

1

SOFTWARE PARA LA ADQUISICIÓN DE DATOS LabWindows/CVI.

INTRODUCCIÓN

LabWindows/CVI es un entorno de desarrollo integrado para programadores en lenguaje C. Se basa fundamentalmente en un entorno interactivo para desarrollo de programas y una librerías de funciones para crear aplicaciones de adquisición de datos y control de instrumentos. LabWindows/CVI contiene además un conjunto de herramientas software para la adquisición, análisis y presentación.

Para utilizar LabWindows/CVI, el programador no necesita ser un experto conocedor de técnicas avanzadas de programación ni de los detalles internos del sistema operativo Windows.

Como ventajas a versiones anteriores, decir, que LabWindows/CVI contiene la especificación del lenguaje ANSI C, lo que proporciona máxima flexibilidad a la hora de desarrollar aplicaciones.

Las herramientas que básicamente forman este entorno de desarrollo y que iremos describiendo a lo largo de este tema son las siguientes:

- Un editor de interfaces gráficas de usuario (GUI).
- Una ventana para editar el código fuente de nuestros programas ANSI C.
- Paneles de funciones para la ejecución interactiva y generación de código de forma automática.
- Compilador, Reubicador y Depurador integrados para el desarrollo y mantenimiento de proyectos.

La clave de la potencia de LabWindows/CVI está en sus librerías que proporcionan funciones para todas las fases del desarrollo de una aplicación de adquisición de datos o de control de instrumentación. Así tenemos:

1. Cinco librerías para adquisición de datos:
 - **Instrument library:** Esta librería contiene drivers GPIB, VXI y RS232 para instrumentos tales como osciloscopios, multímetros, generadores de funciones etc. Estos drivers consisten en funciones de alto nivel programadas en C que permiten controlar instrumentos por cualquiera de los interfaces comentados (RS232, GPIB, VXI). Se proporcionan además el código fuente de los drivers, con lo que si se desea pueden modificarse para adecuarlos a nuestra aplicaciones.

LabWindows/CVI dispone de todas las herramientas de desarrollo necesarias para crear nuestros propios drivers.

- **GPIB/GPIB 488.2 library:** Esta librería contiene funciones para control de instrumentación conectada a un bus GPIB. Son funciones de muy fácil aprendizaje y uso que hacen transparente al usuario todas las operaciones de bajo nivel que se producen en el bus cuando se realiza cualquier tipo de operación.
- **Data Acquisition library:** Esta librería se proporciona con las tarjetas de adquisición de *National Instruments* y consiste en un conjunto de funciones de alto nivel que permitirá controlar dichas tarjetas.
- **RS232 library.:** Conjunto de funciones de alto nivel para comunicaciones a través del puerto serie.
- **VXI library**

2. Dos librerías para análisis de datos:

- **Formatting and I/O library:** Librería formada por un conjunto de funciones para almacenar y recuperar datos de ficheros y para manipular y formatear los datos de nuestros programas.
- **Advanced Analysis library:** Esta librería consiste en un conjunto de funciones para realizar operaciones matemáticas complejas, generación de señales, operaciones con arrays, operaciones con números complejos, procesamiento de señal, probabilidad etc.

3. Una librería para presentación de datos:

- **User Interface library:** LabWindows/CVI permite crear interfaces gráficas de usuario (GUI) mediante la herramienta User Interface Editor. Todos los objetos utilizados en el interface gráfico de nuestra aplicación (menús pull-down, barras de menús, cajas de diálogos, controles de distintos tipos, gráficos etc.) son almacenados en un fichero con extensión .uir denominado User Interface Resource (fichero *resource*). Las funciones de la User Interface Library permiten cargar el interface gráfico de usuario almacenado en el fichero .uir y visualizarlo, recibir entrada de datos de usuario y visualizar datos y resultados sobre los objetos que forman la interface gráfica de una aplicación determinada.

4. Una librería de utilidades

- **Utility library:** Contiene funciones para realizar distintas operaciones de carácter general: temporizaciones, teclado, utilidades de manejo de ficheros, operaciones de E/S, interrupciones, memoria, sonido, etc.

5. Dos librerías para redes y comunicación entre procesos:

- **Dynamic Data Exchange (DDE) library.**
- **Transmission Control Protocol (TCP) library.**

Además, tal y como se comentó anteriormente, la librería completa ANSI C también está integrada en el entorno de desarrollo de LabWindows/CVI.

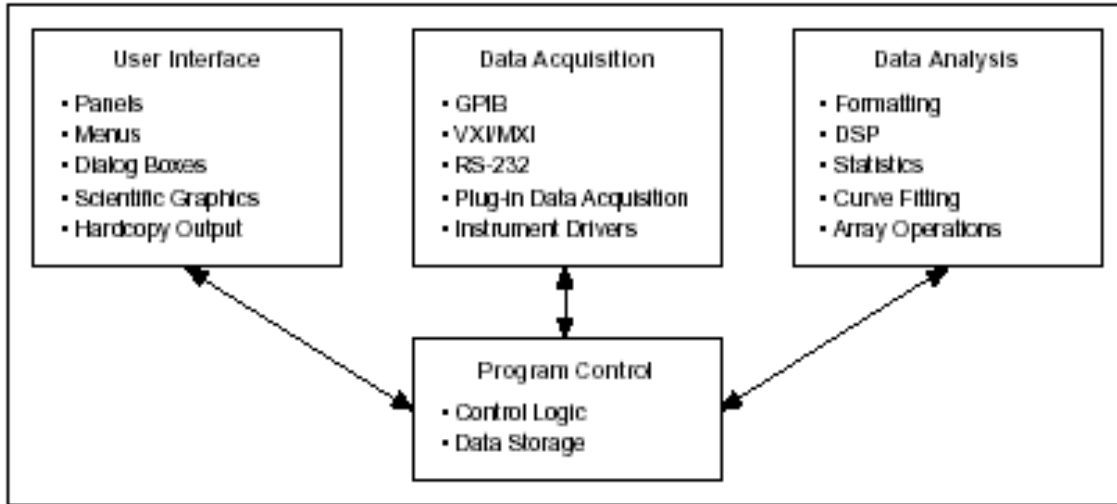
1.- ESTRUCTURA DE UN PROGRAMA EN labWindows/CVI

Una vez realizada la introducción sobre las posibilidades que nos ofrece LabWindows/CVI, pasamos a describir brevemente la estructura genérica de una aplicación LabWindows/CVI.

La mayoría de aplicaciones, incluyen los siguientes elementos:

1. *Interfaz Gráfico de Usuario (GUI) User Interface Editor*
2. *Programa de control (Coordinación adquisición + análisis + GUI)*
3. *Adquisición de datos (Bus externo (GPIB, VXI,RS232,...) / tarjeta adquisición)*
4. *Análisis de datos (Librerías)*

Estos elementos están relacionados tal y como aparece en la figura:



- **User Interface:** Como primer paso para el desarrollo de una aplicación podemos realizar el interface gráfico con el usuario (GUI), utilizando para ello el User Interface Editor.

Los elementos que pueden utilizarse para diseñar el interface de usuario así como las funciones que permiten “conectar” estos elementos al resto de programa serán comentados posteriormente.

- **Programa de control:** El programa de control se encarga de realizar la adquisición de datos bien a través de cualquier bus externo (GPIB, VXI, RS232, LPT, etc) bien a través de una tarjeta de adquisición de datos conectada al bus del PC. Para implementar esta parte del programa utilizaremos las funciones de las librerías.

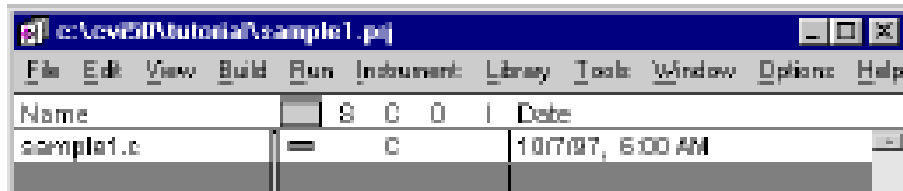
- **Análisis de datos:** Una vez adquiridos los datos habrá que analizarlos, este modulo de programación es el encargado de realizar dicho análisis. En este módulo podemos utilizar lasa funciones incluidas en la librería Formating and I/O library y Advanced Analysis library.

2 EL ENTORNO DE LabWindows/CVI

El entorno de LabWindows/CVI está formado por varias ventanas:

- **Ventana project:** Esta es la ventana que aparece cuando ejecutamos el programa y se utiliza para abrir, editar, construir, ejecutar y salvar aplicaciones completas a las que denominaremos proyectos (project). Cada proyecto está formado por varios ficheros, como son el fichero que contiene el código fuente (.c), fichero de cabecera (.h) que contiene las declaraciones de variables y los prototipos de las funciones utilizadas en el programa, ficheros *resource* (.uir) que contienen

la información del interface gráfico de usuario etc., pues bien, la ventana project nos muestra todos los ficheros que integran dicho proyecto.



Para abrir cualquiera de los ficheros del proyecto únicamente debemos realizar un doble click sobre su nombre.

En esta ventana pueden aparecer seis iconos con el siguiente significado:



Fichero cerrado.



Fichero abierto

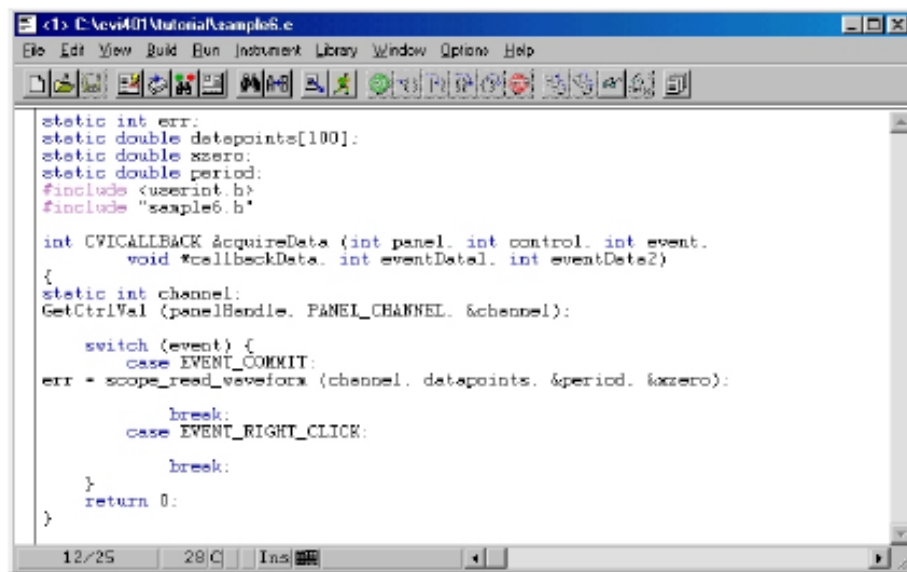
C. Fichero todavía no compilado o modificado desde la última vez que se compiló. Realizando un doble click sobre su nombre el fichero es compilado.

S. Fichero modificado desde la última vez que fue salvado. Realizando un doble click sobre su nombre el fichero es salvado.

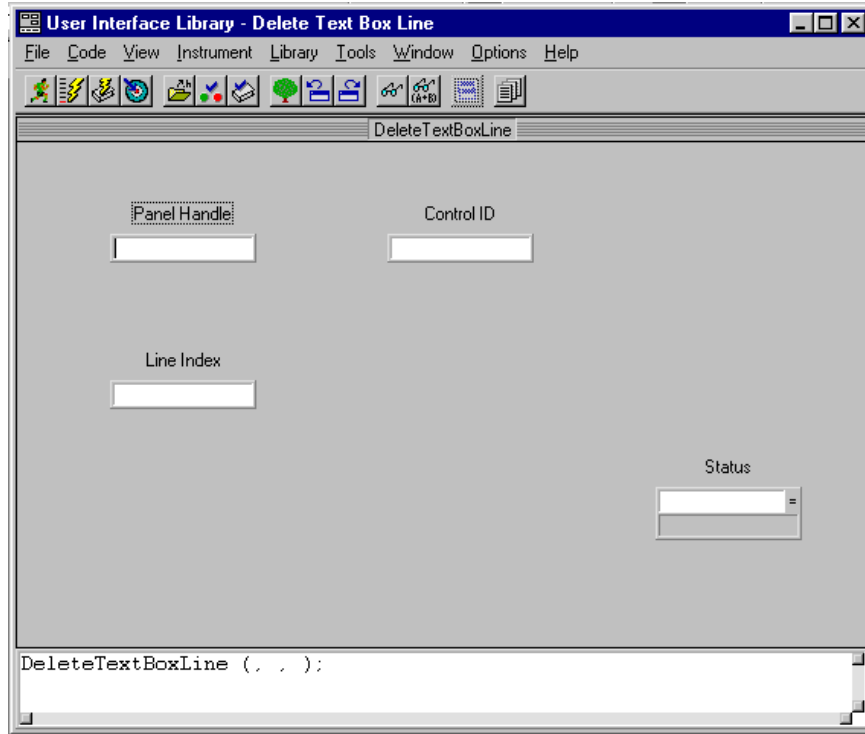
I. Fichero asociado con el driver de un instrumento

O. Se habilita al compilador para crear el código objeto de este fichero

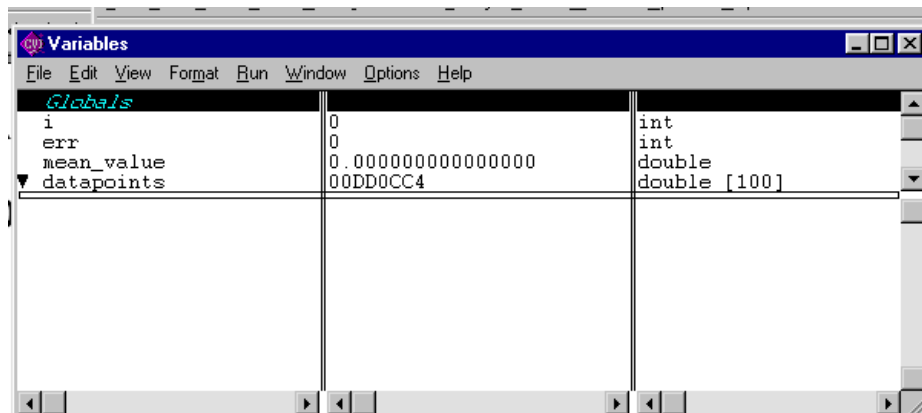
- **Ventana Source:** Utilizada para abrir, crear, editar, ejecutar, depurar y salvar el código fuente de nuestras aplicaciones.

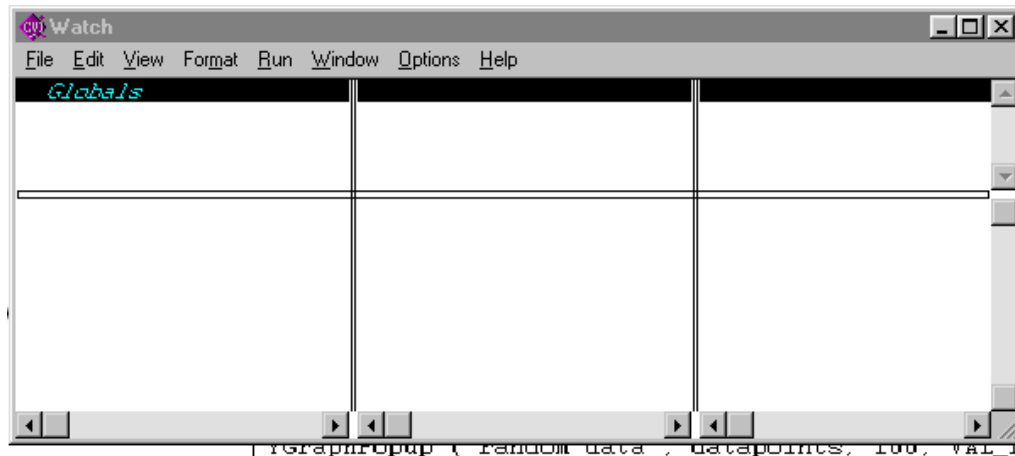


- **Ventana de Function Panel** (Paneles de funciones): Esta ventana se utiliza para ejecutar de forma interactiva las funciones de cualquier librería y para insertar código en el programa abriendo la ventana source. Permite incluso declarar variables de forma interactiva.

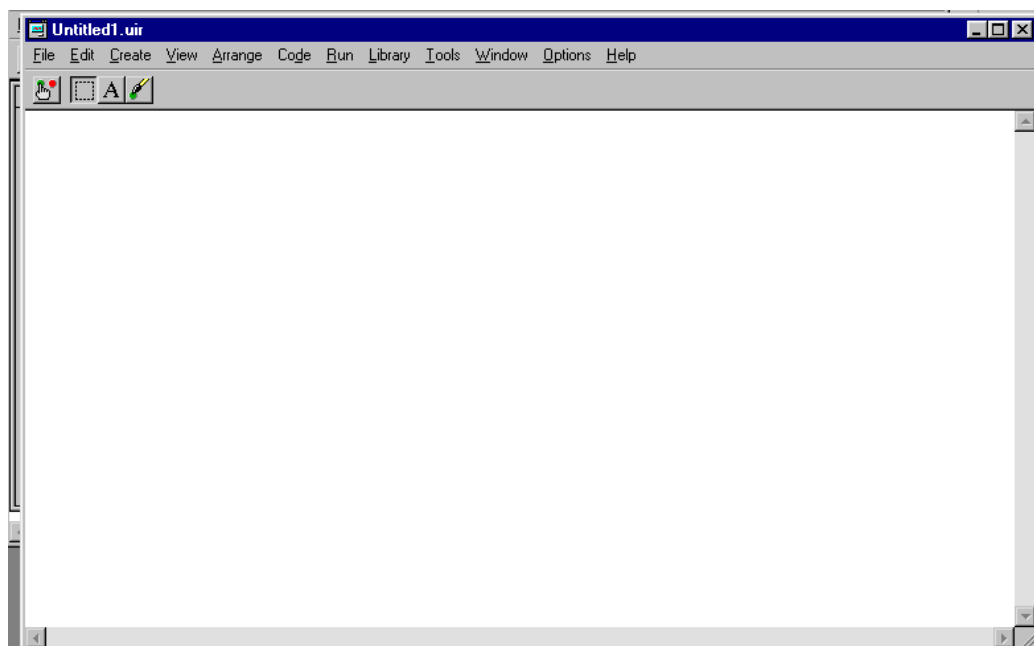


- **Ventana Interactive Execution:** Se utiliza para ejecutar segmentos de código, tiene su utilidad cuando se desea comprobar que cierta parte del programa funciona correctamente.
- **Ventana Variable Display y Watch:** Mediante estas ventanas se puede visualizar el contenido de cualquier variable de nuestro programa. Es muy útil en la fase de depuración.

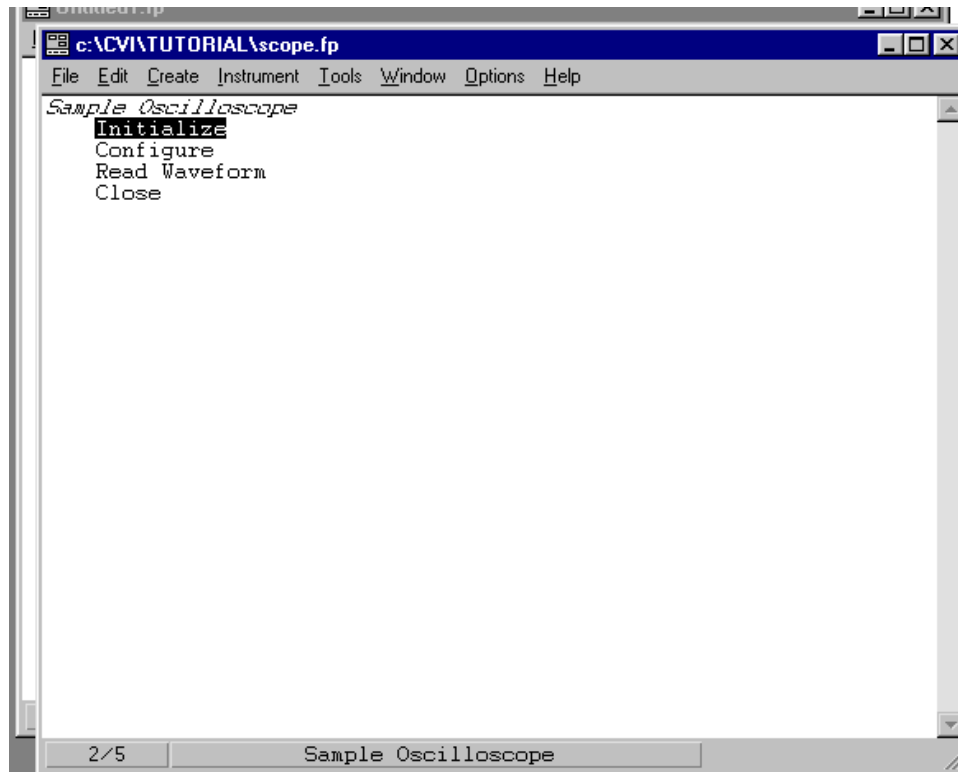




- **Ventana User Interface Editor:** ventana utilizada para editar el interface gráfico de usuario (GUI).



- **Ventana Function Tree:** Esta ventana se utiliza para construir la estructura en árbol de los paneles de funciones



- **Ventana Function panel Editor:** En esta ventana se editan los paneles de las funciones.
- **Ventanas Tree/Help Editor y Function Panel Help Editor:** Utilizadas para añadir ayudas online a los paneles de funciones.
- **Ventana Standst Input/Output:** Utilizada para mostrar mensajes y resultados de los programas y recibir los datos de entrada de usuario.

2

DESARROLLO DE UNA APLICACIÓN CON LabWindows/CVI.

INTRODUCCION

Después de comentar brevemente las posibilidades que nos ofrece LabWindows/CVI, el siguiente paso sería ejecutar LabWindows/CVI y comenzar a diseñar nuestra propia aplicación para ir familiarizándonos con el entorno y con el manejo de cada herramienta.

La aplicación que vamos a desarrollar consistirá en una simulación de un sistema de adquisición de datos. Para simular los datos adquiridos utilizaremos una función que genere datos aleatoriamente, los datos generados se almacenarán en un array. En principio realizaremos únicamente el programa que genera, almacena y presenta en pantalla los datos generados. En una segunda fase editaremos un interface de usuario que consistirá en varios controles y un gráfico donde se presentarán los datos almacenados en la fase anterior.

Nuestra aplicación estará formada por varios ficheros, un fichero que contiene el código fuente de la aplicación (.c), un fichero que contienen el interface gráfico de usuario (.uir), y un fichero de cabecera (.h) que contiene declaraciones de las constantes y prototipos de las funciones utilizadas por el interface gráfico de usuario. Los tres ficheros constituirán el proyecto (.prj).

Comenzaremos por crear el fichero que contienen el código fuente(.c)

2.1.1 EDITANDO EL CÓDIGO FUENTE DE UN PROGRAMA.

Al ejecutar LabWindows/CVI aparece en pantalla la ventana *project* mostrando el proyecto con el que se estuvo trabajando en la última sesión. Mediante la opción **File** podemos cargar, salvar y crear nuevos ficheros. Se puede abrir cualquier tipo de fichero (proyecto, fuente, cabecera o uir).

Para empezar, si hubiese algún proyecto cargado lo que haremos es descargarlo mediante la opción **New** del menú **File**. Al seleccionar la opción **New**, aparece un submenú indicando que tipo de fichero se desea abrir, en nuestro caso indicaremos que deseamos abrir un nuevo proyecto, para ello seleccionaremos *Project (*.prj)*, nos aparecerá la ventana siguiente:

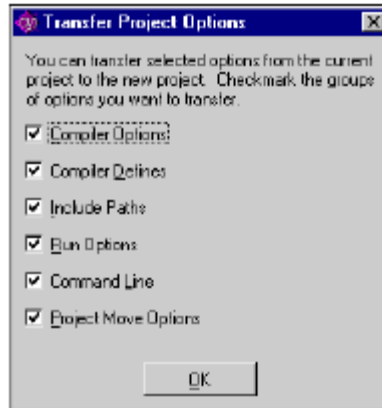


Figure 5-1. Transfer Project Options Dialog Box

Pulsaremos OK para mantener la configuración de LabWindows para el proyecto que vamos a realizar.

Volveremos a realizar la operación anterior pero ahora seleccionando el tipo de fichero *Source (*.c)*

Una vez seleccionado el tipo de fichero que se desea abrir aparece la ventana correspondiente para editarlo, en nuestro caso como hemos abierto un fichero fuente aparece la ventana *source*.

Situados en la ventana de edición de código fuente, editamos el siguiente programa:

```

#include <ansi_c.h>
#include <userint.h>
#include <analysis.h>

int i=0, err;
double mean_value;
double datapoints[100];

int main(int argc, char *argv[])
{
    if (InitCVIRTE (0, argv, 0) == 0)
        return (-1); /* out of memory */

    for (i = 0 ; i < 100 ; i++)
    {
        datapoints[i] = rand()/32768.0;
        printf ("%d \t %f\n", i, datapoints[i]);
    }

    return 0;
}

```

Figure 2-9. Source Code for the sample1.c File

Si analizamos el código del programa, vemos que genera un array de 100 elementos obtenidos de forma aleatoria a través de la función `rand()` y los imprime en pantalla.

Salvamos el programa a un fichero al que denominaremos "*sample1.c*".

Para poder ejecutar el programa debemos , en primer lugar, crear un proyecto. Para ello seleccionamos la opción **Add Files To Project** dentro del menú **Edit** de la propia ventana *source*. Podemos comprobar en la ventana *project*, que ahora nos aparece el nombre del fichero del código fuente que acabamos de añadir. Sólo nos resta salvar el fichero project con el mismo nombre que le dimos al código fuente, *sample1.prj*, ésta operación la debemos realizar desde la ventana project.

Si queremos ejecutar el programa lo haremos desde la opción **Run Project** del menú **Run** de cualquiera de las ventanas.

2.1.2 HERRAMIENTAS DE EDICIÓN

1.- En primer lugar, comentar que el editor incorpora dentro del menú **Edit** la mayoría de características de cualquier editor: **Cut, Copy, Paste, Find, Replace**, etc.

2.- Como sabemos, la mayoría de programas hacen referencia a otros ficheros como pueden ser ficheros de cabecera o ficheros de interface de usuario. Para poder visualizar cualquiera de estos ficheros, situamos el cursor en la línea donde son referenciados y seleccionamos, pulsando el botón derecho, la opción **Open Quoted Text**, dentro de la ventana *source*.

Comprobar el efecto de esta opción intentando visualizar el contenido del fichero ansi_c.h.

3.- Cuando trabajemos con ficheros demasiado largos y deseemos visualizar una zona del código mientras modificamos otra, podemos dividir la ventana en dos subventanas independientes, “pinchando” con el ratón sobre la doble línea que aparece en la parte superior de la ventana de edición.

4.- Para desplazarnos desde cualquier posición del fichero a una línea en particular hay dos métodos:

- Si conocemos el número de la línea al que nos queremos mover, seleccionamos la opción **Line...** del menú **View** y le pasamos el número de línea a la que queremos movernos. Para que aparezca el número de cada línea en el editor, debemos activar la opción **Line Number** del menú **View**.
- Colocando marcas en las líneas a las que deseamos movernos con cierta frecuencia. Para colocar una marca en una línea determinada seleccionamos la opción **Toggle Tag** del menú **View**, cuando se marca una línea aparece una marca de color verde en la columna de la izquierda de la ventana de edición. Para movernos de una marca a otra seleccionamos la opción **Nex Tag** del menú **View**, el cursor pasa de forma inmediata a la línea marcada. Mediante la tecla de función **F2** también podemos saltar entre líneas marcadas.

2.2 GENERACIÓN INTERACTIVA DE CÓDIGO: PANELES DE FUNCIONES.

En éste apartado aprenderemos a utilizar una de las herramientas más potentes de LabWindows/CVI: **los paneles de funciones**. Ésta herramienta simplifica el esfuerzo de desarrollo de forma notable, las características que presenta son las siguientes:

- Ayuda interactiva para indicar como opera la función así como el significado de cada parámetro de ésta.
- Generación automática de código.
- Ejecución interactiva de la función que permite comprobar sus efectos. El poder ejecutar una función de forma interactiva puede sernos de gran ayuda en la fase de depuración de nuestros programas.
- Permite declarar variables utilizadas como parámetros en las funciones para poder ejecutar éstas de forma interactiva.

Para seleccionar una función de alguna de las librerías de LabWindows/CVI, seleccionamos el menú **Library**, apareciendo entonces un submenú en el que aparecen todas las librerías , como podemos ver en la figura:

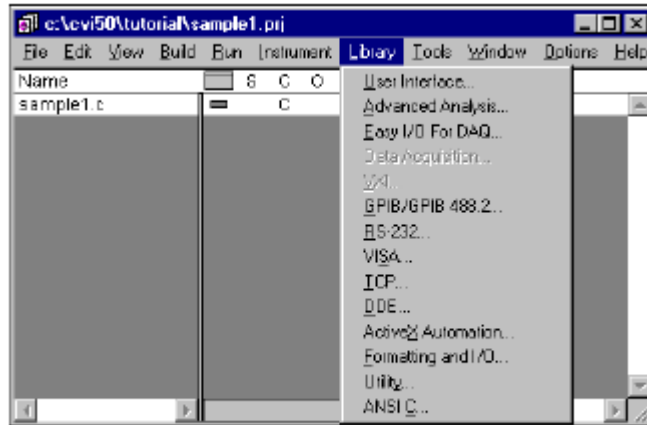


Figure 3-1. Library Menu

Cuando seleccionamos una de las librerías nos aparece un árbol de funciones que representa la jerarquía de la librería. Mediante éste árbol podemos encontrar la función adecuada. En la siguiente figura se muestra, como ejemplo, el árbol de funciones de la librería *User Interface*.

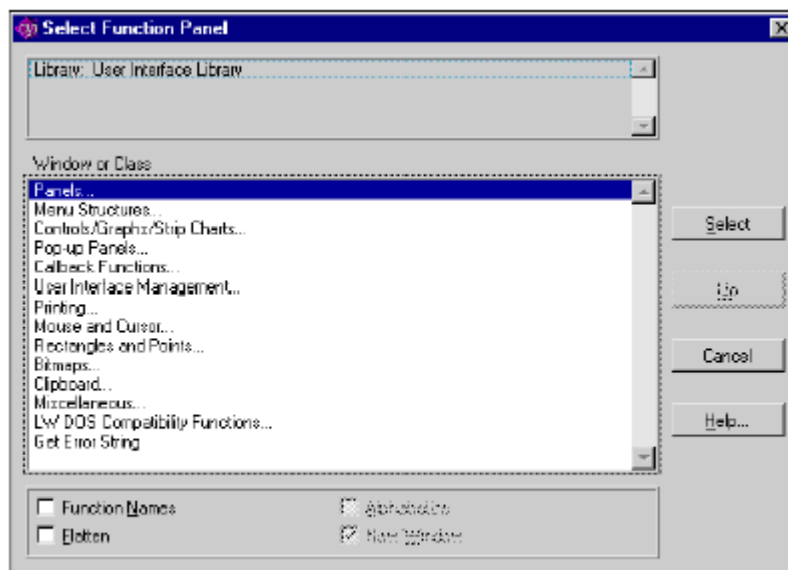


Figure 3-2. User Interface Library Function Tree

Con el fin de comprobar el funcionamiento de los paneles de funciones, vamos a añadir a nuestro programa una función que represente en un gráfico los valores aleatorios almacenados en el array. Para ello nos situamos antes del comando *return 0*, en la última línea del código del programa fuente y seleccionamos la librería **User Interface** del menú **Library**. Una vez en el árbol de funciones de la librería **User Interface** seleccionamos **PoP-up Panels**, nos aparecerá otro árbol de funciones y entonces seleccionamos la función **YGraphPopup**. Una vez seleccionada la función aparecerá en pantalla el panel de la función tal y como aparece en la siguiente figura:

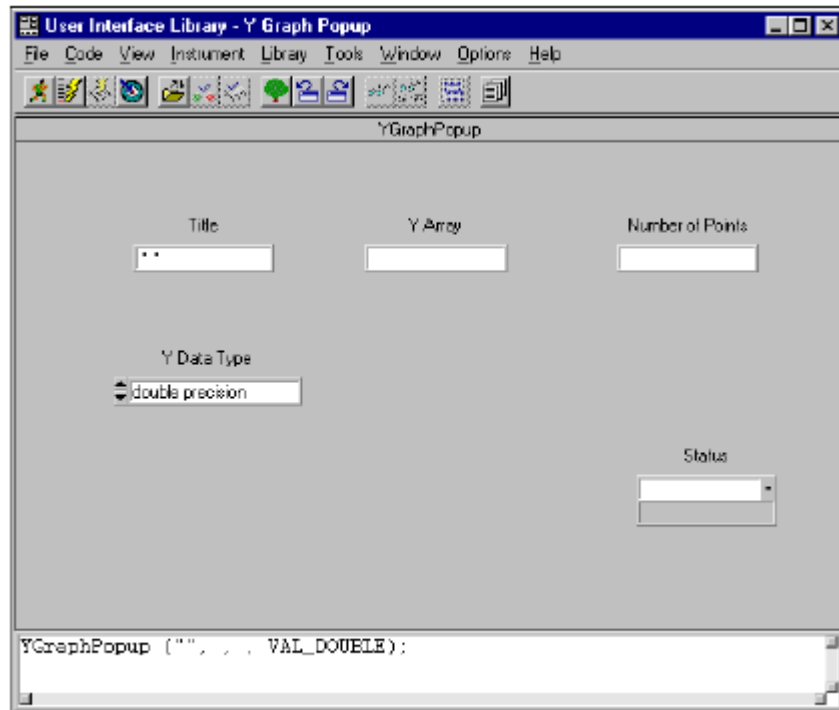


Figure 3-3. Y Graph Popup Function Panel

Como puede observarse en la figura anterior, el panel de una función está formado por una serie de controles, cada control se corresponde con un parámetro de la función. Cada parámetro se edita en su control correspondiente.

Una vez situados en el panel de la función, podemos obtener ayuda sobre ésta o sobre cada uno de los parámetros que utiliza, basta para ello situándonos con el ratón en el control correspondiente y pulsar la tecla derecha del ratón

Completamos los controles de panel de funciones con los siguientes parámetros:

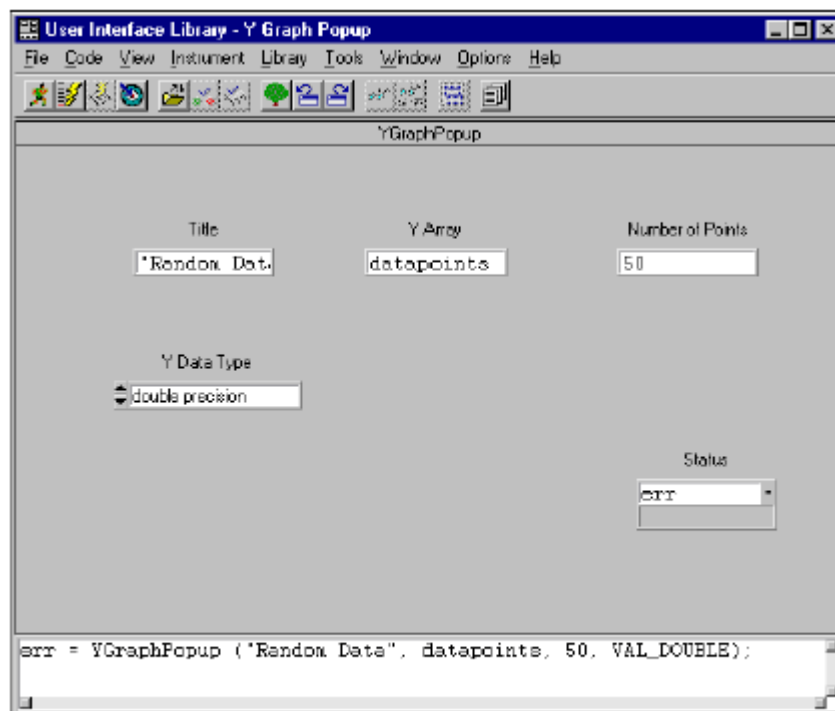


Figure 3-4. Completed Y Graph Popup Function Panel

Observar como a medida que se van rellenando los distintos controles, la ventana que aparece al fondo del panel (ventana de generación de código) donde aparece el código de la función, se va actualizando simultáneamente.

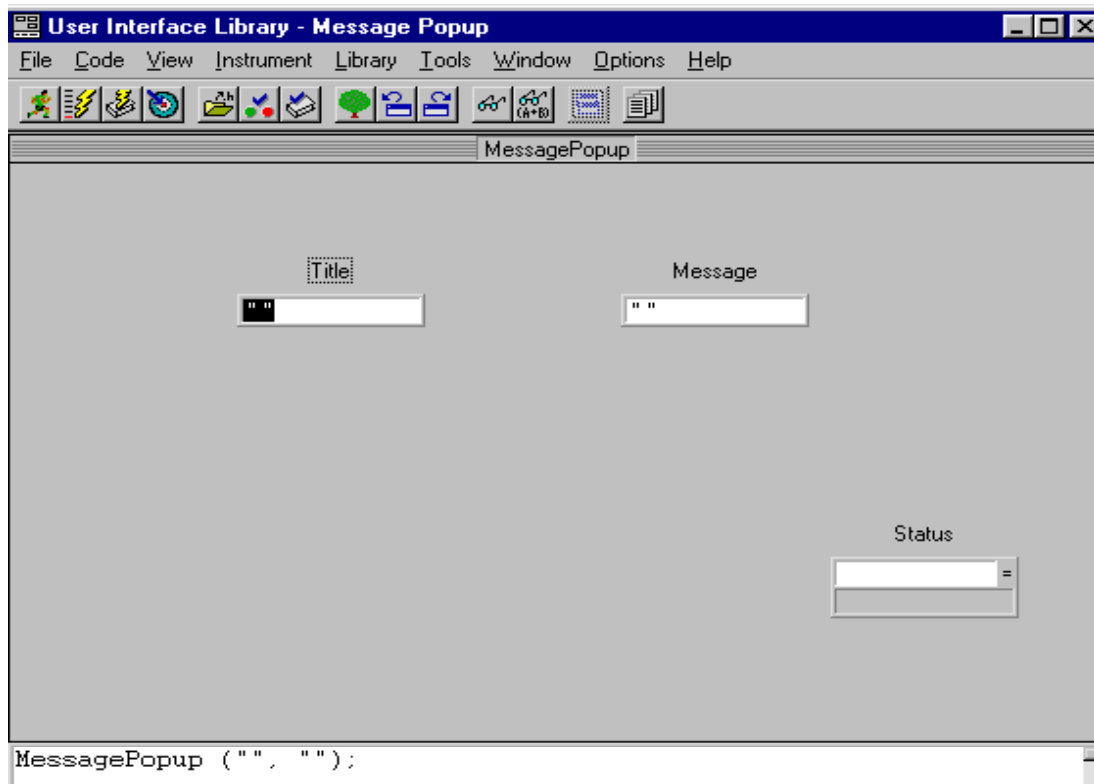
Para trasladar el código de la función a nuestro programa, en primer lugar seleccionamos mediante la opción **Set Target File** del menú **Code** el fichero destino, en nuestro caso el fichero sample1.c. seguidamente insertamos el código de la función en el fichero fuente seleccionando la opción **Insert Function Call** del menú **Code**. Una vez trasladada la función a nuestro programa podemos cerrar la ventana del panel de funciones. Observar que en el código fuente aparece la nueva función añadida.

Ejecutar el programa para comprobar el efecto de la nueva función añadida.

2.2.1 EJECUCIÓN INTERACTIVA DE UN PANEL

Una vez introducidos todos los parámetros de la función en los controles de su panel, podemos ejecutarla de forma interactiva. Para ello seleccionamos la opción **Run Function Panel** del menú **Code**. La función puede ejecutarse tantas veces como se desee, cambiando sus parámetros en cada ejecución hasta que encontremos la combinación que mejor se adecue a nuestras necesidades.

Como ejemplo, ejecutaremos una función de forma interactiva. Seleccionar la librería **User Interface** del menú **Library**, una vez dentro del árbol de funciones seleccionar **Pop-up Panels**, seleccionar la función **Message Popup**, aparecerá entonces su panel de función, completamos sus controles con los siguientes parámetros:

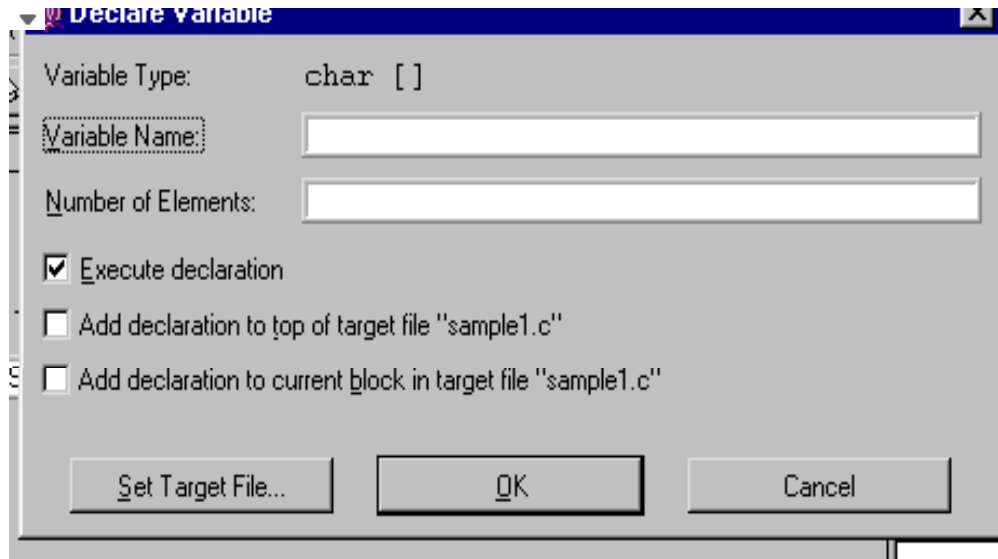


Title: "Mensaje de información"

Message: "Esto es una ejecución interactiva."

Cuando se hayan completado los controles seleccionar la opción **Run Function Panel** del menú **Code**.

Si la función que deseamos ejecutar de forma interactiva necesita algún parámetro variable, podemos declarar estos parámetros de forma inmediata seleccionando la opción **Declare Variable** del menú **Code**, nos aparecerá una ventana como la de la siguiente figura:



Una vez declaradas las variables podemos ejecutar la función

Con el fin de comprobar todo estos, seleccionamos la función **YGraphPopup** de la librería **User Interface**. Cuando aparezca su panel completar los controles con los siguientes parámetros:

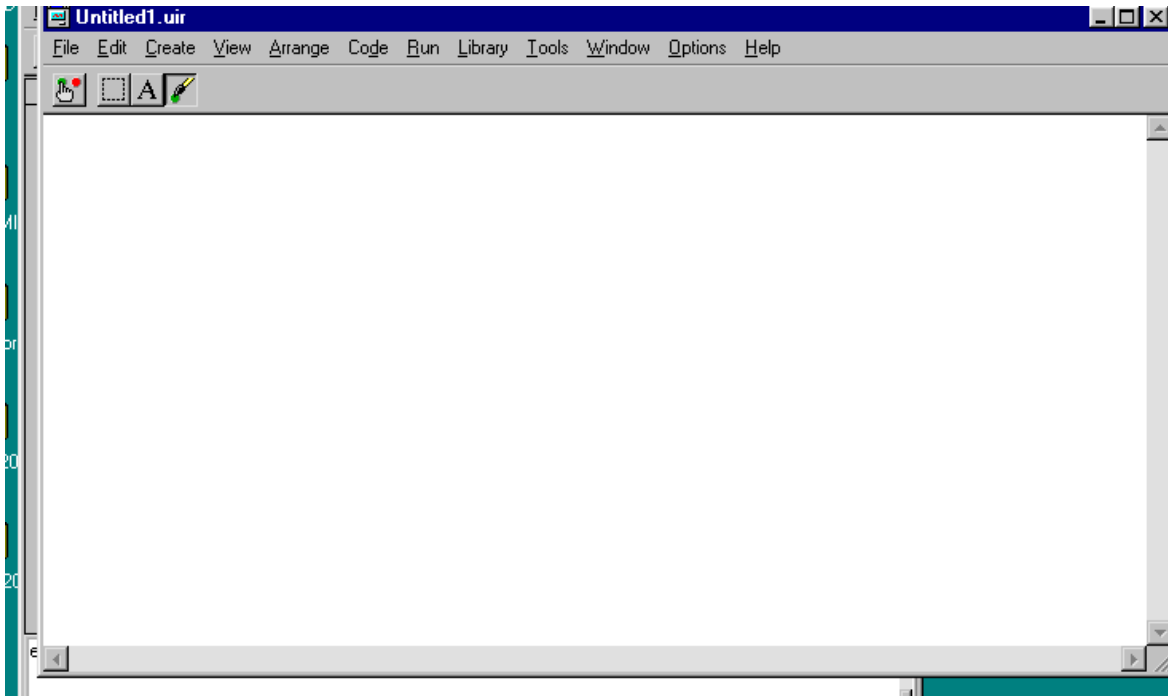
Title: "Datos aleatorios"
Y Array: datapoints
Number of points: 100
Y Data Type : double precision.
Status: err

Seleccionar la opción **Run Function Panel** del menú **Code** para ejecutar la función. Como se puede comprobar, aparece un mensaje de error indicándonos que la variable **datapoints** no ha sido definida. Para poder ejecutar la función de forma interactiva, nos situamos sobre el control de la variable a definir y seleccionamos la opción **Declare Variable** del menú **Code**. Una vez definida la variable podemos ejecutar la función.

2.3 DISEÑO DE INTERFACES GRÁFICO DE USUARIO.

En éste apartado aprenderemos a diseñar un interface gráfico de usuario (GUI) con el **User Interface Editor** y a realizar el programa que controle los distintos eventos que se produzcan en el interface gráfico.

El **User Interface Editor** es un editor que permite crear interfaces de usuario sin tener que generar ni una sola línea de código. Para acceder al editor de interfaces gráfico seleccionar la opción **New** del menú **File** de la ventana project, cuando aparezca el submenú del tipo de fichero que se desea abrir, seleccionar fichero **.uir**, de forma inmediata se abrirá la ventana **User Interface Editor**.



El editor dispone de cuatro herramientas :

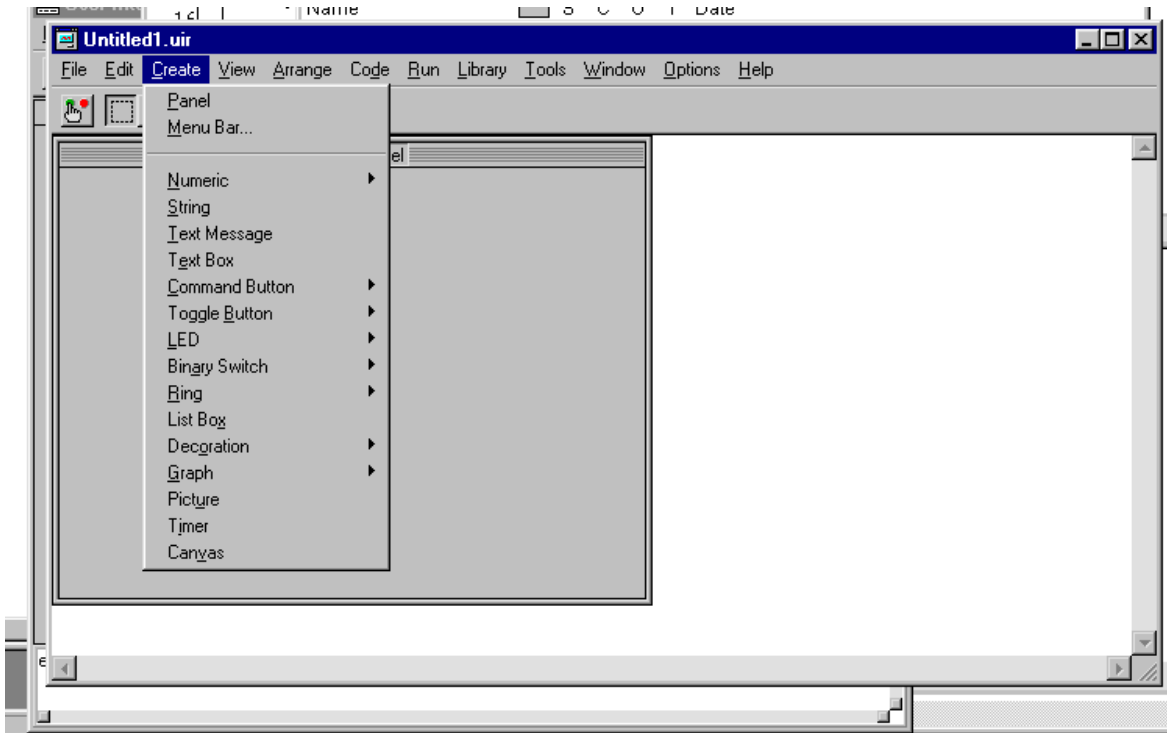


Por orden de izquierda a derecha:

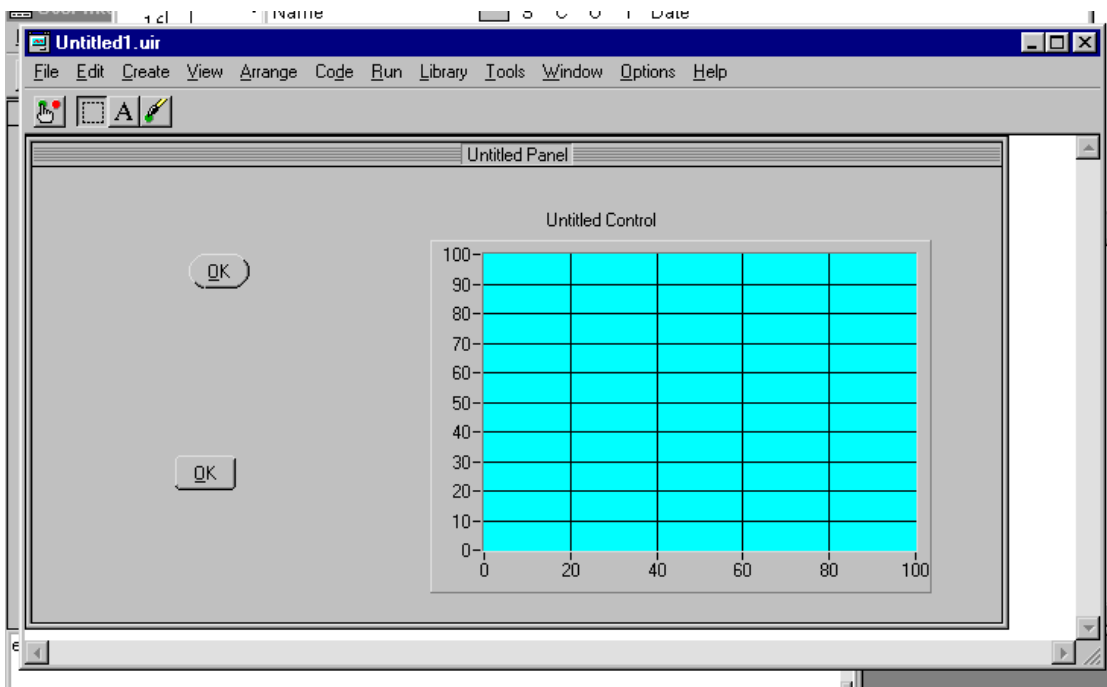
- .- **Herramienta de operación:** Cuando estamos en modo operación, los eventos que genera el interface de usuario aparecen en la parte superior derecha de la barra de herramientas
- .- **Herramienta de edición:** Para seleccionar objetos y cambiar su tamaño y posición.
- .- **Herramienta de edición de texto:** Utilizada para modificar el texto asociado aun control.
- .- **Herramienta de color:** Pulsando el botón derecho del ratón aparece una paleta de colores que nos permite elegir el color.

Una vez dentro del editor, seleccionar la opción **Panel** del menú **Create** para crear un panel

Ya tenemos creado el panel, ahora podemos ir creando los distintos objetos y colocándolos en el sitio que creamos conveniente. El menú Create contiene todos los objetos que podemos utilizar en nuestro interface gráfico: controles numéricos, cajas de texto, Leds, reguladores, mandos, interruptores de distintos tipos, barras deslizantes, gráficos, botones de comandos, etc.



Como ejercicio, vamos a crear un interface de usuario que contenga dos botones de comando y un gráfico tal como aparece en la figura:



Para crear los botones de comando seleccionar **Command Button** del menú **Create**, y para crear el gráfico seleccionar **Graph** del mismo menú. Al crear un botón de comando su etiqueta es `_OK`, de momento lo dejamos así porque a continuación explicaremos como edita los atributos del control, entre los que se incluye la etiqueta. Con el gráfico sucede lo mismo, de momento aparece una etiqueta, colores y escalas por defecto, la forma de adecuarlo a nuestra aplicación se describe a continuación.

Una vez situado el control sobre el panel, podemos moverlo y modificar su tamaño con el ratón. El menú **Edit** proporciona varias opciones de edición: copiar, pegar, borrar, etc, mediante la opción del menú **Arrange** podemos alinear, distribuir y modificar el tamaño de grupos de controles.

Almacenemos el interface gráfico de usuario en un fichero al que denominaremos *prueba2.uir*, fichero que será referenciado a través de las funciones de la **User Interface** cuando deseemos cargar el interface gráfico de usuario. Además, cuando creamos el panel debemos asignarle un prefijo constante de forma que el programa de aplicación referencia a ese prefijo cuando carga el panel desde el fichero *resource*. Por ejemplo, para cargar desde nuestro programa la aplicación el interface de usuario "*prueba2.uir*" ejecutaríamos la siguiente línea de código:

```
panelHandle = LoadPanel (0, "prueba2.uir", PANEL);
```

LoadPanel es la función de la **User Interface** que carga el interface gráfico de usuario en la memoria del ordenador, *prueba2.uir* es el fichero *resource* que contiene el interface gráfico, **PANEL** es el prefijo asignado al panel que se va a cargar, **panelHandle** es un entero que contiene la dirección de memoria a partir de la cual se ha almacenado el interface gráfico, cada vez que queramos hacer referencia al interface gráfico, utilizaremos esta variable, así por ejemplo, una vez cargado el panel en memoria, el siguiente paso será visualizarlo en pantalla, para ello se utiliza la función **DisplayPanel** de la **User Interface**, esta función utiliza como parámetro el valor de la variable **panelHandle** devuelto por la función **LoadPanel**.

```
DisplayPanel (panelHandle);
```

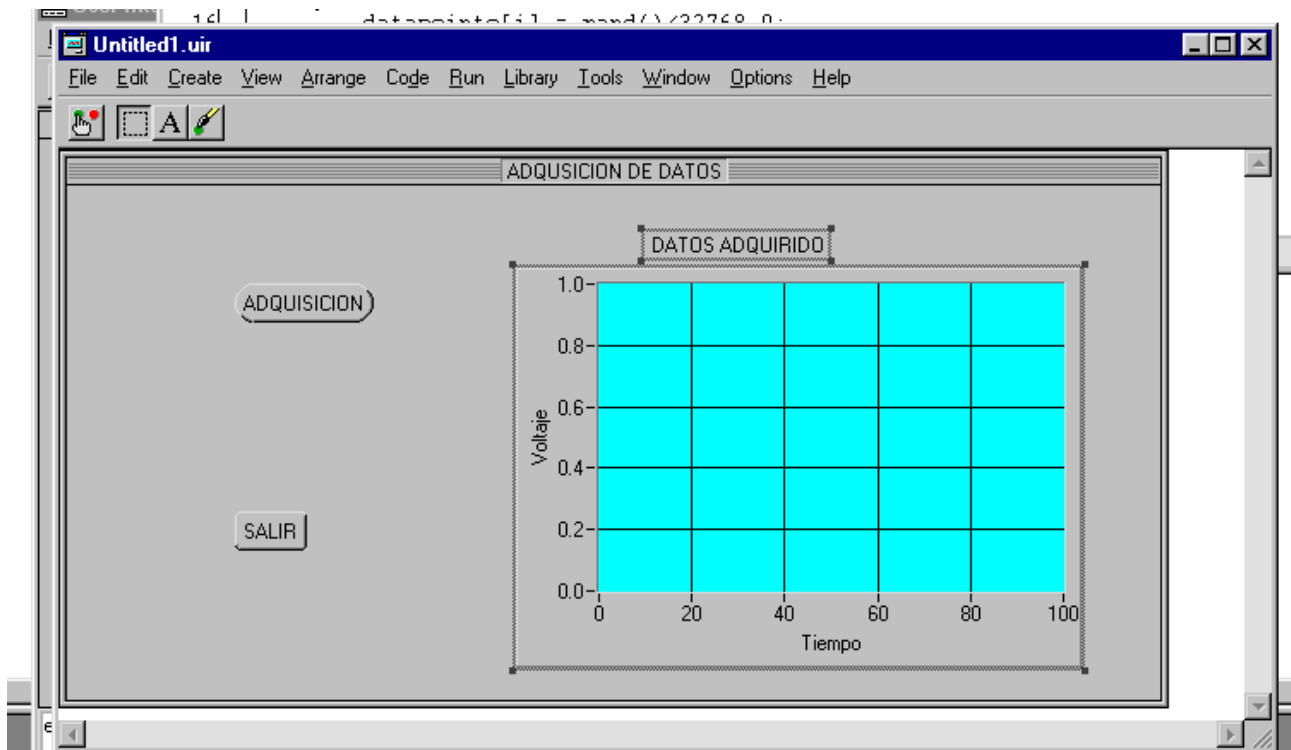
En el caso de los controles sucede algo parecido, debemos asignar a cada control el nombre de una constante. El nombre de la constante asignada a un control se conectará con el prefijo asignado al panel para generar la constante, ésta constante servirá de conexión entre el interface gráfico y el programa de aplicación. Por ejemplo, si al panel le asignamos el prefijo **PANEL**, y definimos un control al que le asignamos como nombre la constante **ADQUISICION**, cuando en nuestro programa queramos hacer referencia a este control utilizaremos la constante **PANEL_ADQUISICION**. Los nombres de las constantes de todos los controles así como el prefijo del panel se guardan en el fichero cabecera (.h) cuyo nombre coincide con el fichero *resource*, el fichero cabecera es generado automáticamente por LabWindows/CVI, y deberá ser incluido en el programa de aplicación.

Para editar los atributos de un control o de un panel, seleccionamos la herramienta de edición, nos situamos sobre el control en cuestión, lo seleccionamos y pulsamos dos veces con el ratón, aparece entonces la correspondiente ventana de diálogo donde editaremos los atributos del control o panel seleccionado.

Anteriormente creamos un panel con dos botones de comando y un gráfico, ahora sería el momento de editar sus atributos tal y como acabamos de comentar. El interface debe presentar el aspecto de la figura siguiente

Para ello editaremos los siguientes atributos:

```
Atributos para el botón de comandos ADQUISICION
.- Constant Name: ADQUIRIR
.- Callback Function: Adquisicion.
.- Control Mode: Hot
.- Label: ADQUISICION
```



Atributos para el botón de comando SALIR

- .- **Constant Name:** SALIR
- .- **Callback Function:** Salir
- .- **Control Mode:** Hot
- .- **Label:** SALIR

Atributos para el control gráfico

- .- **Constant Name:** GRAFICO
- .- **Callback Function:**
- .- **Control Mode:** Indicator
- .- **Label:** DATOS ADQUIRIDOS

Modificar las escalas de los ejes x e y para que aparezcan como en la figura.

Atributos para el panel

- .- **Constant Name:** PANEL
- .- **Callback Function:**
- .- **Label:** ADQUISICION DE DATOS

En lo que respecta al atributo Control Mode, decir que existen cuatro posibles modos de control:

Normal: especifica que el usuario puede operar sobre el control y éste puede también modificarse por programa.

Indicator: indica que el control puede modificarse por programa pero el usuario no puede actuar sobre él. Los controles **LED**, **text message**, **graph** y **strp graph** son siempre de este modo.

Hot: es idéntico que el normal excepto que en este modo el control genera un evento cuando el usuario actúa sobre él. Los eventos se devuelven al programa de aplicación y éste determina la

acción a tomar. Normalmente un control hot genera un evento cuando cambia su estado, por ejemplo si se cambia un conmutador de ON a OFF.

Los siguientes tipos de controles tienen reglas fijas sobre como generar eventos:

- . Los controles **numeric**, **string** y **text box**, generan un evento cuando el usuario pulsa <ENTER>, <TAB> o el ratón después de haber realizado una entrada de datos.

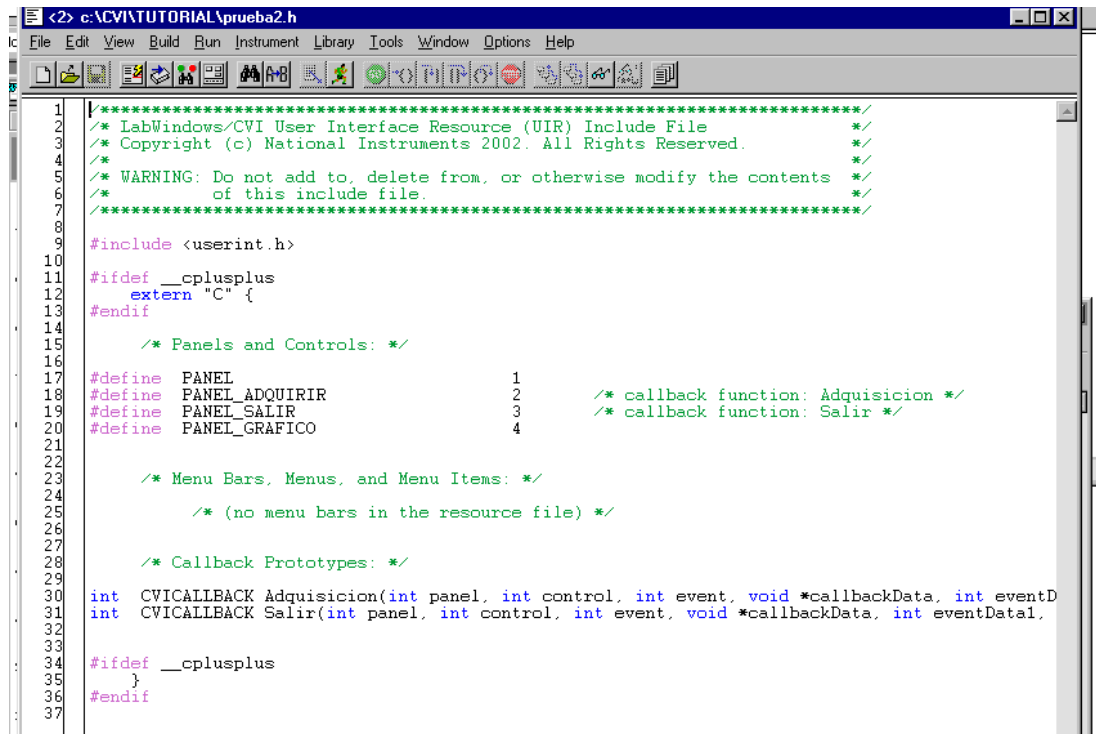
- . Los controles **list box** cuando no están en modo check, generan un evento cuando se pulsa <ENTR> estando el control activo cuando pulsamos dos veces con el ratón sobre una de las opciones de la lista.

- . Los controles **list box** cuando están en modo check, generan un evento cuando se pulsa la barra espaciadora estando el control activo, o cuando pulsamos dos veces con el ratón sobre una de las opciones de la lista.

- . Un control **graph** en modo hot, genera un evento cuando el usuario mueve un cursor mediante las teclas de flechas, o cuando se libera el botón del ratón tras haberlo cambiado de posición el cursor.

Validate: es idéntico que hot, excepto que todos los controles del panel se comprueban con respecto a un rango predefinido antes de generar el evento. Si se encuentra una condición inválida LabWindows/CVI genera una caja de diálogos indicando el porqué y dandonos la opción de poder modificar el valor.

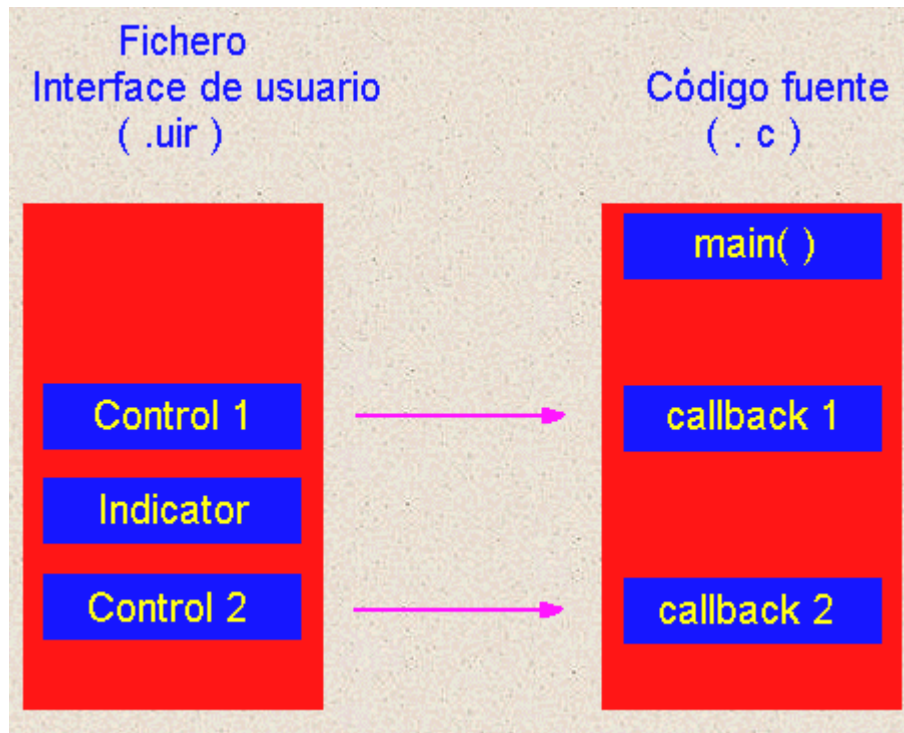
Para finalizar con la edición del interface de usuario, salvamos lo editado al fichero “prueba2.uir”. LabWindows/CVI generará de forma automática el fichero cabecera “prueba2.h” que contiene las declaraciones de las constantes definidas en los controles. Recordar que éste fichero deberá ser incluido en el código fuente de nuestra aplicación.



```
1 | /*****  
2 | /* LabWindows/CVI User Interface Resource (UIR) Include File */  
3 | /* Copyright (c) National Instruments 2002. All Rights Reserved. */  
4 | /*  
5 | /* WARNING: Do not add to, delete from, or otherwise modify the contents */  
6 | /* of this include file. */  
7 | *****/  
8 |  
9 | #include <userint.h>  
10 |  
11 | #ifdef __cplusplus  
12 |     extern "C" {  
13 | #endif  
14 |  
15 |     /* Panels and Controls: */  
16 |  
17 | #define PANEL 1  
18 | #define PANEL_ADQUIRIR 2 /* callback function: Adquisicion */  
19 | #define PANEL_SALIR 3 /* callback function: Salir */  
20 | #define PANEL_GRAFICO 4  
21 |  
22 |  
23 |     /* Menu Bars, Menus, and Menu Items: */  
24 |  
25 |     /* (no menu bars in the resource file) */  
26 |  
27 |  
28 |     /* Callback Prototypes: */  
29 |  
30 | int CVICALLBACK Adquisicion(int panel, int control, int event, void *callbackData, int eventD  
31 | int CVICALLBACK Salir(int panel, int control, int event, void *callbackData, int eventData1,  
32 |  
33 |  
34 | #ifdef __cplusplus  
35 |     }  
36 | #endif  
37 |
```

2.4.1 GENERACIÓN DEL CÓDIGO DE CONTROL

El siguiente paso será realizar el programa de control de la aplicación, es importante dejar claro que los programas de aplicación escritos en LabWindows /CVI son programas **controlados por eventos**. Cuando creamos un control en el editor de interface de usuario, lo que hacemos es definir regiones de la pantalla que pueden generar eventos, nuestro programa se escribe para controlar estos eventos generados en el interface gráfico de usuario.



Cuando se editan los controles, se les asigna una función a través del atributo **Callback Function**, de forma que cada vez que se opera sobre un control, se ejecuta la función correspondiente de forma automática. Por ejemplo en el caso del interface de usuario prueba2.uir le asignamos al botón de comandos ADQUISICION la función *Adquisicion*, pues bien, cada vez que pulsemos éste botón, se ejecuta ésta función. Ésta metodología de programación recibe el nombre de programación con **funciones callback**.

El programador puede reconocer diferentes tipos de eventos en cada función callback, por ejemplo, podemos saber si se ha pulsado el botón derecho o el izquierdo del ratón. En función del evento generado realizaremos una operación u otra.

En la siguiente tabla se representan todos los eventos que se pueden generar desde el interface de usuario.

El paso de información del evento, desde el interface de usuario al programa, se realiza a través de la lista de parámetros de la función callback. Esta lista de parámetros la genera el propio LabWindows/CVI, cada función callback tiene la siguiente lista de parámetros:

```
Function_callback (panel, control, event, callbackData, eventData1, eventData2);
```

TABLE 6-11. GUI MESSAGE EVENTS

Event Type	Event on the GUI	Information Passed to Program
Control and Menu Event	EVENT_COMMIT	Which panel or menu bar, which control or menu item.
Control Event	EVENT_VAL_CHANGED	Which panel, which control.
Control and Panel Event	EVENT_LEFT_CLICK	Which panel, which control, mouse y- and x-coordinates.
	EVENT_LEFT_DOUBLE_CLICK	Which panel, which control, mouse y- and x-coordinates.
	EVENT_RIGHT_CLICK	Which panel, which control, mouse y- and x-coordinates.
	EVENT_RIGHT_DOUBLE_CLICK	Which panel, which control, mouse y- and x-coordinates.
	EVENT_KEYPRESS	Which panel, which control, key code, pointer to key code.
	EVENT_GOT_FOCUS	Which panel, which control
	EVENT_LOST_FOCUS	Which panel, which control.
	EVENT_DISCARD	Which panel, which control.
Timer Control Event	EVENT_TIMER_TICK	Pointer to the current time (<code>double *</code>), pointer to time since the callback last received an <code>EVENT_TIMER_TICK(double *)</code> .
Panel Event	EVENT_CLOSE	Which panel.
	EVENT_PANEL_SIZE	Which panel.
	EVENT_PANEL_MOVE	Which panel.
Main Callback Event	EVENT_IDLE	Obsolete. Use timer controls instead.
	EVENT_END_TASK	Windows only. Received when Windows wants to quit. Return a non-zero value to abort the termination.

Cada parámetro de la función representa información del interface de usuario:

Panel: Informa del panel donde se generó el evento.

Control: Informa del control que generó el evento.

Event: Indica el tipo de evento generado (LEF_CLICK,KEYPRESS,etc)

CallbackData: Es un valor definido por el usuario que se pasa a la función.

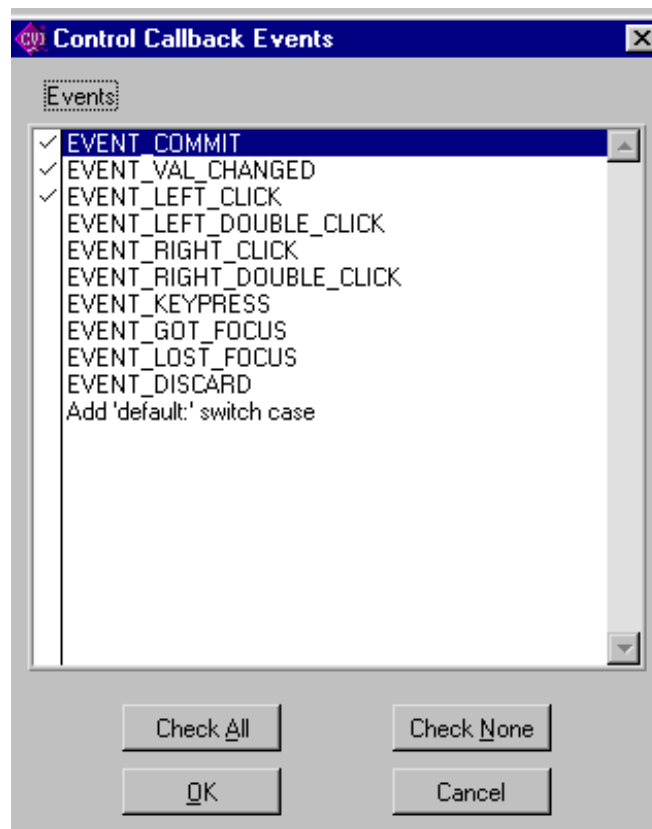
EventData1, EventData2; informa de distintos aspectos según el tipo de evento, por ejemplo si se ha pulsado el ratón estos parámetros contienen las coordenadas X e Y de la posición donde se pulsó el ratón, sin embargo si es KEYPRESS estos parámetros contienen el código de la tecla pulsada.

Los prototipos de las funciones callback de los distintos controles, de las barras de menús y de los paneles se encuentran en el fichero de cabecera “**userint.h**”.

Conocida la metodología de programación en LabWindows/CVI, pasamos a continuación a realizar el código fuente del programa de control de nuestra aplicación.

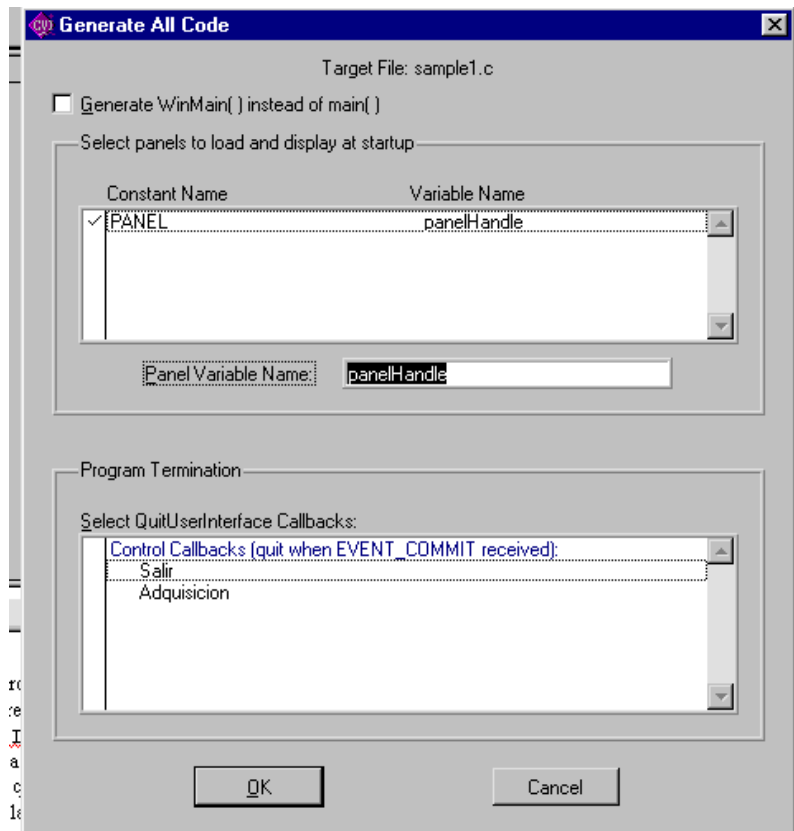
Una de las mayores ventajas del software pensado para la creación de programas profesionales es la potencia de las herramientas que incorpora. Con LabWindows/CVI 5.0 se pensó mucho en la productividad y en ahorrar al programador la pérdida de tiempo a la hora de corregir errores sintácticos, es por ello que se incorpora lo que denomina generador automático de código. Para su utilización deberemos seguir los siguientes pasos:

1. En primer lugar, debemos especificar el tipo de evento al que queremos que responda nuestro program. Dentro del editor gráfico seleccionamos del menú **Code** la opción **Preferences** y dentro de ésta la opción **Default Control Events**. Nos aparecerá el cuadro de dialogo de la figura:



2. Más tarde en éste tutorial desarrollaremos código para mostrar información de ayuda cuando pulsemos el botón derecho del ratón sobre alguno de los controles. Para hacer esto, debemos seleccionar **EVENT_RIGHT_CLICK** del **Control Callback Events** de forma que aparezca marcada junto al nombre. Por lo tanto nuestro programa responderá a dos tipos de eventos más **EVENT_COMMIT** (left-click or <Enter>) que generará los datos e imprimirá en pantalla el gráfico y un **EVENT_RIGHT_CLICK** que mostrará la ayuda. Una vez hecha la selección pulsaremos **OK**.

3. Seleccionamos **Generate»All Code** desde el menú **Code**, nos mostrará el siguiente cuadro de diálogo.



4. Debemos especificar en este cuadro algunas opciones para el generador de código. Primero debemos decirle el panel que queremos que se muestre cuando se ejecute la aplicación, en nuestro caso sólo tenemos uno , lo marcamos.

Nota: Para este ejemplo, *debemos asegurarnos que el nombre de la variable del panel es panelHandle.*

5. En la mitad inferior del cuadro de dialogo aparecen la lista de las funciones callback que hemos incluido en nuestro fichero “prueba2.uir”, de todas ellas podemos seleccionar la función que será la encargada de terminar la ejecución de nuestro programa, en nuestro caso lo será la función **Salir**.

6. Seleccionamos el botón **OK** , el generador de código creará el esqueleto del código fuente necesario para nuestra aplicación. Una nueva ventana aparecerá mostrando el siguiente código:

```
#include <cvirte.h>                /* Needed if linking in external compiler; harmless otherwise */
#include "prueba2.h"
#include <ansi_c.h>
#include <userint.h>
#include <analysis.h>

static int panelHandle;

int i=0, err;
```

```

double mean_value;
double datapoints[100];

int main (int argc, char *argv[])
{
    if (InitCVIRTE (0, argv, 0) == 0) /* Needed if linking in external compiler; harmless otherwise */
        return -1; /* out of memory */
    if ((panelHandle = LoadPanel (0, "prueba2.uir", PANEL)) < 0)
        return -1;
    DisplayPanel (panelHandle);
    RunUserInterface ();
    return 0;
}

```

```

int CVICALLBACK Adquisicion (int panel, int control, int event,
                             void *callbackData, int eventData1, int eventData2)
{
    switch (event)
    {
        case EVENT_COMMIT:

            break;
        case EVENT_RIGHT_CLICK:

            break;
    }
    return 0;
}

```

```

int CVICALLBACK Salir (int panel, int control, int event,
                      void *callbackData, int eventData1, int eventData2)
{
    switch (event)
    {
        case EVENT_COMMIT:
            QuitUserInterface (0);
            break;
        case EVENT_RIGHT_CLICK:

            break;
    }
    return 0;
}

```

7. Seleccionamos **Save** desde el menú **File** en la ventana *source* o pulsamos **Save** en el icono de la barra de herramientas salvando el código fuente con el nombre `prueba2.c`.

2.4.2 ANALIZANDO EL CÓDIGO GENERADO

El código generado en el apartado anterior está incompleto, en esta parte añadiremos el código necesario para que nuestro programa realice lo que expusimos al principio.

El programa consiste en tres funciones, recordar que es importante comprender que tarea realiza cada función del programa, porque en posteriores programas se escribirán funciones similares.

LA FUNCIÓN MAIN

La función `main` es muy simple y representa el primer paso que debemos realizar cuando construimos nuestros propios programas

```
int main (int argc, char *argv[])
{
    if (InitCVIRTE (0, argv, 0) == 0) /* Needed if linking in external compiler; harmless otherwise */
        return -1; /* out of memory */
    if ((panelHandle = LoadPanel (0, "prueba2.uir", PANEL)) < 0)
        return -1;
    DisplayPanel (panelHandle);
    RunUserInterface ();
    return 0;
}
```

El significado de cada una de las líneas de código de la función principal es el siguiente:

- La función `LoadPanel` carga el interface de usuario en memoria. El interface de usuario que carga es el que se le indica como parámetro, en nuestro caso "prueba2.uir". La función devuelve además un entero que identifica al interface cargado.
- La función `DisplayPanel` hace que el interface de usuario se represente en pantalla.
- La función `RunUserInterface` hace que LabWindows/CVI comience a enviar eventos desde el interface de usuario al programa de aplicación. Esta función deja de tener efecto cuando se ejecuta la función `QuitUserInterface` desde alguna función callback.

LA FUNCIÓN ADQUISICION

La función `Adquirir` se ejecuta automáticamente cada vez que pulsemos con el ratón el botón `ADQUISICION` de la interface de usuario . Como vemos en el código sólo nos aparece la estructura de la función que responde con un `switch case` a los evento que nosotros hemos decidido. Es tarea del programador insertar el código correspondiente a cada uno de los eventos contemplados.

```
int CVICALLBACK Adquisicion (int panel, int control, int event,
                             void *callbackData, int eventData1, int eventData2)
{
    switch (event)
    {
        case EVENT_COMMIT:

            break;

        case EVENT_RIGHT_CLICK:

            break;

    }
    return 0;
}
```

LA FUNCION SALIR

Se ejecuta cuando pulsamos el botón `SALIR`, finalizando el programa al ejecutarse la función `QuitUserInterface(0)`.

```
int CVICALLBACK Salir (int panel, int control, int event,
                       void *callbackData, int eventData1, int eventData2)
{
    switch (event)
    {
```

```

        case EVENT_COMMIT:
            QuitUserInterface (0);
            break;
        case EVENT_RIGHT_CLICK:

            break;

    }
    return 0;
}

```

GENERACIÓN DE UN ARRAY ALEATORIO DE DATOS

La tarea siguiente consiste en completar el código fuente de la función Adquisicion, recordar que el programa generaba un array de 100 número aleatorios y los imprimía en pantalla en una gráfica.

Para la generación del array utilizaremos un lazo for:

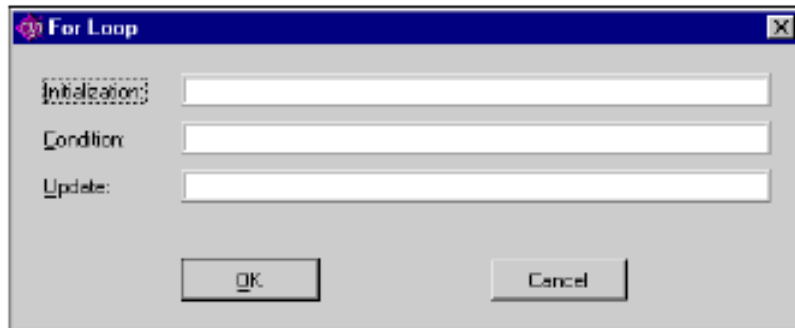
1. declaramos la variable **array datapoints** y una variable indice **i** como variables locales dentro de la función:

```

int i;
double datapoints[100];

```

2. Posicionamos el curso en la ventana del código fuente al lado del case EVENT_COMMIT en la función Adquisicion.
3. LabWindows/CVI tiene utilidades que nos pueden ayudar a generar código de las sentencias de C como lazos for , lazos while , switch . Seleccionar **Insert Construct** desde el menú **Edit** y elegir **For Loop** , mostrará el cuadro de dialogo de la figura:



4. Teclar los siguientes valores en el cuadro de dialogo del lazo for:.

```

Initialization: i=0
Condition: i<100
Update: i++

```

Pulsar **OK**.

5. Teclar la siguiente línea de código dentro del lazo for para generar los números aleatorios:

```

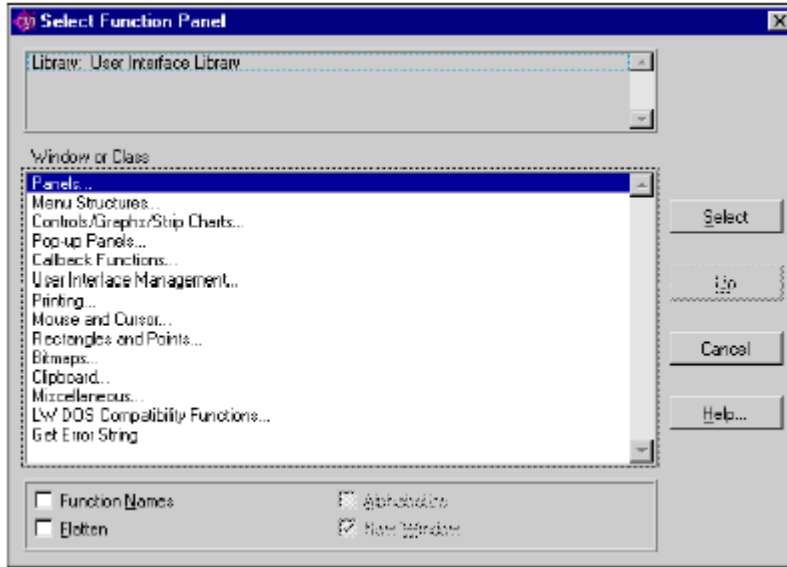
datapoints[i] = rand()/32767.0;

```

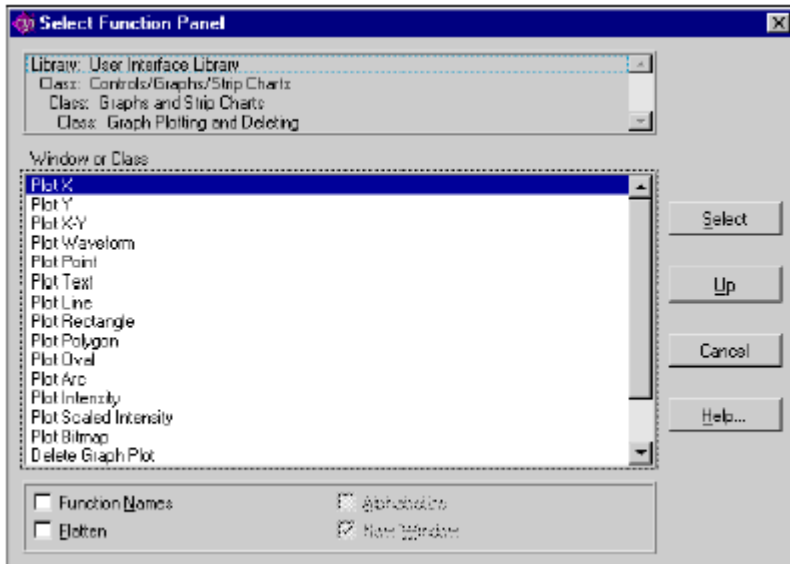
ENCONTRANDO LA FUNCIÓN PLOTY

Deberemos de seguir estos pasos para generar la línea de código que nos permitirá representar en el gráfico los números generados.

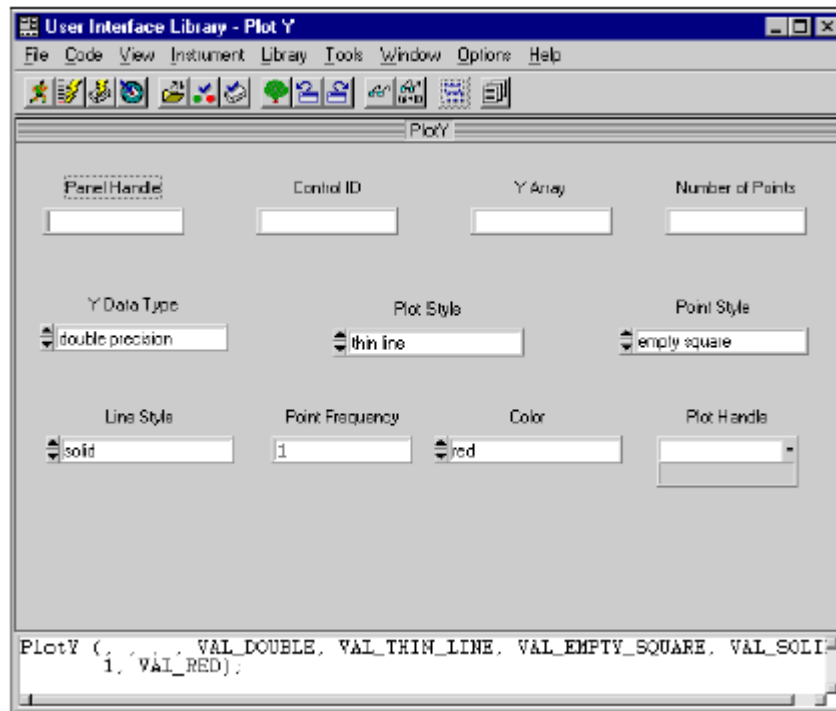
1. Nos posicionamos con el cursor a continuación del lazo for.
2. Seleccionamos el menú **Library** y elegimos **User Interface** para que nos muestre el cuadro de dialogo



3. Usamos la teclas up y down para posicionarnos en **Controls/Graphs/Strip Charts** seleccionamos y pulsamos <Enter>.
4. Usamos la teclas up y down para seleccionar **Graph and Strip Charts** de la lista y pulsamos <Enter>.
5. Pulsamos <Enter> para seleccionar **Graph Plotting and Deleting** de la lista para mostrar todas las funciones de LabWindows/CVI relacionadas con operaciones de datos sobre gráficos y strip charts, tal como muestra la siguiente figura:.

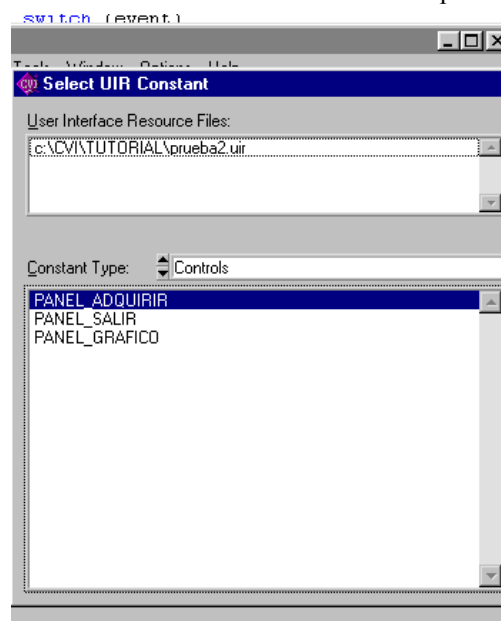


6. Usamos las flechas, o el ratón para seleccionar la función PlotY y pulsamos <Enter>. El panel de la función PlotY aparecerá como en la figura:.

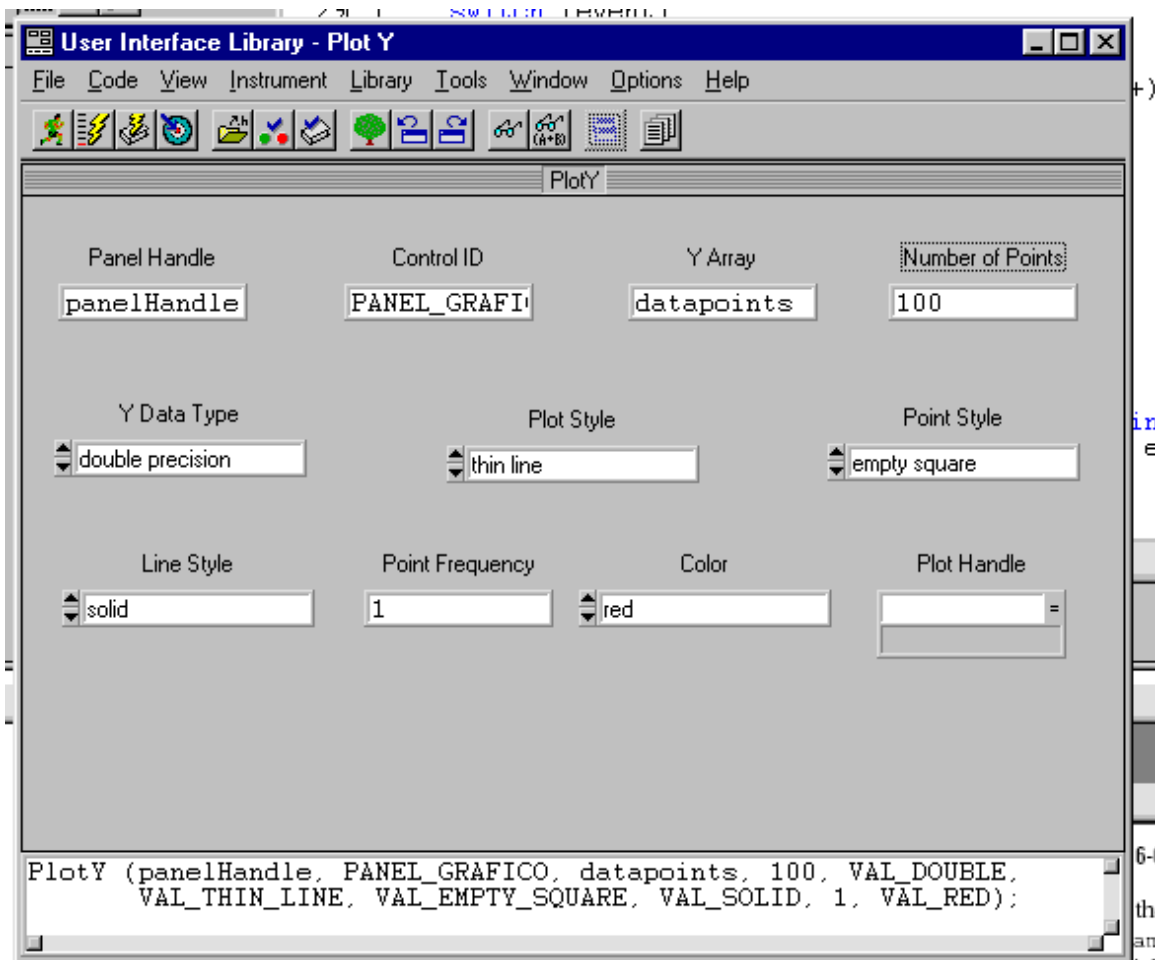


Seguiremos los siguientes pasos para construir la función PlotY.

1. Nos situamos en el control Panel Handle . Elegimos **Select Variable** del menú **Code**. Una lista de variables usadas en nuestro programa nos aparecerán , seleccionaremos panelHandle de la lista.
2. El control ID contiene el nombre de la constante asignado en el gráfico del panel, para conseguir una lista completa de todas las constantes en el fichero .uir deberemos seleccionar **Select UI Constant** desde el menú **Code**. Una lista con los nombres de las constantes nos aparece tal como se muestra en la figura:



3. Tecleamos `datapoints` en el control `Y Array` . Este nombre indica que array de los que hay en la memoria será el que muestre el gráfico.
4. Teclear `100` en el control `Number of Points` . Este número indica el número de elementos del array que deben ser dibujados.
5. Completar el panel de la función hasta quedar como sigue en la figura:



6. Seleccionar **Set Target File** del menú **Code** para indicar en que ventana será pegada la función , seleccionar `prueba2.c` y pulsar `<Enter>` o click **OK**.
7. . Seleccionar **Insert Function Call** del menú **Code** para pegar la función `PlotY` en el código fuente.
8. Cerramos el panel de funciones . El programa debe tener el aspecto siguiente:

```
int CVICALLBACK Adquisicion (int panel, int control, int event,
                             void *callbackData, int eventData1, int eventData2)
{
    int i;
    double datapoints[100];

    switch (event)
    {
        case EVENT_COMMIT: for (i=0; i<100; i++)
        {
```

```

        datapoints[i] = rand()/32767.0;
    }
    PlotY (panelHandle, PANEL_GRAFICO, datapoints, 100, VAL_DOUBLE,
        VAL_THIN_LINE, VAL_EMPTY_SQUARE, VAL_SOLID, 1, VAL_RED);

        break;
    case EVENT_RIGHT_CLICK:

        break;
    }
    return 0;
}

```

9. Debemos salvar el fichero que acabamos de generar, elegiremos **Save** o **Save as** del menú **File**.

2.5 CREANDO EL PROYECTO

La ventana de proyectos contiene la lista de todos aquellos ficheros que conforman nuestro proyecto. En nuestro caso debemos tener tres ficheros que son:

- prueba2.uir
- prueba2.h
- prueba2.c

Seguiremos los siguientes pasos para crear el proyecto:

1. Cerraremos todas las ventanas excepto la ventana project (untitled1.prj).
2. Seleccionar **Add File to Project** desde el menú **Edit**. Elegir **All Files (*.*)** en tipo de fichero. de éste modo nos aparecerá un cuadro de dialogo que nos listará todos los ficheros en el directorio actual.
3. Seleccionar prueba2.c, prueba2.h, y prueba2.uir del cuadro de dialogo.
4. Pulsar **OK**.
5. Seleccionar **Save** desde el menú **File**.
6. Teclear prueba2 en File Name input. Pulsar <Enter> hacer click en **Save**.

2.6 EJECUTANDO EL PROYECTO

Ahora tenemos el proyecto completo, salvado como prueba2.prj. Podemos ver el estado de cada fichero en la ventana de proyectos así como editar cada una de ellas simplemente pulsando con el ratón dos veces sobre el nombre del fichero. Seleccionar **Run Project** desde el menú **Run** para ejecutar el código.

1. LabWindows/CVI compila el código fuente de prueba2.c y lo encadena con las librerías apropiadas de LabWindows/CVI.
2. La interface de usuario nos aparecerá en pantalla lista para aceptar las entradas desde el teclado o con el ratón
3. Cuando pulsamos sobre el botón **ADQUISICION** LabWindows/CVI pasa la información del evento generado por el ratón directamente a la función callback Adquisicion
4. La función Adquisicion genera un array de números aleatorios y los pinta en el control gráfico en la interface de usuario
5. Cuando pulsamos en el botón **Salir**, la información generada por el evento con el ratón es pasada directamente a la función Shutdown, encargada de detener la ejecución del programa

3

AÑADIENDO EL ANÁLISIS DE LOS DATOS A NUESTRA APLICACIÓN EN LabWindows/CVI, Y ALGO MÁS.

INTRODUCCIÓN

Como se comentó en el Tema 1, dentro de la estructura de un programa, la última fase que debemos desarrollar es la de análisis de los datos. En el ejemplo desarrollado en el Tema 2 “prueba2.prj”, nos quedó pendiente esta última fase. en el siguiente capítulo retomaremos el ejemplo anterior y le añadiremos la fase de análisis y algún detalle más.

Recordar que en el programa “prueba2.prj” se simulaba una posible adquisición de datos, para ello, se generaban datos de forma aleatoria y se almacenaban en un array. Finalmente los datos generados se representaban en el gráfico. El interface de usuario únicamente dispone de dos controles:

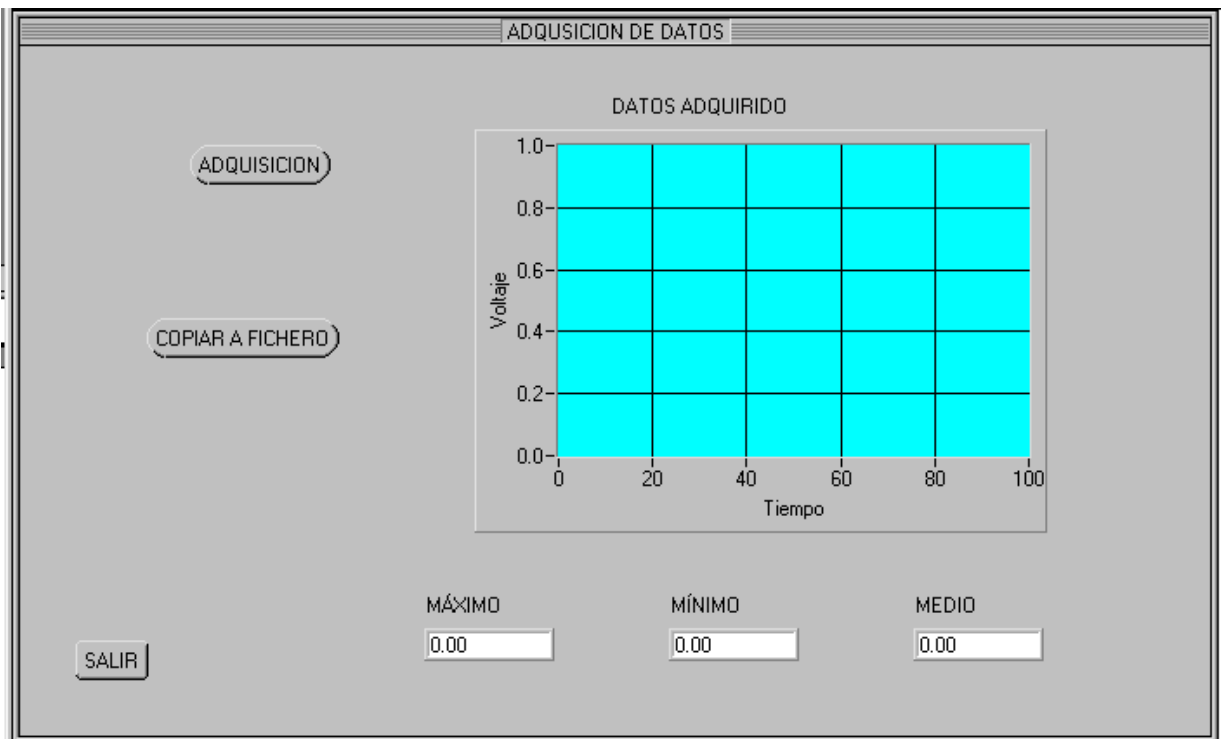
ADQUISICION: para realizar la adquisición de los datos.
SALI: para abandonar la aplicación.

Partiendo de ésta situación, desarrollaremos los siguientes apartados, tomando como base lo que tenemos hecho:

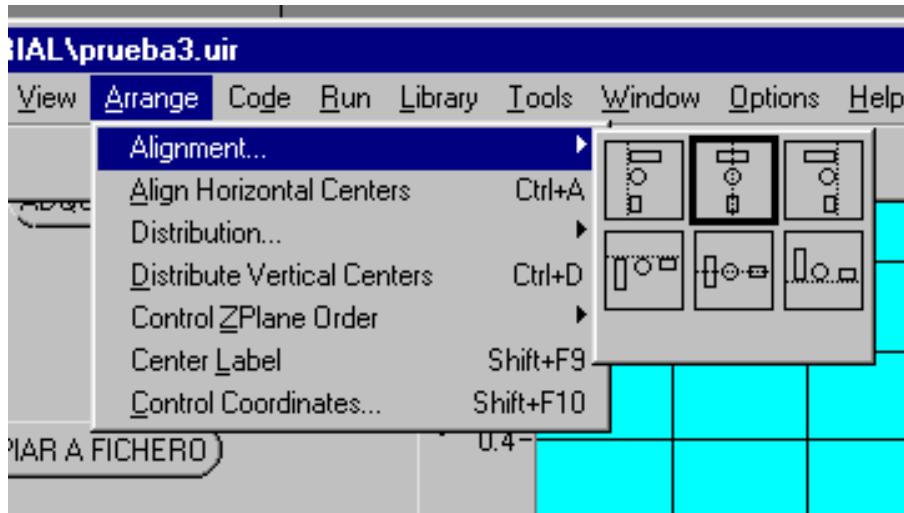
1. Añadir al interface gráfico de usuario tres nuevos controles que indiquen el valor máximo, mínimo y medio de los valores adquiridos. Estos **controles** deberán ser únicamente **indicadores**. Para obtener los valores **máximo**, **mínimo** y **medio**, utilizaremos las funciones de la librería *Advanced Analysis*.
2. Añadiremos un nuevo control, al que denominaremos “COPIAR A FICHERO”, que permitirá volcar las señales adquiridas a un fichero de disco. Utilizaremos para este fin algunas de las funciones de la librería *User Interface*.
3. Añadiremos la opción de ayuda en algunos de los controles, de modo que cuando nos situemos sobre uno de ellos y pulsemos **el botón derecho** del ratón nos aparecerá sobre la pantalla un mensaje de ayuda indicando para que sirve cada control.
Utilizaremos para ello las funciones de **mensajes pop-up** incluidas en la librería *User Interface*.

EDICION DEL INTERFACE DE USUARIO.

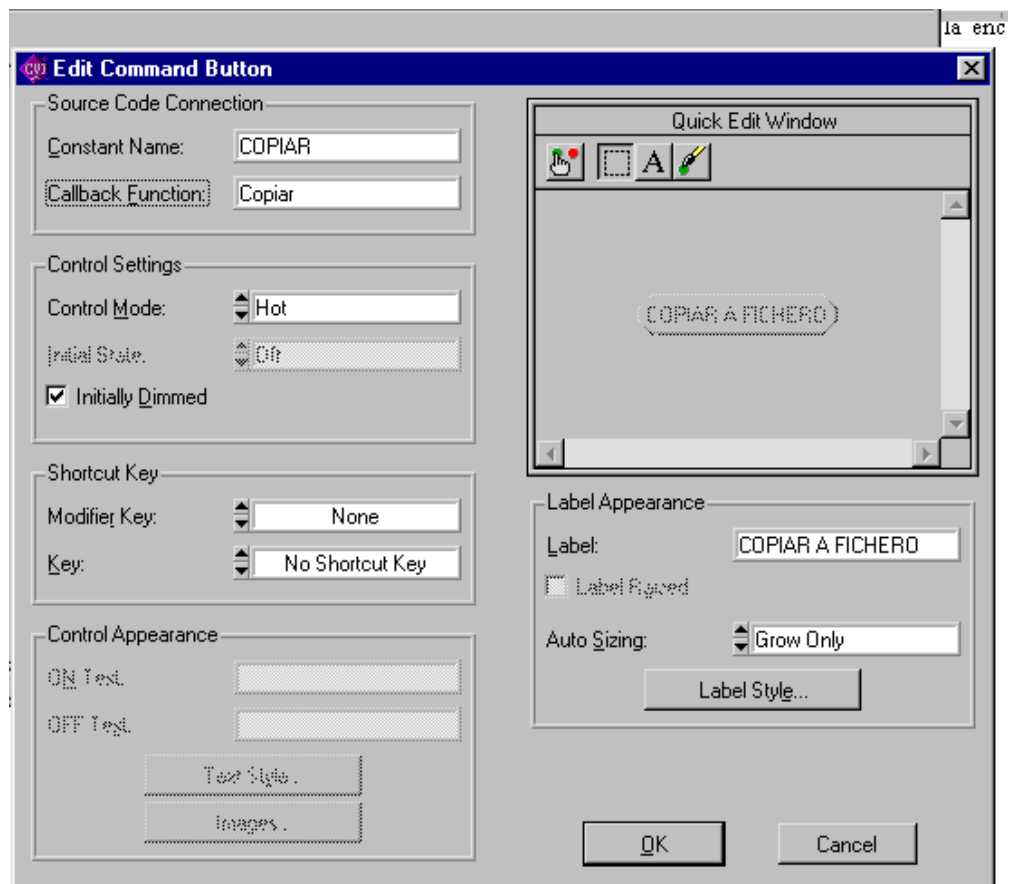
Según lo especificado en la introducción del tema 3, el aspecto que ahora de be presentar nuestro nuevo panel sería el siguiente:



1. Abriremos desde la ventana de project el fichero prueba2.uir, creado en el tema anterior , salvando el fichero antes de modificarlo como prueba3.uir.
2. Añadiremos los nuevos controles que tendrá la aplicación, para ello nos situamos encima del panel y pulsando con el botón derecho del ratón nos aparecerá la lista de todos los controles:
 - Para el control COPIAR A FICHERO elegiremos un de los cuatro modelos de **Command Button**.
 - Para los tres controles numéricos **Numeric-Numeric**
3. El siguiente paso sería distribuir, modificar su tamaño y alinear los controles, para alinear los comandos utilizaremos la opción **Arrange-Alignent** dentro del UIR. Para poder utilizar la opción de alineamiento previamente se deben de haber elegido los controles que se quieren alinear. En nuestro caso seleccionamos los tres nuevos controles numéricos.
4. Etiquetamos los controles con los nombres indicados en la figura anterior y editamos los controles con los nombre de constante **MAXIMO**, **MINIMO** y **MEDIO** respectivamente. El resto de opciones las dejamos tal cual. Comprobando que el modo de los controles sea el de **indicador**.



5. Añadimos el tercer botón de comando etiquetado como, **COPIAR A FICHERO**, con nombre de constante **COPIAR** y en modo **Hot**, y como será el elemento del panel que una vez generado y representado el array en el gráfico será el encargado de guardar el array en un fichero, inicialmente al cargar el panel por primera vez deberá aparecer como **dimmed** y pasará a estar activo una vez que pulsemos el botón de **ADQUIRIR**. Asociado a éste y en el campo de **callback función** colocaremos el nombre de la función que será llamada al pulsar el botón y se encargará de almacenar el array en un fichero (*Copiar*).



6. Salvaremos el fichero como **prueba3.uir** y comenzaremos en la siguiente sección con la generación del código.

GENERACIÓN DEL CÓDIGO.

1. Utilizaremos la herramienta de generación de código tal como vimos en el tema anterior, recordar que consideramos en nuestro programa la posibilidad de que nos aparezca un mensaje pop-up de ayuda, al pulsar el botón derecho sobre cada uno de los controles.

Nota: no es necesario indicar en que fichero vamos a colocar el código (Set Target File del menú Code), automáticamente nos crea el fichero de la figura siguiente bastará con salvarlo con el nombre prueba3.c, si seguimos los pasos del tema anterior.

```
#include <cvirte.h>                /* Needed if linking in external compiler; harmless otherwise */
#include <userint.h>
#include "prueba3.h"

static int panelHandle;

int main (int argc, char *argv[])
{
    if (InitCVIRTE (0, argv, 0) == 0) /* Needed if linking in external compiler; harmless otherwise */
        return -1; /* out of memory */
    if ((panelHandle = LoadPanel (0, "prueba3.uir", PANEL)) < 0)
        return -1;
    DisplayPanel (panelHandle);
    RunUserInterface ();
    return 0;
}

int CVICALLBACK Adquisicion (int panel, int control, int event,
                             void *callbackData, int eventData1, int eventData2)
{
    switch (event)
    {
        case EVENT_COMMIT:

            break;
        case EVENT_RIGHT_CLICK:

            break;
    }
    return 0;
}

int CVICALLBACK Salir (int panel, int control, int event,
                      void *callbackData, int eventData1, int eventData2)
{
    switch (event)
    {
        case EVENT_COMMIT:
            QuitUserInterface (0);
            break;
        case EVENT_RIGHT_CLICK:

            break;
    }
    return 0;
}
```

```

int CVICALLBACK Copiar (int panel, int control, int event,
                        void *callbackData, int eventData1, int eventData2)
{
    switch (event)
    {
        case EVENT_COMMIT:

            break;
        case EVENT_RIGHT_CLICK:

            break;
    }
    return 0;
}

```

2. Ahora analizaremos función por función el código que necesitamos añadir para que haga lo que hemos planteado al principio del ejercicio.

La función **Adquisicion** seguirá el siguiente esquema escrito en *seudocódigo*:

SI EVENT_COMMIT

Borrar el gráfico;
Generar el array;
Dibujar en pantalla el array de números;
Calcular el valor máximo;
Calcular el valor mínimo;
Calcular el valor medio;
Mostrar los resultados en sus casillas correspondientes;
Hacer visible el botón de GUARDAR A FICHERO;
Fin;

SI EVENT_RIGHT_CLICK

Sacar un mensaje pop-up que diga: Botón para comenzar la adquisición;
Fin;

Traducido a código en C sería:

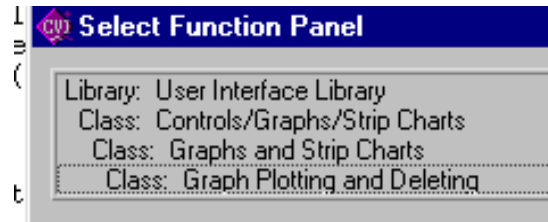
```

int CVICALLBACK Adquisicion (int panel, int control, int event,
                             void *callbackData, int eventData1, int eventData2)
{
    switch (event)
    {
        case EVENT_COMMIT:
            DeleteGraphPlot (panelHandle, PANEL_GRAFICO, -1, VAL_IMMEDIATE_DRAW);
            for(i=0;i<100;i++)
                datapoint[i]=rand()/32767.0;
            PlotY (panelHandle, PANEL_GRAFICO, datapoint, 100, VAL_DOUBLE,
                 VAL_THIN_LINE, VAL_EMPTY_SQUARE, VAL_SOLID, 1, VAL_RED);
            MaxMin1D (datapoint, 100, &max, &max_index, &min, &min_index);
            Mean (datapoint, 100, &medio);
            SetCtrlVal (panelHandle, PANEL_MAXIMO, max);
            SetCtrlVal (panelHandle, PANEL_MINIMO, min);
            SetCtrlVal (panelHandle, PANEL_MEDIO, medio);
            SetCtrlAttribute (panelHandle, PANEL_COPIAR, ATTR_DIMMED, 0);
            break;
        case EVENT_RIGHT_CLICK:
            MessagePopup ("MENSAJE AYUDA",
                          "Pulsar para la adquisición de la señal");
            break;
    }
    return 0;
}

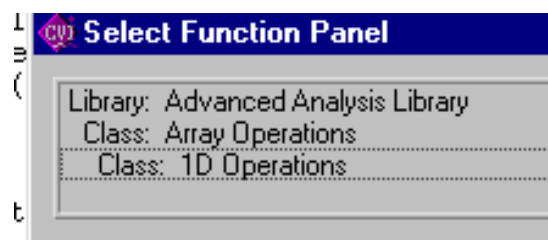
```

Todas las funciones que aparecen en el código escrito están obtenidas de la librería, mediante los paneles de funciones, en la siguiente lista se incluye el camino para obtener cada una de las funciones, pero es labor de vosotros como programadores investigar y descubrir cual es la función que se ajusta a la tarea que queremos realizar:

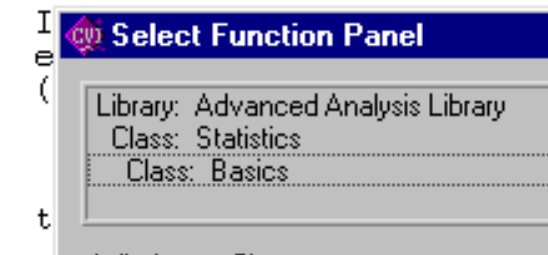
DeleteGraphPlot y PlotY



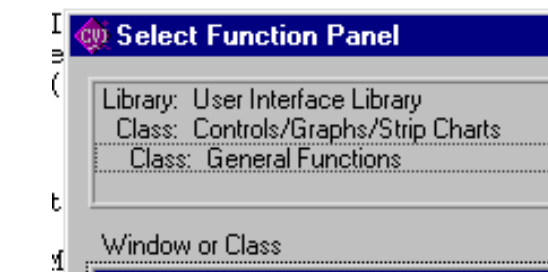
MaxMin1D



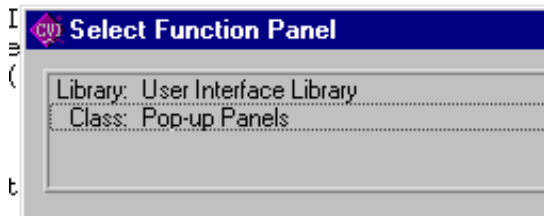
Mean



SetCtrlVal y SetCtrlAttribute



MessagePopup



La función **Salir** seguirá el siguiente esquema escrito en *seudocódigo*:

SI EVENT_COMMIT

Salir de la interface de usuario;

Fin;

SI EVENT_RIGHT_CLICK

Sacar un mensaje pop-up que diga: Botón para salir del programa;

Fin;

Traducido a código en C sería:

```
int CVICALLBACK Salir (int panel, int control, int event,
                      void *callbackData, int eventData1, int eventData2)
{
    switch (event)
    {
        case EVENT_COMMIT:
            QuitUserInterface (0);
            break;
        case EVENT_RIGHT_CLICK:
            MessagePopup ("MENSAJE AYUDA",
                          "Pulsar para salir del programa");
            break;
    }
    return 0;
}
```

La función **Copiar** seguirá el siguiente esquema escrito en *seudocódigo*:

SI EVENT_COMMIT

Si (no se ha seleccionado un fichero) sacar un mensaje “no se ha seleccionado ningún fichero”;

Escribir la tabla al fichero seleccionado;

Fin;

SI EVENT_RIGHT_CLICK

Sacar un mensaje pop-up que diga: Botón para guardar los ficheros a disco;

Fin;

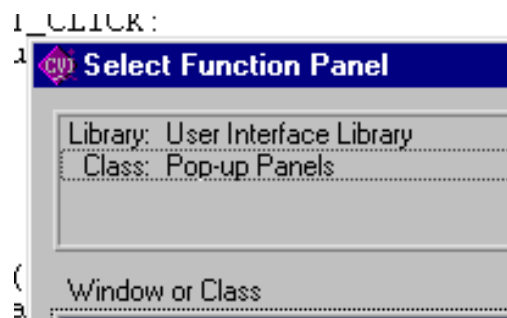
Traducido a código en C sería:

```
int CVICALLBACK Copiar (int panel, int control, int event,
                        void *callbackData, int eventData1, int eventData2)
{
    char nombre[260];
    switch (event)
    {
        case EVENT_COMMIT:
            if( FileSelectPopup ("", " *.*", "", "guardar fichero en", VAL_SAVE_BUTTON,
                                0, 0, 1, 0, nombre)!=0)
                MessagePopup ("MENSAJE", "NO SE HA SELECCIONADO NINGUN
                                FICHERO");
            else
                ArrayToFile (nombre, datapoint, VAL_DOUBLE, 100, 1,
                            VAL_GROUPS_TOGETHER, VAL_GROUPS_AS_COLUMNS,
                            VAL_CONST_WIDTH, 10, VAL_ASCII, VAL_TRUNCATE);

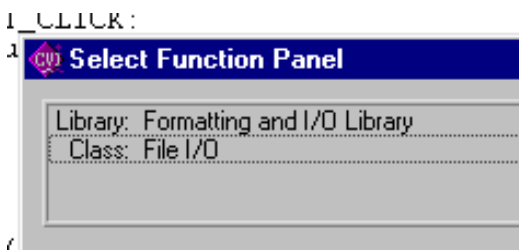
            break;
        case EVENT_RIGHT_CLICK:
            MessagePopup ("MENSAJE AYUDA",
                        "Pulsar para copiar array a fichero");
            break;
    }
    return 0;
}
```

Igual que anteriormente las funciones han sido generadas con los paneles de funciones, podéis encontrar las funciones en las siguientes librerías:

FileSelectPopup



ArrayToFile



3. El código del programa completo será el siguiente:

```

#include <formatio.h>
#include <ANSI_C.h>
#include <analysis.h>
#include <cvirte.h>          /* Needed if linking in external compiler; harmless otherwise */
#include <userint.h>
#include "prueba3.h"

static int panelHandle;
int i,max_index,min_index;
double max,min,medio;
double datapoint[100];
int main (int argc, char *argv[])
{
    if (InitCVIRTE (0, argv, 0) == 0)          /* Needed if linking in external compiler; harmless otherwise */
        return -1; /* out of memory */
    if ((panelHandle = LoadPanel (0, "prueba3.uir", PANEL)) < 0)
        return -1;
    DisplayPanel (panelHandle);
    RunUserInterface ();
    return 0;
}

int CVICALLBACK Adquisicion (int panel, int control, int event,
void *callbackData, int eventData1, int eventData2)
{
    switch (event)
    {
        case EVENT_COMMIT:
            DeleteGraphPlot (panelHandle, PANEL_GRAFICO, -1, VAL_IMMEDIATE_DRAW);
            for(i=0;i<100;i++)
            {
                datapoint[i]=rand()/32767.0;
            }
            PlotY (panelHandle, PANEL_GRAFICO, datapoint, 100, VAL_DOUBLE,
                VAL_THIN_LINE, VAL_EMPTY_SQUARE, VAL_SOLID, 1,
                VAL_RED);
            MaxMin1D (datapoint, 100, &max, &max_index, &min, &min_index);
            Mean (datapoint, 100, &medio);
            SetCtrlVal (panelHandle, PANEL_MAXIMO, max);
            SetCtrlVal (panelHandle, PANEL_MINIMO, min);
            SetCtrlVal (panelHandle, PANEL_MEDIO, medio);
            SetCtrlAttribute (panelHandle, PANEL_COPIAR, ATTR_DIMMED, 0);

            break;
        case EVENT_RIGHT_CLICK:
            MessagePopup ("MENSAJE AYUDA",
                "Pulsar para la adquisición de la señal");

            break;
    }
    return 0;
}

int CVICALLBACK Salir (int panel, int control, int event,
void *callbackData, int eventData1, int eventData2)
{
    switch (event)
    {
        case EVENT_COMMIT:
            QuitUserInterface (0);
            break;
        case EVENT_RIGHT_CLICK:
            MessagePopup ("MENSAJE AYUDA",
                "Pulsar para salir del programa");

            break;
    }
}

```

```

        return 0;
    }
int CVICALLBACK Copiar (int panel, int control, int event,
    void *callbackData, int eventData1, int eventData2)
{
    char nombre[260];
    switch (event)
    {
        case EVENT_COMMIT:
            if( FileSelectPopup ("", " *.*", "", "guardar fichero en", VAL_SAVE_BUTTON,
                0, 0, 1, 0, nombre)==0)
                MessagePopup ("MENSAJE", "NO SE HA SELECCIONADO NINGUN FICHERO");
            else
                ArrayToFile (nombre, datapoint, VAL_DOUBLE, 100, 1,
                    VAL_GROUPS_TOGETHER,
VAL_GROUPS_AS_COLUMNS, VAL_CONST_WIDTH, 10, VAL_ASCII,
VAL_TRUNCATE);
            break;
        case EVENT_RIGHT_CLICK:
            MessagePopup ("MENSAJE AYUDA",
                "Pulsar para copiar gráfico a fichero");
            break;
    }
    return 0;
}

```

4. Como propuesta para completar el ejercicio y para que comencéis vosotros a programar, se propone para completar el ejemplo que se le incluya una opción al programa para que podamos leer los ficheros grabados y los podamos representar en la gráfica.

COMENTARIOS FINALES

Los valores de los controles de interface de usuario pueden actuar como entradas o como salidas, de forma que durante la ejecución de un programa, se puede tanto leer, como actualizar el valor de éstos. Para ello se utilizan las funciones **GetCtrlVal** y **SetCtrlVal**. Por ejemplo, supongamos que el usuario modifica el valor de un mando rotatorio, para leer el nuevo valor utilizamos la función **GetCtrlVal** de la siguiente manera:

```
GetCtrlVal(panelHandle, PANEL_MANDO, &valor_mando);
```

Si lo que se desea es actualizar el valor de un control, utilizaremos la función **SetCtrlVal**:

```
SetCtrlVal(panelHandle, PANEL_MANDO, valor)
```

Por otra parte, una vez creado el interface de usuario, puede ser que deseemos cambiar el aspecto durante la ejecución del programa. Para ello Labwindows/CVI tiene definidas las constantes que representan los atributos de los paneles, controles, barras de menús así como diversas constantes con posibles valores para esos atributos, de forma que utilizando la función **SetCtrlAttribute** podemos cambiar la apariencia del interface de usuario. Por ejemplo, para cambiar el color de fondo de un botón a rojo, utilizaríamos la función **SetCtrlAttribute** de siguiente manera:

```
SetCtrlAttribute(panelHandle, PANEL_BOTON; ATTR_CMD_BUTTON_COLOR, VAL_RED);
```

Para conocer los atributos de un control determinado utilizaremos la función **GetCtrlAttribute**.

En el panel de funciones **SetCtrlAttribute** y **GetCtrlAttribute** se definen todos los atributos, de los controles que pueden utilizarse con dichas funciones, para conocerlos situarse sobre el control adecuado y utilizar la ayuda que proporcionan los paneles de funciones.

4

INCORPORAR MENÚS A NUESTRAS APLICACIONES EN LabWindows/CVI.

INTRODUCCIÓN

En la mayoría de aplicaciones de Windows es necesario incluir menús desplegables, que nos permitan configurar o realizar determinadas operaciones tales como configurar el sistema, trabajos con ficheros, temas de ayuda, etc.

Crear barras de menús con Labwindows/CVI es muy sencillo, y para seguir la línea de nuestra introducción en el entorno de desarrollo lo haremos con un ejemplo, añadiremos a nuestro ejemplo del tema 3 en menú que incluya las opciones que se detallan en la figura .

Archivo	Configuración		AYUDA
Copiar a Fichero Ctr+F	Aspecto de Gráfico	Tipo de línea	Acerca de.....
-----		Color línea	
Cargar Archivo Ctr+C			

Imprimir	Tabla		
-----	Gráfico		
Salir			

Es importante que antes de comenzar la creación del menú no sólo dejemos claro el aspecto que tendrá el menú (figura anterior), si no, también que hará cada una de las opciones que han sido incluidas en nuestro menú y como haremos referencia a los items , por eso debemos pararnos a pensar en:

- que queremos que haga cada uno de los elementos cuando los seleccionemos con el ratón. Es decir, si llamarán a una callback función o por el contrario no.

- Que nombre tendrá cada elemento (nombre de constante). El nombre final de un elemento del menú estará formado por la concatenación de todos ellos comenzando por el nombre constante del menú.
Ejemplo : MENU_ARCHIVO_CARGAR

Nota: Para ver que nombre recibe cada uno de los item creados, una vez realizado el menú podemos encontrar los nombres en el fichero prueba4.h.

- Si estará o no activo al ejecutar el programa (Dimmed).
- Si queremos acceder a la opción del menú desde alguna combinación especial de teclas

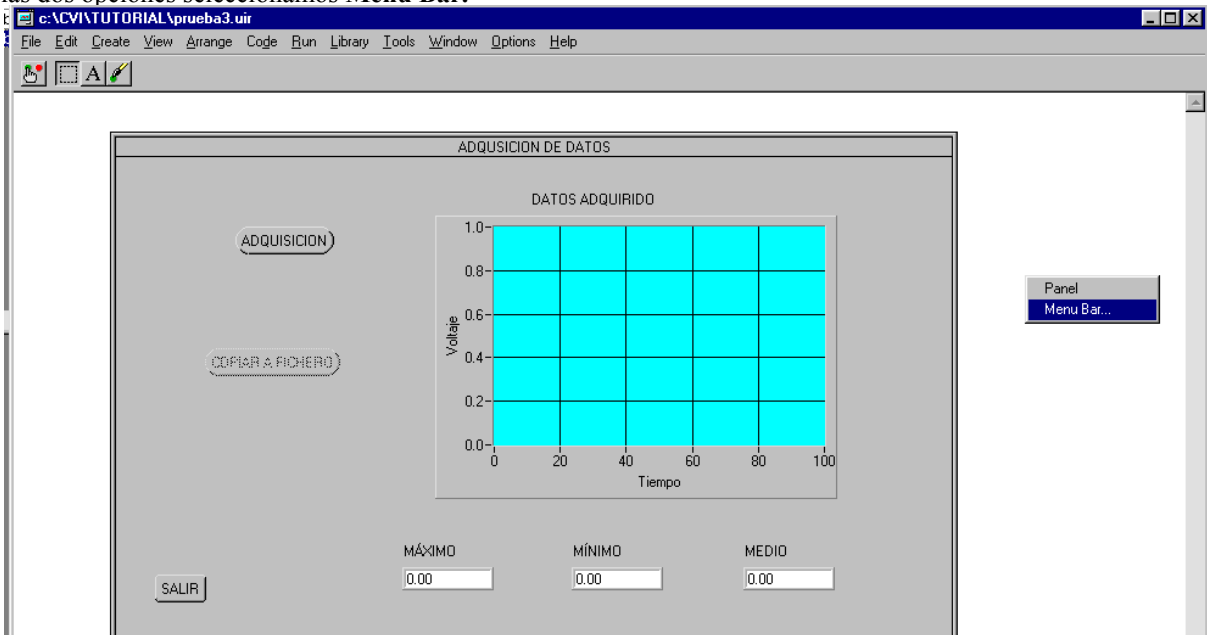
Teniendo en cuenta el párrafo anterior, a estructura del menú deberá cumplir lo siguiente:

1. Al menú le llamaremos **MENU**.
2. Para el elemento **Archivo**. Se debe poder acceder pulsando las teclas **ALT+A** y tendrá como nombre de constante **ARCHIVO**.
 - La opción **Copiar a Fichero** accesible también al pulsar las teclas **Ctrl+F**, estará dentro del menú Archivo. Cuando seleccionemos esta opción se deberá copiar la tabla de valores generada a un archivo. El nombre de constante asociado será **GUARDAR**. Al arrancar el programa esta opción aparecerá **Dimmed**, sólo cuando se genere la tabla se podrá utilizar la opción. Esta opción llamará a una función callback llamada **MenuArchivo**
 - La opción **Cargar Archivo** accesible también al pulsar las teclas **Ctrl+C**, estará dentro del menú Archivo. Cuando seleccionemos esta opción se deberá copiar la tabla de valores almacenada en un archivo que seleccionaremos desde una ventana pop-up en el elemento de control gráfico. El nombre de constante asociado será **CARGAR**. Esta opción llamará a una función callback llamada **MenuArchivo**
 - La opción **Imprimir**, estará dentro del menú **Archivo**. Cuando seleccionemos esta opción se deberá desplegar otro menú con dos opciones **Tabla** y **Gráfico**, de modo que en función de la elección se imprimirá la tabla o el gráfico. El nombre de constante asociado será **IMPRIMIR** para el item **Imprimir** y **TABLA** y **GRAFICO** para los subitems Tabla y Gráfico, ambos subitems, llamarán a la función callback **MenuArchivo**.
 - La opción **Salir** accesible también al pulsar las teclas **Alt+S**, estará dentro del menú Archivo. Cuando seleccionemos esta opción se deberá salir del programa. El nombre de constante asociado será **SALIR**. Esta opción llamará a una función callback llamada **MenuArchivo**
3. Para el elemento **Configuración**. Se debe poder acceder pulsando las teclas **ALT+C**. y tendrá como nombre de constante **CONFIG**.
 - La opción **Aspecto del Gráfico**, estará dentro del menú Configuración. Cuando seleccionemos esta opción se desplegará otro menú con dos opciones **Tipo Línea** y **Color Línea**, de modo que en función de la elección se modificará el tipo de línea con que se pinta el gráfico o el color de la misma. El nombre de constante asociado será **ASPECTO** para el item **Aspecto del Gráfico**, y **LINEA** y **COLOR** para los subitems Tipo de línea y Color línea. Ambos subitems, llamarán a la función callback **MenuConfiguracion**.
 - La opción **Salir** accesible también al pulsar las teclas **Alt+S**, estará dentro del menú Archivo. Cuando seleccionemos esta opción se deberá salir del programa. El nombre de constante asociado será **SALIR**. Esta opción llamará a una función callback llamada **MenuArchivo**
4. Para el elemento **Ayuda**. Tendrá como nombre de constante **AYUDA**.
 - La opción **Acerca de.....**, estará dentro del menú Ayuda. Cuando seleccionemos esta opción se desplegará una ventana pop-up indicando el nombre del programa fecha de realización autores, versión, etc.. El nombre de constante asociado será **AYUDA** para el item **Ayuda**, y **ACERCADE** para el subitem **Acerca de.....**. El subitem llamará a la función callback **MenuAyuda**.

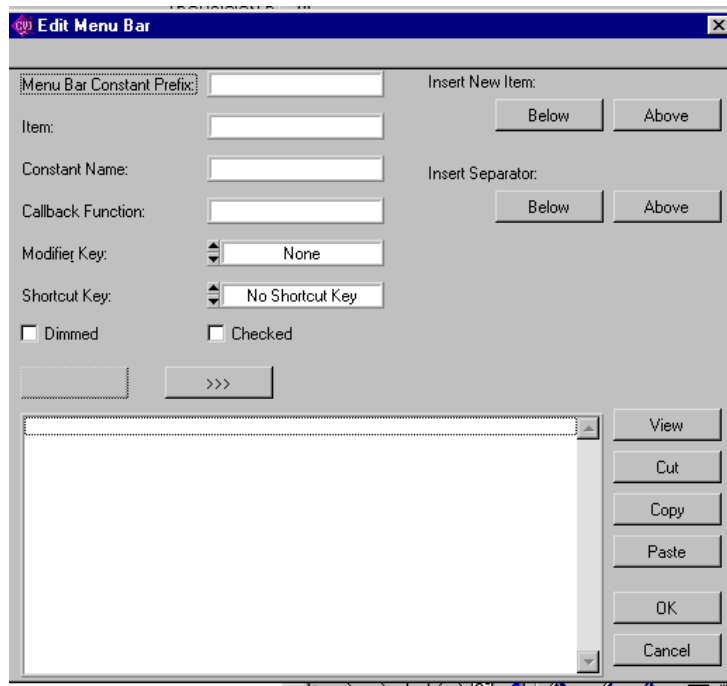
Una vez definida como será la estructura y que realizará cada uno de los items, sólo nos basta ponernos manos a la obra y crear el menú.

Creando las Barras de Menús

En primer lugar y desde la ventana project cargaremos el archivo prueba3.uir del ejemplo anterior visto en el tema 3, una vez abierta la ventana y situándonos dentro de la ventana pero fuera del panel pulsamos el botón derecho y de las dos opciones seleccionamos **Menu Bar**.



1. Nos aparece la ventana de edición del menú y lo primero que debemos hacer es darle un nombre de contante al menú, para nuestro ejemplo le daremos **MENU**.



Nota: hay que decir en este punto que se pueden asociar a un fichero .uir tantos menús como queramos, bastará con etiquetarlos con nombres de constantes distintos. Luego desde el programa podremos cargar menús distintos en situaciones distintas.

2. Desde aquí es desde donde crearemos la estructura del menú . A cada uno de los elementos del menú se les llama **items**, y cada uno de ellos está formado a su vez por las distintos **subitems** y a su vez otros items pueden colgar de los anteriores. Creando un jerarquía dentro del menú.

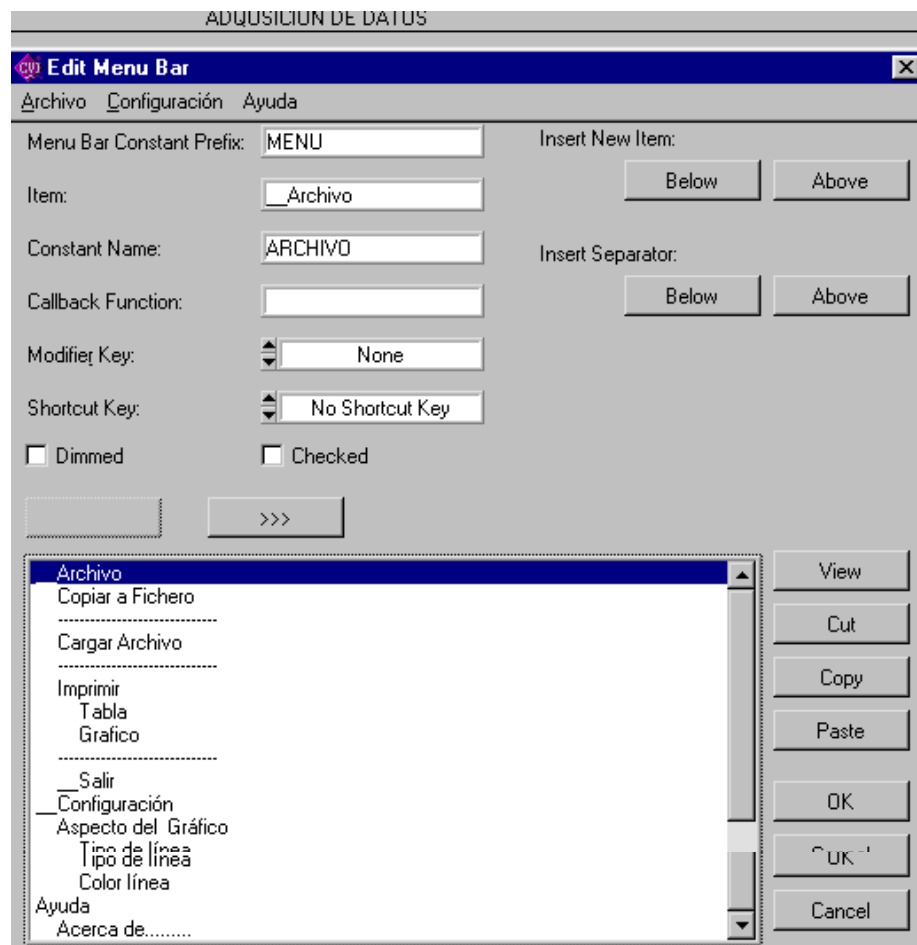
- Crearemos el primer item relleno el campo item: __Archivo .
- Asignaremos un nombre de constante: ARCHIVO.
- Marcaremos el campo Dimmed.
- El resto de campos los dejamos por defecto.

Para ver el efecto que produce nuestro item podemos pulsar con el ratón el botón **VIEW** y automáticamente se insertará en la barra de menúa de la ventana que tenemos abierta **Edit Menu Bar**.

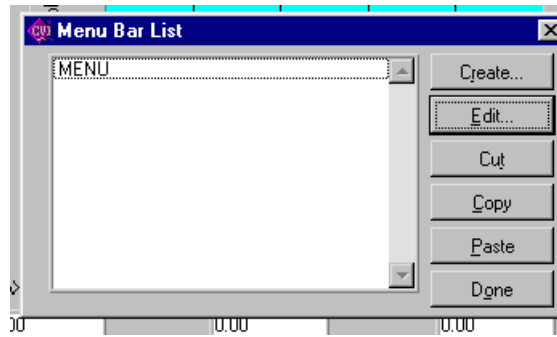
Con los botones Above y Below iremos insertando atrás y adelante del item actual , bien otro item (Insert new Items) o bien un separador (Insert Separator).

Para establecer la jerarquía de los menús basta con pulsar las flechas que aparecen en la ventana, por cada pulsación nos adentramos un grado más en la jerarquía(>>>) o descendemos un grado (<<<).

Crearemos el menú hasta obtener el cuadro de la figura:



Una vez comprobado que el menú presenta el aspecto deseado debemos de pulsar el botón OK y en la ventana que nos aparece pulsar DONE.



Podemos ver el efecto que produce el menú en nuestro panel , si no aparece es debido a que en las propiedades del panel no tenemos activada la posibilidad de que presente barras de menús, si es así deberemos marcar la opción **Title Bar Visible** en **Other Attributes** del editor del panel.

El nuevo panel presentará el siguiente aspecto.

