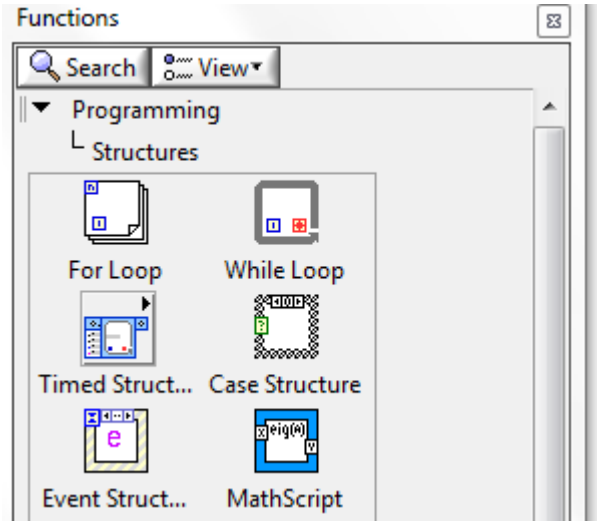
A-0213

CAPITULO 3

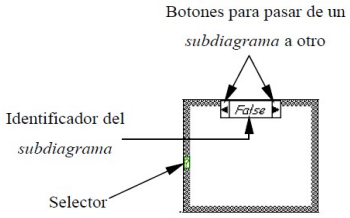
3.1 ESTRUCTURAS



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*.

3.1.1 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.

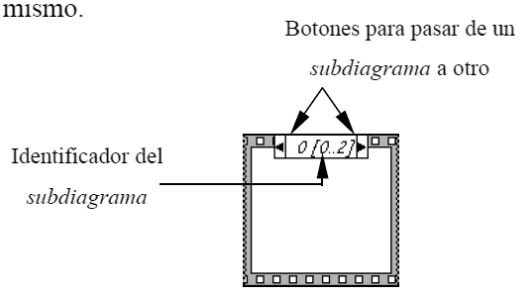
La estructura case consta de un terminal llamado selector y un conjunto de subdiagramas, cada uno de los cuales esta dentro de un case o suceso y etiquetado por un identificador del mismo tipo que el selector; este será booleano o numérico. Si se conecta un valor booleano al selector, la estructura tendrá dos Case: False y True. Pero si se conecta un valor numérico la estructura podra 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
1: ejecutar case 1,
n: ejecutar case n

Case no cuenta con los registros de desplazamiento de las estructuras iterativas pero si 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.

3.1.2 Sequence Structure

mismo.

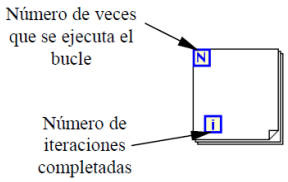


Este tipo de estructuras presenta varios *subdiagramas*, superpuestos como en una baraja de cartas, de modo que únicamente se puede visualizar una en pantalla.
También poseen un identificador del *sudiagrama* mostrado en su parte superior, con posibilidad de avanzar o retroceder a otros *subdiagramas*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

Esta estructura no tiene su homologa en los diferentes lenguajes convencionales, ya que estos 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 despues, 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 ejecutara en orden de aparición: Primero el frame 0 o marco 0, despues el frame 1 y así, sucesivamente, hasta el ultimo.

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

3.1.3 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.

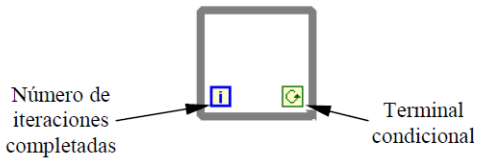
Usaremos For Loop cuando queramos que una operación se repita un número determinado de veces. Su equivalente en lenguaje convencional es:
For i= to N-1

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

1.- Terminal contador: Contiene el número de veces que se ha ejecutado la estructura. El valor del contador se fijara externamente (ver también Arrays en el capitulo 6).
2.- Terminal de interacción: 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 subgdiagrama pero en ningún caso se podrán modificar.

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.

3.1.4 While Loop



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

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 Rrpeat-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 el conectaremos la condición que hará que se ejecute el subdiagrama. LabVIEW comprobara el estado de este terminal al final de cada iteración, si su valor es TRUE(Verdadero) continuara, pero por el contrario si su valor es FALSE (Falso) detendrá la ejecución.

2.- Terminal de iteración: indica el numero de veces que se ha ejecutado el bucle y que como mínimo, siempre será una (i=O).

3.2 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, identificables por el nombre asignado en la etiqueta. Pero puede ocurrir que queramos utilizar el valor de cierta variable en otros subdiagrama o en otro VI o, simplemente, que queramos guardar un resultado intermedio. La forma mas sencilla de hacerlo es generado variables locales y/o globales dependiendo de la aplicación.

3.2.1 VARIABLES LOCALES

En las variables locales los datos se almacenan en algunos de los controles o indicadores existentes en el panel frontal del VI creado, es por eso que estas variables no sirven para intercambiar datos entre VI´s. La principal utilidad de estas variables radican 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 del siguiente menu 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 el variable local será imprescindible que el control tenga asignado un nombre de identificador. 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 LabVIEW no actualiza los cambios.
- Show Label: Muestra una etiqueta con el nombre del VI al que pertenece la variable local.

- Descripción: Permite añadir comentarios.
- Remplace: Sustituye la variable local por cualquier otra función.

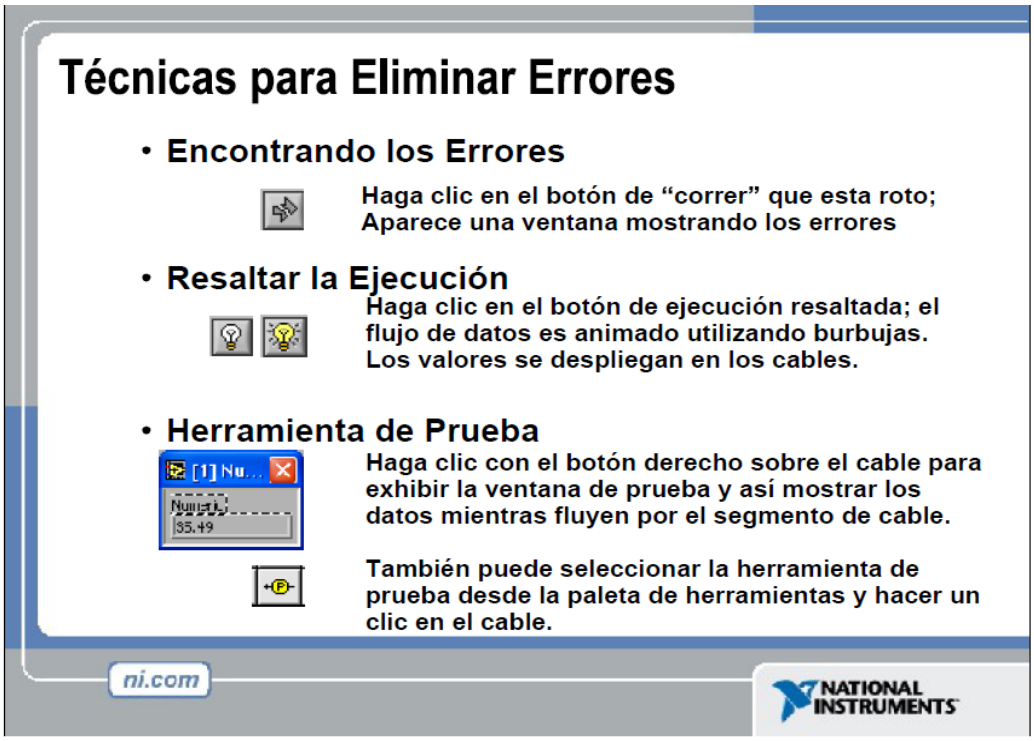
3.2.2 VARIABLES GLOBALES

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

Cuando escogemos la función Global del menú Structs & constants creamos un nuevo terminal en el diagrama, este terminal corresponde a un VI que inicialmente no contiene ninguna variable. Para poderlas añadir haremos doble clic en el terminal y se habría 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 mas 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, bastara con volverlo a abrir e introducir los cambios necesarios. Para añadir nuevas 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 Function 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 Changes To Write Global.

3.3 ERRORES



Cuando su VI no es ejecutable, se despliega una flecha quebrada en el botón de correr en la paleta de herramientas.

Encontrando los Errores: Para hacer una lista de los errores, haga clic en la flecha quebrada. Para localizar el objeto malo, haga clic en el mensaje del error.

Resaltando la Ejecucion: Anima el diagrama y traza el flujo de datos, permitiéndole ver los valores intermedios.

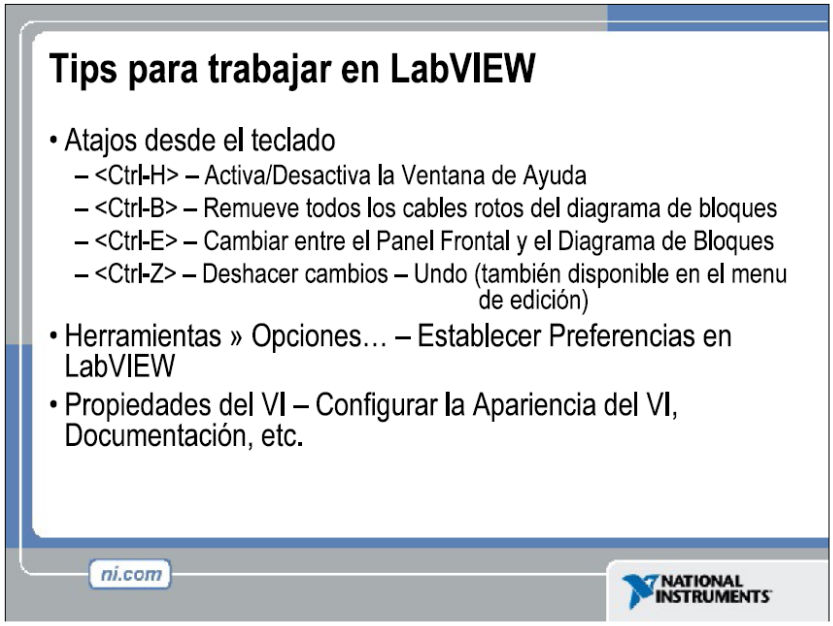
Haga clic en el bombillo incandescente **(light bulb)** en la barra de herramientas.
Probe: Utilizado para ver los valores en los arrays (arreglos) y clusters .

Haga clic en los cables con la herramienta **Probe** o haga clic derecho en el cable para ajustar los PROBES.

Punto de Paro (Breakpoint): Coloca pausas en diferentes lugares del diagrama.
Haga clic en los cables o en los objetos con la herramienta de **Punto de Paro** para colocar los puntos de paro.

Utilice el VI **Debug Demonstrate** del BASICS.LLB para demostrar las opciones y las herramientas

3.4 TIPS



LabVIEW tiene muchas teclas de atajo que hacen el trabajo más fácil. Las más comunes se listan en la diapositiva.

Mientras la Herramienta de Selección Automática (Automatic Selection Tool) es genial para escoger la herramienta que usted desearía utilizar en LabVIEW, a veces hay casos cuando usted desea controles manuales. Utilice la tecla Tab para cambiar entre las cuatro herramientas mas comunes (Operate Value (valor de operacion), Position/Size/Select, (Posición/Tamaño/Selección), Edit Text (editar texto), Set Color on Front Panel (establecer color en panel frontal), Connect wire on Block Diagram (conectar cable en el diagrama de bloque)). Una vez que se ha terminado con la selección de herramientas, puede presionar <Shift-Tab> para activar la Herramienta de Selección Automática.
En el dialogo de Herramientas >> Opciones, hay muchas opciones configurables para el Panel Frontal, Diagrama de Bloque, Colores, Impresión y mucho mas.

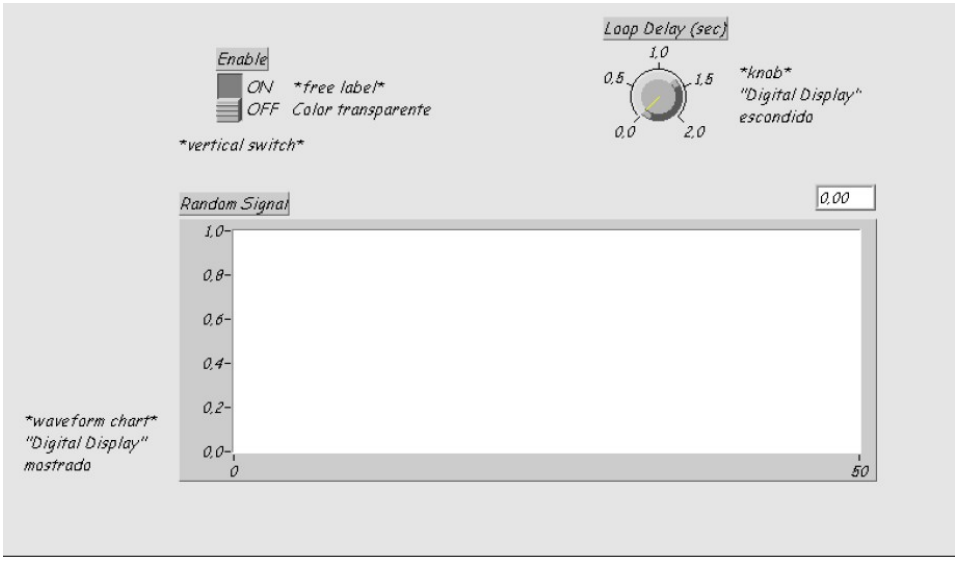
Similar a las opciones de LabVIEW, se pueden configurar propiedades especificas del VI al ir a File >> VI Properties Ahí se puede documentar el VI, cambiar la apariencia de la ventana, y personalizarlos de varias maneras.

3.5 EJEMPLO: CONSTRUCCIÓN DE UN VI

Se mostrará cómo construir una aplicación mediante el empleo del entorno de programación que proporciona LabVIEW.

Panel frontal

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



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.

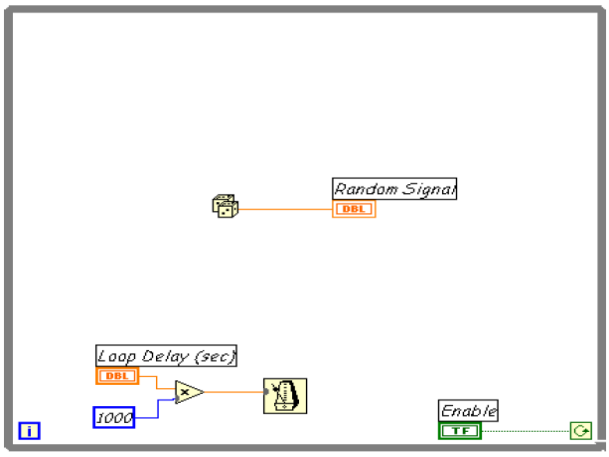
6. Empleando la *Labeling Tool*, pulsar dos veces con el botón izquierdo del ratón sobre el 10.0 en el eje *Y* del gráfico, introducir 1.0 y pulsar fuera del gráfico. Así se habrá cambiado el fondo de escala.

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.

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.