INTRODUCCION

Al estudiar profundamente la configuración de los sistemas de adquisición de datos modernos DAQ (Data Acquisition System), basados en equipos PC (Personal Computer), se aprecia que una de las partes que componen dichos sistemas, es el software quien controla y administra los recursos del ordenador, presenta los datos, y participa en el análisis.

Viendolo de este modo, el software es un tópico muy importante que requiere de especial cuidado. Para los sistemas DAQ se necesita de un software de instrumentación, que sea flexible para futuros cambios, y preferiblemente que sea de facil manejo, siendo lo mas poderoso e ilustrativo posible.

Programas y lenguajes de programación que cumplan con lo dicho existen en gran número en el mercado actual, como por ejemplo el Visual Basic, el C, el C++, el Visual C++, Pascal, LabWindows CVI, Labview, y muchos otros confeccionados específicamente para las aplicaciones que los necesiten.

Para elaborar los algoritmo de control y toma de datos en los proyectos de sísmica, se consideró que el lenguaje más apto es el LabVIEW (Laboratory Virtual Engineering workbench), y las razones son varias:

- Es muy simple de manejar, debido a que está basado en un nuevo sistema de programación gráfica, llamada lenguaje G.
- Es un programa enfocado hacia la instrumentación virtual, por lo que cuenta con numerosas herramientas de presentación, en gráficas, botones, indicadores y controles, los cuales son muy esquemáticos y de gran elegancia. Estos serían complicados de realizar en bases como C++ donde el tiempo para lograr el mismo efecto sería muchas veces mayor.
- Es un programa de mucho poder donde se cuentan con librerías especializadas para manejos de DAQ, Redes, Comunicaciones, Análisis Estadístico, Comunicación con Bases de Datos (Útil para una automatización de una empresa a nivel total).
- Con este las horas de desarrollo de una aplicación por ingeniero, se reducen a un nivel mínimo.
- Como se programa creando subrutinas en modulos de bloques, se pueden usar otros bloques creados anteriormente como aplicaciones por otras personas.
- Es un programa que permite pasar las aplicaciones entre diferentes plataformas como Macintosh y seguir funcionando.

Tomando en cuenta todo lo anterior, se considera de gran utilidad para los estudiantes, los docentes y empleados de la universidad tener un conocimiento básico del método de programación y del manejo de estructuras que posee, por tanto este libro trata de ser una guía básica, que puede usarse para intruducirse en el manejo del LabView.

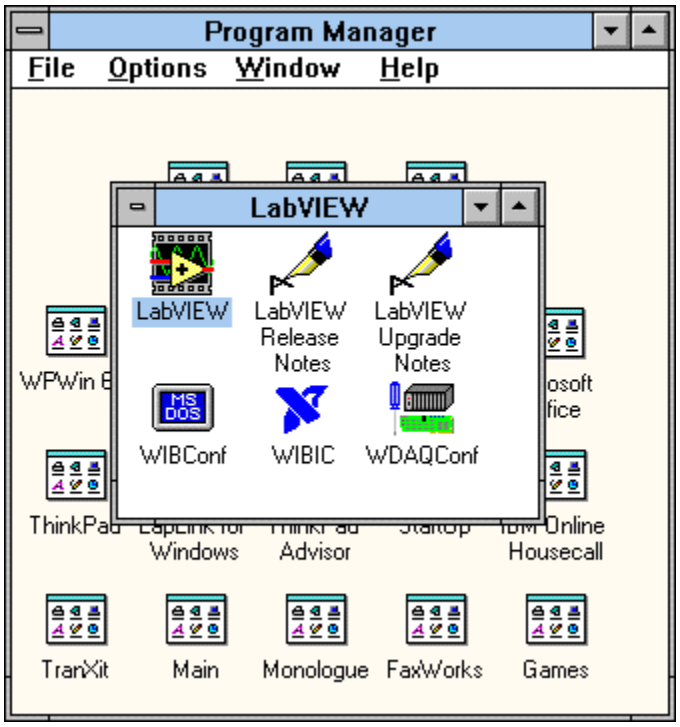
Se dejan algunos ejemplos aclaratorios útiles en control, drivers, y para concluir se explica como funciona el programa desarrollado sobre LabView para la aplicación de Ingeniería Biomedica para poder asi darle soporte.

REQUERIMIENTOS

Como la plataforma más usada en nuestro medio son los PC, en términos de los mismos, lo mínimo para correr LabView, es:
Un micro 386 con coprocesador. Como se requieren muchas operaciones de punto flotante, es indispensable el coprocesador. Los modelo a parir del 486Dx2 en adelante vienen con el coprocesador incluido en si mismos.
Por uso de memoria, se recomienda usar 8 megas de RAM mínimo.
Si se usa un Demo con 2 megas en disco duro basta. Para el paquete completo es bueno disponer entre 40 y 50 megas de espacio en disco duro.
Como se aprecia el requerimiento es alto, pero hoy en día es posible conseguir un ordenador de este tipo a un precio mínimo, y en decenso día a día.

CARGANDO LABVIEW

Después de haber instalado exitosamente LabView, existirá un grupo de iconos correspondientes en Windows.
-El LabView es el programa principal.
-WIBIC es un programa para configurar puertos de tipo GPIB.
-WDAQConf es usado por LabView para configurar las tarjetas insertables de la Nacional Instruments que se usan en la adquisición de datos.



1. INTRODUCCION AL LABVIEW

El LabView es un lenguaje de programación de alto nivel, de tipo gráfico, y enfocado al uso en instrumentación. Pero como lenguaje de programación, debido a que cuenta con todas las estructuras, puede ser usado para elaborar cualquier algoritmo que se desee, en cualquier aplicación, como en análisis, telemática, juegos, manejo de textos, etc.
Cada programa realizado en LabView será llamado Instrumento Virtual (VI), el cual como cualesquier otro ocupa espacio en la memoria del ordenador.

USO DE LA MEMORIA:

La memoria usada la utiliza para cuatro bloques diferentes como son:

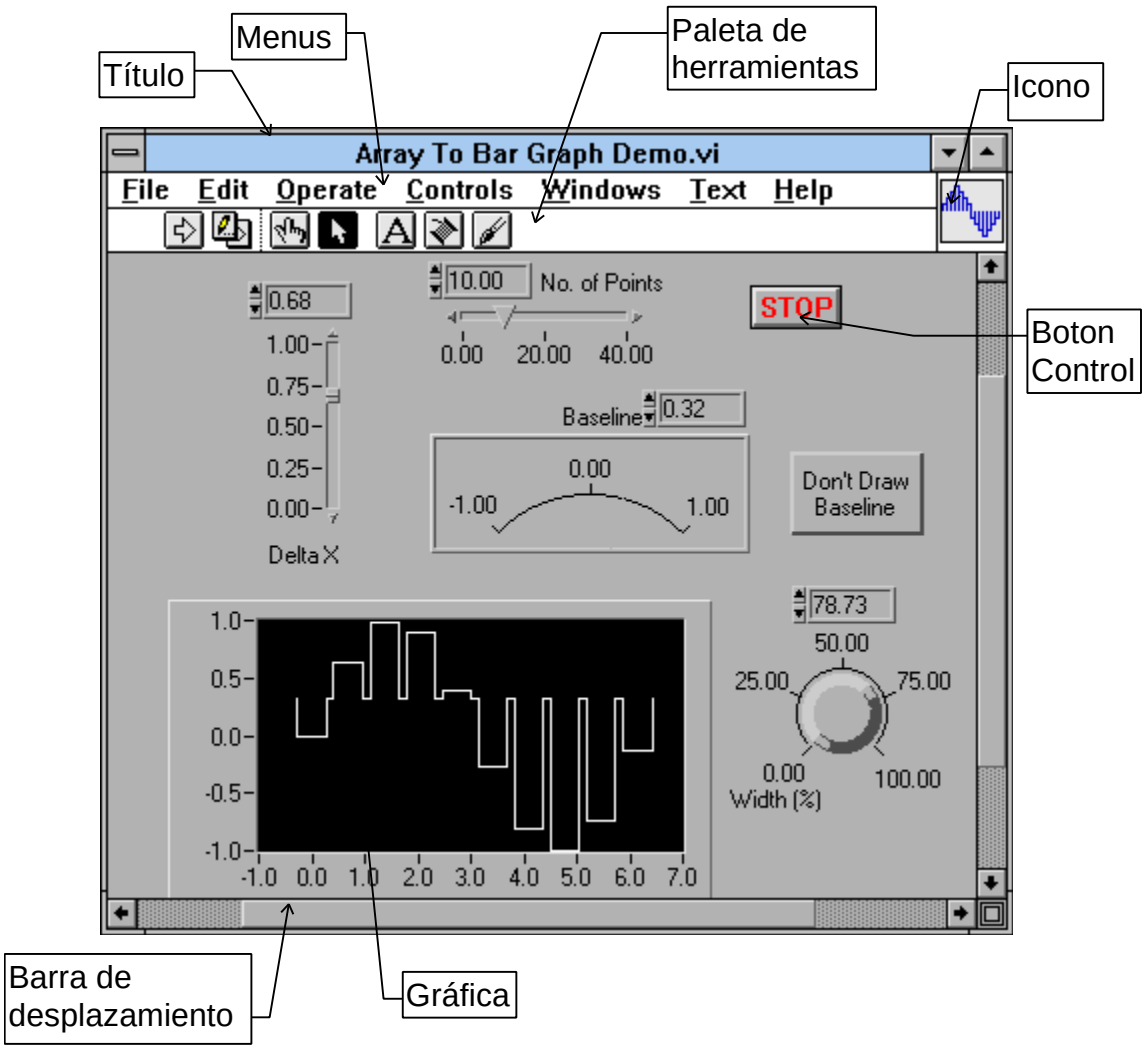
- EL PANEL FRONTAL: Donde se ven los datos y se manipulan y controlan.
- EL DIAGRAMA DE BLOQUES: En este se aprecia la estructura del programa, su función y algoritmo, de una forma gráfica en lenguaje G, donde los datos fluyen a través de líneas.
- EL PROGRAMA COMPILADO: Cuando se escribe en LabView, el algoritmo escrito de forma gráfica no es ejecutable por el ordenador, por tanto, LabView lo analiza, y elabora un código assembler, con base en el código fuente de tipo gráfico. Esta es una operación automática que ocurre al ejecutar el algoritmo, por tanto no es importante entender como sucede esto. Lo que si es algo para apreciar, es que en este proceso, se encuentran los errores de confección que son mostrados en una lista de errores, donde con solo darle doble click al error, se aprecia en el diagrama de bloques, donde ocurre éste, para su corrección.
- LOS DATOS: Como el algoritmo maneja datos, requiere de un espacio en memoria para estos, lo que hace tomar en cuenta que el ordenador usado debe tener la memoria suficiente para manejarlos. Por ejemplo, cuando se usan grandes matrices en calculos se puede requerir de mucho espacio.

Nota: A un programa VI terminado se le puede borrar el diagrama de bloques para que ocupe menos memoria, y no pueda ser editado, y seguirá funcionando. El panel nunca puede ser borrado.

INSTRUMENTOS VIRTUALES

Un programa creado en LabVIEW es llamado como Instrumento Virtual y consta de tres partes a crear.

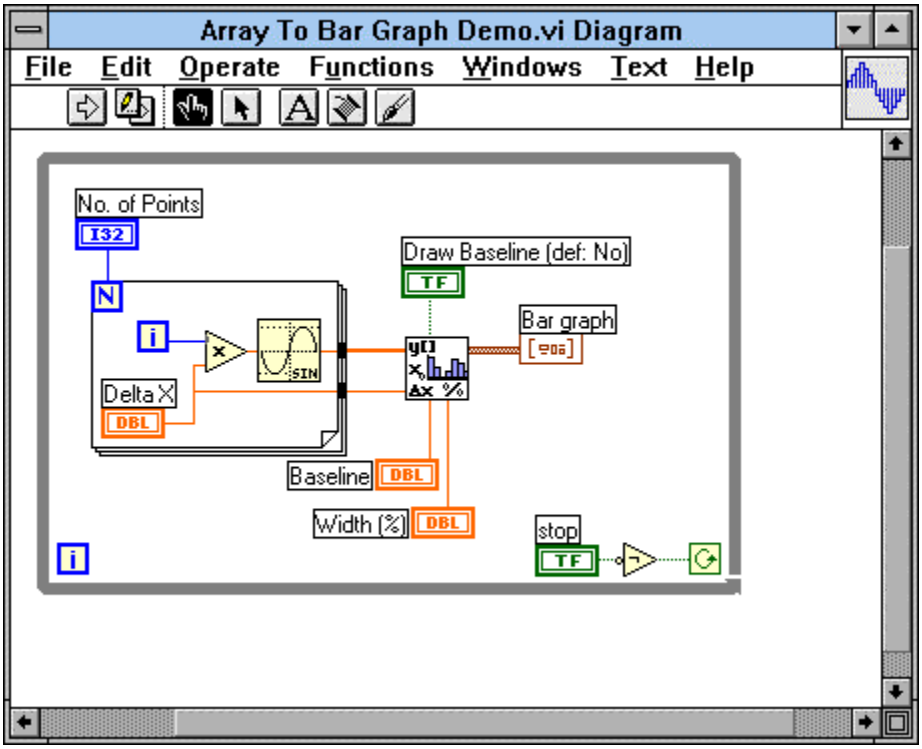
- El Panel frontal , donde estarán ubicados todos los indicadores y controles que el usuario podrá ver cuando el programa este en funcionamiento. Por ejemplo botones, perillas, gráficas,etc.



- El diagrama de bloques muestra el programa en código gráfico G, el cual es el objetivo de aprendizaje en un nivel básico, en este documento. Se usan en este diagrama estructuras de programación, y flujo de datos entre las diferentes entradas y salidas, a traves de líneas. En este las subrutinas son mostradas como iconos de cajas negras, con unas entradas y unas salidas determinadas, donde en el interior se cumple una función específica. El flujo se aprecia, como se dibujaría en un bosquejo de sistemas, cuando se habla

de teoría de sistemas, donde cada subsistema se representa como un cuadro con entradas y salidas.

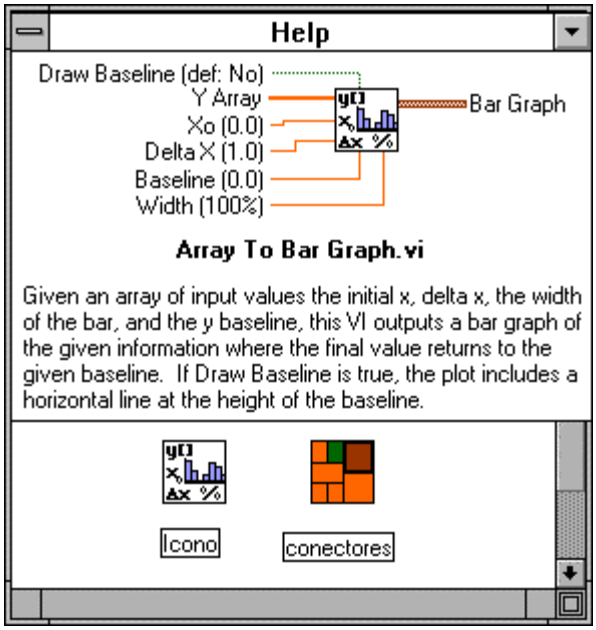
Todos los indicadores y controles ubicados en el panel frontal están respaldados por un terminal de conexión en el diagrama de bloques tal como si se tubiera un tablero de control de una máquina o un avión, donde por el frente se ven los indicadores y por el lado posterior se aprecian todos los cables y terminales de connección.



- El ícono de conexión. Se usa para utilizar el programa creado como subrutina en otro programa, donde el ícono será la caja negra, y las entradas son las conexiones a los controles del programa subrutina, y las salidas son las conexiones a los indicadores del mismo subprograma. Al crear el ícono, se conecta a través del alambre de soldadura a los indicadores y controles en la forma que se desee que se distribuyan las entradas y salidas en la caja negra, tal como en un circuito integrado algunos pines corresponden a alguna función en él. La idea es crear un sistema de programación modular, donde cada rutina creada llame otras rutinas, y estas a su ves otras de menor nivel, en una cadena jerárquica con cualquier límite deseado. Así cuando se use un módulo, no se requiere saber como funciona interiormente, simplemente solo basta conocer sus entradas y salidas para ser así usado.

Para saber el uso de los subvis, la ventana de “help” ofrece la información pertinente a las entradas y salidas. Esta ventana se puede obtener presionando Ctrl-h o por medio del menú “Windows”

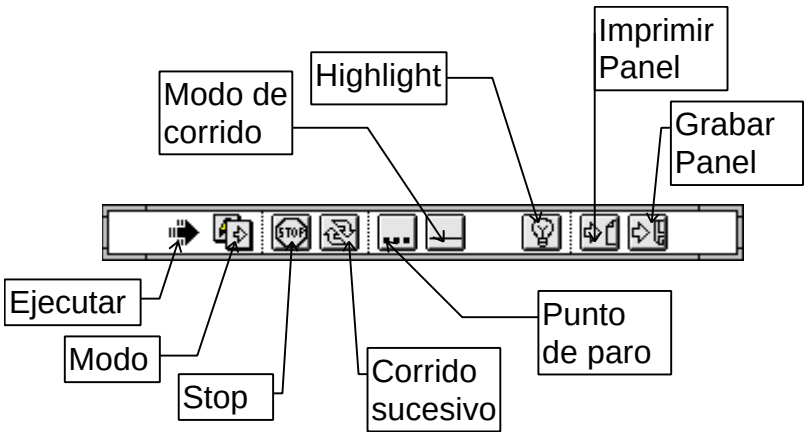
Actualmente existe una asociación de usuarios de LabView donde los miembros estan creando cajas negras de diferentes funciones, las cuales pueden ser usadas para utilidades propias.



PALETAS DE TRABAJO

Tanto en el panel frontal como en el diagrama de bloques, existe una paleta de herramientas, que sirve tanto para editar el VI, o ejecutarlo según el modo de trabajo que se tenga.

Cuando se trabaja en modo de ejecución la paleta es la de la figura.



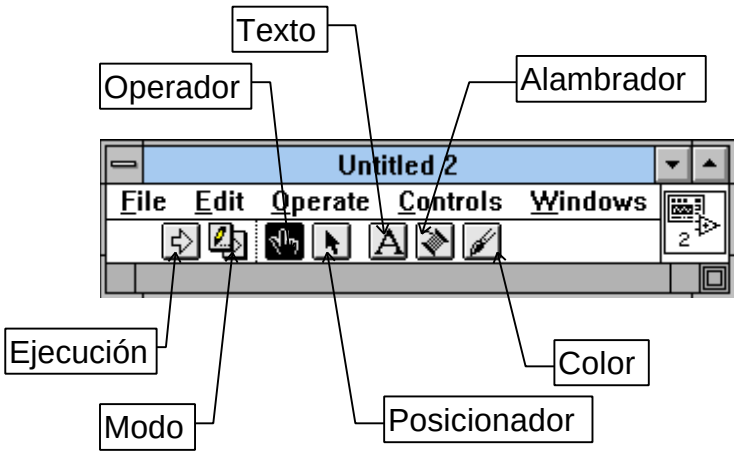
- Con el botón **"Ejecutar"** se corre una ves el programa. Cuando está ejecutando, se cambia a rayado como se aprecia en la figura y aparece un botón de **"Stop"** con el cual se pede detener el programa. No es

recomendado hacer esto, es preferible crear un alagoritmo de paro del programa, con un botón destinado esclusivamente para esto.

Algunos programas al terminar deben de ejecutar algunas operaciones de cierre, como puede ser en la programación de tarjetas de adquisión de datos, o en el cierre de archivos, por tanto si se usa el botón de stop, este parará el programa totalmente, en el punto en el que se encontraba y no permitirá que complete sus rutinas de cierre, pudiendo incurrir en errores y perdida de la información.
Cuando la flecha aparece rota indica que hay un error en el programa. Al hacer click se muestra una lista de errores, y al hacer click en cada uno de los errores se apreciará en el diagrama la ubicación de la falla.

- **“Modo”** cambia entre modo de edición y modo de ejecución. Así está en modo de ejecución.
- **“Corrido sucesivo”** hace que el programa ejecute una ves tras otra hasta que se le de un paro con el boton de stop.
- **“Punto de paro”** al ser presionado cambia a **“!”**, así, al ser llamado como subrutina, abrirá el panel frontal para mostrar como cambia, para encontrar errores de lógica, o por simple visualización.
- **“Modo de corrido”** Al ser presionado cambia a una linea por pasos, así el programa ejecutará paso a paso. cada paso se dará al oprimir el icono de un solo paso.
- **“Highlight”** Muestra como fluyen los datos y que datos, a través de las líneas del diagrama de bloques.”
- **“Imprimir Panel”** Imprime el panel frontal actual cuando termina de ejecutar el programa.
- **“Grabar Panel”** Almacena en un archivo .LOG el estado actual del panel frontal.

En el modo de edición la paleta es la siguiente.

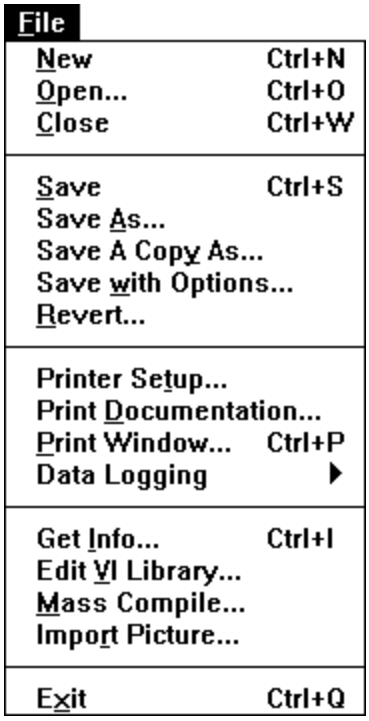


- **“Operador”** Sirve para accionar los controles e indicadores.
- **“Posicionador”** Sirve para cambiar de posición los diferentes elementos en las diferentes pantallas. También permite cambiar el tamaño de estos.
- **“Texto”** Permite crear textos y etiquetas, tanto como cambiar los valores de las escalas de las gráficas.
- **“Alambrador”** Sirve para conectar los elementos en el diagrama de bloques, y para conectar los controles e indicadores a los pines del ícono del programa.
- **“Color”** Permite colorear los diferentes elementos.

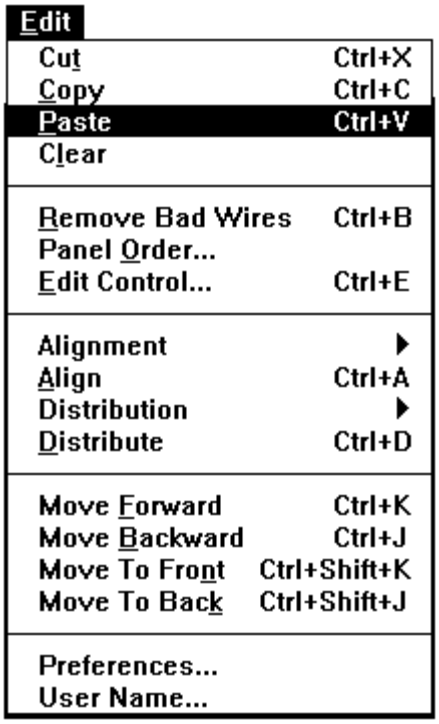
MENÚS DE TRABAJO

Haciendo click en los menús superiores se aprecian las aplicaciones necesarias para trabajar con LabVIEW, como grabar o cargar programas, como editarlos, tipos de letra etc. Los menús se muestran a continuación.

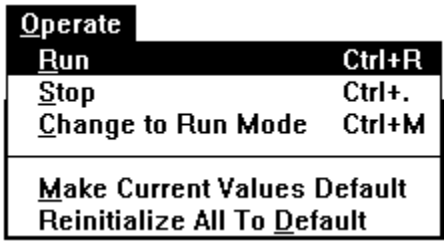
File. En este menú se encuentran las herramientas para el manejo de archivamiento, impresión, y guardado de información de los los programas creados en LabView.



En el menú **Edit** se tienen los comandos para cortar, copiar, pegar y borrar partes; eliminar cables malos y editar controles; alinear y distribuir objetos; cambiar objetos entre diferentes planos; y dar las freferencias de manejo del LabView.

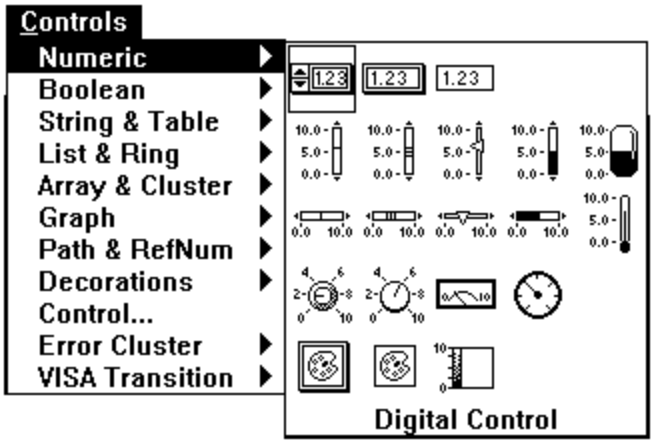


En el menú **Operate** se encuentran herramientas para ejecutar y detener los programas, asi como cambiar el modo de trabajo, y hacer que todos los valores en los controles e indicadores queden como valores iniciales al ser guardado el programa.

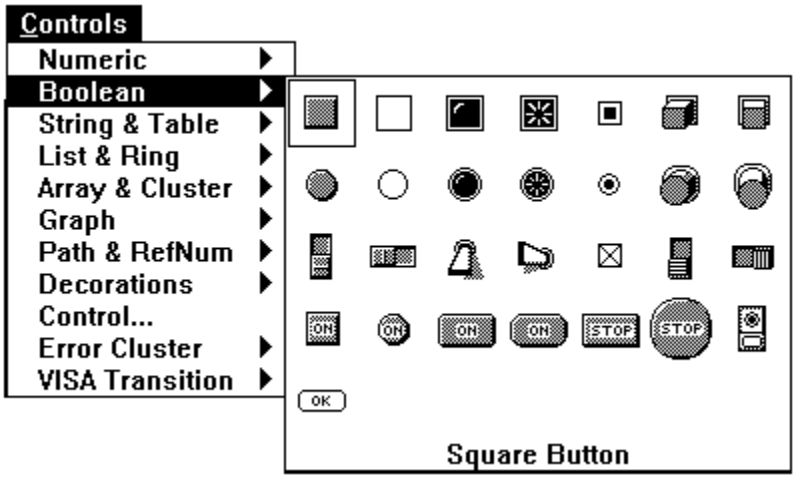


En el menú **Controls** aparecen todos los tipos de controles e indicadores que se pueden colocar en el panel frontal, como son:

1. Numéricos: Permiten la entrada y salida de datos y valores medibles de tipo numérico, ya sea en un número real, enteros, naturales positivos. Por ejemplo un medidor de nivel graficado como un tanque, donde el nivel es el valor dado, o un termómetro, donde la temperatura es un variable continua.



2. Buleanos: Permiten la salida y la entrada de datos de tipo discreto, on-off, como es el caso de los pulsadores, swiches, led's indicadores.

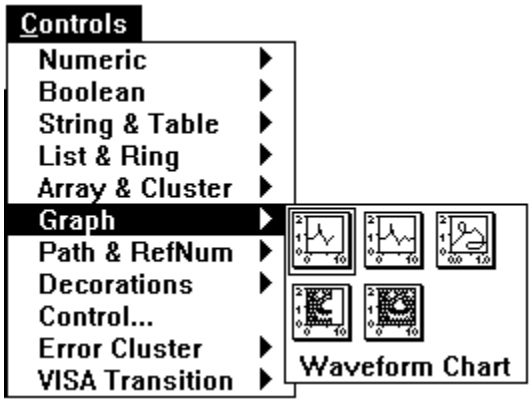


3. String & Table: permite entrar y sacar datos de tipo alfanumérico, vistos en un indicador o control, o en una tabla que tambien puede cumplir las dos funciones.

4. List & Ring: Son controles e indicadores que presentan listas de opciones donde el item seleccionado se entrega como un valor al programa.

5. Array & Clusters: Permite agrupar datos para formar matrices ya sean de entrada o salida. Estas matrices pueden ser de tipo numérico, o de tipo buleano. Tambien se pueden agrupar datos de diferentes tipos de control o de diferentes tipos de indicador, en un cluster, el cual es una agrupación que posee una sola terminal en el diagrama de bloques, semejante a un conector de un ordenador, el cual siendo un solo conector lleva muchas lineas que llevan diferentes señales. en las matrices todas las señales son del mismo tipo.

6. Graph: Controles e indicadores de gráficas. Pueden ser gráficas de barrido, graficas XY, o de tonos de colores.



7. Path & Refnum: Controles útiles en el manejo de archivos.

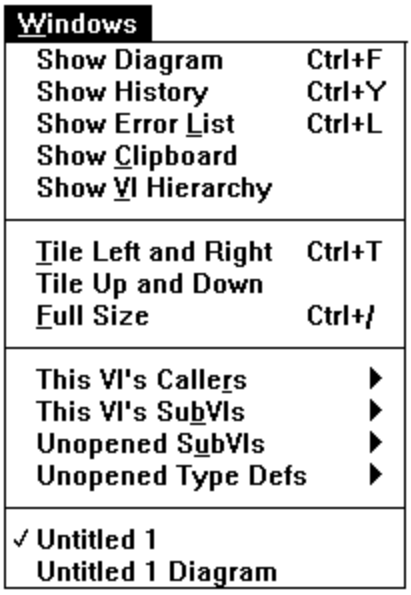
8. Decorations: Se disponen alementos de decorativos para el panel frontal.

9. Controles: Además de poderse ubicar los controles e indicadores presentados en los menús anteriores, también se pueden usar controles editados por el programador, como por ejemplo el dibujo de una bomba, o un pistón neumático.

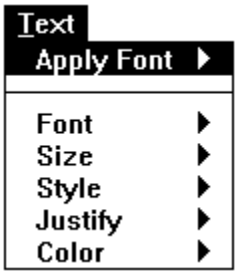
10. Error Cluster: Controles de entrada y salida, para parámetros de algoritmos manejadores de errores.

11. Visa Transition: Utiles para comunicación VISA. No son de uso normal para principiantes.

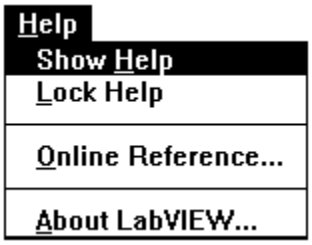
En el menú **Windows** se encuentran las herramientas para hacer cambios entre ventanas de trabajo.
Mostrar diagrama o panel, según la ventana en la que se encuentre.
Mostrar la historia de los cambios en el programa.
Visualizar la lista de los errores que posee el VI o programa.
Desplegar el contenido del Clipboard.
Mostrar el orden jerarqueco, en el cual un programa llama subVis.
Además hay herramientas para ordenar las ventanas, abrir programas que que son usados por el VI principal, y otros.



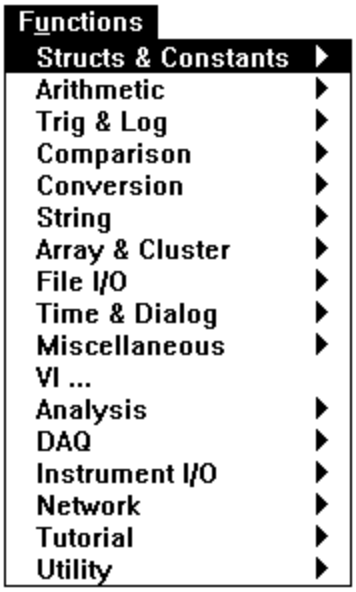
El menú **Text** se encuentran todas las utilidades para seleccionar tipos, colores, estilos y tamaños de letra.



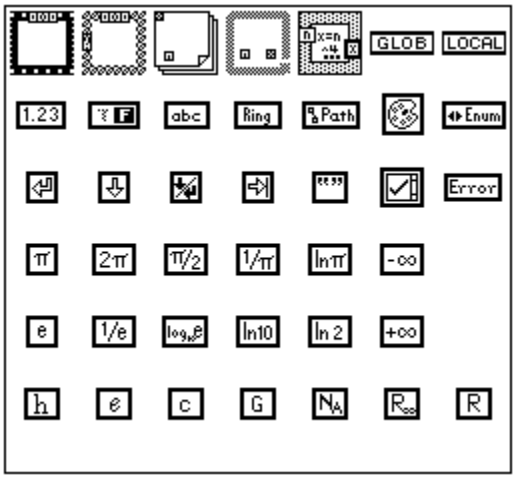
El menú **Help** presenta las ayudas necesarias sobre el programa, y ofrece la opcion para desplegar una ventana donde se explica cada objeto solo con señalarlo. En la ventana mencionada se explica como son las entradas y salidas de cada subVi, y de cada función.



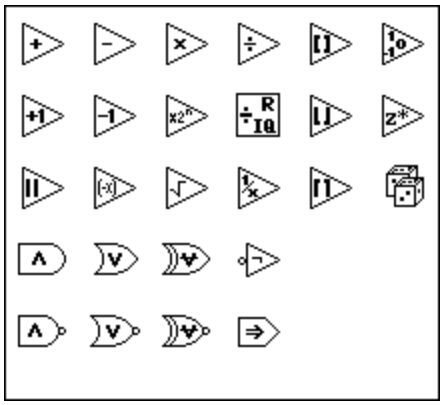
El menú de **Functions** ofrece todas las posibilidades de funciones que se pueden utilizar en el diagrama de bloques, donde al hacer click se escoje y ubica dentro del programa.



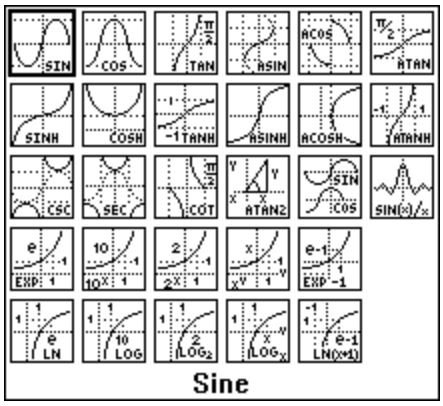
1. Structs & Constants: Contiene las estructuras básicas de programación como son las secuencias, los casos, los ciclos For-Next y Mientras, las variables de tipo global y local, y las constantes de todo tipo, como son las numéricas, las alfanuméricas, las boleanas, y algunos numeros especiales, “e“ por ejemplo.



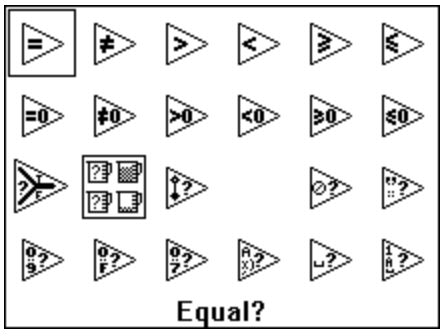
2. Arithmetic: Presenta las operaciones básicas aritméticas como son suma, resta, multiplicación, números al azar, valor absoluto, compuertas and, or, not y muchas otras. Para ver la función de cada una usar la ventana de Help <ctrl-H>.



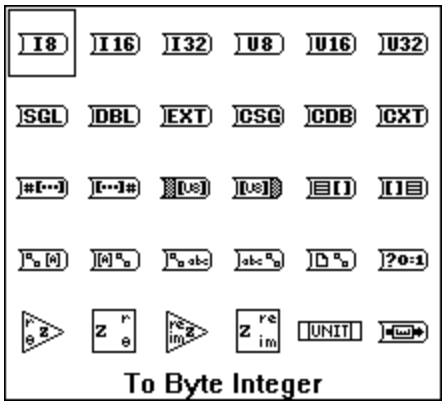
3 Trig & Log: presenta funciones trigonometricas y logaritmicas.



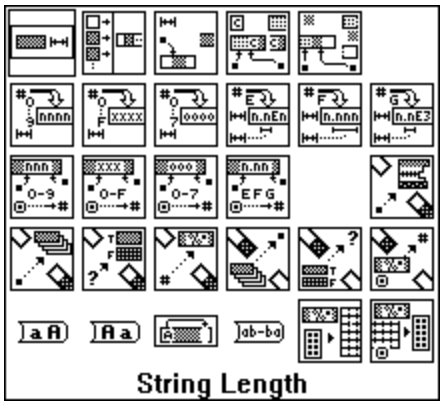
4 Comparison: Funciones de comparación que devuelven un valor de verdadero o falso según se cumpla dicha comparación.



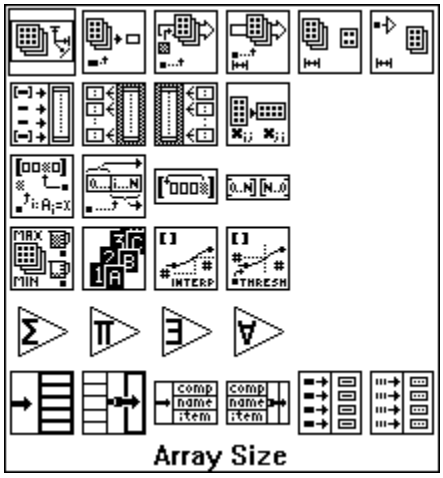
5. Conversion: Conversiones de tipos de variables, de un formato a otro, por ejemplo convertir un número a otro que ocupe 32 bits en memoria, o convertir un número a una matriz de buleanos cuya representación en binario corresponda al número.



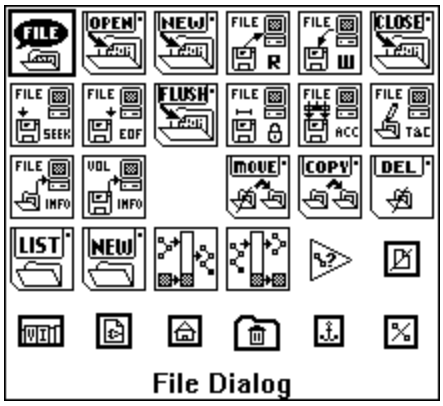
6. String: presenta herramientas para manipular cadenas de caracteres. Por ejemplo connvertir todos los caracteres a mayúsculas, o reportar el valor de la longitud de la cadena.



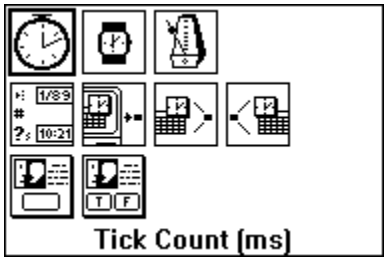
7. Array & Cluster: Maneja las herramientas para el uso de matrices y agrupaciones. Ej. dar las dimensiones de una matriz, en otra de una sola dimensión. Ej agrupar un conjunto de cables en uno solo par manipular menos líneas. El manejo de matrices y clusters será mejor esplicado adelante.



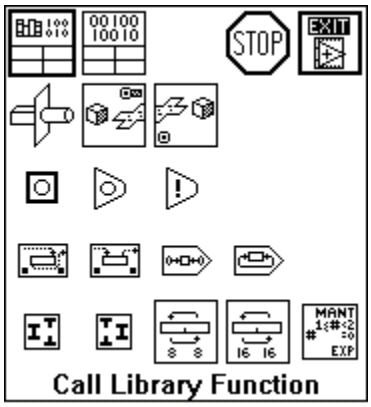
8. File I/O: Para el manejo de archivos y almacenamiento de información en disco.



9 Time & Dialog: Reportadores de tiempo, esperas, fechas, y cuadros que dan anuncios.

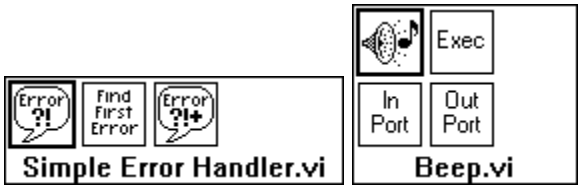


10 Miscellaneous: Bloques de llamada a codigos en C, o a librerías dinámicas de windows DLL. Conversión de datos a binario; manejadores de ocurrencias para ordenar el flujo de datos. Y otras funciones de uso mas avanzado.



11 Vi: Para llamar bloques creados como rutinas.
12 Analysis: Funciones avanzadas de procesamiento de señales, estadísticas, álgebra lineal, filtros, regresión y otras que requieren de un buen entendimiento matemático.

13 DAQ: Para la adquisición de datos, lectura y escritura de datos a las tarjetas insertables, toma y control de señales análogas y digitales, y control de los circuitos contadores que hay en algunas tarjetas.
14 Instrument I/O: Comunicación con instrumentos medidores a través de puertos GPIB, serial o VISA.
15 Network: Para la comunicación de ordenadores en red, y enlace entre diferentes aplicaciones, como es el caso del DDE, Dynamic Data Exchange, que puede servir para enlazar aplicaciones de LabView con Bases de datos como ACCES, para actualizarlas simultaneamente los hechos van ocurriendo. Otros parametros son los de comunicación TCP y UDP para comunicación en red. Todo esto requiere de un aprendizaje especial.
16. Tutorial. Erramientas para el uso de ejemplos de adquisición de datos sin tener las tarjetas insertables.
17. Utility: Utiles para el manejo y análisis de errores en los programas creados. Utiles para el control de los VI (Abrir un VI por ejemplo). Manejadores especiales de archivos. Manejadores de puertos inport y outport.



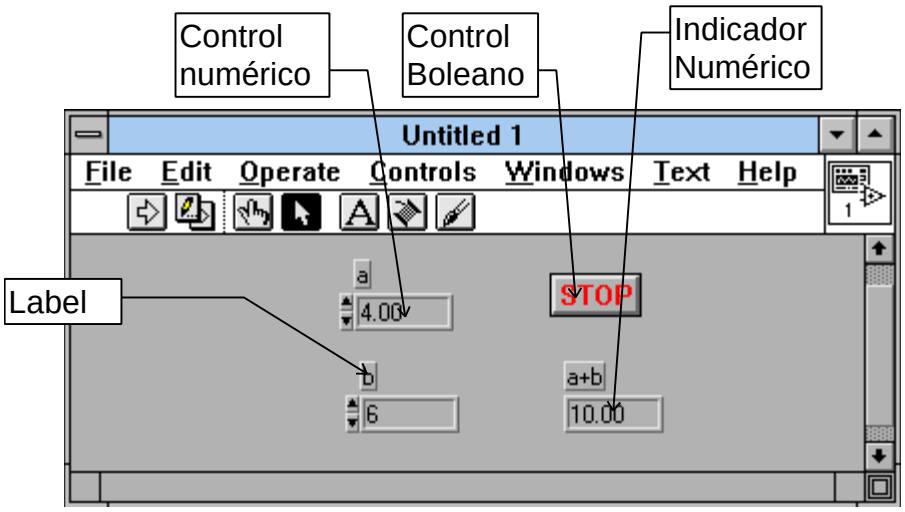
DDE (Dynamic Data Exchange)
PC (Personal Computer),
VI (Instrumento Virtual)**,**
LabVIEW (Laboratory Virtual Engineering workbench),
DAQ (Data Acquisition System),

2. COMO CREAR PROGRAMAS

PANEL FRONTAL

Al desarrollar una aplicación o una subrutina primero se debe tener un claro conocimiento de que valores se van a utilizar, cuales van a ser las entradas y cuales las salidas, para así definir como se van a entrar y sacar estos valores.
Por ejemplo si simplemente se desea realizar un programa que tome dos números y entregue como resultado la suma de estos hasta que se pulse un botón de stop, al final diga que terminó, se sabe que debe haber un instrumento de control para la entrada de cada valor, y un indicador que muestre el resultado.

Crear lo anterior se logra simplemente ubicandose en el panel frontal y sacando dos controles y un indicador del menú Controls. Esto se hace uno a uno, y se debe ir nombrando cada elemento en el label, a medida que se van posicionando.



Se aprecia como en estos instrumentos digitales se diferencian los controles del indicador porque estos cuentan con unas flechas para manipularlos cuando el programa está corriendo. También se pueden cambiar escribiendo sobre ellos.
Estos controles se pueden configurar sacando el Pop-menú de cada uno, señalandolo y oprimiendo el botón derecho del mouse, así si por ejemplo se comete un error al nombrar el instrumento y no se alcanza a escribir el nombre, en este menú en la subsección **show, label,** se puede hacer que reaparezca la marca para así escribir sobre ella.

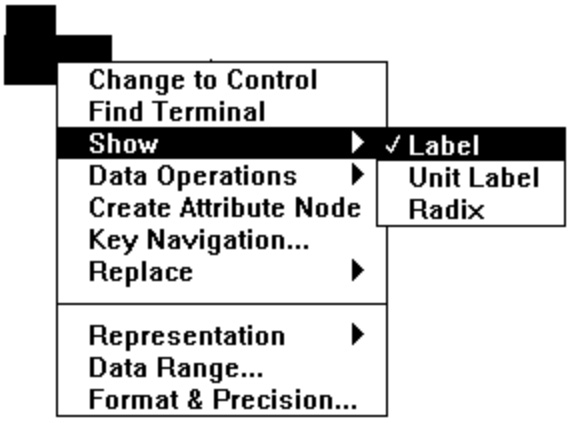
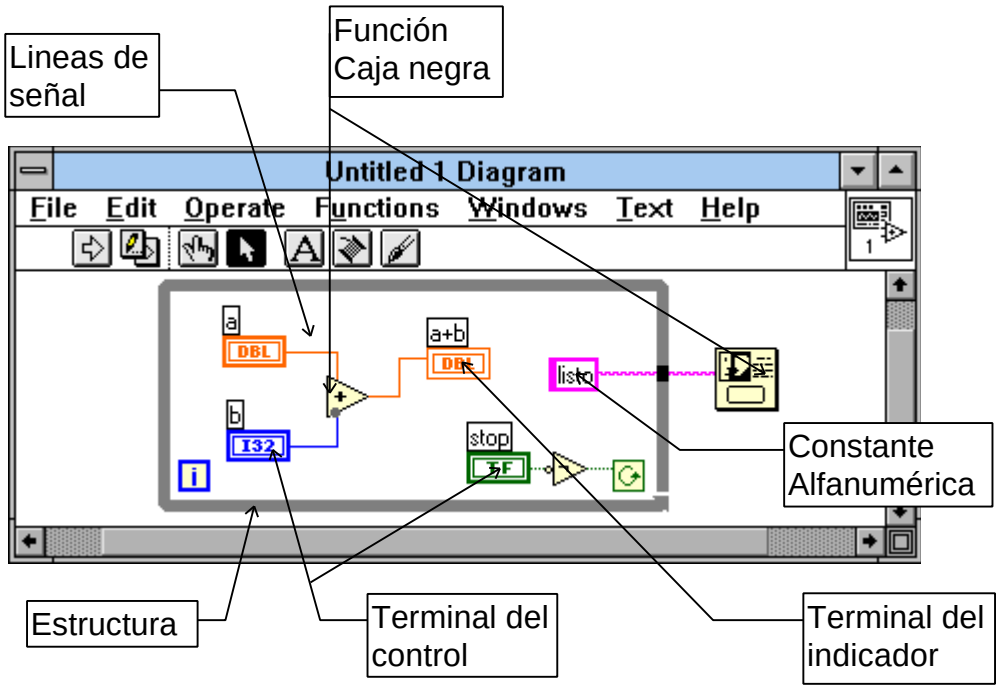


DIAGRAMA DE BLOQUES

En éste se ve el flujo del programa, y se compone de cinco tipos de elementos.

- Las terminales de conexión de los indicadores y de los controles del panel frontal. Se nota que las líneas del dibujo de la conexión de los controles es más gruesa que la de los indicadores, para diferenciarlos.

- Las constantes.
- Las funciones y cajas negras, donde se pocesan las señales.
- Las estructuras de programación.
- Los cables que conducen las diferentes señales, los cuales varían según la señal que conducen.

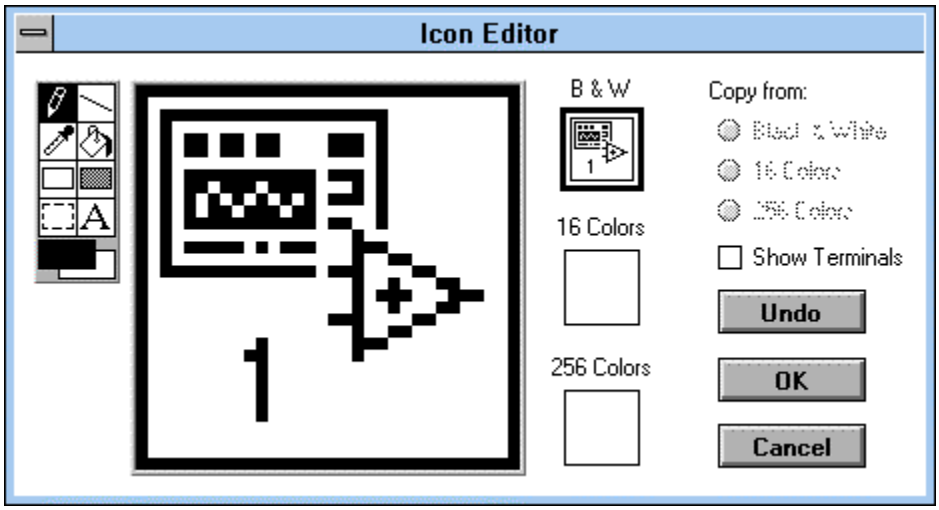


Para realizar el diagrama de bloques se buscan las estructuras necesarias en el menú de functions, estructuras y constantes, donde se encuentra el ciclo mientras (While), el cual será esplicado luego.
Posteriormente se ubican las funciones necesarias en el menú de Functions, como en este caso el sumador y el negador en el submenú arithmetic, y el cuadro de diálogo en el submenú Time & Dialog.
Los terminales aparecen automáticamente en el digrama de bloques al armar el panel frontal.
Por último se hacen las conexiones con ayuda de la herramienta de alambrado.

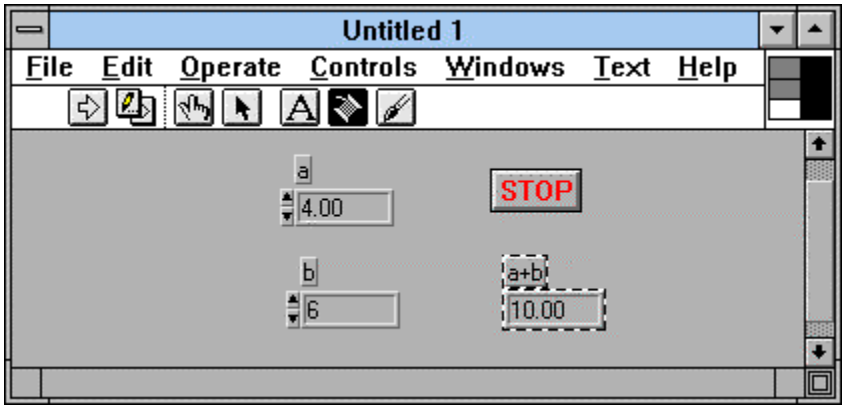
CONEXIÓN DE ICONO

Si se desea que el programa realizado sirva como subrutina para otro VI de mayor jerarquía, como primero se debe realizar un Icono que represente el VI, y luego hacer las conexiones entre los terminales del ícono y los instrumentos del panel frontal. Cabe anotar que se conectan solo los deseados. Los que no se conecten tomaran el valor que poseen como Default, o valor propio inicial correspondiente, para las funciones y operaciones que se deban realizar.

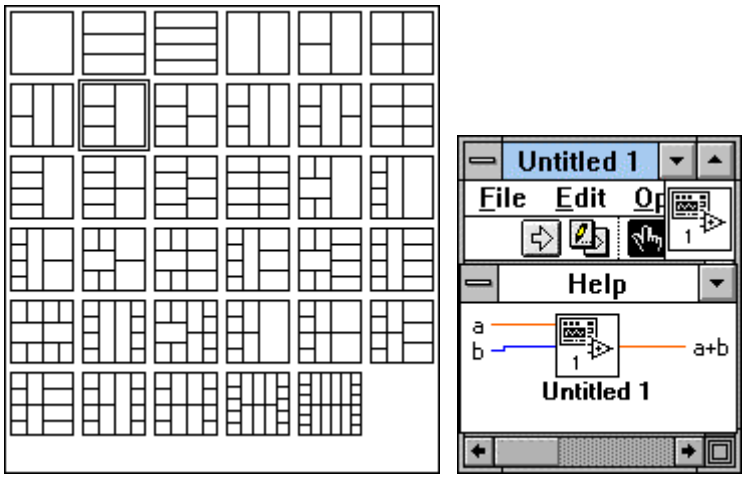
Para editar el icono se selecciona con el botón derecho del ratón en el icono del panel frontal y se selecciona **'Edit Icon'**. En este editor se puede dibujar el icono deseado.



Después de tener el icono deseado se muestran los conectores por medio de **'show connector'** en el mismo pop-up menú y con la herramienta de alambrar se hacen las conexiones con los dispositivos del panel haciendo primero click en el indicador o control y luego en el pin del ícono deseado. Es recomendable conectar las entradas a la izquierda y las salidas a la derecha.



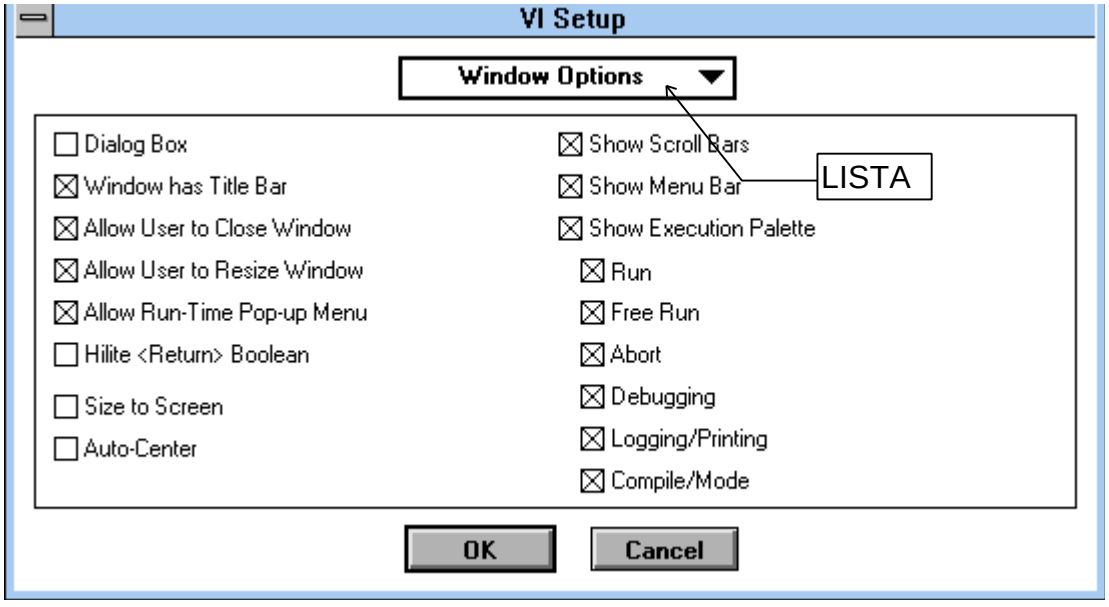
Si se requieren mas conectores, se puede cambiar el esquema de conexiones, por medio de **Patterns** en el pop-up menú, cuando está mostrando los pines.
Cuando está lista la conexión la ventana de help muestra como quedan las entradas y salidas.



GUARDAR PROGRAMAS Y CARGARLOS

PROPIEDADES DE LOS VI

Antes de guardar un VI, se puede configurar éste para que cuando sea cargado ejecute inmediatamente, sin presionar ningún botón. Se puede también lograr que cuando ejecute, no muestre paletas, o la barra de título, que quede centrado, o que no se le pueda modificar el tamaño a la ventana del panel. Todo esto por medio de la opción **VI-Setup** en el pop-up menú del icono principal. En la lista se escoge el tipo de parámetro a configurar (window, execution, history).



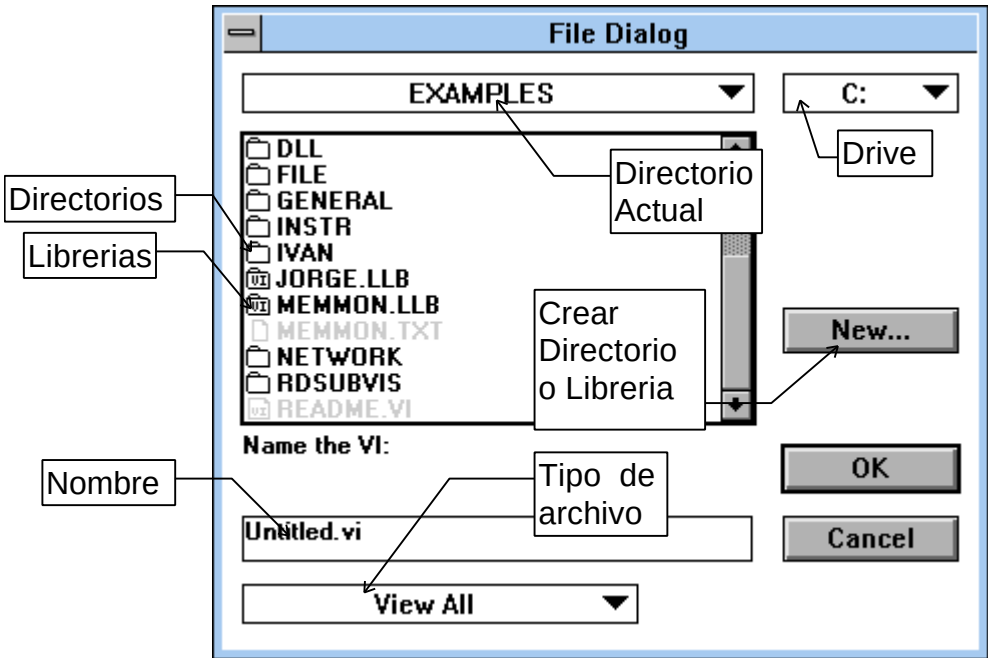
DIRECTORIOS DE ALMACENAMIENTO

Por manejo de memoria, labview permite almacenar los datos, programas y otros, en dos tipos diferentes de directorios, entendibles por LabView.

- Los directorios normales en los que se almacenan los programas con extensión .vi, y el nombre no puede tener más de ocho caracteres para los PC.

Existen un archivo .llb el cual es una libreria en la que solo LabView es capaz de almacenar, y el cual el la entiende como un directorio. Tiene la ventaja de tener internamente comprimidos los programas, economizando memoria en disco. Además los nombres de los programas no tienen restricciones.

OPCION DE GUARDAR
Por medio de la opción SAVE AS, se despliega un menú con los directorios y las librerias en las cuales se puede almacenar programas. Dando click en NEW, se puede crear un directorio nuevo o una librería nueva.



El proceso para cargar es de la misma forma, por medio de OPEN.
El comando Save with Options permite grabar en una librería todos los archivos requeridos para correr una aplicación, pero esto corresponde a un nivel más avanzado.

3. MANEJO DE DATOS EN UN VI

TIPOS DE VARIABLES Y DATOS NUMÉRICOS

NUMERO DE BITS EN UN NUMERO
El ordenador posee una memoria compuesta de una gran lista de números, los

cuales son llamados bytes, que son un conjunto de unos o ceros, llamados bits. Cada byte se compone de ocho bits los cuales pueden representar un número de 0 a 255. Para poder almacenar números mayores se requiere de más bytes, 16 o 32 bits. Este número se relaciona con el número de bits con los que puede trabajar el microprocesador del ordenador, en cuanto a la velocidad de operación. Además un número de más bits ocupa mayor espacio en memoria.

SIGNO EN EL NUMERO

Como se tiene un código binario, hay métodos para dar el caracter de positivo o negativo a un número, dejando bits que representen el signo. Cuando se opera con números con signo el método es diferente a como se hace con números sin signo.

NUMEROS FRACCIONARIOS

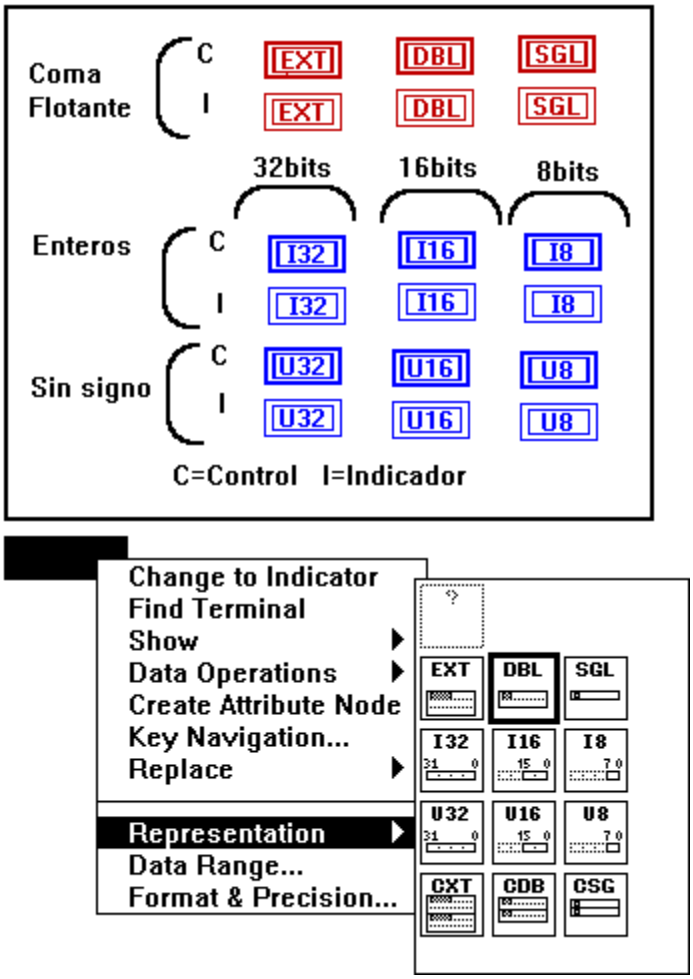
Igual que con el problema del signo, se requieren de algunos métodos para representar la coma en un código binario, y las operaciones también varían. De hecho se requiere de muchos más cálculos para un microprocesador para sumar dos números de coma flotante (que posean coma, fraccionarios), que para sumar dos enteros sin signo. Para esto el microprocesador se vale del coprocesador matemático, que hace operaciones de coma flotante a gran velocidad. Los números de coma flotante dependen del número de bits, para tener una mayor exactitud.

Según lo anterior hay números de tipo entero ‘I’ de 8, 16 y 32 bits, de tipo sin signo (unsigned U) de 8, 16, 32, o de coma flotante de tipo simple (SGL 16), doble (DBL 32), y Extendido (EXT 64 bits). Igualmene números complejos simples, dobles y extendidos.

El tipo de número se aprecia en el terminal de conexión de los controles o indicadores, pues aparece inscrito, y el color de las conexiones de punto flotante son anaranjadas o rojas, mientras que en los enteros y sin signo son azules.

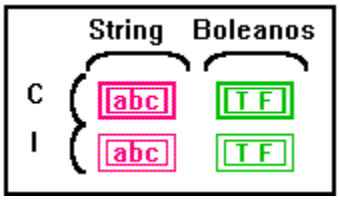
Se recomienda usar datos de menor número de bits, siempre y cuando no se pierda precisión, para que no se ocupe mucha memoria. Los cálculos de punto flotante restan velocidad.

El tipo de dato que manejan los indicadores y controles se configura en el pop-up menú de cada control por la opción **representation**, igualmente con las constantes.



DATOS BULEANOS Y ALFANUMERICOS

Los datos buleanos también tienen su tipo de conector. Para buleanos, el color de las conexiones y los cables es de color verde, y para las de tipo alfanumérico son de color rosado.

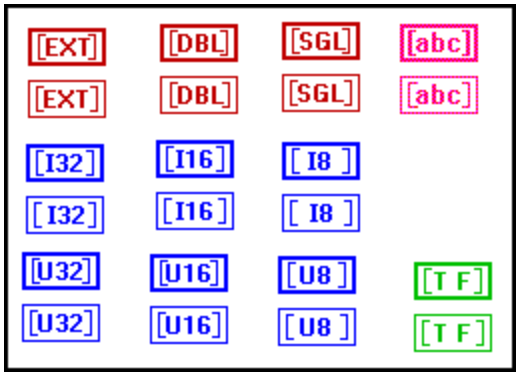


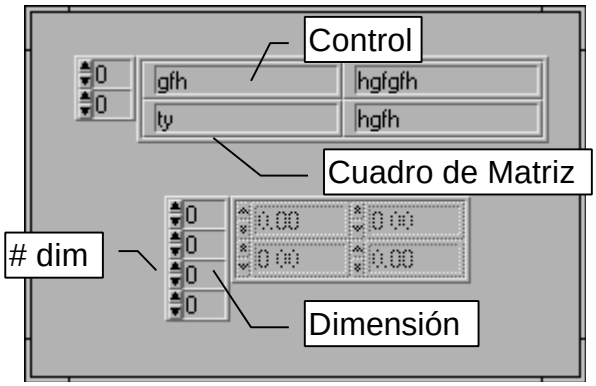
C=Control, I=Indicador

MATRICES

Las matrices son conjuntos de datos de una misma especie. Para crear una matriz se ubica en el panel frontal un cuadro de matriz (Array o arreglo) sacado del menú ARRAY & CLUSTER, y dentro se ubica el control o indicador que se mostrará. Se puede estirar el cuadro para que muestre varios datos

pertenecientes a la misma matriz. Si se estira el display lateral se aumenta el número de dimensiones.
El conector será uno solo para la matriz con todos los datos, y se diferencia de los otros conectores por tener el tipo de datos dibujado entre [ ], en lugar de un rectángulo, así se puede tener una matriz de cualquier clase de número, sea doble, alfanumérico, buleano, etc.
Las líneas o cables que conducen matrices son más gruesos y aumentan de espesor según sea el número de dimensiones que manejen.

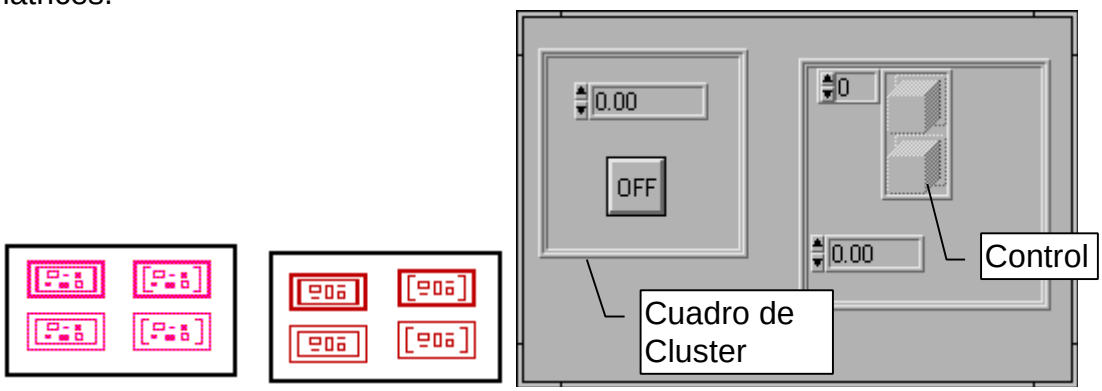




AGRUPACIONES O ESTRUCTURAS

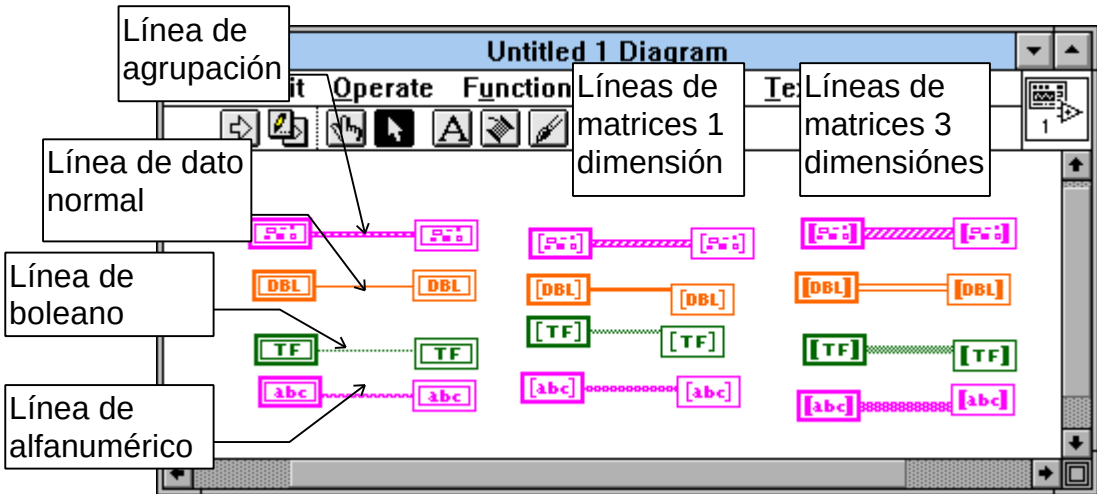
Las agrupaciones o estructuras son conjuntos de datos pero de diferente tipo. Para crear una agrupación se ubica en el panel frontal un cuadro de agrupación (cluster o estructura) sacado del menú ARRAY & CLUSTER, y dentro se ubican los controles o indicadores que se mostrarán.
El conector será uno solo para la agrupación con todos los datos, y se diferencia de los otros conectores por tener dibujado unos cuadritos, en lugar del tipo de dato, así se puede tener una agrupación con cualquier clase de números, sean dobles, alfanuméricos, buleanos, todos mezclados, tal como se agrupan un conjunto de cables del circuito eléctrico de un automovil, donde cada cablecito dentro del cable grande lleva un tipo de dato, y se conecta a un toma donde cada pin tiene un uso, pero en total tienen un solo cometido.
Las líneas o cables que conducen agrupaciones son más gruesos y parecen como mangueras con burbujas.

También se pueden crear matrices de agrupaciones, y agrupaciones de matrices.



CABLES DE TRASMISIÓN

Como se ha mencionado los cables llevan la información de un lado a otro. El cable cambia según el dato que lleve, pero esta es una opción automática que sirve para visualizar en el momento de hacer las conexiones.



POLIMORFISMO

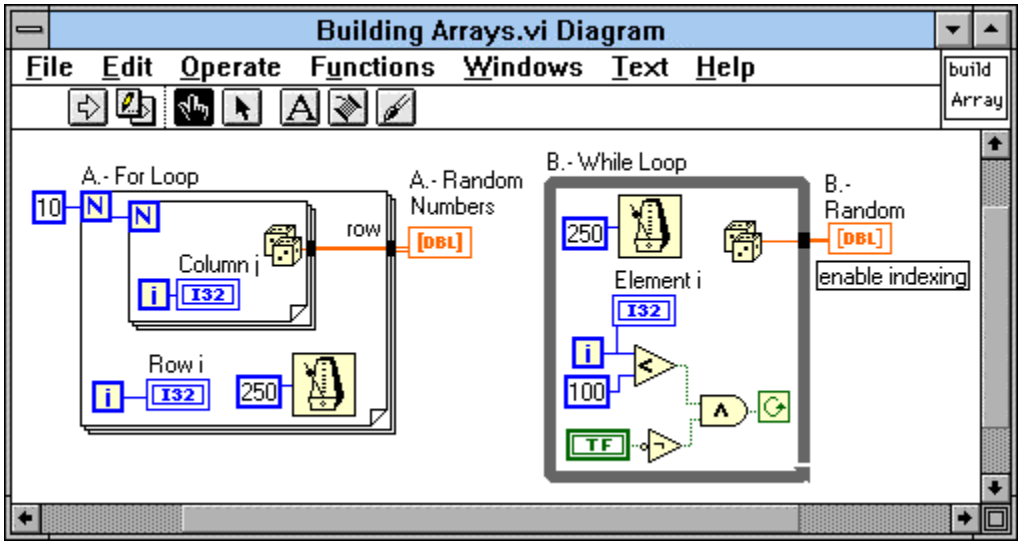
Como se ha mencionado existen números con diferente formato de representación, y según esto al sumar u operar con dos números de diferente clase no es correcto. Si se trata de sumar un número unsigned de 16 y uno de 8 bits, no se tendrá un resultado correcto. Como en el lenguaje C, para hacer este tipo de operaciones se debe convertir el de menor precisión a la mayor para no perder exactitud en el resultado. Una división siempre genera números de punto flotante, por tanto lo correcto es usar este tipo de variables. Para convertir datos se usa un bloque especial, el cual se encuentra en el menú de funciones de conversión.

Sin embargo LabView permite para muchas funciones operar con números de diferente clase en la entrada, sin tomarse como un error que impida la ejecución del programa, lo que se llama polimorfismo. Cuando esto sucede se aprecia un

punto gris (dot) en la conexión, que indica el conflicto. Mirar dibujo en la explicación del diagrama de bloques.

FLUJO DE DATOS EN FUNCIONES

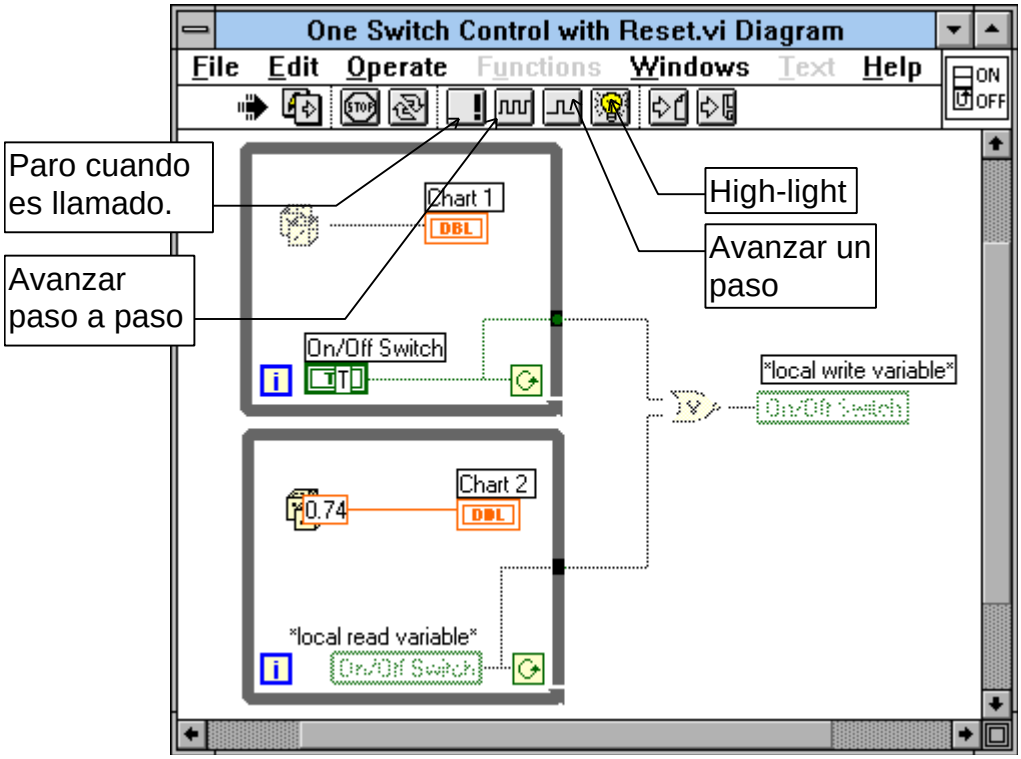
A diferencia de los lenguajes escritos en algoritmo de texto continuo, el LabView es un lenguaje que en cierta forma se puede llamar multiproceso, pues puede ejecutar varias rutinas al mismo tiempo, esto se logra porque el procesador gasta partes de tiempo en cada rutina, dentro de un intervalo de tiempo. Así según un sistema de prioridades se va ejecutando parte de cada programa.
Como se ve en la figura cuando se corre el programa los dos ciclos corren simultaneamente, (cosa que no es cierta en términos de nanosegundos, pero se puede afirmar en segundos). Para hacer que un ciclo corra después de otro se requiere de una estructura que permita esto como es la de secuencia, donde dentro de cada cuadro se ubica el ciclo que se va a realizar.



El flujo de datos a través del programa, se hace a través de los cables que llevan la información a las funciones y a los datos de control a las estructuras. Una función no se ejecuta sino hasta que han llegado todos los datos de entrada, así, en la figura el signo de menor arrojará un dato de verdadero o falso solo cuando hallan llegado los datos de entrada a esta función.
Los datos de salida solo surjen cuando ha cumplido la finción su operación, así mismo ocurre con las estructuras. Es decir, que el dato de salida de la estructura fluirá al resto del programa cuando esta halla concluido, para el caso de la figura, cuando el ciclo haya cumplido todo su número de vueltas.
Se puede usar un ciclo While, o un For-Next para acomular datos en la frontera de salida, y así cuando terminen las iteraciones, tener una matriz como resultado, lo que se logra dando click con el boton derecho en la conexión de salida del ciclo y seleccionando **“Enable Indexing”**.
Con dos ciclos anidados se tendrá una matriz de dos dimensiones de tamaño según el número de vueltas.

Para que no almacene datos en la frontera, seleccionar **“Disable Indexing”**, en el mismo pop-up munú.

Para ver como fluyen los datos a través de el diagrama de bloques se puede hacer click en el boton de high-light ubicado en la paleta de herramientas, para ver como unos puntos luminosos indican los movimientos en dicho diagrama.
Si se desea que esta revisión se haga paso a paso, se debe presionar el icono de marcha a pasos, y presionar en el ícono de un paso para obtener el paso siguiente.
Cuando se llega a una subrutina, normalmente no se ve lo que ocurre adentro. Si se desea que cuando se ejecute ésta porque llegan los datos a ella se abra el panel de esta y se detenga, para ver el flujo dentro, se debe grabar ésta con el ícono de **Paro con Llamada ‘...’**. Cuando esto se hace el ícono cambia a **“!”**.



4. ESTRUCTURAS Y ELEMENTOS DE PROGRAMACIÓN

Para realizar un programa dentro de cualquier lenguaje se requiere del conocimiento de las estructuras que gobiernan un algoritmo. En el LabView como lenguaje tambien cunta con estas.
La estructuras en LabView son:

- Los ciclos While
- Los ciclos For-Next
- Los cuadros de casos

- Las secuencias

Otros elementos de programación son las variables, que pueden ser de tipo global o local, y los cuadros de fórmula.

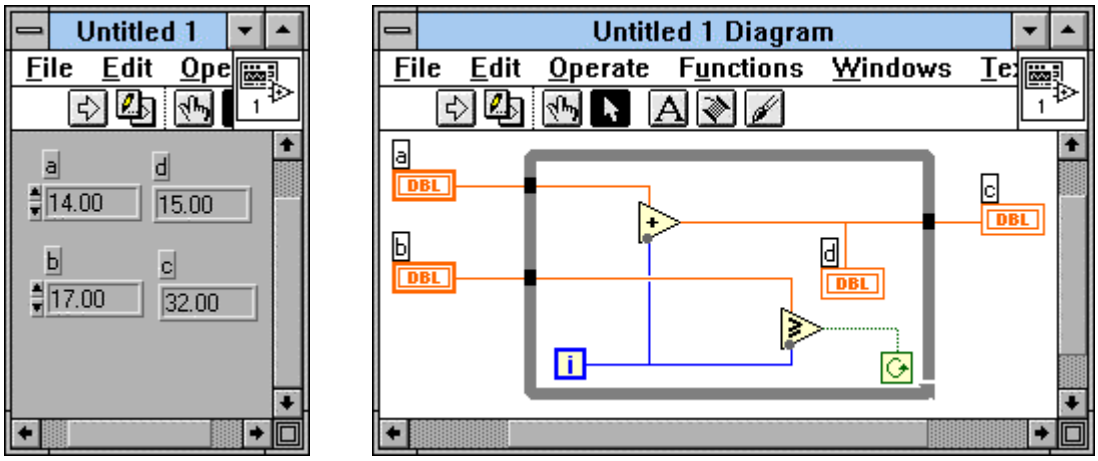
LOS CICLOS WHILE

GENERAL:

Sirven para hacer que una secuencia de instrucciones se repitan una cantidad de veces, siempre y cuando una afirmación sea verdadera. En el LabView se ejecutarán las funciones que se encuentren dentro del cuadro de ciclo, tomando los valores que quedaron almacenados en la frontera de entrada, y sacando los resultados a la frontera de salida. Por ejemplo si se desea contar a partir de un número 'a', durante una cantidad de veces 'b', e ir mostrando el número de conteo en un indicador 'd', y ver el último número contado en 'c', el programa sería el siguiente.

El término 'i' en el ciclo es un contador que se incrementa una unidad cada ves que se repite el ciclo.

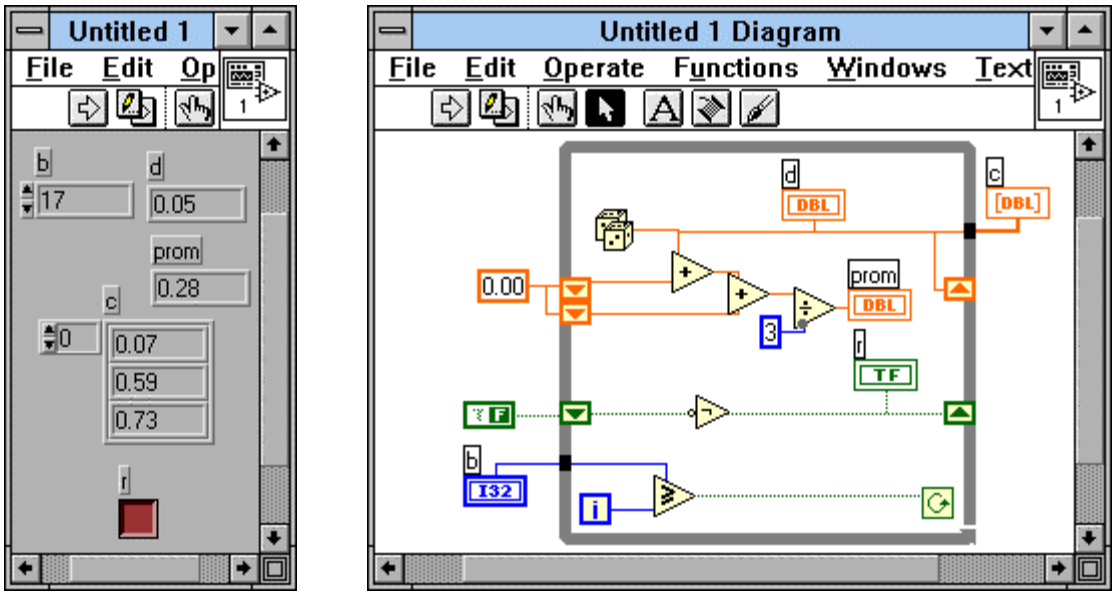
La flecha circular es el parametro que al recivir un valor de true (verdadero), permite repetir el ciclo, y al recibir un falso, lo detiene para que el dato que haya en la frontera de salida valla al indicador c.



Los datos a y b solo llegan una vez a frontera de entrada y allí quedan almacenados en un buffer para ser usados todas las veces que el ciclo repita. Estos datos siempre serán iguales.

En el programa se sumará en cada loop el valor de 'a' con el contador que en cada iteración es mayor en uno. El dato se mostrará en 'd', y se llevará a la frontera de salida, donde se almacena hasta que termine el ciclo. En la iteración siguiente un nuevo dato llega a la frontera borrando el anterior, así cuando el loop para, solo el ultimo valor pasa a 'c'.

Constantemente se evalúa si el número 'b' es mayor o igual al contador. Cuando este contador alcanza 'a' 'b', la comparación se vuelve falsa y el ciclo se detiene.



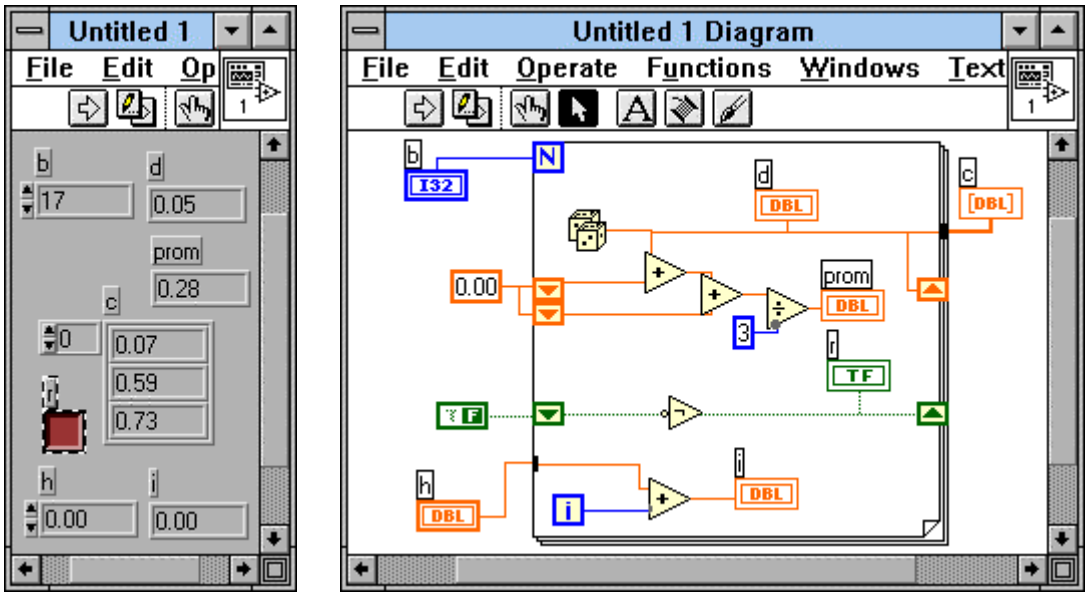
INDEXING:
Los ciclos se pueden utilizar para crear matrices simplemente acumulando los datos en la frontera de salida, sin permitir que el último borre el primero, y más bien apilandolos uno tras otro en matriz. Esto se logra sacando el pop-up menú de el punto negro de la frontera de salida, el cual es el elemento de memoria o buffer, y seleccionande “Enable indexing”. Se aprecia que el cable de salida ahora es mas grueso, y debe llevar los datos a un indicador de matriz.

SHIFT REGISTER:
Se puede hacer que los resultados de un ciclo sirvan como datos para la próxima iteración, mediante unas memorias llamadas Shift Register, las cuales se crean sacando el pop-up menú del ciclo en una de las fronteras. Se crean unas memorias en las fronteras de entrada y salida. Después del ciclo el dato resultado colocado en el shift de la frontera de salida, pasa a ocupar el lugar del shift de la frontera de entrada para participar en las funciones del ciclo. El tipo de dato manejado puede ser cualquiera, como se ve en el ejemplo, se maneja un dato buleano de verdadero falso.
El dato inicial siempre debe ser definido, pues en la primera iteración estas memorias de entrada se encuentran vacías. Esto se logra conectando un valor a las memorias.
En el ejemplo primero se le agrega un ‘falso’ al shift, después en el ciclo es negado y el resultado ‘verdadero’ se muestra en un indicador ‘r’, y se coloca en el shift de salida, el cual será el proximo valor en el shift de entrada, en el próximo ciclo. Se toma el valor del shift de entrada, se niega, y se muestra en el indicador ‘falso’ y de nuevo al shift de salida. Asi sucesivamente, se tiene como resultado un tren de pulsos falso verdadero y un bombillo titilando.
Es posible almacenar no solo datos de la última iteración, sino de la penúltima, y muchas anteriores, agregando shift’s a la entrada, por medio del pop-up menú del shift, con **Add Shift Register**. Asi el ejemplo muestra como tener una secuencia donde se genera una cantidad de números al azar y se calcula el

promedio de los tres últimos números. El indicador ‘d’ muestra el valor actual al azar y los shift almacenan los dos anteriores. Para el caso inicial estos se llenan con cero. El resultado del número al azar se coloca en el shift de salida para que en la próxima iteración pase a la entrada del valor anterior, y en el otro ciclo pase al tras-anterior.

CICLO FOR-NEXT

Se comporta similar al ciclo While. Este hace un número definido de iteraciones el cual esta dado por el valor que se coloca en el parametro ‘N’. Este siempre debe ser definido, pues de no suceder así el programa no se ejecuta.
También se puede usar para crear matrices, y también puede usar valores de ciclos anteriores con los Shift register. Tiene la desventaja respecto al ciclo while de tener que cumplir todas las iteraciones para terminar, mientras que en el while, se termina dependiendo de una condición, por tanto se puede crear un algoritmo que cuando detecte un error termine el ciclo. Mientras que el for-next es un ciclo ciego, el while siempre se está chequeando.

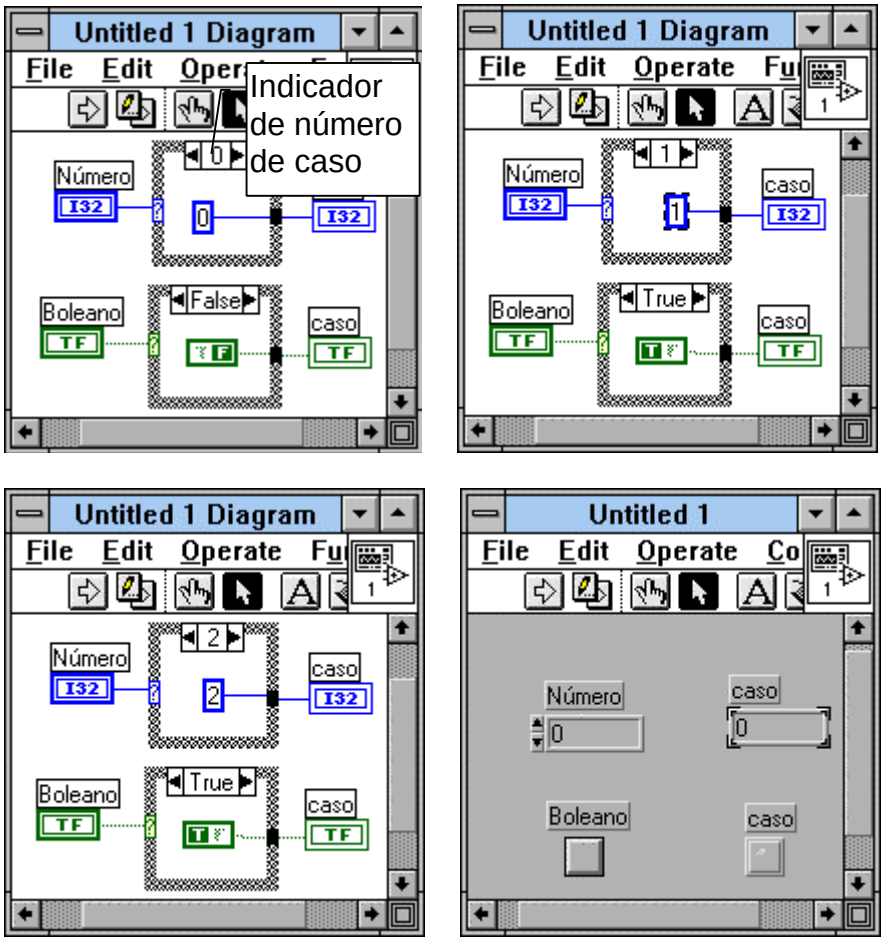


El ciclo For-Next también cuenta con un elemento ‘i’ que sirve de contador para indicar el ciclo actual.
El programa anteriormente realizado con un ciclo While es equivalente al mostrado en la figura. A éste se le ha agregado un contador que suma un valor inicial ‘h’ con el contador, para mostrar en el indicador “i”DBL, un número que va desde h hasta h+b. b es el número de veces que se ejecuta el ciclo por ser el valor que entra a ‘N’

CUADROS DE CASOS ‘CASE’

Es una estructura de comparación y ejecución condicionada donde de acuerdo a algún parámetro se realizan las operaciones de un cuadro u otro. Si el

parámetro de condición es del tipo verdadero-falso cuando éste es verdadero se ejecuta un contenido, y cuando es falso se ejecuta otro. De esta forma solo son posible dos opciones de ejecución.
Si el parametro es un número, se ejecuta un cuadro cuyo número de identificación corresponde al valor de entrada. En este caso pueden haber tantas opciones de ejecuciones como se desee.
Para obtener esta estructura, buscarla en el submenú de **estructuras & constantes**, en el menú de funciones.



Para agregar un cuadro de caso cuando se usa un parámetro de selección numérico, solo basta seleccionar el pop-up menú de la estructura, dando click con el botón derecho y seleccionando “Add Case After” para un caso de número siguiente, o “Add Case Before”. Dentro de este pop menú, se encuentran otros parámetros de control de estas estructuras.
Para ver el contenido de cada caso, solo basta seleccionarlo con las flechas del indicador del caso.
En el ejemplo se aprecian los cuadros de caso de tipo buleano y los de tipo “switch” donde la entrada es numérica. Con solo conectar la entrada, se crea el tipo de caso. Para cada caso en el ejemplo, se conecta una constante a el indicador, que corresponde al caso que se ejecuta.

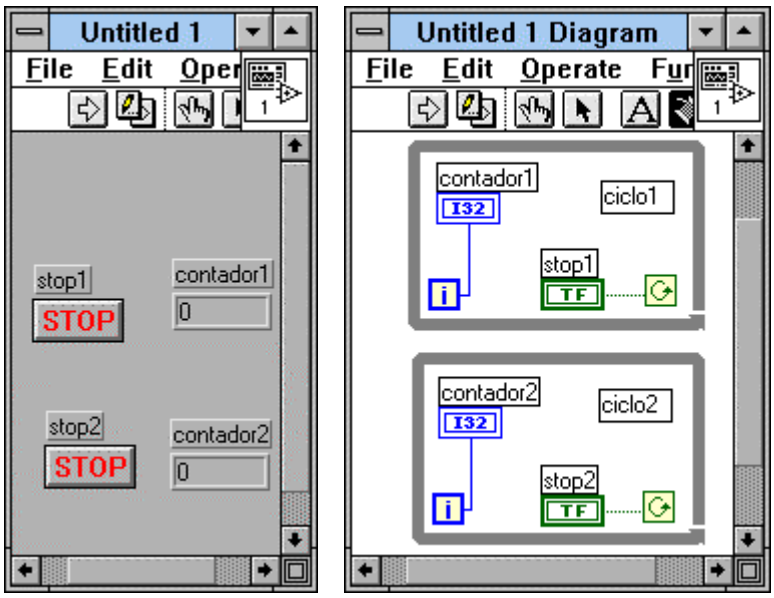
LAS SECUENCIAS

Como el LabView es un lenguaje de tipo multiproceso, puede ejecutar varias partes del programa simultaneamente. Además las funciones se van operando cuando llegan todos los parámetros de entrada de cada una lo que no da mucha certeza de que función se realiza primero. Pero si por alguna razón se desea que un cunjunto de operaciones se realize antes que otro, se puede agregar una estructura de secuencias, la cual ejecuta el contenido del primer cuadro, luego el del segundo, y así sucesivamente tal como en una cinta de fotos para cine, cada foto sigue a la otra.

Para agregar un cuadro adicional tal camo en las estructuras case, se logra por medio del pop-up menú en el borde del marco, Add Frame.

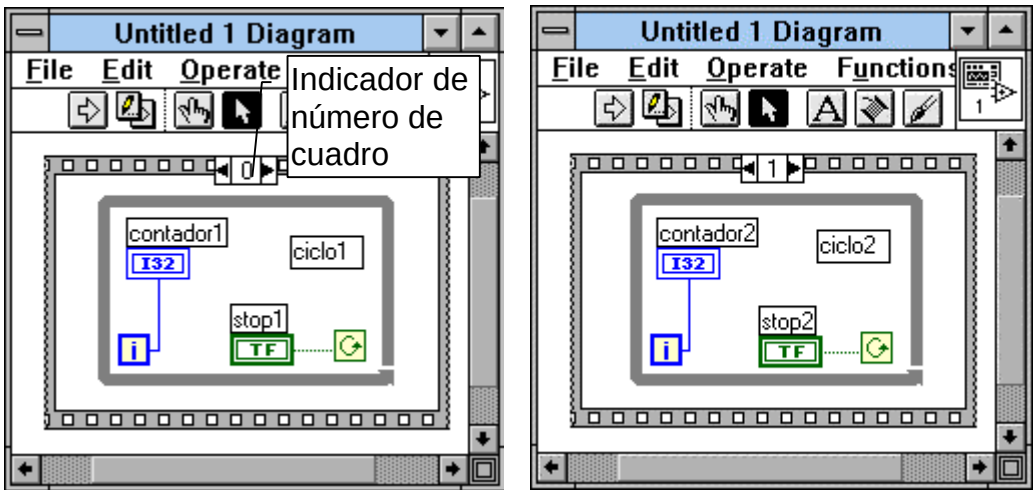
Para seleccionar el cuadro en el que se edita se usa el indicador en el extremo superior del marco.

Un truco posible para lograr que una función siga a la otra sin usar cuadros de secuencias, es usando cables que delimiten un flujo obligatorio.



En el ejemplo se aprecian dos ciclos los cuales al ejecutarlos lo hacen simultaneamente, así al presionar el botón de stop de cada uno, se detienen independientemente. Los ciclos simplemente cuentan números.

Si se desea que primero pase el ciclo uno y al presionar el estop de éste pase el segundo, se puede lograr tal efecto con los cuadros de secuencia así:



El mismo efecto se logra en algunos casos con el truco antes mencionado, como en la figura, pero no siempre es conveniente, por lo que es mejor usar la secuencia, además de que ésta reduce la extensión del diagrama.