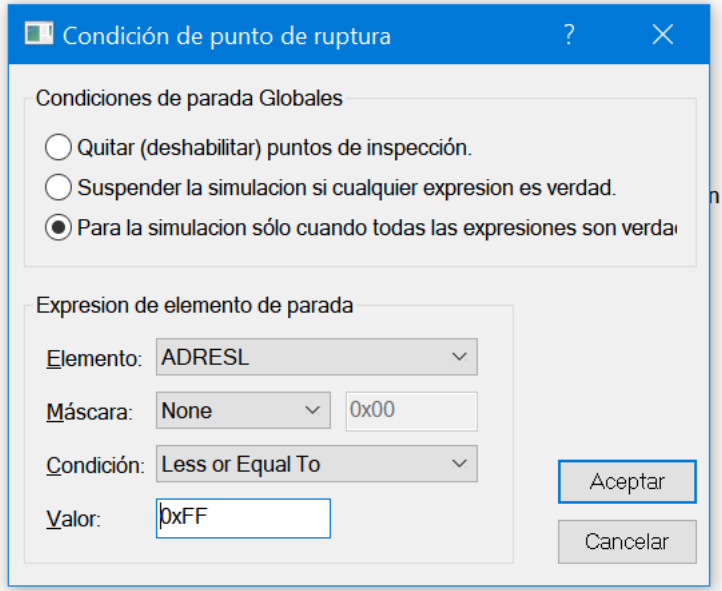
En él seleccionaremos el elemento de menú 'Condición de punto de ruptura…' para que se abra la ventana de diálogo correspondiente.



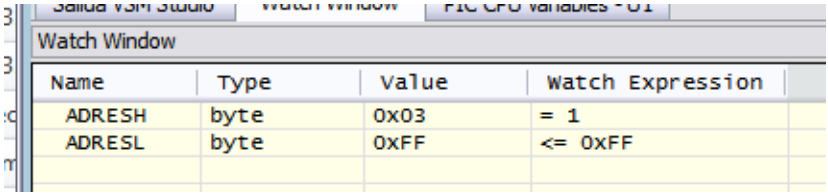
En nuestro caso queremos detener la simulación cuando el registro ADRESH valga 1 y el registro ADRESL sea menor o igual que FF. Con ello lograremos que funcione bien cuando, como en nuestra simulación, estando en el valor más alto vayamos descendiendo poco a poco el valor del potenciómetro hasta el punto intermedio.

Tenemos que realizar la operación en dos pasos, para asignar una condición a cada uno de los dos registros.

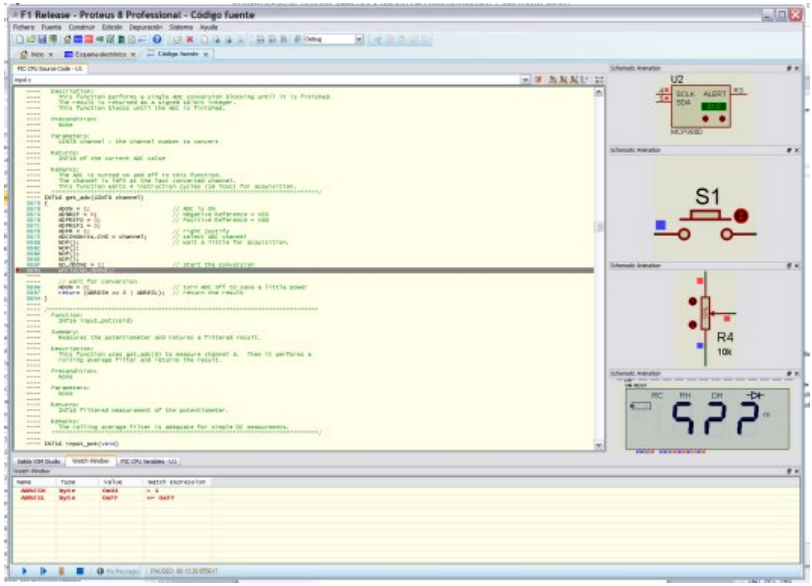

Condición de punto de ruptura
Condiciones de parada Globales
Quitar (deshabilitar) puntos de inspección.
Suspender la simulacion si cualquier expresion es verdad.
Para la simulacion sólo cuando todas las expresiones son verda
Expresion de elemento de parada
Elemento: ADRESH
Máscara: None 0x00
Condición: Equals
Valor: 1
Aceptar
Cancelar

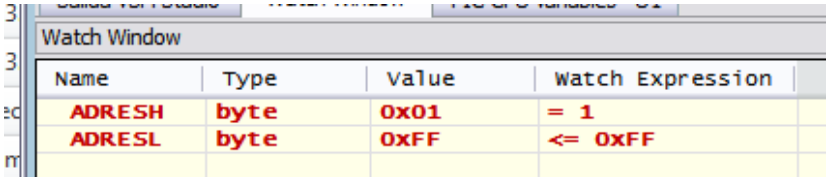

Condición de punto de ruptura
Condiciones de parada Globales
Quitar (deshabilitar) puntos de inspección.
Suspender la simulacion si cualquier expresion es verdad.
Para la simulacion sólo cuando todas las expresiones son verda
Expresion de elemento de parada
Elemento: ADRESL
Máscara: None 0x00
Condición: Less or Equal To
Valor: 0xFF
Aceptar
Cancelar

El resultado final de la ventana de inspección debe ser como el que se muestra en la siguiente imagen.



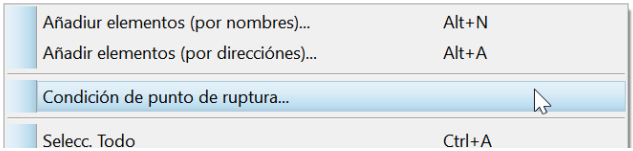
Una vez configuradas las condiciones en la ventana de inspección todo lo que nos resta por hacer es ajustar el valor del potenciómetro con los controles que se encuentran a su lado. Cuando descendamos por debajo de la mitad de su recorrido se producirá el punto de ruptura al cumplirse las condiciones fijadas por nosotros. Cuando se produzca podremos, observar que los valores leídos en la ventana de inspección es 0x1FF.



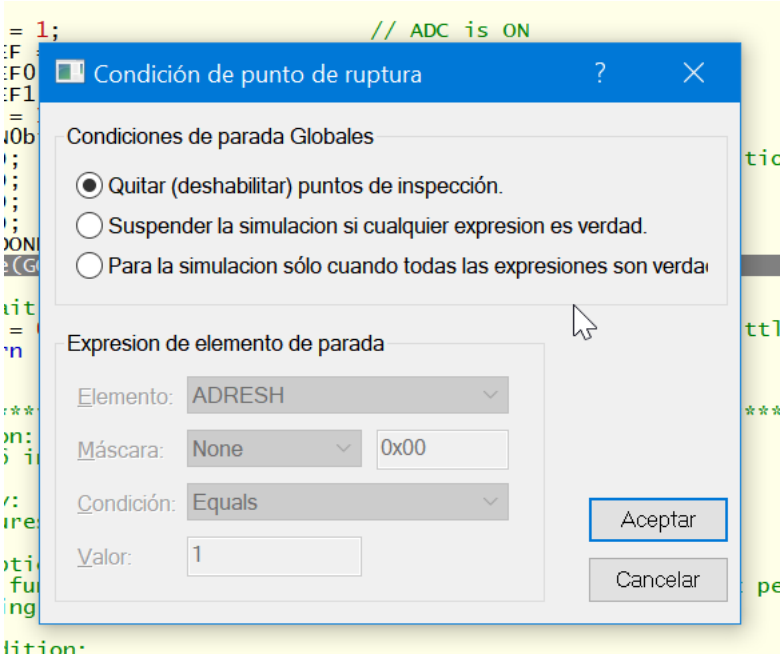


Si lo necesitamos, tenemos disponibles las opciones para seguir ejecutando nuestro programa paso a paso a partir del momento en que se produjo la ruptura por haberse cumplido las condiciones establecidas, de la misma forma que vimos en el epígrafe anterior de este mismo tutorial.

También podemos desactivar el punto de inspección que acabamos de crear pulsando con el botón derecho sobre el elemento de la ventana de inspección para abrir el menú contextual y seleccionar la opción ‘Condición de punto de ruptura’ de nuevo.

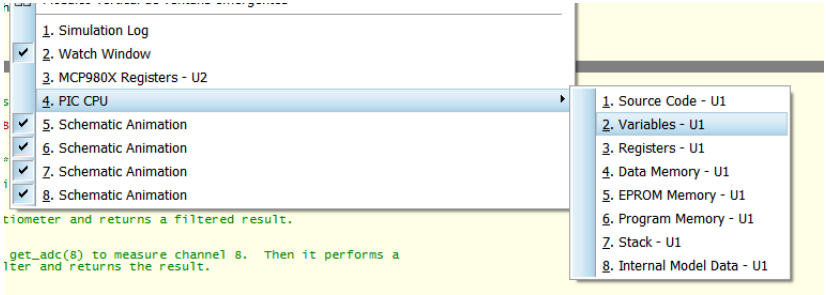


En la ventana de diálogo que se abre deberemos seleccionar la opción ‘Quitar (deshabilitar) puntos de inspección’.

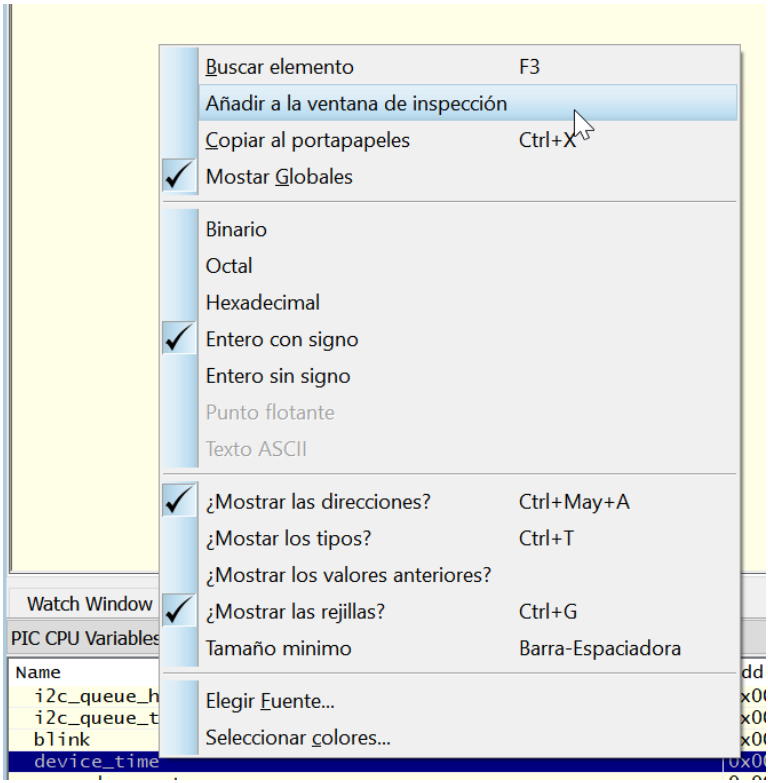


Igual que fuimos capaces de añadir registros por su nombre en la ventana de inspección, PROTEUS nos permite hacer lo mismo pero realizando la selección directamente desde la ventana de variables o desde la ventana de memoria.

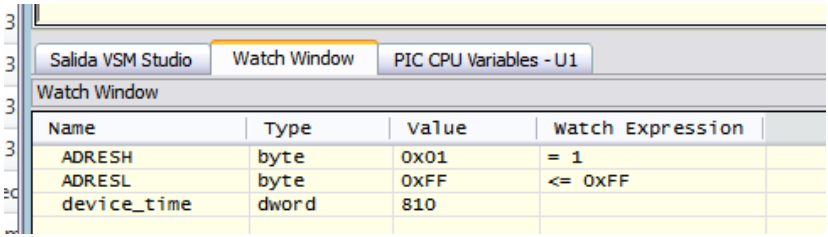
Podemos probar ahora a abrir la ventana de variables desde el menú 'Depuración' (en el caso de que no la tuviéramos ya abierta).



Ahora podemos colocar el cursor sobre una de las variables, pulsar el botón derecho y en el menú contextual seleccionar la opción 'Añadir a la ventana de inspección'.



Y de esta sencilla forma, ya disponemos de esta variable en la ventana de inspección para consultar su valor en cualquier momento de la ejecución de nuestro programa.



Ahora, para continuar avanzando en este tutorial, debemos desactivar las condiciones de punto de inspección como describimos más arriba (o bien podemos borrar completamente los elementos desde la ventana de inspección) y pulsar sobre el botón de STOP desde los controles de la simulación para finalizar.

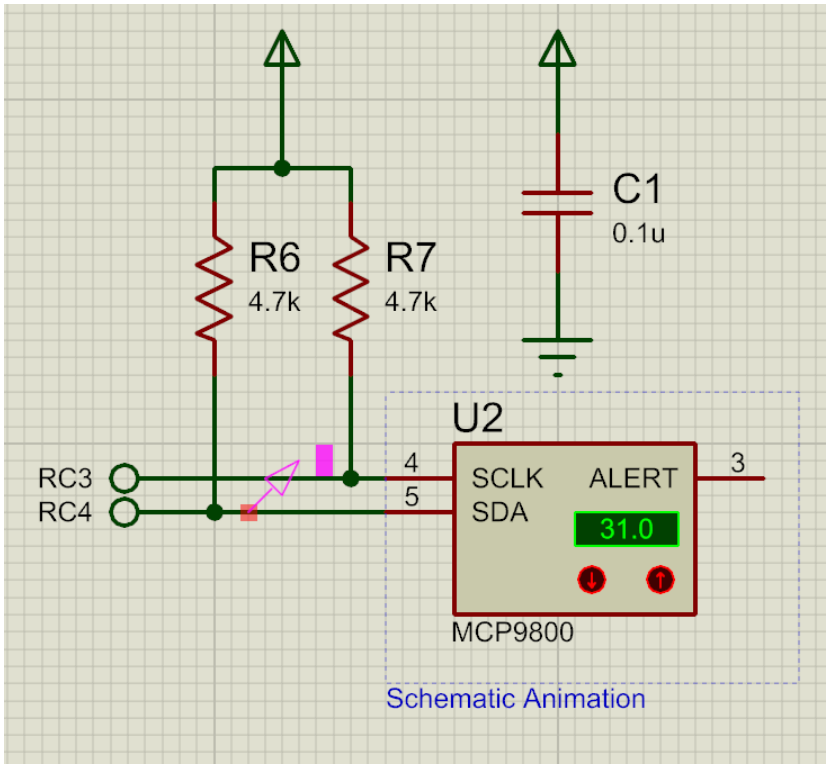
9.-Utilización de los puntos de ruptura por hardware.

Hasta ahora, hemos visto como podíamos interrumpir la ejecución de la simulación fijando puntos de ruptura basados en condiciones que se producían durante la ejecución del software. PROTEUS VSM también nos permite configurar puntos de ruptura basados en el hardware con objeto de comprobar en un momento dado la situación del código cuando ocurre una condición determinada en alguno de los dispositivos que componen nuestro diseño electrónico.

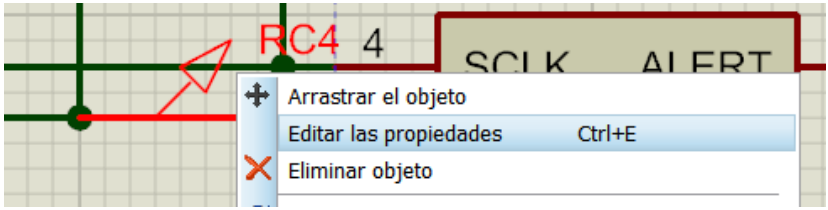
Vamos a introducirnos en el uso de esta nueva técnica, siguiendo la filosofía habitual de este tutorial, con un ejemplo sacado del trabajo cotidiano. Supongamos que deseamos detener la simulación en el momento en el que comienza la actividad en el bus i2c. Para hacerlo, tendremos que configurar un punto de ruptura basado en hardware llevando a cabo los siguientes pasos.

Activemos la pestaña 'Esquema electrónico'. A continuación usaremos el icono 'modo sonda' situado en la barra de herramientas lateral.

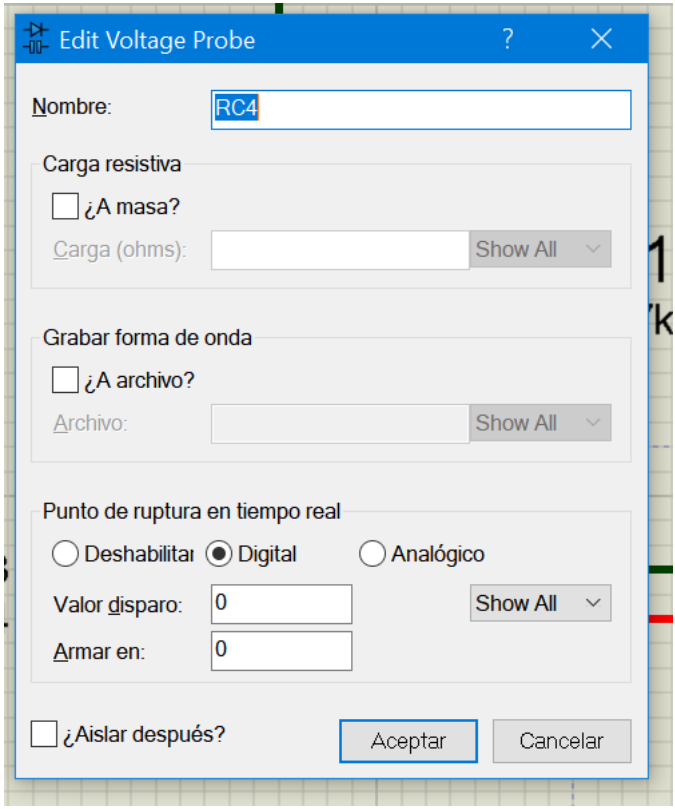
Pondremos el cursor en el área de trabajo y pulsaremos con el botón izquierdo del ratón para comenzar su posicionamiento dentro de nuestro esquema electrónico. Moveremos el cursor del ratón hasta situarlo encima de la línea SDA del bus i2c y pulsaremos de nuevo el botón izquierdo del ratón para situar la sonda sobre el cable en cuestión.



Pondremos el cursor del ratón sobre la sonda que acabamos de instalar (si es necesario utilizaremos el zoom para facilitar esta tarea) y pulsaremos el botón derecho. Seleccionaremos la opción 'Editar las propiedades' en el menú contextual que aparece.

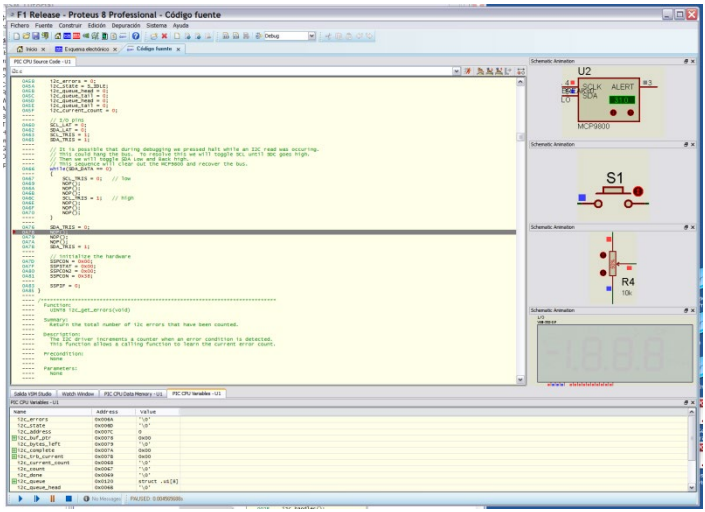


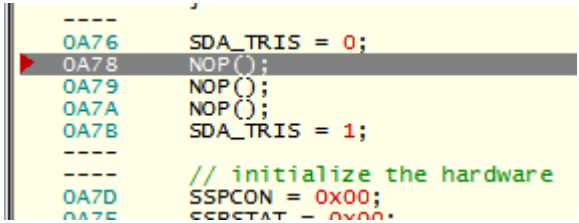
En la ventana de diálogo elegiremos un punto de ruptura digital cuando el valor sea 0.



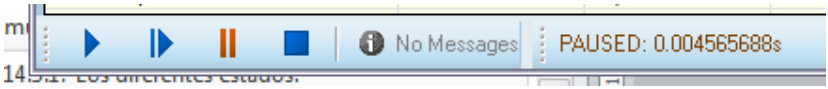
Pulsaremos sobre el botón ‘Aceptar’ para cerrar la ventana de diálogo, utilizaremos el atajo de teclado “F8” para hacer un zoom que contenga todo nuestro diseño electrónico y pondremos en marcha la simulación utilizando el botón PLAY desde los controles de la simulación.

Si todo ha ido bien, la simulación se detendrá nada más empezar y nos mostrará la ventana con el código fuente de nuestro proyecto justo en la línea inmediatamente posterior a aquella donde ejecutamos el comando que fija el nivel del bus SDA al valor 0.



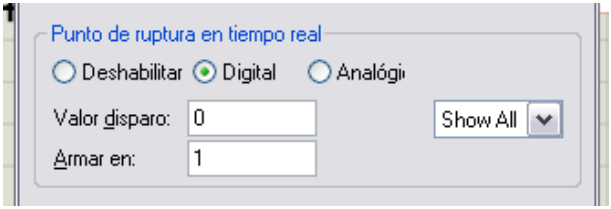


En la barra de estado nos tiene que aparecer un mensaje informándonos de la situación en que se encuentra la simulación.



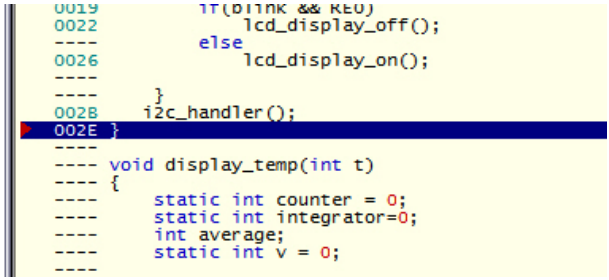
Este punto de ruptura ha tenido lugar dentro de la rutina de inicialización. Si deseamos saltarnos este punto y sólo activar el punto de ruptura cuando ya nos hayamos situado dentro de la ejecución ordinaria del bus i2c podemos seleccionar un tiempo de armado (parámetro ‘armar en’ en la sección ‘Punto de ruptura en tiempo real’) para el punto de ruptura. Para ello detendremos la simulación utilizando la tecla ‘STOP’ de los controles de la

simulación. A continuación editaremos la sonda de voltaje que habíamos colocado antes y fijaremos un tiempo de armado de 1 segundo. Como siempre, nos podemos ayudar del zoom para llevar a cabo este trabajo.



Volveremos a ejecutar la simulación con la tecla ‘PLAY’. Esta vez la simulación no se detendrá hasta que entremos en el modo de lectura de temperatura haciendo uso el pulsador S1 como vimos con anterioridad en otros epígrafes de este tutorial.

Cómo es lógico, la ejecución de la simulación se detendrá en otro punto diferente de nuestro programa (dentro de las rutinas que gestionan el bus i2c).





Como siempre, llegados a este punto, podemos continuar con la ejecución del programa utilizando los diferentes comandos de depuración y salto paso a paso que describimos en secciones anteriores de este tutorial.

Detendremos la simulación de nuevo cuando hayamos acabado, utilizando el botón ‘STOP’ de los controles de simulación.

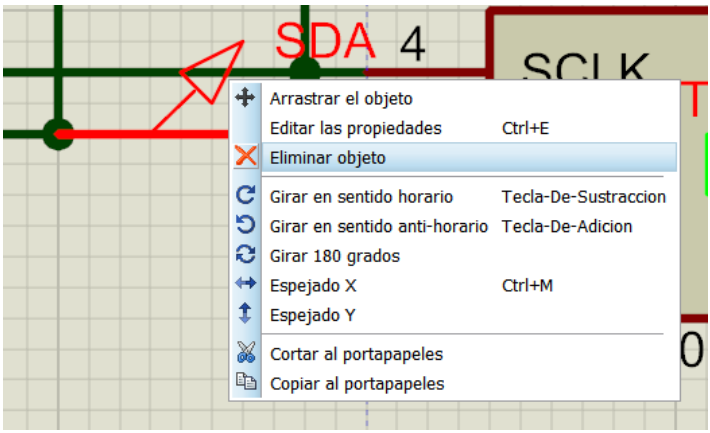
10.-Obtención de medidas durante la simulación.

Una de las taras más habituales durante la comprobación del funcionamiento de un diseño electrónico es examinar los valores de las diferentes señales eléctricas. PROTEUS nos permite realizar esta tarea siguiendo dos caminos diferentes. Como siempre en este tutorial vamos a llevar a cabo un ejemplo para conocer los pormenores de estas dos técnicas que nos brinda PROTEUS.

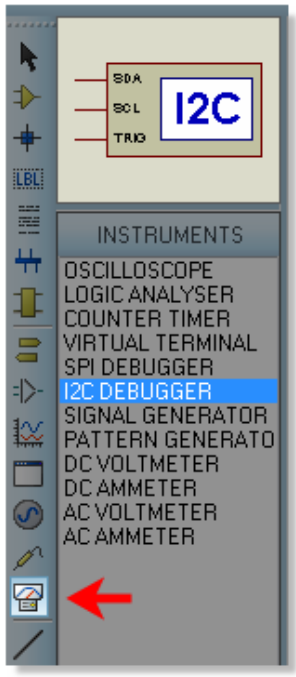
Supongamos que deseamos examinar el tráfico que se genera sobre el bus i2c relacionado con el sensor de temperatura MCP9800. Podemos llevarlo a cabo de forma interactiva o utilizando los métodos gráficos tradicionales.

10.1.-Obtención de medidas utilizando herramientas virtuales.

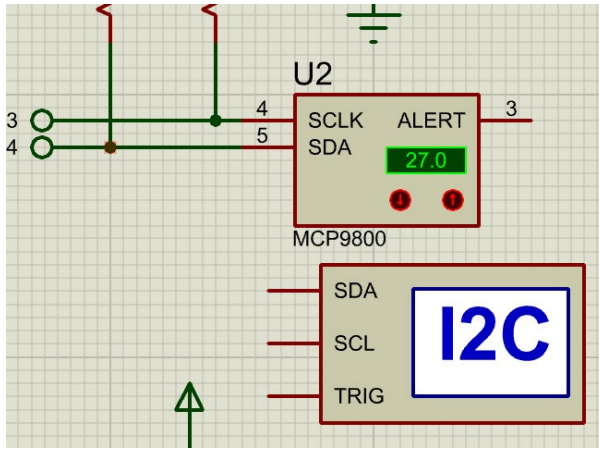
Empezaremos eliminando la sonda de voltaje que habíamos colocado en la línea SDA en el epígrafe anterior de este tutorial. Para ello situaremos el cursor del ratón sobre ella, pulsaremos el botón derecho y seleccionaremos la opción 'Eliminar objeto' del menú contextual que aparece.



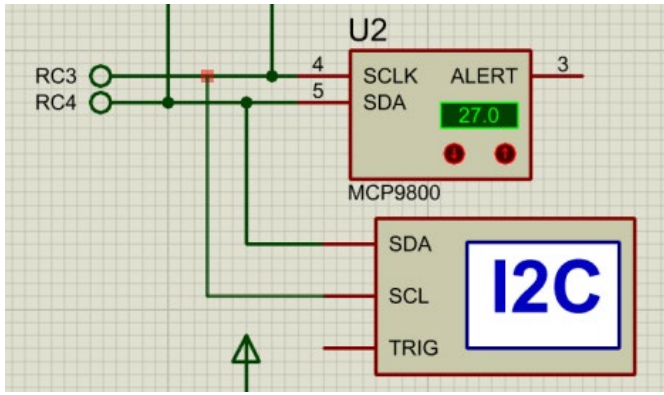
A continuación seleccionaremos el icono ‘Instrumentos’ desde la barra de herramientas lateral. PROTEUS nos ofrece, en la ventana de selección de objetos, una lista de todos los instrumentos de medida virtuales disponibles.



En nuestro caso, la herramienta más apropiada es el analizador de protocolo i2c (i2c debugger). Podemos seleccionarlo y colocarlo en nuestro diseño electrónico junto al sensor de temperatura.

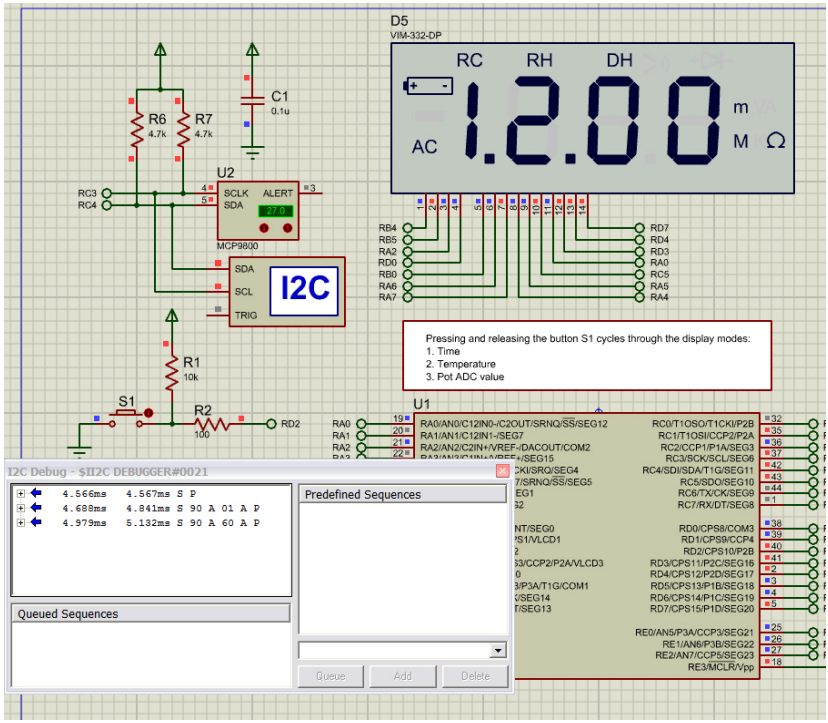


El siguiente paso será cablear los pines de nuestro equipo de medida. El proceso para cablear nuestro equipo de medida es similar al de cualquier otro dispositivo en PROTEUS ISIS. Podemos notar que el cursor se vuelve de color verde al arrancar y finalizar la ruta desde los pines al cableado. Si fuera necesario puede repasar estos conceptos en el tutorial de la pestaña Esquema electrónico.



Conectaremos el pin SDA del analizador i2c al pin SDA del sensor de temperatura y el pin SCL del analizador i2C al pin SLCK del sensor de temperatura.

Una vez que hayamos cableado correctamente nuestro analizador de protocolo i2c podemos arrancar la simulación con la tecla PLAY.

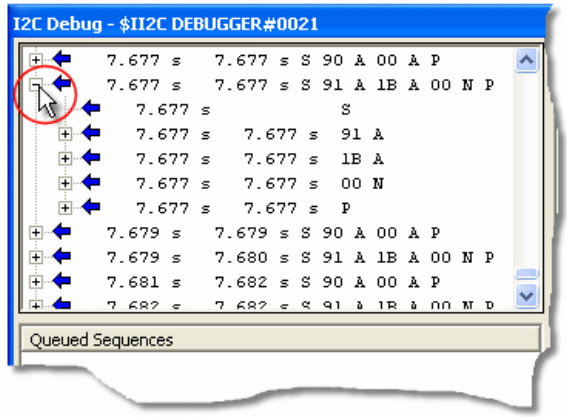


Esta vez debemos ver la nueva ventana adicional para la herramienta que acabamos de utilizar. En la zona izquierda de la nueva ventana seremos capaces de inspeccionar el tráfico inicial generado sobre el bus con los procedimientos de inicialización e identificación de dispositivos. Si deseamos ver el proceso de generación de tráfico sobre el bus, tendremos que actuar sobre el pulsador S1 para cambiar el modo de funcionamiento de nuestro programa.

Al hacerlo se han puesto en marcha las comunicaciones entre el sensor de temperatura y el microprocesador a través del bus i2c. Puesto que el código está continuamente generando lecturas de valores desde el sensor de temperatura apenas nos dará tiempo para ver el flujo de datos que se está generando.

Podemos pulsa sobre la tecla “PAUSE” del control de simulación para detener la simulación y poder analizar el tráfico en un momento dado con más tranquilidad.

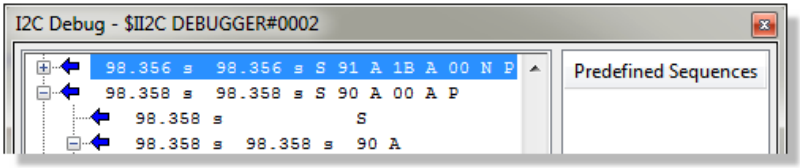




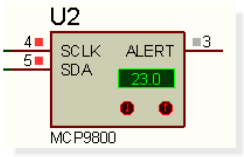
Si estamos trabajando con una sola ventana, la pestaña ‘código fuente’ se activará cuando la simulación se detenga en pausa porque se asume que estamos interesados en consultar el código fuente de nuestro programa. El analizador i2c seguirá en primer plano. Si deseamos ver al mismo tiempo el esquema electrónico y el código fuente, podemos volver a utilizar la técnica de ventanas activas emergentes ya vista con anterioridad en este tutorial.

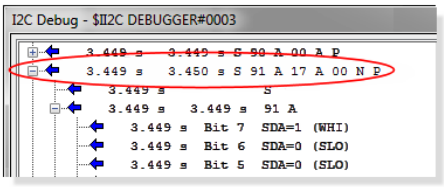
Para aquellos que conozcan el protocolo de comunicaciones i2c la sintaxis utilizada por la herramienta les tiene que resultar familiar. Podemos expandir cualquiera de las líneas utilizando el botón con un signo “+” situado a la izquierda de cada una de ellas. El dispositivo MCP9800 tiene la dirección asignada 0x90 así que podemos ver las solicitudes de lectura que le son dirigidas.

En la imagen siguiente se ha desplegado la respuesta devuelta por el dispositivo MCP9800. La secuencia arranca con un comando START (S), seguido del valor 0x91 (read request), seguido de 0x1B (el valor leído en hexadecimal que se corresponde con los 27 grados que marcaba nuestro dispositivo cuando hicimos la captura).



Podemos experimentar con nuestro analizador de protocolo i2c cambiando la temperatura en el dispositivo MCP9800 y observando los resultados obtenidos en el bus de comunicaciones. Para ello volveremos a poner en marcha la simulación (botón PAUSE) cambiaremos el valor de temperatura con los controles del dispositivo, volveremos a detener la simulación (botón PAUSE) y volveremos a examinar la ventana del analizador de protocolo i2c.





Es importante que tengamos en cuenta que el almacenamiento continuado de datos que se produce en nuestro caso provoca una considerable sobrecarga de trabajo y una lógica disminución del rendimiento general de la herramienta, puesto que se está generando la escritura de forma generalizada de datos en la ventana. Sin embargo, esta circunstancia no devalúa el valor de esta herramienta puesto que la utilización de la instrumentación virtual y las herramientas de depuración se hace en un contexto donde la velocidad de la simulación es secundaría respecto al fin principal que es la resolución de problemas en el software y el hardware.

Podemos encontrar información más detallada de los diversos instrumentos virtuales de medida disponibles en PROTEUS dentro de la ayuda en línea. Por ejemplo, el analizador de protocolo i2c que acabamos de ver, puede ser utilizado como un simple inspector del tráfico que se produce dentro de la red de comunicaciones o como un maestro de bus desde el que podemos generar comandos para comprobar el buen funcionamiento de los diversos dispositivos enlazados al bus.

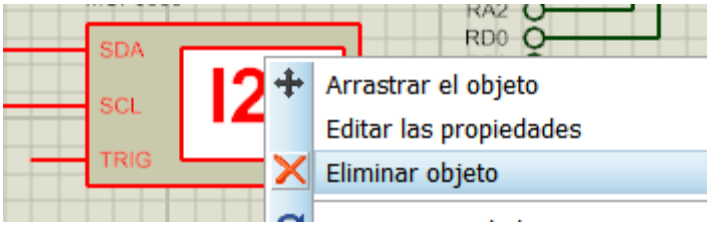
10.2.-Obtención de medidas utilizando gráficos.

La otra técnica que PROTEUS pone a nuestra disposición para realizar medidas se basa en la utilización de gráficos. Existen importantes diferencias entre uno y otro método que afectan de forma sustancial a la forma en que se desarrollará nuestra simulación:

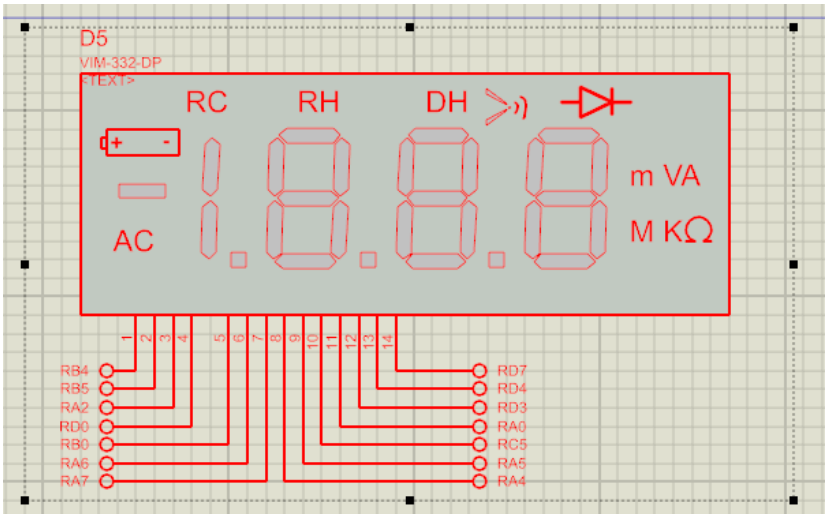
- Cuando utilizamos los gráficos no podemos interactuar al mismo tiempo sobre el circuito.
- Un gráfico ejecuta la simulación durante un periodo de tiempo concreto y perfectamente definido.
- Los resultados no son visualizados en forma de gráfico hasta que el tiempo fijado transcurre completamente y la simulación se ha detenido.

Teniendo en cuenta todos estos condicionantes vamos a ver un ejemplo práctico de la utilización de los gráficos para inspeccionar el tráfico en el bus de comunicaciones i2c.

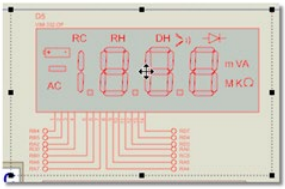
Antes de nada, borraremos el dispositivo virtual de medida que habíamos creado. Para ello colocaremos sobre él el cursor del ratón y pulsaremos el botón derecho. En el menú contextual que aparece, seleccionaremos la opción 'Eliminar objeto'.

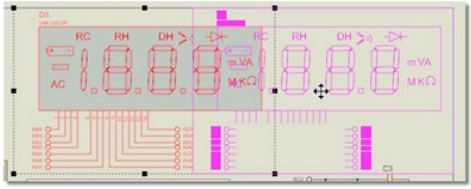


Ahora dibujaremos una caja alrededor de la pantalla LCD y todos los cables que están unidos a ella con el botón derecho del ratón con lo que lograremos que quede todo resaltado y seleccionado.



Una vez que lo hayamos hecho pulsaremos el botón izquierdo y desplazaremos todo el conjunto hacia la derecha del área de trabajo.



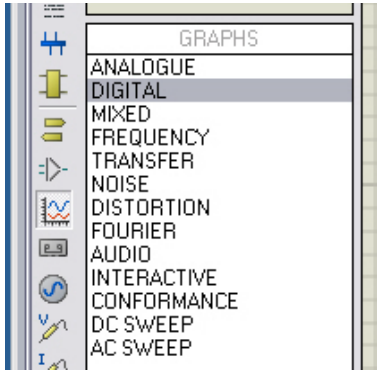


A continuación seleccionaremos el icono ‘Modo gráfico’ de la barra de herramientas lateral.

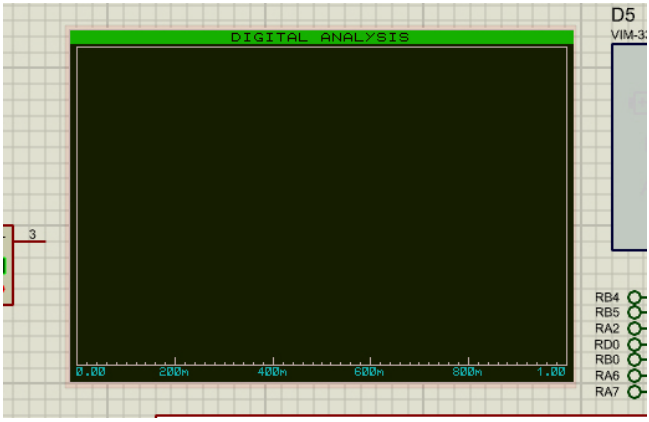


En la ventana de selección de objetos, elegiremos el tipo de gráfico de tipo ‘digital’. Desplazaremos el cursor del ratón a la zona donde deseemos colocar la esquina superior izquierda del gráfico y pulsaremos el botón izquierdo. Moveremos el cursor del ratón hasta donde deseamos colocar la esquina inferior derecha del gráfico y

volveremos a pulsar el botón izquierdo para terminar la colocación del gráfico.

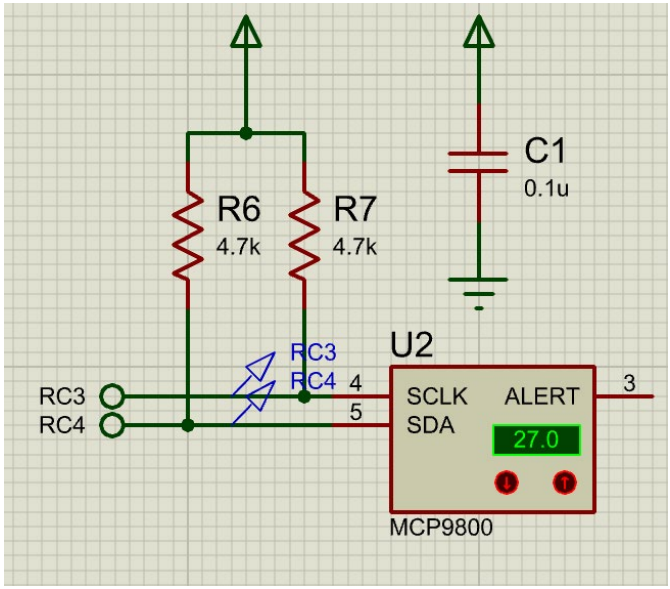




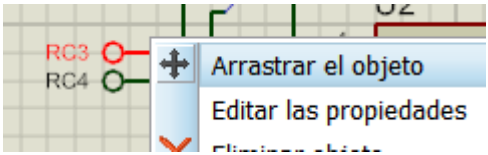


No necesitamos que el gráfico sea muy grande. Podremos más tarde maximizar la pantalla cuando deseemos analizar los resultados obtenidos para visualizarlos con comodidad.

Una vez que hemos situado el gráfico en nuestra área de trabajo, ahora necesitamos añadir las sondas necesarias y cablearlas en los lugares donde nos interesa capturar los datos. Seleccionaremos el icono ‘Modo sonda’ de la barra de herramientas lateral y situaremos dos sondas, una en cada uno de los dos hilos del bus i2c. Lo llevaremos a cabo de la misma forma que hicimos con anterioridad en el apartado dedicado a los puntos de ruptura por hardware.



Si tenemos dificultades para colocar las sondas porque el espacio es pequeño, podemos colocar el cursor sobre cada uno de los dos terminales (RC3 y RC4) y pulsar el botón derecho. En el menú contextual seleccionamos la opción ‘arrastar el objeto’ y lo desplazamos hasta donde nos resulte más cómodo.

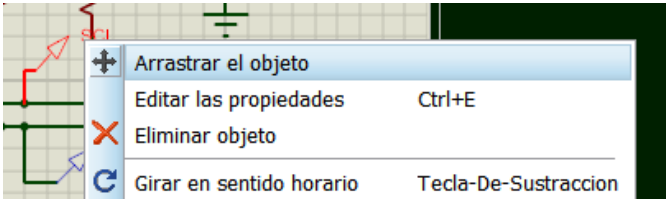


Como siempre podemos hacer uso de la rueda del ratón y los atajos de teclado “F6” y “F7” para facilitar nuestra tarea con el zoom. El

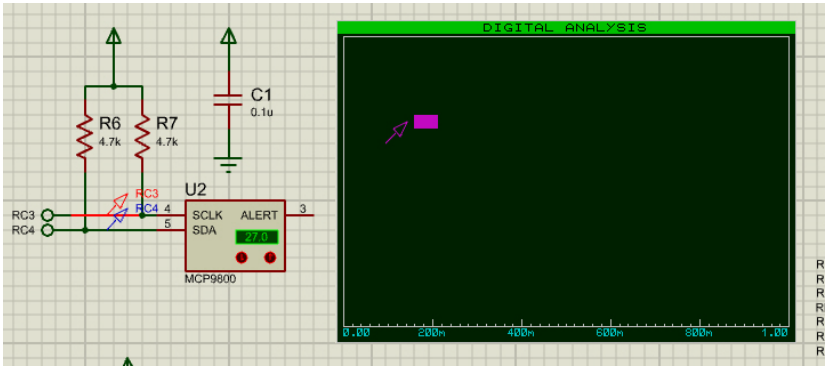
atajo de teclado “F8” nos permitirá volver a realizar un zoom que abarque a todo nuestro diseño.

Por defecto, al colocar las sondas sobre el cable, estas toman el nombre del terminal al que dicho cable está conectado. Y al colocarlas sobre el gráfico mantienen el mismo nombre. Puede resultar útil renombrarlas para hacerlas más fácil de identificar en el contexto de nuestra simulación. Podemos llevarlo a cabo colocando el cursor sobre ella, pulsando el botón derecho, editando sus propiedades y asignándoles el nombre de SCL y SDA respectivamente.

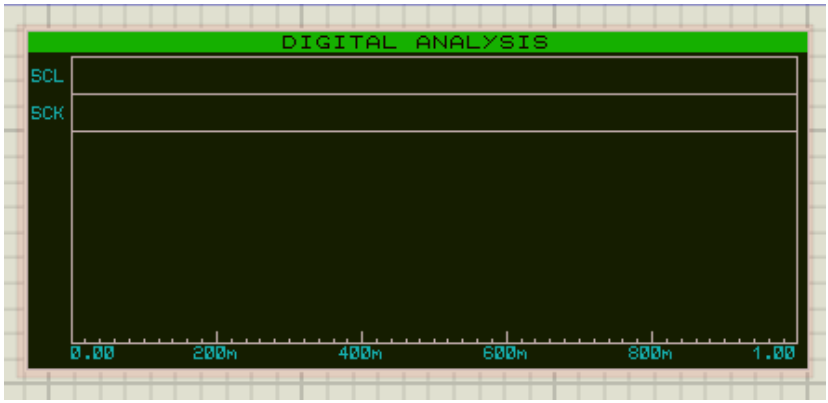
Una vez situadas las dos sondas colocaremos el cursor del ratón sobre una de ellas, pulsaremos el botón derecho y en el menú contextual que aparece elegiremos la opción “Arrastrar el objeto”.



A continuación desplazaremos el cursor del ratón sobre el gráfico y pulsaremos el botón izquierdo para soltar la sonda sobre él.

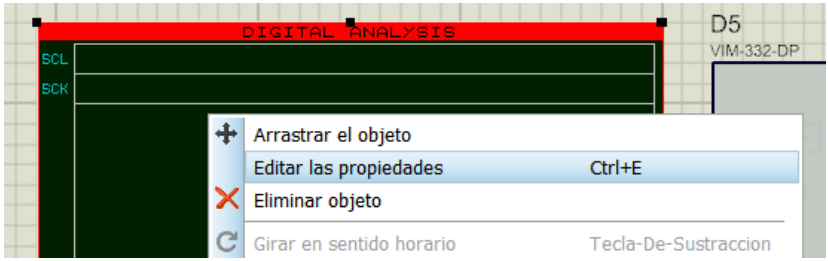


Podemos repetir el proceso con la otra sonda.

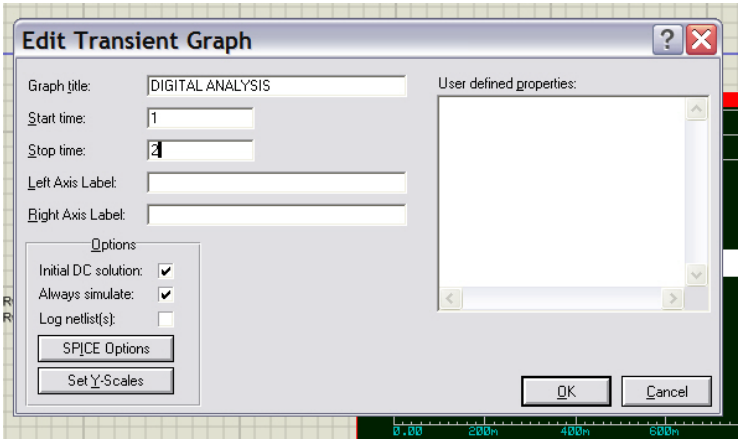


Si no indicamos nada en contra, la simulación basada en un gráfico se desarrollará desde el momento cero hasta que transcurra un segundo. En nuestro caso sería más interesante realizar la simulación, para evitar el tráfico propio de la fase de inicialización, desde el segundo 1 hasta el segundo 2.

Podemos hacerlo editando las propiedades del gráfico colocando el cursor del ratón sobre él, pulsando el botón derecho y seleccionando la opción ‘Editar las propiedades’ desde el menú contextual que aparece.



En la ventana de diálogo mostrada modificaremos los valores de los campos “start time” y “stop time”. Terminaremos pulsando sobre el botón “ok”.

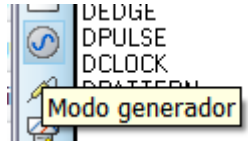


A continuación pulsaremos con el botón izquierdo sobre cualquier superficie vacía del área de trabajo para asegurarnos que ningún objeto queda seleccionado. Esto es una operación habitual después de editar las propiedades de cualquier objeto y salir de la ventana de diálogo correspondiente. En caso contrario este objeto quedaría seleccionado.

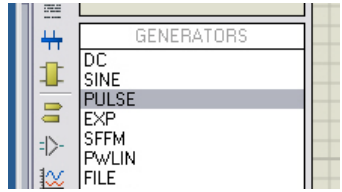
Ahora estaríamos listos para generar la simulación excepto por un importante detalle. Como mencionamos con anterioridad no podemos interactuar con nuestro circuito electrónico durante la simulación basada en gráficos. Sin embargo nosotros necesitamos cambiar el modo de funcionamiento de nuestro programa para captar datos desde el sensor de temperatura y arrancar el tráfico de datos por el bus i2c.

Puesto que no podemos utilizar el pulsador S1 necesitamos generar una señal similar a la que generaría este pulsador en el pin correspondiente del microprocesador PIC para que el software que corre en él, salte al modo de lectura de temperaturas.

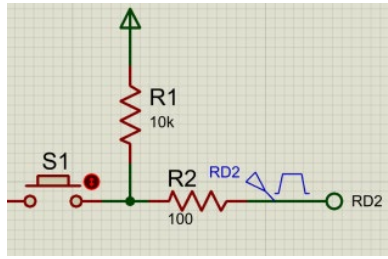
Podemos llevar a cabo esta tarea utilizando los generadores que pone a nuestra disposición PROTEUS VSM. Seleccionemos el icono ‘Modo generador’ en la barra de herramienta lateral.



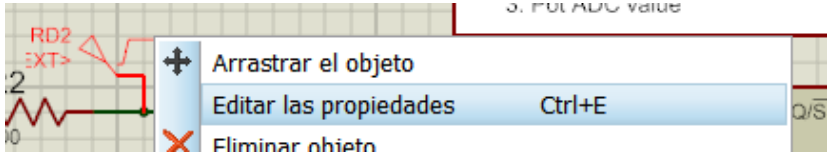
Elegiremos el generador de tipo ‘pulse’ en la ventana de selección de objetos.



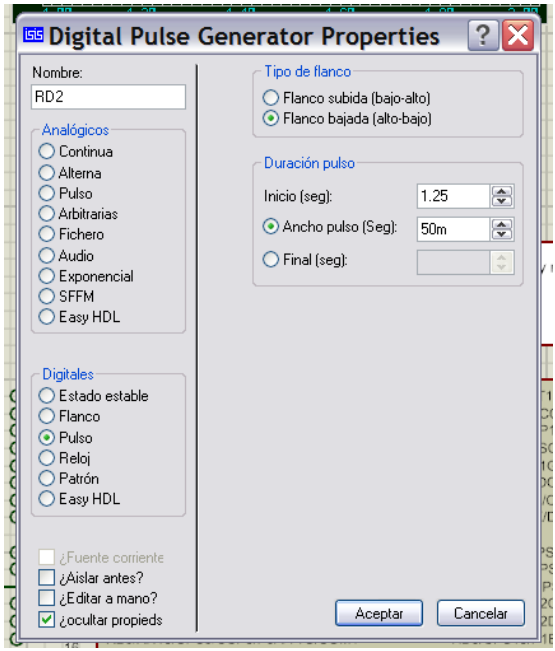
Colocaremos el generador a la derecha de la resistencia R2 y antes del terminal RD2. Puesto que hay muy poco espacio entre ambos, es posible que sea necesario desplazar primero el terminal RD2 hacia la derecha para hacer hueco y poder colocar el generador.



A continuación configuraremos el generador siguiendo el procedimiento habitual. Colocaremos el cursor del ratón sobre él, pulsaremos el botón derecho y seleccionaremos la opción ‘Edtiar las propiedades’ desde el menú contextual.

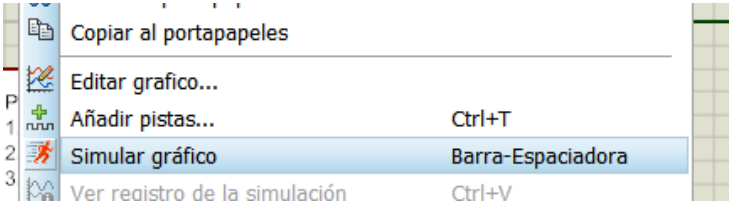


En la ventana de diálogo que se abre, seleccionaremos los siguientes parámetros:

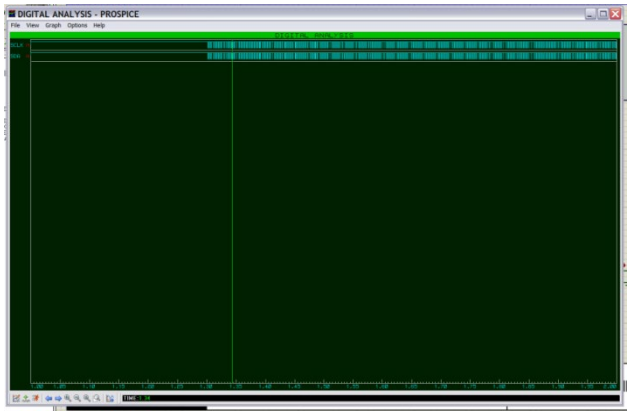


Ahora ya estamos en condiciones de ejecutar la simulación. A diferencia de lo que ocurría con la simulación mediante instrumentos virtuales, no necesitamos utilizar los controles de simulación y queremos realizar la simulación desde el segundo 1 hasta el segundo 2.

Así que ahora utilizaremos el comando 'Simular gráfico' disponible desde el menú contextual del gráfico. También podemos utilizar el atajo de teclado "barra espaciadora".

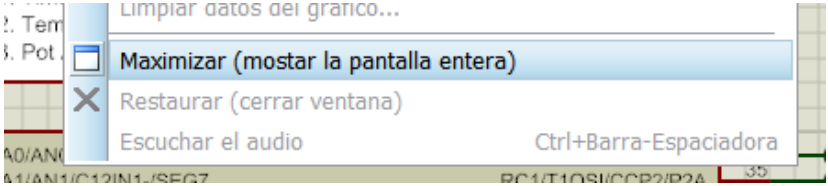


Al ejecutar la simulación tenemos que obtener el siguiente grafico.

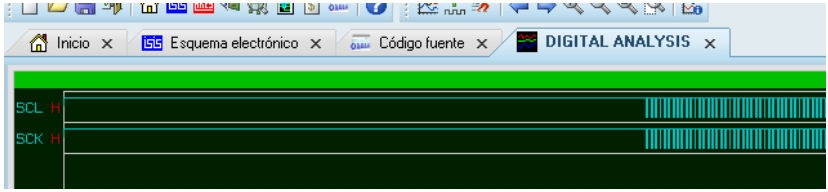


A parte de comprobar que la generación de pulsos no se produce hasta un poco antes del instante 1’’25 (el tiempo que especificamos para comenzar la generación del pulso) no es posible sacar muchas más conclusiones de esta vista.

Es posible maximizar la ventana del gráfico utilizando el comando ‘Maximizar’ del menú contextual o haciendo una doble pulsación con el botón izquierdo del ratón sobre el título del gráfico.



Al hacerlo el gráfico se mostrará en una nueva pestaña.



Afortunadamente PROTEUS VSM nos ofrece muchas posibilidades para analizar los datos recogidos en el gráfico. Los controles para

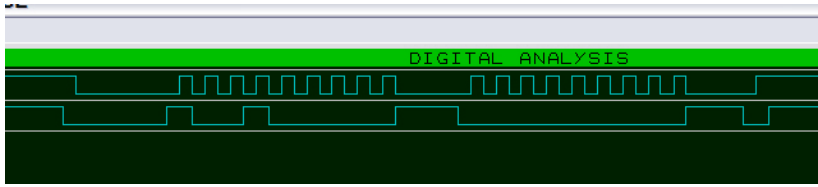
manejar el zoom se encuentran en la barra de herramientas de la zona superior.



El más práctico es el icono para realizar el zoom a un área específica de nuestro gráfico. Seleccionaremos esta herramienta y dibujaremos un rectángulo alrededor de una pequeña zona del gráfico, justo donde deseamos hacer el zoom.



Podemos repetir el mismo procedimiento tantas veces como lo deseemos para ir centrando cada vez más la imagen en un punto concreto dentro de nuestro gráfico. Por ejemplo, en la siguiente imagen vemos ampliada la zona correspondiente a un primer bloque de datos.

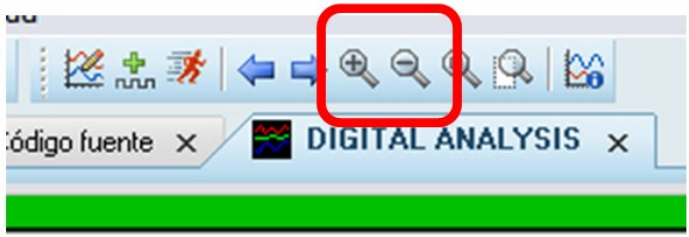


También podemos hacer la misma función, manteniendo pulsada la tecla para seleccionar mayúsculas del lado izquierdo del teclado mientras con el ratón trazamos una caja alrededor de la zona donde queremos hacer el zoom.

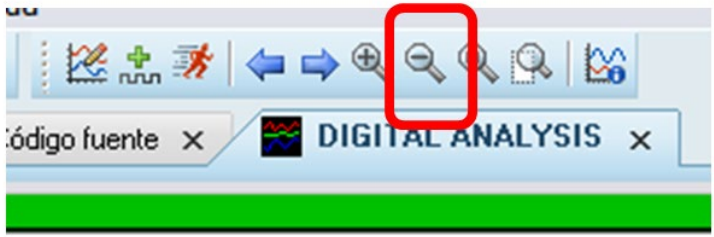
Aquellos que estén familiarizados con el protocolo de comunicaciones i2c y la utilización del osciloscopio para su análisis, pueden de esta forma contrastar los paquetes de datos con los que

hemos visto con anterioridad al utilizar la herramienta del analizador de protocolo i2c.

También podemos aumentar o disminuir el zoom en cualquier momento con los dos controles habituales para hacerlo que se muestran en la siguiente imagen.



Por último, en lo que al zoom se refiere, podemos volver a dejar el gráfico como estaba al principio con el último de los botones disponibles para el zoom.



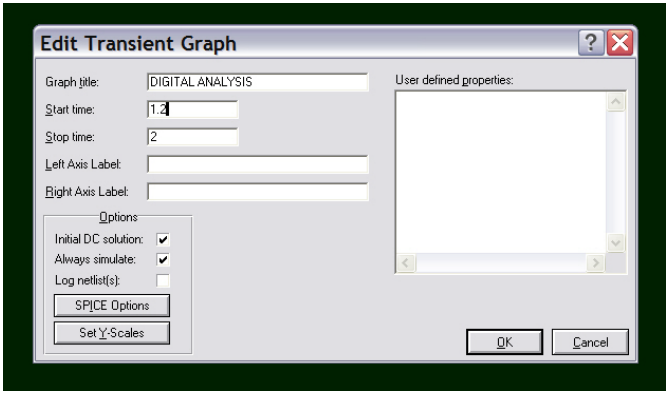
Con las flechas izquierda y derecha podemos desplazar nuestro gráfico respecto al eje temporal.



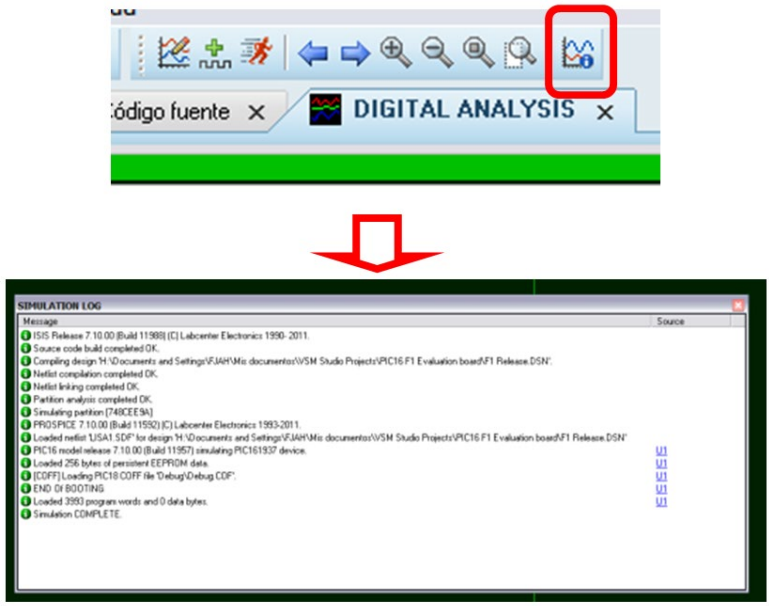
El icono situado a la izquierda nos permite modificar algunos de los valores de configuración de nuestro gráfico.



Por ejemplo, podemos modificar el tiempo inicial y el final y algunas otras opciones como se muestra en la pantalla siguiente.



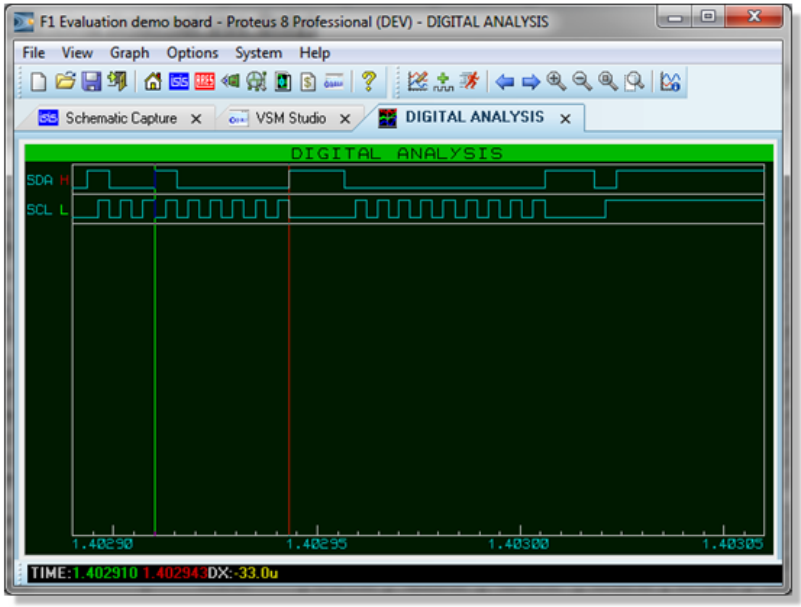
El icono ‘ver registro de la simulación’ nos permite consultar el histórico con todos los mensajes que se han generado durante la ejecución del gráfico.



También podemos utilizar el gráfico para obtener medidas a partir suyo, siguiendo el siguiente procedimiento.

- Pulsaremos el botón izquierdo del ratón para colocar una primera marca sobre el gráfico (de color verde). No soltar el ratón hasta estar seguro de que la marca se ha situado en el lugar que deseamos. Hasta que soltemos el botón podemos desplazar nuestra primera marca.
- Una vez colocada la primera marca, pulsar la tecla CTRL y mantenerla pulsada mientras desplazamos el ratón, para colocar la segunda marca (de color rojo) y pulsaremos el botón izquierdo del ratón. Si deseamos desplazar la segunda marca sólo tenemos que volver a pulsar la tecla CTRL y desplazar el ratón hasta donde deseamos moverla y una vez allí volver a pulsar el botón izquierdo del ratón.
- Una vez puestas las dos marcas, podemos variarlas de posición con el siguiente procedimiento. Cada vez que pulsemos el botón izquierdo colocaremos en ese lugar la marca verde y cada vez que pulsemos el botón izquierdo mientras mantenemos pulsada la tecla CTRL fijaremos en ese lugar la marca roja.

- En la barra de estado nos aparecerá los valores de tiempo sobre la que están situadas las dos marcas y la distancia de separación entre ambas.

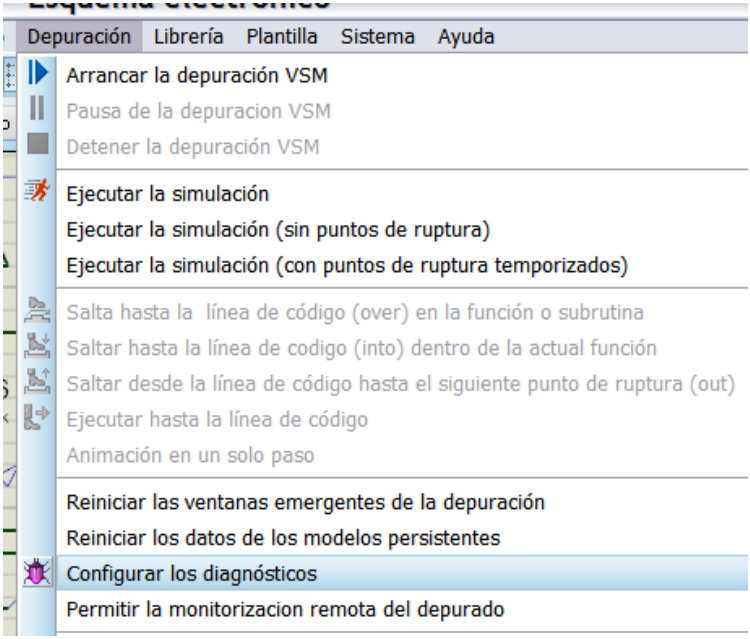


Como siempre en la ayuda en línea podemos encontrar ampliada la información referente a las sondas, los generadores y la simulación basada en gráficos para proceder a su estudio con más profundidad.

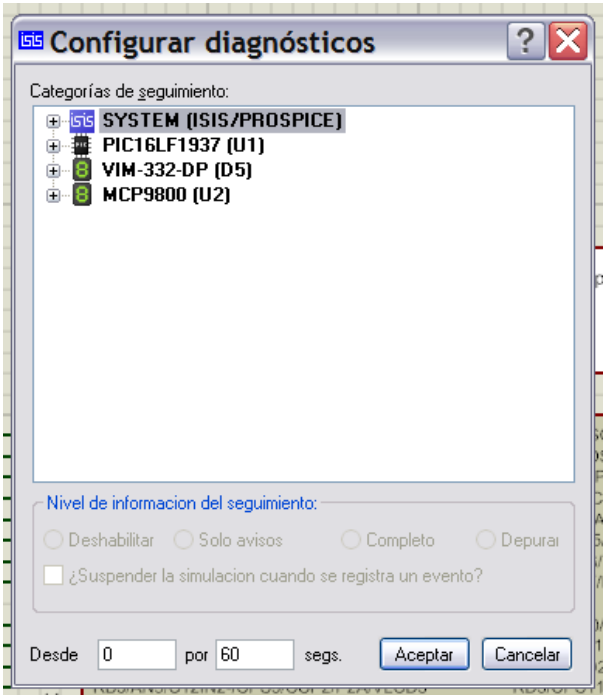
11.-Obtención de informes para el diagnóstico de incidencias.

Otra de las herramientas que pone PROTEUS VSM a nuestra disposición para ayudarnos durante la ejecución de la simulación son los mensajes de diagnóstico. Esta técnica nos permite solicitar al sistema un informe con la actividad de un particular periférico o de varios de ellos. Para introducirnos en su utilización vamos a arrancar el informe de diagnóstico desde la pantalla LCD.

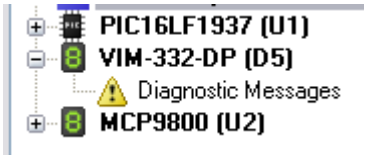
1. En el menú 'Depuración' seleccionamos la opción 'Configurar los diagnósticos'.



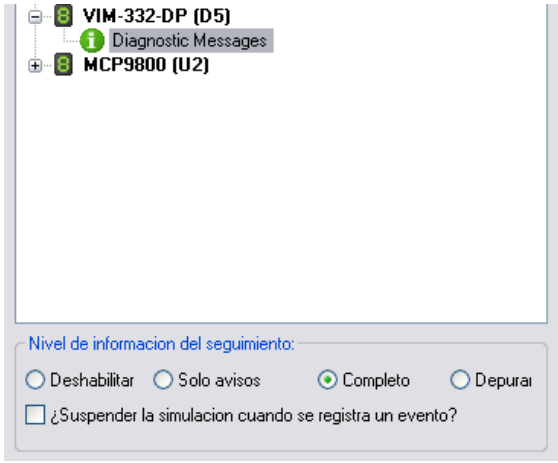
2. En la ventana de diálogo que aparece situar el cursor sobre el símbolo con una cruz que está delante de la línea 'VIM-332-DP (D5)' y pulsar con el botón izquierdo para expandirla.



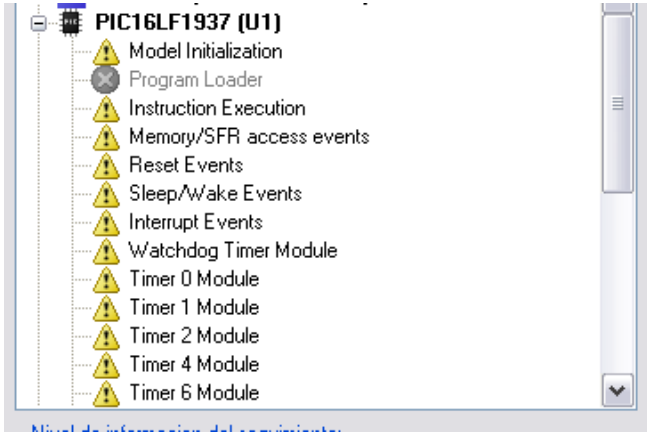
3. Pulsar con el botón izquierdo sobre el mensaje que acaba de aparecer (con un triángulo amarillo a la izquierda).



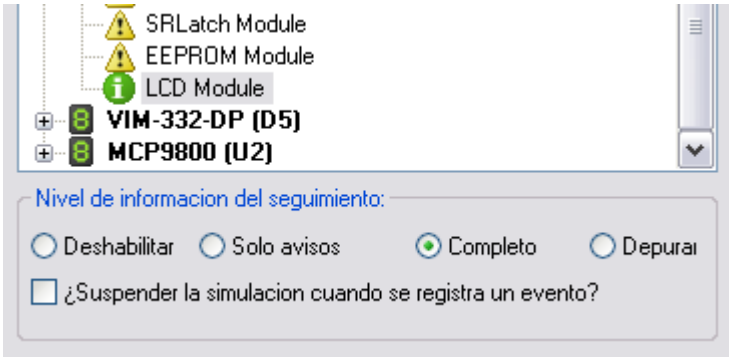
4. En la zona inferior de la ventana de diálogo, en la sección titulada ‘Nivel de información de seguimiento’, seleccionamos la opción ‘Completo’.



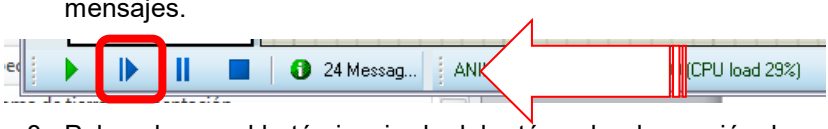
5. Expandimos también el árbol referente al PIC16LF1937 (U1) pulsando con el botón izquierdo sobre el signo “+” situado a su izquierda.



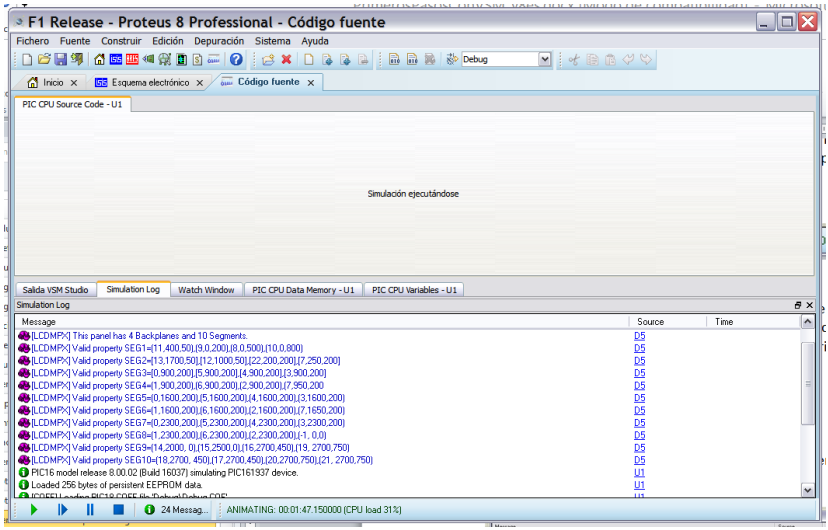
6. Nos desplazamos hacia abajo hasta encontrar la línea referente a la pantalla LCD (LCD Module) para seleccionarla y elegir también la opción ‘Completo’.



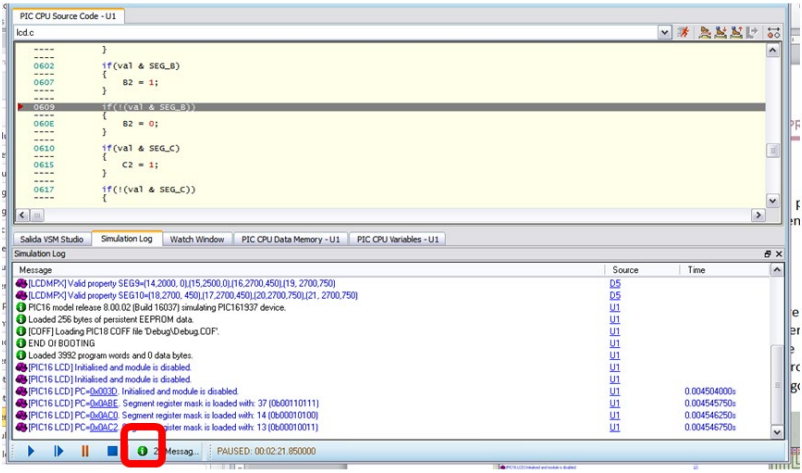
7. Cerramos la ventana de diálogo con el botón “Aceptar”. Ahora ejecutamos la simulación con la tecla PLAY desde los controles de simulación. En la barra de estado nos tiene que aparecer un aviso de que se han generado una serie de mensajes.



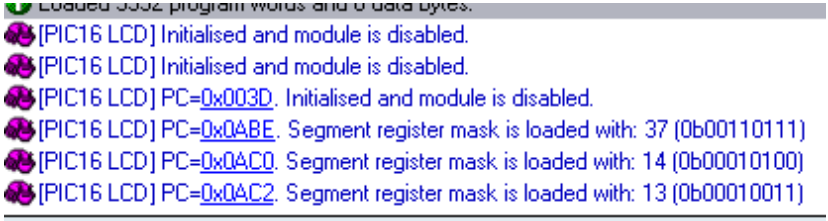
8. Pulsando con el botón izquierdo del ratón sobre la sección de la barra de estado donde nos informa de la presencia de mensajes, provocaremos la activación del panel con el histórico de mensajes de la pestaña ‘Código fuente’.



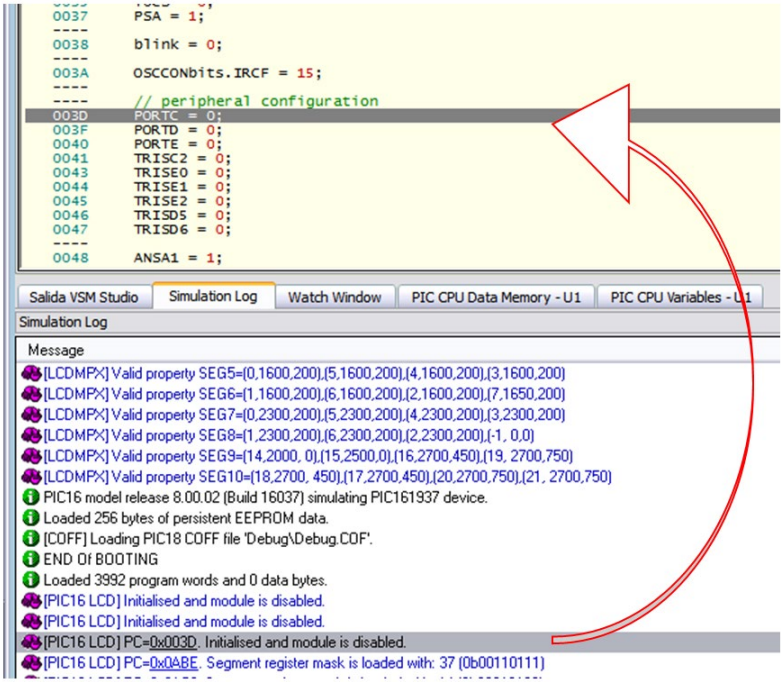
9. Es posible que tengamos que redimensionarla para poder ver todos los mensajes. Pongamos la simulación en pausa con el control correspondiente.



10. Podemos fijarnos en los mensajes generados referentes al módulo LCD. En cada una de ella aparece el valor del puntero del contador de programa como un enlace (PC=0xnnnn).



11. Pulsando sobre cada uno de estos enlaces nos llevarán directamente a la línea de programa del código que se visualiza en la ventana de código fuente en la que se produjo ese aviso.



12. A continuación, paremos la simulación completamente con el botón STOP de los controles de simulación y volvamos a dejar los mensajes de diagnóstico como estaban deseleccionándolos desde la ventana de configuración como lo hicimos en los pasos 1 a 4.

La navegación desde la ventana de mensajes a la ventana del código fuente sólo funciona cuando la simulación se encuentra en pausa.

Los mensajes de diagnósticos son una herramienta muy útil cuando nos encontramos con una situación problemática durante el proceso de simulación. Nos permiten recabar información antes de decidir donde debemos buscar el error.

Sin embargo, hay que tener en cuenta que nos penalizará el rendimiento de la simulación. Por ejemplo podemos probar a habilitarlo para el “PIC MSSP PERIPHERAL” y el “MCP9800” y seleccionar el modo de lectura de temperatura para comprobar la pérdida de rendimiento. Por lo tanto deberíamos utilizarlo sólo

cuando lo consideremos necesario para poder depurar nuestro diseño en caso de errores.

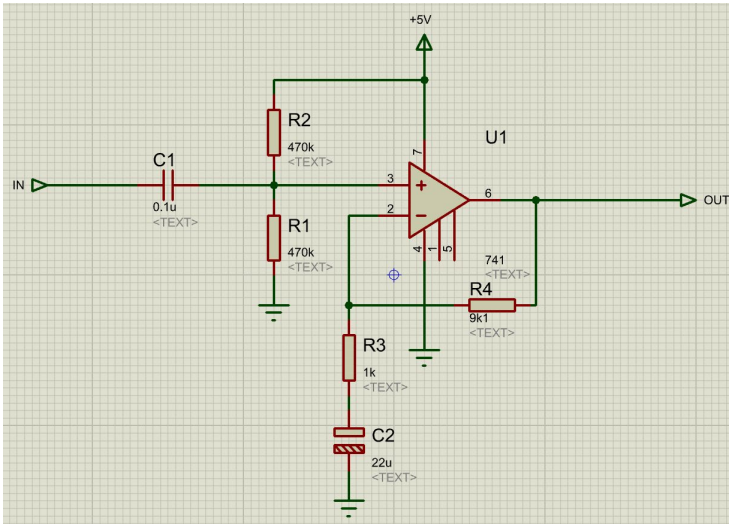
También podemos seleccionar desde la ventana de simulación, en la zona inferior, el periodo de tiempo en el que estará habilitada la generación de mensajes. De esta forma podemos concretar su uso a un momento concreto de la simulación (por ejemplo durante el primer minuto de la ejecución al arrancar).



12.-Simulación gráfica de circuitos electrónicos sin microprocesadores.

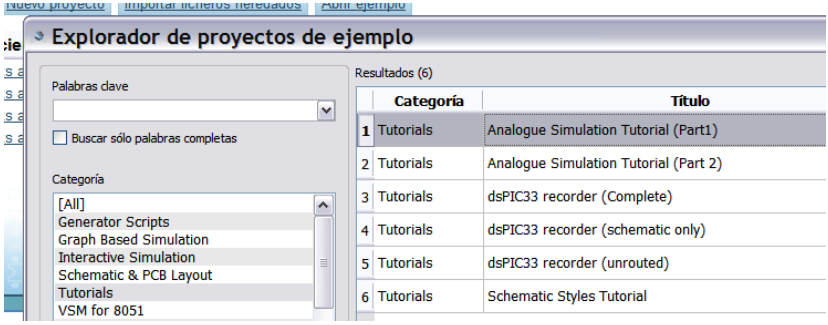
Hasta ahora nos hemos centrado en las enormes posibilidades que nos brinda PROTEUS VSM para la simulación de circuitos electrónicos donde se utilizan microprocesadores. Con las herramientas disponibles podemos observar, depurar y modificar el software que se está ejecutando en nuestro diseño. Pero PROTEUS VSM también nos brinda muchas opciones para la simulación de circuitos electrónicos en los que no están presentes los microprocesadores. De acuerdo con la filosofía de “aprender con la práctica” que gobierna todos nuestros tutoriales, vamos a realizar el aprendizaje de estas técnicas utilizando un circuito amplificador básico sobre el que iremos descubriendo las diferentes técnicas que PROTEUS VSM pone a nuestra disposición.

El circuito en cuestión es un amplificador de audio basado en el amplificador operacional 741 como se muestra en la pantalla siguiente.



La utilización de un amplificador operacional 741 de esta forma es bastante inusual puesto que lo estamos alimentando con una tensión simple de 5V. Las resistencias de retroalimentación, R3 y R4, nos proporcionan una ganancia cercana a 10. Los componentes a la entrada (R1, R2 y C1) acondicionan la señal de entrada formando una referencia a tierra falsa en la entrada no inversora con lo que se desacopla de la señal de entrada.

En este punto, podemos escoger entre introducir el esquema electrónico manualmente o aprovechar el que incluye Proteus en sus ejemplos. Para ello elegimos la opción 'Abrir ejemplo' de la sección 'Empezar a trabajar' de la pestaña 'Inicio'



Seleccionaremos la categoría 'Tutorials' y en los resultados que aparecen el fichero 'Analogue Simulation Tutorial (Part1)'.

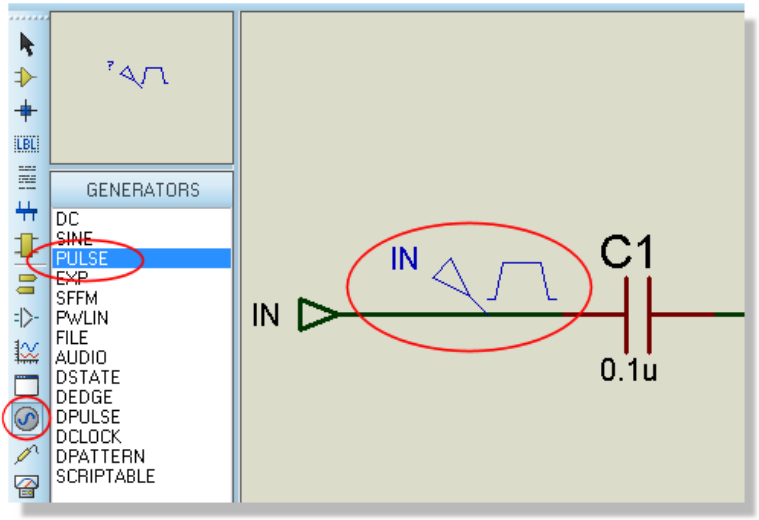
Como en el caso de la utilización de microprocesadores, en este punto también suponemos que ya tenemos suficiente dominio de la pestaña 'Esquema electrónico' por lo que no vamos a perder el tiempo en explicar cómo se dibuja nuestro esquema electrónico y partiremos de la utilización del fichero del ejemplo.

12.1.-Colocación de generadores y sondas.

Vamos a partir de una situación que se presenta en la vida real al diseñar un circuito electrónico: obtener un análisis de transitorios del circuito. Esta forma de análisis es la más usual y facilita una gran cantidad de información sobre el comportamiento del circuito. Cuando tengamos completada la descripción de la simulación utilizando análisis de transitorios, los otros tipos de análisis podrán ser fácilmente abordados.

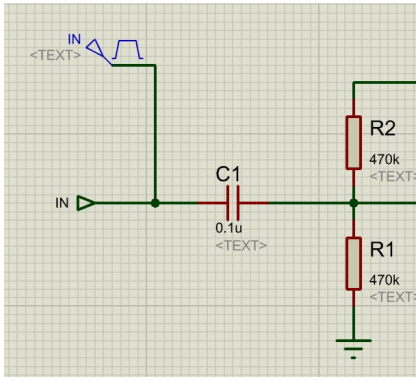
En el epígrafe 7 ya vimos la técnica de utilización de los generadores. Para comprobar el comportamiento de nuestro circuito utilizaremos una señal de entrada adaptada a nuestras necesidades. Para ello, vamos a utilizar un generador de voltaje con una salida en forma de onda cuadrada.

Seleccionaremos el icono 'Modo generador' desde la barra de herramientas lateral. En el selector de objetos seleccionaremos entre las opciones ofrecidas el "pulse". Llevaremos el cursor del ratón hasta la derecha del terminal IN y pulsaremos el botón izquierdo para colocar nuestro generador.



Los generadores son similares a cualquier otro objeto de PROTEUS. Podemos usar todas las herramientas habituales para mover, editar, girar o borrar el generador una vez que lo hayamos colocado sobre nuestro diseño.

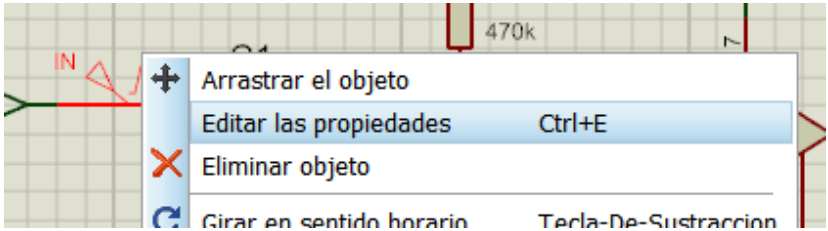
En nuestro caso hemos colocado el generador sobre un cable existente. Pero también es posible, y a veces resulta más cómodo, colocar el generador en cualquier parte de nuestra zona de trabajo y más tarde cablearlo como cualquier otro componente.



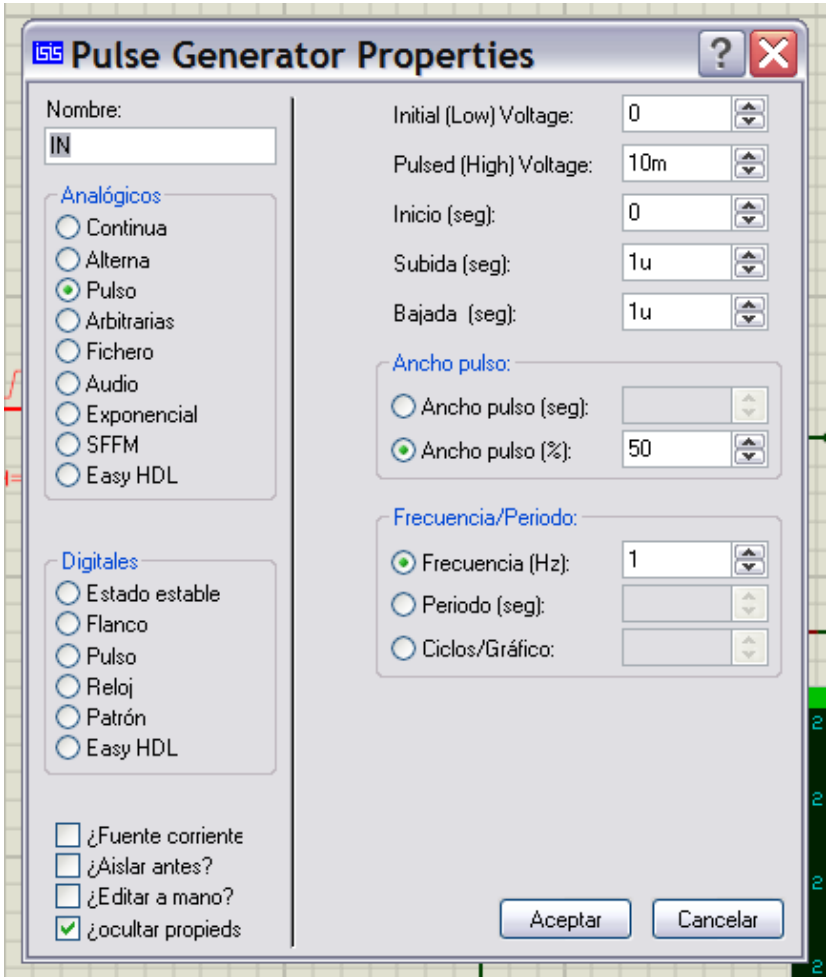
Puesto que nuestro generador lo hemos colocado en un cable, PROTEUS le ha asignado automáticamente el mismo nombre que el terminal al que se encontraba unido ese cable (IN). El procedimiento seguido por PROTEUS para asignar un nombre a un generador es:

1. Primero busca si el cable pertenece a una red concreta y le asigna ese nombre.
2. Si la red no tiene nombre, y el cable está conectado a un terminal le asigna el nombre del terminal.
3. Si el cable no está conectado a una red con nombre ni a un terminal, le asigna el nombre del pin del componente más cercano al que está unido el cable.

A continuación tenemos que ajustar la configuración del generador para conseguir el pulso de las características que deseamos para la comprobación de nuestro circuito. Para configurar el generador, colocaremos el cursor del ratón sobre el generador y pulsaremos el botón derecho. En el menú contextual que aparece seleccionaremos la opción 'Editar las propiedades'



En la ventana de diálogo que se ha abierto introduciremos los siguientes valores: pulsed (high) voltage = 10m y ancho pulso (%) = 50



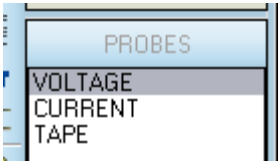
Cerraremos la ventana de diálogo con el botón ‘Aceptar’.

En este circuito sólo vamos a necesitar un generador, pero PROTEUS no pone ningún tipo de limitación al número de generadores que necesitemos utilizar en la simulación de nuestro circuito.

Una vez que hemos colocado el generador que necesitamos para generar la señal de entrada de prueba de nuestro circuito, ahora debemos colocar las sondas en los puntos del circuito donde deseemos monitorizar los valores obtenidos. Obviamente, nosotros estamos interesados en conocer el valor obtenido a la salida del amplificador. También puede ser interesante leer los valores obtenidos en el pin 3, a la entrada del amplificador, después de que la señal de entrada sea tratada. Como es lógico si necesitamos más sondas podemos añadirlas en cualquier momento y repetir la simulación.

Para colocar una sonda seguiremos un procedimiento similar al utilizado para los generadores. Seleccionaremos el icono 'Modo sonda' desde la barra de herramientas lateral (asegurémonos de que no hemos seleccionado una sonda de corriente por error).

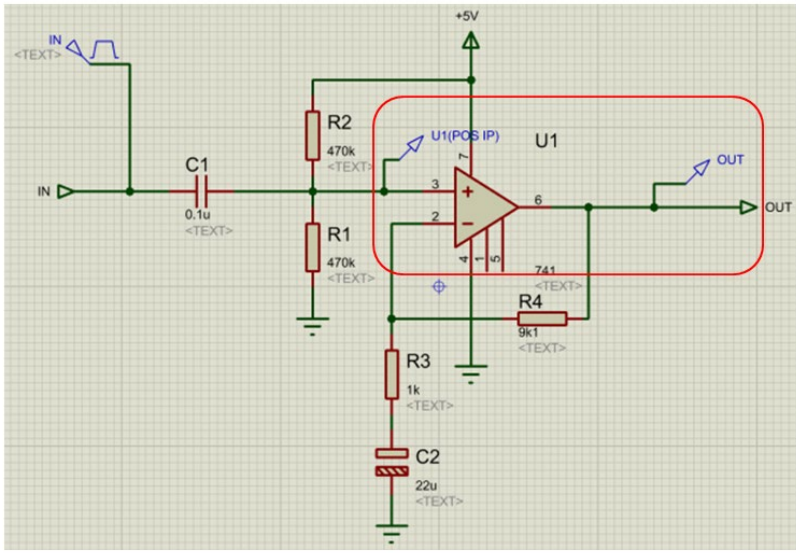




Las sondas pueden ser colocadas en el medio de los cables existentes o situados en espacios libres de nuestra área de trabajo y más tarde cablearlas.

Colocaremos una sonda a la izquierda del pin 3 del componente U1 y después de la unión de los cables que provienen de las resistencias R1 y R2.

La segunda sonda la colocaremos justo a la izquierda del terminal OUT.



El nombre que PROTEUS les asigna a las sondas se corresponde con las reglas que mencionamos con anterioridad en el caso de los generadores. Nombre del dispositivo y nombre del pin entre paréntesis y nombre del terminal más cercano.

Al igual que en el caso de los generadores y como con cualquier otro objeto de PROTEUS, podemos mover, cambiar, borrar o girar las sondas colocadas. Las sondas pueden ser editadas para cambiar su etiqueta, aunque en nuestro caso los valores asignados por defecto nos sirven.

Con la realización de estas tareas, ya tenemos nuestro circuito listo para llevar a cabo la simulación.

12.2.-Utilización de gráficos.

Los gráficos juegan un importante papel en la simulación de circuitos electrónicos en los que no se utilizan microprocesadores. No sólo son un medio de visualizar los resultados, sino que además definen como se va a llevar a cabo la simulación.

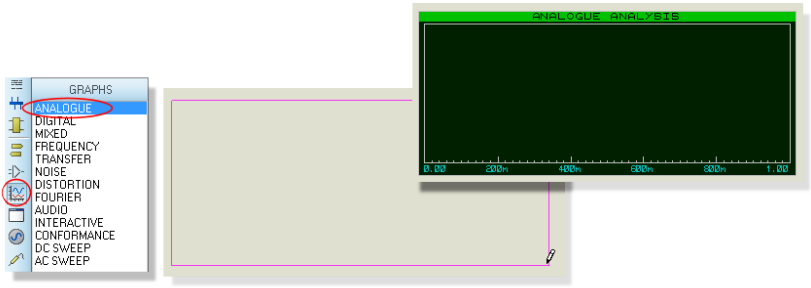
Al colocar uno o más gráficos le estamos indicando a PROTEUS qué tipo de dato estamos esperando recibir (digital, voltaje, impedancia,

etc.). Con esta información PROTEUS elije que tipo de simulación va ejecutar y qué partes del circuito necesita ser incluidas en el proceso de simulación.

Para un análisis de transitorios necesitamos un tipo de gráfico Analógico. Utilizamos el término analógico, en lugar del de gráfico de transitorios, con objeto de marcar la distinción con respecto al gráfico digital, que es el que usamos para visualizar los datos resultantes de un análisis digital, que no es sino una forma especializada de análisis de transitorios.

PROTEUS también nos permite visualizar los resultados procedentes de ambos tipos de análisis conjuntamente (analógico y digital) utilizando un mismo eje compartido de tiempos. Para ello tenemos que utilizar un gráfico de tipo mixto.

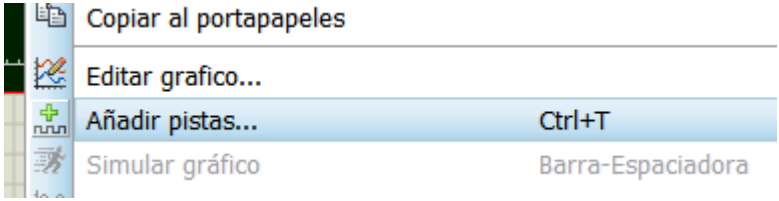
Para colocar un gráfico en nuestro diseño seguiremos los mismos pasos que hemos visto ya con anterioridad en este tutorial. Seleccionamos el icono 'Modo gráfico' de la barra de herramientas lateral y lo situamos en el área de trabajo.



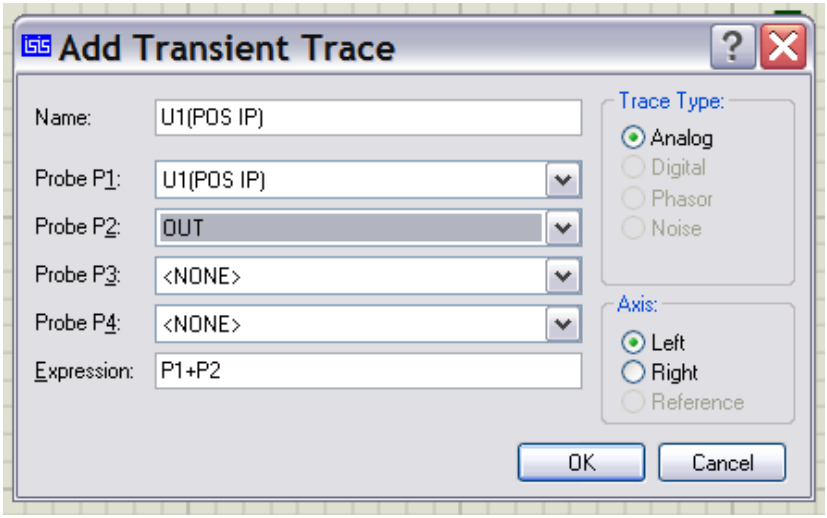
Como ya dijimos con anterioridad, un gráfico es un objeto más de PROTEUS y por lo tanto podemos realizar con él todas las operaciones habituales: cambiar sus propiedades, moverlo de sitio, cambiar su tamaño, maximizarlo, etc.

Es el momento de añadir nuestros generadores y sondas al gráfico. Para ello, 'cogemos y arrastramos' cada una de los generadores y sondas que deseamos visualizar y los 'soltamos' sobre el gráfico. Debemos tener en cuenta que podemos forzar que el generador o la sonda se asocie con el eje y-izquierdo o con el eje y-derecho del gráfico según lo soltemos más cerca de uno o del otro.

También es posible añadir generadores o sondas a nuestro gráfico utilizando un segundo camino, seleccionando la opción ‘añadir pistas…’ en el menú contextual del gráfico.

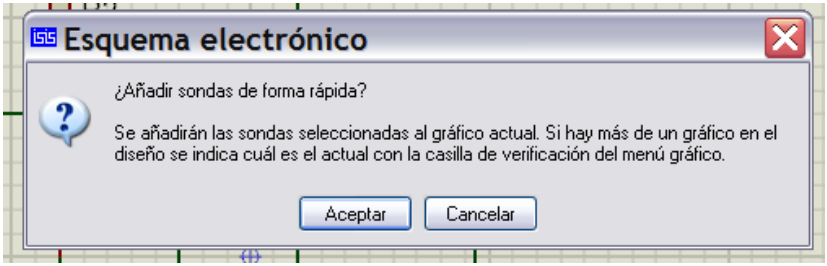


Este comando nos muestra una ventana de diálogo desde la que podemos añadir al gráfico los generadores y sondas que hayamos creado previamente, utilizando las diferentes listas desplegables que pone a nuestra disposición.



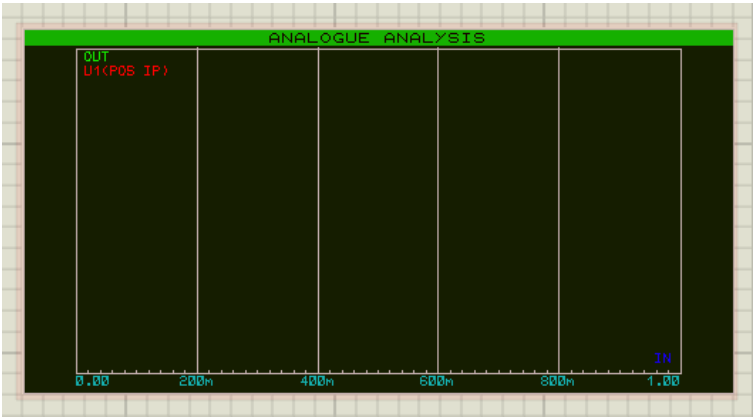
Si nuestro diseño contiene más de una hoja y el gráfico se encuentra situado en una de ellas y la sonda o generador que queremos visualizar en él se encuentra en otra, éste es el único método posible para llevarlo a cabo.

Si tenemos alguna sonda o generador seleccionado, se nos muestra la ventana de diálogo siguiente.



Si damos al botón ‘aceptar’ las sondas y generadores seleccionados se añaden automáticamente al gráfico activo. Si pulsamos el botón ‘Cancelar’ o no tenemos ninguna sonda o generador seleccionados nos muestra la misma ventana de diálogo.

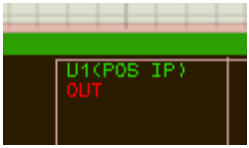
Utilizando cualquiera de los tres métodos, tenemos que colocar nuestro generador y las tres sondas en el gráfico.



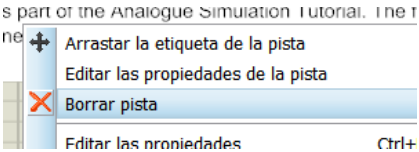
En este momento, cada pista sólo se compone de una etiqueta con el mismo rótulo que el generador o la sonda sobre nuestro gráfico. En la imagen se puede observar que las dos sondas aparecen en la zona superior izquierda y el generador en la zona inferior derecha. Si no puede ver los nombres de las pistas en su gráfico es posible que se deba a que su tamaño en muy pequeño. Aumente el tamaño del

gráfico seleccionándolo y alargando una de las esquina hasta que tenga el tamaño suficiente para ver las pistas que acabamos de crear.

Las pistas se han colocado por orden alfabético de forma automática. Este es un criterio arbitrario pero conveniente en muchos casos. De todas formas, si deseamos cambiarlo, podemos hacerlo de forma sencilla. En primer lugar, nos aseguramos de que el gráfico no está seleccionado, pinchando con el ratón en alguna parte libre de la superficie de trabajo. A continuación ponemos le ratón sobre la etiqueta de la sonda o generador a modificar (el cursor aparece con forma de mano con un crucero de cuatro flechas). Pulsamos el botón izquierdo (la etiqueta se resalta cambiando de color) y, sin soltarlo, lo movemos hasta la nueva posición. Soltamos el botón del ratón y la etiqueta habrá cambiado de sitio.



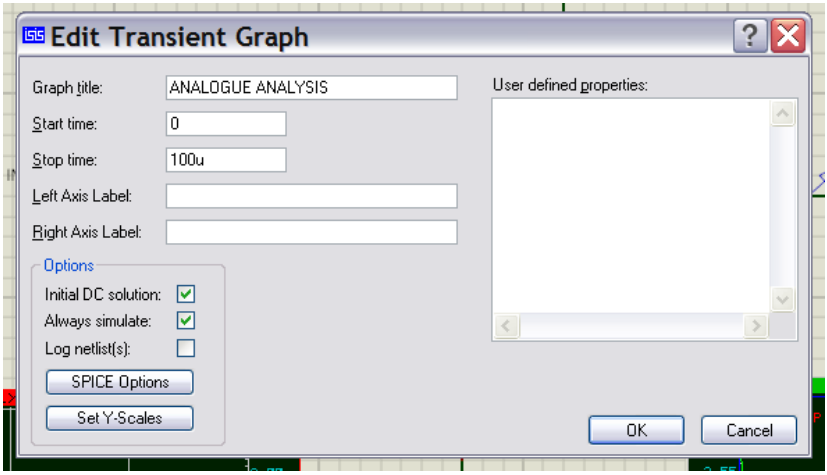
Si en lugar de utilizar el botón izquierdo, usamos el botón derecho del ratón, abriremos el menú contextual con más opciones (por ejemplo, eliminar la pista del gráfico).



Para desmarcar todas las pistas actualmente seleccionadas, solo tenemos que pulsar con el botón izquierdo del ratón sobre cualquier parte del gráfico que no sea alguna de las etiquetas.

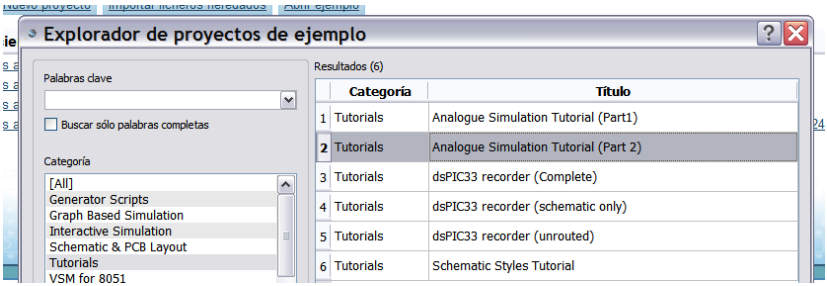
Nos resta un paso final antes de comenzar con la ejecución de la simulación. PROTEUS realizará la simulación del circuito electrónico de acuerdo con la escala temporal que se muestra en el gráfico. Por defecto desde el instante sgundo 0 al instante segundo 1. Puesto que nuestra intención es analizar una onda cuadrada de sonido de muy alta frecuencia (unos 10kHz) significa que el periodo será de 100ms.

Seleccionaremos el gráfico y utilizaremos el botón izquierdo del ratón para abrir la ventana de diálogo de edición. En ella tenemos un campo que nos permite darle un título al gráfico, especificar el instante en que comenzará y finalizará la simulación, darle un nombre a los ejes izquierdo y derecho (en los gráficos digitales esta opción no está disponible).



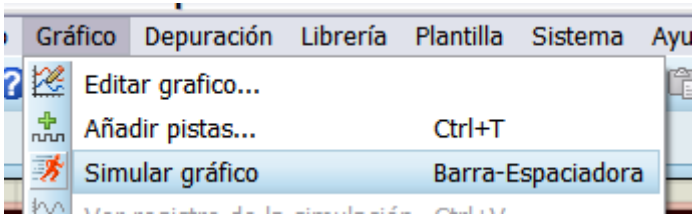
Lo único que tenemos que cambiar es el campo ‘stop time’ y sustituir el ‘1 segundo’ por ‘100u’ (100 microsegundos = $1x10^{-7}$segundos) y pulsar el botón ‘Ok’.

El diseño está ahora listo para la simulación. El fichero de ejemplo ‘Analogue Simulation Tutorial (Part2) es el mismo diseño pero con los generadores, sondas y gráfico ya preparados. Sólo hay que cambiar el tiempo de análisis de 100useg según acabamos de indicar.

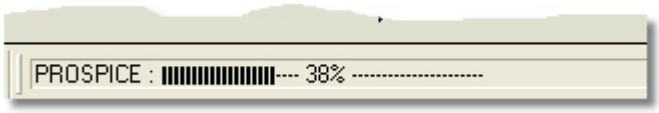


12.3.-Ejecución de la simulación.

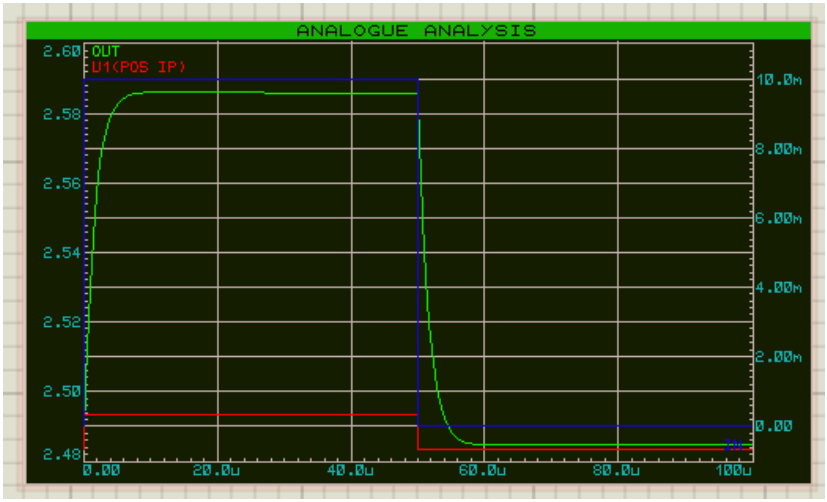
Para ejecutar la simulación podemos utilizar el comando ‘simular gráfico’ desde el menú ‘Gráfico’ o utilizar el atajo de teclado “BARRA ESPACIADORA”. Cualquiera de los dos métodos que elijamos provocará la simulación de nuestro circuito.



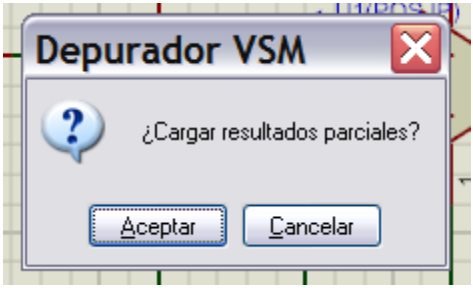
En la barra de estado se va mostrando la evolución de la simulación y, si el proceso es largo porque la simulación es compleja, podremos observar una indicación del tanto por ciento del proceso que se ha ejecutado.



Al finalizar la simulación, el gráfico actualmente activo (el que tenga el foco) será actualizado para mostrar el resultado de la simulación.



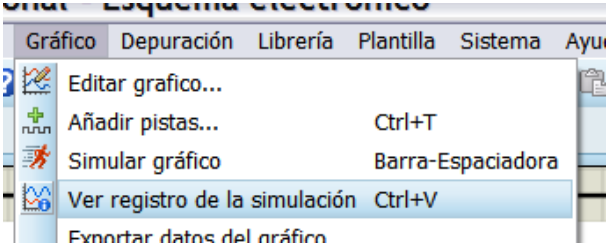
En cualquier momento se puede abortar la simulación utilizando el atajo de teclado “ESC”. Si lo hacemos aparecerá una ventana de diálogo que nos pregunta si queremos mostrar los datos analizados hasta el momento de pulsar la tecla ESCAPE.



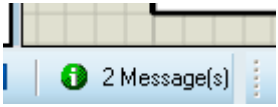
En PROTEUS, el tiempo de inicio de la generación del gráfico es ignorado. En todos los casos la simulación comienza en el momento cero y se ejecuta hasta el momento final fijado en el gráfico. Pero, una vez finalizada la simulación sólo se muestran los datos referentes al periodo seleccionado en el gráfico.

PROTEUS crea siempre un fichero histórico con las incidencias que se producen durante la simulación. Cada vez que se genera una nueva simulación el fichero anterior es borrado y se crea uno nuevo.

Podemos consultar el fichero de incidencias utilizando el comando ‘Ver registro de la simulación’ del menú ‘Gráfico’.



Alternativamente, también podemos utilizar el atajo de teclado “CTRL-V” o pulsar con el botón izquierdo del ratón sobre la barra de estado en el lugar donde se muestra el número de mensajes existentes o desde la opción ‘Ver registro de la simulación’ del menú contextual del gráfico activo.

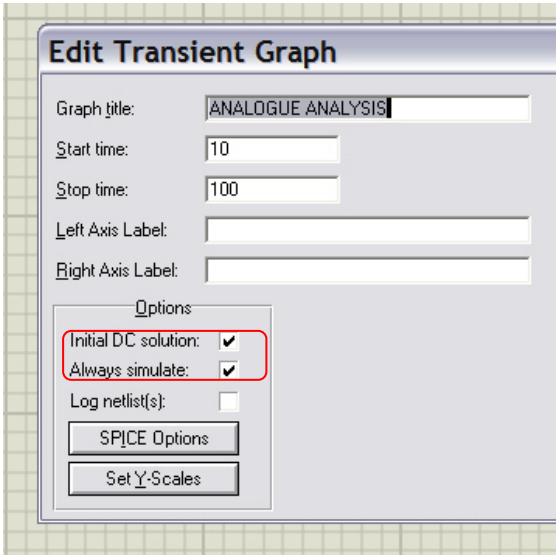


El fichero histórico de una simulación analógica rara vez muestra nada interesante, salvo que se produzcan avisos de advertencia o error. Pero es una buena fuente de información cuando se producen errores en la simulación para encontrar la fuente del problema.

Si volvemos a ejecutar una segunda vez el comando para ejecutar la simulación podemos comprobar algo que puede chocarnos: la simulación no se lleva a cabo. La razón de este comportamiento es que el motor de PROTEUS es suficientemente inteligente para no trabajar de nuevo si alguna parte del diseño no ha sido cambiado. Como en nuestro sencillo caso, el esquema no ha variado, la simulación, simplemente, no vuelve a ejecutarse.

Si por cualquier motivo, deseamos de todas formas volver a ejecutar de nuevo la simulación cada vez que se lo indiquemos aunque no se haya producido ningún cambio, podemos hacerlo utilizando la opción ‘editar las propiedades’ del menú contextual del gráfico o del menú ‘Gráfico’. En la ventana de diálogo que se abre podemos marcar la

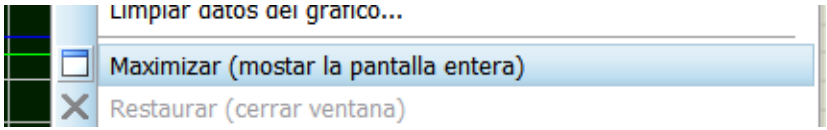
casilla de verificación ‘always simulate’ para forzar este comportamiento.



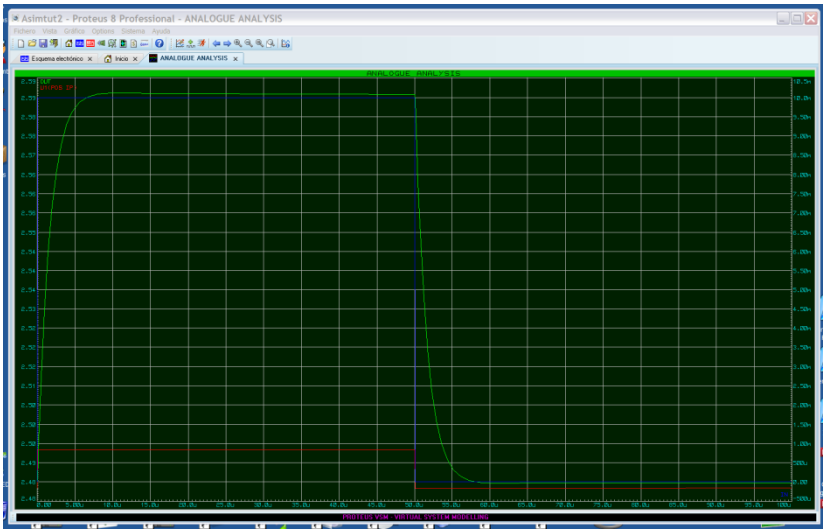
Como ya vimos en la sección 8.2 los gráficos ponen a nuestra disposición muchas posibilidades en lo referente a cambiar el zoom, tomar medidas, etc.

12.4.-Obtención de medidas.

Siempre que colocamos un gráfico en nuestro diseño, por defecto aparece minimizado. Para facilitar la toma de medidas, lo primero que tenemos que hacer es maximizarlo. Para ello, colocaremos el cursor del ratón sobre el gráfico y utilizaremos el botón derecho para abrir el menú contextual. Por último seleccionaremos la opción ‘Maximizar’.



De esta forma se abrirá una nueva pestaña con el gráfico y la barra de herramientas mostrará los iconos relativos a la gestión del gráfico.



En la zona izquierda superior y en la zona derecha inferior se encuentran las etiquetas de las pistas. En la zona central se muestran las propias pistas. Si colocamos el cursor del ratón sobre una de las pistas y pulsamos el botón izquierdo, el valor de la pista en ese punto se muestra en la zona inferior del gráfico. A la izquierda el tiempo y a la derecha la magnitud medida.



El color de cada pista coincide con el color de su etiqueta para facilitar su identificación.

Las etiquetas OUT y U1(POS IP) se encuentran en la zona izquierda y la etiqueta IN a la derecha. La razón de esta situación es que las

magnitudes medidas en un caso y otro son muy diferentes. Mientras que las dos primeras estamos hablando de valores en voltios en la tercera se trata de milivoltios. Si pusiéramos todas en el mismo eje, la pista correspondiente a la etiqueta IN casi no tendría significación frente a los valores de las otras dos.

Proteus nos permite utilizar dos escalas de medida en cada gráfico (lado izquierdo y derecho). Para colocar una sonda o generador en uno u otro lado, sólo la tenemos que 'coger y arrastrar' de un lado del gráfico al otro. Podemos probar en este momento a llevar la etiqueta IN al lado izquierdo y observar los resultados (la lectura de esta pista se hace muy complicada). Volvamos a dejar la etiqueta IN en su posición original.

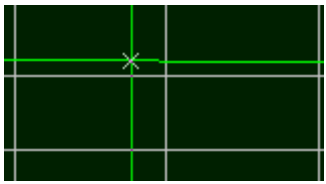
Conviene acostumbrarse al uso de dos escalas en Proteus observando y entendiendo que la pista IN, aunque parece del mismo valor que la pista OUT en el gráfico, no lo es en absoluto, porque estamos utilizando dos escalas diferentes para mostrar una y otra pista (voltios y milivoltios).

Borremos la pista IN colocando el ratón sobre esta etiqueta, pulsando el botón derecho para abrir el menú contextual y seleccionando la opción 'borrar pista'. Nos quedaremos sólo con dos pistas.

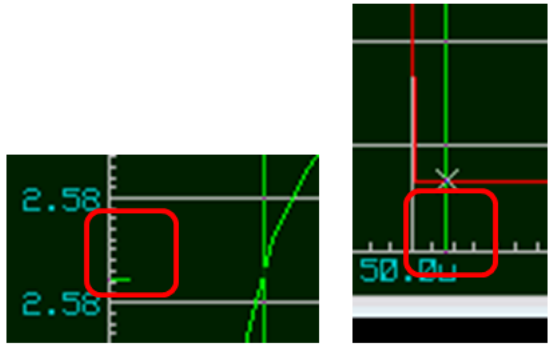
Vamos a practicar con la medición de dos cantidades utilizando cursores:

- La ganancia de tensión del circuito.
- El tiempo de retardo de la salida.

Cada gráfico dispone de dos cursores. Uno de ellos recibe el nombre de 'Referencia' y el otro el de 'Primario'. El cursor de referencia se muestra de color rojo y el cursor primario en color verde. Uno de los cursores siempre está bloqueado en una pista y el lugar donde el cursor está bloqueado sobre la pista se muestra con una pequeña 'x'.



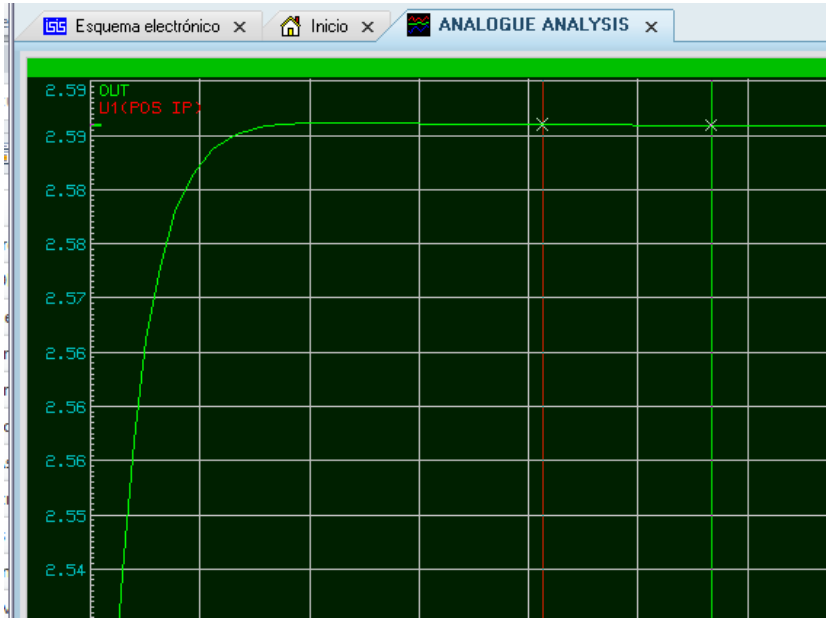
Una pequeña marca en el eje ‘x’ y en el eje ‘y’ señalan el valor de ese punto de bloqueo para facilitar su lectura y conocer el valor en ese punto.



Si utilizamos la flecha izquierda y derecha del teclado, nos desplazaremos por el gráfico a saltos de igual tamaño a la unidad de medida del eje ‘x’ (la división más pequeña de ese eje).

Hasta ahora sólo hemos utilizado el cursor primario de color verde. Vamos a colocar en este momento el cursor de referencia de color rojo. Las mismas funciones y posibilidades están disponibles para uno y otro cursor. La única diferencia es que para utilizar el cursor de referencia tenemos que mantener pulsada la tecla ‘CTRL’[6]. Para colocar el cursor de referencia sólo tenemos que colocar el puntero del ratón sobre el lugar de la curva donde lo queremos colocar, mantener pulsada la tecla ‘CTRL’ y actuar sobre el botón izquierdo. El cursor de referencia aparece de color rojo.

6 Se ha diseñado la utilización de la tecla CTRL para el cursor de referencia porque se utiliza mucho menos que el cursor primario.



Debemos tener cuidado y no olvidar pulsar la tecla ‘CTRL’ si queremos colocar o mover el cursor de referencia. Si no lo hacemos y lo olvidamos, moveremos de lugar el cursor primario. Si mantenemos pulsada la tecla ‘CTRL’, mantenemos pulsado el botón izquierdo y desplazamos el cursor del ratón, desplazaremos el cursor primario al lugar del gráfico que deseemos.

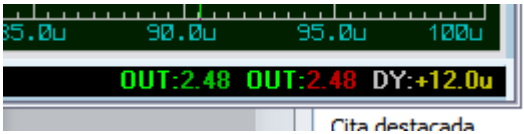
Practiquemos esta técnica y coloquemos el cursor de referencia sobre el instante 75u del eje ‘x’.



En el lado izquierdo de la barra de estado veremos que se muestran los valores de los tiempos correspondientes a ambos cursores. En rojo el tiempo correspondiente a la medida del cursor de referencia y en verde el del cursor primario.

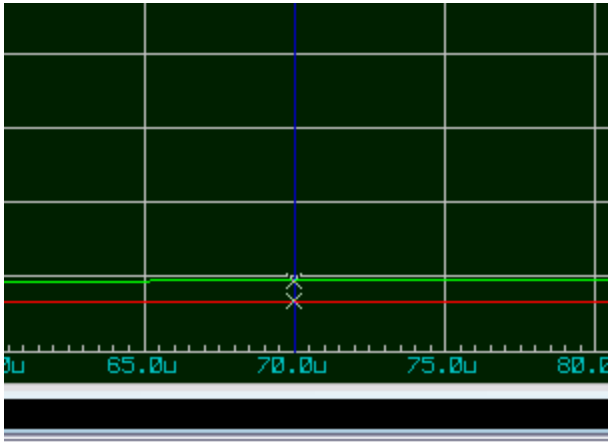


En el lado derecho de la barra de estado veremos que se muestran los valores de las magnitudes correspondientes a ambos cursores. En rojo la tensión correspondiente a la medida del cursor de referencia y en verde la del cursor primario.



Podemos mover el cursor de referencia a intervalos del mismo tamaño de la escala del eje ‘x’ manteniendo pulsada la tecla CTRL y utilizando las teclas izquierda y derecha del teclado. Si no mantenemos pulsada la tecla CTRL moveremos el eje primario. Es un buen momento para practicar con esta técnica.

Probemos a colocar ahora el cursor primario sobre la pista que nos da el valor a la entrada del amplificador. Los dos cursores en el instante 70u. Tendremos que tener algo similar a esto.





En el lado izquierdo de la barra de estado podemos ver que ambos cursores están en el mismo momento (70u) y en el lado derecho podemos ver la diferencia en ese punto entre el valor medido a la entrada del amplificador operacional y la salida.

Fijemos nuestra atención en que Proteus nos facilita la toma de medidas. Para ello nos ofrece un tercer valor en ambos casos denominado DX (delta X) y DY (delta Y). Este valor representa la diferencia de valor entre el cursor de referencia y el cursor primario tanto en el eje X como en el eje Y.

Así que, interpretando ahora los datos de las imágenes anteriores, podemos ver que entre el cursor de referencia y el primario no ha habido diferencia de tiempo (DX=0) y hay una diferencia entre los valores de tensión medida de -1,36mV (DY=-1.36m).

Coloquemos los cursores ahora en el espacio de tiempo 20u para practicar de nuevo las técnicas vistas. Colocamos el cursor alrededor de la marca 20u sobre la pista OUT. Luego con las teclas izquierda y derecha del teclado la ajustamos a la medida de 20u. Ahora hacemos lo mismo pulsando la tecla CTRL y colocando el cursor sobre la pista U1(POS IP). Tenemos que obtener lo mismo que se muestra en las siguientes imágenes.



Como podemos ver, y puesto que ya sabemos interpretar perfectamente nuestras medidas, la entrada es de 2,49mV, la salida es de 2,59mV y la diferencia de 92,6mV (casi 100mV).

Podemos seguir practicando con estas técnicas para obtener la ganancia del amplificador operacional.

Midamos con los cursores la diferencia de los valores de la pista U1(POS IP) en la entrada en los instantes 40u y 70u (pulso alto y bajo). Tendremos.



Hagamos ahora lo mismo con la curva OUT. Tendremos.

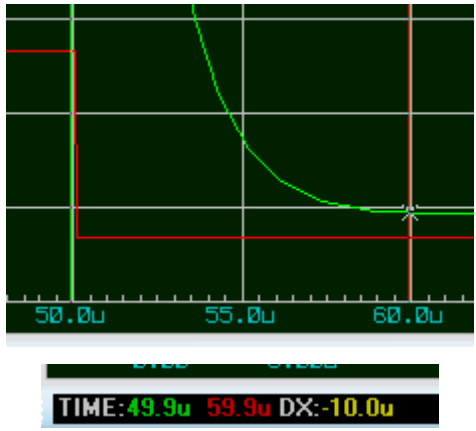


Interpretando los datos obtenidos vemos que en la entrada del amplificador tuvimos una variación de 10mV y en la salida una variación de 100mV. Es decir una ganancia de 10 veces, lo que coincide con el valor teórico que habíamos obtenido de los valores de las resistencias R3 y R4.

Los valores de la diferencia se muestran en negativo por las posiciones relativas del cursor primario y de referencia. Si los hubiéramos intercambiamos, los valores serían positivos.

Para terminar, vamos a medir el tiempo de reacción del amplificador operacional. Para eso colocamos el cursor de referencia en el momento en que la entrada pasa del valor alto al bajo. Luego colocamos el cursor primario en el punto en que la curva de salida se pone en el valor bajo.

Obtendremos los datos siguientes:



Es decir que ha reaccionado aproximadamente en 10useg.

Como hemos terminado con las medidas, ya podemos cerrar la pestaña de nuestro gráfico y volver a la pestaña esquema electrónico.

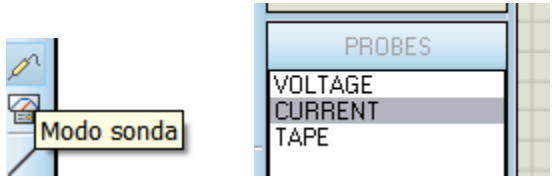
12.5.-Sondas de corriente.

Hasta ahora sólo hemos utilizado las sondas de voltaje para medir la tensión en un determinado punto de nuestro circuito. En este apartado vamos a utilizar una sonda de corriente para examinar la intensidad que circula a través de la resistencia R4.

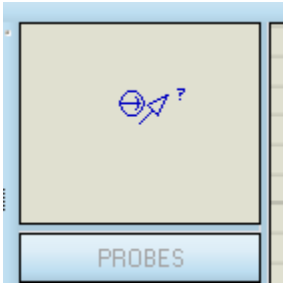
Las sondas de corriente se utilizan de forma similar a las de voltaje, pero con una importante diferencia. Las sondas de corriente necesitan tener una dirección asociada con ellas. Una sonda de corriente funciona como si efectivamente rompiéramos el cable en ese punto y la insertáramos entre los extremos resultantes. Tenemos que indicarle en qué dirección va a circular la corriente y lo haremos según la forma en que la coloquemos.

Por defecto la orientación predeterminada (la flecha inclinada hacia la derecha) la sonda mide el flujo de corriente de izquierda a derecha. Para medir la corriente que circula por un cable orientado verticalmente necesitamos girar la sonda un ángulo de 90° o 270°. Si colocamos la sonda en un ángulo equivocado se generará un error cuando ejecutemos la simulación que podemos encontrar en el fichero histórico. En caso de duda recordemos siempre la siguiente regla nemotécnica: la dirección de la flecha de la sonda indica la dirección en que fluye la corriente.

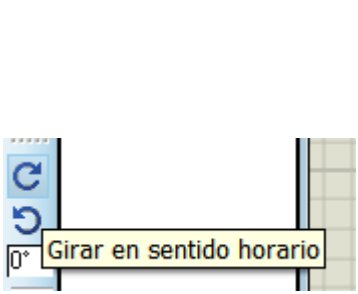
Para colocar nuestra sonda de corriente, empezaremos por seleccionar el icono ‘Modo sonda’ de la barra de herramientas lateral y eligiendo en las opciones que aparecen en el selector de objetos ‘current’.

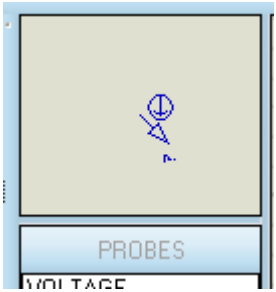


Observa que en la ventana del selector de objetos se muestra una sonda apuntando el flujo de corriente hacia la derecha.

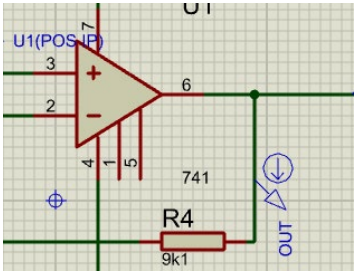


A continuación pulsaremos una vez con el ratón izquierdo sobre el icono ‘girar en sentido horario’ de la barra de herramientas lateral para que el flujo de corriente (en el selector de objetos) apunte hacia abajo.

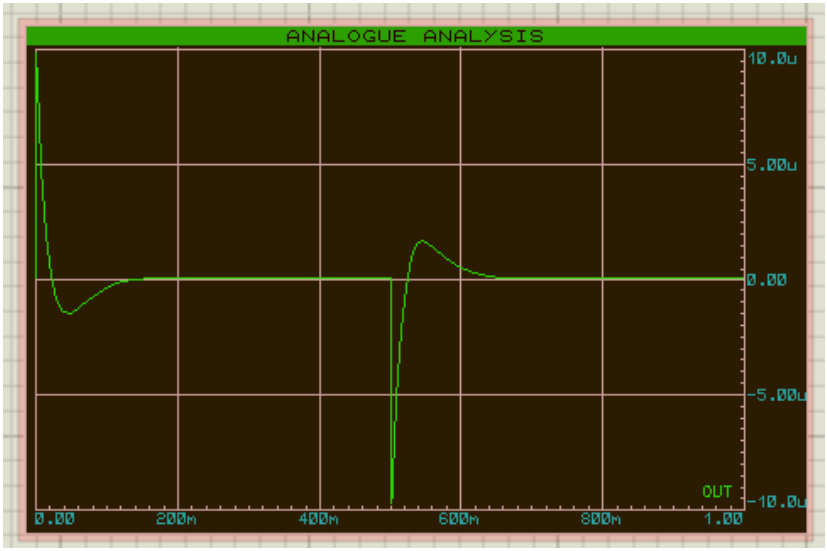




A continuación vamos a colocar la sonda en el cable vertical que une la parte derecha de la resistencia R4 y el pin 6 del amplificador operacional U1.

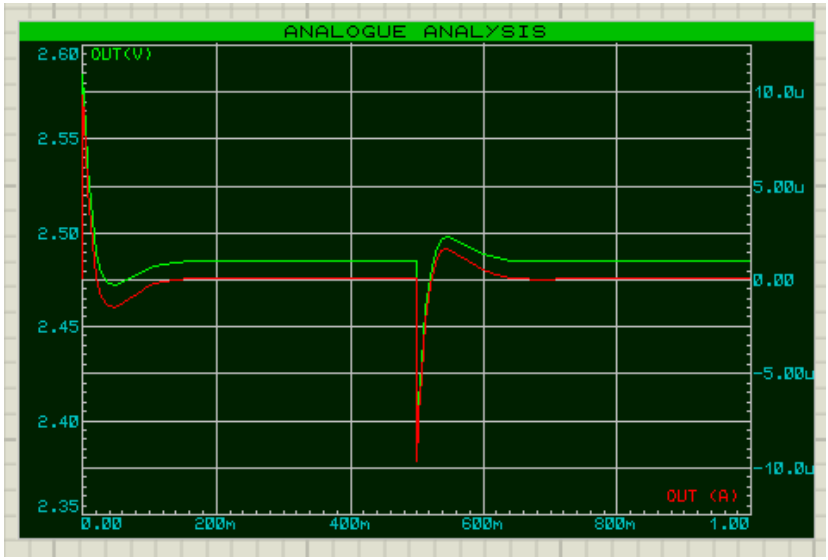


Ahora crearemos un gráfico nuevo y “cogeremos” nuestra sonda y la “soltaremos” sobre el gráfico. A continuación ejecutaremos la simulación utilizando el atajo de teclado “BARRA ESPACIADORA”. El gráfico que obtendremos tiene que ser similar al mostrado en la imagen siguiente.



Podemos combinar en un mismo gráficos sondas de voltaje con sondas de corriente. Cuando lo hagamos conviene situar las de un tipo cerca de lado izquierdo del gráfico y las otras cercas del lado derecho, porque las unidades de medida en uno y otro caso son sensiblemente diferentes.

En la imagen siguiente se muestra un gráfico donde se puede ver la medida de la sonda de tensión situada a la salida (en verde) y la medida de corriente que acabamos de crear (en rojo). Las escalas izquierda y derecha se refieren, lógicamente, una a tensiones y la otra a corrientes. Para facilitar la interpretación del gráfico hemos renombrado la sonda de corriente etiquetándola como OUT (A) y la de tensión como OUT (V).



Incluso en el gráfico sin maximizar, podemos comprobar que la corriente en el lazo de retroalimentación sigue muy de cerca la forma de onda de la tensión de salida, tal y como era de esperar en el funcionamiento de un amplificador operacional. Como siempre, podemos maximizar el gráfico para verlo más cómodamente.

12.6.-Análisis de frecuencias.

Al igual que hemos realizado análisis de transitorios, podemos realizar otros tipos de análisis más durante la simulación de circuitos analógicos. Todos ellos se utilizan de la misma forma, con ayuda de generadores, sondas y gráficos, puesto que todos son variaciones de un mismo tema.

El próximo tipo de análisis que abordaremos en este tutorial es el análisis de frecuencias. Con los análisis de frecuencias podemos conocer cómo se comporta un determinado circuito a diferentes frecuencias. La simulación utilizando el análisis de frecuencias funciona de forma similar a si utilizáramos a la entrada del circuito analizado un generador de frecuencias y midiéramos los valores recibidos a la salida con un voltímetro de corriente alterna. Bien es verdad, que con nuestro análisis de frecuencias podemos medir en

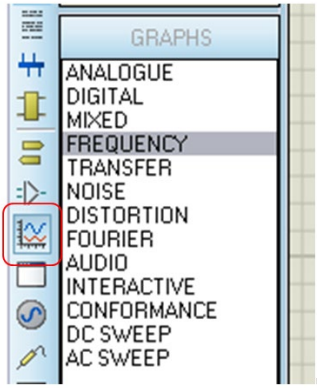
el punto de salida no sólo la magnitud de la señal sino también su desfase.

Los análisis de frecuencia nos facilitan las respuestas de nuestro circuito a diferentes frecuencias. Por ello resultan muy útiles para chequear que el funcionamiento de filtros es el esperado o que una determinada etapa de amplificación funciona correctamente para el rango de frecuencias en el que es previsible su utilización.

Para obtener más información acerca de la forma en que PROTEUS realiza los cálculos necesarios durante la ejecución de un análisis de frecuencias, se puede consultar en la ayuda en línea.

En este tipo de análisis, el eje x se utiliza para representar la frecuencia (utilizando una escala logarítmica), mientras que en el eje y se representarán ambos, la magnitud y la fase medidas en el punto donde coloquemos la sonda.

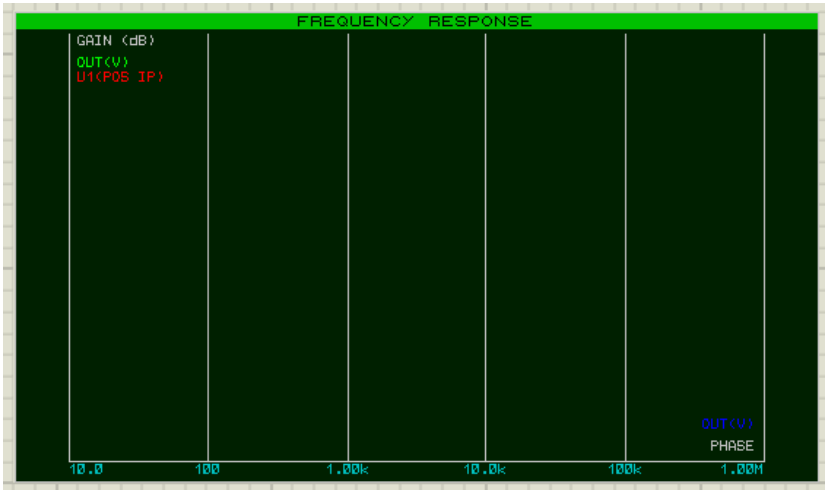
Para realizar un análisis de frecuencias, necesitamos utilizar un gráfico de frecuencias. Para usarlo pulsaremos sobre el icono 'Modo gráfico' de la barra de herramientas lateral y seleccionaremos el tipo 'frecuency' en la ventana de selección de objetos.



A continuación, colocaremos el grafico en el área de trabajo siguiendo el mismo procedimiento que hemos utilizado hasta ahora. No es necesario que borremos el gráfico que ya tenemos colocado, pero puede ser conveniente hacerlo para obtener más espacio libre y poder crear el nuevo gráfico de un tamaño que nos resulte más cómodo para usarlo.

El siguiente paso consiste en añadir las sondas a nuestro gráfico. Vamos a utilizar las dos existentes, OUT y U1(POS IP). En los gráficos de frecuencias, los dos ejes y (el izquierdo y el derecho) tienen un significado previamente fijado. El eje y-izquierdo se utiliza para visualizar la magnitud de la señal medida por la sonda y el eje y-derecho para la fase. Con objeto de poder visualizar ambas, tenemos que añadir las dos sondas a ambos lados del gráfico.

Cogeremos la sonda OUT[7] y la soltaremos en el lado izquierdo del gráfico. A continuación, volveremos a cogerla y la volveremos a soltar, esta vez en el lado derecho del gráfico. Haremos la misma operación con la sonda U1(POS IP) pero esta vez solo en el lado izquierdo. El gráfico resultante se muestra en la imagen siguiente
.



Los valores de la magnitud y fase deben ser especificados con referencia alguna fuente que sirva de base. En PROTEUS esto se realiza especificando un generador de referencia. Un generador de referencia siempre tiene una salida de 0dB (1Volt con un desfase de 0°). Cualquier generador existente puede ser definido como el generador de referencia. Todos los otros generadores que se encuentren en el circuito serán ignorados para la realización del análisis de frecuencias.

7 Si renombramos nuestra sonda OUT según el epígrafe anterior para distinguir la sonda de tensión y la de corriente, aquí tendremos que utilizar la sonda routulada OUT (V).

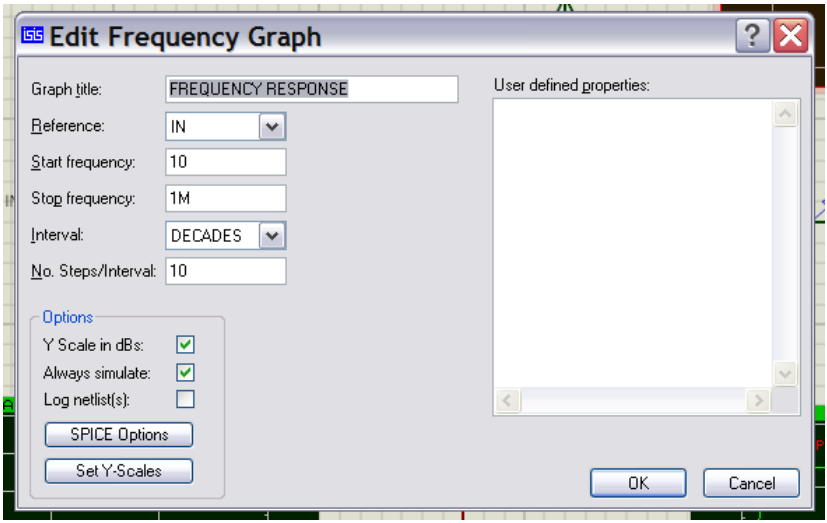
Para especificar que el generador que ya hemos utilizado en nuestro diseño (IN) es el que queremos definir como generador de referencia simplemente tenemos que “cogerle” y “soltarle” dentro del gráfico, igual que hicimos para añadir nuestras sondas. PROTEUS asume que, puesto que es el único generador, estamos añadiendo a nuestro gráfico el que se va a utilizar como referencia y mostrará un mensaje en la línea de estado confirmando esta circunstancia.



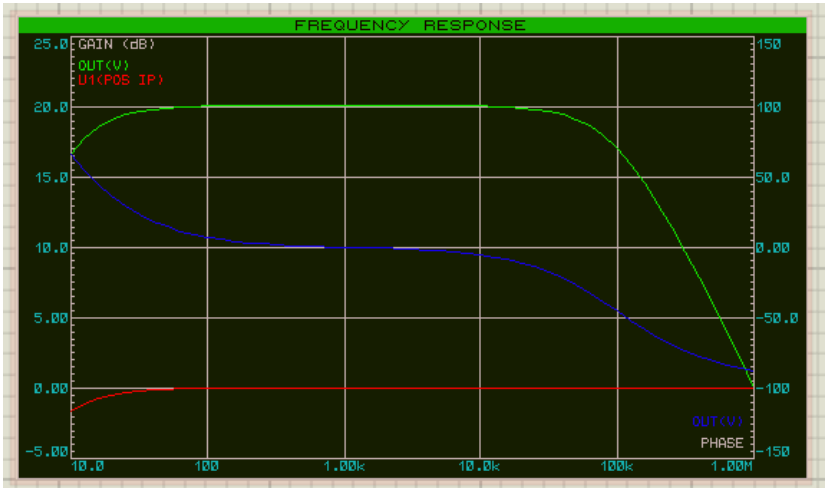
Debemos asegurarnos siempre de realizar este paso siguiendo las instrucciones que acabamos de indicar, porque si no lo hacemos la simulación no funcionará correctamente.

En nuestro caso, no es necesario modificar ninguna propiedad del gráfico porque el rango ofrecido por defecto es el adecuado. De todas formas, si lo deseamos, podemos modificarlos según nuestras necesidades. Para ello colocaremos el cursor del ratón sobre el gráfico para que ostente el foco y utilicemos el atajo de teclado “CTRL-E” o seleccionaremos la opción ‘Editar las propiedades’ desde el menú contextual o desde el menú ‘Gráfico’. Con cualquiera de los métodos que hayamos elegido, la ventana de diálogo se nos mostrará.

El caso de los gráficos de frecuencia es un poco diferente a la de los gráficos de análisis de transitorios. Hay una casilla de verificación que nos permite seleccionar si los datos referentes a la magnitud se visualizaran en unidades normales o en unidades relativas (decibelios). Esta opción es preferible dejarla como está por defecto porque, de otra forma, los valores visualizados no serán los reales obtenidos del circuito.

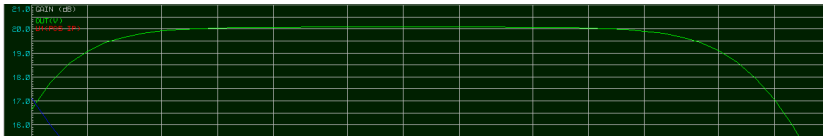


Ejecutaremos la simulación, como siempre, utilizando el atajo de teclado “BARRA ESPACIADORA” o las opciones de menú ya conocidas.



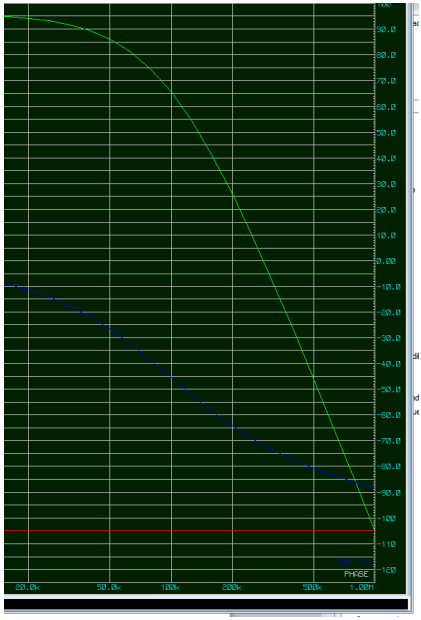
Podemos maximizar la ventana para consultar los datos recibidos con más comodidad.

Fijemos nuestra atención en primer lugar sobre la representación de la curva de la magnitud OUT.

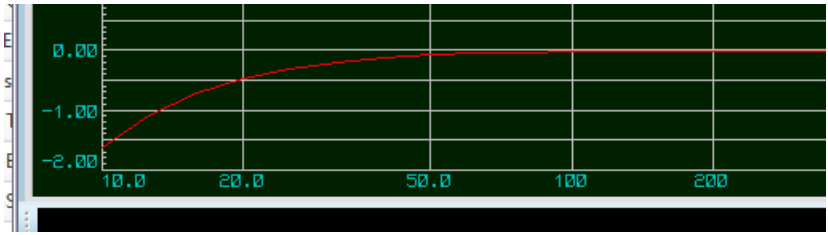


La ganancia está situada justo sobre los 20dB y el rango de frecuencias válido entre 50Hz y 20kHz. El cursor trabaja en la misma manera que ya vimos. Simplemente lo colocaremos sobre el punto de la curva del que queremos conocer su valor y se nos mostrará en la barra de estado. En nuestro caso puede ser especialmente útil puesto que la escala del eje X es logarítmica.

La representación de la curva de la fase en OUT también muestra la distorsión de fase predecible en los extremos de la gráfica, cayendo a -90° justo en la zona derecha del gráfico cuando la ganancia toma el valor de la unidad.



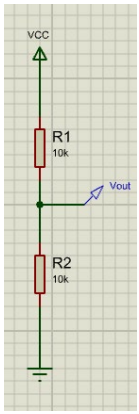
El efecto del filtro pasa-alta colocado a la entrada del amplificador operacional se puede ver claramente en la representación de la curva de la magnitud U1(POS IP).



12.7.-Análisis de barrido AC y DC.

PROTEUS nos permite visualizar cómo se ve afectada la salida de nuestro diseño por los cambios realizados en el circuito. Hay dos tipos de análisis que nos permiten hacerlo: el barrido DC y el barrido AC. El análisis de barrido DC nos permite obtener un gráfico donde se visualiza la serie de los valores obtenidos en la salida, en función de las variaciones experimentadas por el valor definido en la variable de barrido. El análisis de barrido AC hace lo equivalente con los valores (magnitud y fase) obtenidos en la salida. La variable de barrido puede ser cualquier propiedad de nuestro circuito. Por ejemplo, el valor de una resistencia, la ganancia de un transistor o la temperatura del circuito.

Puesto que ambos tipos de análisis se utilizan de forma similar, sólo vamos a centrar nuestra atención en uno de ellos, el barrido DC. Para aprender el uso de esta técnica vamos a construir un circuito muy simple que hace las veces de divisor de tensión utilizando dos resistencias, según se puede ver en la imagen adjunta.

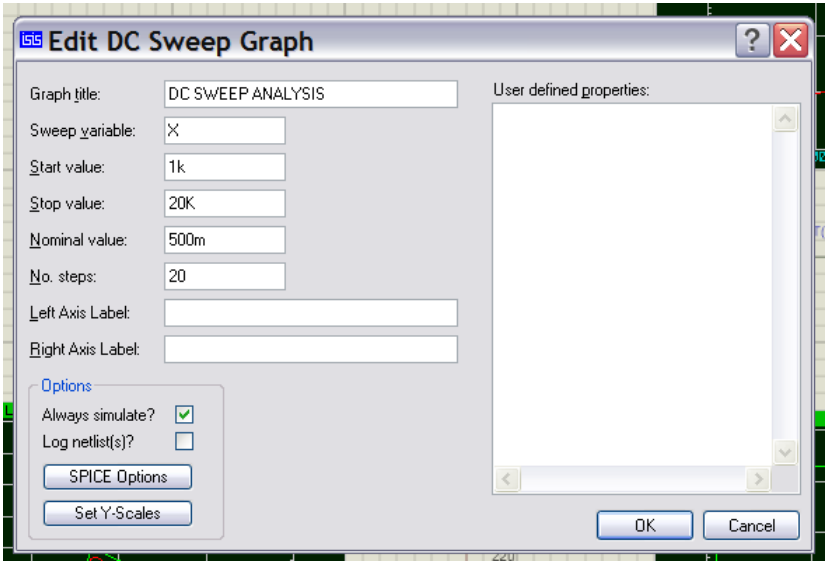


Si ejecutáramos la simulación el valor leído en la sonda Vout será, lógicamente, de 2,5V puesto que el valor de ambas resistencias es el mismo.

Supongamos ahora que deseamos saber cómo varía la tensión en el punto donde hemos situado la sonda Vout en función de que utilizáramos resistencias de diferentes valores para R1.

Lo primero que tenemos que hacer es editar las propiedades de la resistencia R1 y sustituir el valor de 10k por la letra “X” en mayúsculas. Con ello estamos indicándole a PROTEUS que el valor de la resistencia es variable y que tome los datos a partir de una variable de nombre “X” (obviamente podemos elegir el nombre de la variable que deseemos).

A continuación utilizaremos la herramienta ‘Modo gráfico’ de la barra de herramientas lateral para colocar un gráfico en nuestro área de trabajo. En la ventana de selección de objetos elegiremos un gráfico de tipo ‘DC sweep’. Una vez que lo hayamos colocado editaremos sus propiedades abriendo la ventana de diálogo de la forma habitual.

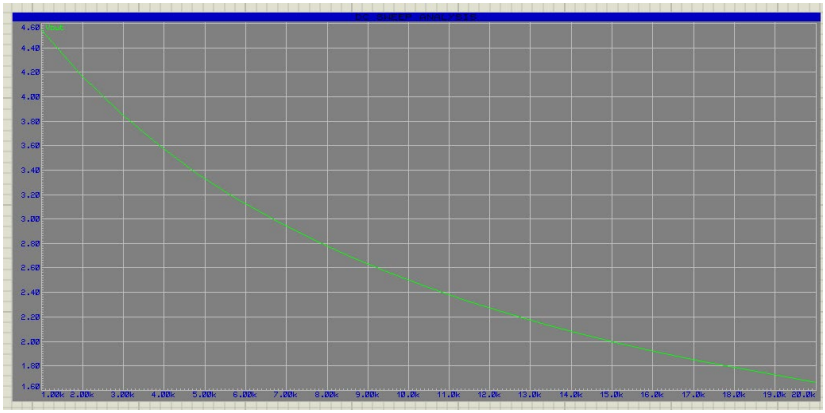


En el campo ’sweep variable’ tenemos que indicar la variable que será utilizada como base para analizar las modificaciones que

estamos analizando con nuestro gráfico. En este caso la magnitud que varía es la resistencia R1 y la variable, por lo tanto, 'X'.

Los dos siguientes campos indican el valor mínimo y máximo del rango de valores que queremos examinar. En nuestro caso vamos a estudiar el valor de tensión obtenido para valores de la resistencia R1 entre 1k y 20k.

El parámetro 'No. Steps' nos indica el número de saltos que tomaremos entre en el rango de valores posibles para la variable de barrido. Vamos a indicarle 20 para que haga un análisis de todos valores de resistencia de 1 en 1k. Finalmente, acabaremos cerrando la ventana de diálogo con el botón "ok" y generando la creación de la gráfica con el atajo de teclado "BARRA ESPACIADORA".



En la gráfica resultante podemos ver los diferentes valores de tensión que obtendremos en la sonda Vout en función del valor de la resistencia R1. Cómo era predecible en el funcionamiento del divisor de tensión, como el valor de R2 es fijo, cuanto mayor sea el valor de R1 menor será la tensión medida en la sonda.

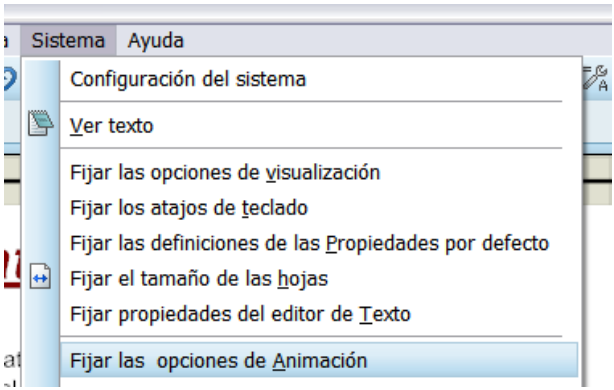
En la ayuda en línea podemos encontrar más información relativa al resto de tipos de análisis que podemos realizar con PROTEUS VSM.

13.-Configuración de las simulaciones.

PROTEUS nos permite configurar la simulación para adaptarla lo mejor posible a nuestras necesidades. Los parámetros de configuración se dividen en dos grandes apartados: opciones de animación y opciones de simulación.

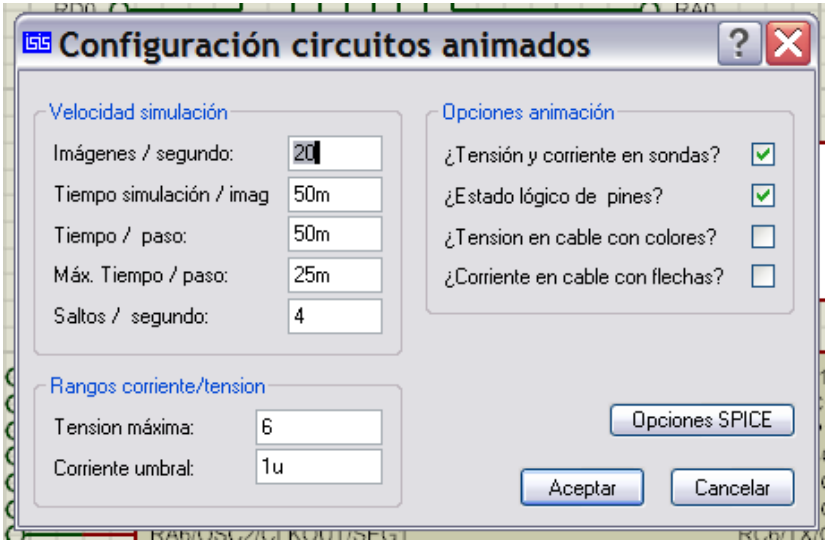
13.1.-Opciones de animación.

Los parámetros de animación nos permiten controlar cómo la simulación interactiva evoluciona en tiempo real. Podemos acceder a ellos desde la opción 'Fijar las opciones de Animación' del menú 'Sistema'.



Están divididos en tres apartados: Velocidad simulación, opciones de animación y Rangos corriente/tensión.

En la sección 'Velocidad simulación' disponemos de cuatro parámetros.



El parámetro 'imágenes / segundo' determina el número de veces por segundo que la pantalla es refrescada con los nuevos estados que se van produciendo.

El parámetro 'tiempo simulación / imag' fija qué cantidad de tiempo de simulación se produce durante cada cambio de pantalla.

Por ejemplo podemos fijar un cambio de pantalla cada 100ms y que se produzcan 1 cambio de pantalla por segundo. Con ello estaríamos logrando que la simulación se produjera dándonos la sensación de que se está desarrollando a cámara lenta (cada segundo de evolución del circuito en tiempo real tardaría en simularse diez segundos en Proteus).

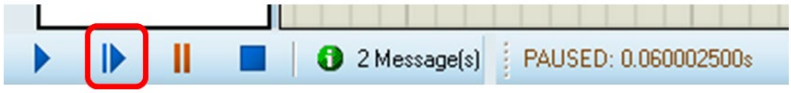
Evidentemente, para que obtengamos simulaciones en tiempo real los dos parámetros tienen que estar armonizados. Si fijamos un valor de 10 cambios de pantalla por segundo, cada pantalla tiene que llevar a cabo la simulación de 100ms. Igualmente 20 pantallas por segundo se correspondería con 50ms por pantalla para obtener tiempo real y 50 pantallas por segundo se correspondería con 20ms por pantalla para obtener, de nuevo, tiempo real.

Si la potencia del ordenador personal con el que estamos llevando a cabo la simulación fuera ilimitada, lógicamente la mejor alternativa sería esta última, puesto que nos permitiría una simulación en

tiempo real con un refresco de pantalla máximo. Pero desgraciadamente, sólo unos pocos privilegiados disponen de un ordenador con tantos recursos. Por eso lo habitual es configurar nuestra simulación para obtener los mejores resultados en nuestro ordenador personal jugando con el reparto de recursos entre el cálculo y la presentación de datos en pantalla.

En ocasiones tampoco es nada despreciable la posibilidad de realizar simulaciones a cámara lenta para afinar la exactitud de la representación, aunque se produzca en tiempos varias veces superior al real. De esta manera logramos que circuitos rápidos se ejecuten lentamente, dando tiempo a observar los efectos que se van produciendo.

El parámetro 'tiempo/paso' controla el tiempo que va a avanzar la simulación cuando utilizamos el botón de "salto simple" en el panel de controles de la simulación.



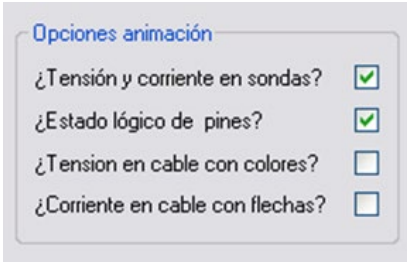
Podemos probar a asignar para este parámetro un valor de 10m y luego pulsar repetidamente el botón de 'salto simple' observando como el tiempo de simulación se va incrementando en la barra de tareas en unidades de 10 milisegundos.

El parámetro 'Max tiempo / paso' fija un tiempo máximo para desarrollar la simulación durante un paso. Valores más altos producen simulaciones más rápidas a costa de la exactitud de los resultados.

El parámetro 'saltos / segundo' fija las instrucciones que se ejecutarán en cada salto que se produce durante la simulación del software que se ejecuta en los microprocesadores. Por defecto, el valor es cuatro, que significa que cada vez que ejecutemos el programa con la opción 'animación en un solo paso' del menú 'depuración' se ejecutarán cuatro instrucciones por segundo.



En el apartado ‘Opciones de animación’ disponemos de cuatro parámetros más.



Cuando activamos el parámetro ‘¿tensión y corriente en sondas?’ forzamos a que durante la simulación aparezca junto a cada sonda situada en nuestro circuito el valor medido en ese punto.

Cuando activamos el parámetro ‘¿estado lógico de pines?’ forzamos a que junto a cada pin de los componentes que forman nuestro circuito aparezca un cuadrado rojo, azul o gris en función de que el estado “lógico” de ese pin sea activado, desactivado o indefinido.

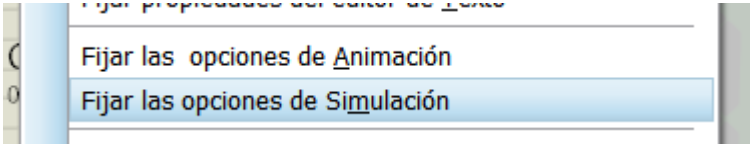
Cuando activamos el parámetro ¿tensión en cable con colores? forzamos a que cada cable adquiere una tonalidad más oscura en función de la mayor o menor tensión a la que se encuentre conectado. La tonalidad más oscura se utilizará para la tensión fijada con el valor del parámetro ‘Tensión máxima’ del apartado ‘Rangos corriente/tensión’.

Cuando activamos el parámetro ‘¿corriente en cable con flechas?’ forzamos a que en cada cable se indique la dirección en la que circula la corriente con una flecha. El nivel mínimo de corriente que debe circular por un cable para que aparezca la flecha indicando su dirección se fija con el valor del parámetro ‘corriente umbral’ del apartado ‘Rangos corriente/tensión’

Cuántos más parámetros de esta sección estén seleccionados, más recursos se gastarán durante la simulación, aunque recibiremos más información gráfica y será más fácil evaluar de forma intuitiva el estado del circuito. Por eso se debe procurar alcanzar siempre un compromiso entre recursos consumidos y beneficios gráficos obtenidos.

13.2.-Opciones de simulación.

Las opciones relativas a la realización de los cálculos durante la simulación se pueden fijar desde dos sitios diferentes. Si nos encontramos en la ventana de configuración de las opciones de animación y pulsamos el botón ‘Opciones spice’ accederemos a los parámetros de simulación para el diseño que nos ocupa. Sin embargo si utilizamos la opción ‘Fijar las opciones de simulación' del menú ‘Sistema’ estaremos fijando estos parámetros como valores por defecto para todas las simulaciones que se realicen con PROTEUS.

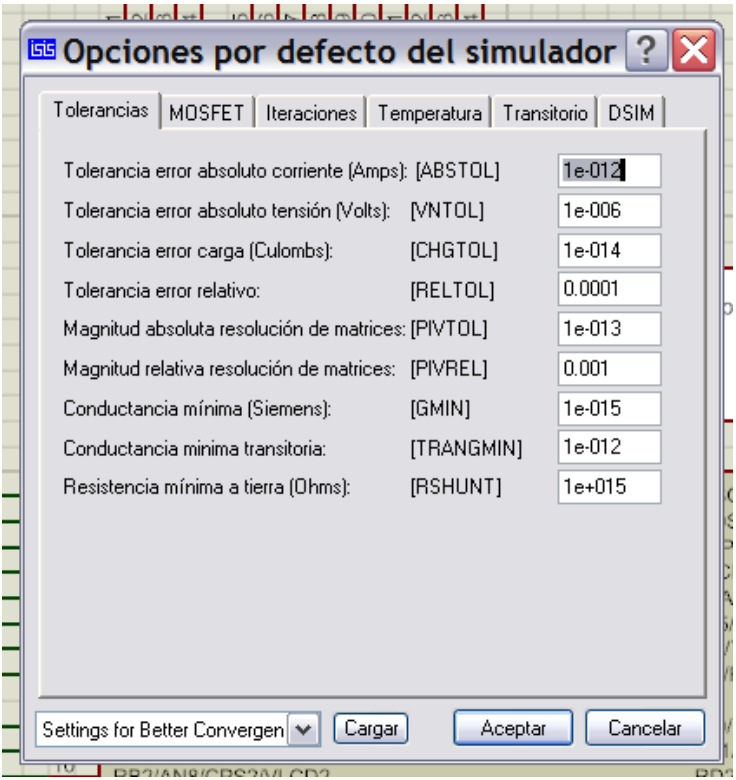


Para entender las opciones de simulación hay que comprender cómo funciona una simulación. La simulación se realiza mediante la utilización de algoritmos de aproximaciones sucesivas mediante la ejecución de iteraciones. Cuando el resultado obtenido en una de las ejecuciones del bucle varía con respecto al obtenido en la ejecución anterior del bucle por debajo de una cantidad definida, se considera alcanzado el resultado y se detiene la simulación.

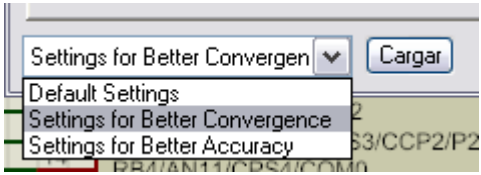
Si el bucle se ejecuta un número predeterminado de veces sin alcanzar el resultado esperado (una variación menor de la cantidad definida) la simulación se considera imposible y se aborta generándose los mensajes de error correspondientes.

Por eso la configuración consiste en jugar con los diferentes valores posibles de números de iteraciones, precisiones necesarias, temperaturas, etc. para logar la mayor exactitud posible en los resultados con las posibilidades que nos brinda el ordenador personal en el que estamos llevando a cabo la simulación.

Los parámetros disponibles se agrupan en varios apartados: tolerancias, componentes MOSFET, iteraciones, temperaturas, transitorios y simulación de componentes digitales (DSIM).



Antes de abordar el significado de los parámetros más importantes, conviene fijar nuestra atención en una caja de selección desplegable situada en la zona inferior de la ventana de diálogo.



Con ella podemos seleccionar tres configuraciones de los valores por defecto para tres supuestos posibles: configuración para lograr una simulación con un equilibrio entre obtención de resultados de convergencia y exactitud en los datos obtenidos (Default Settings), configuración para lograr una simulación orientada a obtener los

mejores resultados de convergencia sacrificando exactitud (Settings for Better Convergence) y configuración para lograr la máxima exactitud en los resultados obtenidos (Settings for Better Accuracy). Una vez que seleccionemos el conjunto de parámetros debemos pulsar sobre el botón 'Cargar' para que Proteus seleccione los valores de los parámetros según nuestra elección.

Del apartado tolerancias destacamos los siguientes parámetros.



'Tolerancia error absoluto corriente'[8] es el valor que debe variar la corriente entre una etapa del bucle y la siguiente para que la convergencia sea aceptada.

'Tolerancia error absoluto tensión' es el valor que debe variar la tensión entre una etapa del bucle y la siguiente para que la convergencia sea aceptada.

'Tolerancia error carga' es el valor que debe variar la carga entre una etapa del bucle y la siguiente para que la convergencia sea aceptada.

8 Todos los parámetros terminan con el nombre que recibe esta propiedad en el motor de simulación SPICE. Se puede, de esta manera, ampliar la información de cada parámetro acudiendo a libros especializados sobre simulación SPICE.

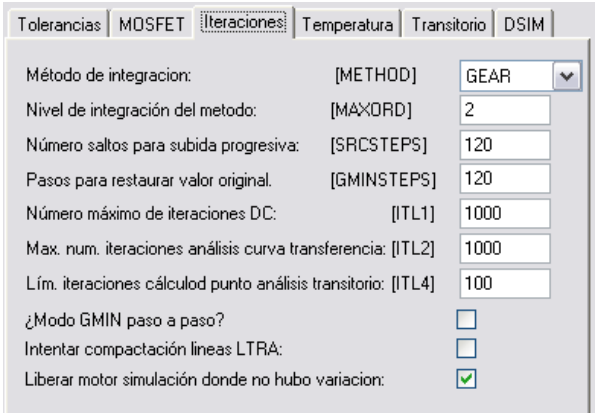
‘Tolerancia error relativo’ es el umbral, en términos relativos, que debe variar la corriente, el voltaje y la carga entre una etapa del bucle y la siguiente para que la convergencia sea aceptada.

‘Conductancia mínima (GMIN)’ es la conductancia mínima permitida por el programa. Reducir este valor puede ayudar a encontrar la convergencia en simulaciones que abortan. Pero utilizar valores mayores de $1e^{-9}$ puede generar resultados de dudosa fiabilidad.

‘Conductancia mínima transitoria’ es un valor de convergencia específico del algoritmo utilizado por Labcenter que permite incrementar el valor de GMIN de forma dinámica para obtener mejores resultados en la convergencia de los análisis de transitorios.

‘Resistencia mínima a tierra’ es el valor de la mínima resistencia a tierra válida que se establece para todos los nodos presentes en el circuito. En ocasiones será necesario añadir a alguno de los nodos de nuestro circuito una resistencia de puesta a tierra para lograr cumplir esta exigencia.

Del apartado iteraciones destacamos los siguientes parámetros.



‘Método de integración’ determina el método utilizado para realizar la integración. El valor “trapezoidal” se ofrece para lograr compatibilidad con versiones anteriores. El valor recomendado es “gear” que ofrece mayor exactitud con menos iteraciones.

‘Nivel de integración del método’ especifica el nivel de integración del método utilizado. Con el método “gear” se puede seleccionar valores más altos de 2.

‘Número de saltos para subida progresiva’. Todas las fuentes de alimentación utilizadas en nuestro diseño son inicializadas durante el arranque de la simulación desde el valor 0 hasta el valor especificado, en una subida en forma de rampa. Este parámetro especifica el número de saltos en que se producirá esta subida progresiva.

‘Pasos para restaurar valor original’. El valor del parámetro GMIN se reduce hasta alcanzar la convergencia durante la simulación. Cuando lo hace se vuelve a incrementar a su valor original. El número de pasos en que se produce esta restauración del valor original viene definido por este parámetro.

‘Número máximo de iteraciones DC’ especifica el máximo número de iteraciones que se llevarán a cabo en un punto.

‘Max. num. Iteraciones análisis curva transferencia’ especifica el máximo número de iteraciones que se llevarán a cabo en un análisis de curva de transferencia.

‘Lim. Iteraciones cálculo punto análisis transitorio’ especifica el máximo número de iteraciones que se llevarán a cabo en un análisis de transitorios.

‘Modo GMIN paso a paso’ especifica si deseamos adoptar desde el principio el modo “gmin paso a paso” que se utiliza cuando se produce un fallo de convergencia en el punto de operación.

‘Liberar motor simulación donde no hubo variación’ permite una optimización del proceso de simulación porque libera al motor de simulación de re-calcular los valores de los dispositivos semiconductores en aquellos puntos donde el voltaje no ha variado desde la última evaluación.

Detallar el significado del resto de los parámetros desborda el alcance de este tutorial. Podemos encontrar la información referente a todos ellos utilizando la ayuda contextual que nos brinda PROTEUS.

14.-Utilización de modelos SPICE en la simulación.

14.1.-Conceptos básicos sobre modelos SPICE.

La mayoría de los fabricantes de componentes electrónicos proporcionan modelos compatibles SPICE con sus productos. Puesto que el motor PROSPICE que utiliza PROTEUS está basado y es compatible con el código original SPICE de la Universidad de Berkeley, no tendríamos que tener mayores problemas en poder utilizar estos modelos en nuestros diseños y usarlos dentro de nuestra simulación. Bien es verdad que hay algunos puntos importantes a tener en cuenta antes de utilizar modelos suministrados por terceras partes.

Un modelo SPICE es un fichero en código ASCII con su listado de conexiones e información de su comportamiento electrónico. No contiene ninguna información de tipo gráfica y no es susceptible, por lo tanto, de ser utilizado directamente en uno de nuestros diseños electrónicos realizados con la herramienta PROTEUS. Así que, para usarlos, necesitamos tanto la representación gráfica del componente de la librería como el modelo SPICE.

Desgraciadamente, no existen formatos estándar con la información gráfica para los componentes, así que tendremos, en la mayoría de los casos, que dibujar nuestro componente con las herramientas disponibles en PROTEUS y luego asociarlo con el modelo SPICE que hayamos descargado desde la red.

Un modelo SPICE se utiliza referenciándolo como un sub-circuito dentro de nuestro diseño usando una ficha para definirlo. La unión de los diferentes nodos del circuito (el listado de conexiones SPICE) con los nodos del componente (los terminales de nuestro dibujo) se determina según el orden en que los nodos del circuito aparecen en la ficha. Por ejemplo la ficha de un cierto modelo denominado “XU1” podría ser la siguiente:

```
XU1  46  43  32
```

Lo que significa que el nodo 1 del modelo llamado XU1 se unirá con el nodo 46 de nuestro diseño, el nodo 2 del modelo con el nodo 43 del diseño y el nodo 3 del modelo con el nodo 32 del diseño.

Desgraciadamente, este sistema significa que los distintos nodos de un modelo no tienen nombre. Para empeorar las cosas, raramente hay una correspondencia entre el número de nodos de un modelo y el número de patillas físicas del componente al que representa el modelo, puesto que muchas veces el empaquetado tiene patillas sin utilizar que casi nunca se tienen en cuenta en la enumeración de nodos del modelo SPICE. Las complicaciones pueden ser incluso mayores cuando se trata de componentes multi-dispositivo como, por ejemplo, el caso de un amplificador operacional TL074 que puede contener dentro del mismo encapsulado hasta cuatro amplificadores operacionales de igual funcionalidad.

En la práctica, todas estas consideraciones se traducen en la necesidad de establecer algún sistema de referencia explícita para indicarle a PROTEUS qué nodo del modelo SPICE se corresponde con qué patilla de la representación gráfica que vamos a utilizar, puesto que, por lo que hemos venido explicando, no hay forma de utilizar ni el nombre ni el número de la patilla. Con objeto de facilitarnos la tarea se ha incluido en el estándar SPICE una nueva propiedad denominada SPICEPINS.

Por si todo lo dicho no fuera suficientemente complejo, en la industria electrónica se utilizan diferentes variantes del estándar SPICE. El denominador común a todas ellas se denomina SPICE2 y lo habitual es que la mayoría de los modelos estén escritos de acuerdo con él. La última versión publicada del estándar es la SPICE3F5. Pero un número significativo de fabricantes declaran que los modelos que facilitan son compatibles con PSPICE™ una variante propietaria de la versión SPICE2 desarrollada por una firma comercial. Muchas de las características contempladas por el estándar SPICE3F5 no son soportadas por el estándar PSPICE™ y algunas de las primitivas de PSPICE™ son diferentes y utilizan sintaxis diferentes para características similares que han sido añadidas en ambos estándares.

El resumen final de todo esto es que modelos especificados para PSPICE™ pueden no funcionar con el motor PROSPICE de PROTEUS. Como receta práctica, cada vez que vayamos a

descargar un modelo SPICE desde la página web de un fabricante debemos comprobar si está escrito para SPICE2 o SPICE3.

Las buenas noticias son que LABCENTER proporciona sus librerías con un importante número de componentes (más de 1.500) con los modelos suministrados por los propios fabricantes y que ya incluyen su correspondiente representación gráfica, listos para usarlos en PROTEUS.

14.2. Enlazado de un modelo SPICE con un componente del esquema.

Para llevar a cabo un ejemplo práctico, en este tutorial vamos a utilizar el amplificador operacional LF411 que se encuentra en la librería TEXOAC suministrada con PROTEUS. El fichero con el modelo SPICE lo podemos encontrar en la web de NATIONAL en el siguiente enlace:

http://www.ti.com/general/docs/lit/getliterature.tsp?literatureNumber=sloj040&fileType=zip[9]

Habitualmente, para los ficheros que contienen modelos SPICE se utilizan las extensiones .TXT, .CIR, .LIB, .MOD. Pero esto no es una regla fija ni una condición obligatoria con lo que nos podemos encontrar con extensiones diferentes.

El código completo del modelo que nos ocupa se muestra a continuación:

```
*////////////////////////////////////////////
//////////////////////////
* (C) National Semiconductor, Inc.
* Models developed and under copyright by:
* National Semiconductor, Inc.
```

9 Desgraciadamente, internet es un 'animal vivo' que está cambiando continuamente, por lo que es posible que cuando usted esté leyendo este tutorial el enlace no funcione correctamente. En este caso, es una buena opción utilizar el buscador google con la cadena 'spice model LF411' y actualizar el enlace. El fichero a descargar puede ser ligeramente diferente (por ejemplo, no contener tantos comentarios).

```
*////////////////////////////////////////////
////////////////////////
* Legal Notice: This material is intended for
free software support.
* The file may be copied, and distributed;
however, reselling the
*  material is illegal

*////////////////////////////////////////////
////////////////////////
* For ordering or technical information on
these models, contact:
* National Semiconductor's Customer Response
Center
*                 7:00 A.M.--7:00 P.M.  U.S.
Central Time
*                                (800) 272-
9959
* For Applications support, contact the
Internet address:
*  amps-apps@galaxy.nsc.com

*
////////////////////////////////////////////
////////////////////
* User Notes:
*
* 1. Input resistance (Rin) for these JFET op
amps is 1TOhm.  Rin is
*    modeled by assuming the option
GMIN=1TOhm.  If a different (non-
*    default) GMIN value is needed, users may
recalculate as follows:
*    Rin=(R1||GMIN+R2||GMIN), where R1=R2,
*    to maintain a consistent Rin model.

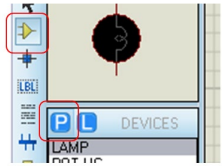
*
////////////////////////////////////////////
////////////////////
*////////////////////////////////////////////
////////////
*LF411 LOW OFFSET, LOW DRIFT JFET INPUT OP-AMP
MACRO-MODEL
```

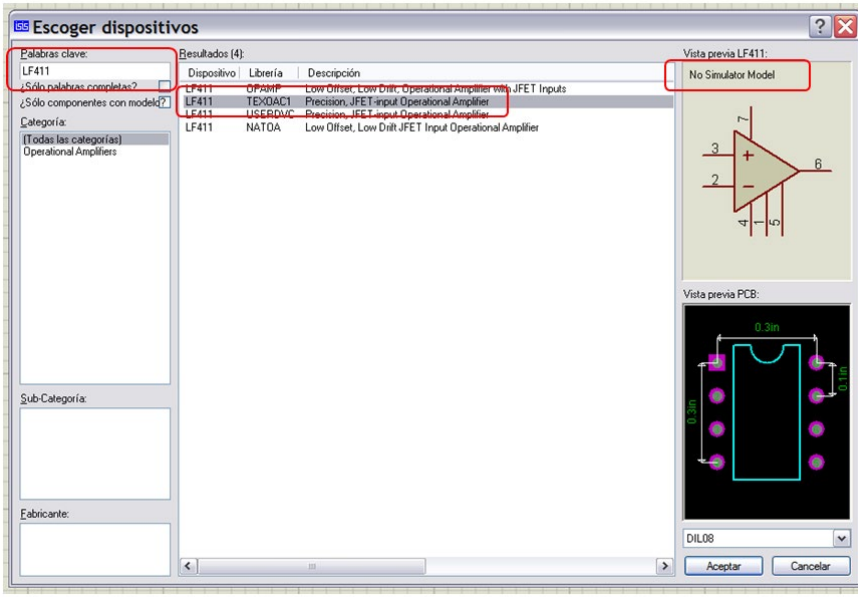
```
*////////////////////////////////////////////
/////////////
*
* connections:    non-inverting input
*                 |   inverting input
*                 |   |   positive power
supply
*                 |   |   |   negative power
supply
*                 |   |   |   |   output
*                 |   |   |   |   |
*                 |   |   |   |   |
.SUBCKT LF411/NS  1   2  99  50  28
*
*Features:
*Fast settling time (.01%) =           2uS
*High bandwidth =                     3MHz
*High slew rate =                    10V/uS
*Low offset voltage =                  .5mV
*Low supply current =                 1.8mA
*
***************INPUT STAGE*************
*
IOS 2 1 25.0P
*^Input offset current
CI1 1 0 3P
CI2 2 0 3P
R1 1 3 1E12
R2 3 2 1E12
I1 99 4 1.0M
J1 5 2 4 JX
J2 6 7 4 JX
R3 5 50 650
R4 6 50 650
*Fp2=28 MHZ
C4 5 6 4.372P
*
***********COMMON MODE EFFECT***********
*
I2 99 50 800UA
*^Quiescent supply current
EOS 7 1 POLY(1) 16 49 .8E-3 1
*Input offset voltage.^
R8 99 49 80K
```

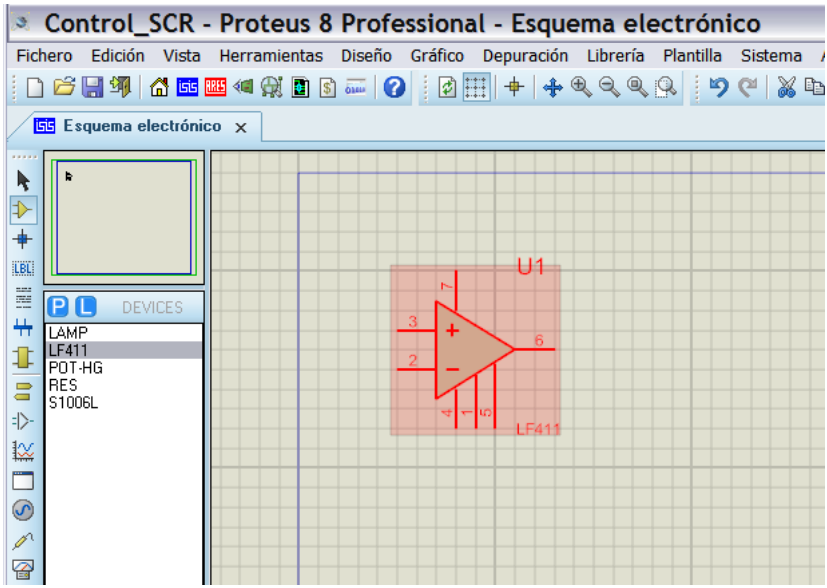
```
R9 49 50 80K
*
*********OUTPUT VOLTAGE LIMITING********
V2 99 8 2.13
D1 9 8 DX
D2 10 9 DX
V3 10 50 2.13
*
**************SECOND STAGE**************
*
EH 99 98 99 49 1
G1 98 9 5 6 20E-3
R5 98 9 10MEG
VA3 9 11 0
*Fp1=18 HZ
C3 98 11 857.516P
*
***************POLE STAGE***************
*
*Fp=30 MHz
G3 98 15 9 49 1E-6
R12 98 15 1MEG
C5 98 15 5.305E-15
*
*********COMMON-MODE ZERO STAGE*********
*
G4 98 16 3 49 1E-8
L2 98 17 144.7M
R13 17 16 1K
*
**************OUTPUT STAGE**************
*
F6  99 50 VA7 1
F5  99 23 VA8 1
D5  21 23 DX
VA7 99 21 0
D6  23 99 DX
E1  99 26 99 15 1
VA8 26 27 0
R16 27 28 50
V5  28 25 0.646V
D4  25 15 DX
V4  24 28 0.646V
D3  15 24 DX
```

```
*
***************MODELS USED**************
*
.MODEL DX D(IS=1E-15)
.MODEL JX PJF(BETA=1.183E-3 VTO=-.65 IS=50E-
12)
*
.ENDS
*$
```

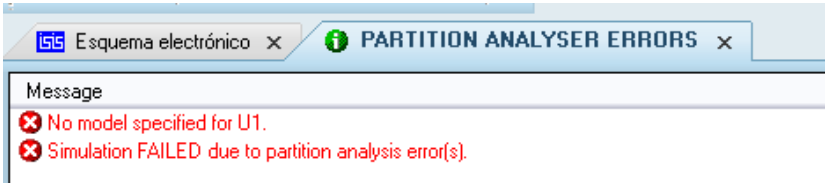
Empezaremos nuestro trabajo creando un nuevo diseño y colocando en el área de trabajo un componente LF411. Para ello seleccionaremos el icono ‘Modo componente’ de la barra de herramientas lateral y utilizaremos el botón ‘P’ del selector de objetos para buscar nuestro componente LF411 desde la librería TEXOAC (observe que en la ventana de vista previa indica que no tiene modelo de simulación).







Una vez que lo hemos colocado vamos a ejecutar la simulación con el botón PLAY de los controles de simulación. Nos aparecerá la ventana mostrando los siguientes errores en el proceso.

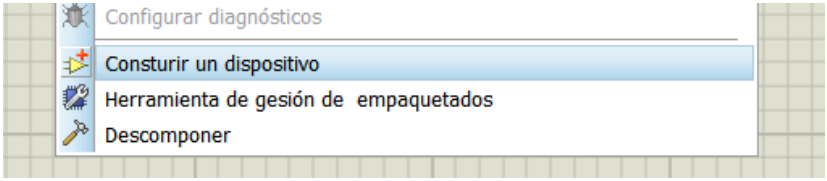


Como era de esperar, PROTEUS nos informa que el componente U1 que acabamos de colocar en nuestro diseño no tiene asociado ningún modelo para poder llevar a cabo la simulación.

Vamos a enlazar el modelo que hemos obtenido de la red con nuestro dispositivo. Lo primero que haremos es cerrar la pestaña con la relación de mensajes de error. A continuación colocaremos el fichero que hemos descargado en el directorio MODELS de la instalación de PROTEUS.

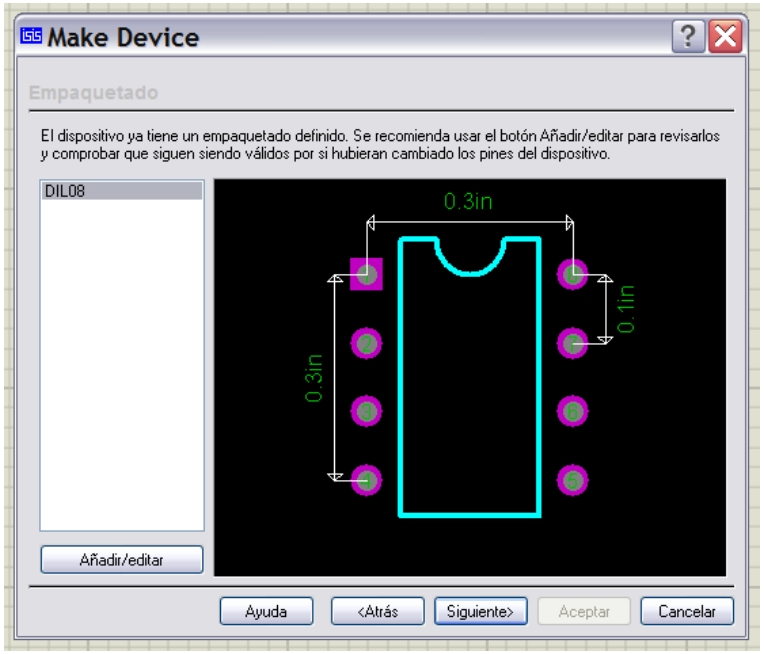
El siguiente paso será colocar el cursor del ratón sobre el componentes y pulsar el botón derecho para abrir el menú contextual y seleccionar la opción ‘Construir un dispositivo’ para abrir la ventana de diálogo correspondiente

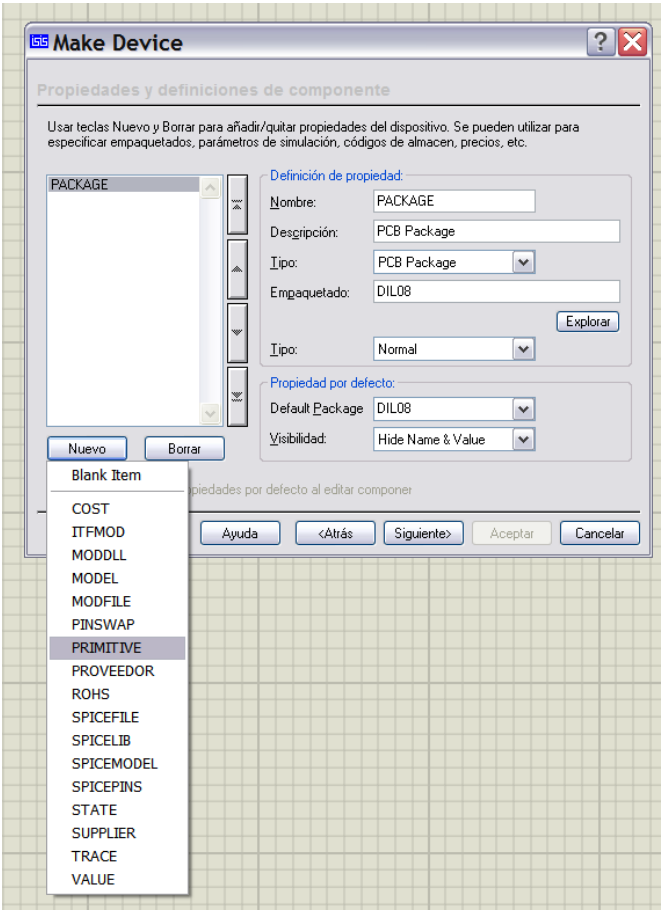
.



Los datos de lass dos primeras pantallas están bien como están y sólo tendremos que pulsar el botón ‘siguiente’ para ir hasta la tercera.




Make Device
Empaquetado
El dispositivo ya tiene un empaquetado definido. Se recomienda usar el botón Añadir/editar para revisarlos y comprobar que siguen siendo válidos por si hubieran cambiado los pines del dispositivo.
DIL08
Añadir/editar
0.3in
0.3in
0.1in
Ayuda
<Atrás
Siguiente>
Aceptar
Cancelar



En esta ventana tenemos que añadir las propiedades que el motor PROSPICE necesita para poder simular el modelo SPICE. Necesitamos añadir cuatro propiedades: PRIMITIVE, SPICEFILE, SPICEMODEL y SPICEPINS.

Para añadir una propiedad utilizaremos el botón ‘Nuevo’ y seleccionaremos desde la lista desplegable que aparece la opción que deseemos.

14.2.1.-La propiedad primitive.

Empezaremos por la propiedad PRIMITIVE cuyo valor tiene que ser 'analog,subckt'. Con ello le estamos indicando al motor que es un modelo analógico formado por un sub-circuito. Es importante que tengamos en cuenta que se escribe en minúsculas y que no hay espacios antes y después de la coma. Si no lo hacemos así la simulación abortará.



14.2.2.-La propiedad spicemodel.

La propiedad SPICEMODEL se utiliza para especificar el nombre del modelo que figura en el propio fichero del modelo SPICE que vamos a utilizar. Para conocer qué nombre se utiliza en el fichero, podemos abrirlo con un editor de texto y buscar la cadena ".subckt". En la línea donde aparece este literal, encontraremos que detrás hay un espacio y el nombre del modelo.

En nuestro caso:

```
.SUBCKT LF411C  1   2   3   4    5
```

LF411C.301

```
1  * LF411C OPERATIONAL AMPLIFIER "MACROMODEL" SUBCIRCUIT
2  * CREATED USING PARTS RELEASE 4.01 ON 06/27/89 AT 08:19
3  * (REV N/A)      SUPPLY VOLTAGE: +/-15V
4  * CONNECTIONS:   NON-INVERTING INPUT
5  *                | INVERTING INPUT
6  *                | | POSITIVE POWER SUPPLY
7  *                | | | NEGATIVE POWER SUPPLY
8  *                | | | | OUTPUT
9  *                | | | | |
10 .SUBCKT LF411C   1 2 3 4 5
11 *
12   C1   11 12 3.498E-12
13   C2    6  7 15.00E-12
14   DC    5 53 DX
```

Así que el valor que introduciremos para la propiedad SPICEMODEL será LF411C.



14.2.3.-La propiedad spicefile.

Con esta propiedad le indicamos al motor en qué fichero se encuentra la información referente a nuestro modelo SPICE. Debemos recordar que el fichero debe estar situado en el directorio MODELS de PROTEUS. Si PROTEUS no es capaz de encontrar el fichero en este directorio, lo buscará en el directorio actual de trabajo (donde se encuentra nuestro diseño. Y, si tampoco lo encuentra ahí, la simulación abortará.

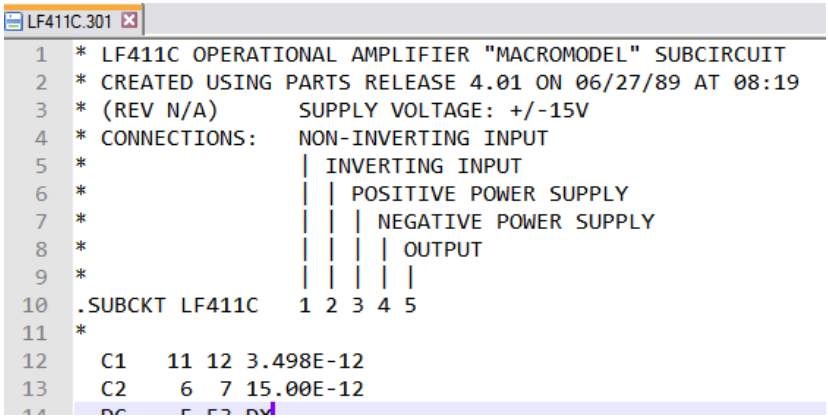


14.2.4.-La propiedad spicepins.

Este punto resulta absolutamente imprescindible para que la simulación pueda ser llevada a cabo de forma correcta. Esta propiedad es una lista de los nombres de las patillas utilizadas en el esquema gráfico que colocamos en ISIS, pero en el orden en que se encuentran definidos en el fichero con el modelo SPICE.

En la mayoría de los amplificadores operacionales el orden es positive input, negative input, positive power, negative power, output. Y en el diagrama gráfico utilizado en ISIS las patillas reciben los nombres de +IP (positive input), -IP (negative input), V+ (positive power), V- (negative power) y OP (output).

En el diagrama gráfico del modelo LF411 de TEXOAC también encontramos otras dos patillas adicionales: BAL1 y BAL2. Como estas patillas no están modeladas en el fichero las vamos a ignorar. Y, debemos fijarnos, que en este caso la patillas reciben nombres ligeramente diferentes: +IP, -IP, VCC+, VCC- y OP. El orden en que debemos introducirlos es el que se fija en el fichero con el modelo.



Ya tenemos toda la información que necesitamos, así que ya podemos incluir la información de la propiedad SPICEPINS.

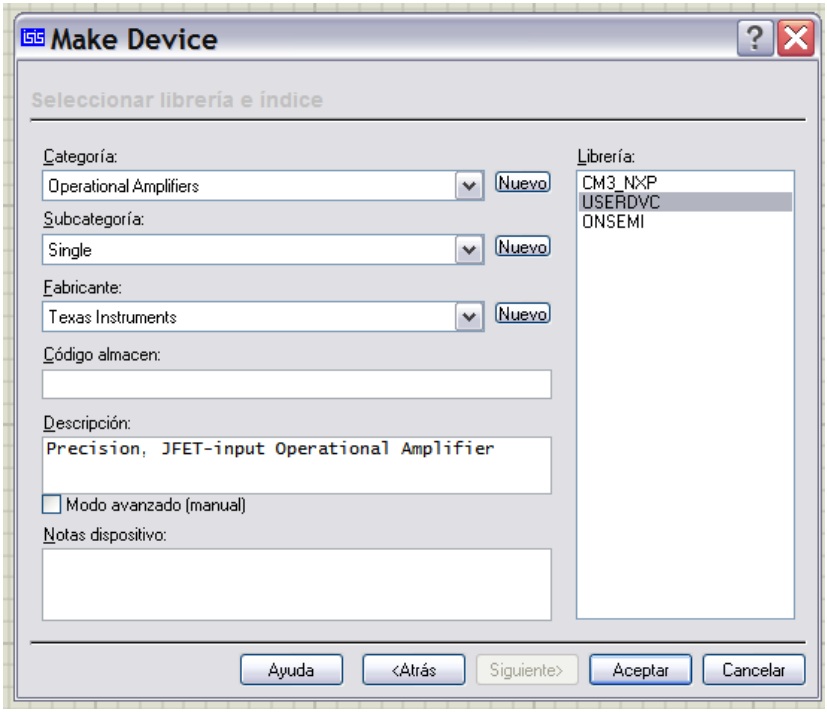
Debemos recordar que no hay que usar espacios antes y detrás de las comas, que debemos escribir los nombres de las patillas como aparecen en la representación gráfica y en el orden que aparecen en el fichero con el modelo SPICE.



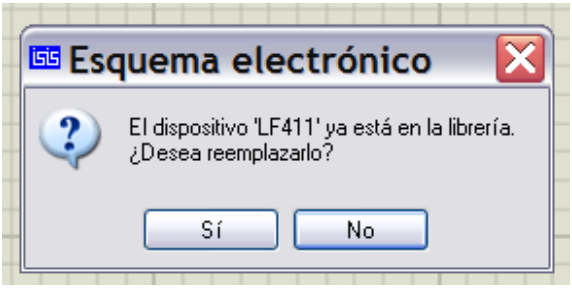
Una vez terminado este punto, podemos pasar a la fase siguiente utilizando el botón ‘Siguiente’. En la pantalla siguiente, dejaremos la información como está.

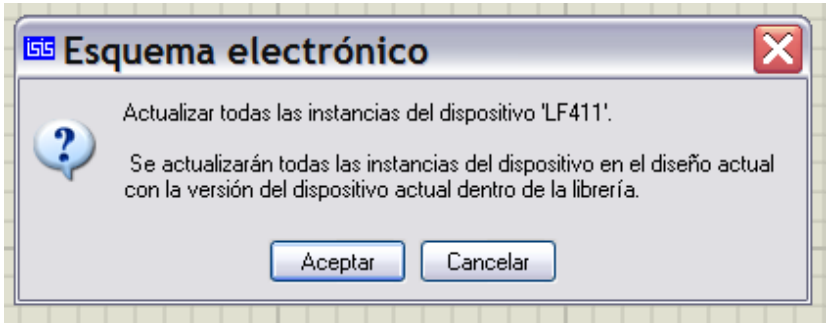


En la última pantalla, seleccionaremos la librería USERDVC (la que define PROTEUS por defecto para almacenar los modelos de usuario) para almacenar este modelo. El resto de parámetros los podemos dejar como están.

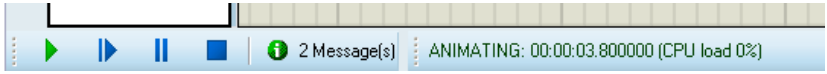


Y finalizaremos nuestra tarea pulsando sobre el botón ‘Aceptar’. Recibiremos un mensaje con una advertencia preguntándonos si deseamos actualizar las instancias de este modelo utilizadas en nuestro diseño. Contestaremos afirmativamente con el botón ‘Sí’.



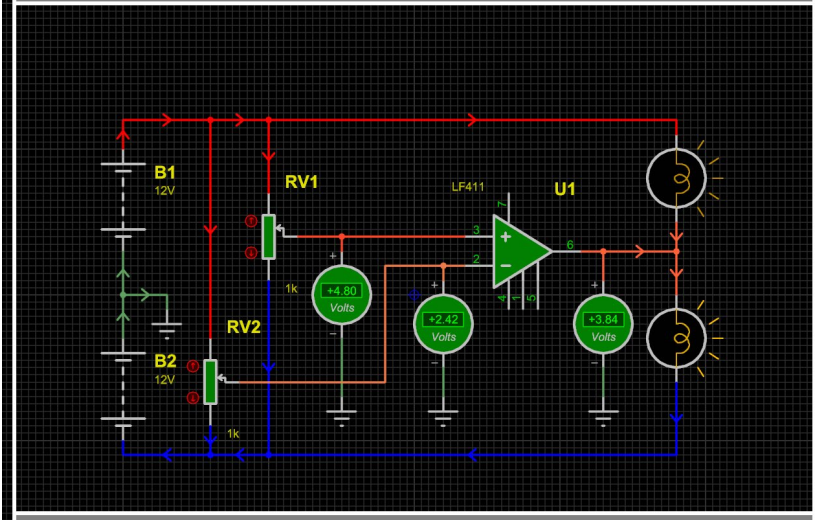


Si todo ha ido bien, ya podemos utilizar nuestro nuevo componente en la simulación. Utilizaremos el botón PLAY de los controles de simulación.



Esta vez la simulación se ejecutará y no se generan errores. En la pantalla siguiente podemos ver un ejemplo utilizando este modelo recién creado.

Operational Amplifiers (1)



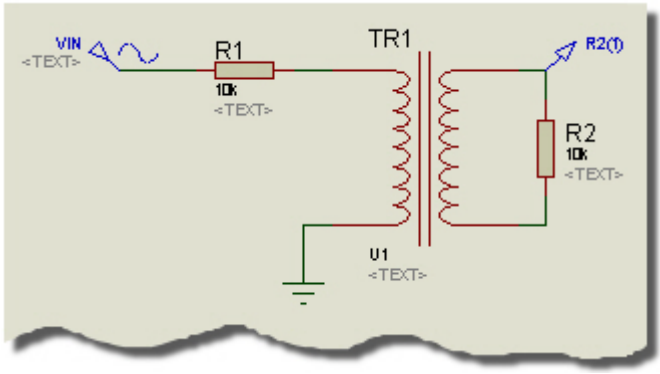
The op-amp compares the voltage at its positive input with the voltage at its negative input. If the difference is postive, the output swings to the positive supply rail and if the difference is negative it swings to the negative supply rail. If the inputs are the same, the op-amp output rests at zero.

15.-Aspectos a tener en cuenta en el diseño para obtener una mejor simulación.

15.1.-Toma de tierra y alimentación.

Todas las simulaciones para funcionar correctamente necesitan que la red de tierra esté perfectamente definida. Es un requisito técnico que todos los componentes de un circuito deben tener un camino a tierra. Si no lo hacemos así, podemos encontrar que una sonda colocada en nuestro circuito mida valores sin aparente sentido, porque en realidad las colocaríamos en partes del circuito flotantes sin valor de referencia.

Por ejemplo en el circuito siguiente, la sonda colocada en un extremo de la resistencia no tendría sentido porque el secundario del transformador está flotante sin referencia a tierra.



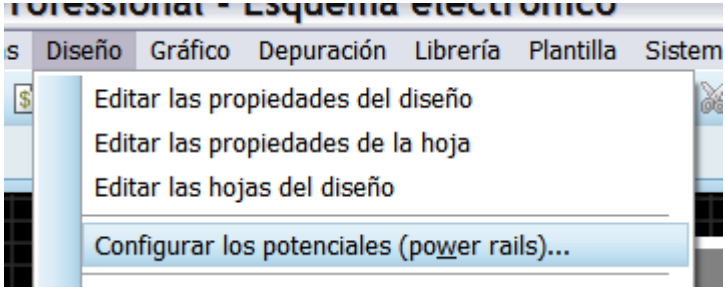
PROTEUS había podido, teóricamente, implementar un polímetro virtual con dos sondas (como los de la vida real) pero existen serías dificultades matemáticas para resolver los cálculos necesarios en el

caso de circuitos que no dispongan de tierra. Y el estándar de la Universidad de Berkeley no lo ha contemplado.

El punto clave en este tema, **es que todas las secciones de circuito de nuestro diseño tienen que tener una conexión a tierra**. La buena noticia es que el motor PROSPICE comprueba esta circunstancia por nosotros y en la mayoría de los casos detecta el problema y genera un mensaje de alerta cuando considera que no se ha cumplido este criterio.

En PROTEUS hay dos maneras de definir que una red de un circuito está conectada a tierra. Una es explícitamente, colocando un terminal de tierra a la red, como en el caso de la figura anterior en el extremo inferior del transformador TR1, o etiquetando un cable con el nombre GND. La otra es implícitamente, utilizando un componente con una patilla oculta conectada a tierra.

La forma de definir en PROTEUS las diferentes tensiones de alimentación que queremos utilizar en nuestro diseño es utilizar la opción 'Configurar los potenciales (power rails)…' del menú 'Diseño'.



Desde su ventana de diálogo podemos crear tantas tensiones diferentes como necesitemos y asignarles el valor deseado. También podemos definir potenciales colocando terminales de alimentación en el esquema y asignándoles el nombre de la tensión que deseamos: +5V, +12V, -12V, -5V, etc. En este caso, el uso de los signos "+" y "-" resulta crítico para que funcione bien.

Por defecto, PROTEUS define los siguientes potenciales. VCC y VDD unidos a +5V. VEE unido a -5V. VSS y GND unidos a tierra. Con estas definiciones por defecto, tenemos la ventaja de que la mayoría de los dispositivos TTL, CMOS y microprocesadores

incluidos en las librerías suministradas con PROTEUS pueden ser simulados sin necesidad de cablear sus patillas de alimentación.

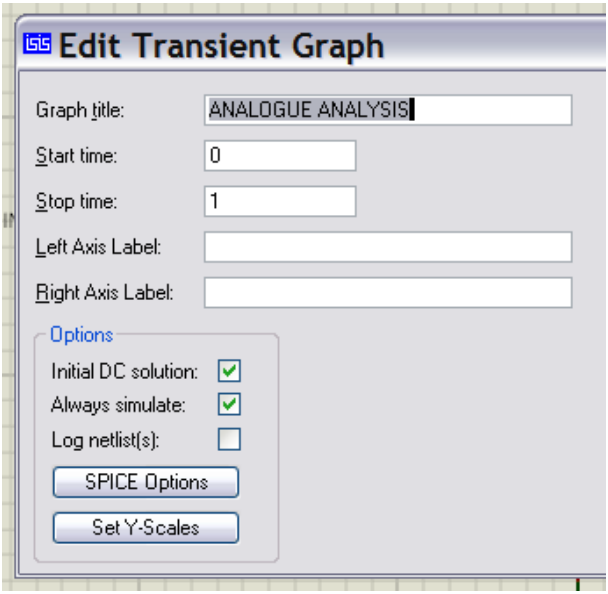
Se puede ampliar la información referente a la utilización de los potenciales y de la puesta a tierra, en la ayuda en línea.

15.2.-Establecimiento de las condiciones iniciales.

Al arrancar la ejecución de la simulación, la primera tarea que ejecuta el motor PROSPICE consiste en el cálculo del punto de operación del circuito. El punto de operación del circuito es el estado estacionario previo a la aplicación de cualquier señal de entrada proveniente de los generadores o potenciales definidos en el diseño. Hay dos modos distintos de ejecutar estos cálculos.

El primer modo consiste en encontrar el punto de operación con todos los condensadores cargados y las bobinas de inducción llenas de flujo. En este caso, el análisis de transitorios mostrará el comportamiento del circuito como si las alimentaciones fueran aplicadas antes del momento inicial con el tiempo necesario para que al arrancar la simulación todas ellas estén estabilizadas. Este es el modo que utiliza el motor PROSPICE por defecto y puesto que PROTEUS permite establecer las condiciones iniciales de cada componente y de cada red que forman el diseño de forma particularizada, sirve para la mayoría de las simulaciones.

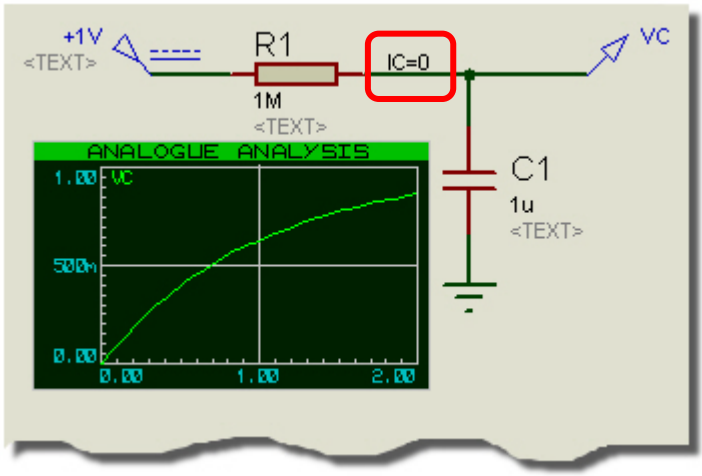
El segundo modo de operación consiste en encontrar el punto de operación con todos los condensadores cortocircuitados y con las bobinas en circuito abierto. En este caso, el análisis de transitorios mostrará el comportamiento del circuito como si las alimentaciones fueran aplicadas justo en el momento inicial. Este modo de operación se selecciona desmarcando la casilla de verificación ‘initial dc solution’ de la ventana de diálogo donde se editan las propiedades de un gráfico.



Tanto en un modo como en el otro, es posible especificar las condiciones iniciales de cada componente y de cada red. Esta posibilidad es muy útil cuando estamos simulando circuitos que contengan osciladores o para aquellos circuitos cuyo funcionamiento dependa de que un determinado condensador se encuentre descargado al iniciar la simulación. Algunas simulaciones pueden fracasar absolutamente si no se establecen correctamente las condiciones iniciales.

15.2.1.-Defínción de las condiciones iniciales para una red.

La forma más sencilla de especificar las condiciones iniciales de una red es especificar su voltaje inicial. En el siguiente circuito definimos la condición inicial de la red, añadiendo al cable de la red donde está ubicada la sonda VC, una etiqueta con el texto “IC=0”



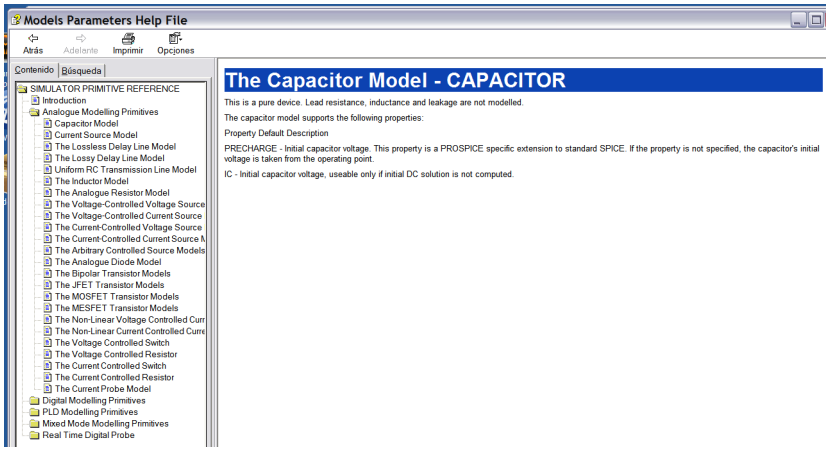
Si no lo hubiéramos hecho así, el motor PROSPICE hubiera calculado el punto de operación del circuito asignando a la red el valor del voltaje de C1, por ejemplo 1 voltio, con lo que la gráfica resultante hubiera sido una línea horizontal plana.

Para las redes que sirvan para conectar componentes digitales se pueden fijar los estados lógicos de inicio con etiquetas con los valores 1, 0, H, L, HIGH, LOW, SHI, WHI, SLO, WLO y FLT asignados a la propiedad BS (boot state).

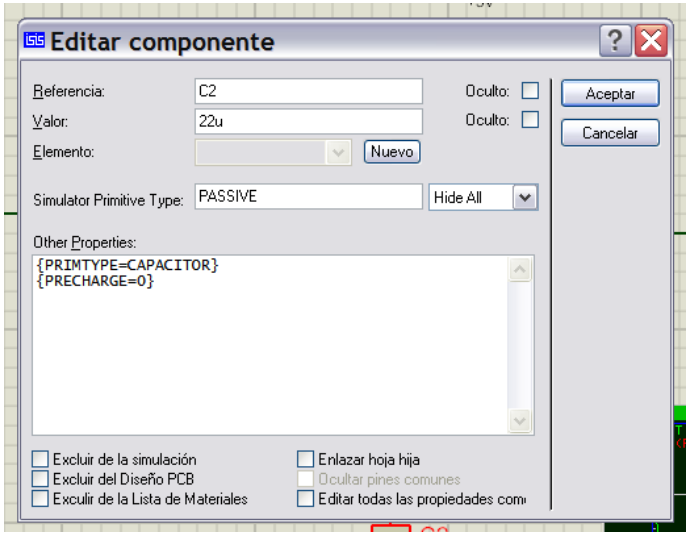
15.2.2.-Definición de las condiciones iniciales para un componente.

Esta posibilidad sólo se puede utilizar cuando la opción 'initial dc solution' no está marcada. En este caso todas las redes tendrán un voltaje igual a cero en el momento inicial, excepto aquellas en que se haya establecido otra cosa como vimos en el epígrafe anterior.

Por ejemplo podemos establecer el voltaje inicial de un condensador escribiendo entre sus propiedades IC=1. Se puede encontrar la información detallada de los parámetros soportados para cada tipo de componente en la ayuda en línea.



Es una lástima que estos valores iniciales no se encuentren disponibles cuando se calcula el punto de operación pero esta es la forma en que se ha normalizado por el estándar de Berkeley. Para solucionar esta dificultad el motor PROSPICE ha añadido una propiedad denominada PRECHARGE que sólo es específica de él y no se incluye en el estándar SPICE3F5. Con la propiedad PRECHARGE podemos indicar el voltaje inicial de cualquier condensador o bobina presentes en el diseño.



15.2.3.-La propiedad NODESET.

En las ocasiones en que una simulación aborta al intentar encontrar el punto de operación se puede solucionar el problema facilitando al motor PROSPICE la utilización de ciertas redes. Esta técnica es diferente de la definición de los valores iniciales. En aquella sólo se fija la condición inicial de una red durante la primera iteración y después la red queda flotante hasta que se encuentre una solución en la que se produce la convergencia. Con esta técnica, fijamos el punto de convergencia y no afectamos a las condiciones iniciales.

El punto de convergencia de una red se establece utilizando la etiqueta NS=<valor> asignada al cable de la red que deseemos estabilizar.

15.3.-Simulación digital.

15.3.1.-Los diferentes estados.

Podría parecer lógico suponer que la simulación digital debería ser capaz de modelar únicamente dos estados: alto (HIGH) y bajo (LOW). Sin embargo, durante la simulación digital se producen situaciones en las que es necesario tomar decisiones para dilucidar, en caso de conflicto, qué estado se debe imponer al otro para reproducir lo que ocurre en la vida real. Por eso el motor PROSPICE define en total nueve estados diferentes para la simulación digital.

ESTADO	CLAVE	DESCRIPCIÓN
Power high	PHI	LOGIC=1, POWER=RAIL
Strong high	SHI	LOGIC=1, ACTIVE OUTPUT
Weak high	WHI	LOGIC=1, PASSIVE OUTPUT
Floating	FLT	FLOATING OUTPUT, HIGH IMPEDANCE
Undefined	WUD	MID VOLTAGE FROM ANALOGUE SOURCE
Contention	CON	MID VOLTAGE FROM DIGITAL CONFLICT
Weak low	WLO	LOGIC=0, PASSIVE OUTPUT
Strong low	SLO	LOGIC=0, ACTIVE OUTPUT
Power low	PLO	LOGIC=0, POWER RAIL

Esencialmente, un estado viene definido por su polaridad (alto, bajo o medio) y su intensidad. La intensidad es una forma de medir la cantidad de corriente que la salida puede proporcionar o absorber, lo cual es especialmente relevante cuando dos o más salidas son conectadas a una misma red.

Por ejemplo si una salida a colector abierto es cableada a través de una resistencia a VCC, cuando la salida es conducida a bajo nivel, en ese punto se producen dos estados al mismo tiempo: un estado WEAK HIGH y un estado STRONG LOW. El estado STRONG LOW se impone al WEAK HIGH y la red se coloca a un estado bajo.

Otro caso interesante se produce cuando dos salidas tri-estado se activan al mismo tiempo en una misma red conduciendo en sentidos opuestos. En este caso ninguna gana y se produce un estado CONTENTION.

Este sistema permite la simulación digital de circuitos con salidas de colector abierto o de emisor abierto y utilización de resistencias tipo PULL-UP. Y también circuitos con salidas tri-estado en oposición unas con otras a través de resistencias.

De todas formas conviene no perder de vista que el simulador digital está pensado únicamente para ser utilizado en la simulación de circuitos digitales y no modela adecuadamente el comportamiento del circuito cuando éste se convierte en uno decididamente analógico. Por ejemplo, conectar resistencias demasiado grandes a entradas de tipo TTL trabajará correctamente en el simulador digital, pero podría fallar cuando construyéramos nuestro circuito en la vida real debido a que es posible que se conduzca una corriente insuficiente para excitar la entrada.

15.3.2.-Estado indefinido.

Cuando una entrada de un modelo digital esté indefinida, su acción repercutirá en el resto del modelo acorde con lo que podríamos decir son reglas de sentido común. Por ejemplo, si una puerta AND tiene una entrada en nivel bajo, la salida se colocará en nivel bajo. Pero si todas menos una están altas y la que no está alta se encuentra en un estado indefinido, entonces la salida se situará en un estado indefinido.

Los dispositivos que se activan por flancos requieren una transición desde el estado lógico 0 al estado lógico 1 (o viceversa). Las transiciones desde el estado lógico 0 a indefinido (o viceversa) y desde el estado 1 a indefinido (o viceversa) no son considerados flancos de subida o de bajada.

Los dispositivos lógicos más complejos (contadores, básculas, etc.) tienen a menudo entradas indefinidas de acuerdo con la codificación de su lógica interna de forma similar a como ocurre en la vida real.

Es frecuente (si no en todos los casos, si con cierta asiduidad) confiar que una entrada TTL no conectada se comportará como si estuviera en un estado lógico 1. Pero no siempre ocurre así. Esta situación puede generarse no sólo cuando un cable no se conecta, sino también cuando la entrada está conectada a una salida tri-estado inactiva.

El motor de simulación digital no puede dejar este aspecto indefinido y tiene que tomar algún tipo de decisión en estos casos, puesto que los modelos internos del dispositivo requieren para evaluar su comportamiento lógico contar con entradas en nivel alto o bajo.

La forma de que el diseñador pueda definir qué solución se toma en estos casos se lleva a cabo utilizando la propiedad FLOAT. Esta propiedad se puede asignar a un componente o a un modelo de librería. Los modelos de las librerías suministradas con PROTEUS de tipo TTL tienen asignada la propiedad FLOAT como HIGH, lo que provoca que las entradas flotantes sean interpretadas como de nivel alto de acuerdo con los supuestos prácticos mencionados anteriormente. Para especificar el comportamiento contrario debemos definir en el componente que nos interese la propiedad FLOAT como LOW.

En el diseño del motor de simulación digital se dedicó mucho tiempo al debate sobre la mejor forma de manejar las simulaciones de modelos sujetos a pulsos muy cortos. El problema fundamental en estos casos es que, bajo este tipo de condiciones, asumir el paradigma más importante de la simulación digital (que los modelos deben presentar un comportamiento puramente digital) conduce a rupturas de la simulación.

Por ejemplo un dispositivo real 7400 excitado con un impulso de 5 nanosegundos genera algún tipo de pulso en su salida, pero no uno

que alcance el nivel eléctrico definido como nivel lógico ALTO en las especificaciones TTL. Si esta salida se utilizara como entrada de reloj de un contador puesto a continuación, esta situación podría suponer un problema para evaluar correctamente la simulación.

La mejor solución consiste en definir claramente los extremos. Así, un pulso de entrada de 1 nanosegundo no logrará excitar la salida correspondiente y un pulso de entrada de 20 nanosegundos lo hará perfectamente. En los puntos intermedios, la puerta no generará la excitación de la salida de forma acorde con los pulsos de entrada y suprimirá lo que podemos considerar como interferencias.

Este supuesto (absolutamente convencional y arbitrario) nos introduce la necesidad de utilizar un nuevo concepto, el de “tiempo umbral de captura” (glitch threshold time) que manejaremos con dos propiedades adicionales para asociar con los modelos digitales: TDLH y TDHL. Determinan el tiempo que debe transcurrir para considerar terminada la transición entre nivel alto a bajo o viceversa.

15.4.-Simulaciones mixtas con modelos analógicos y digitales.

En aquellas ocasiones en que diseñemos un circuito electrónico donde pretendemos realizar una simulación mixta (analógica y digital) utilizando el motor PROSPICE debemos dedicar un poco de tiempo a especificar las características analógicas de la familia de dispositivos digitales.

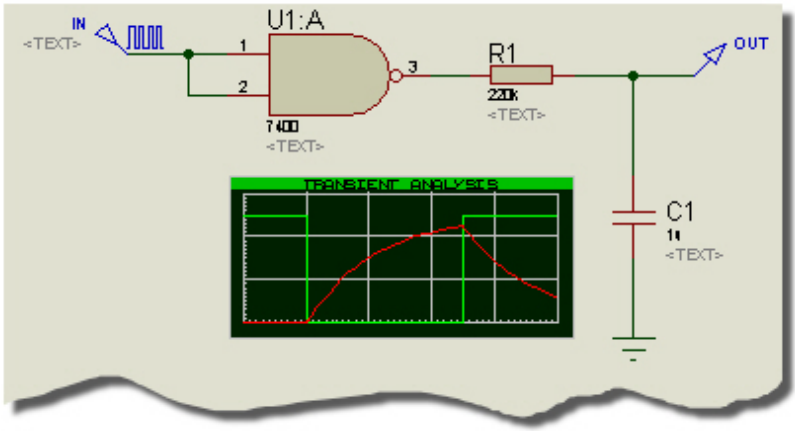
Entre las propiedades que hay que tener en cuenta tenemos:

- La impedancia de entrada y salida de los dispositivos.
- El umbral lógico de las entradas de los dispositivos.
- Los niveles de voltaje para los estados alto y bajo de las salidas.
- Los tiempos que tarda en producirse la subida y la bajada en las salidas de los dispositivos.
- El estado lógico por defecto para una entrada cuando se encuentra flotante.

Un sistema que intentara especificar todos estos parámetros para cada uno de los elementos incluidos en las librerías de dispositivos TTL sería muy complejo de manejar.

Además hay otro problema significativamente complicado (al menos para los que comenzamos a utilizar PROTEUS) es la manera en que se especifican las diferentes alimentaciones.

Mucha gente realiza la simulación de un circuito similar al que se muestra en la imagen siguiente.



El problema surge cuando espera obtener los mismos resultados al volcar el diseño en la vida real y se siente decepcionado al no lograrlo. El problema en este caso es que PROTEUS VSM asume que el 7400 está alimentado con 5V a través de sus pines de alimentación ocultos que están conectados automáticamente a las redes VCC y GND.

A todos estos problemas se les da una respuesta conjunta con la utilización de la propiedad ITFMOD asociada a los componentes. Esta nueva propiedad es muy similar a la propiedad MODEL que utilizamos para proporcionar al motor PROSPICE una referencia a un conjunto de valores con objeto de configurar el comportamiento del dispositivo. Aunque ITFMOD es un poco diferente en tanto que activa un mecanismo especial.

Esencialmente su funcionamiento es el siguiente.

- Para los dispositivos que contiene una propiedad del tipo ITFMOD se realiza una llamada especial durante el establecimiento de la red que especifica los parámetros de control tanto para los objetos ADC como para los objetos DAC

y asigna nombres específicos a los pines por los que se lleva a cabo la alimentación. En el ejemplo de la imagen anterior que estamos estudiando, el dispositivo U1:A tiene definida la propiedad ITFMOD=TTL.

- Una vez que se asignan los nombres de los pines de alimentación (VCC y GND en el caso que nos ocupa), PROTEUS crea una directiva especial y los conecta automáticamente a la fuente de alimentación adecuada a través de estos pines. ISIS asigna nombres a estos objetos que necesita crear, de forma similar a cualquier otro objeto situado en otra hoja del diseño o modelo, así que, en el circuito que nos ocupa, el objeto fuente de alimentación será denominada U1:A_#P.
- Cuando el motor PROSPICE simula un circuito en el modo mixto, crea un objeto ADC y otro DAC y considera entonces cuál tiene que aplicar al objeto en cuestión. En el caso que nos ocupa creará un objeto de tipo DAC con el nombre U1:A_DAC#0000 porque será de esta manera en la que se relacionará la salida del U1:A del dispositivo.
- La ventaja de este sistema es que con este método busca la fuente de alimentación para relacionarse con el objeto que tenga el mismo nombre (en nuestro caso U1:A), con lo que encuentra la fuente U1:A#P. A continuación fuerza a UA1:A_DAC#000 a heredar las propiedades de la fuente UA1:#P según el modelo especificado en la propiedad ITFMOD. Con ello se logra que el modelo funcione con las especificaciones definidas para la familia lógica TTL.
- El interface creado con la fuente de alimentación también conecta el mismo voltaje de alimentación a las partes del dispositivo digital con salidas totem-pole con lo que se pueden modelar de forma mucho más cercana a su comportamiento en la vida real, absorbiendo o generando corriente desde los pines de alimentación del componente.

La propiedad IFMOD puede tomar los siguientes valores válidos.

Property	Description
TTL	Standard TTL (74 Series)
TTLLS	Low Power Schottky TTL (74LS series)
TTLS	Standard Schottky
TTLHC	High Speed CMOS TTL (74HC series)
TTLHCT	High Speed CMOS TTL with TTL outputs (74HCT series)
CMOS	4000 series CMOS
NMOS	Microprocessor type MOS circuits with TTL logic levels
PLD	PLD type MOS circuits

Con esta técnica se posibilita que nuevos dispositivos puedan ser asignados a una determinada familia añadiéndoles simplemente la propiedad ITFMOD=TTL.

Las definiciones asociadas a cada familia se guardan dentro del fichero ITFMOD.MDF que se encuentra dentro del directorio de modelos de la instalación de PROTEUS. Cada definición contiene una o varias de las propiedades necesarias para definirla.

Con este sistema obtenemos una ventaja adicional. Cada una de las propiedades definidas para una familia puede ser reconfigurada de forma sencilla. Así, por ejemplo, si deseamos realizar una simulación para comprobar cómo van a responder los dispositivos de la serie 4000 de nuestro diseño a un tiempo de flanco largo, podemos modificar la propiedad TRISE=10u directamente en este fichero y comprobar los resultados.

15.5.-Consejos para mejorar la velocidad de las simulaciones.

Aunque PROTEUS es capaz de realizar muchas simulaciones interactivas en tiempo real, es evidente que no puede llevarlo a cabo con cualquier tipo de diseño. Por ejemplo, podemos incluir en

nuestro diagrama electrónico un oscilador que funcione a 1Ghz, pero no hay forma de ejecutar su simulación en un ordenador personal que no sea capaz de ejecutar una instrucción en código máquina en tiempos inferiores a 1milisegundo.

En este apartado vamos a intentar dar unas normas para determinar la complejidad real que va a suponer la simulación de nuestro diseño y cómo llevar a cabo optimizaciones en el circuito para maximizar la velocidad de la simulación.

El primer aspecto y el más fundamental es entender bien la diferencia entre realizar una simulación digital o una simulación analógica con el motor PROSPICE. Siempre hay que tener presente que la simulación digital de un circuito es hasta mil veces más rápida que la simulación analógica de ese mismo circuito. Esta es la razón de fondo por la que el motor PROSPICE contiene un simulador digital en su interior. Con ello se puede evitar el consumo de una importante parte de los recursos de cálculo de nuestro ordenador personal.

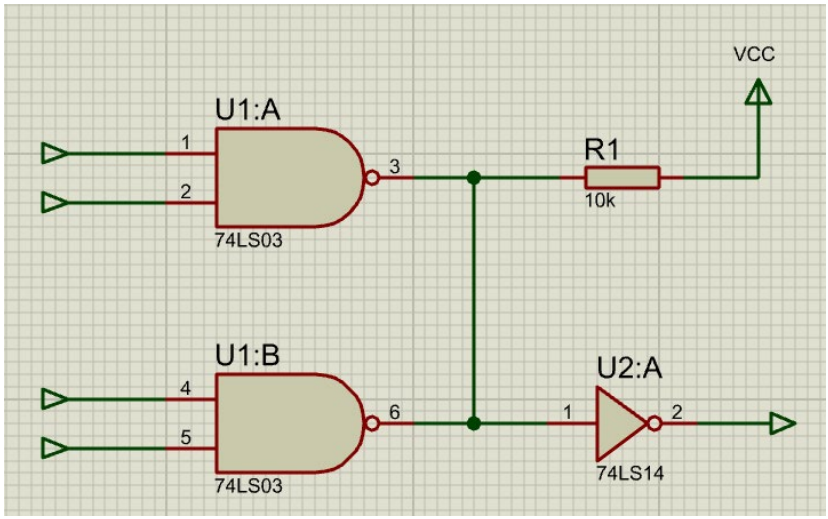
Por ejemplo, mientras que un ordenador personal basado en un procesador P3 a 600MHz puede simular alrededor de dos millones de eventos digitales por segundo, el mismo ordenador sólo será capaz de simular la generación de ondas senoidales de hasta 2kHz antes de que la carga de la CPU alcance el 100% de su capacidad. Para poder modelar una curva senoidal se requiere calcular unos 60.000 puntos de valores analógicos por segundo, y cada uno de esos puntos requiere la obtención de la solución convergente de las ecuaciones en cada nodo del circuito para estabilizarlo. A todas luces, un trabajo muchísimo más complejo que el procesado de cualquier evento digital simple.

Para la mayoría de los componentes electrónicos, debería ser evidente la elección sobre qué tipo de simulación, digital o analógica, es necesaria realizar con él. Por ejemplo, prácticamente todos los equipos TTL y CMOS son representados por modelos de simulación digital y los circuitos integrados como los amplificadores operacionales o los comparadores se representan por modelos de simulación analógica. Además, todos aquellos componentes que se representan por los modelos estándar contemplados en el estándar SPICE se representan con modelos analógicos de simulación.

Existe otra categoría difusa de componentes, que cada día crece más en número, que aunque son de estricta naturaleza analógica, pueden ser representados por un modelo digital con óptimos resultados. En particular, conviene destacar los diodos y, lo que puede resultar más sorprendente aún, las resistencias. Esto es especialmente relevante en el contexto de redes formadas con resistencias de pull-up, dispositivos con salidas de colector abierto, dispositivos cableados en circuitos lógicos OR o conjuntos formados por diodo y resistencia.

15.5.1.-Simulación de circuitos cableados en configuración OR.

Analizaremos el circuito siguiente cableado en una red lógica OR, como ejemplo.



U1:A y U1:B tienen ambos salidas a colector abierto y sólo pueden absorber corriente. Una salida de nivel lógico 1 da como resultado una condición de alta impedancia. En términos del motor de simulación digital, esto significa que las salidas de las puertas lógicas pueden presentar únicamente los estados SLO (strong low) y FLT (flotante).

Ahora supongamos que la resistencia R1 se modela como un dispositivo analógico. En este caso, el motor PROSPICE necesitará insertar un convertidor (virtual) digital a analógico entre las puertas

lógicas y la resistencia y otro convertidor (virtual) analógico a digital entre la resistencia y la entrada del componente U2:A. Con ello obtendremos una simulación hermosamente detallada de las formas de las ondas en los flancos de subida y bajada en ese nodo de nuestro diseño y del flujo de corriente que circulará a través de la resistencia R1. Pero lo conseguiremos a expensas de consumir una ingente cantidad de recursos de cálculo cada vez que la salida de cualquiera de los dos componentes (U1:A o U1:B) cambien de estado.

Podemos evitarlo, indicando que la resistencia R1 se modele de forma digital. En este supuesto su comportamiento pasará de ser simulado como si se encontrara en un estado lógico SHI (el estado lógico de la tensión VCC) a hacerlo como un estado lógico WHI (weak high).

Cuando cualquiera de las dos puertas NAND coloque su salida a un estado lógico SLO (bajo), este estado se impondrá en ese nodo de la red sobre el estado WHI que trata de imponer la resistencia y el nodo se pondrá en estado lógico 1.

Pero cuando ninguna de las dos puertas AND estén absorbiendo corriente (las dos en estado lógico 1) el estado lógico WHI impuesto por la resistencia se impondrá a los estados FLT que tratan de imponer las dos puertas lógicas a su salida y todo el nodo se pondrá en estado lógico 1.

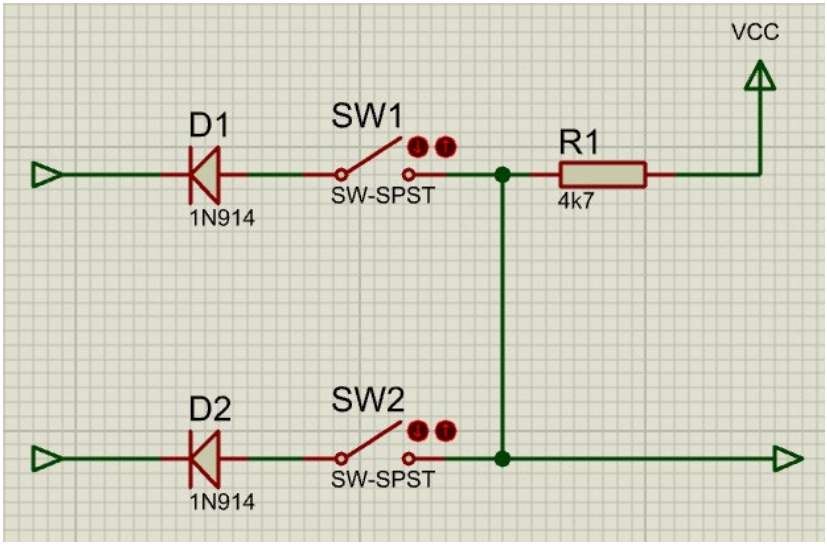
Todo este proceso se puede manejar dentro del paradigma de la simulación digital y sin requerir el consumo de recursos de la simulación analógica.

De forma similar la utilización de resistencias pull-up puede ser modelada digitalmente con el consiguiente ahorro de recursos.

15.5.2.-Simulación de circuitos lógicos con conjuntos diodo-resistencia.

Analicemos ahora un segundo caso donde otro circuito aparentemente analógico puede ser simulado utilizando modelos digitales. Es el caso del circuito representado en la imagen siguiente. Es habitual encontrar este tipo de configuración en circuitos que vigilan teclados donde los diodos realizan una función de prevención

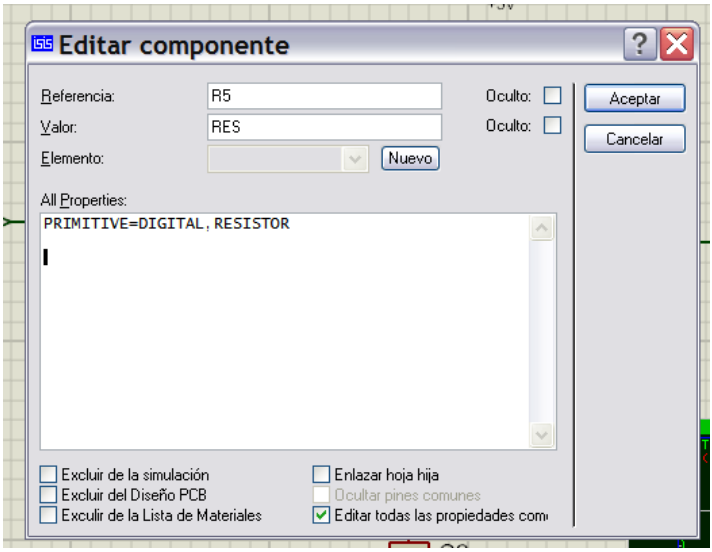
de posibles cortocircuitos entre las diferentes filas de líneas en el caso de que más de una tecla se presione al mismo tiempo.



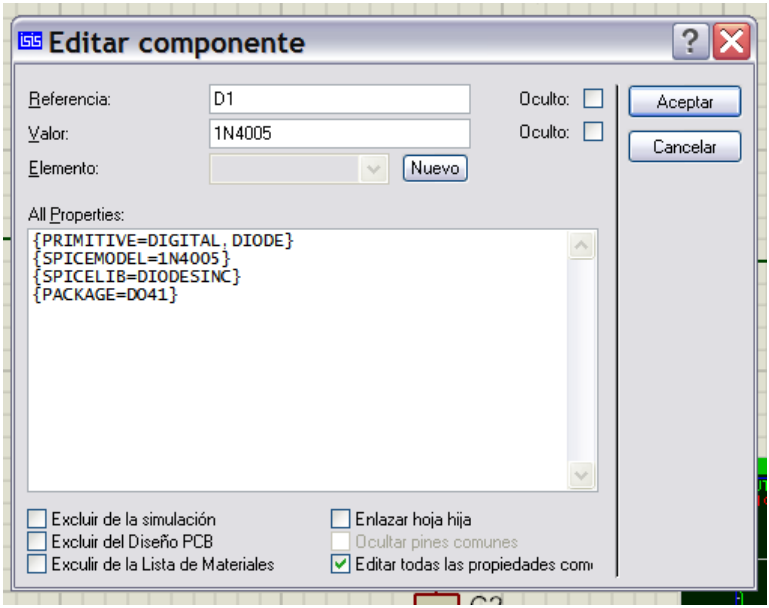
Al igual que en el ejemplo analizado anteriormente, el motor PROSPICE será brillante modelando este circuito dentro de la simulación analógica pero con enorme gasto de recursos de cálculo. Si seleccionamos un modelo digital para la resistencia R1 y los diodos se definen como dispositivo que son sólo capaces de dejar pasar el estado LOW desde el cátodo al ánodo y sólo dejar pasar el estado HIGH del ánodo al cátodo, entonces podremos de esta forma simular todo el circuito dentro del ámbito de la simulación digital.

Puesto que en un teclado operan todos sus sub-circuitos a la misma velocidad y se componen de numerosos diodos y resistencias, la optimización de recursos es tremendamente importante.
Para especificar que una resistencia se configura como un modelo digital, tendremos que activar desde el menú contextual la ventana de edición de sus propiedades.

Marcaremos la casilla ‘Editar todas las propiedades como’ y en el campo ‘All Propierties’ introduciremos la línea:



El mismo procedimiento debemos seguir para hacer otro tanto con los diodos.



16.-Resolución de problemas durante la simulación.

Cada vez que una simulación es ejecutada se genera un fichero histórico. El fichero histórico contiene toda la información y los mensajes de aviso y error que genera tanto el motor PROSPICE en sí mismo como los modelos simulados.

Cuando un mensaje es generado por un modelo en concreto lleva el prefijo con la referencia del componente entre corchetes:

```
[U1] Loaded 26 files from PROGRAM.HEX
```

En todo momento se puede visualizar la ventana donde se muestra el histórico utilizando el atajo de teclado 'CTRL-V'. Cada vez que la simulación aborte, la ventana de depuración se abrirá automáticamente para presentar la información recogida.

Un error se genera como resultado de un problema encontrado por el motor PROSPICE.

16.1.-Errores de simulación.

Los más frecuentes son:

- **Device type not recognized**. Este error se produce cuando hemos especificado un tipo de primitiva no soportado o cuando algún modelo de los que estamos utilizando en la simulación lo está haciendo.

- **No DC path to ground**. Hemos analizado con detenimiento en el epígrafe 15.1 (toma de tierra y potenciales) la necesidad de fijar los caminos a tierra para todos los componentes. Cuando no se hace correctamente se genera este error.

- **Could not find probe**. Este error se produce cuando tratamos de hacer referencia a una sonda o a un generador que no se ha definido.

- **Cannot open SPICE source file**. Este error se produce cuando en una propiedad SPICEMODEL hacemos referencia a un fichero fuente que no se encuentra accesible.

- **Cannot find library model**. Este error se produce cuando hemos llamado a un modelo o 'subckt' que no existe en la librería o en el directorio de trabajo.

- **Model DLL not found**. Este error se produce cuando hemos utilizado un modelo situado en una librería DLL que no existe en los directorios donde debe encontrarlos PROTEUS.

16.2.-Errores de convergencia.

Los más frecuentes son:

- **Singular matrix**. Este error se genera cuando sucede una situación equivalente a la que se produce en la resolución de los sistemas de ecuaciones cuando existen más incógnitas que ecuaciones. Por lo general se produce cuando el circuito está mal diseñado o cuando no se han establecido las condiciones iniciales de forma suficientemente definida. Cuando este error se genera después de otro error del tipo 'DC path to ground', debemos investigar en el cableado alrededor de los pines especificados después de este mensaje.

- **To many iterations without convergence**. Este error significa que no se ha encontrado una solución estable para el circuito. Los circuitos que utilizan las primitivas VSWITCH o CSWITCH pueden generar este error fácilmente, pero también cualquier circuito que incluya funciones de transferencia discontinuas pueden generar el problema.

- **Timestep too small**. Este error se produce cuando el circuito varía de tal manera (es tan nervioso) que el avance de la simulación en un espacio de tiempo, aunque sea en cantidades muy pequeñas del orden de $1e^{-18}$ segundos no

produce un cambio aceptablemente pequeño en los voltajes medidos en el circuito.

La experiencia nos dicta que muchos de los errores producidos durante la simulación se deben:

- A componentes conectados incorrectamente
- A modelos de simulación que no soportan el estándar SPICE2 o SPICE3 correctamente
- A la presencia de osciladores en los que no se han definido correctamente las condiciones iniciales.
- Algunos parámetros que se puede modificar para tratar de solucionar los problemas referentes a la convergencia durante la simulación SPICE son los siguientes.
- Incrementar el valor de GMIN. Por defecto el valor es $1e^{-12}$. Valores que estén por encima de $1e^{-9}$ pueden ser probados.
- Incrementar los valores de ABSTOL y RELTOL para controlar la precisión a alcanzar para validar los resultados.
- En circuitos que utilicen amplificadores operacionales, utilizar la propiedad MODFILE=OA_IDEAL para simplificar la simulación.
- Reducir el valor de TRTOL. Esto solo funcionará si el error se produce debido al análisis de transitorios.