GUIA DE LABVIEW

Programación general y uso en Control
Aplicación en Ingeniería.

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 computador, 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 computador de este tipo a un precio mínimo, y en decenso día a día.

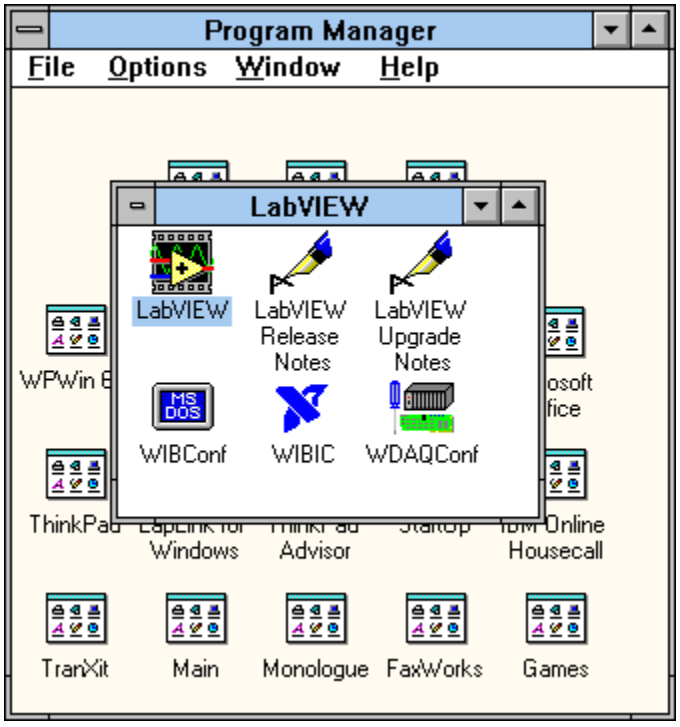
CARGANDO LABVIEW

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 computador.

USO DE LA MEMORIA:

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

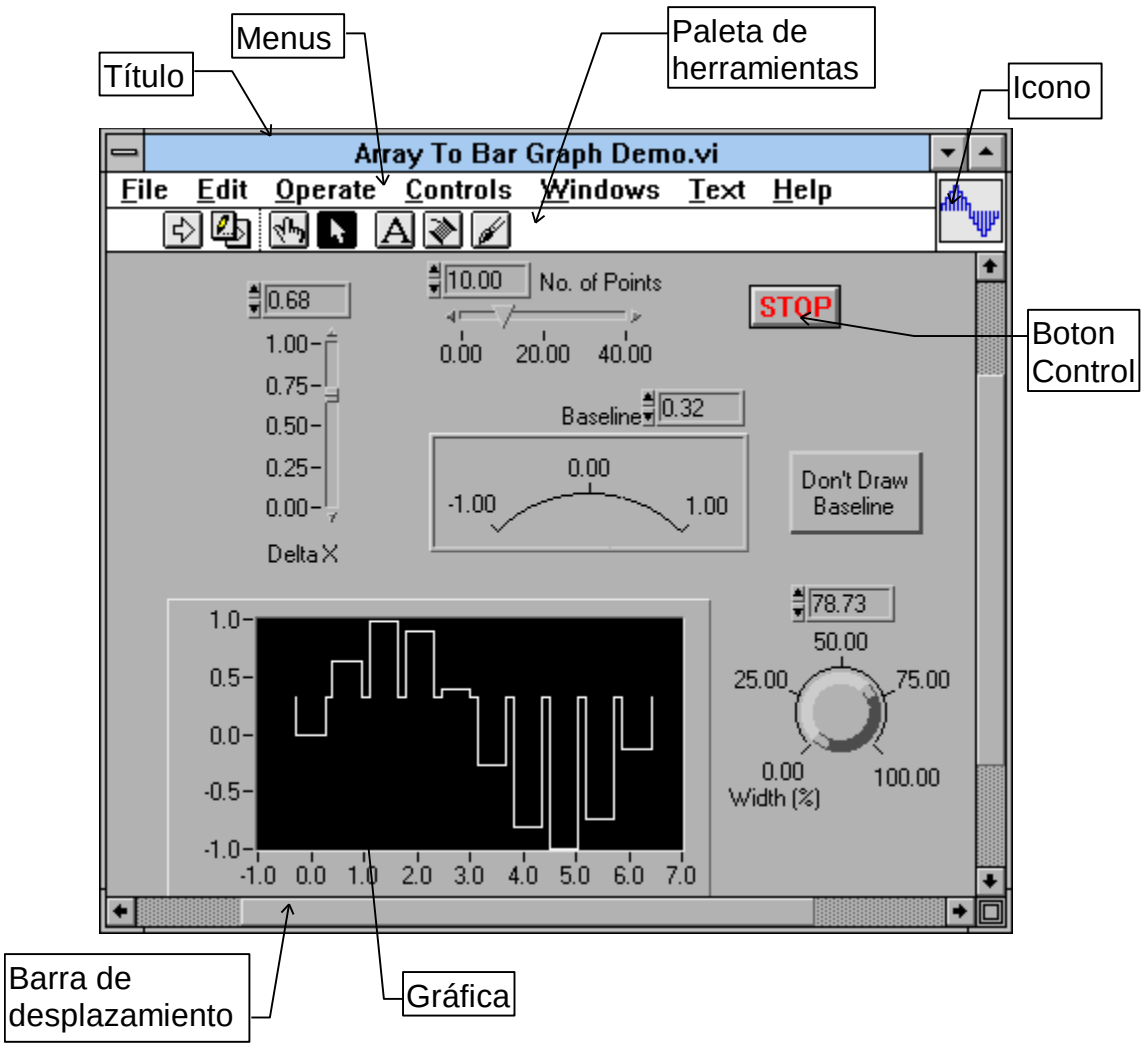
- EL PANEL FRONTAL: Donde se ven los datos y se manipulan y controlan.
- EL DIAGRAMA DE BLOQUES: En este se aprecia la estructura del programa, su función y algoritmo, de una forma gráfica en lenguaje G, donde los datos fluyen a través de líneas.
- EL PROGRAMA COMPILADO: Cuando se escribe en LabView, el algoritmo escrito de forma gráfica no es ejecutable por el computador, por tanto, LabView lo analiza, y elabora un código asembler, 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 computador 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.

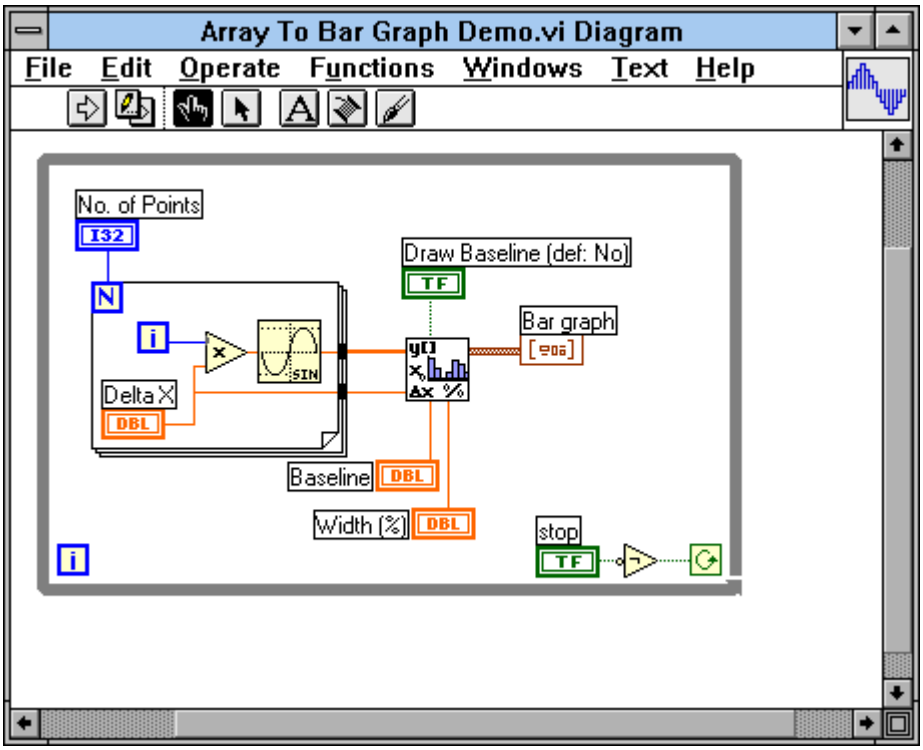
- El Panel frontal , donde estarán ubicados todos los indicadores y controles que el usuario podrá ver cuando el programa este en funcionamiento. Por ejemplo botones, perillas, gráficas,etc.



- El diagrama de bloques muestra el programa en código gráfico G, el cual es el objetivo de aprendizaje en un nivel básico, en este libro. 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.

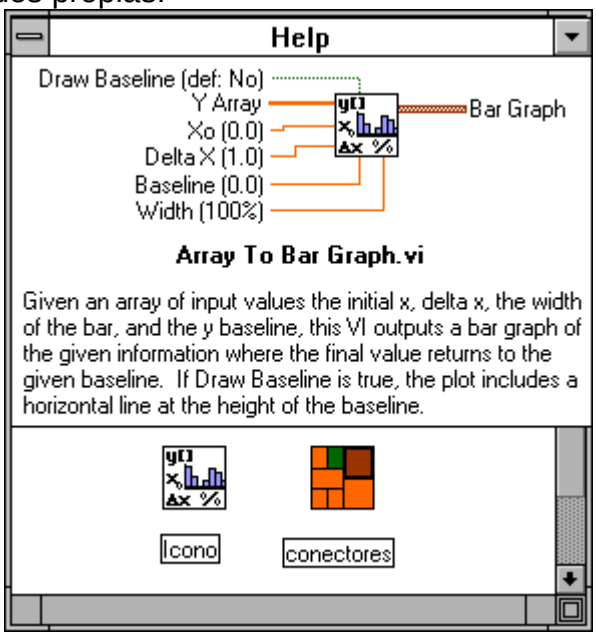
Todos los indicadores y controles ubicados en el panel frontal están respaldados por un terminal de conección en el diagrama de bloques tal como si se tubiera un tablero de control de una máquina o un avión, donde por el frente se ven los indicadores y por el lado posterior se aprecian todos los cables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



- El ícono de conección. Se usa para utilizar el programa creado como subrutina en otro programa, donde el ícono será la caja negra, y las entradas son las conecciones a los controles del programa subrutina, y las salidas son las conecciones 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 menu “Windows”

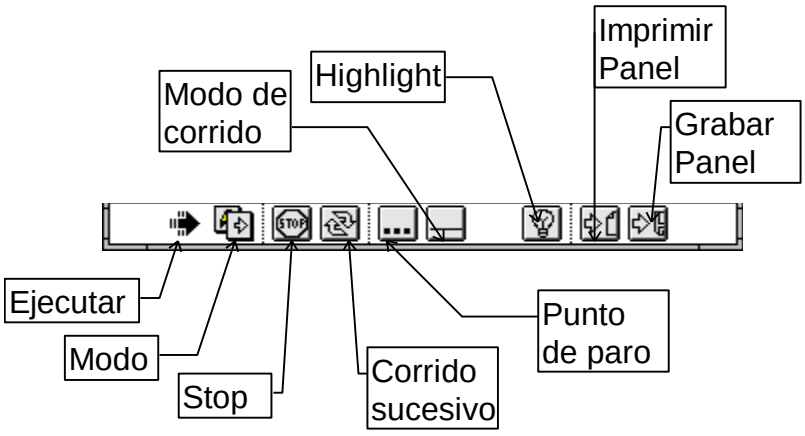
Actualmente existe una asociación de usuarios de LabView donde los miembros estan creando cajas negras de diferentes funciones, las cuales pueden ser usadas para utilidades propias.



PALETAS DE TRABAJO

Tanto en el panel frontal como en el diagrama de bloques, existe una paleta de erramientas, 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.

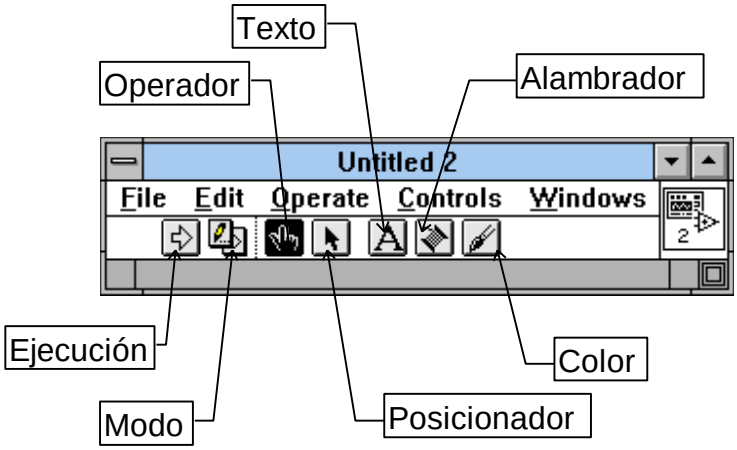


- Con el botón **"Ejecutar"** se corre una ves el programa. Cuando está ejecutando, se cambia a rayado como se aprecia en la figura y aparece un botón de **"Stop"** con el cual se pede detener el programa. No es recomendado hacer esto, es preferible crear un alagoritmo de paro del programa, con un botón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



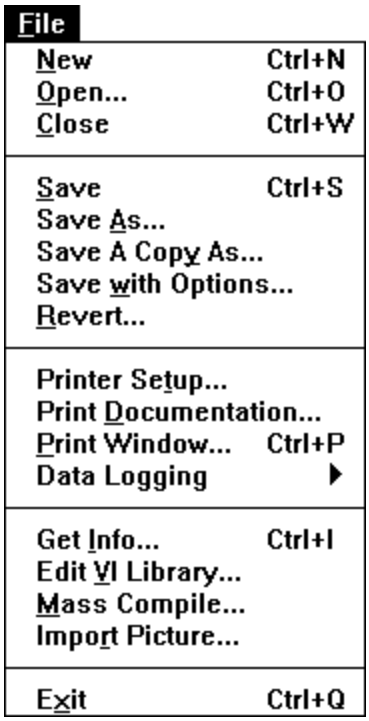
- **"Operador"** Sirve para accionar los controles e indicadores.
- **"Posicionador"** Sirve para cambiar de posición los diferentes elementos en las diferentes pantallas. También permite cambiar el tamaño de estos.

- **“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.

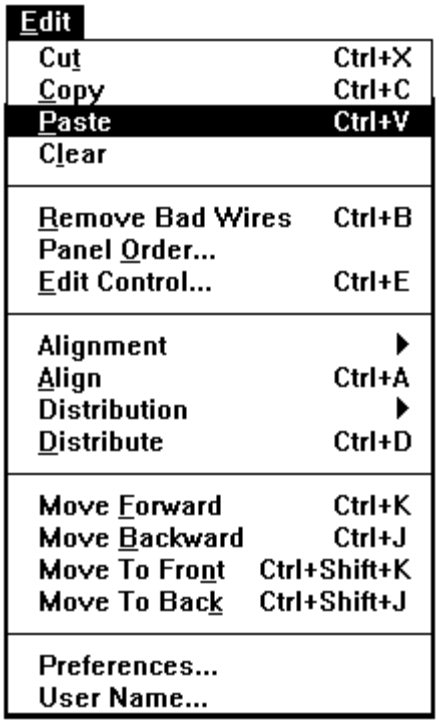
MENUS DE TRABAJO

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

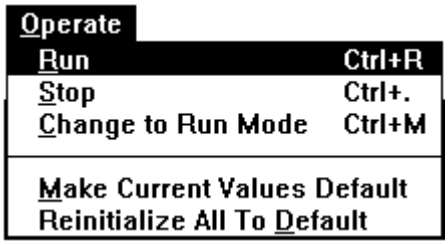
File. En este menú se encuentran las herramientas para el manejo de archivamiento, impresión, y guardado de información de los los programas creados en LabView.



En el menu **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.

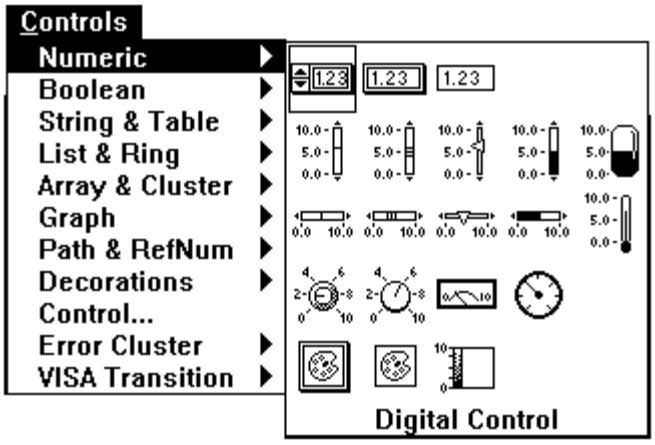


En el menu **Operate** se encuentran erramientas para ejecutar y detener los programas, asi como cambiar el modo de trabajo, y hacer que todos los valores en los controles e indicadores queden como valores iniciales al ser guardado el programa.

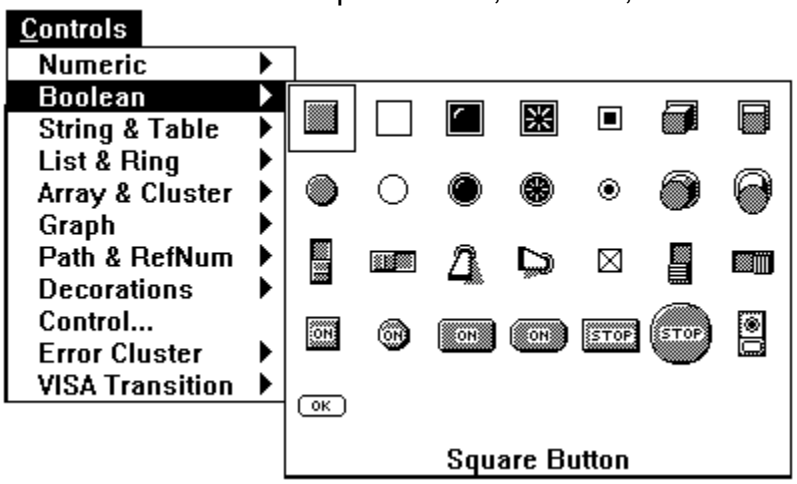


En el menu **Controls** aparecen todos los tipos de controles e indicadores que se pueden colocar en el panel frontal, como son:

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

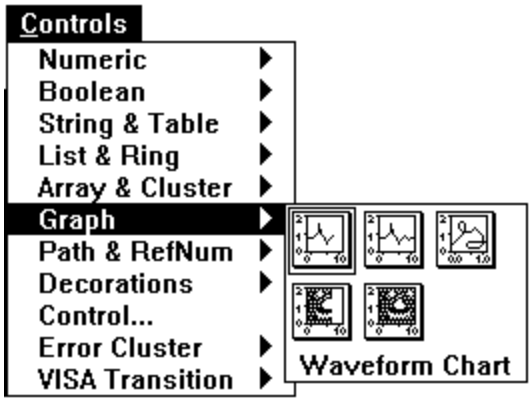


3. String & Table: permite entrar y sacar datos de tipo alfanumérico, vistos en un indicador o control, o en una tabla que tambien puede cumplir las dos funciones.

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 boleano. 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 computador, 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.

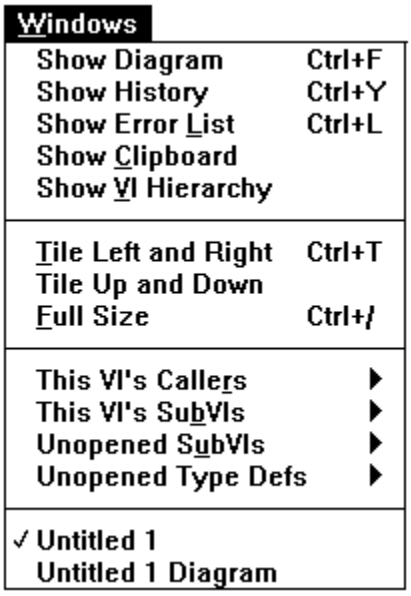
8. Decorations: Se disponen alementos de decorativos para el panel frontal.

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

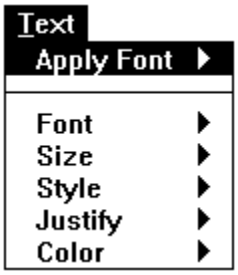
10. Error Cluster: Controles de entrada y salida, para parámetros de algoritmos manejadores de errores.

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

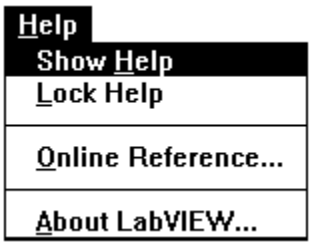
En el menú **Windows** se encuentran las herramientas para hacer cambios entre ventanas de trabajo.
Mostrar diagrama o panel, según la ventana en la que se encuentre.
Mostrar la historia de los cambios en el programa.
Visualizar la lista de los errores que posee el VI o programa.
Desplegar el contenido 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.



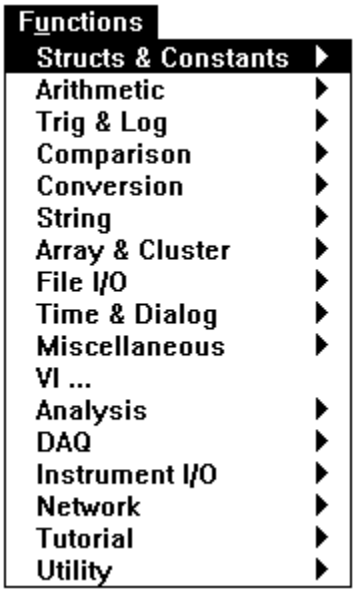
El menu **Text** se encuentran todas las utilidades para seleccionar tipos, colores, estilos y tamaños de letra.



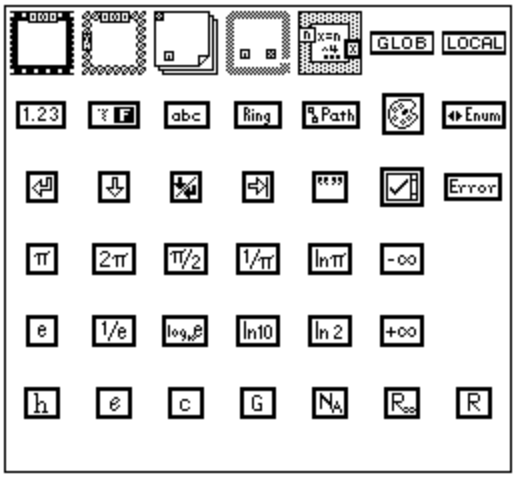
El menu **Help** presenta las ayudas necesarias sobre el programa, y ofrece la opcion para desplegar una ventana donde se explica cada objeto solo con señalarlo. En la ventana mencionada se explica como son las entradas y salidas de cada subVi, y de cada función.



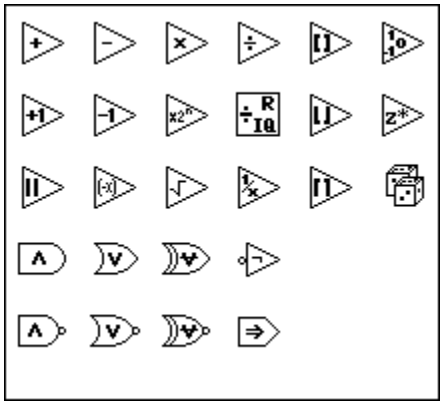
El menu de **Functions** ofrece todas las posibilidades de funciones que se pueden utilizar en el diagrama de bloques, donde al hacer click se escoje y ubica dentro del programa.



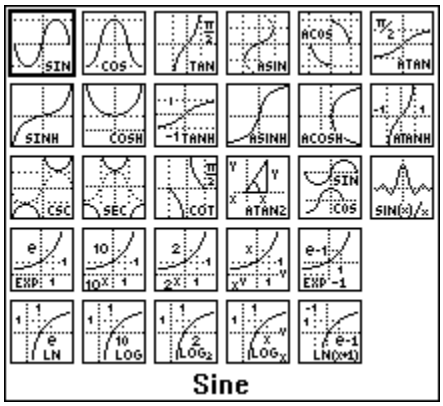
1. Structs & Constants: Contiene las estructuras básicas de programación como son las secuencias, los casos, los ciclos For-Next y Mientras, las variables de tipo global y local, y las constantes de todo tipo, como son las numéricas, las alfanuméricas, las boleanas, y algunos numeros especiales, “e“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



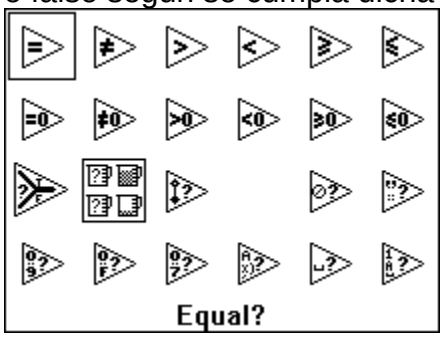
2. Arithmetic: Presenta las operaciones básicas aritméticas como son suma, resta, multiplicación, números al azar, valor absoluto, compuertas and, or, not y muchas otras. Para ver la función de cada una usar la ventana de Help <ctrl-H>.



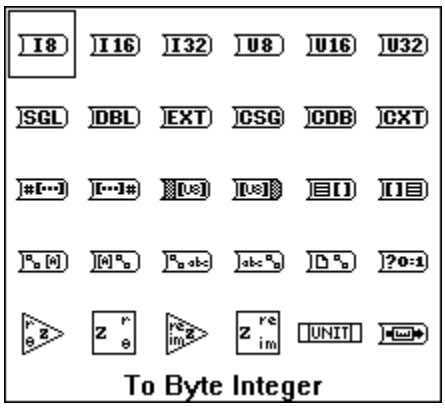
3 Trig & Log: presenta funciones trigonometricas y logaritmicas.



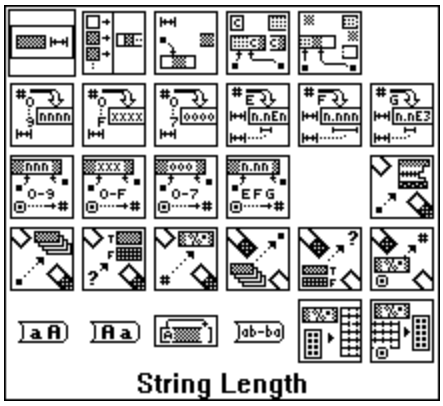
4 Comparison: Funciones de comparación que devuelven un valor de verdadero o falso según se cumpla dicha comparación.



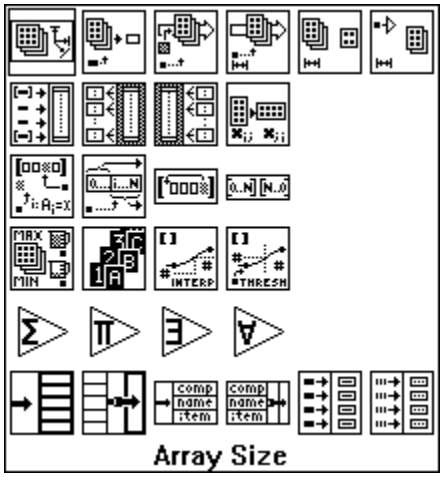
5. Conversion: Conversiones de tipos de variables, de un formato a otro, por ejemplo convertir un número a otro que ocupe 32 bits en memoria, o convertir un número a una matriz de boleanos cuya representación en binario corresponda al número.



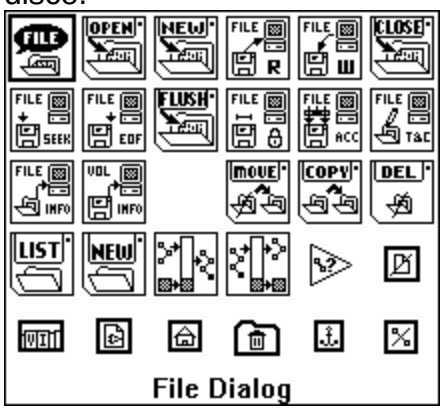
6. String: presenta herramientas para manipular cadenas de caracteres. Por ejemplo connvertir todos los caracteres a mayúsculas, o reportar el valor de la longitud de la cadena.



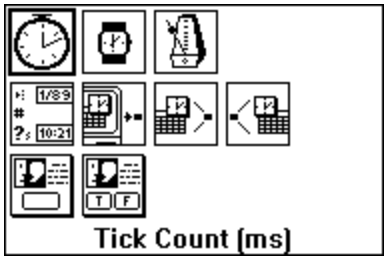
7. Array & Cluster: Maneja las herramientas para el uso de matrices y agrupaciones. Ej. dar las dimensiones de una matriz, en otra de una sola dimensión. Ej agrupar un conjunto de cables en uno solo par manipular menos líneas. El manejo de matrices y clusters será mejor esplicado adelante.



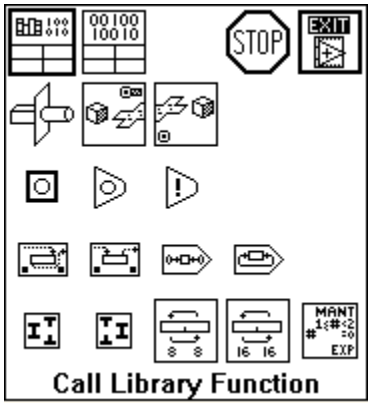
8. File I/O: Para el manejo de archivos y almacenamiento de información en disco.



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

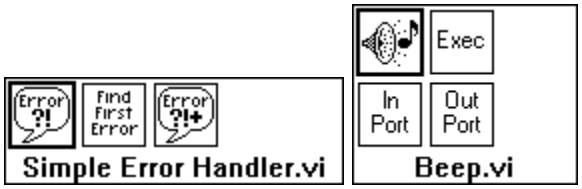


10 Miscellaneous: Bloques de llamada a codigos en C, o a librerías dinámicas de windows DLL. Conversión de datos a binario; manejadores de ocurrencias para ordenar el flujo de datos. Y otras funciones de uso mas avanzado.



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 computadores 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.



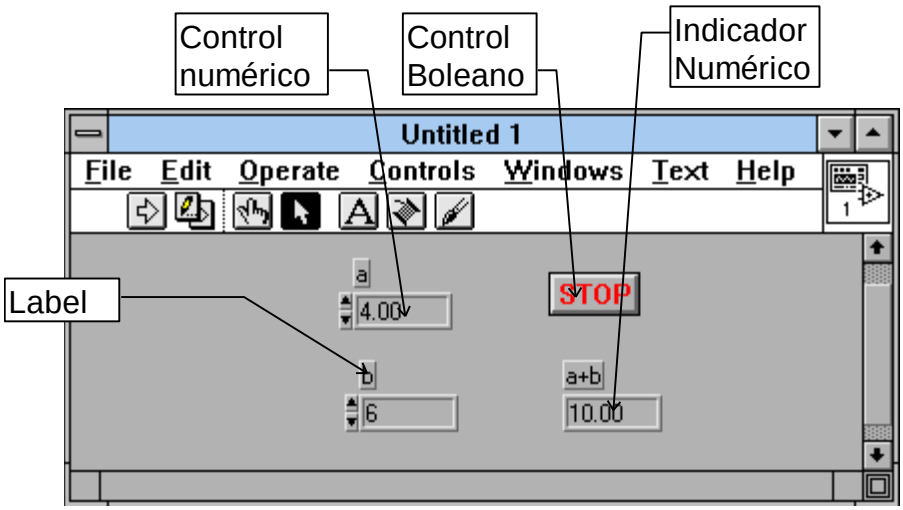
DDE (Dynamic Data Exchange)
PC (Personal Computer),
VI (Instrumento Virtual)**,**
LabVIEW (Laboratory Virtual Engineering workbench),
DAQ (Data Acquisition System),

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 menu 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-menu 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 menu en la subsección **show, label,** se puede hacer que reaparezca la marca para así escribir sobre ella.

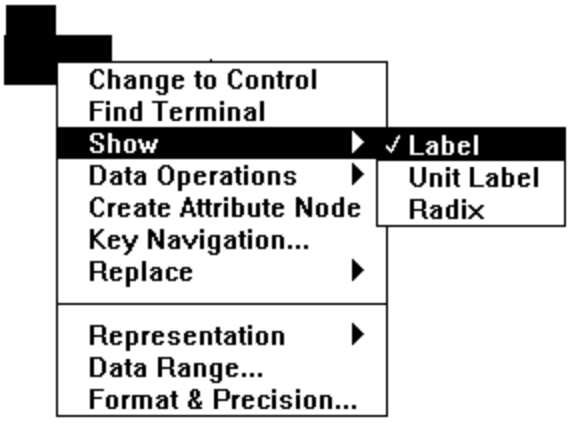
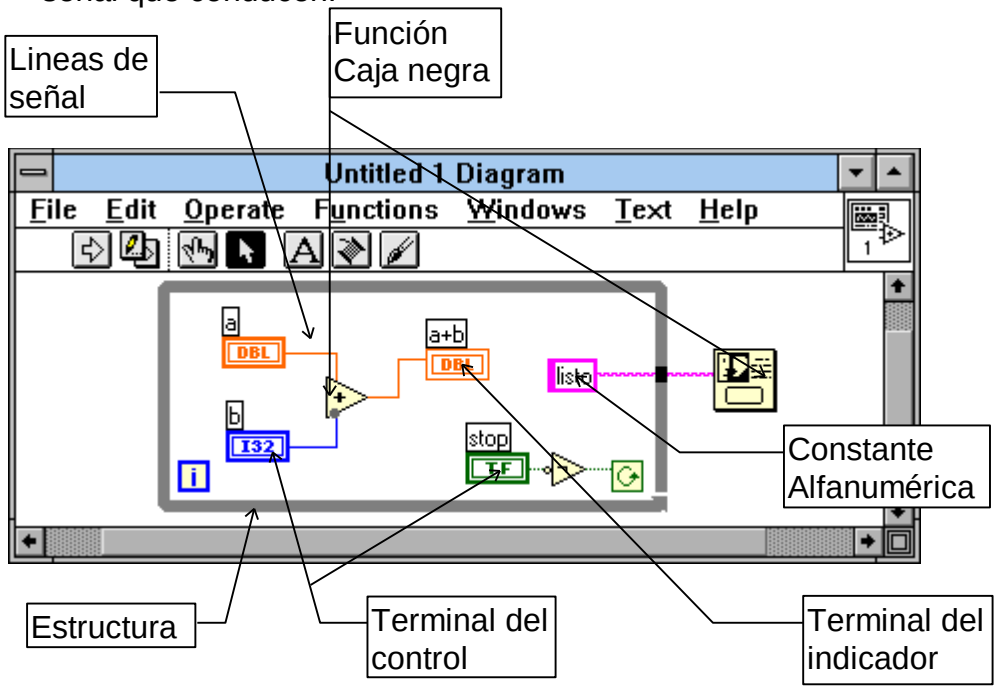


DIAGRAMA DE BLOQUES

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

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

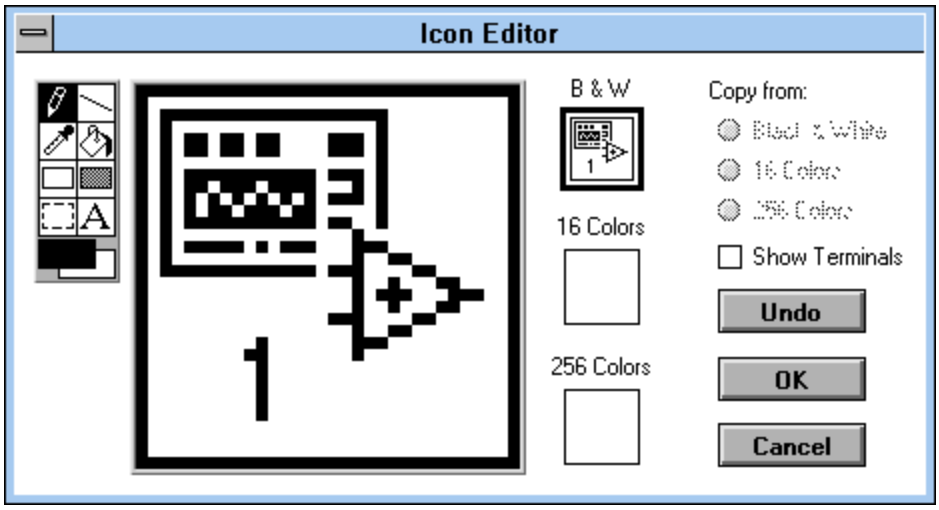
- Las constantes.
- Las funciones y cajas negras, donde se pocesan las señales.
- Las estructuras de programación.
- Los cables que conducen las diferentes señales, los cuales varían según la señal que conducen.



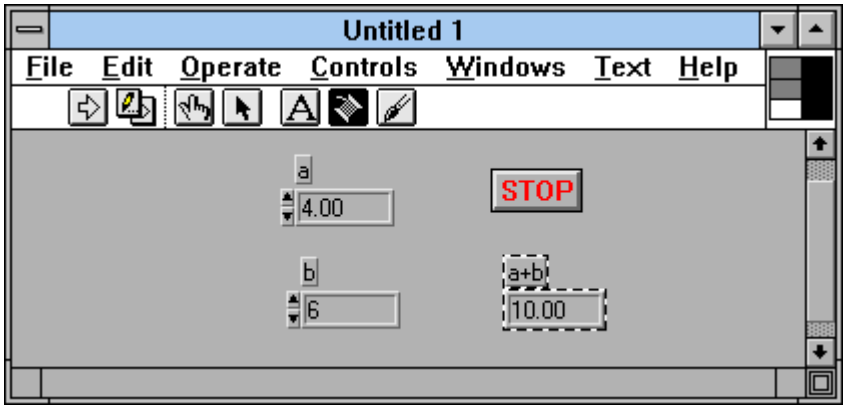
Para realizar el diagrama de bloques se buscan las estructuras necesarias en el menu de functions, estructuras y constantes, donde se encuentra el ciclo mientras (While), el cual será esplicado luego.
Posteriormente se ubican las funciones necesarias en el menu de Functions, como en este caso el sumador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

CONECCION DE ICONO

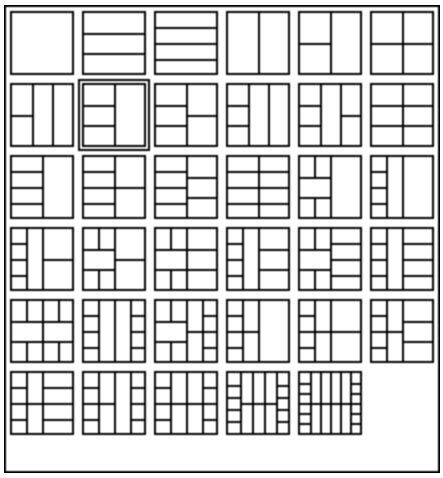
Si se desea que el programa realizado sirva como subrutina para otro VI de mayor jerarquía, como primero se debe realizar un Icono que represente el VI, y luego hacer las conecciones entre los terminales del ícono y los istrumentos del panel frontal. Cabe anotar que se conectan solo los deseados. Los 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 mouse en el icono del panel frontal y se selecciona **'Edit Icon'**. En este editor se puede dibujar el ícono deseado.

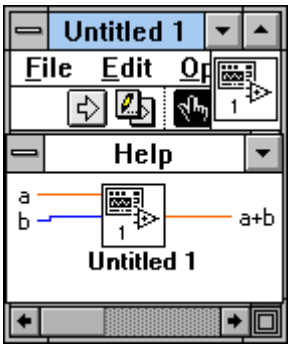


Después de tener el icono deseado se muestran los conectores por medio de **'show connector'** en el mismo pop-up menu y con la herramienta de alambrar se hacen las conecciones con los dispositivos del panel haciendo primero click en el indicador o control y luego en el pin del ícono deseado. Es recomendable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



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



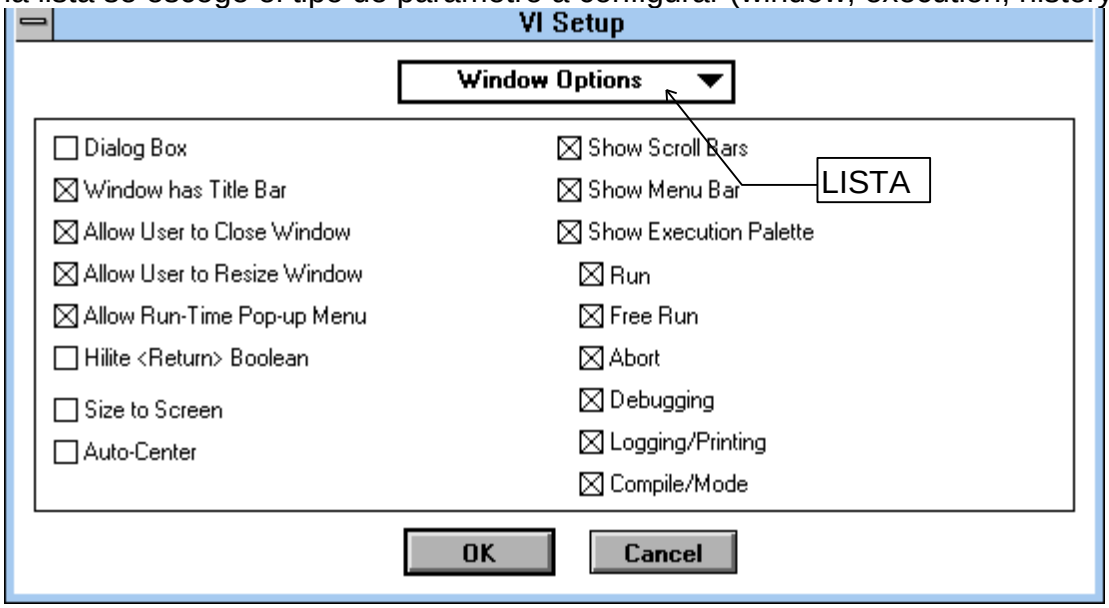


Luego de terminar el programa solo es salvarlo.

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).



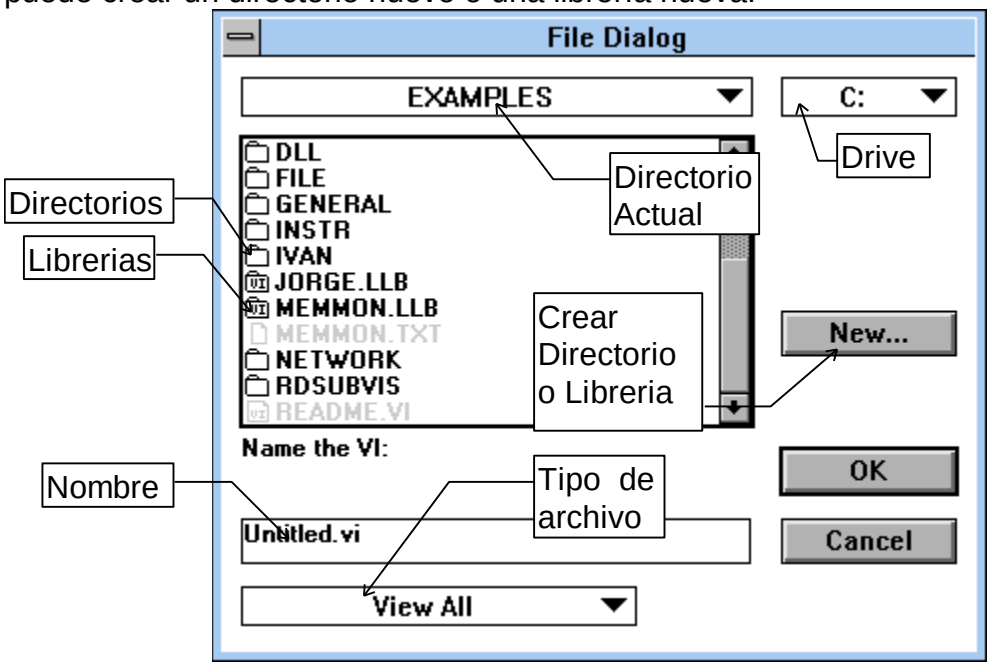
DIRECTORIOS DE ALMACENAMIENTO

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 menu 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

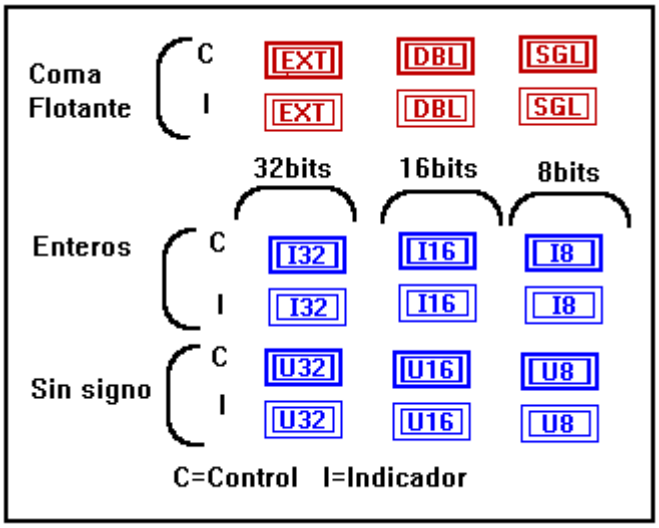
NUMERO DE BITS EN UN NUMERO
El computador posee una memoria compuesta de una gran lista de números, los cuales son llamados bytes, que son un conjunto de unos o ceros, llamados bits.

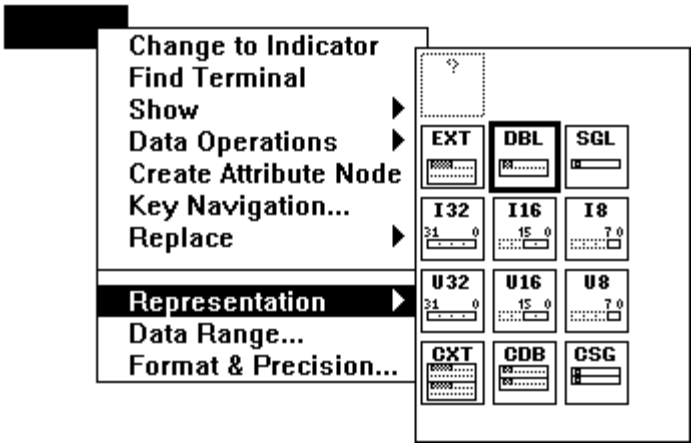
Cada byte se compone de ocho bits los cuales pueden representar un número de 0 a 255. Para poder almacenar números mayore se requere de más bytes, donde se tengan 16 o 32 bits. Este número se relaciona con el número de bits con los que puede trabajar el microprocesador del computador, en cuanto a la velocidad de operación. Además un número de más bits ocupa mayor espacio en memoria.
Tomando en cuenta esto, si se desea manejar el 258, y se usa un byte (8 bits), el número obtenido es 255, perdiendo toda la exactitud. Se deben usar 16bits.

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.

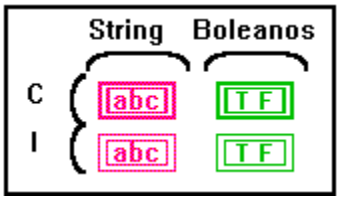
Según lo anterior hay números de tipo entero 'I' de 8, 16 y 32 bits, de tipo sin signo (unsigned U) de 8, 16, 32, o de coma flotante de tipo simple (SGL 16), doble (DBL 32), y Extendido (EXT 64 bits). Igualmene números complejos simples, dobles y extendidos.
El tipo de número se aprecia en la terminal de conección de los controles o indicadores, pues aparece inscrito, y el color de las conecciones 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 menu de cada control por la opción **representation**, igualmente con las constantes.





DATOS BOLEANOS Y ALFANUMERICOS

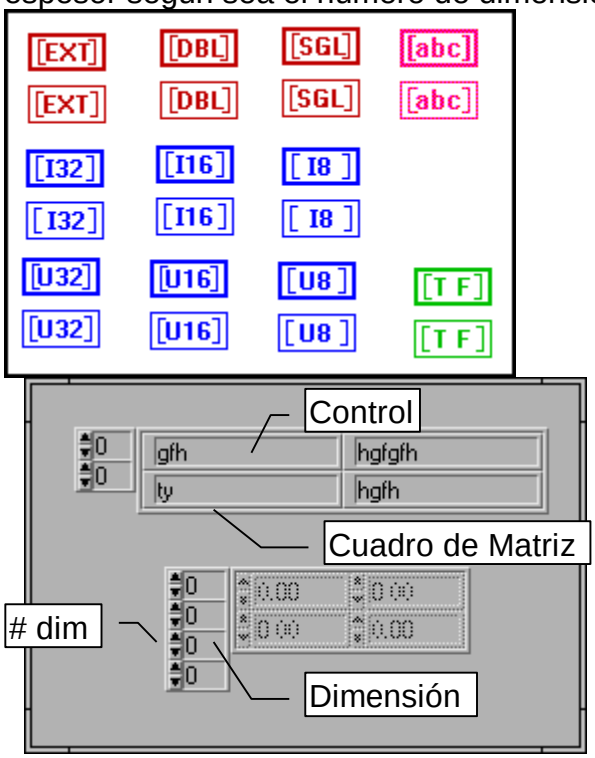
Los datos boleanos también tienen su tipo de conector. Para boleanos El color de las conecciones y los cables es de color verde, y para las de tipo alfanumérico son de color rosado.



MATRICES

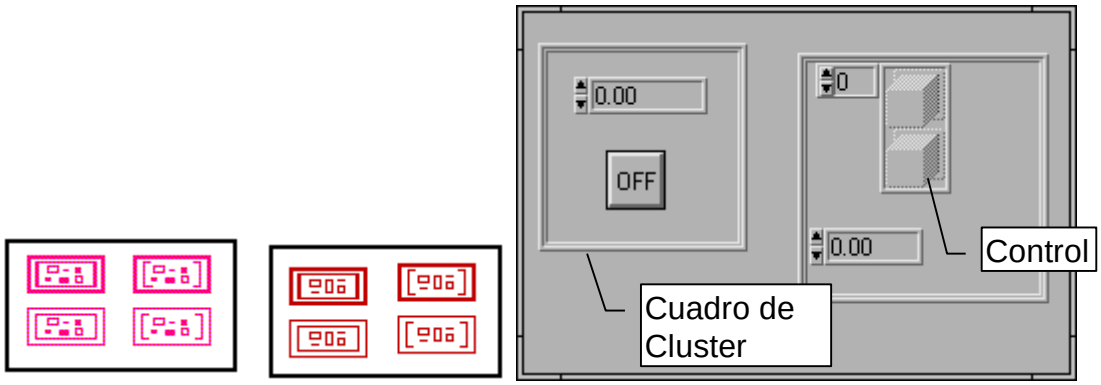
Las matrices son conjuntos de datos de una misma especie. Para crear una matriz se ubica en el panel frontal un cuadro de matriz (Array o arreglo) sacado del menú ARRAY & CLUSTER, y dentro se ubica el control o indicador que se mostrará. Se puede estirar el cuadro para que muestre varios datos pertenecientes 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 recuadro, así se puede poseer una matriz de cualquier clase de número, sea doble, alfanumérico, boleano, etc.
Las líneas o cables que conducen matrices son más gruesos y aumentan de espesor según sea el número de dimensiones que manejen.



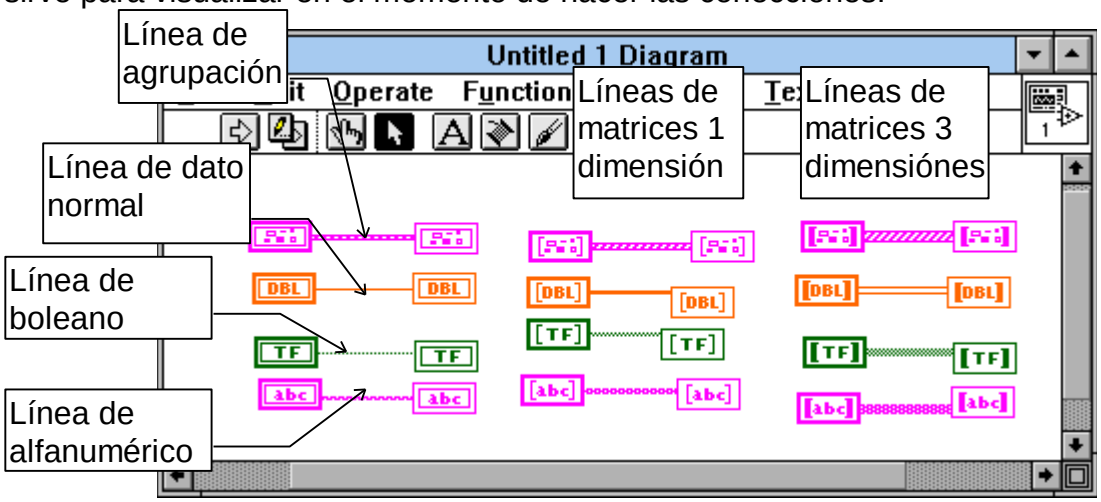
AGRUPACIONES O ESTRUCTURAS

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 poseer una agrupación con cualquier clase de números, sean dobles, alfanuméricos, boleanos, 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 un solo toma.
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 conecciones.



POLIMORFISMO

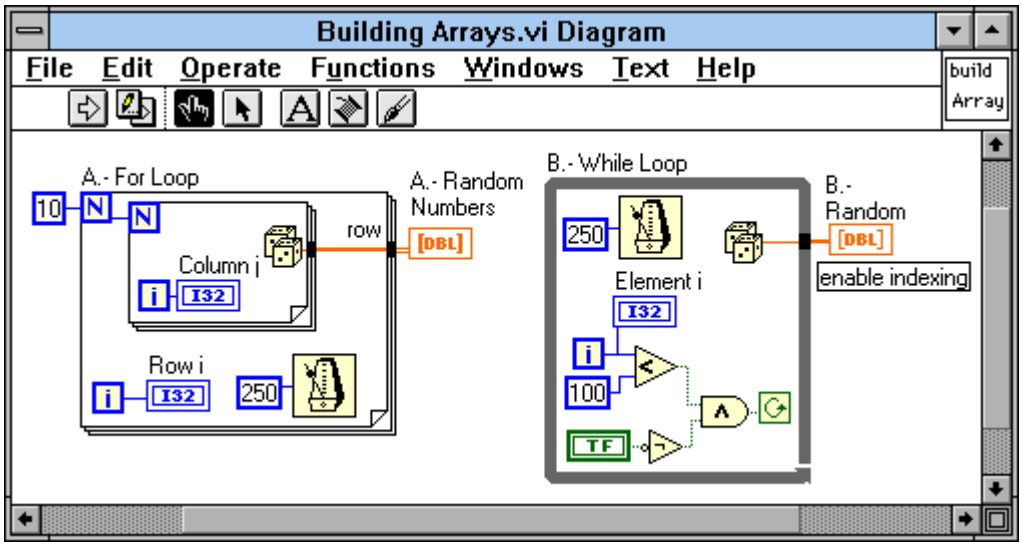
Como se ha mencionado existen números con diferente formato de representación, y según esto al sumar u operar con dos números de diferente clase no es correcto. Si se trata de sumar un número unsignado de 16 y uno de 8 bits, no se tendrá un resultado correcto. Como en el lenguaje C, para hacer este tipo 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 menu de funciones de conversión.

Sin embargo LabView permite parra 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 conección, 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.

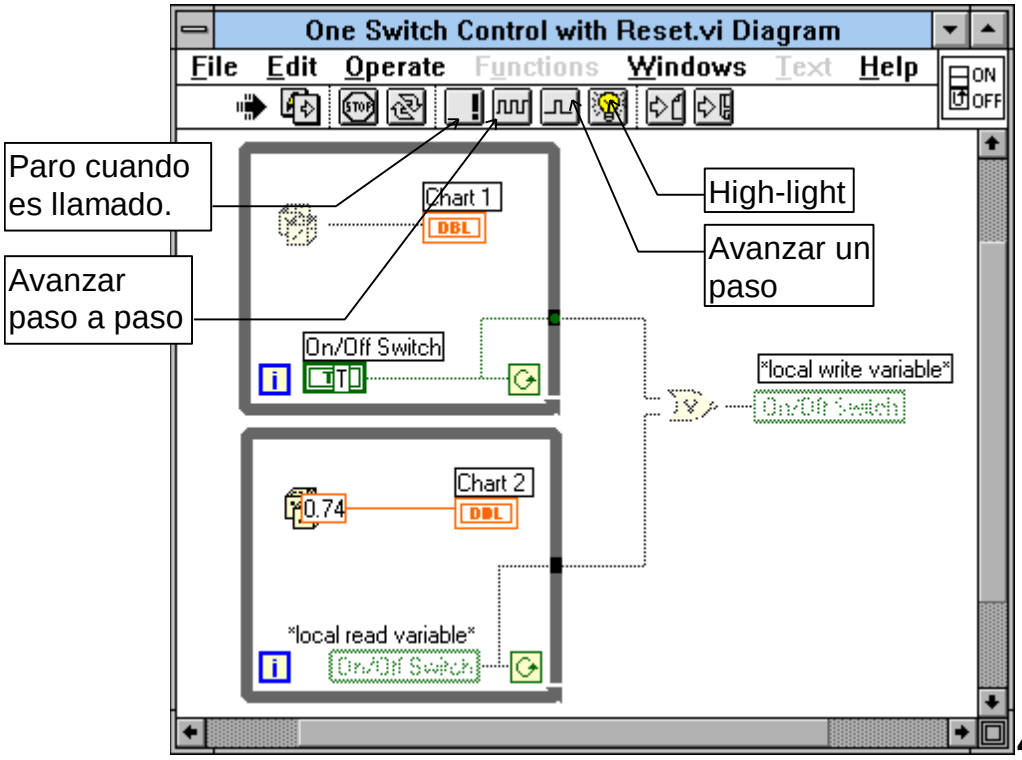
Los datos de salida solo surjen cuando ha cumplido la finción su operación, así mismo ocurre con las estructuras. Osea que el dato de salida de la estructura fluirá al resto del programa cuando esta halla concluido, para el caso de la figura, cuando el ciclo haya cumplido todo su número de vueltas.

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 conección 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 de conocer el manejo de las estructuras que gobiernan un algoritmo. En el LabView como lenguaje tambien cuenta 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.

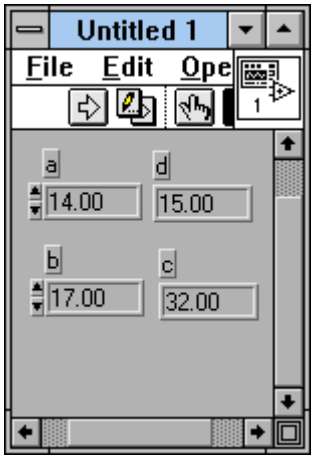
LOS CICLOS WHILE

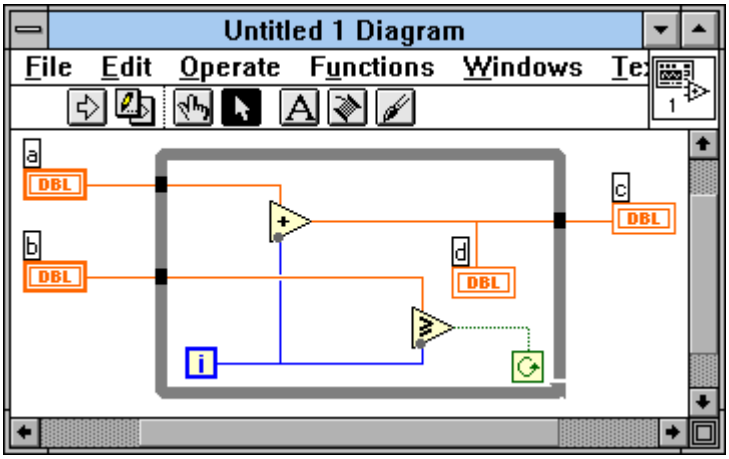
GENERAL:

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 recibir 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.

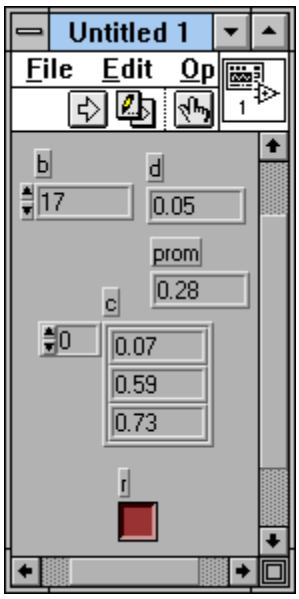


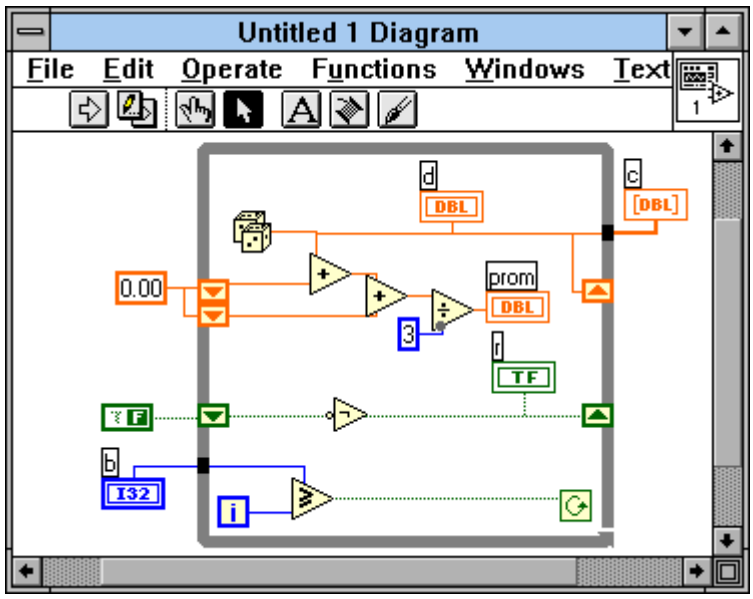


Los datos a, y b solo llegan una ves a frontera de entrada y alli quedan almacenados en un buffer para ser usados todas las veces que el ciclo repita. Estos datos siempre serán iguales.

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 valór pasa a 'c'.

Constantemente se evalua 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.





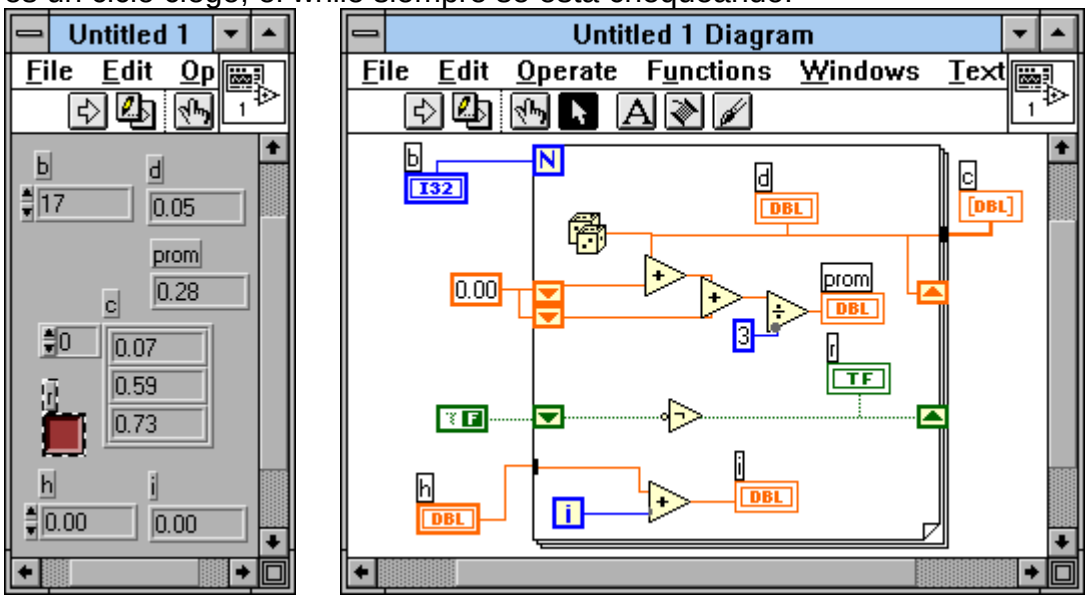
INDEXING:
Los ciclos se pueden utilizar para crear matrices simplemente acomulando los datos en la frontera de salida, sin permitir que el último borre el primero, y más bien apilandolos uno tras otro en matriz. Esto se logra sacando el pop-up menú de el punto negro de la frontera de salida, el cual 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. Despúes 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 boleano 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 bombillo 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 corre.
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.

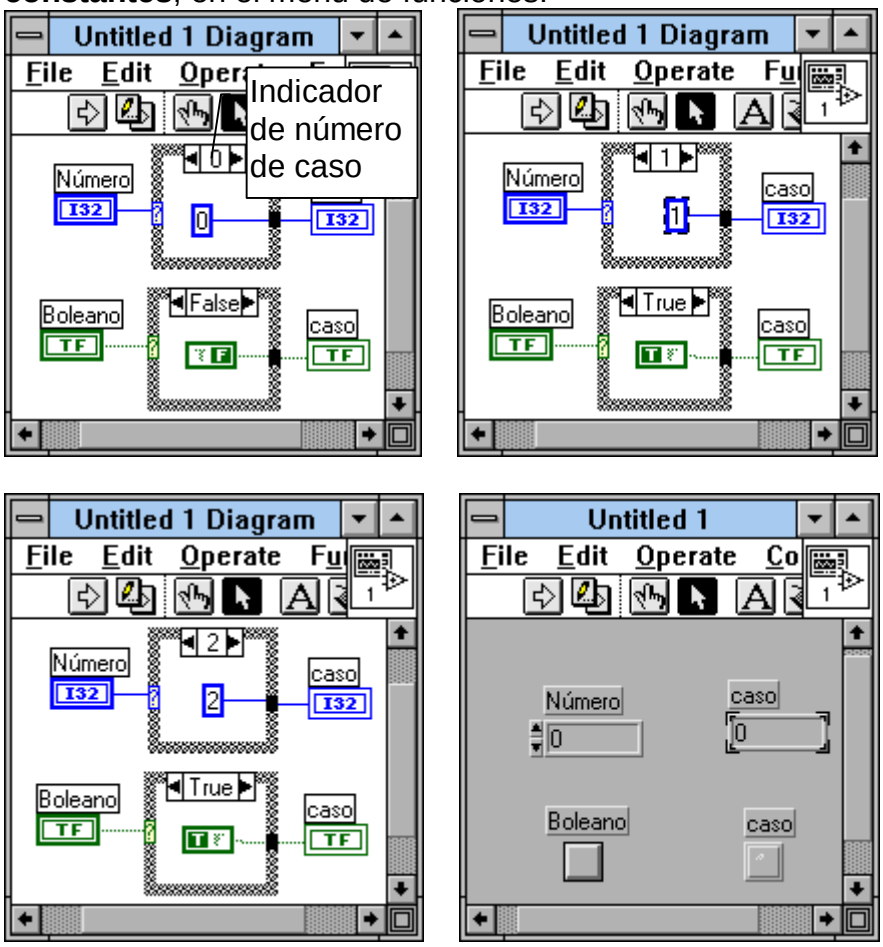


El ciclo For-Next- también cunta con un elemento ‘i’ que sirve de contador para decir en que ciclo va.
El programa anteriormente realizado con un ciclo While es equivalente al mostrado en la figura. A éste se le ha agregado un contador que suma un valor inicial ‘h’ con el contador, para mostrar en el 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 boleano 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.

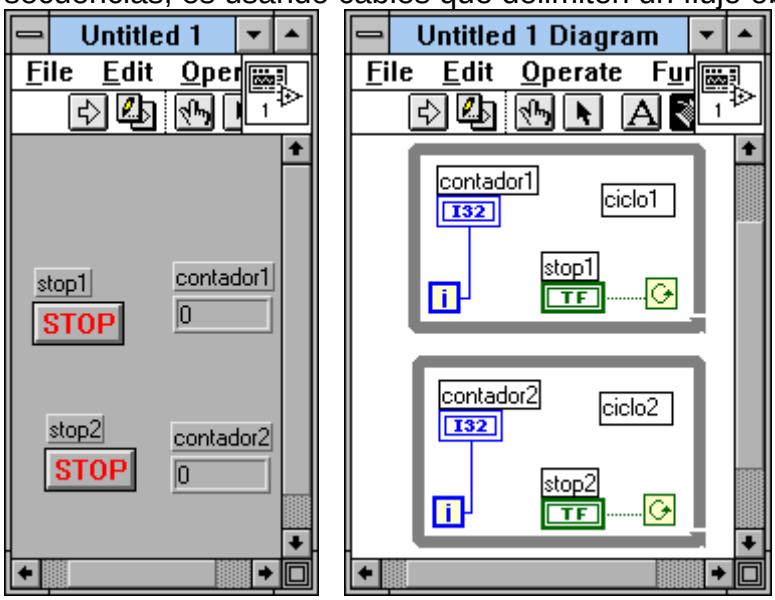
LAS SECUENCIAS

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.

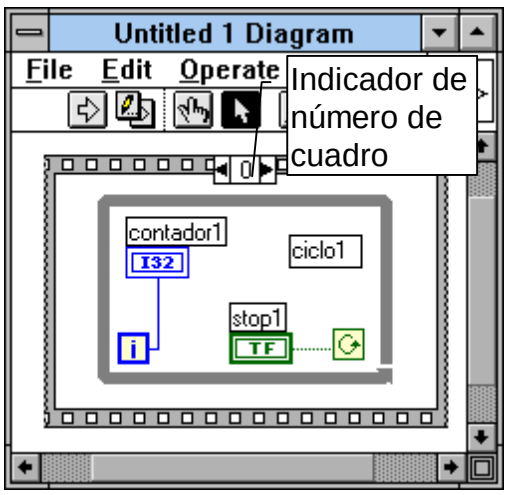
Para seleccionar el cuadro en el que se edita se usa el indicador en el extremo superior del marco.

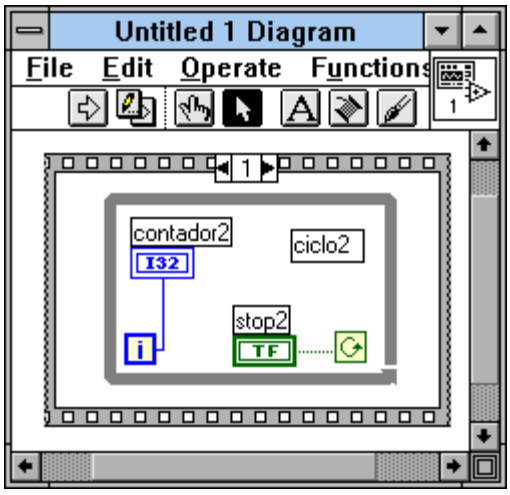
Un truco posible para lograr que una función siga a la otra sin usar cuadros de secuencias, es usando cables que delimiten un flujo obligatorio.



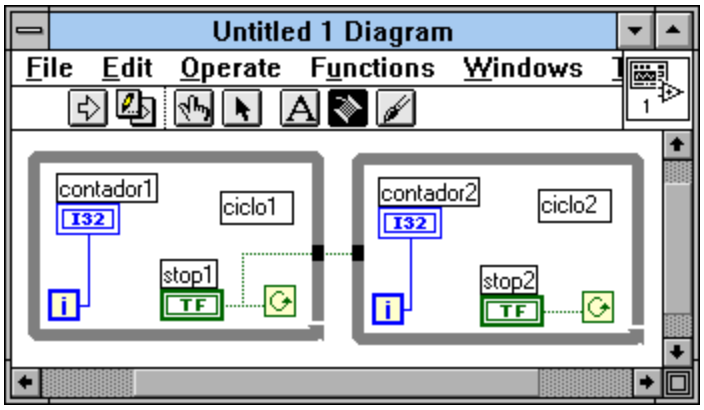
En el ejemplo se aprecian dos ciclos los cuales al ejecutarlos lo hacen simultaneamente, así al presionar el botón de estop de cada uno, se detienen independientemente. Los ciclos simplemente cuentan números.

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 comveniente, por lo que es mejor usar la secuencia, además de que ésta reduce la extensión del diagrama.



VARIABLES

Los parametros de entrada para una función pueden lograrse a través de controles, indicadores, e incluso variables. Las salidas pueden usarse para mostrarse en indicadores o simplemente para ser almacenadas en los mismos elementos los cuales son las variables. Estas se usan para almacenar datos y pueden ser de dos tipos según su uso, de tipo Local o de tipo Global.

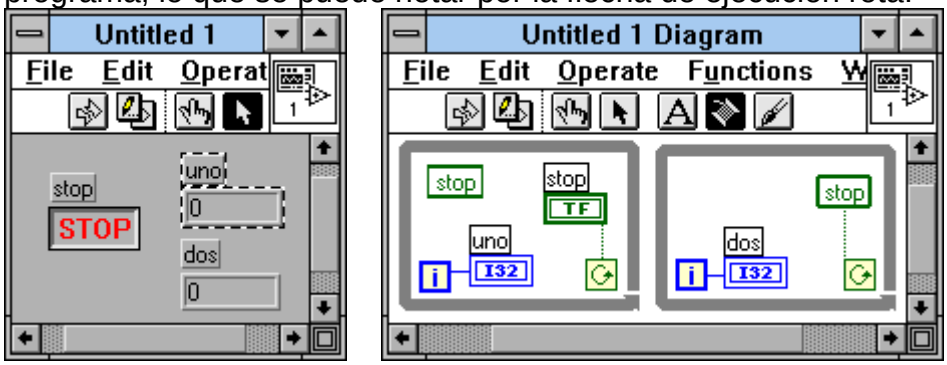
LOCALES

Son variables asociadas a algún control o indicador dentro de un programa VI, en el cual son usadas. Cuando se escribe en una de éstas, elcontenido del indicador o control cambia.

El uso de estas variables facilita la visualización en el digrama cuando se va a accesar varias veces un mismo dato, puesto que evita llenar de cables conductores la pantalla, que conduscan el valor desde el control al lugar requerido.

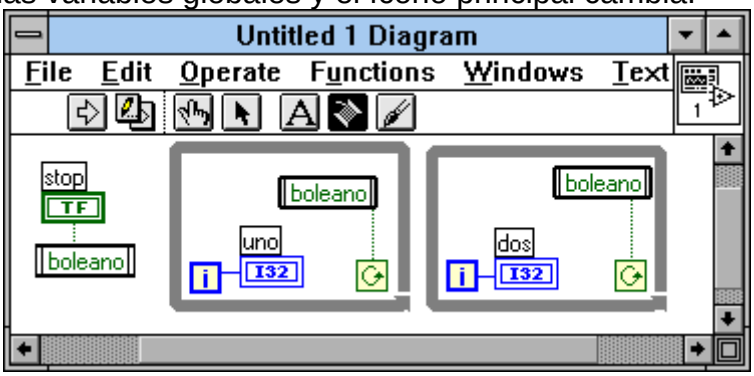
Son una forma adecuada de escribir sobre un control, desde el algoritmo.

Las variables locales solo son entendidas por el programa VI que las posee, ninguna subrutina entiende el contenido de éstas, ni ningún programa VI diferente.
Para obtener una de estas variables, se busca por el menú de estructuras y constantes, LOCAL.
Para seleccionar el indicador o control que se accesa, seleccionar por medio del pop-up menú de la variable, SELECT ITEM.
Si se desea leer un valor de la variable seleccionar en el pop-up menu CHANGE TO READ. El ícono de la variable mostrará las paredes laterales mas gruesas que en el caso de la variable a la que se escribe. Para escribir seleccionar en el pop-up menú CHANGE TO WRITE.
En el ejemplo se aprecia como para no tener que llevar cables desde el terminal del botón de stop al otro ciclo, para detenerlo simultáneamente, simplemente se lee el dato de una variable asociada. En el primer cuadro se muestra un ícono de una variable para ser escrita con un dato, pero como no está conectado a ninguna fuente de valores, se genera un error que no permite ejecutar el programa, lo que se puede notar por la flecha de ejecución rota.



GLOBALES
La diferencia con las variables locales radica en que estas pueden ser entendidas por cualquier programa y/o subrutina VI, y pueden ser actualizadas por los mismos. Estas se almacenan en un archivo diferente de extensión **.GLB** que consta únicamente de un panel frontál donde se encuentran todas las variables asociadas a dicho archivo, lo que quiere decir que en un archivo se pueden guardar numerosas variables.
Para crear una vaiable global, se selecciona del munu de estructuras y constantes. Luego con el pop-up menú se da la orden de abrir el panel frontal de esta variable, y allí se colocan todos los indicadores y controles que almacenan los datos deseados. Posteriormente se graba como cualquier otro programa VI, pero con extensión .DBL.
Con la variable creada solo es seleccionar en el pop-up menú del ícono de la variable con SELECT ITEM, el valor al que se lee o escribe.
Para colocar en el diagrama otra variable global de el mismo dato, ahora se hace por medio de la opcion VI, en el menú de funciones, tal como si se fuera a usar una subrutina ya creada.

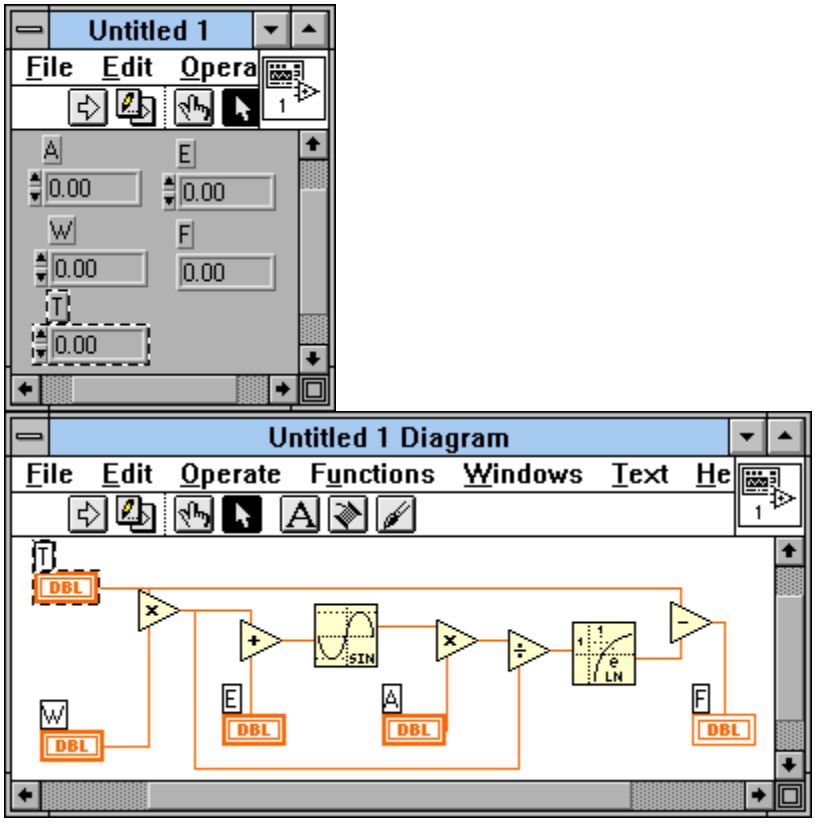
Se ve como la paleta de las variables globales y el ícono principal cambia.



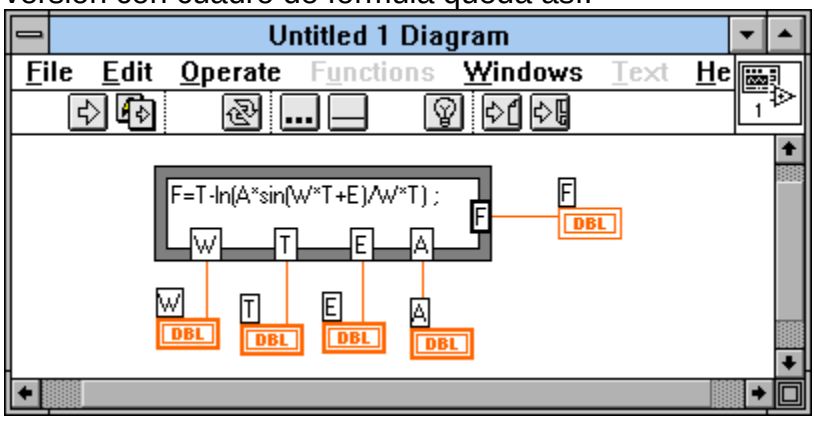
En el ejemplo el dato del botón stop se escribe a una variable global, que se lee en otros lados del programa. También puede ser leido en otro subprograma.
Los cambios entre lectura y escritura, son idénticos al procedimiento con las variables locales. Igualmente se aprecia que el ícono en estado de lectura es diferente al de escritura.

CUADROS DE FÓRMULA

Cuando se realizan operaciones matemáticas complejas donde hay muchos cálculos distintos, tener un ícono por cada operación puede ser complejo e implicar muchos cables y conecciones, que dificultan el entendimiento del diagrama y alargan el tiempo de programación.
Un mejor camino es tomar todas esas funciones y juntarlas en un cuadro de fórmula, donde se escribe la operación de una forma textual. Al cuadro se le agregan unos conectores de entrada y salida de datos, con el nombre de los parámetros inscritos, y de alli se alambran los conductores. Para agregar entradas o salidas hacerlo por medio del pop-up menú **Add Input** o **Add Output**.
Para escribir y correjir usar la herramienta de texto.
Como regla despues de cada función se debe colocar un punto y coma “;”.
El cuadro de fórmula se encuentra en el menú de funciones, estructuras y constantes.



En el ejemplo se aprecia una fórmula complicada con muchos alambres, la versión con cuadro de fórmula queda así:



5. FUNCIONES, MATRICES, STRINGS, FILE, I/O

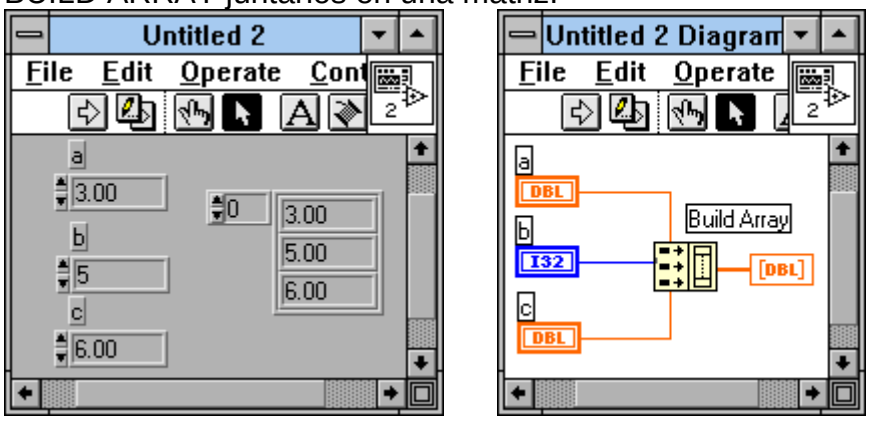
En éste capitulo se tratan algunos tipos de funciones especiales como son las usadas para manejar Matrices (Arrays), Strings o cadenas de caracteres, Archivos, y formas de entrada y salida como es el sistema DAQ.

MATRICES

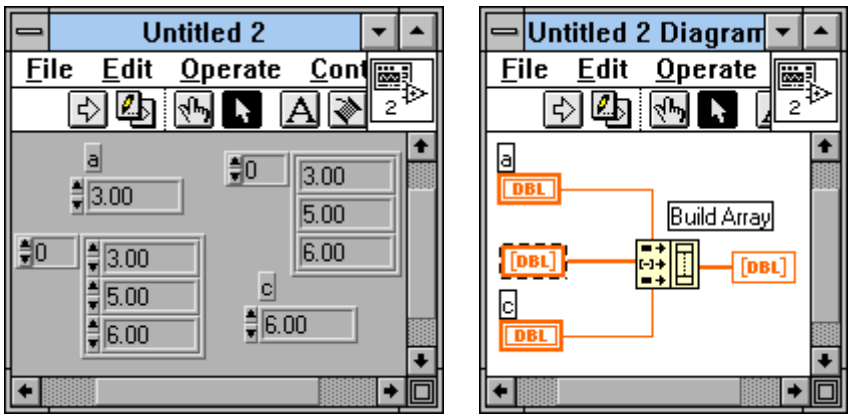
Las matrices son conjuntos de datos los cuales se almacenan uno detrás de otro, en fila o columna, si se habla de una dimensión, y en cuadros para matrices

bidimensionales. Las matrices o arreglos pueden tener tantas dimenciones como se desee.
El manejo de cálculos y operaciones con matrices es tema de materias como Algebra Lineal, por lo cual no se esplica a fondo su comportamiento.
Las matrices son la base de los grandes cálculos realizados por el computador, quien se puede decir que posee su memoria en forma de matriz, donde cada valor está ubicado en un lugar de la memoria.
Como se ha mencionado se pueden usar ciclos para formar matrices de datos, e incluso ciclos anidados para formar matrices de varias dimensiones.
Se recomienda mirar los ejemplos de manejo de matrices, arrays, en el directorio de EXAMPLES, GENERAL ARRAYS.LLB, en la carpeta de Archivo del LABVIEW, y cargar los VI que son de ayuda para comprender más a fondo.

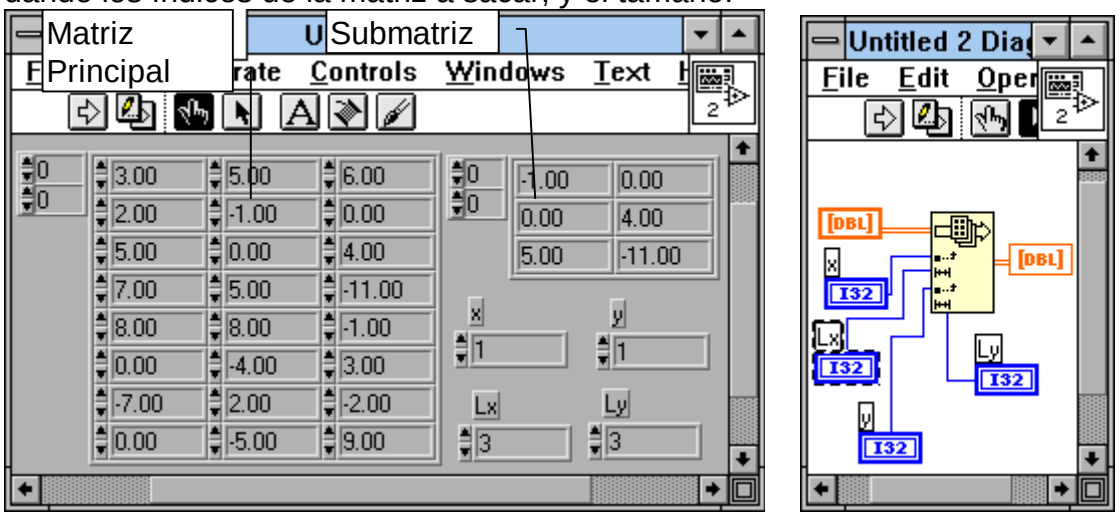
Como se ha visto los cables que conducen datos de matrices son más gruesos.
Hay funciones para multiplicar matrices por una constante, sacar el valor determinante, sacar una submatriz que contenga parte de los elementos, averiguar un elemento de la matriz, reemplazarlo, o agregar uno nuevo.
Para el LabView una matriz puede ser de dos dimensiones así una de las dimensiones sea de un elemento, osea que un vector se puede tomar como matriz bidimencional, siempre y cuando se especifique así.
Es perfectamente factible tomar varios elementos y por medio de una función BUILD ARRAY juntarlos en una matriz.



Todos los elementos de la matriz deben ser del mismo tipo, de no ser así como en el ejemplo, la matriz formada si los tendrá, produciendose un cambio polimórfico que puede ser traer consecuencias de impresición.
Si se desea unir un elemento a una matriz, se puede usar la misma función Build Array para encadenarlos,esto se logra entrando la matriz inicial como arreglo y no como elemento. Para esto seleccionar el pop-up menú de la entrada de Buid Array, y seleccionar **Change to Array**. Se nota como la ebtrada cambia en su configuración para aceptar una matriz.



Si se da entrada a dos o más matrices de una dimensión, del mismo tamaño, como elementos, se crea una matriz de dos dimensiones, como apilando capas de filas.
Si se da entrada a dos o más matrices de dos dimensión, del mismo tamaño, como elementos, se crea una matriz de tres dimensiones, como apilando capas de planos. Así sucesivamente.
Para agregar una dimensión al control solo basta cambiar el tamaño de los indicadores de índice de la matriz, con la herramienta de posicionamiento.
Para sacar una submatriz de otra mayor, se puede usar la función Array Subset, dando los índices de la matriz a sacar, y el tamaño.

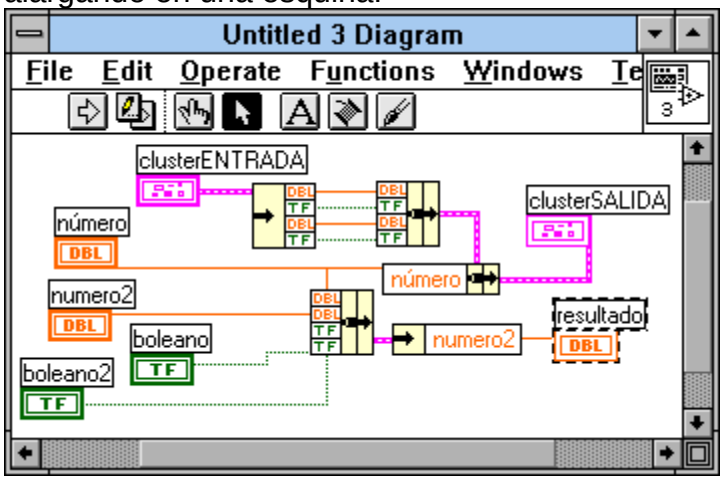


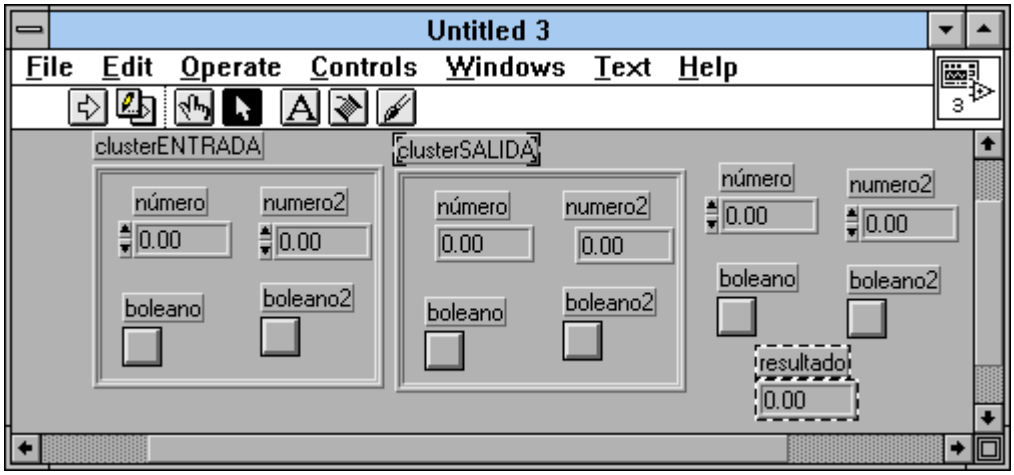
Las funciones para operar con matrices se encuentran dentro de los menus de funciones, Array & Cluster, y Analysis.
Para crear un control de Matriz, o un indicador, solo hace falta seleccionarlo en el panel frontal del menú Controls, Array & Cluster, ARRAY, y ubicarlo. Luego colocar dentro el tipo de control o indicador deseado, sea numérico, o boleano.

CLUSTERS

Un Cluster es una agrupación de datos, de diferente tipo, como ya se había mencionado, donde todos los cables se juntan en uno solo para facilitar el

cableado en el diagrama de bloques. Para juntar los datos se usa la función Boundle, y para separarlos la función Unbundle.
Igual que en una conección electrica de un automovil, también es posible sacar una sola línea de todo el ramal, si se sabe como se llama la linea, por medio de la función Unbundle By Name.
Cuando se usa la función Bundle, se debe conectar todos los terminales. Cuando se usa la función Unbundle, la dimensión de las conecciones debe coincidir con el número de cables que éste lleva, lo que se logra cambiandole el tamaño por medio de la herramienta de posicionar “la flecha de la paleta”, alargando en una esquina.



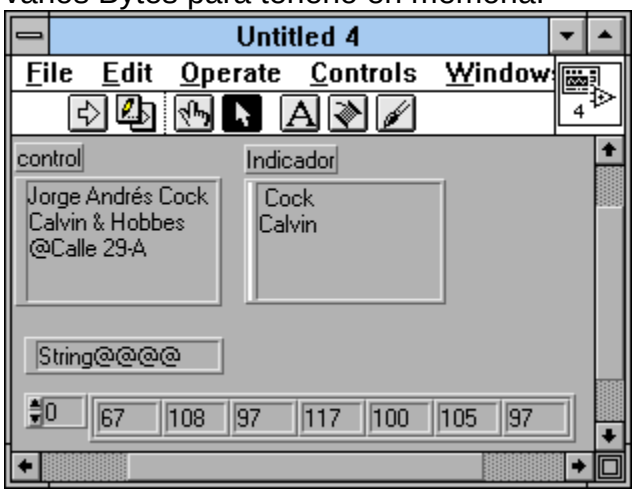


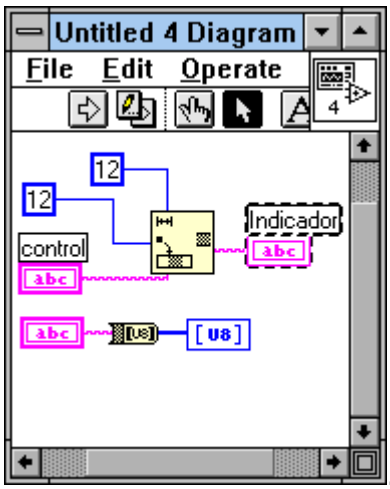
Cuando se usan las funciones Bundle y Unbundle, se debe tener especial cuidado con el orden en que los instrumentos son conectados dentro de los controles o indicadores de cluster, pues esto determina el orden de los cables en los conectores Bundle y Unbundle, y dentro de los mismos cables para conectar con otros cluster. Para cambiar el orden interno se puede hacer por el pop-up menú del marco de cluster dentro del panel de control, con la función Cluster Order, y cambiando el número de orden de cada elemento dentro del cuadro.
Errores de compatibilidad de cables pueden resultar de un orden inadecuado.
En el ejemplo se aprecia como un cable de cluster proveniente de una entrada, se desbarata en todos sus elementos y se vuelve a agrupar. Luego a este cable

grueso se accesa el conductor ‘Número’ y en este se coloca el dato del indicador Número. El resultado se lleva al cuadro cluster de salida.
En el ejemplo también se aprecia como un conjunto de datos se agrupan en un cable, y a este mayor se accesa el dato ‘Número2’ el cual se visualiza en el indicador de resultado.

STRINGS

Los string o cadenas de caractere son conjuntos de datosalfanuméricos como lo son letreros, anuncios y tablas de letras. El almacenamiento de estos es en código ASCII, en modo de texto.
Operar con strings tiene mucha similitud con el manejo de matrices. Hay funciones para dar la longitud de la cadena, para comvertir parte de la cadena a números, para sacar una subsección de la cadena, y para juntar cadenas menores para lograr una mayor, entre muchas otras.
Es muy diferente almacenar un “1” como número que como caracter, pues como número representa el valor uno, usado para cálculos, mientras que como caracter se almacena en memoria como su identificador ASCII, el cual e un código con un número muy distinto que se encuentra entre 0 y 255, el cual es el rango de este código, ya que posee un solo byte.
Generalmente almacenar caracteres ocupa poco espacio en memoria, pues un solo byte es menos que lo ocupado por un número tipo DBL donde se usan varios Bytes para tenerlo en memoria.

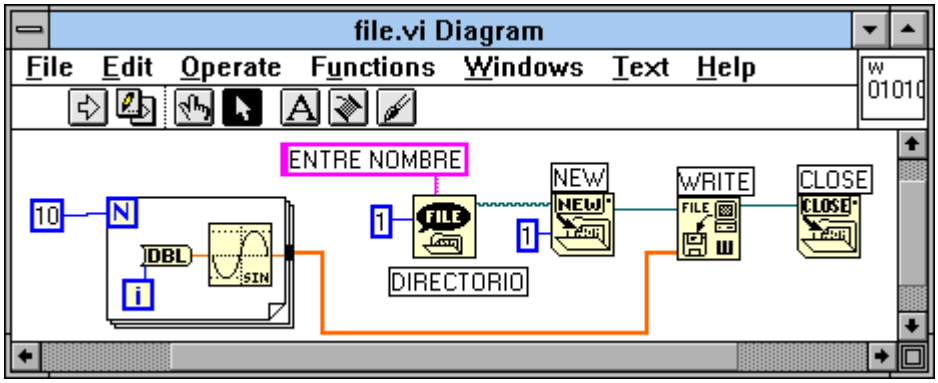




En el ejemplos se muestra como tomar un String inicial dado en el string control, y por medio de una función de String Subset, se saca un trozo de letrero de una longitud de doce caracteres a partir de el caracter número doce, y el trozo se visualiza en el indicador. También se encuentra un ejemplo donde se convierte una cadena de caracteres en una matriz de números cuyo valor ASCII representa la cadena en el control.

FILE

Son funciones que permiten accesar información guardada en disco duro o disquettes, por medio de los directorios.
Con estas instrucciones se pueden almacenar datos obtenidos en el transcurso de la ejecución del VI.
Se recomienda mirar los ejemplos sobre el manejo de estas funciones en el Directorio de Examples, FILE, en el LabVIEW, para tener una buena comprensión.
Para manejar archivos, se debe seguir varios pasos.
-Saber que archivo se va a manejar. Esto se puede lograr tomando una lista de directorio en el lugar donde el archivo se encuentre.
-Abrir el Archivo. Por medio de una instrucción Open. Si el archivo no existe, se debe crear, con la función NEW. Cuando el archivo está abierto se le asigna un número de identificación, para escribir o leer bajo este concepto.
-Escritura o lectura en el archivo, bajo el número de identificación dado.
-Cerrado del archivo. Si el archivo no se cierra, permanece abierto, limitando la posibilidad de trabajar en otros, y ocupando al computador.



En el ejemplo se aprecia como brabar un grupo de datos de una fuente senoidal, en un archivo binario. Para hacer uso de cada función buscar su manejo en la ventana de help.
Dentro de las funciones de manejo de archivos hay algunas listas para escribir en formato de hojas de cálculo, como las usadas en Excel, donde los datos entre columnas se saparan por tabulaciones, y entre filas por ‘enter’.

DAQ, I/O Y OTRAS.

LabView cuenta además de las funciones normales de programación, con otras en forma de VI para ser usadas como subrutinas en aplicaciones de grado más complejo, como son las herramientas usadas para comunicaciones.
Estas funciones cuentan con los programas Driver, para manejar dispositivos que se conectan al computador, como balanzas, osciloscopios, y muchos más. Como se tienen estos Driver, no se hace necesario conocer a fondo el método de programación de estos equipos, solo basta usar las rutinas que los manejan.
Un programa Driver consta de los siguientes elementos:

- Iniciador o Configurador: Entrega el modo de empleo, y como será la comunicación al instrumento o tarjeta. Establece metodos de gatilleo, y configuración del dispositivo para que funcione.
- Iniciador de medición y presentador de estado: Ordena a la tarjeta o dispositivo que tome datos, o tome su propio estado.
- Trasladar datos: Llevar o traer los datos del dispositivo al computador, para ser usados como sea necesario.
- Utiles: Para leer errores, resetear el periferico, o pedir autochequeo y autocalibración.
- Cerrar: Cerrar la comunicación con el dispositivo, para que se mantenga en un estado esperado.

En comunicaciones existen numerosos métodos de trabajo, como son:

Comunicación con tarjetas DAQ.

Las tarjetas DAQ son tarjetas insertables que permiten la entrada y salida de datos del computador a otros aparatos, donde se conectan sensores, y actuadores, para interactuar con el mundo real. Los datos que entran y salen pueden ser señales digitales o análogas, o simplemente conteos de ocurrencias digitales, tanto de entrada, como de salida.

Las tarjetas se comportan como si fueran un puerto más en el computador, y poseen todo un protocolo y sistema de manejo, por lo que entender cada tarjeta, como su funcionamiento, al igual que cualquier instrumento, requiere de tiempo y cuidado.

Existen tarjetas de alto desempeño, y de bajo. Las de alto son programables, y facilitan altas ratas de manejo de información, pues son en cierta forma inteligentes y suficientes, tal como un sistema Stand Alone, y por tanto no comprometen mucho la velocidad y rendimiento del computador.

Las tarjetas de bajo desempeño requieren de un control directo del computador, y se ven limitadas por la velocidad de éste. El windows en cierta forma es un sistema operativo que no trabaja en tiempo real, para operaciones donde la rata de muestreo es muy alta, como en aplicaciones de audio, radar, vivraciones y video, aunque para aplicaciones de lentitud considerable es bueno, como en controles de hornos. En aplicaciones lentas Windows y tarjetas simples bastan porque los tiempos perdidos por el sistema de interrupciones de Windows (sea por mover el mouse o cualquier otra cosa) no afectan comparativamente.

Para aplicaciónes de alta velocidad y tiempo real, se requiere de hardware especial, osea tarjetas inteligentes, que se programen, y transfieran los datos a memoria, ya sea por rutinas de DMA (acceso directo a memoria), o por rutinas de interrupciones al procesador.

Las tarjetas como cualquier otro periférico, requiere de sus parámetros de programación, y hasta protocolos de comunicación, por lo que se requiere de un software Driver que maneje lo bajo de programación, y deje en la superficie, la posibilidad de programar aplicaciones con los beneficios de dichas tarjetas, de una forma sencilla.

LabVIEW ofrece acceso a los driver desde las rutinas de configuración. Los driver disponibles son para las tarjetas de la NI National Instruments, pero en el mercado se consiguen driver para otras marcas como PC-LAB.
La configuración se hace a través del programa anexo a LabView, NI-DAQ.

Comunicación a través del puerto serial.
Se trasmite la información por un puerto que puede ser el COM1 o el COM2, de forma serial, osea a través de un solo cable, y cada bit pasa uno tras otro a alta velocidad. Para la comunicación entre computadores se establece un protocolo común para que la información sea entendida por ambos. Se debe definir el tamaño de los BUFFER para almacenar datos mientras se realiza la comunicación. También se debe definir si hay Handshaking, el cual consiste en que el que recive cuando valla a tener lleno el bufer de información mande una instrucción (Si es por software es un comando <ctrl-S>, si es por hardware por una linea) para detener la trasmisión, y otra para reanudar la transmisión de información (Por software <Ctrl-Q>, por hardware una linea).
Se debe tener cuidado al trasmitir caracteres como <Ctrl-Q> y <Ctrl-S> porque pueden ser tomados como isntrucciones.
LabView cuenta con funciones para iniciar, escribir y leer el puerto serial.

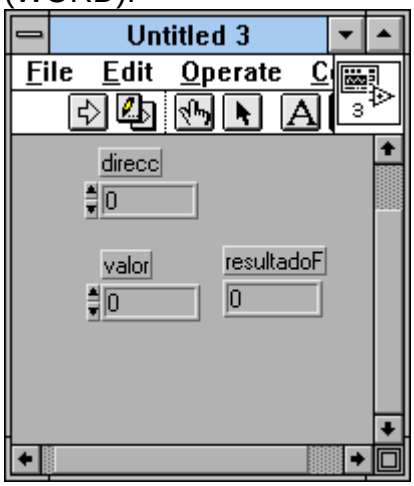
Comunicación a través de un puerto de GPIB.
EL GPIB (General Purpose Interface Bus ANSI/IEEE 488.1 y 488.2), es un puerto diseñado por la Hewlett Packard, para establecer comunicación con instrumentos de medición. Muchos de los instrumentos como son Balanzas, Osciloscopios, multímetros y equipos de tipo “Stand Alone” (que no requieren de un computador para funcionar, son independientes) cuentan con este tipo de puerto.
Los manejos de este puerto y los protocolos y controles de software Driver para GPIB son complejos y no se esplicaran en este libro, por ser éste simplemente una introducción para los principiantes. Si se desea tener un mejor conocimiento favor remitirse a los catálogos y manuales del LabView.
Esta capacidad de comunicación es uno de los fuertes del LabView, pues este viene con variedad de librerías de software driver, controlador de los instrumentos, con los protocolos de comunicación que maneja cada uno, así no se requiere de conocer el nivel más bajo de programación de dichos instrumentos, simplemente es mirar si se tiene el driver de dicho instrumento, y conectarlo.

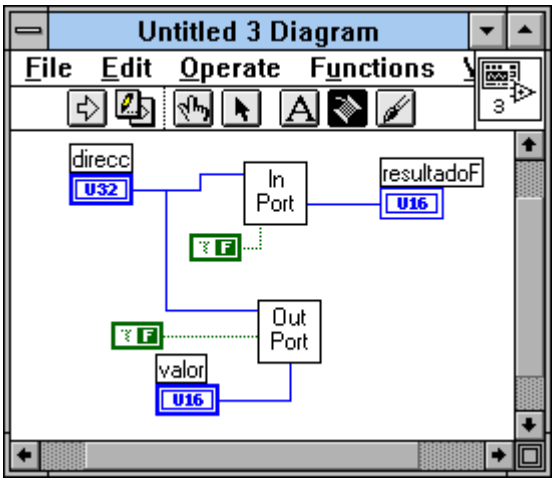
Comunicaciones con redes TCP, UDP
Para comunicación de equipos en red. Favor remitirse al manual.

Comunicaciones dinámicas entre programas de Windows DDE.
Si se tiene una base de datos abierta, es posible accesar datos de esta, y usarlos por el LabView, y viceversa, lo que sirve para una actualización dinámica. Ver los ejemplos en el directorio de Examples. Se recomienda tener

un conocimiento claro del uso de aplicaciónes en Windows, como es Acces, y tenerlo cargado y funcionando.
Remitirse al manual.

Para usuarios con INPORT, OUTPORT.
Si simplemente se tiene conectada una tarjeta fabricada en casa, y se conoce la dirección de memoria del puerto al que está conectada, y solo hace falta leer o escribir datos en esta localización, se puede hacer por medio de los comandos INPORT u OUTPORT, dando el valor a escribir, orecibiendo el valor a leer, dando la dirección, y especificando si se lee o escribe un solo byte, o dos (WORD).





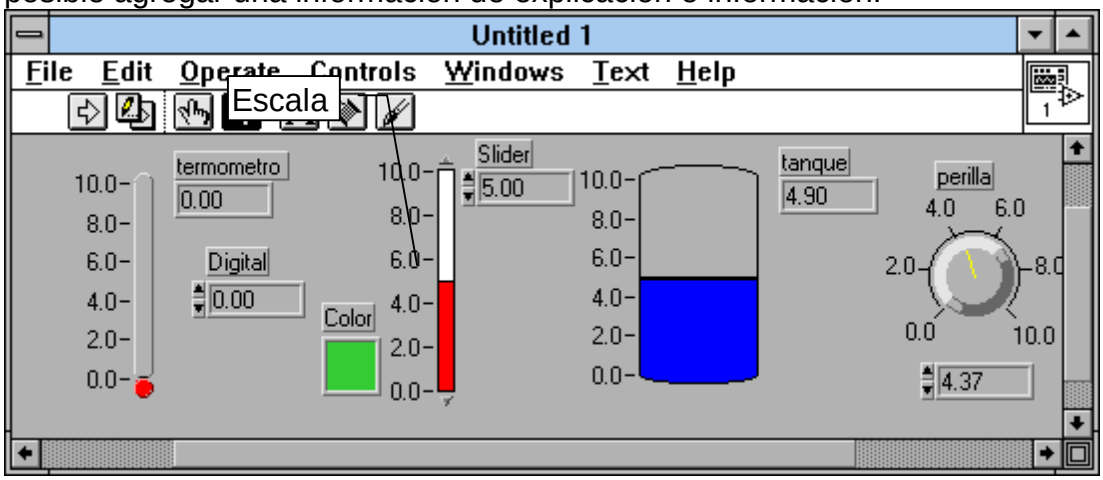
6. PANEL FRONTAL DE CONTROL

PRESENTACION DEL PANEL FRONTAL

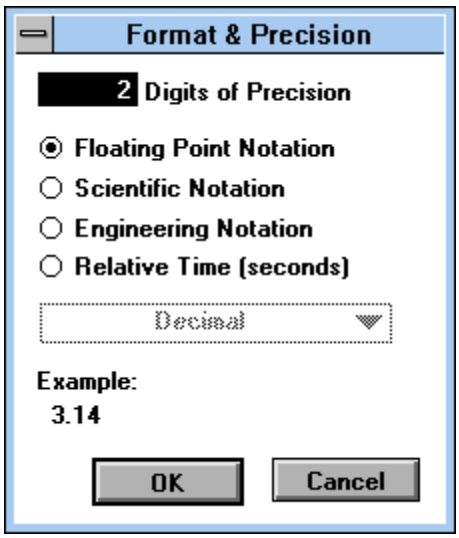
El panel frontal es el medio con el que el programa interactua con el usuario. Mientras más esquemático sea, más humano, más entendible, y más cerca del usuario se encuentre, mucho mejor.
Posiblemente quien use el programa no tenga la menor idea de como usar un computador, por lo que en aplicaciones como en la automatización de plantas, si se contara con un dibujo mímico del proceso facilitaría mucho el trabajo del operario.
LabView como lenguaje gráfico cuenta con múltiples herramientas de diagramación y presentación de la informacíon, de una forma clara, como son los indicadores numéricos, los de tipo boleano, strings, arrays, tablas, etc y los controles equivalentes.
Todo indicador se puede usar como control si se configura como éste, por medio del pop-up menú del mismo, con la opción **Change to Control**. Lo mismo con los controles con **Change to Indicator**.

Los controles e indicadores son coloreables con la herramienta de color para una mejor presentación.
El menú de Controls cunenta con un submenú de decoraciónes que pueden ayudar con la presentación.

INDICADORES Y CONTROLES NUMÉRICOS
Algunos controles e indicadores Numéricos son los digitales, los de perilla, los de deslizador, y otros tal como se encuentran en los equipos de sonido u otros aparatos comunes.
Cada control tiene sus diferentes parámetros para configurar a través del pop-up menú propio. Se puede lograr que muestre un label o letrero con el nombre, mostrar un display con el dato marcado, que muestre o no la escala. También es posible agregar una información de explicación e información.



Para cambiar la escala solo hace falta usar la herramienta de texto, para usarla sobre la escala del control directamente y alli escribir los valores máximos y mínimos deseados.
Un control numérico se puede configurar para que muestre los valores en números OCTALES, HEXADECIMALES, BINARIOS, DECIMALES, si el dato que maneja es de tipo entero con o sin signo, por medio dela opción Format & Presicion del pop-up menu del control. Si es de tipo punto flotante, se puede variar el FIX o número de ceros decimales, además de mostrar notaciones científicas o de ingeniería, a través de la mísma opción.

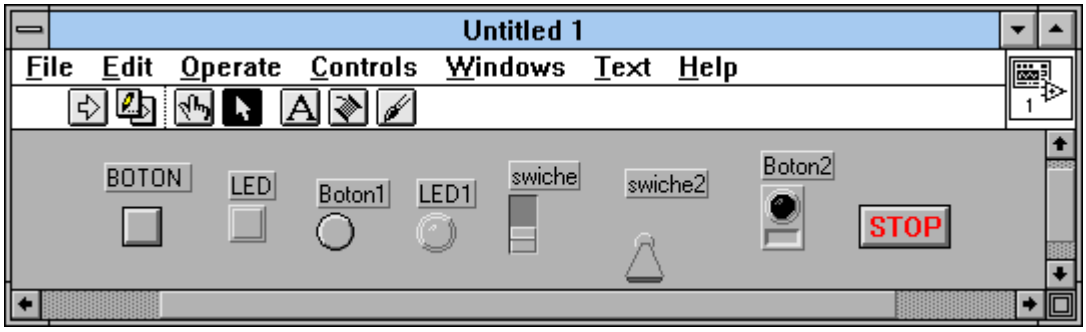


INDICADORES BOLEANOS

Los swiches boleanos son controles qu retornan solo dos valores al programa, o uno o cero, verdadero o falso. Los indicadores solo muestran esas posibilidades. Indicadores boleanos existen muchos, como son los bombillos piloto, los LED, y otros. Swiches tambien existen de numerosos tipos, como son palancas, pulsadores, botones, etc.

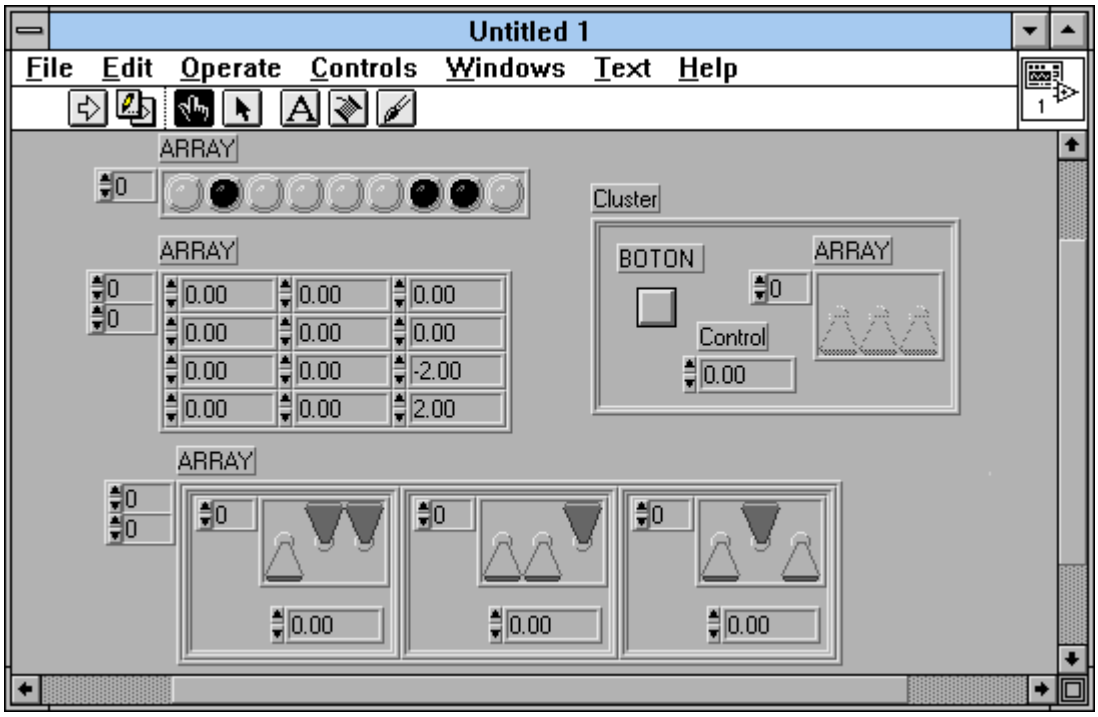
Entre los botones y pulsadores hay numerosas configuraciones de funcionamiento, como por ejemplo que el botón retorne un verdadero después de oprimir y soltar, o que retorne el verdadero al oprimir y al soltar regrese a su posición de falso, etc..

Para cambiar la configuración del boleano, hacerlo por su pop-up menú a través del submenú **Mechanical Action**.

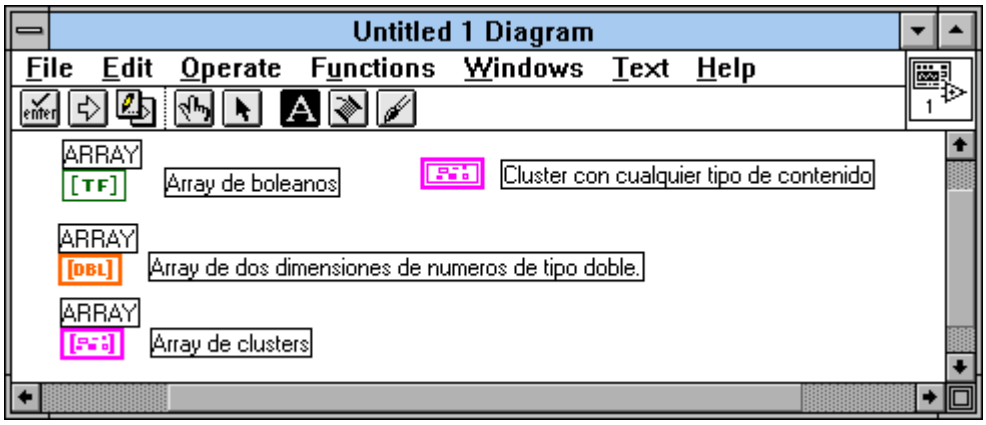


ARRAY & CLUSTER

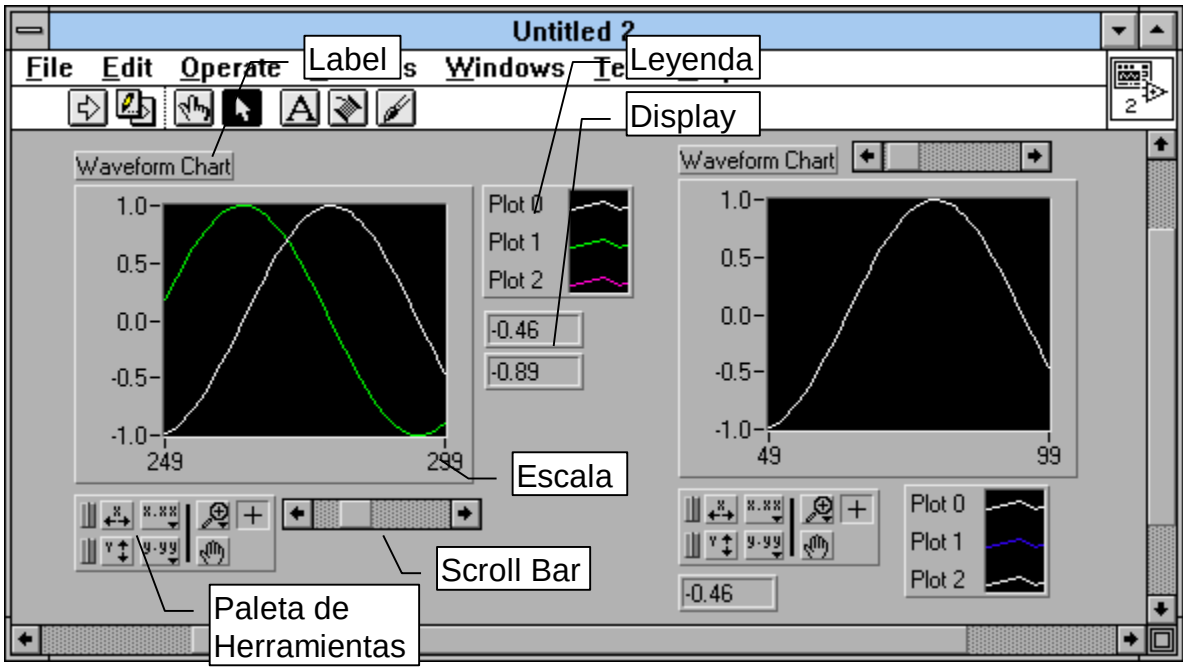
Los indicadores y controles de array y cluster son cuadros donde se colocan otros indicadores o controles, para el manejo de estas estructuras. Un control solo sirve como control, y un indicador solo sirve como indicador. Para hacer el cambio hacerlo como cualquier otro control.

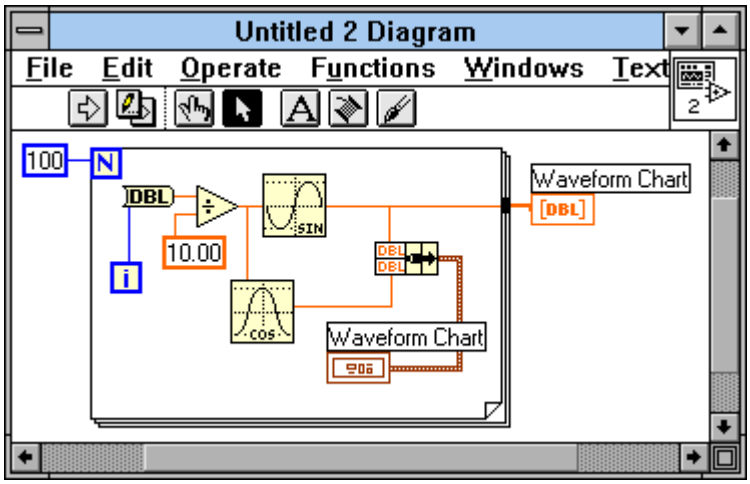


Para agregar dimensiónes a las matrices hacerlo con la herramienta de posicionamiento, agrandando el indicador de las dimensiones.
Un cluster puede contener un Array interno , junto con otros elementos.
Un array o matriz puede estar formada de clusters. Esto es útil en aplicaciones de bases de datos.



GRAFICADORES
LabView cuenta con algunos controles o indicadores que presentan gráficas de los datos obtenidos en el programa. Estos se encuentran en el submenú Graph en el menú de controles. Para cada uno se pueden configurar muchos parámetros como escala de la gráfica, autoescala, color de las líneas, número de líneas en una gráfica, presentación de letreros, paletas de control, indicadores, etc.

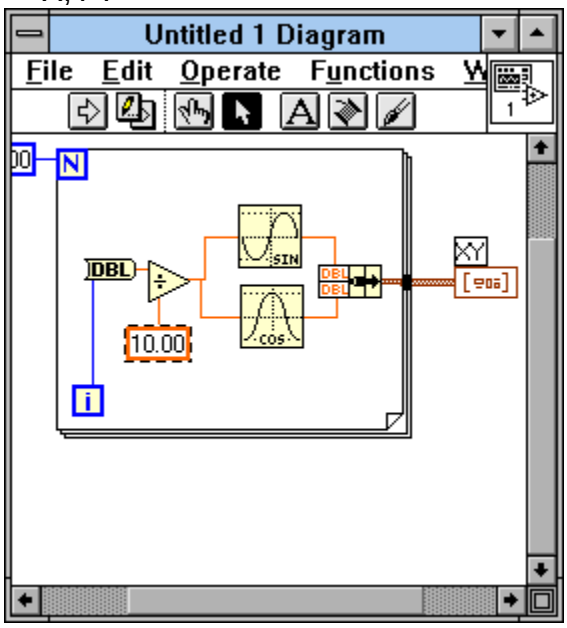


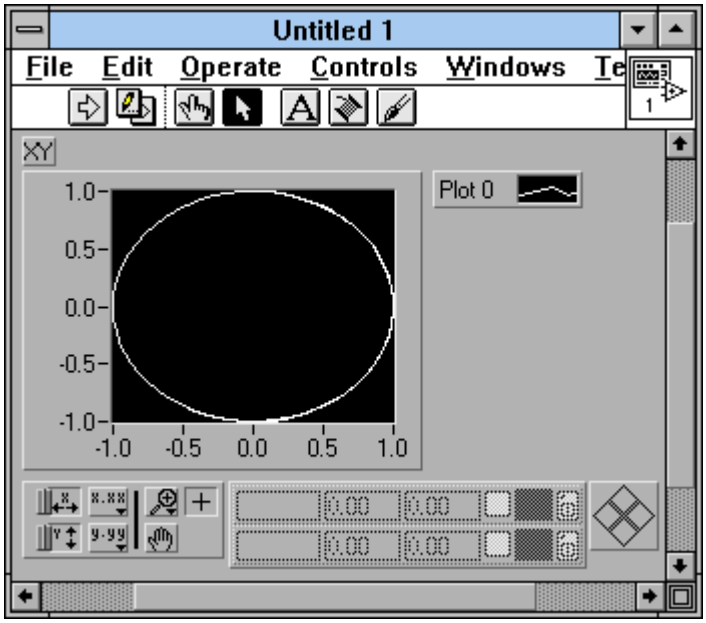


Entre los graficadores se encuentran:

- Waveform Charts: Como el anteriormente dibujado permite entrar datos de a un número, o en una matriz de una dimensión. Grafica dando automaticamente la secuencia en el eje X, simplemente por el dato que sigue al anterior, osea por pasos. El eje X siempre corresponde a tiempo. Si se grafican dos o más datos al tiempo se deben entrar como un cable cluster, como se muestra en la figura. Si se deseea entrar todos los datos al tiempo para llenar un buffer de la gráfica, hacerlo como una matriz, y si se desea graficar llenando buffers de varias graficas simultaneamente, entrar los datos como una matriz de clusters.
- Waveform Graph: De comportamiento similar a la Waveform Chart, pero con esta ya se puede definir la escala en el tiempo, a los valores deseados, mientras que en la anterior la escala en X es propiamente de pasos, más que tiempo.

XY Graph: En ésta se entran los datos por pares ordenados en una matriz bidimensional, o una matriz de clusters de dos datos cada uno X,Y. Permite graficar funciones matemáticas, circulos, etc, dando una secuencia de puntos, X,Y .





- Intensity Graph e Intensity Chart: Para graficar planos de diferentes colores, para matrices de dos dimensiones, donde los valores contenidos corresponden a un color.

ATRIBUTOS DE LOS CONTROLES

Cuando se crean controles e indicadores se puede desear que estos cambien su configuración durante el curso del programa, para efectos de visualización. También se puede desear obtener datos del estado de los controles. Por ejemplo que al oprimir un botón aparezca un panel adicional de controles, sin necesidad de llamar una subrutina, o que con un botón se tenga el zoom de una gráfica.

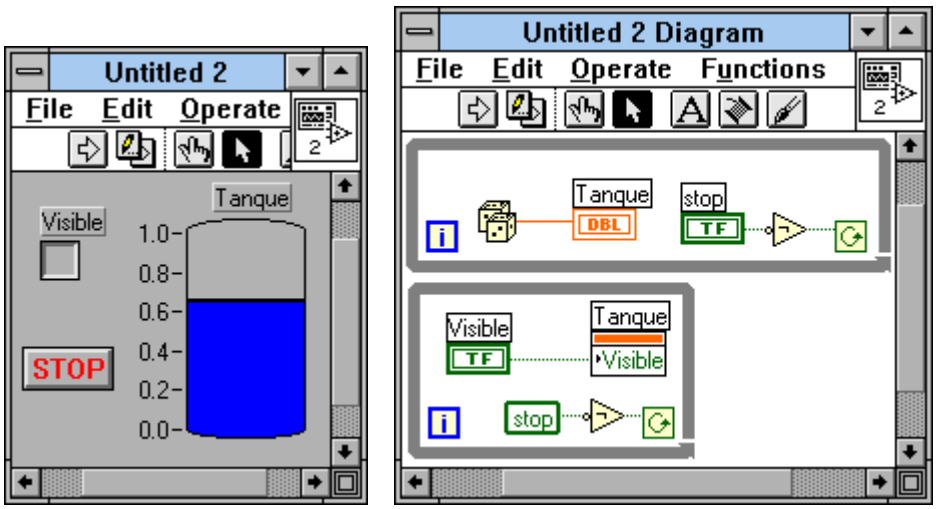
Todo esto se logra a través de un nodo de atributos, el cual se obtiene por medio del pop-up menú de cada control, por la opción **Create Aatribute Node**. Cuando esto se hace aparece un ícono asociado al control en el diagrama de bloques, el cual puede usarse para recibir o entregar datos.

En el ejemplo se tiene un nodo asociado al indicador del tanque. Este nodo constantemente controla si se ve o no, el tanque en el panel frontal ('Visible'). Esta operación es independiante a lo que muestra el tanque, una serie de números aleatorios, hasta que se de stop al programa.

Los atributos de nodos pueden controlar muchos factores en los controles. Para cambiar entre los posibles factores a variar o leer en el nodo, hacerlo por medio del pop-up del menú del modo, a través de la opción **Select Item**.

A través de estos nodos esposible leer la posición de un gursor dentro del graficador, y mucho más.

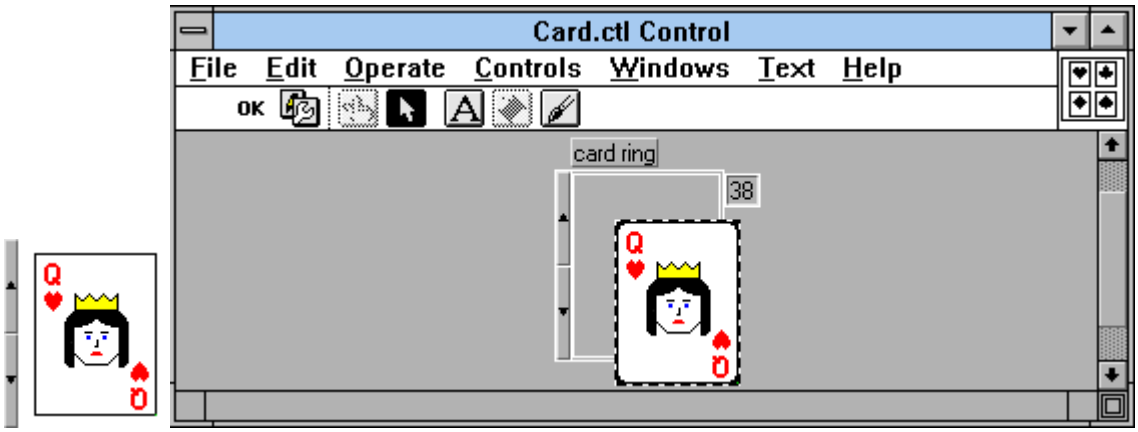
Para tener una mejor idea de los Nodos de Atributos, mirar los ejemplos en el directorio de Examples, General, Atribute del LabView.



EDICION DE CONTROLES

Si los controles ofrecidos por el LabView no parecen satisfactorios, se pueden editar para lograr que tengan una presentación como se desee. Por ejemplo que muestren el mímico de una bomba en estado de prendido, y una bomba en el estado de apagado. Es tal el grado de edición que no sería ambicioso esperar que un control fuera el rostro de una persona guiñando un ojo, donde este movimiento sería de tipo boleano.

Para editar controles se debe seleccionar con la herramienta de posicionamiento un control con funciones y características similares al control que se desea obtener. Luego en el munú de edit seleccionar **Edit Control**. Aparecera una nueva ventana para editar el control y modificarlo. El control nuevo se puede grabar como un archivo de extensión .CTL Por ejemplo se ve la carta de Poker lograda desde un control de figuras Pict Ring.



El panel mostrado presenta herramientas similares a las del panel frontal en modo de edición pero en ves de tener un botón para ejecutar posee uno de OK para aceptar cuando está listo.
El botón de cambio de modo varía entre un modo de edición general tal como en el panel frontal, y un modo de despieze del control. Se aprecia con forma de llave inglesa. En este modo se puede tomar cada parte del control y modificarlo independiente.
Si se tiene un dibujo dentro del Clip-Board, con el pop-up menú de cada parte se puede usar la opción **Import Picture** para reemplazar esa parte por el dibujo en el Clip.
Para ubicar los controles ya grabados en el panel frontal, hacerlo por la opción **Control...** en el menú de controles.

7. PROGRAMAS EJEMPLO

A. CONTROL DE UN MOTOR PASO A PASO:

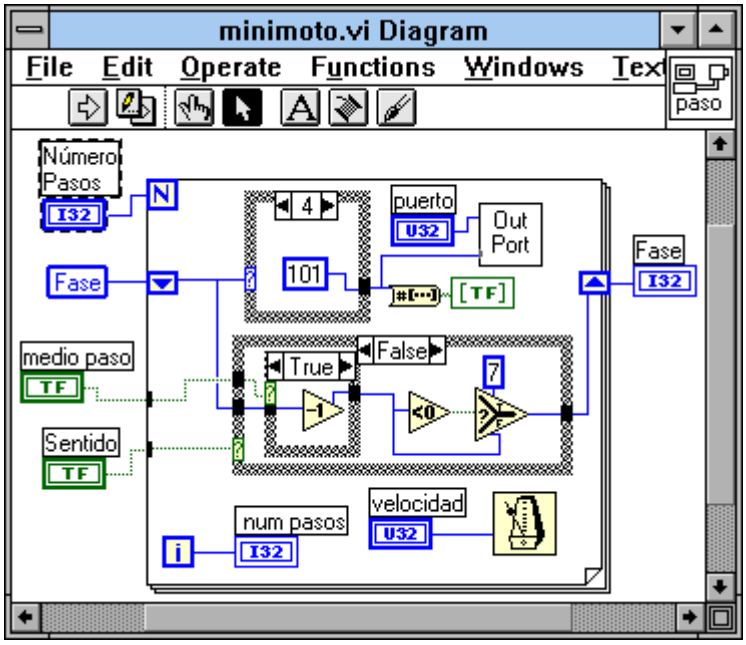
ENUNCIADO
La idea es controlar un motor de pasos el cual está conectado a una tarjeta que sale del puerto paralelo del computador, la cual está localizada en la posición de memoria H378.
El motor consta de cuatro bobinas que se deben emergizar intercaladamente, según una secuencia dada que determina el sentido de giro y el tamaño del paso. Como son cuatro bits, son 16 las posibles combinaciones de las cuales se usan ocho. La combinacion esta dada asi en orden: 1010, 1000, 1001, 0001, 0101, 0100, 0110, 0010, de forma que si el coambio va de izquierda a derecha el motor sigue una dirección, con movimientos de medio paso. Si va de derecha a izquierda, el motor gira en sentido contrario, también de a medio paso.

Para variar pasos enteros se debe saltar un número en la secuencia, de código binario.
Se debe controlar el número de pasos que se desea dar, la velocidad del motor, y se debe determinar con un botón el sentido de giro, y con otro botón si avanza de a uno o medio paso. Como cada puesta en un código implica estar en una fase de estado entre ocho posiciones, se debe reportar en que fase quedó al final, y al empezar de nuevo el programa debe comenzar por la fase en que terminó anteriormente, para evitar bloqueos del motor.
Debe mostrar La combinación de bits en que pasa a cada momento, y el número de pasos que ha dado actualmente.

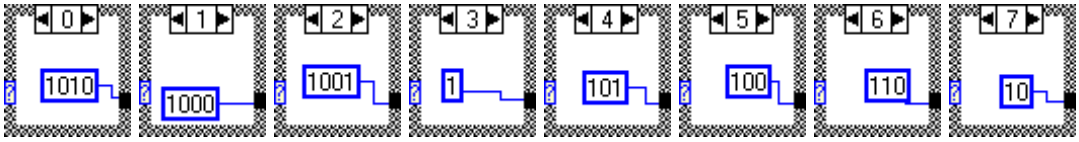
DESARROLLO

1. Crear el pánel frontal con los botones y controles deseados: Número de pasos, Puerto, Sentido, Medio Paso, Velocidad, Bits de Control, Fase, Número de pasos actuales,
Tomando en cuenta cuales son controles y cuales indicadores (fase, Bits de cont., Num de pasos actuales).
Como número de pasos no es fraccionario, se configura como un entero de 32 bits, que es lo que entiende un ciclo For-Next.
Tomar en cuenta que Sentido y Medio Paso son controles boleanos.
Bits de Control es un array de indicadores boleanos LED.

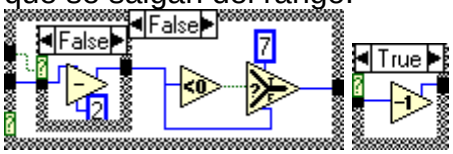
2. Crear el diagrama de bloques o programas. Un ciclo For-Next repetirá su contenido el número de pasos o mediopasos que se vallan a dar.

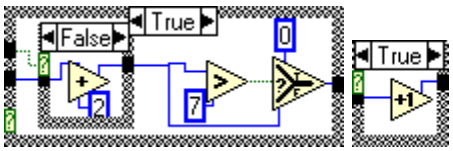


Un Shift Register almacenará el estado de fase en el que se encuentra y servirá para alimentar un cuadro de casos del uno al 7, donde según el número de estado permitirá a un número binario ir al cuadro outport.
Outport escribe dentro de la localización del puerto, el número dado por el cuadro Case.
El número binario se traduce a una matriz de boleanos por medio de un conversor, y es mostrado en la matriz Bits de Control.



Según el sentido de giro se resta o suma un valor al número de estado para que pase al estado siguiente, y este valor se lleva al Shift Register para que sea el número que realmente dice el estado actual.
Si el nuevo estado es un numero menor que cero, el estado actual debe volverse siete, y si es mayor que siete debe volverse cero, para que nunca hallan valores que se salgan del rango.





Si se trata de un paso entero se sumará o restará dos estados y si se trata de medio paso se sumará o restará uno al valor del estado.
El indicador de número de pasos se conecta al contador del ciclo para saber cuantos ciclos se han dado.
El control de velocidad se conecta a un temporizador que hace que pasen determinado número de milisegundos entre ciclo y ciclo.
Al terminar todos los ciclos el contenido del Shift Register pasa al indicador de fase para ver en que fase quedo. Se aprecia como el contenido del Shift inicial sale de una variable local asociada al contenido de la fase, así cuando comieza el programa en el shift inicial entra el contenido de esta variable, que no es más que el contenido de la fase final de la última vez que se corrió el programa.
Este algoritmo es una de las posibles soluciónes a este problema, aunque existen muchas mas soluciones.
Una mejora que se deja al usuario es hacer que cuando valla en el estado siete y trate de aumentar un paso, tratando de pasar al 9 y como este no existe pasa al cero, debiendo pasar al uno, haciendo que de sobresaltos el motor. Esto se logra haciendo que en ves de cambiar el estado por cero, simplemente reste ocho. Muy similar para el sentido contrario.

3. Crear el ícono del programa, y hacer las conecciones del panel frontal con los conectores del ícono, para poder ser llamado como subrutina, y grabar el VI.

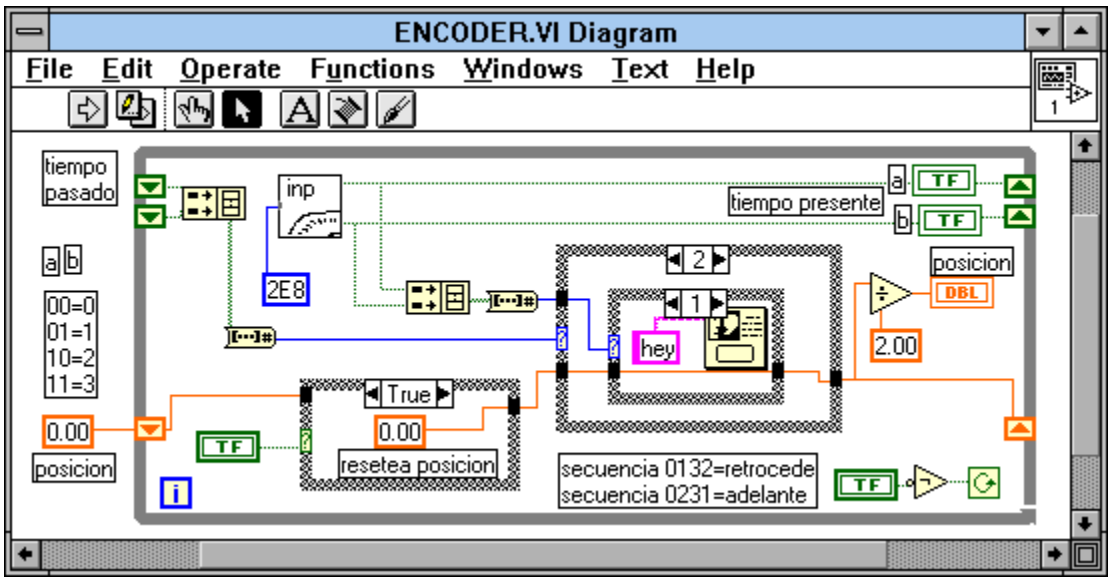
B. PROGRAMA DRIVER DE UN ENCODER DE DOS BITS.

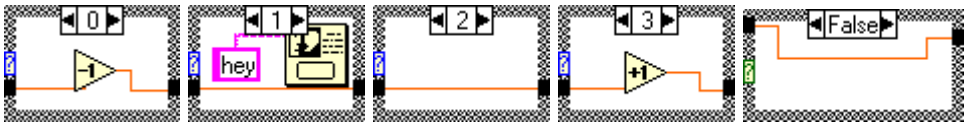
ENUNCIADO
Se tiene un sensor encoder el cual consta de una regla perforada, dos emisores de luz y dos fotosensores. Según alternen las señales en los fotosensores, con luz, no luz, uno y cero, se puede detreminar el movimiento de los sensores respecto a la regla, según la secuencia. Entre cambio y cambio de estado existe medio paso de avance de la regla, donde un paso es el ancho de un hueco, o el ancho entre separaciones de huecos. Los fotosensores estan separados paso y medio.
La secuencia es 00, 01, 11, 10 en retroceso, y 10, 11, 01, 00 en avance hacia adelante.
Se desea que el programa muestre el estado de los sensores, muestre la posición absoluta respecto al origen de la regla, permita resetear el origen al lugar donde e encuentra, y detener el programa.

1. Crear el pánel frontal con los controles e indicadores requeridos, Dos LEDS, uno para el estado de cada sensor, un indicador para la posición, un botón de stop, y un de reset.

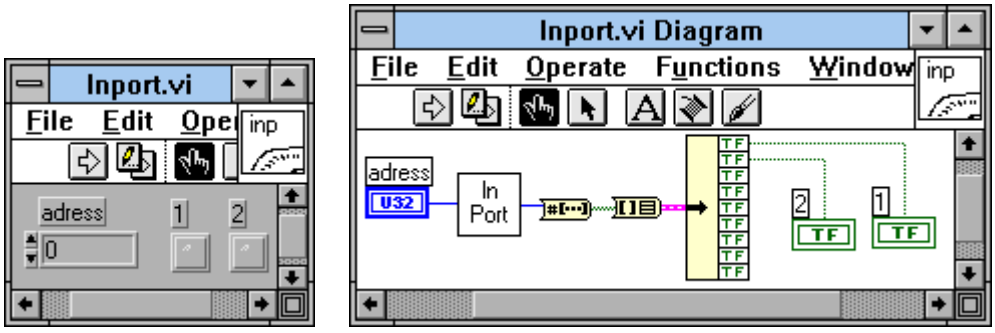
2. Crear el diagrama de bloques. Como se debe chequear constantemente hasta que se detenga el programa, se habre una estructura While, y se conecta al botón de stop a través de un elemento negador, para que cuando el boton sea presionado se haga verdadero, y al negar se haga falsa la entrada al control del ciclo para que se detenga. El boton ha sido configurado como pulsador, por medio de la opción Mechanical Action en el pop menú del stop.
Como se trata de un elemento con memoria donde se compara un estado actual con uno anterior, se colocan dos Shift register para almacenar el estado anterior de los sensores y actualizar esta información con lo leido desde el puerto.
Otro Shift almacena la posición respecto el origen, y mediante los cambios se actualiza.
El nuevo estado de los sensores es tomado por la subrutina import a la cual se le da como parámetro, el puerto en el que debe leer, y retorna los dos bits deseados, que son llevados a los shift de nuevo valor, pero siendo antes mostrados por los indicadores de estado actual de los sensores.
Tanto los bits de estado anterior como los del nuevo estado leido son agrupados en una matriza de dos bits, por medio de las funciones Build Array, y estas matrices son traducidas a números por medio de un conversor. Según estos números se analiza que pasa cuando cambia el estado de los sensores a través de cuadros case anidados, donde según el estado anterior se evalua el nuevo estado, y se retorna o no un cambio en la posición. Por ejemplo si el estado anterior es 11 y el nuevo es 11, simplemente los cuadro case dejan pasar el valor de posición sin variarlo. Si el anterior es 00 y el nuevo es 11, es porque hubo un error pues físicamente es imposible, entonces los cuadros no alteran el valor, y además lanzan un anuncio diciendo que algo pasó. Si es caso anterior es 11 y el sigiente es 10 es porque se está retrocediendo, entoncesse resta uno a la posición. y por último si el caso anterior es 11 y el nuevo 01, se suma uno a la posición. En total son 16 combinaciones de las cuales 12 son físicamente posibles.
Según el botón de reset se controla un case, que en caso de no presionar, deja pasar el valor de posición, y en caso de dar reset, coloca un cero en la linea de posición, para que se almacene en el shift.

Finalmente el resultado de posición se divide en dos para dar el valor en terminos del paso, pues con avanzar medio paso hay un cambio de estado en los sensores.





Para lograr traer los bits hay que hacer la rutina de import.
Se aplica el mismo procedimiento de crear VI, primero el panel de control, con los bit de salida y el puerto de entrada.
Luego el diagrama, donde el valor del puerto se lleva a una rutina import. el resultado se convierte a una matriz de boleanos. La matriz se traduce en un cable de cluster para que por medio de la función Unbundle se puedan sacar los dos bits deseados para llevarlos a los indicadores.



Luego se crea el ícono, y se hace la conección del panel frontal con el icono en los conectores que este posee. Finalmente se graba.

Desde el VI del encoder se llama la subrutina inport a través de la opción VI... del menú de funciones, y se enlaza con el diagrama de bloques.

Por último grabar el driver del encoder. Si este se desea usar como subrutina, no se debe olvidar crear el ícono de este y hacer las conexiones del caso con el panel frontal.