

PROGRAMA DE INSTRUMENTACIÓN VIRTUAL LABVIEW

USO DE LabVIEW

En este apartado se discuten los aspectos necesarios para familiarizarse con el uso de LabVIEW, incluyendo las ventanas Panel y Diagram, menús de LabVIEW y la ventana de jerarquía.

Asimismo se discuten otros aspectos necesarios como el uso de los modos **edit** y **run**; creación de objetos; herramientas y obtención de ayuda.

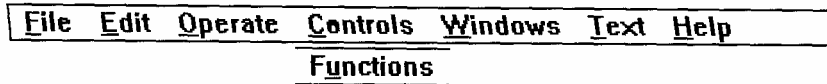
Ventanas Panel y Diagram

Cada VI tiene dos ventanas separadas, pero relacionadas entre sí. La ventana **Panel** contiene el panel frontal de nuestro Vi. La ventana **Diagram** es aquella en la cual se construye el diagrama de bloques. Se puede conmutar entre ambas pantallas con el comando **Show Panel/Show Diagram** (Mostrar Panel/Mostrar Diagrama) del menú **Windows** (Ventanas). Usando los comandos **Tile** (literalmente "baldosas"; podemos traducir por Parcelas), dentro de ese mismo menú, podemos posicionar las ventanas **Panel** y **Diagram** una al lado de la otra o una encima de la otra.

Menús de LabVIEW

La programación en LabVIEW obliga a utilizar con frecuencia los diferentes menús. La barra de menús de la parte superior de la ventana de un Vi contiene diversos menús **pul;-down** (desplegables). Cuando hacemos clic sobre un ítem o elemento de esta barra, aparece un menú por debajo de ella. Dicho menú contiene elementos comunes a otras aplicaciones Windows, como **Open** (Abrir), **Save** (Guardar) y **Paste** (Pegar), y muchas otras particulares de LabVIEW.

La siguiente figura muestra la barra de menús para la versión 3.1 cuando la ventana **Panel** está activa. El menú **Functions** reemplaza al **Controls** cuando la ventana **Diagram** está abierta.



- File** (Archivo) Sus opciones se usan básicamente para abrir, cerrar, guardar imprimir Vis.
- Edit** (Edición) Se usa principalmente para organizar el panel frontal y el diagrama de bloques y establecer nuestras preferencias.
- Operate** (Función) Sus comandos sirven para ejecutar el Vi.
- Controls** (Controles) Con este menú, podemos añadir controles e indicadores al panel frontal. Cada opción dentro de este menú visualiza una paleta con los controles e indicadores para esa opción. El menú **Controis** sólo está disponible cuando la ventana **Panel** está activa.

| | |
|------------------------------|--|
| Functions (Funciones) | Construimos el diagrama de bloques con este menú. Cada opción visualiza una paleta con sus íconos disponibles. El menú Functions sólo está disponible cuando la ventana Diagram está activa. |
| Windows (Ventanas) | Se usa para situar rápidamente las ventanas abiertas y para abrir ventanas de los diferentes subVIs. |
| Text (Texto) | Se utiliza para cambiar la fuente, estilo y color de texto. |
| Help (Ayuda) | Presenta ayuda sobre los diferentes íconos y otros aspectos de LabVIEW. |

En el caso de la versión 4.0, este menú ha quedado de la siguiente manera:

File Edit Operate Project Windows Help

Exactamente igual para las ventanas **Panel** y **Diagram**. Las funciones de estos comandos respecto a la versión 3.1 son:

| | |
|---------------------------|---|
| File (Archivo) | Misma función. |
| Edit (Edición) | Misma función. |
| Operate (Función) | Presenta nuevas opciones como pueden ser la impresión cuando acaba la ejecución. |
| Project (Proyecto) | Presenta los niveles de jerarquía, los subvis que lo integran, los que están sin abrir, busca Vis, etc. |
| Windows (Ventanas) | Se utiliza básicamente para mostrar (Show) ventanas, como pueden ser las de información, historia, controles/funciones, herramientas, portapapeles, etc. |
| Help (Ayuda) | Misma función. |

El menú de LabVIEW que utilizaremos con más frecuencia es el menú **pop-up** (emergente) de objetos, al cual accedemos situando el cursor sobre el objeto en cuestión y pulsando el botón derecho de; ratón. Si la pulsación se hace sobre un espacio vacío, el menú que se obtendrá vendrá en función de la herramienta seleccionada.

Uso de los modos **EDIT** (Edición) y **RUN** (Ejecución)

Podemos crear o cambiar un VI cuando éste está en el modo **Edit**. En él, las herramientas de edición se habilitan en la paleta de; modo **Edit**, por debajo de la barra de; menú de ventana, como



se indica a continuación:

Cuando estamos listos para probar nuestro VI, hacemos clic sobre el botón de modo Q o seleccionamos **Change to Run Mode** (Cambio al Modo de Ejecución) desde el menú **Operate**. Haciendo esto compilamos el Vi y lo ponemos en el modo **Run**. En este punto podemos disponer de las opciones de depuración, ejecución de; VI, diferentes modos de ejecución, impresión de datos, etc.

Si lo que queremos es ejecutar el Vi desde el modo **Edit** sin pasar al modo **Run**, hemos de hacer clic sobre la flecha de ejecución. Si fuese necesario, LabVIEW compilaría primero el VI, después conmuta al modo **Run**, ejecuta el Vi y vuelve al modo **Edit** una vez que el Vi se ha ejecutado.

Éste es uno de los puntos que ha sufrido una mayor modificación en la versión

Los iconos correspondientes a estos modos se indican a continuación:

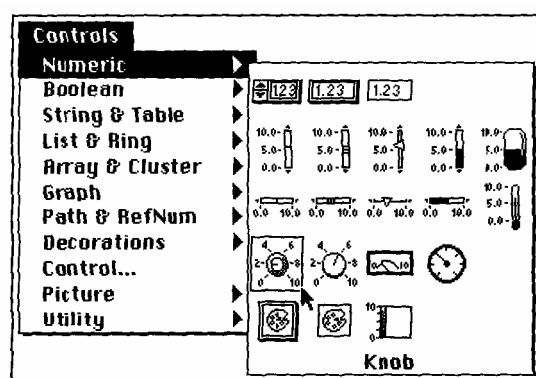
Se puede observar que es aquí donde aparece el tratamiento de los diferentes tipos de letras (en la versión 3.1 lo hacíamos con **Text**) y la alineación y distribución de objetos. Así mismo vemos

que no aparece ninguna herramienta. En la nueva versión se han independizado pasando a tener una ventana propia, a la cual accedemos con **Show Tools Palette** (Mostrar paleta de herramientas) del menú **Windows**.

Otro aspecto a destacar es el botón **Pause** (pausa) IM. Al hacer clic en él se para la ejecución del VI y vamos al diagrama de bloques, parpadeando la siguiente secuencia que se ejecutará.

Creación de objetos

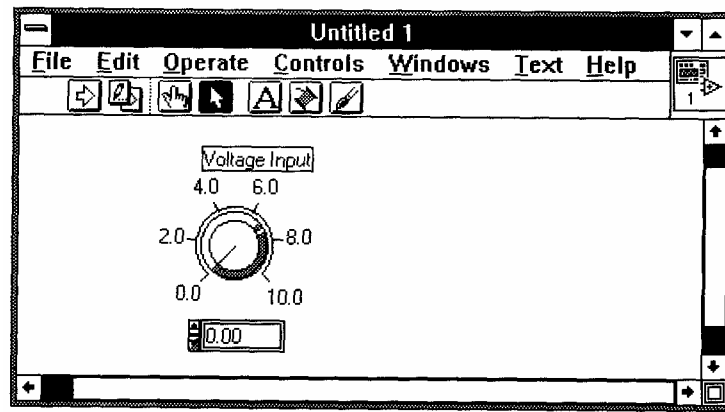
Para elaborar el panel frontal hemos de situar sobre él los objetos deseados mediante su selección desde el menú **Controls**. Creamos objetos sobre el diagrama de bloques seleccionándolos desde el menú **Functions**. Por ejemplo, si queremos crear un knob o botón rotatorio sobre el panel frontal, primero hemos de seleccionarlo desde la paleta **Numeric** (Numérico) del menú **Controls**, como se indica en la siguiente figura 2.2.



El objeto aparecerá en la ventana **Panel** con un rectángulo negro o gris que representa una etiqueta de identificación o **Label**. Si queremos usarla en ese mismo momento, introduciremos el texto desde el teclado. Después de haberlo hecho, cualquiera de las siguientes acciones completa la entrada:

- Pulsar < Shift + Enter >

- Pulsar < Enter > de; teclado numérico.
- Clic sobre el botón Enter en la paleta de herramientas.
- Clic fuera de la etiqueta.



Cuando creamos un objeto sobre el panel frontal, al mismo tiempo se crea el terminal; correspondiente sobre el diagrama de bloques. Este terminal se usa tanto para leer datos desde un control como para enviarlos a un indicador.

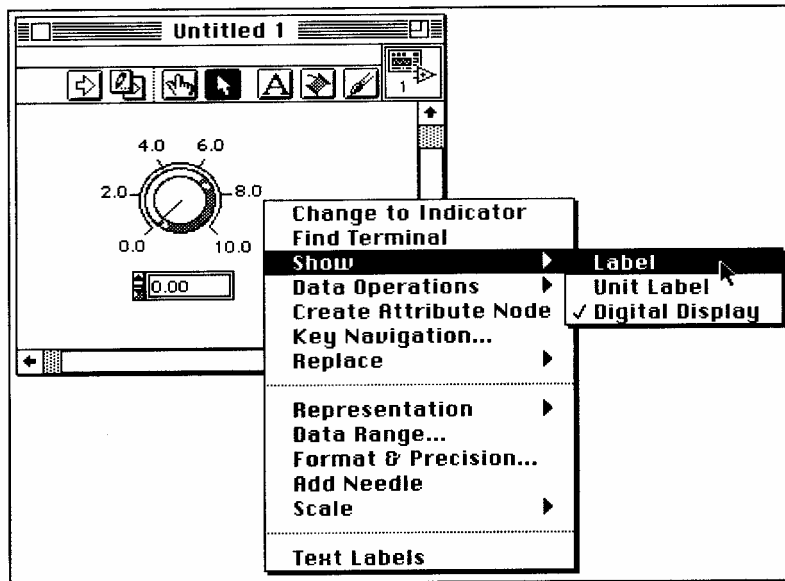
Si se selecciona **Show Diagram** (Mostrar Diagrama) desde el menú **Windows**, podremos ver el diagrama correspondiente al panel frontal. Este diagrama contendrá terminales para todos los controles e indicadores de; panel frontal.



Todos los objetos en LabVIEW tienen asociados menús **pop-up**, los cuales podemos obtener pulsando el botón derecho del ratón sobre dicho objeto. Mediante la selección de sus diferentes opciones podremos actuar sobre determinados parámetros, como el aspecto o comportamiento de ese objeto.

Por ejemplo, si no hubiésemos introducido texto en la etiqueta del control anterior, ésta habría desaparecido al hacer clic en cualquier otro lado. Para volver

A visualizarla tendríamos que obtener el menú **pop-up** de ese control y seleccionar Label de; menú **Show** (figura 2.4).



En la versión 4.0 el proceso en sí es el mismo. La única diferencia es que no tenemos los menús **Controls** y **Functions** en la barra superior, sino que son ventanas flotantes que podemos tener visibles o no. Si no lo están, utilizaremos la opción **Show Controls Palette** (Mostrar paleta de controles) o **Show Functions Palette** (Mostrar paleta de funciones) del menú **Windows**. Una vez visibles actuaremos tal y como se ha indicado para la versión 3.1.

Otra opción es hacer clic con el botón derecho de ratón en cualquier área libre de la pantalla: Aparecerá el menú **Controls** o **Functions** según estemos en la ventana **Panel** o **Diagram**, respectivamente.

HERRAMIENTAS DE LabVIEW

Una herramienta es un modo de funcionamiento especial del ratón. Las usamos para llevar a cabo funciones específicas de edición o ejecución.

- La herramienta **Operating** (Funcionamiento) maneja los controles de panel frontal (y los indicadores en el modo **Edit**). Es la única herramienta disponible en el modo **Run**.
- La herramienta **Positioning 1** (Situación) selecciona, mueve y redimensiona objetos.
- La herramienta **Labeling** [A] (Etiquetado) crea y edita textos.
- La herramienta **Wiring** [Y] (Cableado) enlaza objetos del diagrama de bloques y asigna a los terminales del conector de los controles e indicadores del panel frontal.
- La herramienta **Coloring** (Coloración) colorea diversos objetos y los fondos.

Se puede cambiar de herramienta haciendo lo siguiente:

- Clic sobre el icono de la herramienta que queremos.
- Usando la tecla TAB para seleccionar la siguiente herramienta.
- Pulsando la tecla SPACE para cambiar entre la herramienta **Operating** y **Positioning** cuando la ventana **Panel** está activa, y entre las herramientas **Wiring** y **Positioning** cuando la ventana **Diagram** es la activa.

La versión 4.0 implementa nuevas herramientas y cambia el nombre de las ya existentes:

- **Operate Value** (Valor Operativo) H. Misma función que **Operating**.
- **Position/Size/Select** (Situación/Tamaño / Selección) M. Realiza la misma función que **Positioning**.
- **Edit Text** (Edición de Texto) Misma función que **Labeling**.
- **Connect Wire** (Conexión de Cables) @. Misma función que **Wiring**.

- **Object Popup** (Menú pop-up del objeto) M. Función nueva. Despliega el menú pop-up asociado al objeto. Tiene el mismo efecto que si pulsamos el botón derecho del ratón sobre el objeto.
- **Scroll Window** (Desplazamiento de la pantalla) IJ. Función nueva. Desplaza la pantalla en la dirección que deseemos para ver posibles zonas ocultas.
- **Set/Clear Breakpoint** (Establecer/Quitar puntos de ruptura) ál. Función nueva. Permite poner tantos puntos de ruptura como deseemos a lo largo del diagrama de bloques. Cuando durante la ejecución se llega a uno de ellos, LabVIEW conmuta automáticamente al diagrama de bloques. Usamos esta misma herramienta para quitar los puntos.
- **Probe Data** (Sonda de datos) 3. Funciona como la opción **Probe** de la versión 3.1

TIPOS DE DATOS EN LABVIEW. CONTROLES E INDICADORES

LabVIEW ofrece una gran variedad de tipos de datos con los que podemos trabajar respondiendo a las necesidades reales con las que nos encontraremos. Uno de los aspectos más significativos de LabVIEW es la diferenciación que efectúa en el diagrama de bloques entre los diferentes tipos de controles o indicadores, basada en que cada uno de ellos tiene un color propio.

De esta manera, y como consecuencia de una memorización o asimilación práctica, nos será muy fácil identificarlos y reconocer inmediatamente si estamos trabajando con el tipo de datos adecuado. Distinguimos los siguientes tipos, los cuales pueden funcionar tanto como controles como indicadores (entre paréntesis queda reflejado el color con el que queda representado en el diagrama de bloques):

Boolean (verde claro)

Los tipos de datos booleanos son enteros de 16 bits. El bit más significativo contiene el valor Booleano. Si el bit 15 se pone a 1, entonces el valor del control o indicador es **true** (verdadero); por el contrario, si este bit 15 vale 0, el valor de la

variable booleana será **false** (falso).

Numéricos: Hay diferentes tipos

Extended (naranja)

Según el modelo de ordenador que estemos utilizando los números de coma flotante con precisión extendida presentan el siguiente formato:

Macintosh: 96 bits (formato precisión extendida MC68881 - MC68882)

Windows: 80 bits (formato precisión extendida 80287)

Sun: Formato 128 bits

HP-UX: Son almacenados como los números en coma flotante de doble precisión.

Double (naranja)

Los números en coma flotante de doble precisión cumplen con el formato de doble precisión IEEE de 64 bits. Es el valor por defecto de LabVIEW.

Single (naranja)

Los números en coma flotante de precisión simple cumplen con el formato de precisión simple IEEE de 32 bits.

Long Integer (azul)

Los números enteros largos tienen un formato de 32 bits, con o sin signo.

Word Integer (azul)

Estos números tienen un formato de 16 bits, con o sin signo.

Byte Integer (azul)

Tienen un formato de 8 bits, con o sin signo.

Unsigned Long (azul) Entero largo sin signo.

Unsigned Word (azul) Palabra sin signo.

Unsigned Byte (azul) Byte sin signo.

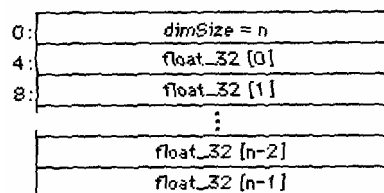
Complex Extended (naranja) Número complejo con precisión extendida.

Complex Double (naranja) Complejo con precisión doble.

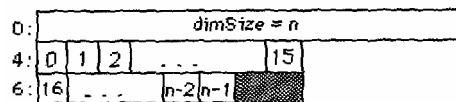
Complex Single (naranja) Complejo con precisión simple.

Arrays (depende del tipo de datos que contenga)

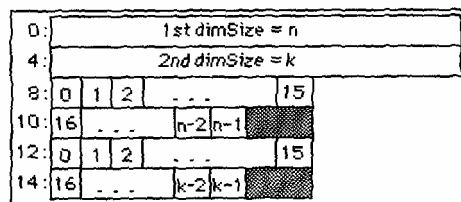
LabVIEW almacena el tamaño de cada dimensión de un array como **long integer** seguido por el dato. El ejemplo que sigue muestra un array unidimensional con números en coma flotante de precisión simple. Los números decimales a la izquierda presentan el desplazamiento donde empieza cada array en la posición de memoria.



Los arrays booleanos se almacenan de manera diferente a los booleanos escalares. Estos arrays se almacenan como bits empaquetados. El tamaño de la dimensión viene dado en bits en lugar de bytes. El bit 0 se guarda en la posición más alta de memoria (215), y el bit 15 en la posición más baja (20).

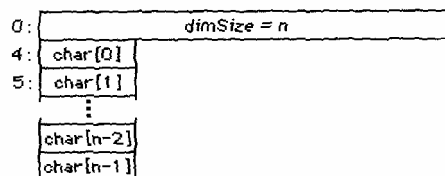


La figura 3.1 muestra un ejemplo de un array booleano bi-dimensional. El elemento 0 de cada dimensión se almacena en una nueva palabra entera ignorándose los bits sin usar de las



dimensiones previas.

Strings (rosa) LabVIEW almacena los strings como si fueran un array un_i-dimensional de bytes enteros (caracteres de 8 bits).



Handies Un handie es un puntero que apunta a un bloque de memoria relocizable. Un handie sólo apunta a datos definidos por el usuario. LabVIEW no reconoce qué es lo que hay en ese bloque de memoria. Es especialmente útil para pasar un bloque de datos por referencia entre nodos de interficie de código (Code Interface Nodes o CiNs).

Paths (verde oscuro)

LabVIEW almacena las componentes tipo y número de un path en palabras enteras, seguidas inmediatamente por las componentes de path. El tipo de path es 0 para un path absoluto y 1 para un path relativo. Cualquier otro valor indicaría que el path no es válido. Cada componente del path es una cadena Pascal (P-string), en la cual el primer byte es la longitud de la P-string (sin incluir el byte de longitud).

Clusters (marrón o rosa) Un cluster almacena diferentes tipos de datos de acuerdo a las siguientes normas:

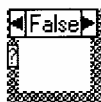
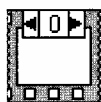
Los datos escalares se almacenan directamente en el cluster; los arrays, strings, handies y paths se almacenan indirectamente. El cluster almacena un handie que apunta al área de memoria en la que LabVIEW ha almacenado realmente los datos.

Para conectar terminales se usa la herramienta **Wiring** (cableado).

PROGRAMACIÓN ESTRUCTURAL

A la hora de programar, muchas veces es necesario ejecutar un mismo conjunto de sentencias un número determinado de veces, o que éstas se repitan mientras se cumplan ciertas condiciones. También puede ocurrir que queramos ejecutar una u otra sentencia dependiendo de las condiciones fijadas o simplemente forzar que unas se ejecuten siempre antes que otras.

Para ello LabVIEW dispone de cuatro estructuras fácilmente diferenciables por su apariencia y disponibles en la opción **Structures** de menú **Function** de la ventana **Diagram**:



ESTRUCTURAS ITERATIVAS: FOR LOOP Y WHILE LOOP

FORLOOP

Usaremos **For Loop** cuando queramos que una operación se repita un número determinado de veces. Su equivalente en lenguaje convencional es:

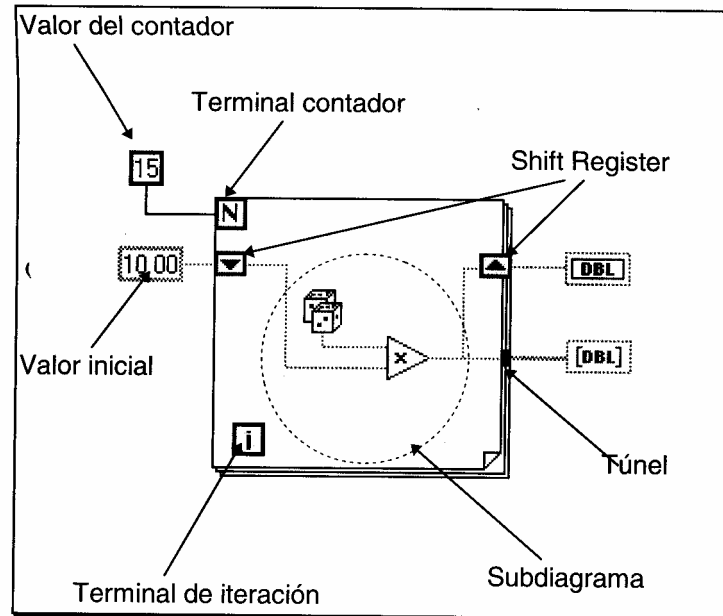
For i = 0 to N-1
Ejecuta subdiagrama

Al colocar un **For Loop** en la ventana **Diagram** observamos que tiene asociados dos terminales:

1.- Terminar contador: Contiene el número de veces que se ejecutará el subdiagrama creado en el interior de la estructura. El valor del contador se fijará externamente (ver también **Arrays** en el capítulo 6).

2.- Termina; de iteración: Indica el número de veces que se ha ejecutado la estructura: Cero durante la primera iteración, uno durante la segunda y así hasta N-1.

Ambos terminales son accesibles desde el interior de la estructura, es decir, sus valores podrán formar parte del subdiagrama pero en ningún caso se podrán modificar.



WHILE LOOP

Usaremos **While Loop** cuando queramos que una operación se repita mientras una determinada condición sea cierta. Su equivalente en lenguaje convencional es:

Do ejecutar subdiagrama

While condición **is TRUE**

(Aunque esta estructura es más similar al comando *Repeat-Until*, ya que se repite como mínimo una vez, independientemente del estado de la condición).

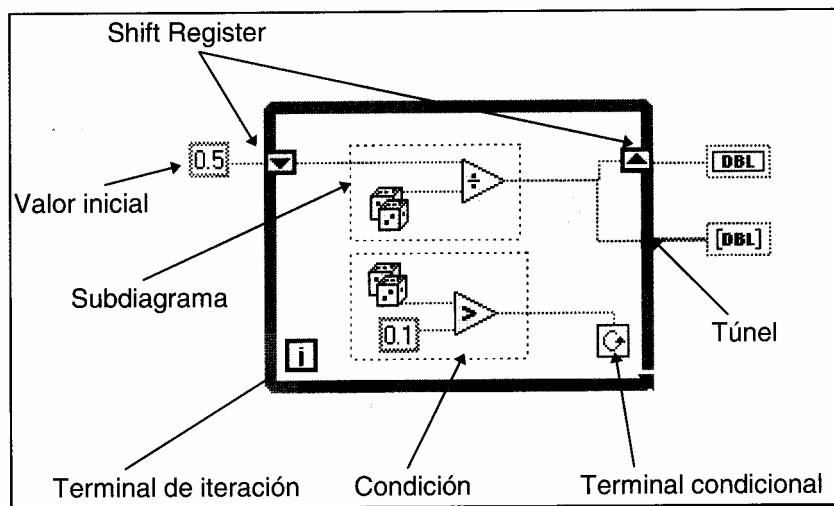
Al igual que **For Loop** contiene dos terminales:

1.-Terminal condicional: A él conectaremos la condición que hará que se ejecute el subdiagrama. LabVIEW comprobará el estado de este terminal; al final de cada iteración, si su valor es TRUE (verdadero) continuará, pero por el contrario si su valor es FALSE (falso) detendrá la ejecución.

2.- Terminal de iteración: Indica el número de veces que se ha ejecutado el bucle y que, como mínimo, siempre será una ($i=0$).

Al hacer pop-up tanto en el **For Loop** como en el **While Loop** se despliega el siguiente menú:
(La versión 4.0 presenta además la opción de Ayuda en Línea u **Online Help**).

- Show Label: Oculta o visualiza la etiqueta de identificación de; **Loop y**, si no existe, permite ponerla.
- Description: Permite añadir comentarios.
- Replace: Cambia el **For Loop** o el **While Loop** por cualquier otra función de la paleta **Structs & Constants**.
- Remove Loop: Borra la estructura **While** o **For** pero sin eliminar el subdiagrama de su interior.
- Add Shift Register: Añade los **shift register** (registros de desplazamiento).



REGISTROS DE DESPLAZAMIENTO

Los registros de desplazamiento o **shift register** son variables locales, disponibles tanto en el **For Loop** como en el **While Loop**, que permiten transferir los valores de; final de una iteración al principio de la siguiente.

Inicialmente **shift register** tiene un par de terminales colocados a ambos lados de; **Loop**; el terminal de la derecha almacena el valor final de la iteración hasta que una nueva hace que este valor se desplace al terminal de la izquierda, quedando en el de la derecha el nuevo valor. Un mismo registro de, desplazamiento puede tener más de un terminal en el lado izquierdo; para añadirlo escogeremos la opción **Add Element** (añadir elemento) de; menú pop-up. Cuantos más terminales tengamos en el lado izquierdo más valores de iteraciones anteriores podremos almacenar.

El menú pop-up tiene otros dos comandos:

- Remove element: Borra un terminal de; lado izquierdo siempre y cuando el registro de desplazamiento tenga asociado más de uno.
- Remove All: Borra todo el registro de desplazamiento, tanto los terminales de la izquierda como el de la derecha.

Un mismo **Loop** puede tener varios registros de desplazamientos siendo conveniente inicializarlos, para que los terminales de la izquierda tengan el valor deseado cuando se produzca la primera iteración. **Shift register** puede trabajar con cualquier tipo de datos siempre y cuando los datos que se conecten a cada terminal sean de; mismo tipo.

Al finalizar la ejecución de todas las iteraciones el último valor quedará en el terminal de la derecha; uniéndolo a un indicador de; mismo tipo de dato fuera de; **Loop** podremos obtener su valor.

Pero existe otra posibilidad para pasar datos de forma automática desde el interior de la estructura al exterior. Cuando un cable atraviesa los límites de; **Loop**, aparece en el borde un nuevo terminal llamado túnel que hace de conexión entre el interior y el exterior, de forma que los datos

fluyen a través de él después de cada iteración de; **Loop**, pudiendo guardar de esta manera no sólo el último valor de todas las iteraciones sino también los valores intermedios. A esta posibilidad que tienen tanto el **For** como el **While** de acumular arrays en sus límites automáticamente se le llama **auto-indexing** o autoindexado.

LabVIEW habilita por defecto **auto-indexing** en el **For Loop** ya que es más frecuente utilizar esta estructura para crear arrays que no el **While Loop**, en el cual esta opción está deshabilitada por defecto y cuya utilización podría provocar

4 PROGRAMACIÓN ESTRUCTURADA

problemas de memoria debido a que no sabemos cuantas veces se va a ejecutar. No obstante, haciendo pop-up en el túnel se puede habilitar o deshabilitar esta opción.

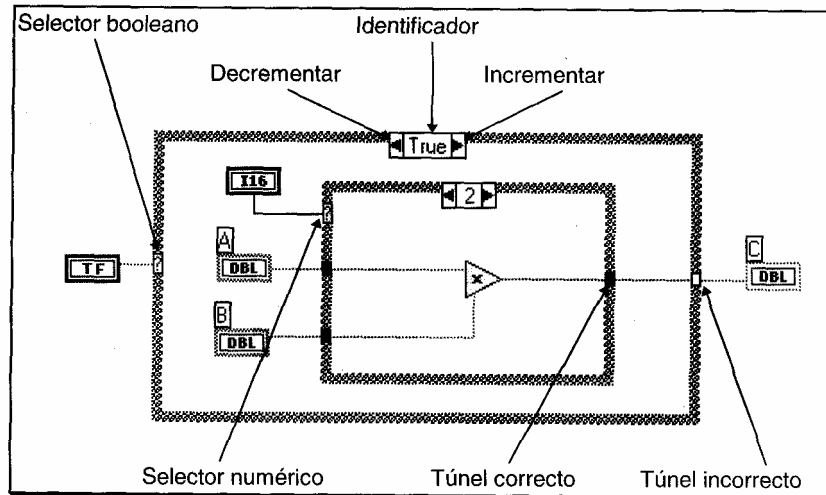
ESTRUCTURAS CASE Y SEQUENCE

Este tipo de estructuras se diferencia de las iterativas en que puede tener múltiples subdiagramas, de los cuales solamente uno es visible a la vez. En la parte superior de cada estructura existe una pequeña ventana que muestra el identificador de; subdiagrama que se está mostrando. A ambos lados de esta ventana existen dos botones que decrementan o incrementan el identificador de forma que podamos ver el resto de subdiagramas.

CASE

Usaremos la estructura **Case** en aquellas situaciones en las que el número de alternativas disponibles sean dos o más. Según qué valor tome el selector dentro de los n valores posibles, se ejecutará en correspondencia uno de los n subdiagramas.

La estructura **Case** consta de un terminal llamado selector y un conjunto de subdiagramas, cada uno de los cuales está dentro de un case o suceso y etiquetado por un identificador del mismo tipo que el selector; éste será booleano o numérico. Si se conecta un valor booleano al selector, la estructura tendrá dos **Case:** False y True. Pero si se conecta un valor numérico la estructura podrá tener hasta 214 **Case**.



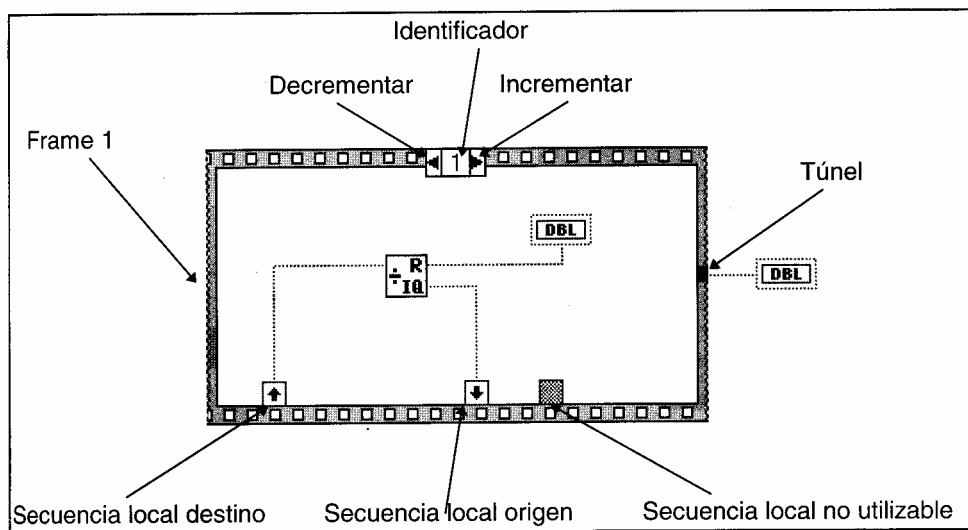
En este caso la estructura **Case** engloba dos sentencias diferentes de otros lenguajes convencionales:

- 1.- If condición true then ejecutar case true
else ejecutar case false
- 2.- Case selector of
l:ejecutar case 1;
n:ejecutar case n
end

Case no cuenta con los registros de desplazamiento de las estructuras iterativas pero sí podemos crear los túneles para sacar o introducir datos. Si un case o suceso proporciona un dato de salida a una determinada variable será necesario que todos los demás también lo hagan; si no ocurre de esta manera será imposible ejecutar el programa.

SEQUENCE

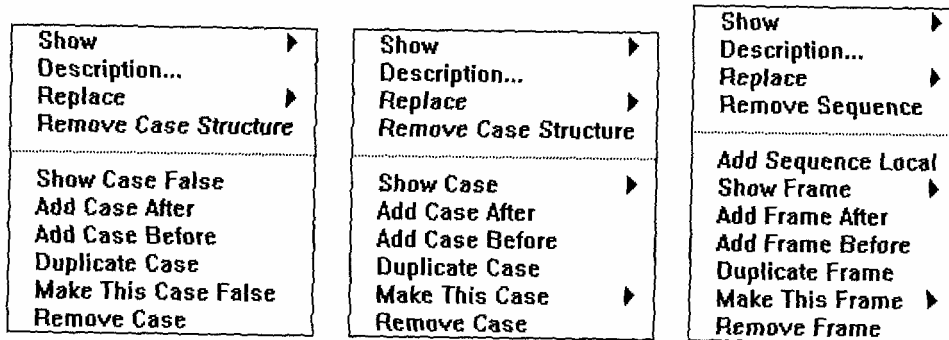
Esta estructura no tiene su homóloga en los diferentes lenguajes convencionales, ya que en éstos las sentencias se ejecutan en el orden de aparición pero, como ya sabemos, en LabVIEW una función se ejecuta cuando tiene disponible todos los datos de entrada. Se produce de esta manera una dependencia de datos que hace que la función que recibe un dato directa o indirectamente de otra se ejecute siempre después, creándose un flujo de programa.



Pero existen ocasiones en que esta dependencia de datos no existe y es necesario que un subdiagrama se ejecute antes que otro; es en estos casos cuando usaremos la estructura **Sequence** para forzar un determinado flujo de datos. Cada subdiagrama estará contenido en un **frame** o marco y estos se ejecutarán en orden de aparición: Primero el *frame 0* o *marco 0*, después el *frame 1* y así, sucesivamente, hasta el último.

Al contrario de; **Case**, si un **frame** aporta un dato de salida a una variable los demás no tendrán por qué hacerlo. Pero tendremos que tener en cuenta que el dato estará solamente disponible cuando se ejecute el último **frame** y no cuando se ejecute el **frame** que transfiere el dato.

Debido a la similitud de los menús pop-up de la estructuras **Case** y **Sequence** vamos a estudiarlos de forma conjunta indicando en cada caso las posibles diferencias que puedan existir:

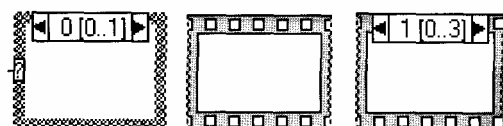


- **Show Label:** Oculta o visualiza la etiqueta de identificación de la estructura y, si no existe, permite ponerla.
- **Description:** Permite añadir comentarios.
- **Replace:** Cambia la estructura **Case** o **Sequence** por cualquier otra función de la paleta **Structs & Constants**.
- **Remove Case Structure** o **Sequence:** Borra completamente la estructura **Case** o **Sequence** y todos los subdiagramas menos el que se esté visualizando en el momento de la ejecución de este comando.
- **Add Sequence Local** (añadir secuencia local): Esta opción está sólo disponible en el menú de la estructura **Sequence** y se utiliza para pasar datos de un **frame** a otro. Una pequeña flecha con la punta hacia el exterior de la estructura indica el **frame** de origen de la secuencia local, mientras que una flecha apuntando hacia el interior indica que la secuencia

local contiene un dato de salida. Todos los **frames** posteriores al que contiene la secuencia local que origina el dato podrán disponer de él, no siendo así para los **frames** anteriores en los cuales aparecerá un cuadrado vacío que indicará que los datos no están disponibles.

- **Show Case o Show Frame:** Nos permite ir directamente al subdiagrama que queremos visualizar sin tener que pasar por todos los case o frame intermedios que pudiera haber. Al pulsar esta opción, un menú conteniendo todos los identificadores se desplegará y sólo tendremos que señalar con el cursor de ratón el que deseamos ver. Si sólo hubiese dos subdiagramas nos aparecerá directamente el nombre de único identificador que podemos visualizar, como es el caso de case con selector booleano.
- **Add Case After o Add Frame After:** Este comando inserta un subdiagrama vacío inmediatamente después de que se está visualizando.
- **Add Case Before o Add Frame Before:** Inserta un subdiagrama vacío justo un nivel por encima de que se está visualizando.
- **Duplicate Case o Duplicate Frame:** Inserta una copia de subdiagrama visible inmediatamente después de él.
- **Make This Case o Make This Frame:** Mueve un subdiagrama a otra posición.
- **Remove Case o Remove Frame:** Borra el subdiagrama visible. Este comando no está disponible si solamente existe un **case** o un **frame**.

La versión 4.0 presenta una ligera mejora, pero que resulta muy útil. Se puede ver en el caso de las estructuras **Sequence** y **Case** numérico. En el primer caso, si sólo hay una secuencia, no aparece ningún identificador de **frame**; mientras que si hay más de uno, se nos indica en cuál estamos y cuántos hay. Lo mismo pasa con la estructura **Case**, sólo que, en este caso tendremos, como mínimo, dos posibles estados. Todo ello queda reflejado a continuación:



Formula Node o nodo de fórmula es una función de características similares a las estructuras vistas anteriormente, disponible en la paleta **Structs & Constants** de; menú **Functions**, pero que, en lugar de contener un subdiagrama, contiene una o más fórmulas separadas por un punto y coma. Usaremos **Formula Node** cuando queramos ejecutar fórmulas matemáticas que serían complicadas de crear utilizando las diferentes herramientas matemáticas que LabVIEW incorpora en sus librerías.

Una vez escrita la fórmula en el interior del rectángulo sólo tendremos que añadir los terminales que harán la función de variables de entrada o de salida; para ello desplegaremos el menú pop-up de la estructura y ejecutaremos el comando **Add Input** (añadir entrada) o **Add Output** (añadir salida).

Menú pop-up Formula Node

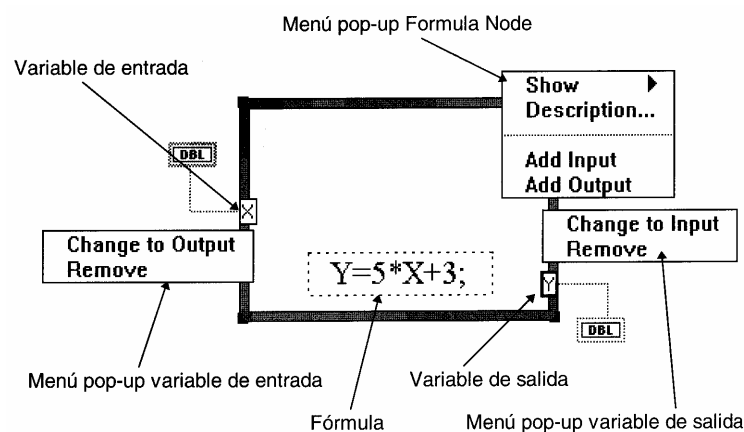


Figura 4.5 Formula Node

Cada variable, además, tendrá otro menú pop-up que permitirá definirla como de salida si anteriormente era de entrada, o de entrada si en un principio era de salida (**Change to Output** o Cambiar a Salida, **Change to Input** o Cambiar a Entrada). También podremos eliminarla mediante el comando **Remove**.

No hay límite para el número de variables o de fórmulas y nunca podrá haber dos entradas o dos salidas con el mismo nombre, aunque una salida sí podrá tener el mismo nombre que una entrada. Todas las variables de salida deberán estar asignadas a una fórmula por lo menos una vez.

La tabla muestra algunas de las funciones de **Formula Node**:

| | |
|-----------|--|
| abs(x) | Devuelve el valor absoluto de x. |
| acos(X) | Calcula el coseno inverso de x en radianes.,l |
| acosh(x) | Calcula el coseno hiperbólico inverso en radianes. |
| asin(x) | Calcula el seno inverso de x en radiahes. |
| asinh(x) | Calcula el seno hiperbólico inverso en radianes. |
| atan(x,y) | Calcula la tangente inversa de y/x en radianeg.,. |
| atanh(x) | Calcula la tangente hiperbólico inversa en radianes. |
| COS(X) | Calcula el coseno de x en radianes. |
| cosh(x) | Calcula el coseno hiperbólico de x en radianes. |
| cot(x) | Calcula la cotangente de x en radianes. |
| CSC(X) | Calcula la cosecante de x en radianes. |
| exp(x) | Calcula el valor de e elevado a x. |
| ln(x) | Calcula el logaritmo natural de x. |

| | |
|------------------------|--|
| <code>log(x)</code> | Calcula el logaritmo de x en base 10. |
| <code>log2(X)</code> | Calcula el logaritmo de x en base 2. |
| <code>max(X,Y)</code> | Compara x con y, y devuelve el mayor valor. |
| <code>min(x,y)</code> | Compara x con y, y devuelve el menor valor. |
| <code>mod(x@y)@</code> | Calcula el cociente de x/y. |
| <code>rando</code> | Genera un número aleatorio entre 0 y 1. |
| <code>sic(x)</code> | Calcula la secante de x en radianes. |
| <code>sign(x)</code> | Devuelve 1 si x es mayor que 0, 0 si x es igual a 0 y -1 si x es menor que cero. |
| <code>sin(x)</code> | Calcula el seno de x en radianes. |
| <code>sinc(x)</code> | Calcula el seno de x dividido por x en radianes. |
| <code>sinh(X)@</code> | Calcula el seno hiperbólico de x en radianes. |
| <code>sqrt(x)</code> | Calcula la raíz cuadrada de x. |
| <code>tan(x)</code> | Calcula la tangente de x en radianes. |
| <code>tanh(x)</code> | Calcula la tangente hiperbólico de x en radianes. |

VARIABLES LOCALES Y GLOBALES

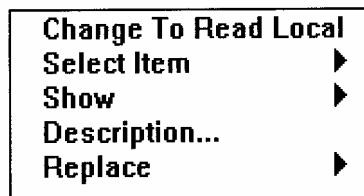
Las variables son imprescindibles en cualquier tipo de problemas, ya que permiten almacenar la información necesaria para su resolución.

En LabVIEW todos los controles introducidos en el Panel Frontal que generan un terminal en la ventana Diagrama van a ser variables, identificadas por el nombre asignado en la etiqueta. Pero puede ocurrir que queramos utilizar el valor de cierta variable en otro subdiagrama o en otro Vi o, simplemente, que queramos guardar un resultado intermedio. La forma mas sencilla de hacerlo es generando variables locales y/o globales dependiendo de la aplicación.

VARIABLES LOCALES

En las variables locales los datos se almacenan en algunos de los controles o indicadores existentes en el Panel Frontal de; Vi creado; es por eso que estas variables **no** sirven para intercambiar datos entre VIs. La principal utilidad de estas variables radica en el hecho de que una vez creada la variable local no importa que proceda de un indicador o de un control, ya que se podrá utilizar en un mismo Diagrama tanto de entrada como de salida.

Las variables locales están disponibles en el menú **Structs & Constants** de la paleta **Function** y disponen de; siguiente menú pop-up:



- Change To Read Local o Change To Write Local: Permite escoger entre leer o escribir en el control.
- Select Item: Visualiza una lista con el nombre de todos los controles existentes en el Panel Frontal y de ella escogeremos el control al cual queremos que haga referencia nuestra variable. Es por esto que para poder crear la variable local será imprescindible que el control tenga asignado un nombre de identificación. Una vez creada la variable local, si en algún momento se cambia el nombre del control origen, será necesario cambiar también el nombre de la variable local ya que LabVIEW no actualiza los cambios.

- Show Label: Muestra una etiqueta con el nombre del Vi al que pertenece la variable local.
- Description: Permite añadir comentarios.
- Replace: Sustituye la variable local por cualquier otra función.

VARIABLES GLOBALES

Las variables globales son un tipo especial de VI, que únicamente dispone de Panel Frontal, en el cual se define el tipo de dato de la variable y el nombre de identificación imprescindible para después podernos referir a ella.

Cuando escogemos la función **Global** de; menú **Structs & Constants** creamos un nuevo terminal en el Diagrama; este terminal corresponde a un VI que inicialmente no contiene ninguna variable. Para poderias añadir haremos doble clic en el terminal y se abrirá el panel frontal. Una vez abierto, las variables se definen igual que cualquier control o indicador de un VI normal. Podemos crear un Vi para cada variable global o definirlas todas en el mismo, que es la opción más indicada para cualquier aplicación. Cuando terminemos de colocar todas las variables grabaremos el VI y lo cerraremos. Si una vez cerrado queremos añadir nuevas variables, bastará con volverlo a abrir e introducir los cambios necesarios. Para añadir nuevos terminales que hagan referencia a las variables globales creadas, no volveremos a ejecutar la función *Global* ya que esto crearía un nuevo Vi sino que abriremos el ya existente mediante el comando VI.. del menú *Función* y seleccionaremos la variable en concreto a través del comando *Select Item* del menú pop-up. Además, este mismo menú cuenta con otra opción que nos permite utilizar una variable ya creada para leer datos o para almacenarlos: Se trata del comando *Change To Read Global o Change To Write Global*

ATTRIBUTE NODE

Los **attribute nodes** o nodos de atributos se pueden considerar como variables que dependen únicamente del terminal a partir del cual se han creado y que permiten leer o modificar atributos del panel frontal de un control o indicador como, por ejemplo, cambiarlo de color, hacerlo invisible, desactivarlo, leer posiciones de cursores, cambiar escalas, etc. Para crear un **attribute node** basta con seleccionar la opción *Create Attribute Node* del menú pop-up de cualquier control del Panel Frontal o terminal del Diagrama de Bloques (en la versión 4.0 primero desplegamos el menú pop-up del objeto y a continuación tomamos la opción **Create**. Podremos crear un **attribute node** o variable local). Una vez creado aparece en el Diagrama un nuevo nodo que puede ser tanto de escritura como de lectura. Una pequeña flecha a la izquierda de cada nodo indica que éste es de escritura, mientras que una flecha a la derecha indica que es de lectura. Además los **attribute nodes** tienen su propio menú pop-up como se muestra a continuación.

- **Change All To Read** o **Change All To Write**: Dependiendo de si el nodo es de escritura o de lectura aparecerá una opción u otra que nos permitirá cambiar entre ambas. Debido a que un mismo **attribute node** puede tener más de un terminal usaremos esta opción cuando queramos que todos ellos sean de escritura o de lectura.

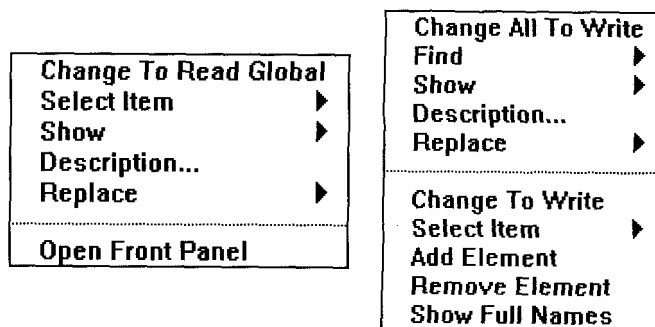
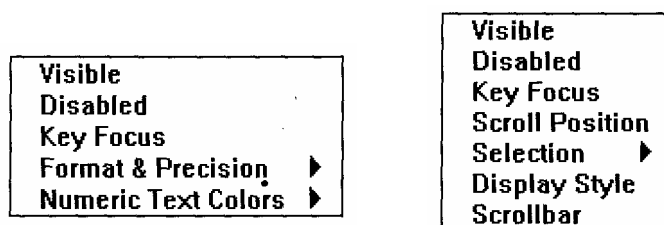


Figura 4.6 Menú pop-up de una variable global

- **Find Control**: Encuentra el control asociado a dicho **attribute node** en el Panel Frontal.
- **Find Terminal**: Encuentra el terminal asociado a dicho **attribute node** en el Diagrama de Bloques.

- Find Attribute Nodes: Muestra todos los **attribute nodes** existentes en el Diagrama y que están asociados a dicho control.
- Show Label: Oculta o visualiza la etiqueta identificativa de; **atribuye node**.
- Description: Permite añadir comentarios.
- Replace: Sustituye el **attribute node** por cualquier otra función.
- Change To Read o Change To Write: Cambia a modo de escritura o de lectura únicamente el terminaj; seleccionado dejando los demás tal y como estaban.
- Select Item: Visualiza todos los atributos disponibles para el control asociado al **attribute node** y permite cambiar un atributo por otro diferente. Podemos acceder directamente a esta opción colocándonos encima de; atributo que deseamos cambiar y pulsando el botón izquierdo del ratón.

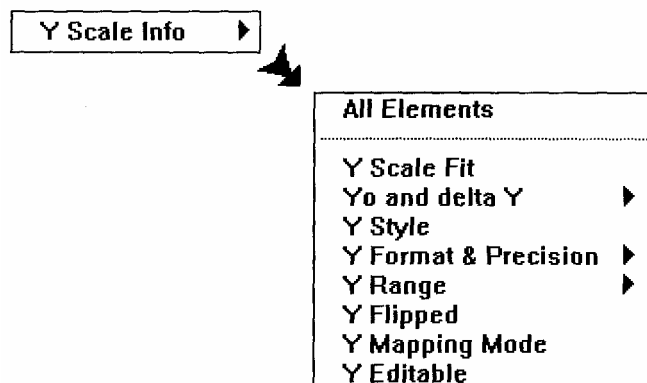
Por ejemplo, los atributos para un control numérico son:



Y para un string :

Remove Element.- Borra el terminaj; seleccionado.

Add Element. Añade un nuevo terminal.

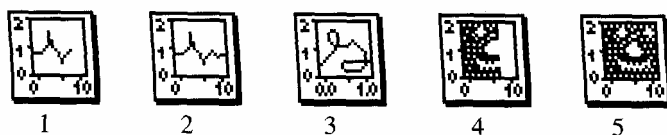


Algunos controles como los **graph** tienen un gran número de atributos. Muchos de estos atributos se agrupan en categorías, como es el caso de **Y Scale Info** para un indicador **XY Graph**. Una pequeña flecha a la derecha del atributo nos indica que se trata de una categoría.

Se pueden seleccionar todos los atributos de una categoría de una sola vez mediante el comando **All elements** (todos los elementos), aunque también podemos seleccionarlos individualmente escogiendo el atributo específico.

INDICADORES GRAFICOS

En muchas ocasiones es necesario para una mayor comprensión de los resultados obtenidos representarlos gráficamente. Para ello LabVIEW dispone de cinco tipos de gráficos accesibles desde el menú *Controls* del panel Frontal bajo el ítem *Graph*, divididos en dos grupos: Los *indicadores chart* y los *indicadores graph*.



Un indicador *graph* o indicador gráfico es una representación bidimensional de una o más gráficas. El *graph* recibe los datos como un bloque. Un indicador *chart* de trazos también muestra gráficas, pero éste recibe los datos y los muestra punto por punto o array por array, reteniendo un cierto número de puntos en pantalla mediante un buffer disponible para ello.

INDICADORES CHART

WAVEFORM CHART

Waveform chart es un tipo especial de indicador numérico que muestra una o mas gráficas, reteniendo en pantalla un cierto número de datos definido por nosotros mismos. Los nuevos datos se añaden al lado de los ya existentes, de forma que se pueden comparar entre ellos.

Los datos se pueden pasar uno a uno al *chart* o mediante arrays, Evidentemente es mucho conveniente pasar múltiples puntos a la vez ya que de esta manera sólo es necesario redibujar la gráfica una vez y no una por cada punto (figura 5.1).

Es posible dibujar varias gráficas en un mismo *chart*, uniendo los datos de cada gráfica en un cluster de escalares numéricos de forma que cada escalar que contiene el cluster se considera como un punto de cada una de las gráficas para una misma abscisa. Se puede ahorrar tiempo uniando los clusters en arrays y después pasando todo el array a la gráfica.

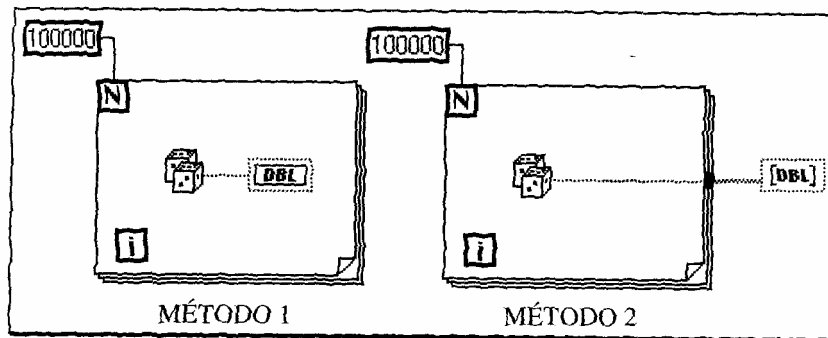
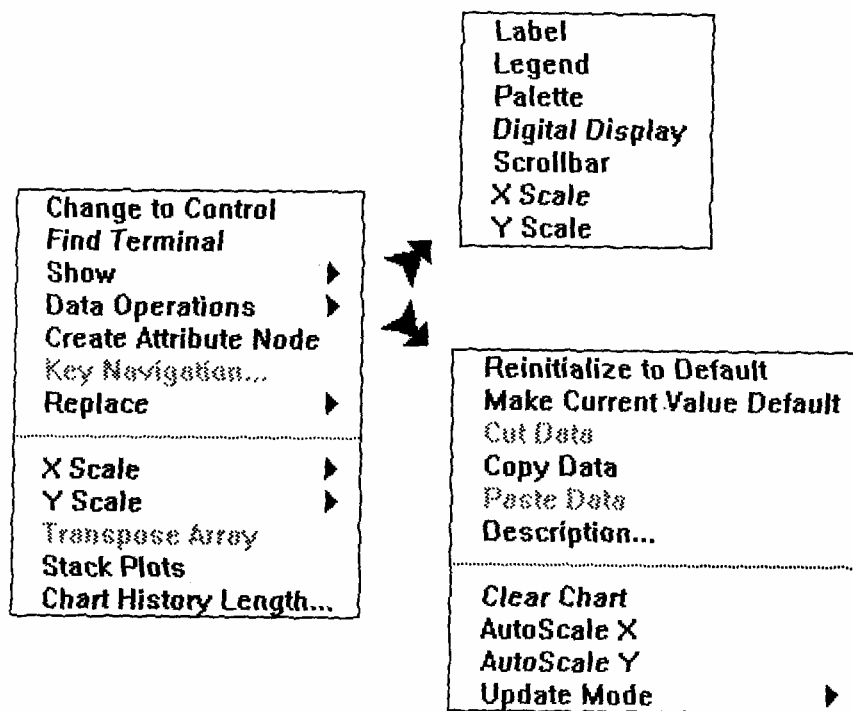


Figura 5.1

Desplegando el menú pop-up se tiene acceso a las siguientes opciones:



- **Change to Control o Change to Indicator:** Dependiendo de si la waveform es un control o un indicador nos permitirá cambiar entre ellas.
- **Find Terminal:** Muestra el terminal asociado en el Diagrama de bloques.
- **Show Label:** Permite poner una etiqueta de identificación a la waveform chart y, si ya existe, la visualiza.
- **Show Legend:** Permite poner una etiqueta de identificación a cada una de las gráficas.
- **Show Palette:** Activa una paleta que permite hacer zooms, desplazar la gráfica de forma rápida, ajustar automáticamente la escala de los ejes, cambiar el formato y la precisión de los indicadores numéricos y elegir entre escala lineal o logarítmica.
- **Show Digital Display:** Es un indicador que muestra el último valor que se ha cargado en pantalla. Hay un indicador por cada gráfica.

- Show Scrolibar: Permite ver los valores anteriores contenidos en el buffer.
- Show X Scale: Visualiza la escala de; eje de abcisas.
- Show Y Scale: Visualiza la escala de; eje de ordenadas.
- Reinitialize to Default: Actualiza el último punto obtenido al valor por defecto.
- Make Current Value Default: Convierte el último punto obtenido en el valor por defecto.
- Description: Permite añadir comentarios.
- Clear Chart: Borra el contenido de; buffer.
- AutoScale X: Ajusta de forma automática el rango de valores de X para una correcta visualización.
- AutoScale Y: Ajusta de forma automática el rango de valores de Y para una correcta visualización.
- Update Mode: Permite escoger entre tres modos de visualizar los nuevos strip chart, scope chart y sweep chart. El modo strip chart es el modo por defecto y consiste en que cada nuevo valor se coloca a la derecha del display, mientras que valores anteriores se desplazan hacia la izquierda. En el modo scope chart cada nuevo valor se coloca a la derecha de; anterior, empezando por el margen izquierdo del display. Cuando se llega al margen derecho se borra todo el display y se comienza de nuevo desde la izquierda. El modo scope chart es mucho más rápido que el modo strip chart ya que no es necesario realizar todo el proceso de desplazar la pantalla hacia la izquierda para cada nuevo punto. El modo sweep chart actúa como el modo scope chart, salvo que ahora cuando se llega al final de la pantalla, ésta no se borra y se comienza de nuevo desde el principio, donde una línea vertical se mueve hacia la derecha cada vez que se añade un nuevo punto.

- **Create Attribute Node:** Crea un nodo asociado al terminal de que procede en el Diagrama de Bloques.
- **Replace:** Permite sustituir la waveform chart por cualquiera de los controles e indicadores del Panel Frontal.
- **X Scale and Y Scale:** Permite escoger el estilo de la escala, tipo de rejilla, punto inicial, incremento entre punto y punto, formato y precisión de estos puntos.
- **Transpose Array:** Cuando se representa más de una gráfica en una misma chart utilizando arrays, waveform chart interpreta por defecto las filas como gráficas diferentes. Pero si a nosotros nos interesa que sean las columnas las gráficas diferentes, utilizaremos este comando para convertir las columnas en filas.
- **Stack Plots:** Normalmente cuando se representan más de una gráfica todas ellas se sitúan en un mismo display. Pero puede ocurrir que las escalas de las ordenadas sean muy diferentes entre ellas o que simplemente nos interese representarlas por separado, cada una en un display. Para conseguir esto activaremos el comando Stack Plot de forma que cada gráfica aparecerá con su propia escala y su propio display. Cuando Stack Plots está activado, en su lugar aparece el comando Overlay Plot que es el que dibuja todas las gráficas en un mismo display.

Chart History Length: Mediante este control podemos fijar el número de puntos que waveform chart almacenará en el buffer que, por defecto, serán 1024.

INTENSITY CHART

Mediante *intensity chart* podemos mostrar datos tridimensionales colocando bloques de colores sobre planos cartesianos. Para ello crearemos arrays bidimensionales de números donde los índices de un elemento corresponderán a las coordenadas X e Y, y el contenido a la coordenada Z, que tendrá asociado un color para cada posible valor. Previamente será necesario definir la escala de colores que vamos a utilizar a través de los *atribute nodes* mediante el ítem *Z Scale Info: Color Array o Color Table*, o a través de la rampa de colores visualizada junto a la gráfica.

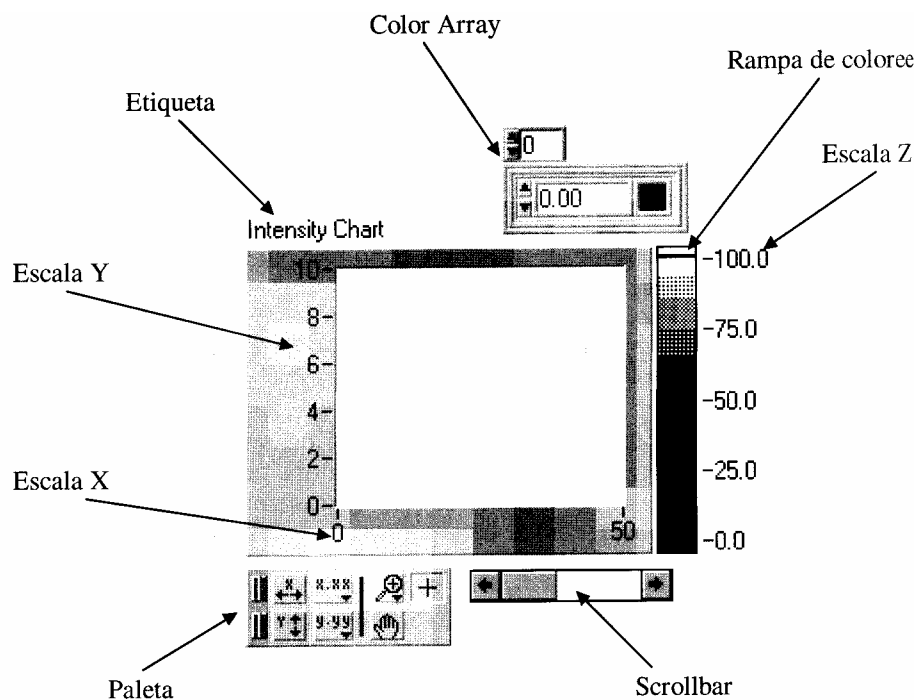


Figura 5.2 Indicador intensity chart

Evidentemente, la escala de colores que podamos visualizar dependerá de la resolución de nuestro monitor.

Cada vez que se envíe un nuevo conjunto de datos, estos aparecerán representados a la derecha de los ya existentes. *Intensity chart* soporta los tres modos de visualización de *waveform chart* y también dispone de un buffer cuyo tamaño es, por defecto, de 128 puntos. Las opciones disponibles para *intensity chart* son prácticamente las mismas que para *waveform chart*. únicamente, debido a que existe una nueva coordenada, aparecen en el menú opciones para ésta, como son:

- Show Ramp: Visualiza u oculta la rampa de colores.
- Show Color Array: Permite fijar los colores de la rampa.

- Show Z Scale: Visualiza u oculta la escala z.
- AutoScale Z: Ajusta de forma automática el rango de valores de z a la escala de colores.
- Z scale: Permite escoger el estilo de la escala, tipo de rejilla, punto inicial, incremento entre punto y punto, formato y precisión de estos puntos.

INDICADORES GRAPH

WAVEFORM GRAPH

Waveform graph representa una serie de valores Y equiespaciados dada siempre una distancia delta de X (ΔX) comenzando a partir de un valor inicial X_0 . A un mismo punto X, sólo le puede corresponder un valor de Y,. Cuando se representa una nueva serie de datos, al contrario de lo que ocurría en los indicadores chart, estos datos reemplazan a los ya existentes en lugar de añadirse al lado, y pierden los valores representados con anterioridad.

Existen dos posibilidades a la hora de representar una única gráfica en una *waveform graph*. La primera consiste en unir un array de valores numéricos directamente a la *graph* de forma que ésta interpreta cada valor como un nuevo punto comenzando en $X=0$ e incrementando X en 1 para cada punto.

La segunda consiste en crear un cluster en, el cual, junto con el array de valores, se indica el valor inicial X_0 y el incremento ΔX .

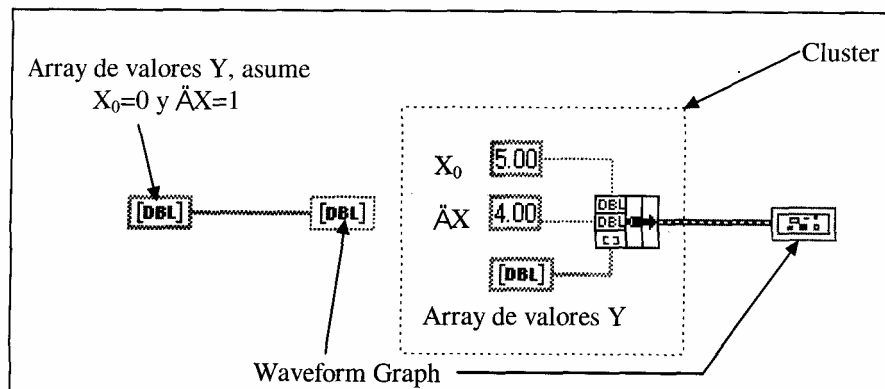


Figura 5.3 Representación de una sola gráfica

Existe la posibilidad de representar más de una gráfica en una misma *waveform graph*. Para ello es necesario unir los datos de las diferentes gráficas en un formato que LabVIEW sepa interpretar. Utilizar un formato u otro vendrá determinado principalmente por las características de las gráficas a mostrar. Así, si todas las gráficas tienen un mismo escalado X y un mismo número de puntos, bastará con crear un array bidimensional de valores numéricos donde cada fila de datos es una única gráfica. LabVIEW interpretará estos datos como puntos en la gráfica comenzando en $X=0$ e incrementándola en 1. Si nos interesa cambiar el punto inicial o el incremento de x, crearemos un cluster que contendrá el array bidimensional y los valores de x_0 y Δx .

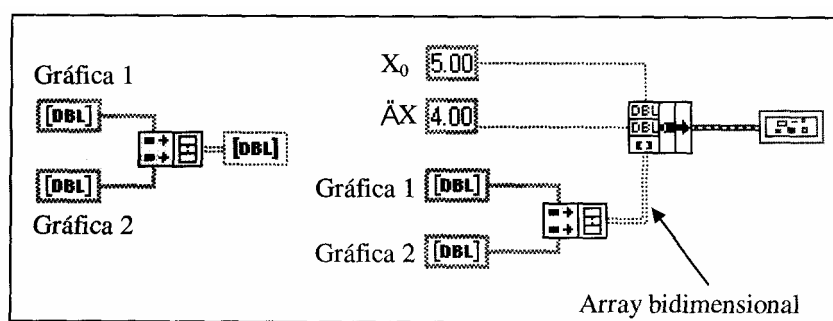


Figura 5.4 Representación de múltiples gráficas con el mismo número de puntos

Mediante el comando *Transpose Array* del menú pop-up podemos hacer que LabVIEW interprete las columnas como gráficas diferentes en lugar de las filas.

Puede ocurrir que el número de elementos de cada gráfica sea diferente. En ese caso es necesario crear un cluster para cada array de datos y después unir todos los clusters en un array. Esto es necesario debido a que LabVIEW no permite crear arrays de arrays. Al igual que anteriormente si nos interesa que el punto inicial sea diferente de cero o que el incremento sea diferente de 1, crearemos un cluster que contenga el array de clusters de array y los nuevos valores de X_0 y ΔX .

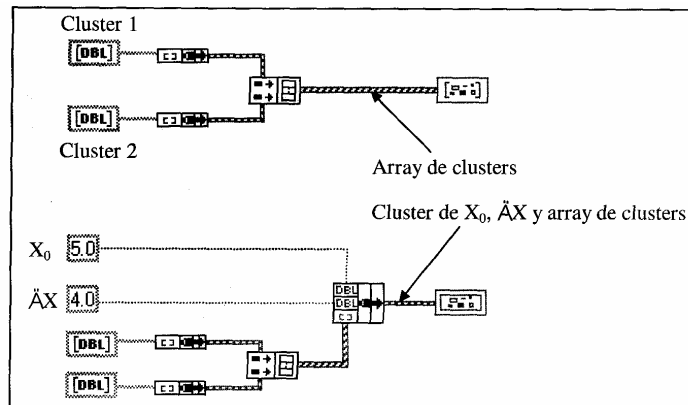


Figura 5.5 Representación de múltiples gráficas con diferente números de puntos

Finalmente, si ni el escalado ni el número de puntos de la gráfica es el mismo para todas ellas, lo que haremos será crear un cluster por cada gráfica que contendrá un array de datos, un valor X_0 y un valor $\ddot{A}X$. Y con todos los clusters de las diferentes gráficas crearemos un array. Este último formato es el más completo de todos porque permite fijar un valor X_0 y un valor $\ddot{A}X$ diferente para cada gráfica.

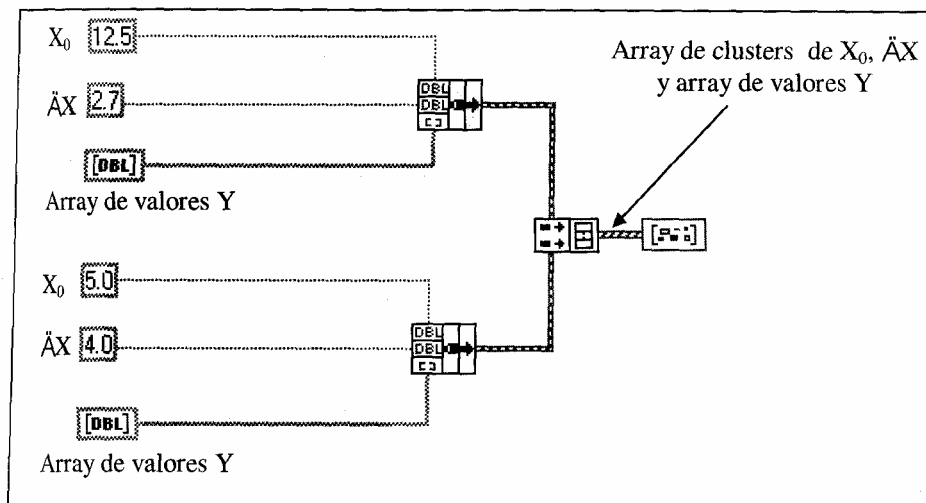


Figura 5.6 Representación de múltiples gráficas con diferente número de puntos y diferente escalado

XYGRAPH

En *XY Graph* un punto X, puede tener varios valores Y, lo que permite, por ejemplo, dibujar funciones circulares. *XY Graph* representa una coordenada (X_i, Y_i) donde los valores de X no tienen porque estar equiespaciados como ocurría en *waveform graph*.

Para representar una única gráfica en una *XY Graph* existen dos posibilidades. La primera consiste en crear un cluster que contenga un array de datos X y un array de datos Y. La segunda consiste en crear un array de clusters, donde cada cluster contiene un valor de X y un valor de Y.

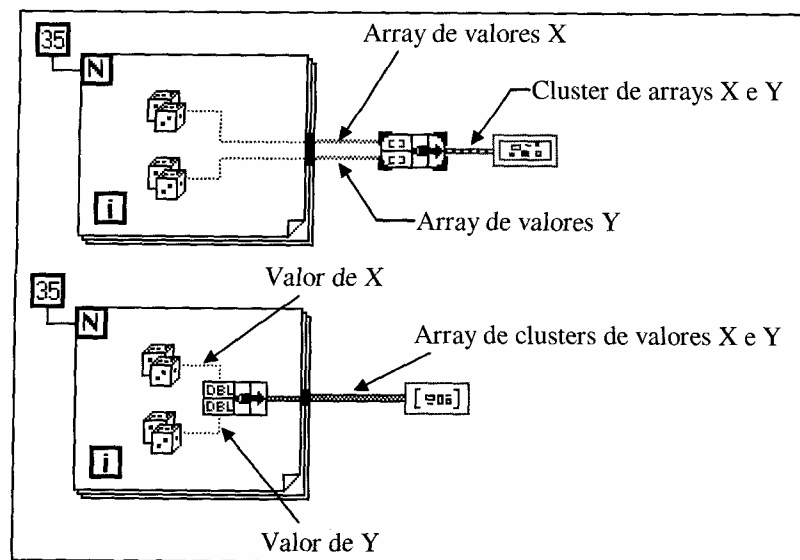


Figura 5.7 Posibles representaciones de una única XY Graph

Al igual que en las *waveform graph* existe la posibilidad de representar más de una gráfica en una misma *XY Graph* (figura 5.8). Pero, en este caso, tan sólo existen dos formatos posibles derivados de los dos formatos vistos anteriormente para una única gráfica. El primer formato es un array de gráficas, donde cada gráfica es un cluster de un array X y un array Y. Y el segundo formato es un array de clusters de gráficas, donde cada gráfica es, a su vez, otro array de clusters conteniendo un valor X y un valor Y.

INTENSITY GRAPH

Intensity graph es exactamente igual que *intensity chart* salvo que *intensity graph* no retiene valores anteriores, por lo que cuando un nuevo bloque de valores se carga, éstos sustituyen a los ya existentes.

Los comandos disponibles en los menús pop-up de los indicadores *graph* tienen las mismas utilidades que los descritos en los indicadores *chart*, por lo que no se han mencionado en este apartado. Solamente existe una diferencia importante y es que los indicadores *graph* disponen de cursores que nos permiten movernos por la gráfica.

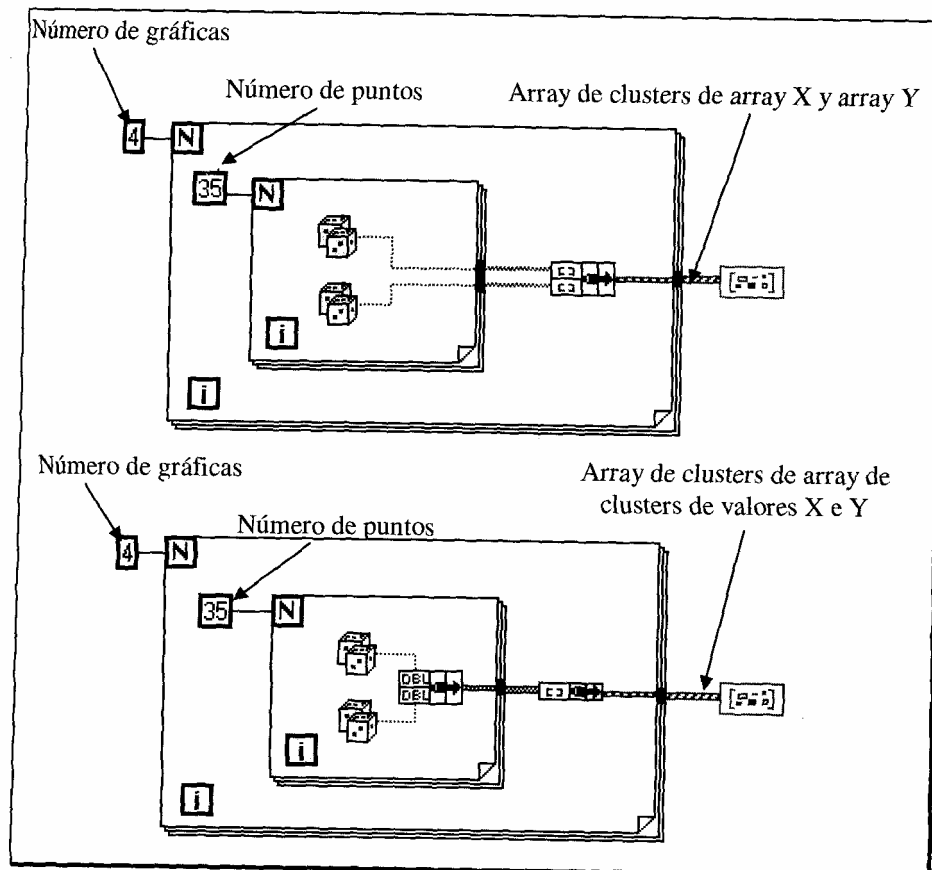


Figura 5.8 Posibles representaciones de múltiples XY Graph

GRAPH DURSORS

La paleta de cursores está disponible desde la opción *Show Cursor Display* de; menú pop-up (figura 5.9).

- Nombre del cursor: Permite introducir una etiqueta de identificación del cursor. Podemos tener tantos cursores como deseemos.
- Posición X, Posición Y : Indica las coordenadas en las que se encuentra el cursor; en los indicadores *intensíty graph* aparece también la coordenada Z. Podemos mover el cursor directamente a una posición concreta introduciendo las coordenadas del punto deseado.

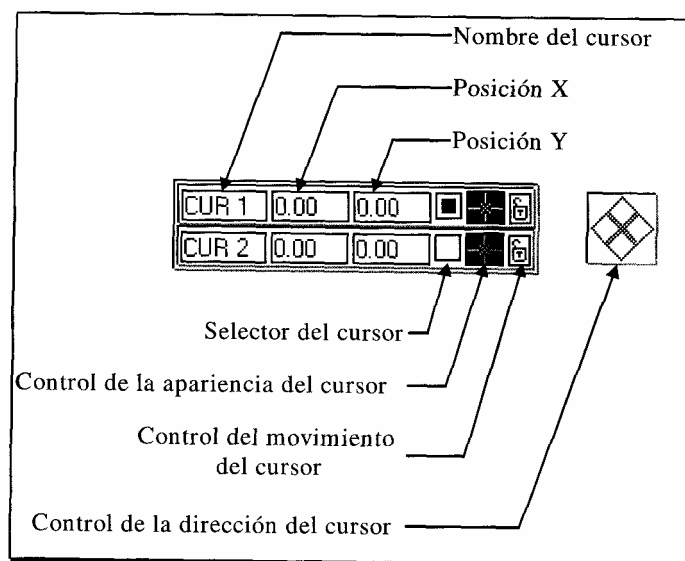
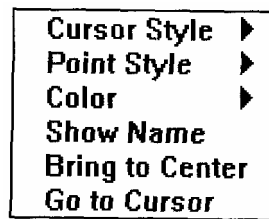


Figura 5.9 Paleta de cursores

- Selector del cursor: Selecciona el cursor a mover. Se pueden seleccionar a vez tantos cursores como deseemos.
- Control de la apariencia del cursor: Abriendo el menú mediante el bot izquierdo del ratón podemos modificar algunas características del cursor:



- Cursor Style: Selecciona la forma con la que se indica el punto sobre el cual encuentra el cursor.

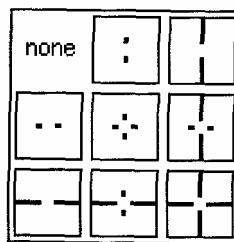


Figura 5.10 Submenú Cursor Style.

- Point Style: Selecciona el estilo de; punto que marca la posición del cursor.

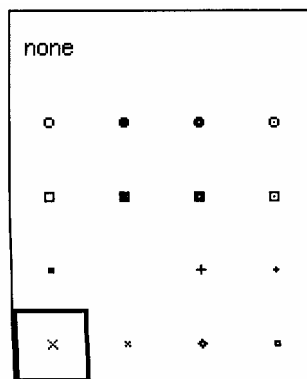
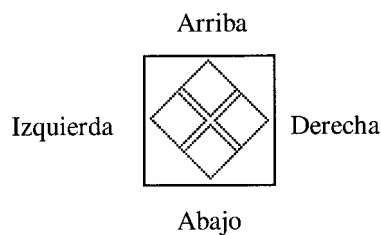


Figura 5.11 Submenú Point Style

- Color. Selecciona el color del cursor.
- 9 Show Name: Muestra el nombre del cursor sobre la gráfica.

- Bring to Center: Mueve el cursor hasta el centro de la pantalla cambiando las coordenadas de éste.
- Go to Cursor,. Modifica las escalas X e Y de forma que podamos ver el cursor, pero sin cambiar las coordenadas de éste.
- Control del movimiento del cursor: El candado cerrado indica que el cursor se moverá siguiendo la gráfica (opciones *Lock to plot* y *Snap to point*), mientras que el candado abierto indica que el cursor se moverá libremente (opción *Free*). Si hubiese más de una gráfica el menú nos permitirá escoger sobre cual de ellas queremos que se mueva el cursor. El comando *Allow Drag*, cuando está activo, permite desplazar la gráfica directamente con el puntero del ratón.

Control de la dirección de; cursor: Mueve los cursores seleccionados punto por punto en la dirección indicada.



5.4 CONCLUSIONES

Ante los diferentes tipos de indicadores se plantea la necesidad de escoger entre uno u otro. Decir cuándo se debe utilizar cada uno es muy difícil ya que depende de cada aplicación y, además, puesto que en programación no hay nada imposible, podemos llegar a hacer que una gráfica simule el comportamiento de otra; sólo hace falta un poco de tiempo y paciencia. Pero sí podemos indicar para qué es aconsejable cada indicador. Cuando tengamos datos que dependan de; eje de las abscisas y no estén equiespaciados en el tiempo tendremos que utilizar, sin más remedio, un indicador *XY Graph*. Si los datos dependieran de; eje de las abscisas pero están equiespaciados podremos utilizar un indicador *Waveform Graph* si queremos que los nuevos datos sustituyan a los anteriores o un indicador *Waveform Chart* si queremos que los nuevos datos se añadan a continuación de los ya existentes, como puede ser en el caso de un electrocardiograma en el que interesa ver el comportamiento a lo largo de; tiempo y la utilización

de un indicador *Graph* supondría la pérdida de información. Por último, si tenemos que representar sobre ejes cartesianos funciones de tres variables utilizaremos los indicadores *Intensity* ya sea *Chart* o *Graph*.

La versión 4.0 presenta mínimas diferencias, como pueden ser la creación directa de variables locales o la ayuda en línea, pero en cuanto a funcionamiento todo lo dicho para la versión 3,1 es totalmente válido.