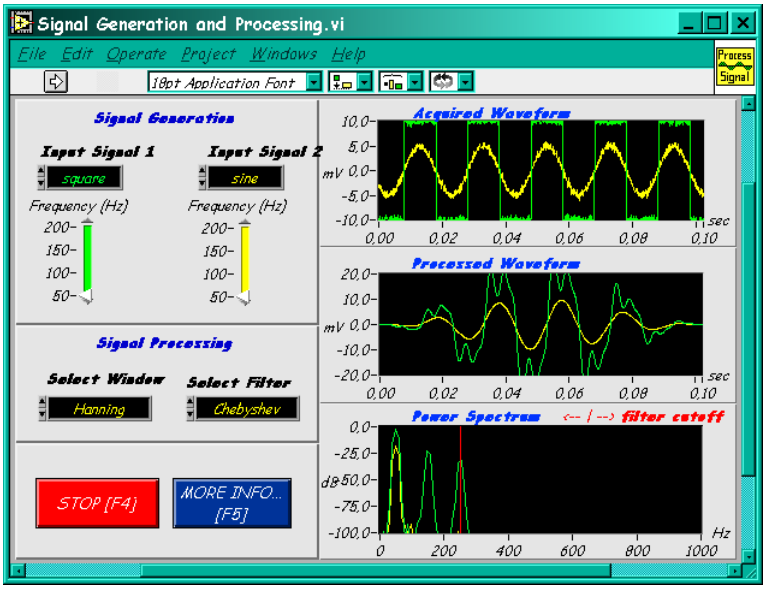
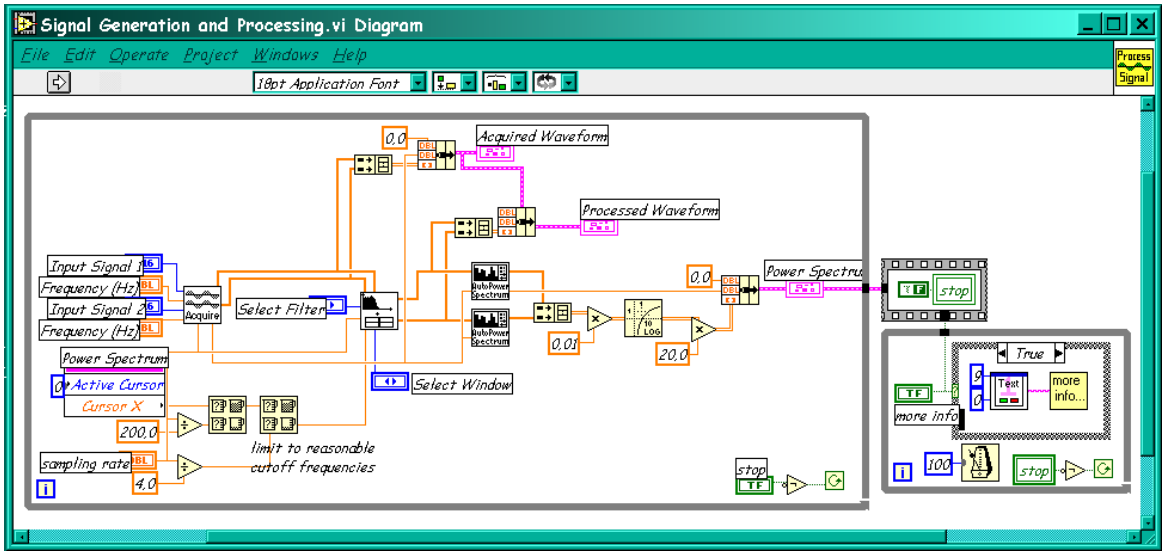
TUTORIAL DE LABVIEW





1.- INTRODUCCIÓN

LabVIEW constituye un revolucionario sistema de programación gráfica para aplicaciones que involucren adquisición, control, análisis y presentación de datos. Las ventajas que proporciona el empleo de LabVIEW se resumen en las siguientes:

- Se reduce el tiempo de desarrollo de las aplicaciones al menos de 4 a 10 veces, ya que es muy intuitivo y fácil de aprender.
- Dota de gran flexibilidad al sistema, permitiendo cambios y actualizaciones tanto del hardware como del software.
- Da la posibilidad a los usuarios de crear soluciones completas y complejas.
- Con un único sistema de desarrollo se integran las funciones de adquisición, análisis y presentación de datos.
- El sistema está dotado de un compilador gráfico para lograr la máxima velocidad de ejecución posible.
- Tiene la posibilidad de incorporar aplicaciones escritas en otros lenguajes.

LabVIEW es un entorno de programación destinado al desarrollo de aplicaciones, similar a los sistemas de desarrollo comerciales que utilizan el *lenguaje C* o *BASIC*. Sin embargo, LabVIEW se diferencia de dichos programas en un importante aspecto: los citados lenguajes de programación se basan en líneas de texto para crear el código fuente del programa, mientras que LabVIEW emplea la programación gráfica o *lenguaje G* para crear programas basados en diagramas de bloques.

Para el empleo de LabVIEW no se requiere gran experiencia en programación, ya que se emplean iconos, términos e ideas familiares a científicos e ingenieros, y se apoya sobre símbolos gráficos en lugar de lenguaje escrito para construir las aplicaciones. Por ello resulta mucho más intuitivo que el resto de lenguajes de programación convencionales.

LabVIEW posee extensas librerías de funciones y subrutinas. Además de las funciones básicas de todo lenguaje de programación, LabVIEW incluye librerías específicas para la adquisición de datos, control de instrumentación VXI, GPIB y comunicación serie, análisis presentación y guardado de datos.

LabVIEW también proporciona potentes herramientas que facilitan la depuración de los programas.

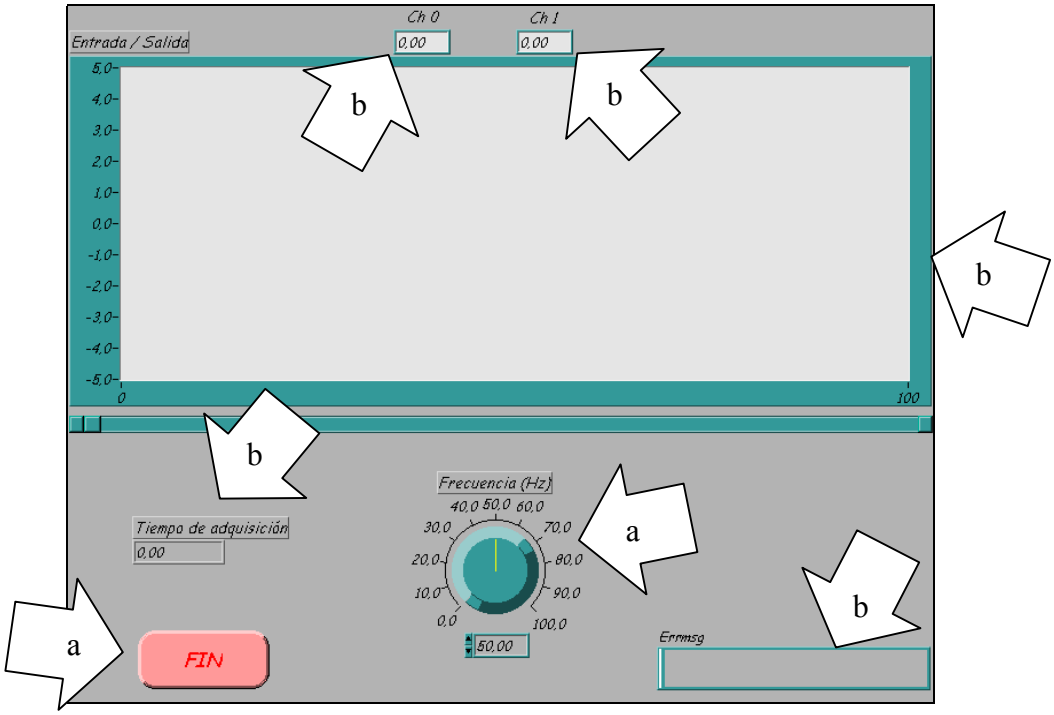
2.- ¿CÓMO TRABAJA LABVIEW?

Los programas desarrollados mediante LabVIEW se denominan *Instrumentos Virtuales* (*VIs*), porque su apariencia y funcionamiento imitan los de un instrumento real. Sin embargo son análogos a las funciones creadas con los lenguajes de programación convencionales. Los *VIs* tienen una parte interactiva con el usuario y otra parte de código fuente, y aceptan parámetros procedentes de otros *VIs*.

Todos los *VIs* tienen un *panel frontal* y un *diagrama de bloques*. Las *paletas* contienen las opciones que se emplean para crear y modificar los *VIs*. A continuación se procederá a realizar una somera descripción de estos conceptos.

A) Panel Frontal

Se trata de la interfaz gráfica del *VI* con el usuario. Esta interfaz recoge las entradas procedentes del usuario y representa las salidas proporcionadas por el programa. Un *panel frontal* está formado por una serie de botones, pulsadores, potenciómetros, gráficos, etc. Cada uno de ellos puede estar definido como un *control* (a) o un *indicador* (b). Los primeros sirven para introducir parámetros al VI, mientras que los indicadores se emplean para mostrar los resultados producidos, ya sean datos adquiridos o resultados de alguna operación.

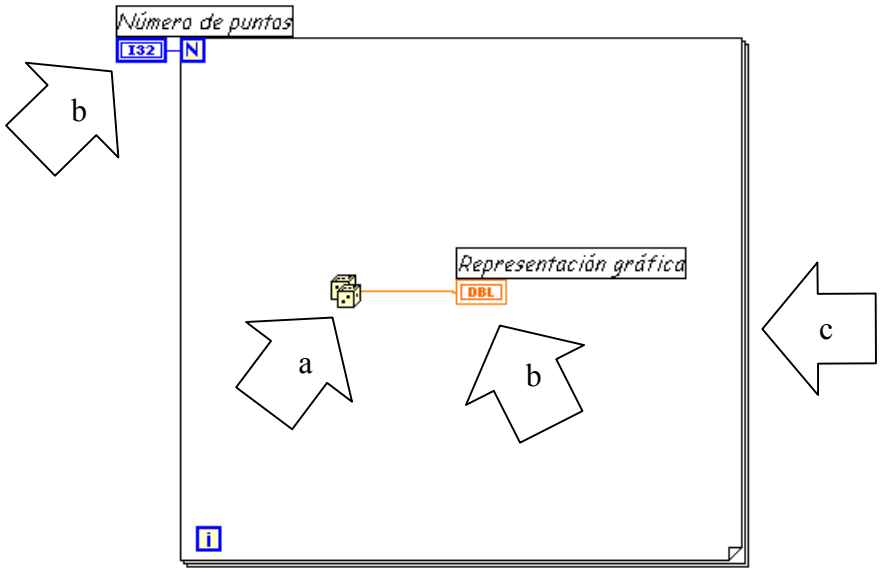


B) Diagrama de bloques

El *diagrama de bloques* constituye el código fuente del *VI*. En el *diagrama de bloques* es donde se realiza la implementación del programa del *VI* para controlar o realizar cualquier procesado de las entradas y salidas que se crearon en el *panel frontal*.

El *diagrama de bloques* incluye *funciones* y *estructuras* integradas en las librerías que incorpora LabVIEW. En el *lenguaje G* las *funciones* y las *estructuras* son nodos elementales. Son análogas a los operadores o librerías de funciones de los lenguajes convencionales.

Los *controles* e *indicadores* que se colocaron previamente en el Panel Frontal, se materializan en el diagrama de bloques mediante los *terminales.* A continuación se presenta un ejemplo de lo recién citado:



(a) Función.

(b) Terminales (control e indicador).

(c) Estructura.

El *diagrama de bloques* se construye conectando los distintos objetos entre sí, como si de un circuito se tratara. Los cables unen terminales de entrada y salida con los objetos correspondientes, y por ellos fluyen los datos.

LabVIEW posee una extensa biblioteca de *funciones*, entre ellas, aritméticas, comparaciones, conversiones, funciones de entrada/salida, de análisis, etc.

Las *estructuras*, similares a las declaraciones causales y a los bucles en lenguajes convencionales, ejecutan el código que contienen de forma condicional o repetitiva (bucle *for*, *while*, *case*,...).

Los cables son las trayectorias que siguen los datos desde su origen hasta su destino, ya sea una función, una estructura, un terminal, etc. Cada cable tiene un color o un estilo diferente, lo que diferencia unos tipos de datos de otros.

C) Paletas.

Las *paletas* de LabVIEW proporcionan las herramientas que se requieren para crear y modificar tanto el *panel frontal* como el *diagrama de bloques*. Existen las siguientes paletas:

Paleta de herramientas *(Tools palette)*

Se emplea tanto en el *panel frontal* como en el *diagrama de bloques*. Contiene las herramientas necesarias para editar y depurar los objetos tanto del *panel frontal* como del *diagrama de bloques*.

Las opciones que presenta esta paleta son las siguiente:

 *Operating tool* – Cambia el valor de los controles.

 *Positioning tool* – Desplaza, cambia de tamaño y selecciona los objetos.

 *Labeling tool* – Edita texto y crea etiquetas.

 *Wiring tool* – Une los objetos en el *diagrama de bloques*.

 *Object Pop-up Menu tool* – Abre el menú desplegable de un objeto.

 *Scroll tool* – Desplaza la pantalla sin necesidad de emplear las barras de desplazamiento.

 *Breakpoint tool* – Fija puntos de interrupción de la ejecución del programa en *VIs*, *funciones* y *estructuras*.

Probe tool – Crea puntos de prueba en los cables, en los que se puede visualizar el valor del dato que fluya por dicho cable en cada instante.

Color Copy tool – Copia el color para después establecerlo mediante la siguiente herramienta.

Color tool – Establece el color de fondo y el de los objetos

Paleta de controles *(Controls palette)*

Se utiliza únicamente en el *panel frontal*. Contiene todos los *controles* e *indicadores* que se emplearán para crear la interfaz del *VI* con el usuario.

El menú *Controls* de la ventana correspondiente al panel frontal contiene las siguientes opciones:

Numeric – Para la introducción y visualización de cantidades numéricas.

Boolean – Para la entrada y visualización de valores booleanos.

String & Table – Para la entrada y visualización de texto.

List & Ring – Para visualizar y/o seleccionar una lista de opciones.

Array & Cluster – Para agrupar elementos.

Graph – Para representar gráficamente los datos.

Path & RefNum – Para gestión de archivos.

Decorations – Para introducir decoraciones en el *panel frontal*. No visualizan datos.

User Controls – Para elegir un *control* creado por el propio usuario.

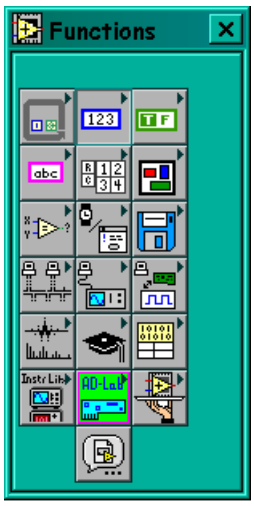
ActiveX – Para transferir datos y programas de unas aplicaciones a otras dentro de Windows.

Select a Control – Para seleccionar cualquier *control*.

Al seleccionar objetos desde el menú *Controls* estos aparecen sobre el *panel frontal*, pueden colocarse donde convenga, y además tienen su propio menú desplegable que permite la configuración de algunos parámetros específicos de cada tipo de *control*..

Paleta de funciones (functions palette)

Se emplea en el diseño del *diagrama de bloques*. La *paleta de funciones* contiene todos los objetos que se emplean en la implementación del programa del *VI*, ya sean *funciones* aritméticas, de entrada/salida de señales, entrada/salidad de datos a fichero, adquisición de señales, temporización de la ejecución del programa,...



Para seleccionar una *función* o *estructura* concretas, se debe desplegar el menú *Functions* y elegir entre las opciones que aparecen. A continuación se enumeran todas ellas, junto con una pequeña definición.

Structures – Muestra las *estructuras* de control del programa, junto con las variables locales y globales.

Numeric – Muestra *funciones* aritméticas y constantes numéricas.

Boolean – Muestra *funciones* y constantes lógicas.

String – Muestra *funciones* para manipular cadenas de caracteres, así como constantes de caracteres.


Array – Contiene *funciones* útiles para procesar datos en forma de vectores, así como constantes de vectores.


Cluster – Contiene *funciones* útiles para procesar datos procedentes de gráficas y destinados a ser representados en ellas, así como las correspondientes constantes.


Comparison – Muestra *funciones* que sirven para comparar números, valores booleanos o cadenas de caracteres.


Time & Dialog – Contiene *funciones* para trabajar con cuadros de diálogo, introducir contadores y retardos, etc.


File I/O – Muestra *funciones* para operar con ficheros.


Communication – Muestra diversas *funciones* que sirven para comunicar varios ordenadores entre sí, o para permitir la comunicación entra distintos programas.


Instrument I/O – Muestra un submenú de *VIs*, que facilita la comunicación con instrumentos periféricos que siguen la norma ANSI/IEEE 488.2-1987, y el control del puerto serie.


Data Acquisition – Contiene a su vez un submenú donde puede elegirse entre distintas librerías referentes a la adquisición de datos.


Analysis – Contiene un submenú en el que se puede elegir entre una amplia gama de *funciones* matemáticas de análisis.


Tutorial – Incluye un menú de *VIs* que se utilizan en el manual LabVIEW Tutorial.


Advanced – Contiene diversos submenús que permiten el control de la ayuda, de los *VIs*, manipulación de datos, procesado de eventos, control de la memoria, empleo de programas ejecutables o incluidos en librerías DLL, etc.

Instrument drivers – En él se muestran los drivers disponibles de distintos instrumentos.

 *User Libraries* – Muestra as librerías definidas por el usuario. En este caso, la librería mostrada contiene los drivers de la tarjeta de adquisición de datos de Advantech.

 *Aplication control* – Contiene varias *funciones* que regulan el funcionamiento de la propia aplicación en ejecución.

 *Select a VI* – Permite seleccionar cualquier *VI* para emplearlo como *subVI*.

3.- PROGRAMACIÓN EN LABVIEW

Con el entorno gráfico de programación de LabVIEW se comienza a programar a partir del *panel frontal*.

En primer lugar se definirán y seleccionarán de la *paleta de controles* todos los *controles* (entradas que dará el usuario) e *indicadores* (salidas que presentará en pantalla el *VI*) que se emplearán para introducir los datos por parte del usuario y presentar en pantalla los resultados.

Una vez colocados en la ventana correspondiente al *panel frontal* todos los objetos necesarios, debe pasarse a la ventana *Diagram* (menú *Windows > Show Diagram*), que es donde se realiza la programación propiamente dicha (*diagrama de bloques*). Al abrir esta ventana, en ella se encuentran los terminales correspondientes a los objetos situados en el *panel frontal*, dispuestos automáticamente por LabVIEW.

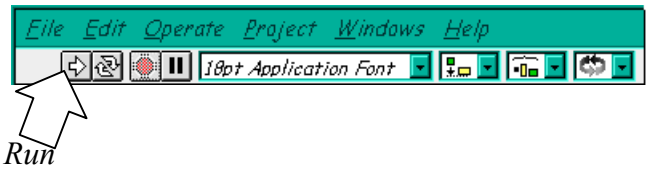
Se deben ir situando las *funciones*, *estructuras*, etc. que se requieran para el desarrollo del programa, las cuales se unen a los terminales mediante cables.

Para facilitar la tarea de conexión de todos los terminales, en el menú "*Help*" puede elegirse la opción "*Show Help*", con lo que al colocar el cursor del ratón sobre un elemento aparece una ventana con información relativa a éste (parámetros de entrada y salida). Además, si se tiene seleccionado el cursor de cableado, al situar éste sobre un elemento se muestran los terminales de forma intermitente.

4.- EJECUCIÓN DE UN VI

Una vez se ha concluido la programación del VI se debe proceder a su ejecución. Para ello la ventana activa debe ser el *panel frontal* (si se está en la ventana del *diagrama de bloques*, se debe seleccionar la opción *Show Panel* del menú *Window*).

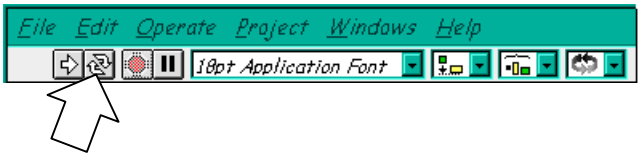
Un a vez situados en el *panel frontal*, se pulsará el botón de *Run*, situado en la barra de herramientas.



Run

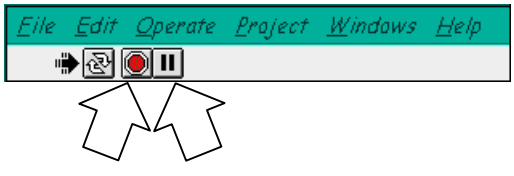
El programa comenzará a ejecutarse. Mientras dura la ejecución del mismo, la apariencia del botón de *Run* es la que se muestra a continuación:

De este modo el programa se ejecutará una sola vez. Si se desea una ejecución continua, se pulsará el botón situado a la derecha del de *Run* (*Continuous Run*). Si durante el funcionamiento continuo del programa se vuelve a pulsar el citado botón, se finalizará la última ejecución del mismo, tras lo cual el programa se parará.



Continuous Run

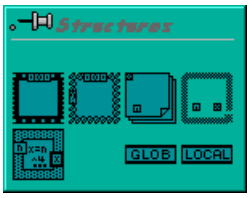
Para finalizar la ejecución de un programa se puede operar de dos formas. La primera, y la más aconsejable, es emplear un botón en el *panel frontal* del *VI*, cuya pulsación produzca la interrupción del bucle de ejecución de la aplicación. La segunda forma de detener la ejecución del *VI* es pulsando el botón de *pausa* o el de *stop*. La diferencia entre ambos es que si se pulsa *stop*, la ejecución del programa finaliza inmediatamente, mientras que si se pulsa *pausa*, se produce una detención en el funcionamiento del programa, retomándose su ejecución una vez se vuelve a pulsar el mismo botón.



stop *pausa*

5.- ESTRUCTURAS

En la *paleta de funciones* la primera opción es la de las *estructuras*. Éstas controlan el flujo del programa, bien sea mediante la secuenciación de acciones, ejecución de bucles, etc.



Las estructuras se comportan como cualquier otro nodo en el diagrama de bloques, ejecutando automáticamente lo que está programado en su interior una vez tiene disponibles los datos de entrada, y una vez ejecutadas las instrucciones requeridas, suministran los correspondientes valores a los cables unidos a sus salidas. Sin embargo, cada estructura ejecuta su *subdiagrama* de acuerdo con las reglas específicas que rigen su comportamiento, y que se especifican a continuación.

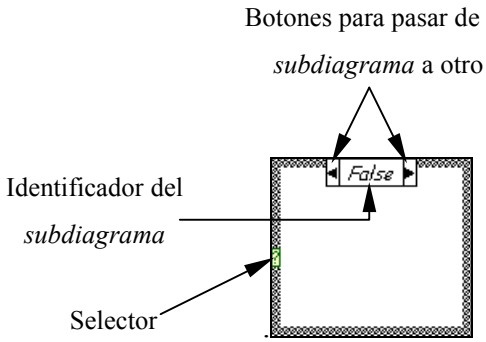
Un *subdiagrama* es una colección de nodos, cables y terminales situados en el interior del rectángulo que constituye la estructura. El *For Loop* y el *While Loop* únicamente tienen un subdiagrama. El *Case Structure* y el *Sequence Structure*, sin embargo, pueden tener múltiples subdiagramas, superpuestos como si se tratara de cartas en una baraja, por lo que en el diagrama de bloques únicamente será posible visualizar al tiempo uno de ellos. Los subdiagramas se construyes del mismo modo que el resto del programa

Las siguientes estructuras se hallan disponibles en el *lenguaje G*.

Case Structure

Al igual que otras estructuras posee varios *subdiagramas*, que se superponen como si de una baraja de cartas se tratara. En la parte superior del subdiagrama aparece el identificador del que se está representando en pantalla. A ambos lados de este identificador aparecen unas flechas que permiten pasar de un *subdiagrama* a otro.

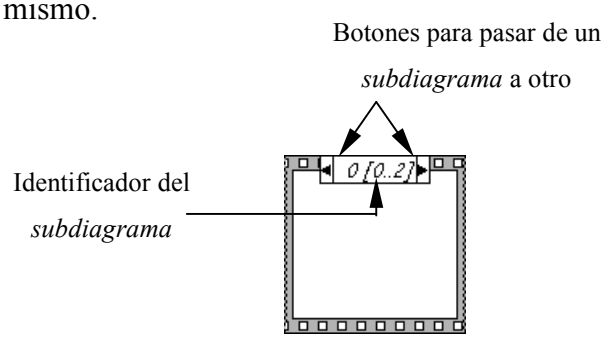
En este caso el identificador es un valor que selecciona el subdiagrama que se debe ejecutar en cada momento.



La estructura *Case* tiene al menos dos *subdiagramas* (*True* y *False*). Únicamente se ejecutará el contenido de uno de ellos, dependiendo del valor de lo que se conecte al *selector*.

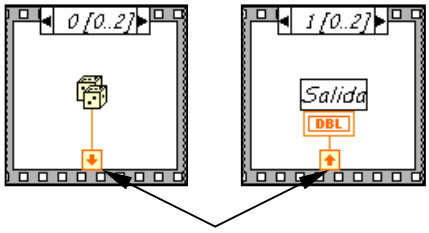
Sequence Structure

De nuevo, este tipo de estructuras presenta varios *subdiagramas*, superpuestos como en una baraja de cartas, de modo que únicamente se puede visualizar una en pantalla. También poseen un identificador del *sudiagrama* mostrado en su parte superior, con posibilidad de avanzar o retroceder a otros *subdiagramas* gracias a las flechas situadas a ambos lados del mismo.



Esta estructura secuencia la ejecución del programa. Primero ejecutará el *subdiagrama* de la hoja (*frame*) nº0, después el de la nº 1, y así sucesivamente.

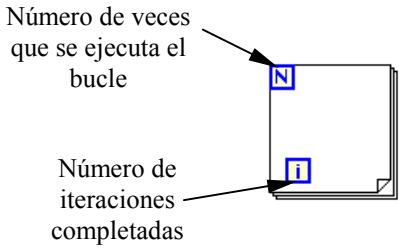
Para pasar datos de una hoja a otra se pulsará el botón derecho del ratón sobre el borde de la estructura, seleccionando la opción *Add sequence local*.



Sequence local: paso de un dato de la *frame 0* a la *1*

For Loop

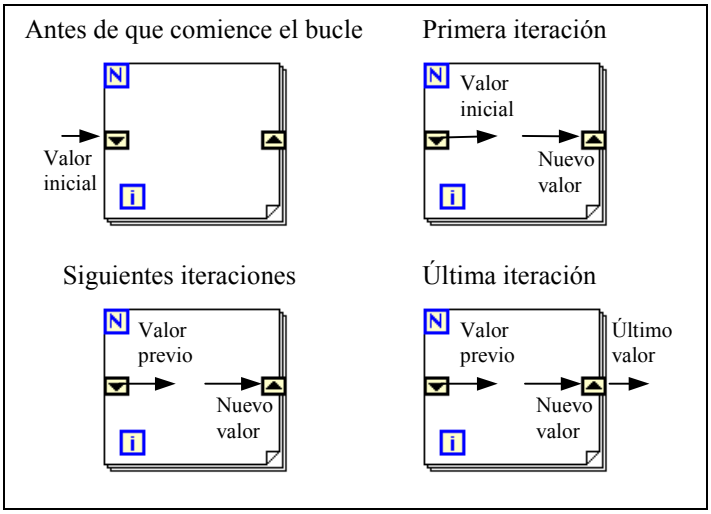
Es el equivalente al bucle *for* en los lenguajes de programación convencionales. Ejecuta el código dispuesto en su interior un número determinado de veces.



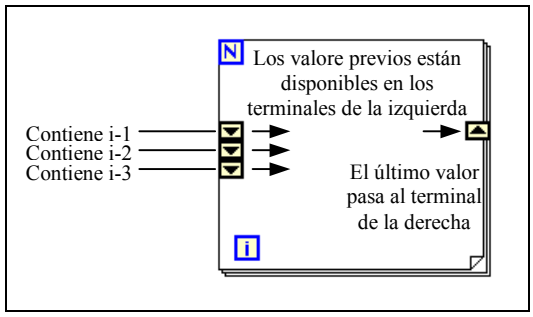
Ejecutar el bucle *for* es equivalente al siguiente fragmento de código:

```
For i = 0 to N – 1
Ejecutar el subdiagrama del interior del Bucle
```

Para pasar valores de una iteración a otra se emplean los llamador *shift registers*. Para crear uno, se pulsará el botón derecho del ratón mientras éste se halla situado sobre el borde del bucle, seleccionando la opción *Add Shift Register*. El *shift register* consta de dos terminales, situados en los bordes laterales del bloque. El terminal izquierdo almacena el valor obtenido en la iteración anterior. El terminal derecho guardará el dato correspondiente a la iteración en ejecución. dicho dato aparecerá , por tanto, en el terminal izquierdo durante la iteración posterior.

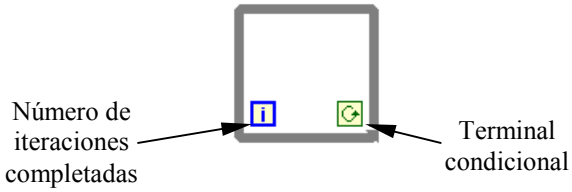


Se puede configurar un *shift register* para memorizar valores de varias iteraciones previas. Para ello, con el ratón situado sobre el terminal izquierdo del *shift registe,r* se pulsará el botón derecho, seleccionando a continuación la opción *Add Element*.



While Loop

Es el equivalente al bucle *while* empleado en los lenguajes convencionales de programación. Su funcionamiento es similar al del bucle *for*.



El bucle *while* es equivalente al código siguiente:

```
Do

Se ejecuta lo que hay en el interior del bloque

while terminal condicional is true
```

El programa comprueba el valor de lo que se halle conectado al terminal condicional al finalizar el bucle. Por lo tanto, el bucle siempre se ejecuta al menos una vez.

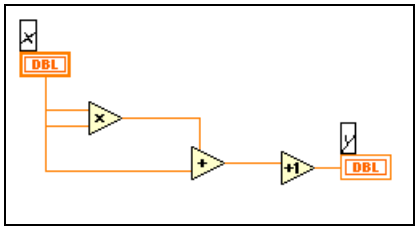
Con esta estructura también se pueden emplear los *shift registers* para tener disponibles los datos obtenidos en iteraciones anteriores (es decir, para memorizar valores obtenidos). su empleo es análogo al de los bucles *for*, por lo que omitirá su explicación.

Formula Node

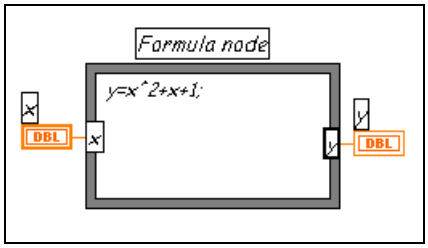
La estructura denominada *Formula Node* se emplea para introducir en el diagrama de bloques fórmulas de un modo directo. Resulta de gran utilidad cuando la ecuación tiene muchas variables o es relativamente compleja. Por ejemplo, se desea implementar la ecuación:

$y = x^2 + x + 1$

Empleando bloques pertenecientes al *lenguaje G* quedaría:



Si se utiliza la *formula node*, se obtiene:



Para definir una fórmula mediante este estructura, se actuará del siguiente modo:

- En primer lugar, se deben definir las variables de entrada y las de salida. Para ello, se pulsa con el botón derecho del ratón sobre el borde de la *formula node*. A continuación se seleccionará *Add Input* o *Add Output*, según se trate de una entrada o una salida, respectivamente. Aparecerá un rectángulo, en el que se debe escribir el nombre de la variable (se distingue entre mayúsculas y minúsculas). Todas las variables que se empleen deben estar declaradas como entradas o salidas. Las que se empleen como variables intermedias se declararán como salidas, aunque posteriormente no se unan a ningún bloque posterior.
- Una vez definidas las variables a emplear, se escribirán la o las fórmulas en el interior del recuadro (para ello se emplea la *labeling tool*). Cada fórmula debe finalizar con un "*;*".
- Los operadores y funciones que se pueden emplear se explican en la ayuda de LabVIEW, y son los que se muestran a continuación:

Operadores:
asignación =
condicional ?:
OR lógico ||
AND lógico &&
relacionales == != > < >= <=
aritméticos + - * / ^

Funciones:
abs acos acosh asin asinh atan atanh ceil cos cosh cot csc exp expm1 floor getexp getman int intrz ln lnp1 log log2 max min mod rand rem sec sgn sin sinc sinh sqrt tan tanh

La sintaxis de una expresión incondicional es la siguiente:

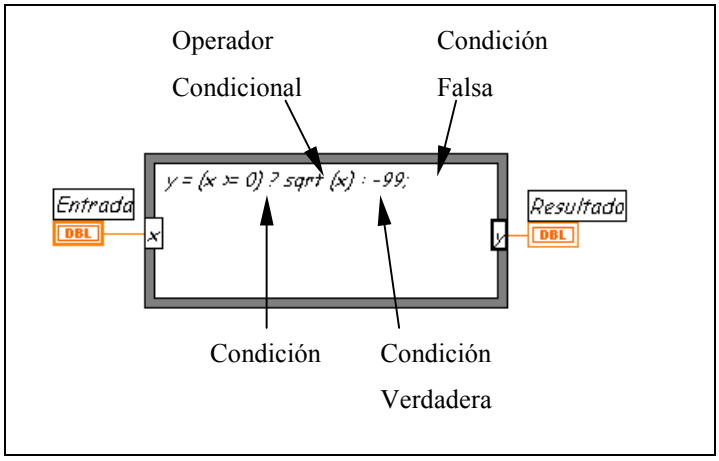
<expresión condicional> ? <texpresión> : <fexpresión>

Si el valor lógico de la *expresión condicional* es *true* se ejecutará *texpresión*. Si, por el contrario, fuese *false*, lo que se aplicará será *fexpresión*

Como ejemplo considérese el siguiente fragmento de código:

```
if (x >= 0) then
y = sqrt (x)
else
y = -99
end if
```

Se puede implementar este fragmento de código empleando un *formula node*, tal y como se muestra en la siguiente figura:

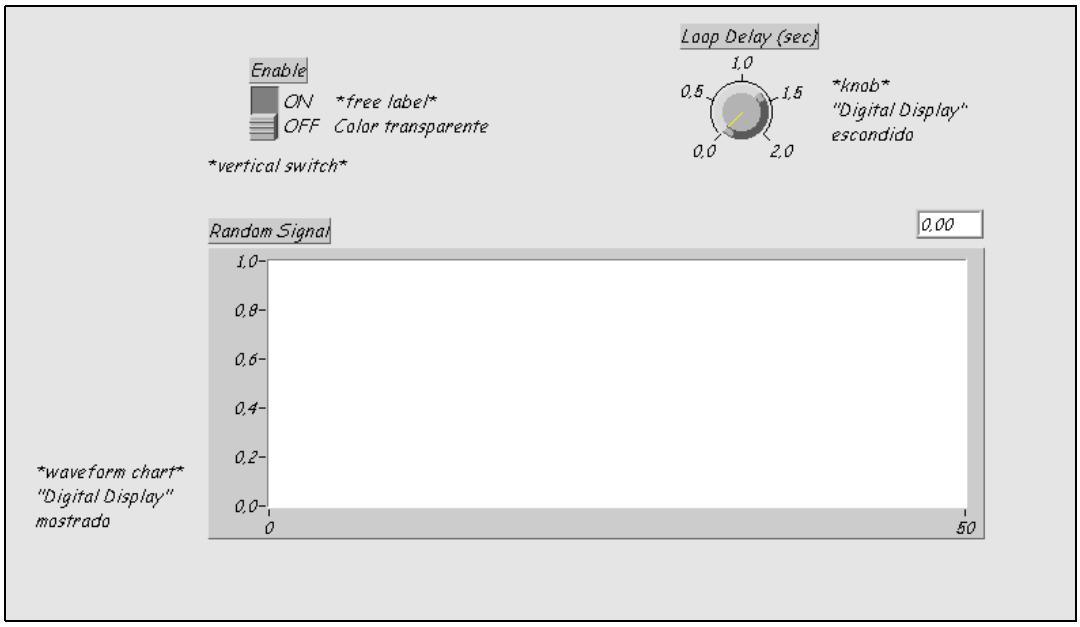


6.- EJEMPLO: CONSTRUCCIÓN DE UN *VI*

En este apartado se mostrará cómo construir una aplicación mediante el empleo del entorno de programación que proporciona LabVIEW.

6.1.- Panel frontal

En primer lugar, se debe construir el panel frontal deseado, que en este ejemplo debe tener el siguiente aspecto:



Proceso a seguir:

1. Abrir un *panel frontal* nuevo.
2. Colocar un *"vertical switch"* (paleta *Boolean*), cuyo nombre será *Enable*. Su finalidad será finalizar la adquisición.
3. Emplear la *Labeling Tool* para crear una etiqueta libre para *ON* y *OFF*. Utilizar la *Coloring Tool* para hacer que el borde de dicha etiqueta sea transparente. La *T* en el borde inferior izquierdo de la paleta de colores hace transparente un objeto.
4. Colocar el gráfico (*waveform chart*), situado en la paleta *Graph*. Su nombre será *Random Signal*. El gráfico representará valores aleatorios en tiempo real.
5. El gráfico tiene un display digital que muestra el último dato. Pulsar el botón derecho del ratón situado sobre el gráfico, y seleccionar *Digital Display* del

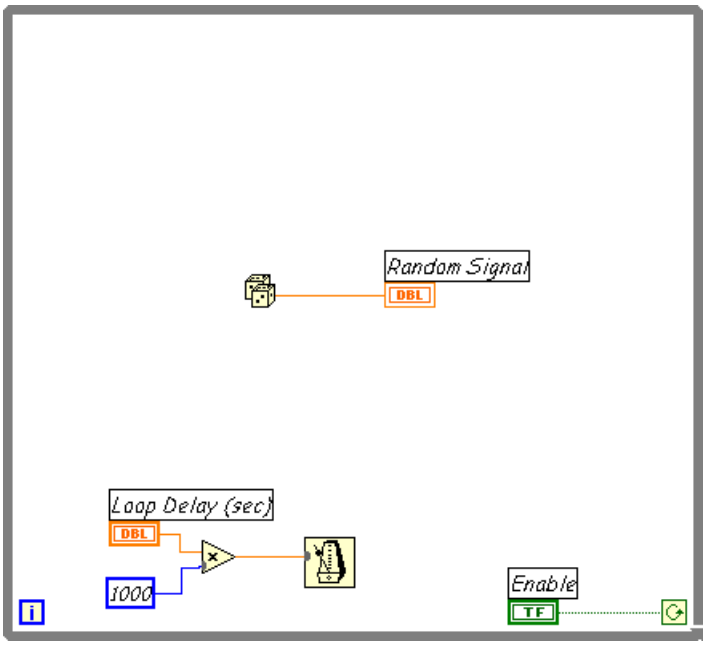
submenú *Show*. Asimismo se deberá deseleccionar *Legend* y *Palette* del mismo submenú .

6. Empleando la *Labeling Tool*, pulsar dos veces con el botón izquierdo del ratón sobre el 10.0 en el eje *Y* del gráfico, introducir 1.0 y pulsar fuera del gráfico. Así se habrá cambiado el fondo de escala.
7. Colocar un *knob* (paleta *Numeric*), cuyo nombre será *Loop Delay (sec)* Este control determinará la velocidad de ejecución del bucle. Pulsar sobre él con el botón derecho del ratón y deseleccionar *Digital Display* del submenú *Show*.

8. Empleando la *Labeling Tool*, pulsar dos veces con el botón izquierdo del ratón sobre el 10.0 de la escala, introducir 2.0 y pulsar fuera del control para introducir el nuevo valor.

6.2.- Diagrama de bloques

El siguiente es el aspecto que presentará el diagrama de bloques una vez finalizada su construcción:



1. Abrir el diagrama de bloques (menú *Window*, opción *Show Diagram*).
2. Colocar el *While Loop* (subpaleta *Structures* de la paleta de funciones). Dicha estructura, como todas las demás es de tamaño ajustable.

3. Seleccionar la función *Random Number (0-1)* de la subpaleta *Numeric* del menú de funciones.

4. Seleccionar la función *Wait until Next ms Multiple* de la subpaleta *Time & Dialog* del menú de funciones.

5. Seleccionar la función de multiplicación de la subpaleta *Numeric*, del menú de funciones, así como una constante numérica, introduciendo el valor 1000 en lugar de 0, que es el que aparece por defecto.

6. Colocar los cables tal y como se muestra en la figura anterior, empleando para ello la *Wiring Tool*.

7. Volver al *panel frontal*. Con la *Operating Tool* poner el interruptor en su posición *ON*. Ejecutar el programa pulsando el botón *run*.
   La frecuencia de ejecución de las iteraciones del bucle *While* es la indicada en el *panel frontal* con el control *Loop Delay (sec)*. Es decir, se generará y representará un valor aleatorio cada periodo de tiempo (en segundos) seleccionado.
8. Para finalizar la ejecución del bucle, colocar el interruptor en la posición de *OFF*. De ese modo la condición de ejecución del bucle *While* será falsa, por lo que se detendrá a la siguiente iteración.