

CURSO BASICO

LabVIEW 6i

GERMÁN ANDRÉS HOLGUÍN LONDOÑO

Profesor Auxiliar, Facultad de Ingeniería Eléctrica

SANDRA MILENA PÉREZ LONDOÑO

Docente Especial, Facultad de Ingeniería Eléctrica

ÁLVARO ÁNGEL OROZCO GUTIÉRREZ

Profesor Asociado, Facultad de Ingeniería Eléctrica

UNIVERSIDAD TECNOLÓGICA DE PEREIRA

CURSO BÁSICO DE LABVIEW 6i

GERMÁN ANDRÉS HOLGUÍN LONDOÑO
Profesor Auxiliar, Facultad de Ingeniería Eléctrica

SANDRA MILENA PÉREZ LONDOÑO
Docente Especial, Facultad de Ingeniería Eléctrica

ÁLVARO ÁNGEL OROZCO GUTIÉRREZ
Profesor Asociado, Facultad de Ingeniería Eléctrica

UNIVERSIDAD TECNOLÓGICA DE PEREIRA

ISBN: 958-8065-33-X

Facultad de Ingeniería Eléctrica
Universidad Tecnológica de Pereira

Impreso en Pereira, 2002.

TABLA DE CONTENIDO

1. AMBIENTE DE DESARROLLO DE LabVIEW.....	5
1.1 OBJETIVO	5
1.2 DESCRIPCIÓN	5
1.3 Panel Frontal.....	7
1.4 Diagrama de bloques	8
1.5 Paleta de herramientas.....	9
1.6 Paleta de controles	11
1.7 Paleta de funciones.....	14
1.8 Barras de LabVIEW	18
1.8.1 Barra de Menús.....	18
1.8.2 Barra de Herramientas del Panel Frontal	22
1.8.3 Barra de Herramientas del Diagrama	24
1.9 Técnicas de cableado	26
1.10 Técnicas de Edición de diagramas.....	29
1.11 Técnicas de Navegación	30
1.12 Tipos de cables.....	32
1.13 Tipos de datos numéricos	33
1.14 Tipos de Terminales.....	34
1.15 Ayudas de depuración	35
1.16 AMBIENTE DE DESARROLLO INTEGRADO de LabVIEW	36
1.16.1 Ejecución de LabVIEW	36
1.16.2 Paletas y ventanas de LabVIEW	39
1.16.3 Ubicación de los objetos.....	41
1.16.4 Edición de objetos del panel.....	47
1.16.5 Cambiar el tamaño de los objetos	50
1.16.6 Ejecución de una aplicación	52

1.16.7 Guardar un VI.....	53
EJERCICIO 1.1 CREACIÓN DE UN NUEVO VI.....	55
1.17 EJERCICIOS PROPUESTOS.....	62
2. ESTRUCTURAS.....	63
2.1 OBJETIVO	63
2.2 DESCRIPCIÓN	63
2.3 Estructura “ <i>While Loop</i> ”	64
EJERCICIO 2.1 Simulación de la lectura de una temperatura.....	66
EJERCICIO 2.2 Generación de una onda seno.....	69
2.4 Estructura “ <i>For Loop</i> ”	71
EJERCICIO 2.3 Gráfica de 100 números aleatorios entre 10 y 50	72
EJERCICIO 2.4 SUMAR los números enteros entre 1 y 100.....	74
EJERCICIO 2.5 promediAR los últimos dos datos aleatorios	77
2.5 Estructura “ <i>SEQUENCE</i> ”	78
EJERCICIO 2.6 Medir el tiempo que el pc tarda.....	80
2.6 Estructura “ <i>Case</i> ”	83
EJERCICIO 2.7 Menú de opciones.....	85
2.7 Estructura “ <i>Formula Node</i> ”	92
EJERCICIO 2.8 Utilización de los nodos de fórmula.....	93
EJERCICIO 2.9 Realizar el ejercicio 1.1 con un nodo de fórmula.....	95
EJERCICIO 2.10 ejercicio 2.7 con un nodo de fórmula.....	96
EJERCICIO 2.11 La ecuación cuadrática.....	98
2.8 acciones mecánicas de los controles booleanos	101
2.8.1 Switch When Pressed	103
2.8.2 <i>Switch When Released</i>	104
2.8.3 <i>Switch Until Released</i>	107
2.8.4 <i>Latch When Pressed</i>	108
2.8.5 <i>Latch When Released</i>	110
2.8.6 <i>Latch Until Released</i>	111
EJERCICIO opcional 2.12 Aplicación de una estructura de secuencia.....	112

2.9 EJERCICIOS PROPUESTOS.....	117
3. ARREGLOS Y CLUSTERS.....	118
3.1 OBJETIVO	118
3.2 ARREGLOS	118
EJERCICIO 3.1 De un arreglo 1D, generar las salidas REQUERIDAS.....	132
EJERCICIO 3.2 Extraer datos de un arreglo 2D	135
3.3 CLUSTERS.....	138
EJERCICIO 3.3 Utilización de los clusters.....	143
3.4 EJERCICIOS PROPUESTOS	146
4. GRAFICADORES.....	147
4.1 OBJETIVO	147
4.2 DESCRIPCIÓN	147
4.3 GRAFICADOR <i>WAVEFORM CHART</i>	150
EJERCICIO 4.1 Gráfico de escalares <i>waveform chart</i>	151
EJERCICIO 4.2 Gráfico de vectores con <i>waveform chart</i>	155
4.4 Tipo de Dato <i>WDT</i>	158
EJERCICIO 4.3 <i>waveform data type</i> con <i>waveform chart</i>	161
EJERCICIO 4.4 MÚLTIPLES GRÁFICAS EN UN <i>waveform chart</i>	163
EJERCICIO 4.5 <i>cluster</i> de escalares EN UNA <i>waveform chart</i>	165
4.5 GRAFICADOR <i>WAVEFORM GRAPH</i>	167
EJERCICIO 4.6 <i>waveform graph</i> utilizando datos <i>WDT</i>	169
Ejercicio 4.7 <i>waveform graph</i> a partir de un <i>cluster</i>	172
Ejercicio 4.8 múltiples gráficos.....	173
4.6 GRAFICADOR <i>XY GRAPH</i>	175
Ejercicio 4.9 <i>XY Graph</i>	176
EJERCICIO 4.10 Múltiples Gráficos en un <i>XY Graph</i>	179
4.7 EJERCICIOS PROPUESTOS.....	181
5. SUBVIS, VARIABLES LOCALES Y GLOBALES.....	182
5.1 ObjetivoS	182
5.2 SubVIs	182

5.3 Edición del Icono.....	184
5.4 Los Conectores.....	187
5.5 UTILIZACIÓN DE UN SUBVI.....	191
EJERCICIO 5.1 Filtrado de una señal.....	193
EJERCICIO 5.2 Menús en los Paneles frontales.....	201
EJERCICIO 5.3 Cargar el panel frontal de un subVI durante la ejecución.....	204
5.6 VARIABLES LOCALES.....	210
EJERCICIO 5.4 Graficar datos de distinta fuente en un mismo <i>chart</i>	211
5.7 VIARIABLES GLOBALES.....	218
Ejercicio 5.5 Variables Globales.....	221
5.8 ejercicios propuestos.....	223
6. CADENAS Y ARCHIVOS.....	224
6.1 OBJETIVO.....	224
6.2 CADENAS.....	224
Ejercicio 6.1 CONCATENACIÓN, CONVERSIÓN Y BÚSQUEDA.....	238
EJERCICIO 6.2 DATOS DE UN MEDIDOR.....	240
6.3 ARCHIVOS.....	243
Funciones.....	244
6.3.1 Escribir Datos en un Archivo.....	249
Ejercicio 6.3 guardar datos en un archivo ascii.....	249
Ejercicio 6.4 CONSTRUIR UN ARCHIVO TIPO <i>TAB DELIMITER</i>	251
6.3.2 Leer un archivo.....	254
Ejercicio 6.5 RECUPERACION DE DATOS DE UN ARCHIVO ASCII.....	254
Ejercicio 6.6 ESCRIBIR UN ARCHIVO BINARIO.....	256
Ejercicio 6.7 RECUPERACIÓN DE datos de UN ARCHIVO BINARIO.....	258
6.4 Funciones de alto nivel para el manejo de archivos.....	259
6.5 Ejercicios propuestos.....	261

1. AMBIENTE DE DESARROLLO DE LabVIEW.

1.1 OBJETIVO

Estudiar los conceptos fundamentales, la terminología empleada y los métodos básicos de utilización del sistema de desarrollo LabVIEW.

1.2 DESCRIPCIÓN

LabVIEW (*Laboratory Virtual Instrument Engineering Workbench*) es un sistema de desarrollo basado en programación gráfica orientado a desarrollar aplicaciones para instrumentación que integra una serie de librerías para comunicación con instrumentos electrónicos como GPIB, RS232 o RS485 con tarjetas de adquisición de datos, sistemas de adquisición y acondicionamiento como VXI o SCXI, comunicaciones en redes TCP/IP, UDP, o en los estándares de software COM, OLE, DDE, DLL o ActiveX para Windows, así como AppleEvents para MacOS o PIPE para UNIX.

Los programas realizados en LabVIEW se llaman instrumentos virtuales “VIs”, ya que tienen la apariencia de los instrumentos reales, sin embargo, poseen analogías con funciones provenientes de lenguajes de programación convencionales.

Las principales características de los VIs se pueden describir como:

Los VIs contienen una interface interactiva de usuario, la cual se llama panel frontal, ya que simula el panel de un instrumento físico. Se puede entrar datos usando el teclado o el ratón y tener una visualización de los resultados en la pantalla del computador. El Panel Frontal es la interface hombre-máquina de un VI.

Los VIs reciben instrucciones de un diagrama de bloques construido en lenguaje G el cual suministra una solución gráfica a un problema de programación. El diagrama de bloques es el código fuente de un VI.

Los VIs usan una estructura hereditaria y modular que permite realizar programas por niveles o hacer programas con otros programas o subprogramas. Un VI contenido en otro VI es denominado subVI. Todo VI se puede convertir en subVI sin ningún tipo de cambio en su estructura.

Con estas características LabVIEW permite dividir un programa en una serie de tareas las cuales son divisibles nuevamente hasta que una aplicación complicada se convierte en una serie de subtareas simples. Todos los anteriores conceptos están de acuerdo con las concepciones modernas de la programación modular.

Además LabVIEW puede ser usado con poca experiencia en programación pues utiliza metodologías familiares a técnicos, ingenieros, doctores y la comunidad científica en general.

Cada VI de LabVIEW cuenta con dos interfaces: panel frontal y diagrama de bloques. Éstas cuentan con paletas que contienen los objetos necesarios para implementar y desarrollar tareas. La figura 1.1. muestra estas interfaces dentro de un entorno Windows.

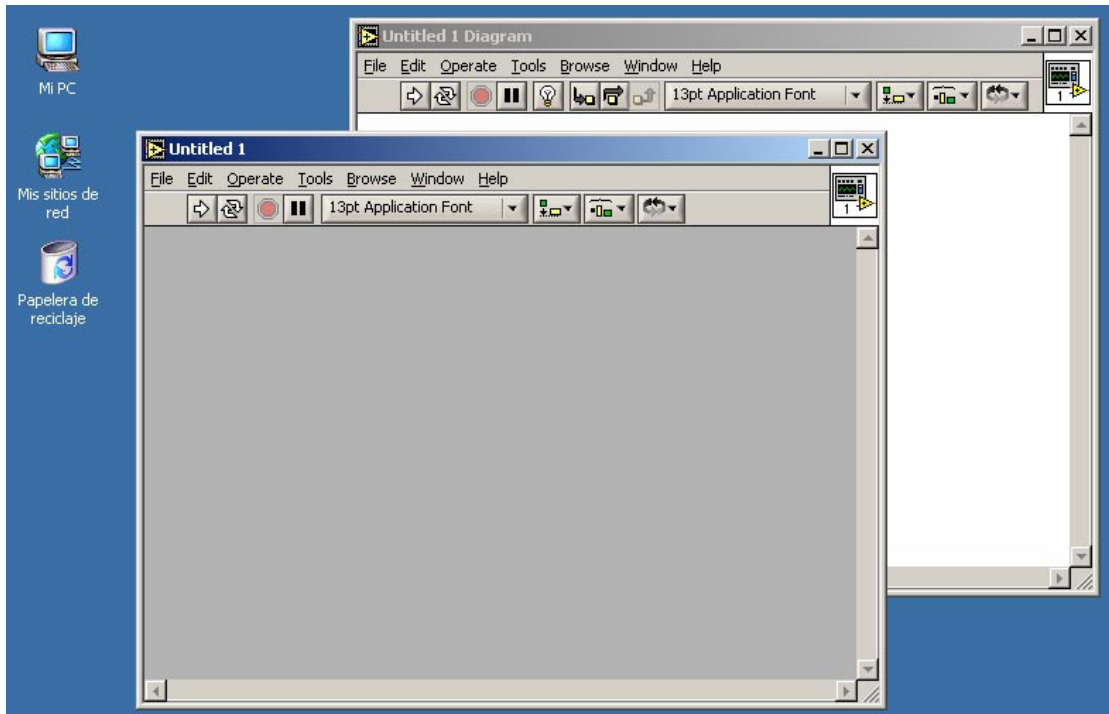


Figura 1.1. Interfaces de un VI.

1.3 PANEL FRONTAL

Es la interface gráfica que simula el panel de un instrumento real, permite la entrada y salida de datos, puede contener pulsadores, perillas, botones, gráficos y en general controles e indicadores. Figura 1.2.

Los controles son objetos que sirven para entrar datos al programa y pueden ser manipulados por el usuario. Los controles son variables de entrada.

Los indicadores sirven para presentar los resultados entregados por el programa y no pueden ser manipulados por el usuario. Los indicadores son variables de salida.

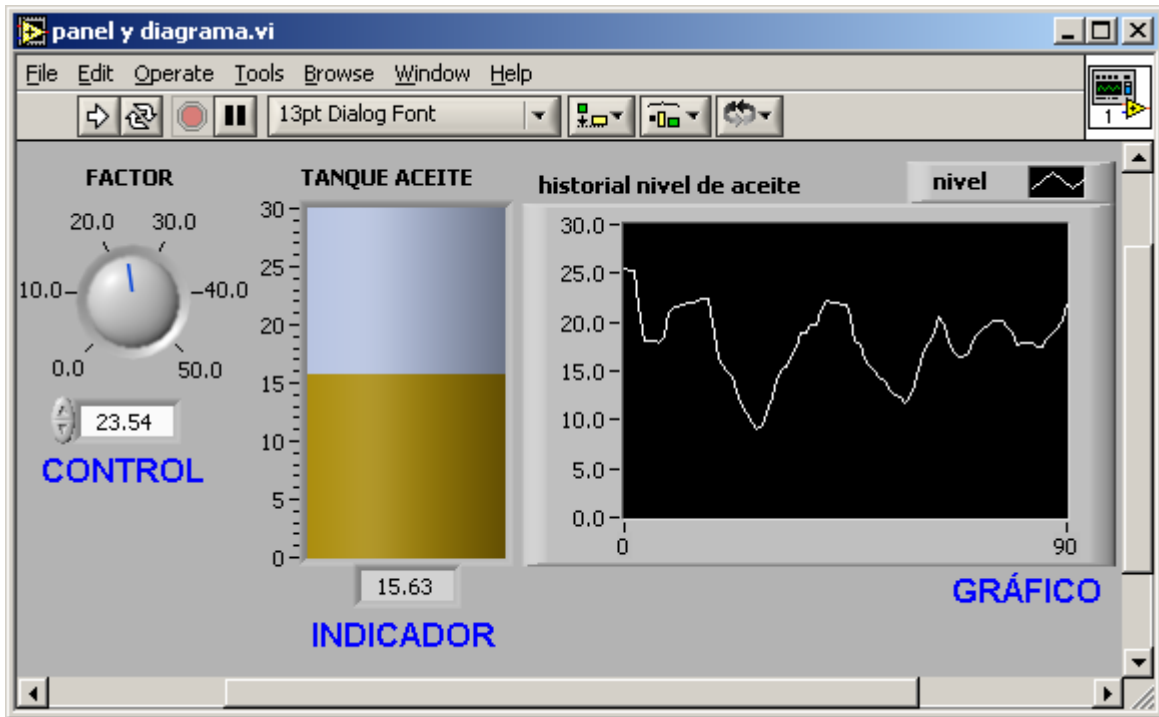


Figura 1.2. Panel Frontal de una aplicación.

1.4 DIAGRAMA DE BLOQUES

El diagrama de bloques contiene el código fuente gráfico del VI, posee funciones y estructuras que relacionan las entradas con las salidas creadas en el panel frontal.

En un diagrama se distinguen: Terminales, que representan los controles e indicadores del panel. Funciones y SubVIs, que realizan tareas específicas. Estructuras y Cables que determinan el flujo de los datos en el programa. En general, cualquiera de estas partes del diagrama de un VI se denomina NODO.

El diagrama de bloques de la figura 1.3 muestra el código fuente correspondiente al panel de la figura 1.2.

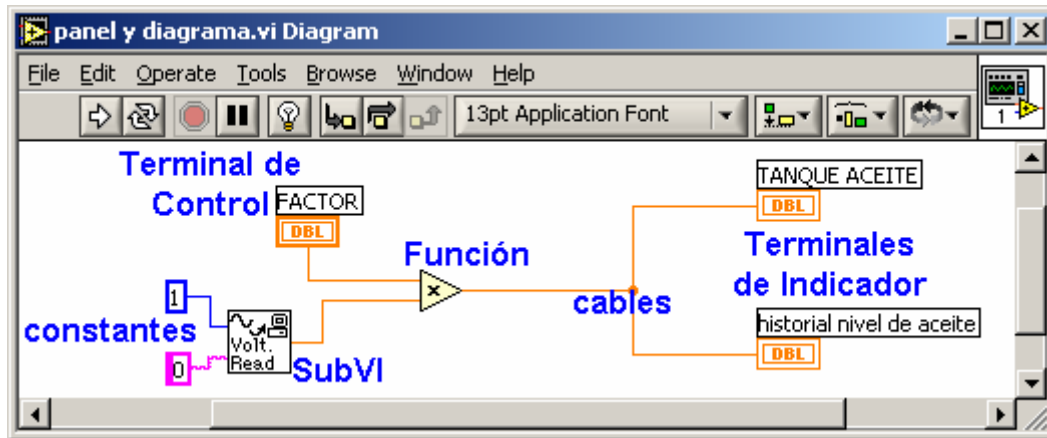


Figura 1.3. Diagrama de Bloques de una aplicación.

1.5 PALETA DE HERRAMIENTAS

Contiene las herramientas necesarias para editar y depurar los objetos tanto del panel frontal como del diagrama de bloques. Figura 1.4.



Figura 1.4. Paleta de Herramientas.



Operación

Asigna valores a los controles del panel frontal, se encuentra

disponible cuando se corre y edita la aplicación. Cuando edita objetos basados en texto o números, cambia el icono del puntero por el que se muestra.



Posición

Selecciona, mueve y redimensiona objetos. La herramienta cambia el icono del puntero cuando pasa por encima de objetos que pueden modificar su tamaño.



Etiquetado

Crea y edita textos tanto en el panel frontal como en el diagrama de bloques. El icono del puntero asociado a esta herramienta es el que se muestra.



Cableado

Se utiliza para generar la estructura lógica de eventos mediante la conexión de los terminales de cada objeto. Los cables determinan el flujo de los datos.



Menú desplegable

Permite obtener el menú de opciones de un objeto. Esta misma función se puede realizar haciendo un clic derecho del ratón sobre el objeto.



Desplazamiento

Mueve todos los objetos dentro de la ventana activa.





Punto de quiebre

Detiene la ejecución del programa en el punto del diagrama donde se ponga. Se utiliza con fines de depuración.



Punto de prueba

Se pone sobre algún cable de conexión para verificar de forma temporal el valor que fluye a través de éste.



Capturar color

Obtiene el color del objeto que se señale.



Colorear

Cambia los colores de objetos y fondos.

1.6 PALETA DE CONTROLES

Se utiliza únicamente en el panel frontal y contiene los objetos necesarios para crear una interface de entrada y salida de datos (controles e indicadores).

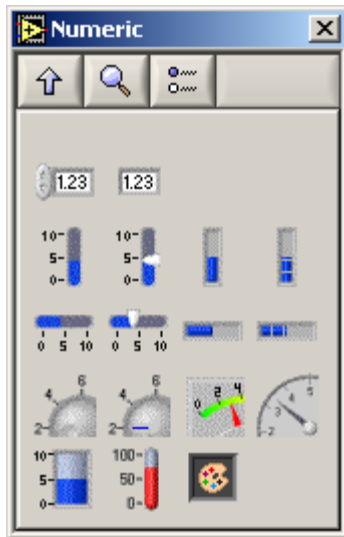
Esta paleta se obtiene de la barra de menús con la opción **Window>>Show Controls Palette**, o haciendo clic derecho sobre el panel frontal. La paleta de controles se muestra en la figura 1.5.



Figura 1.5. Paleta de Controles.

Existen submenús correspondientes a *toolkits* que sólo aparecen cuando se han instalado. Los *toolkits* son herramientas adicionales de software con fines específicos y especializados que se suministran como productos por separado.

Cada submenú de la paleta contiene controles e indicadores respectivos de una clase de objetos. Por ejemplo, la figura 1.5b muestra los submenús más importantes:



Controles e Indicadores Numéricos.



Controles e Indicadores Booleanos



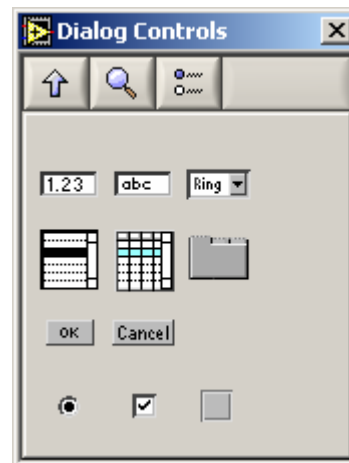
Controles e Indicadores *String* y *Path*.



Controles e Indicadores *Array&Cluster*.



Gráficas.



Controles de Diálogo.

Figura 1.5b. Submenús de la paleta de controles.

En la parte superior de las paletas hay tres herramientas como se muestra en la figura 1.6 que de izquierda a derecha sirven para: subir un nivel de submenú, buscar una función en la paleta y personalizar el contenido de la paleta respectivamente.

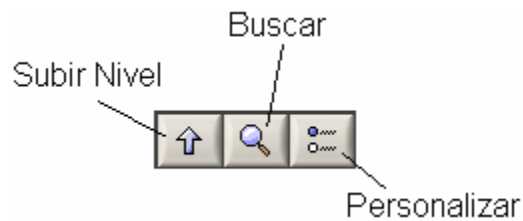


Figura 1.6. Herramientas de las paletas y submenús.

1.7 PALETA DE FUNCIONES

Se usa únicamente en el diagrama de bloques y contiene todos los objetos para crear y editar el código fuente.

Esta paleta se obtiene de la barra de menús con la opción **Window>>Show Functions Palette**, o haciendo clic derecho en el diagrama. La paleta de controles se muestra en la figura 1.7.

Esta paleta también puede ser personalizada haciendo uso de la herramienta mostrada en la figura 1.6.

Además, los submenús correspondientes a los *toolkits* no estarán presentes hasta que se hayan adquirido e instalado.

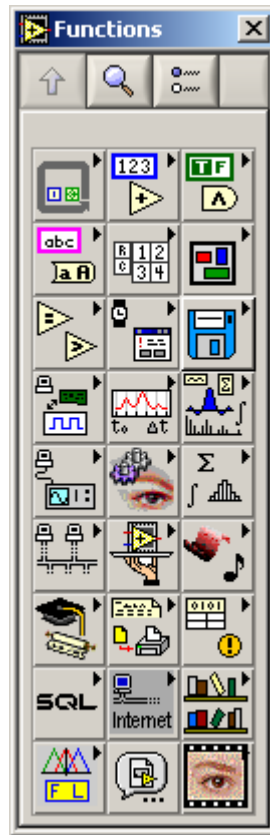
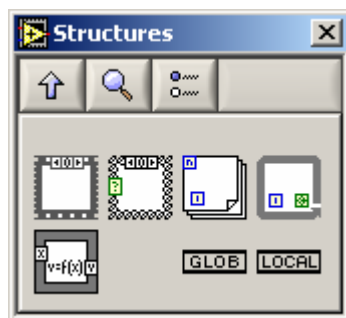


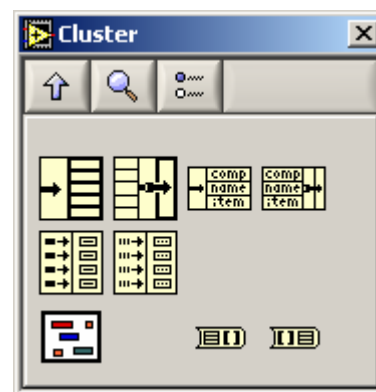
Figura 1.7. Paleta de Funciones.

Cada submenú de la paleta contiene funciones para distintas tareas.

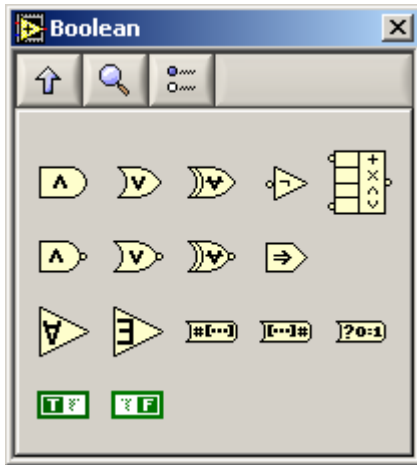
La figura 1.7b muestra los submenús más importantes:



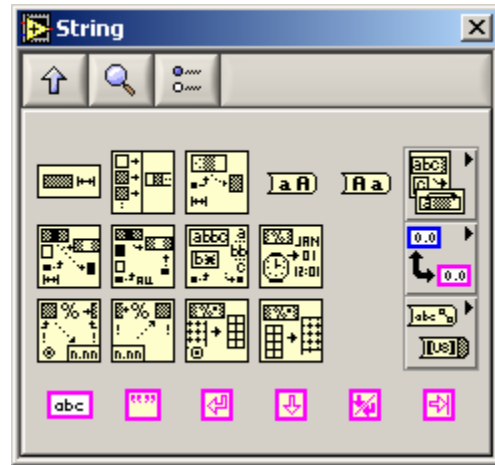
Estructuras.



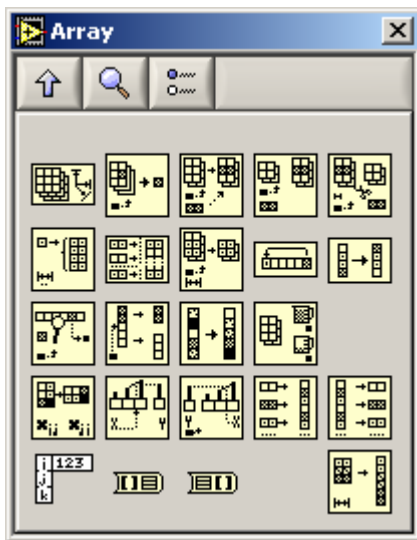
Funciones para Clusters.



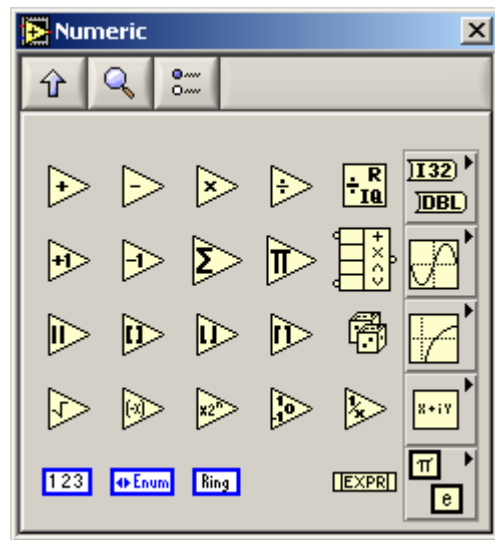
Funciones Booleanas.



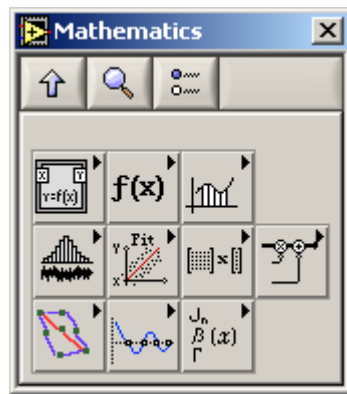
Funciones de Cadena.



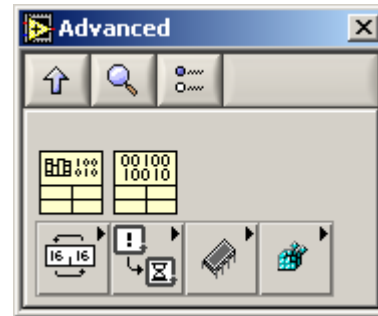
Funciones de Arreglos.



Funciones Numéricas.



Matemáticas.



Funciones Avanzadas.

Figura 1.7b. Submenús de la paleta de funciones.

1.8 BARRAS DE LABVIEW

1.8.1 Barra de Menús

La barra de menús que presenta LabVIEW en la parte superior de un VI contiene diversos menús desplegables que cumplen diferentes funciones:



Figura 1.8. Barra de menús.

Menú *File*: Contiene entre otras las opciones convencionales del ambiente Windows (abrir, cerrar, guardar, imprimir, salir).

New VI	Ctrl+N	Crea un nuevo VI
New...		Crear un nuevo componente de LabVIEW
Open...	Ctrl+O	Abre un VI existente
Close	Ctrl+W	Cierra la ventana activa
Close All		Cierra todas las ventanas
Save	Ctrl+S	Guarda el VI activo
Save As...		Guarda el VI activo con otro nombre
Save All		Guarda todos los VIs abiertos
Save with Options...		Guarda el VI activo con opciones
Revert...		Devuelve un VI a la última versión guardada
Page Setup...		Configura la página para impresión
Print...		Ejecuta el asistente para impresión
Print Window...	Ctrl+P	Imprime la ventana activa
VI Properties...	Ctrl+I	Abre las propiedades del VI activo
Recently Opened Files	▶	Muestra un menú con los VIs abiertos recientemente
Exit	Ctrl+Q	Sale de LabVIEW

Figura 1.9. Menú *FILE*.

Menú *Edit*: Permite realizar las acciones de edición como copiar, cortar, pegar, deshacer, rehacer, borrar, importar y manipular componentes de LabVIEW.

Undo	Ctrl+Z	Deshace las últimas acciones
Redo	Ctrl+Shift+Z	Rehace las acciones deshechas
Cut	Ctrl+X	Corta los objetos seleccionados y los envía al portapapeles
Copy	Ctrl+C	Copia los objetos seleccionados y los envía al portapapeles
Paste	Ctrl+V	Pega el contenido del portapapeles en la ventana activa
Clear		Borra los objetos seleccionados
Find...	Ctrl+F	Ejecuta el asistente de búsqueda de objetos y componentes
Show Search Results	Ctrl+Shift+F	Muestra los resultados de la búsqueda
Customize Control...		Ejecuta el editor de controles e indicadores
Scale Object With Panel		Escala los objetos al redimensionar el panel frontal
Set Tabbing Order...		Permite definir el orden de los objetos del panel frontal
Import Picture from File...		Pone en el portapapeles un archivo gráfico.
Remove Broken Wires	Ctrl+B	Remueve los cables con errores
Create SubVI		Crea un subVI del diagrama de bloques seleccionado
Run-Time Menu...		Permite crear un menú personalizado para el VI activo

Figura 1.10. Menú *EDIT*.

Menú *Operate*: Contiene las opciones necesarias para controlar la operación de los VIs.

Run	Ctrl+R	Ejecuta el VI activo
Stop	Ctrl+.	Aborta la ejecución del VI activo
Suspend when Called		Suspende la ejecución de un VI cuando es invocado
Print at Completion		Imprime el panel frontal cuando el VI se termina de ejecutar
Log at Completion		Genera un registro cuando el VI se termina de ejecutar
Data Logging		Muestra las opciones del generador de registros
Make Current Values Default		Predetermina los valores actuales de los objetos
Reinitialize All To Default		Hace que los objetos regresen a su valor predeterminado
Change to Edit Mode	Ctrl+M	Conmuta entre los modos edición y ejecución

Figura 1.11. Menú *OPERATE*.

Menú *Tools*: Contiene herramientas para la configuración de LabVIEW, de los proyectos y de los SubVIs.

Measurement & Automation Explorer...		Accede al Explorador de componentes de NI
Instrumentation		Permite buscar o importar <i>drivers</i> para instrumentos
Data Acquisition		Configura los canales y ejecuta el asistente de soluciones
Compare		Compara el código y la jerarquías de VIs
Source Code Control		Accede las funciones de control de código fuente
VI Revision History	Ctrl+Y	Permite documentar los cambios del VI activo
User Name...		Permite cambiar el nombre de usuario actual de LabVIEW
Build Application or Shared Library (DLL)...		Ejecuta las opciones de compilación
VI Library Manager...		Accede al explorador de librerías de VIs
Edit VI Library...		Edita o crea una librería de VIs LLB
Database Browser...		Navegador de Bases de Datos del SQL <i>toolkit</i>
Fuzzy Logic Controller Design...		Diseñador para control difuso del <i>FuzzyLogic toolkit</i>
Internet Toolkit		Herramientas del Internet <i>toolkit</i>
Web Publishing Tool...		Herramientas de publicación en Web
Advanced		Herramientas Avanzadas de LabVIEW
Options...		Accede las opciones de configuración de LabVIEW

Figura 1.12. Menú *TOOLS*.

Menú *Browse*: Contiene opciones que permiten observar aspectos del VI activo y toda su jerarquía.

Show <u>V</u> I Hierarchy	Muestra la ventana de Jerarquía de un VI
This VI's Callers	Muestra una lista de VIs que llaman al VI activo como un subVI
This VI's SubVIs	Muestra una lista de los subVIs que componen el VI activo
Unopened SubVIs	Muestra una lista de los subVIs del VI activo que no están abiertos
Unopened Type Defs	Muestra una lista de tipos definidos del VI activo que no están abiertos
Breakpoints	Busca puntos de quiebre en el VI actual

Figura 1.13. Menú *BROWSE*.

Menú *Window*: Permite configurar la apariencia de las paletas y ventanas.

Show Panel	Ctrl+E	Conmuta entre el panel frontal y el diagrama de bloques
Show Functions Palette		Muestra la paleta de funciones en el diagrama o controles en el panel
Show Tools Palette		Muestra la paleta de herramientas
Show Clipboard		Muestra el contenido del portapapeles
Show Error List	Ctrl+L	Muestra la lista de errores del VI activo
Tile Left and Right	Ctrl+T	Arregla las ventanas abiertas a la derecha e izquierda
Tile Up and Down		Arregla las ventanas abiertas arriba y abajo
Full Size	Ctrl+/	Maximiza la ventana activa
Untitled 1		Muestra una lista de las ventanas abiertas
✓ Untitled 1 Diagram		

Figura 1.14. Menú *WINDOW*.

Menú *Help*: Presenta la ayuda en línea, los manuales de referencia, la documentación impresa, los recursos de web, enlaces en Internet y los archivos de ayuda de cada *toolkit* instalado.

✓ Show Context Help Ctrl+H Lock Context Help Ctrl+Shift+L	Muestra la ventana de ayuda básica de VIs funciones y controles Fija el contenido de la ventana de ayuda básica
Contents and Index Ctrl+? View Printed Manuals... Help for This VI	Accede a la documentación electrónica completa de LabVIEW Carga la versión PDF del conjunto de manuales de LabVIEW Abre el archivo de ayuda HLP del VI activo
Examples... Web Resources Explain Error...	Accede al buscador de ejemplos de LabVIEW Permite acceder los recursos de web de National Instruments Muestra la información de referencia de un mensaje de error
FlexMotion VI Online Help... IMAQ Vision... Internet Toolkit... NI-IMAQ VIs... SQL Toolkit... ValueMotion VI Online Help...	Ayuda del <i>toolkit FlexMotion</i> Ayuda del <i>toolkit</i> de procesamiento de imágenes Ayuda del <i>toolkit</i> para desarrollo en Internet Ayuda de los VIs de adquisición de imagen Ayuda del <i>toolkit</i> SQL Ayuda del <i>toolkit ValueMotion</i>
About LabVIEW...	Muestra entre otros la versión y el número de serie de LabVIEW

Figura 1.15. Menú *HELP*.

1.8.2 Barra de Herramientas del Panel Frontal

La barra de herramientas permite la ejecución, depuración y organización de los VIs.

Esta localizada debajo de la barra de menús y se muestra en la figura 1.16.



Figura 1.16. Barra de herramientas.



Ejecutar

Botón para correr la aplicación, su forma varía de acuerdo al nivel jerárquico donde se ejecute.



Botón Ejecutar cuando se corre una aplicación a nivel superior.



Botón Ejecutar cuando se corre una aplicación a nivel inferior



Botón Ejecutar cuando hay problemas con el código que impiden correr la aplicación. Haciendo clic sobre él se pueden localizar las causas del problema.



Ejecutar continuamente

Botón para ejecutar la aplicación repetidamente hasta que sea presionado de nuevo o se presione abortar.



Apariencia del botón cuando un VI se está ejecutando continuamente.



Abortar ejecución

Aborta la ejecución de una aplicación.



Apariencia del botón Abortar cuando un VI se está ejecutando.

**Pausar**

Botón para pausar la aplicación. Si se presiona nuevamente la ejecución continuará.



Apariencia del botón Pausar cuando un VI está pausando.

**Fuentes**

Menú para seleccionar tipos de letra en objetos y textos.

**Alineación**

Permite alinear objetos tanto en el panel frontal como en el diagrama.

**Distribución**

Permite distribuir uniformemente objetos tanto en el panel frontal como en el diagrama.

**Reorganización**

Permite reorganizar la posición de los objetos tanto en el panel frontal como en el diagrama.

1.8.3 Barra de Herramientas del Diagrama

Esta barra contiene además de los mismos botones de la barra del panel frontal las herramientas de depuración.

La figura 1.17 muestra la barra de herramientas del diagrama de bloques.

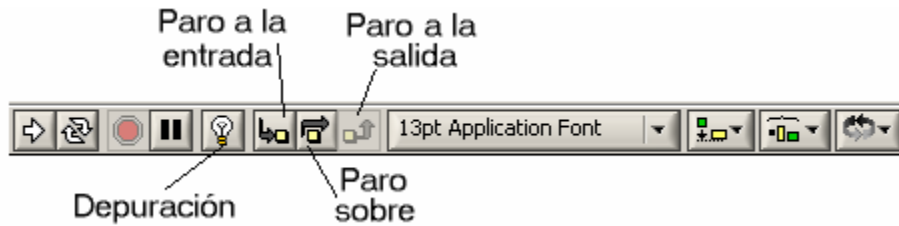


Figura 1.17. Barra de herramientas del diagrama.



Depuración

Botón para observar el flujo de datos en el diagrama de bloques.



Apariencia del botón cuando un VI está en modo de depuración.



Paro a la entrada

En modo depuración sirve para generar un paro a la entrada de un nodo.



Paro sobre

En modo depuración sirve para saltar un nodo.



Paro a la salida

En modo depuración sirve para salir de un nodo.



Advertencia

Si está habilitado aparece al lado izquierdo de la barra de herramientas indicando que existen observaciones al código fuente pero que no impiden la ejecución del VI.



Entrar

Botón que aparece cuando se editan textos o números y sirve para dar entrada a los datos. Esta función se puede cumplir también con la tecla <INTRO>.

1.9 TÉCNICAS DE CABLEADO

El código fuente de un programa en LabVIEW se define con la interconexión gráfica de los objetos que lo componen. A continuación se muestra las técnicas de alambrado para la construcción de un VI.

1. Para comenzar a alambrear entre dos terminales se debe seleccionar la herramienta de cableado de la paleta de herramientas. Para enlazar los objetos del diagrama de bloques se coloca el puntero sobre el origen, cuando el origen parpadee se hace un clic sobre él, luego se lleva el puntero hasta el destino y cuando parpadee se hace otro clic sobre él.

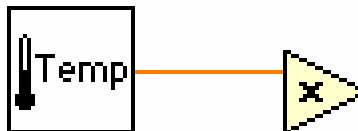


Figura 1.18. Objetos cableados.

2. Si se realiza un doble clic sobre el elemento origen sin llegar hasta el elemento destino el cable será temporal.

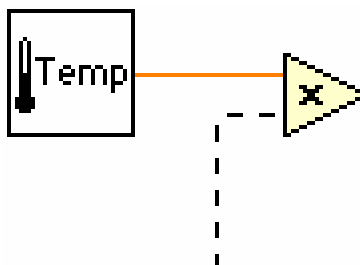


Figura 1.19. Cable temporal.

3. Para cambiar la dirección de un cable mientras se construye se utiliza la barra espaciadora.

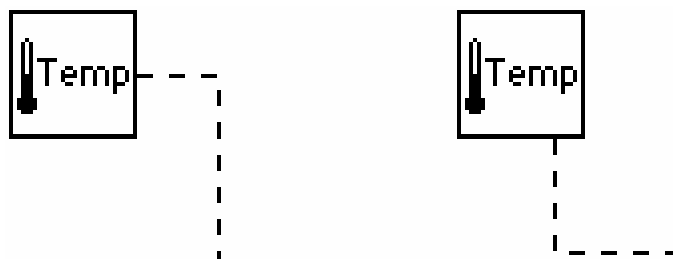


Figura 1.20. Cambio de dirección en cableado.

4. Para remover los cables malos, utilice <control+B>, o seleccione el menú **Edit>>Remove Broken Wires**.
5. Para resaltar porciones del cableado:

- Con un solo clic

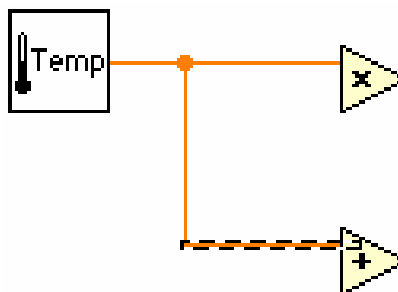


Figura 1.21. Selección simple de cableado.

- Con doble clic

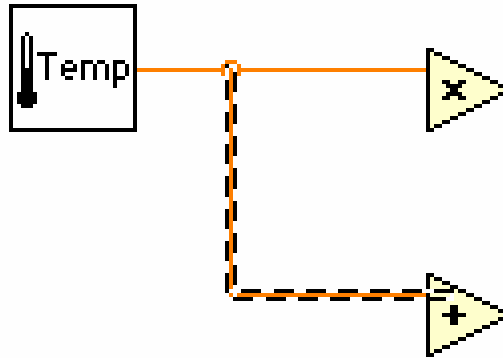


Figura 1.22. Selección de un tramo de cableado.

- Con triple clic

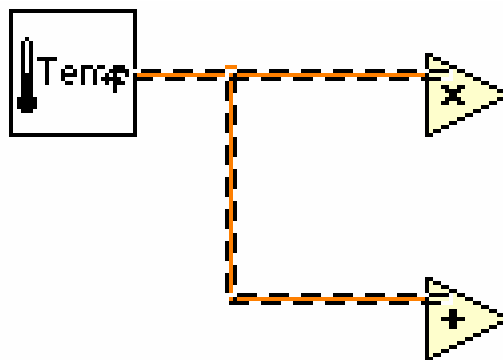


Figura 1.23. Selección de todo un cable.

LabVIEW 6i posee la capacidad de alambrar automáticamente los objetos a medida que se van colocando, conectando los terminales que mejor concuerden con el tipo de dato que se este manejando. Se puede activar o desactivar el alambrado automático utilizando la barra espaciadora mientras se acomodan objetos en el diagrama.

1.10 TÉCNICAS DE EDICIÓN DE DIAGRAMAS

Cada objeto en LabVIEW posee un menú de opción que permite configurar su funcionamiento facilitando las tareas de edición.

Este menú se puede acceder haciendo clic derecho sobre el objeto, sea un terminal, una estructura o un subVI, como se muestra en la figura 1.24.

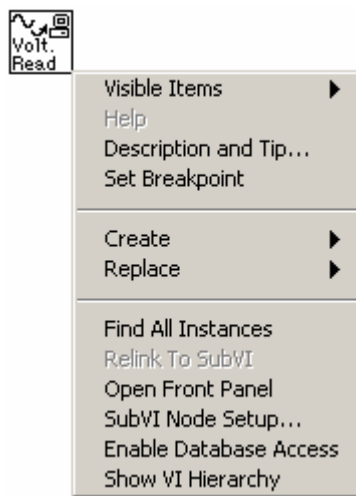


Figura 1.24. Menú de opciones de un subVI.

Además existen métodos abreviados y herramientas para realizar tareas específicas comunes en la edición de VIs.

Algunas de estos métodos y herramientas son:

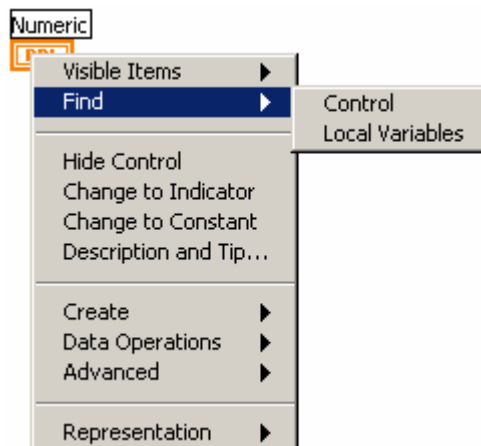
1. Cuando se necesite crear una constante, un control o un indicador desde el diagrama de bloques para un objeto, función o nodo, hacer clic sobre el mismo y seleccionarlo del menú que se presenta.

2. Para visualizar la lista de errores cometidos durante la edición del diagrama, seleccionar el botón **List Error** que aparece en la barra de menús reemplazando el botón de ejecución.
3. Para rotar rápidamente entre los elementos de la paleta de herramientas se puede utilizar la tecla <Tab>.
4. Para rotar entre la herramienta de posición y la herramienta de operación en el panel frontal, utilizar la barra espaciadora. En el diagrama la barra espaciadora rota entre la herramienta de posición y la de cableado.
5. Para duplicar objetos usando la herramienta de posición, se debe señalar el objeto y oprimir la tecla <control> para arrastrarlo hasta el sitio deseado.
6. Para reemplazar un objeto, se hace clic con el botón derecho sobre el mismo y del menú que aparece seleccionar **Replace**.

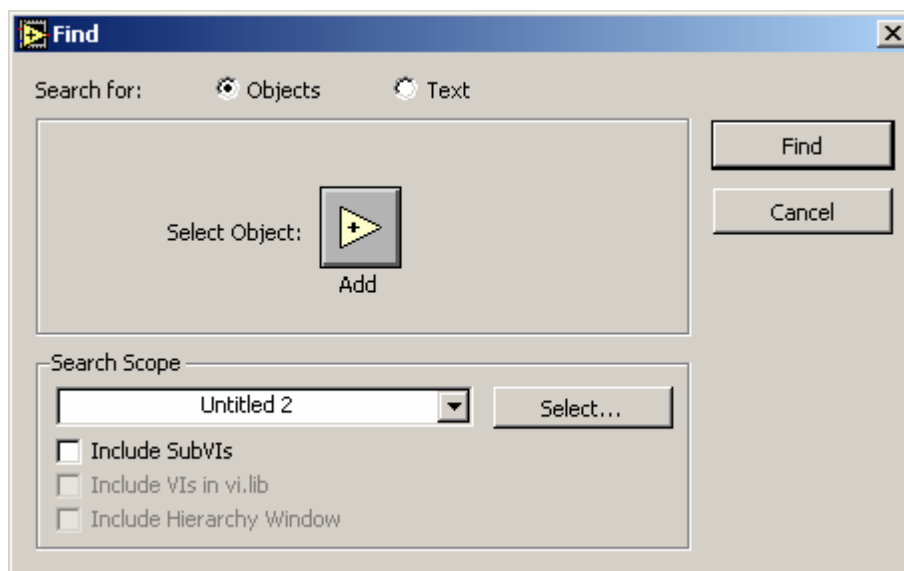
1.11 TÉCNICAS DE NAVEGACIÓN

En la edición de los VIs es necesario localizar objetos como textos, gráficos, funciones, subVIs entre otros. Para tal efecto es recomendable utilizar las siguientes técnicas de navegación:

1. Para encontrar un terminal, un control, una variable local o un atributo de nodo asociado con un objeto, hacer clic derecho sobre el control y seleccionar **Find**.

Figura 1.25. Opción *Find*.

2. Para encontrar texto y objetos en memoria, seleccionar **Edit>>Find** o hacer <Control + F>.

Figura 1.26. Menú *Find*.

3. Para abrir un subVI desde la ventana de diagramación, hacer doble clic sobre él.

4. Para guardar el trabajo elija la opción del menú **File>>Save**. Si no se especifica una extensión, LabVIEW añadirá “.vi”.

1.12 TIPOS DE CABLES

En lenguaje G, es posible identificar los tipos de datos por la forma, tamaño y color de sus terminales y cables.

Entre controles e indicadores se pueden manejar datos escalares y arreglos de una o más dimensiones.

Cuando el dato es escalar, la característica del cableado se muestra en la figura 1.27.



Figura 1.27. Cable con dato escalar.

Se observa la diferencia de contorno entre un indicador y un control.

Cuando los datos pertenecen a un arreglo de una dimensión, el cableado se torna un poco más grueso.



Figura 1.28. Cable con dato arreglo.

Para arreglos bidimensionales, el cable es doble.



Figura 1.29. Cable con arreglo bidimensional.

Cada control e indicador numérico encierra el tipo de dato que tiene asignado. Por ejemplo DBL significa que es de precisión doble.

El color del cable es indicativo del tipo de dato. Por ejemplo Naranja es para números de punto flotante, Azul para enteros, Verde para booleanos, Fucsia para cadenas, etc.

1.13 TIPOS DE DATOS NUMÉRICOS

Al igual que en otros lenguajes de programación los datos numéricos son de diferentes tipos según su naturaleza y el tamaño que ocupan en memoria.

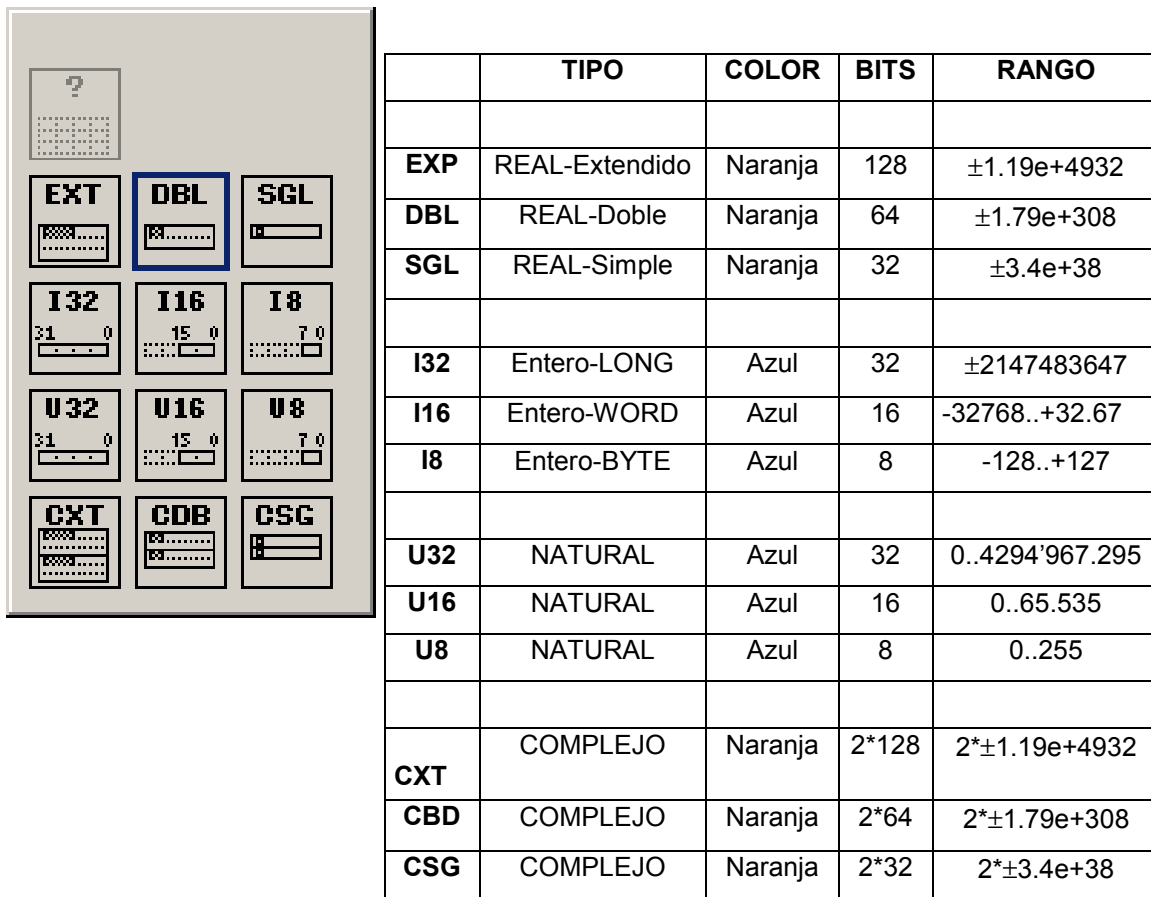


Figura 1.30. Representaciones Numéricas.

1.14 TIPOS DE TERMINALES

Todos los controles e indicadores que sean ubicados en el panel frontal de LabVIEW generarán un terminal en la ventana de diagramación.

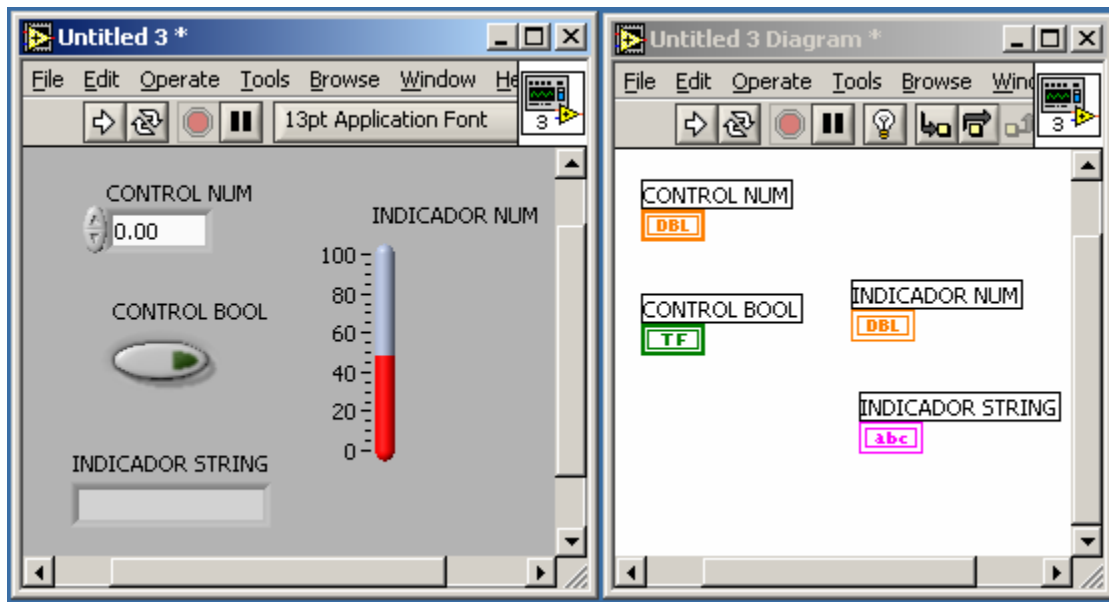


Figura 1.31. Terminales.

Los terminales son objetos del diagrama de bloques que representan un control o un indicador del panel frontal. Toman el color respectivo de la variable que manejan. Por medio de ellos se obtienen los datos de los controles y se envían datos a los indicadores. Un terminal de control se diferencia de uno de indicador en que los primeros poseen un borde doble mientras los segundos uno sencillo.

1.15 AYUDAS DE DEPURACIÓN

Cuando se ejecuta un VI es importante visualizar los resultados intermedios que se están generando. Para conseguir esto se utiliza un nodo de prueba o **Probe** que es similar a un indicador pero temporal y en una ventana flotante.

Haciendo clic con el botón derecho sobre el cable y seleccionando **Probe**, aparece la ventana como se indica en la figura 1.32.

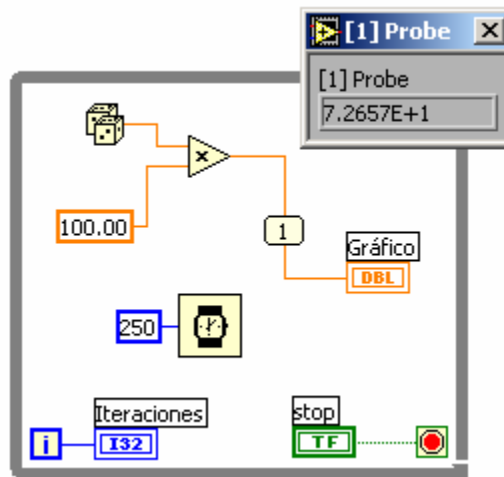


Figura 1.32. Nodo de Prueba.

El número 1 que aparece sobre el cable indica el número de la ventana flotante *Probe*, donde se está mostrando temporalmente el dato que pasa por el cable. En este caso, el dato del indicador llamado “Gráfico”.

1.16 AMBIENTE DE DESARROLLO INTEGRADO DE LABVIEW

En estas páginas se desarrollarán paso por paso actividades orientadas al entendimiento del IDE (*Integrated Development Environment*) de LabVIEW y a la familiarización con la terminología y técnicas de programación del sistema de desarrollo.

1.16.1 Ejecución de LabVIEW

La ejecución de LabVIEW, depende del sistema operativo en que se esté trabajando y de la forma en como haya sido instalado. Generalmente la instalación permite iniciar el programa de una de las siguientes formas:

Windows 95,98,Me, NT,2000	De no encontrar un acceso directo a LabVIEW en el escritorio, puede buscarlo en el menú de inicio por: <i>Programas >> National Instruments >> LabVIEW 6 >> LabVIEW.</i>
MacOS 68x, PowerPC	Debe existir un acceso directo en la “Lanzadera” o “Launcher”. De lo contrario se debe buscar el archivo ejecutable LabVIEW en la carpeta LabVIEW.
SunOS, UNIX, LINUX, HP-UX...	Debe ejecutar un ambiente gráfico como openWin o CDE; o de forma remota con alguna aplicación cliente X. La ruta por defecto asignada al ejecutable durante la instalación es <code>/opt/1v51/LabVIEW</code>

El programa tarda unos segundos en cargar y por defecto comienza su ejecución con la ventana de inicio que se muestra la figura 1.33.

Esta ventana se puede obviar seleccionando la opción “***Do not show this window when launching***”, que aparece en ella misma.



Figura 1.33. Ventana de inicio de LabVIEW.

Las opciones de esta ventana de inicio son:

- New VI** Permite crear un nuevo VI. Si se hace clic en su menú desplegable permite crear algún otro nuevo componente de LabVIEW.
- Open VI** Abre un VI o un componente ya existente. Su menú desplegable muestra una lista de los componentes abiertos recientemente.
- DAQ Solutions** Permite acceder a la configuración de canales o al asistente de soluciones de adquisición de datos.
- Search Examples** Permite buscar ejemplos tanto en la base de datos instalada como en los recursos disponibles en Internet.
- LabVIEW Tutorial** Tutorial en línea de LabVIEW.

Exit Salir de LabVIEW

El recuadro **Quick Tip**, muestra sugerencias cortas de gran utilidad en el desarrollo de aplicaciones. Para mostrar otras sugerencias se debe presionar **NEXT**.

Durante la ejecución de LabVIEW se puede regresar a esta ventana cuando se cierra todos los VIs. Aquí la ventana de inicio podrá ser configurada en modo corto o largo como se muestra en la figura 1.34.

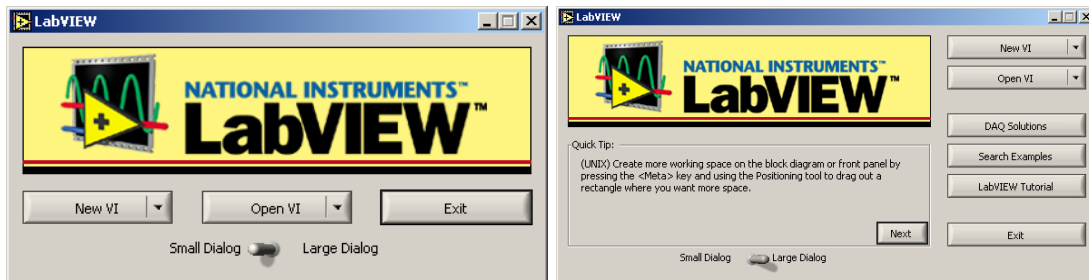


Figura 1.34. Modos corto y largo de la ventana de inicio.

1.16.2 Paletas y ventanas de LabVIEW

La paleta de controles aparece únicamente en el panel frontal. Para su llamado existen dos formas:

Paleta Flotante	Haciendo clic derecho en algún lugar del panel frontal.
Ventana	Utilizando el menú Window>>Show Controls Palette .

Las paletas flotantes desaparecen al hacer un clic en cualquier parte o al seleccionar uno de sus objetos.

Si se desea convertir la paleta flotante en ventana se hace clic sobre la tachuela que aparece en la esquina superior izquierda.

Esto es también válido para la paleta de funciones que se obtiene únicamente en la ventana de diagramación.

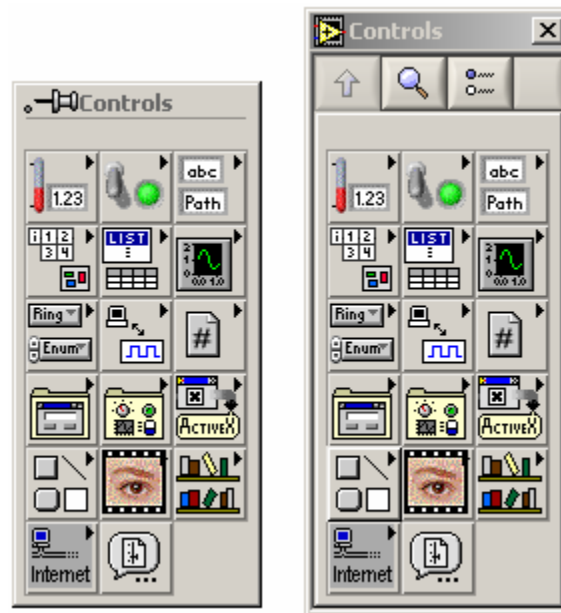


Figura 1.35. Paleta de controles flotante y como ventana.

Para cambiar rápidamente entre las ventanas de diagrama y de panel frontal existen varias técnicas.

Menús En el panel frontal se escoge el menú:
Window>>Show Diagram.

Teclado <CONTROL> + E

Ventanas Utilizando el método de Windows para cambio de ventanas <ALT> + <TAB>.

Barra de Tareas Directamente haciendo clic en la ventana correspondiente de la barra de tareas de Windows.

La paleta de herramientas es común a las dos ventanas y se puede obtener de dos formas:

Ventana	En el menú Window>>Show Tools Palette.
Paleta flotante	Con el método <SHIFT> + <clic derecho>

1.16.3 Ubicación de los objetos

Dentro de las paletas de controles y funciones existen submenús que pueden contener objetos u otros submenús.

Para tomar un objeto basta con hacer un clic sobre él en la paleta correspondiente y luego otro clic en el lugar del panel o del diagrama donde se desee colocar.

Este procedimiento se puede observar en la secuencia de figuras 1.36, 1.37 y 1.38.

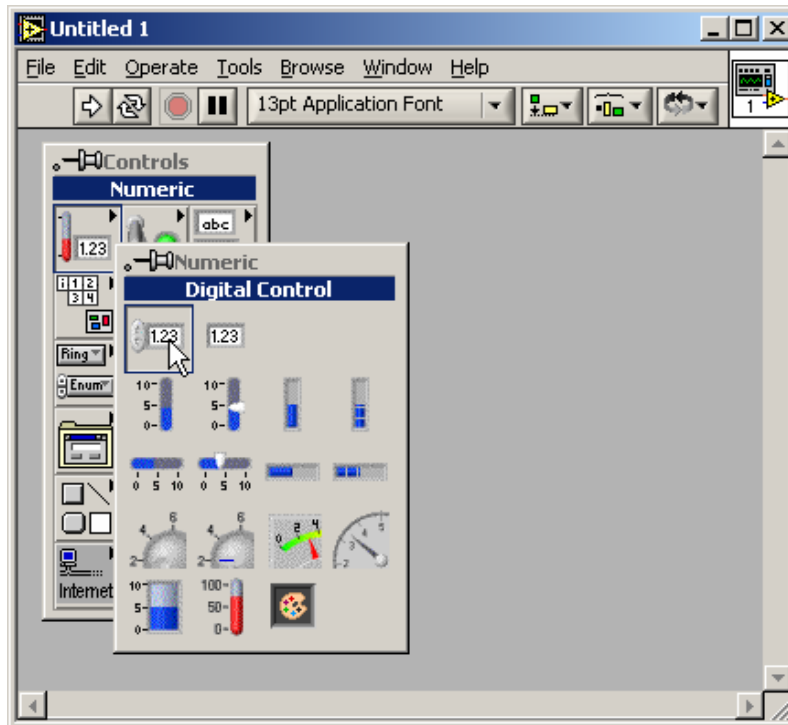


Figura 1.36. Clic sobre el objeto en la paleta de controles.

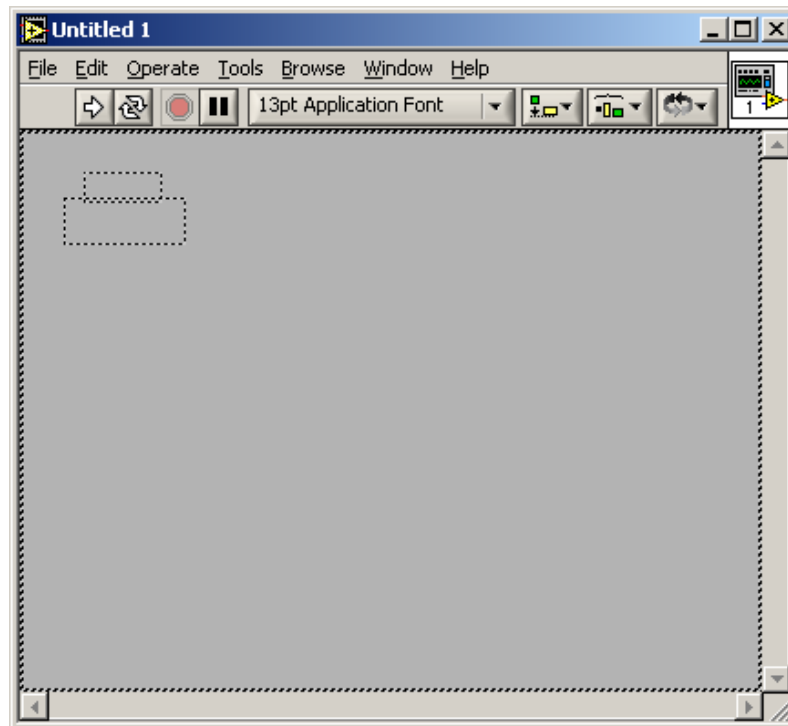


Figura 1.37. Otro clic en el lugar del panel deseado.

Al hacer un nuevo clic el objeto habilita automáticamente la opción de colocar una etiqueta o nombre que lo identifique. Es posible escribir una etiqueta para el objeto o dejar que LabVIEW la asigne por defecto.

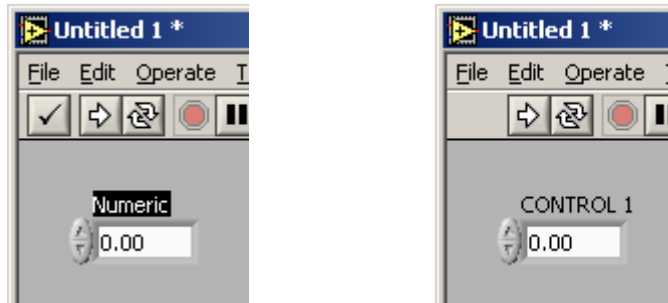


Figura 1.38. Objeto puesto y etiquetado.

A este control le corresponde un terminal en el diagrama de bloques como se observa en la figura 1.39.

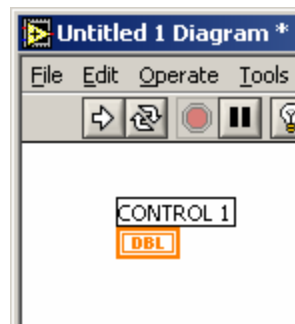


Figura 1.39. Terminal.

La ubicación de objetos en el diagrama tomados de la paleta de funciones es igual a la del panel frontal.

Esto se observa en la secuencia de figuras 1.40 a 1.42.

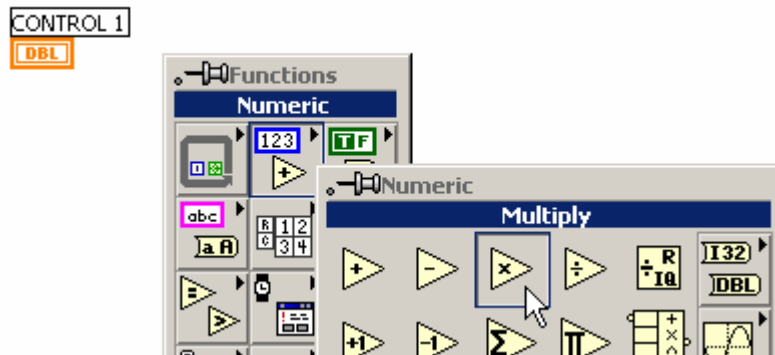


Figura 1.40. Clic sobre el objeto Multiplicación.

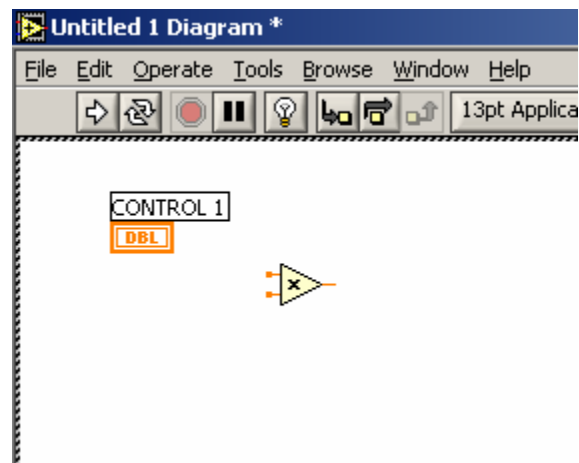


Figura 1.41. Función en el lugar deseado.

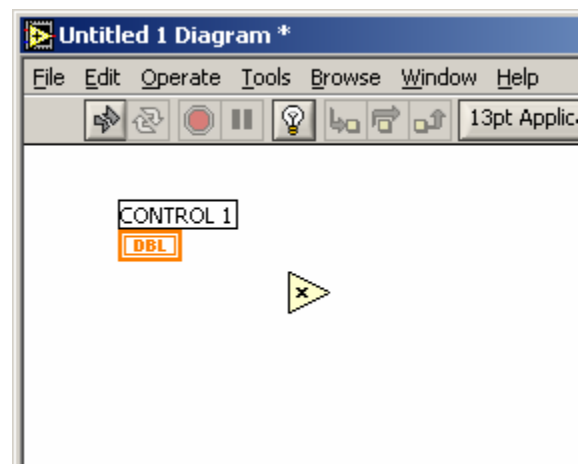


Figura 1.42. Objeto puesto y listo para cablear.

Este es el momento para utilizar las técnicas de cableado ya expuestas. Con la herramienta de cableado, se hace un clic en el punto de origen y luego otro clic en el punto de destino. La figura 1.43 muestra la secuencia de cableado.



Figura 1.43. Secuencia de cableado.

En el diagrama de bloques es posible crear directamente constantes, controles e indicadores a partir de las funciones que requieren entradas y salidas. Este procedimiento se muestra en la figura 1.44.

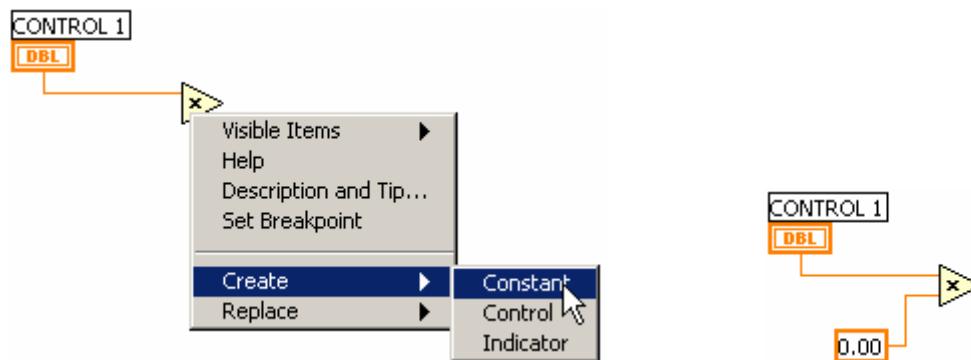


Figura 1.44. Creación de una constante.

Para editar el valor de la constante se utiliza la herramienta de texto y se hace doble clic sobre el texto o número a editar. La secuencia de edición de un valor numérico se observa en la figura 1.45.

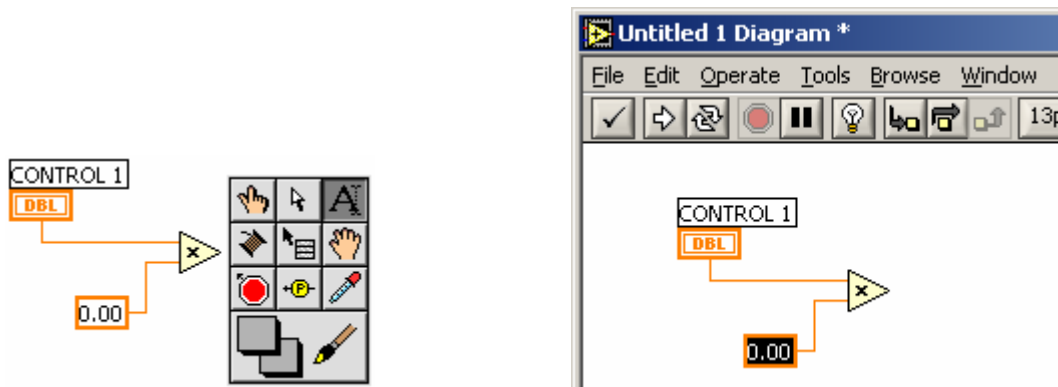


Figura 1.45. Edición de valores numéricos.

Para introducir texto es conveniente utilizar el botón **enter** (Figura 1.46) que se ubica a la izquierda de la barra de herramientas. La función de este botón puede ser cumplida también con la tecla <INTRO> del teclado numérico extendido.

Recuérdese que <INTRO> es diferente de <ENTER> que está en el teclado convencional, aunque en muchos programas y tareas de Windows se pueden usar indistintamente. En LabVIEW <INTRO> y <ENTER> son equivalentes sólo cuando se usan para entrar datos de variables numéricas y de ruta (path).

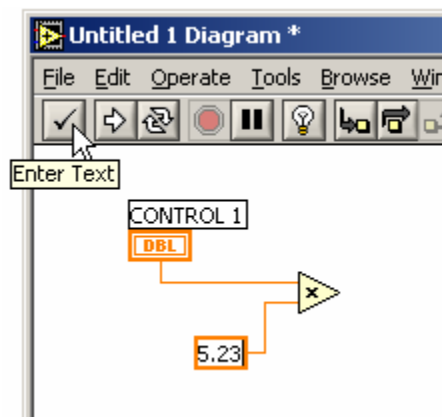


Figura 1.46. Entrada de texto.

El anterior procedimiento es también válido para crear controles e indicadores. El nuevo terminal tendrá su respectivo indicador en el panel frontal. Figura 1.47.

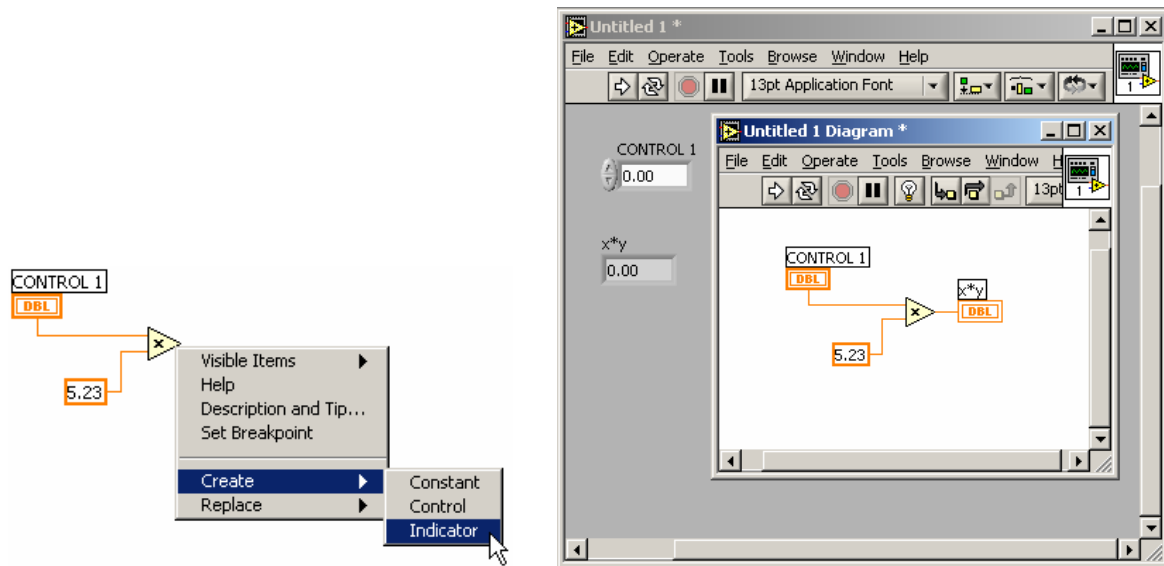


Figura 1.47. Creación de un indicador.

Cuando un control o indicador ha sido creado desde el panel su etiqueta será la que corresponda a la entrada o salida con la que se creó.

1.16.4 Edición de objetos del panel

Para asignar un rótulo a cada objeto se utiliza la herramienta de texto, haciendo doble clic sobre el rótulo (*Label*) existente:

La forma convencional de colocar los rótulos es sin embargo hacerlo en el momento de crear los controles o indicadores.

Si por alguna razón el objeto no posee rótulo se debe seleccionar **Visible Items** >> **Label** del menú del objeto.



Figura 1.48. Asignación de un rótulo.

Para cambiar el color del panel o de cualquier objeto se toma la herramienta de colorear y se aplica color a los objetos que se desee haciendo clic derecho sobre ellos y seleccionando el color deseado.

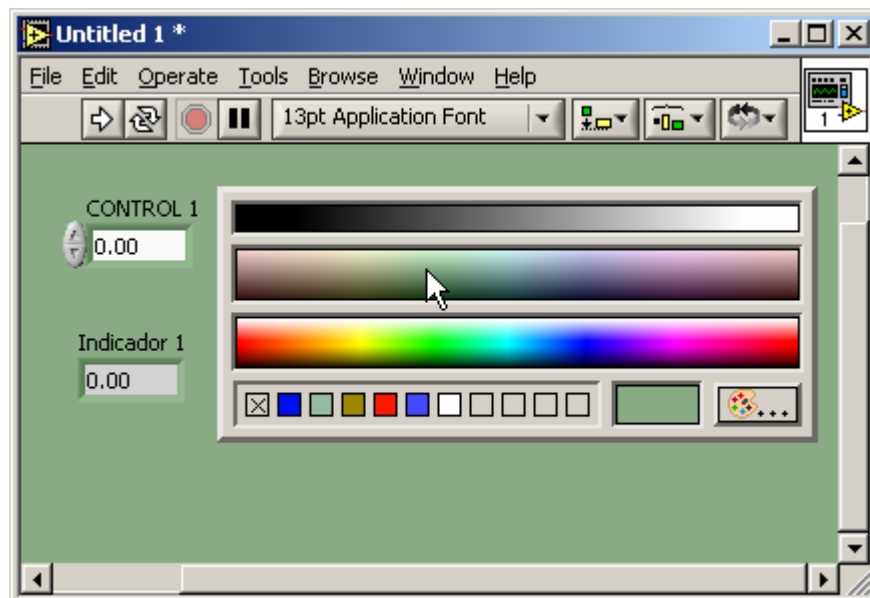


Figura 1.49. Edición del color de fondo del panel frontal.

Para editar cualquier texto se debe seleccionar con la herramienta de texto y luego escoger **Font Dialog** en el menú de fuentes de la barra de herramientas. Esto mostrará el menú de la figura 1.50.

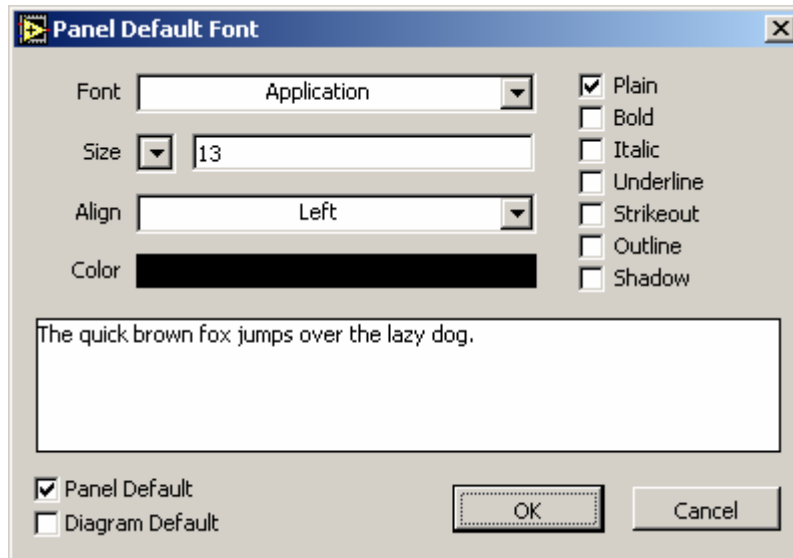


Figura 1.50. Cuadro de diálogo para edición de fuentes.

Otra forma de editar los tipos de letra es haciendo uso directo del menú de fuentes de la barra de herramientas.

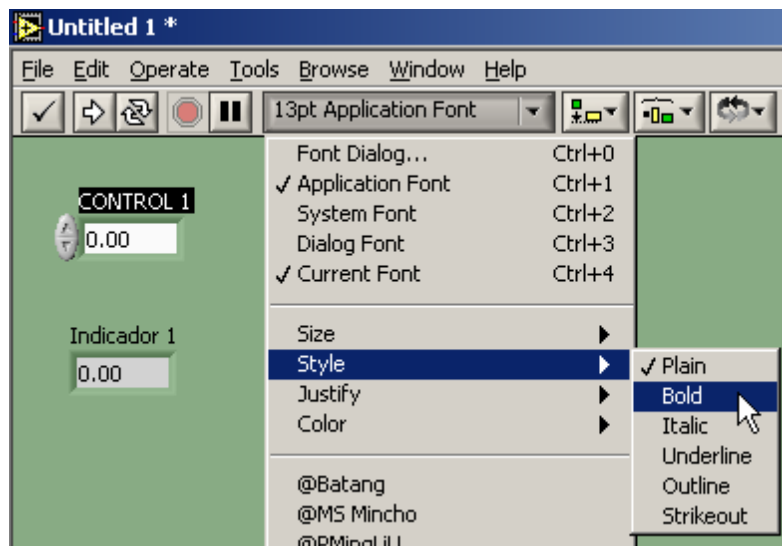


Figura 1.51. Menú de fuentes en la barra de herramientas.

En este menú se puede modificar el tamaño (*Size*), el estilo (*Style*), la alineación

(*Justify*) y el Color (*Color*) de las fuentes asociadas a cada objeto. Como ejemplo se han modificado las fuentes para obtener la figura 1.52.

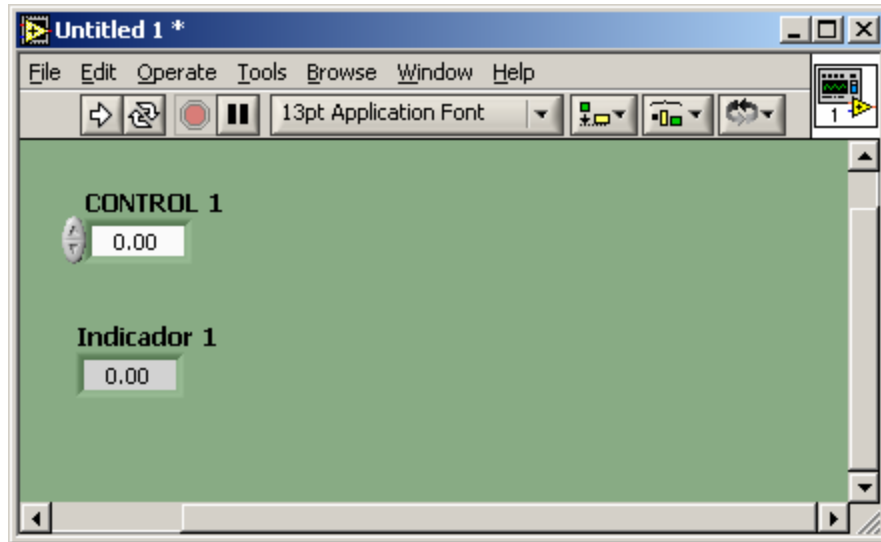


Figura 1.52. Etiquetas en negrita y valores numéricos centrados.

1.16.5 Cambiar el tamaño de los objetos

Para cambiar el tamaño de un objeto se debe acercar la herramienta de posición a uno de los bordos del objeto y arrastrar el ratón hasta la nueva posición deseada.

La selección de la herramienta de texto se muestra en la figura 1.53.

Cuando el objeto está listo para ser cambiado de tamaño, toma el aspecto que se puede observar en el objeto "CONTROL 1" en la figura 1.54.

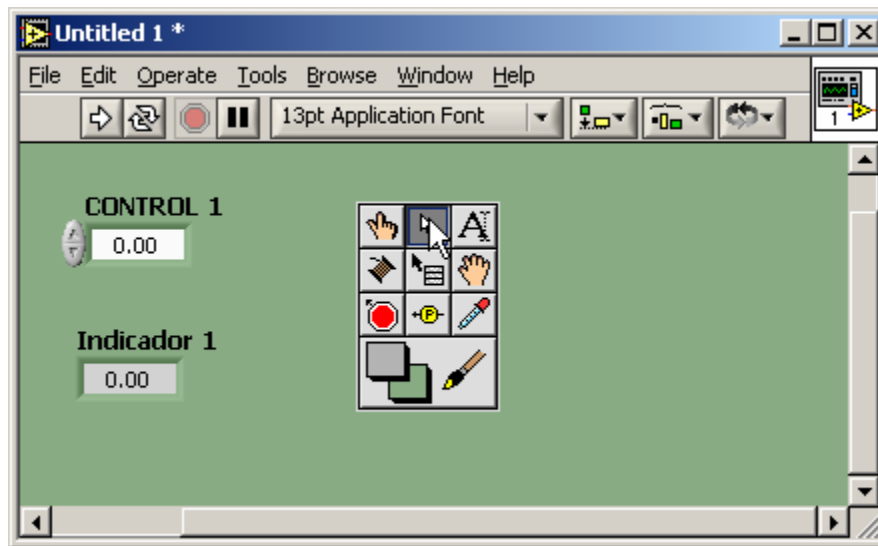


Figura 1.53. Herramienta de posición/tamaño/selección.

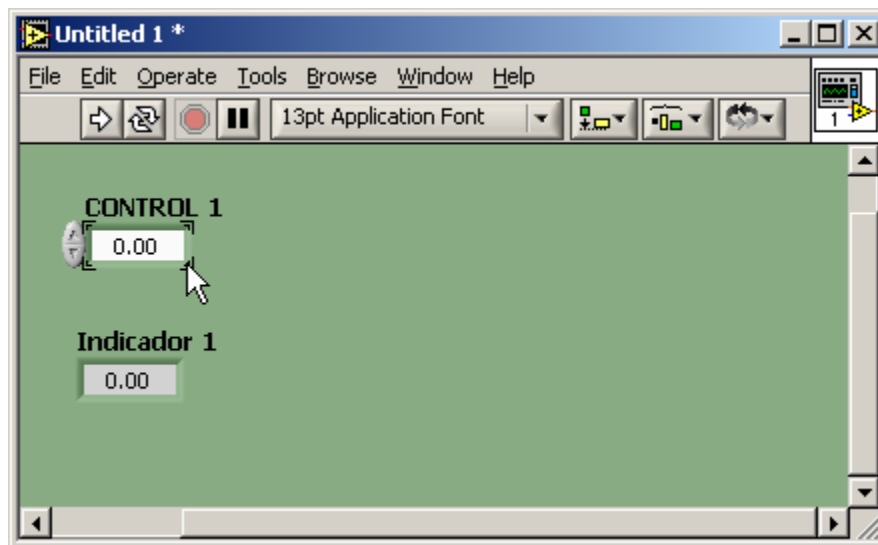


Figura 1.54. Ubicación en un bordo del objeto.

La figura 1.55 muestra como se debe arrastrar el ratón hasta donde se desee modificar el tamaño del objeto.

La figura 1.56 muestra la modificación de tamaño realizada sobre los dos objetos del panel.

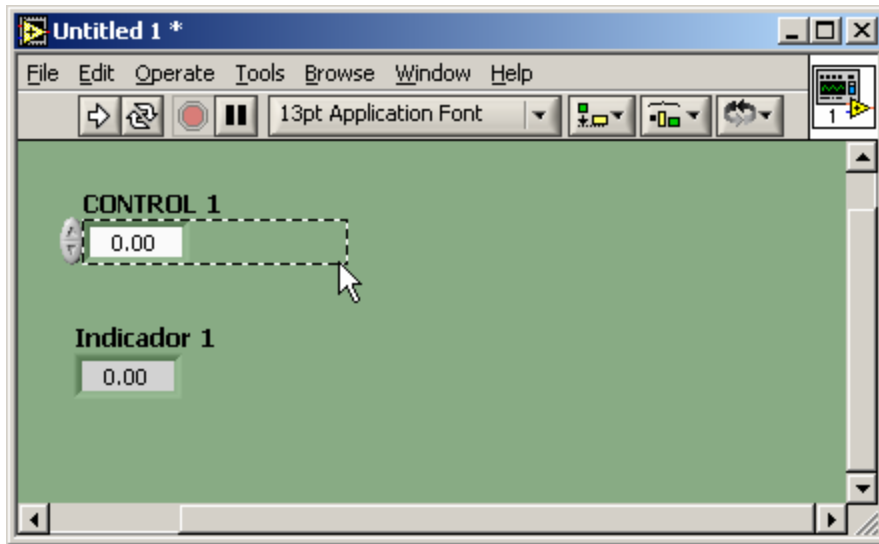


Figura 1.55. Arrastre del ratón hasta el nuevo tamaño deseado.

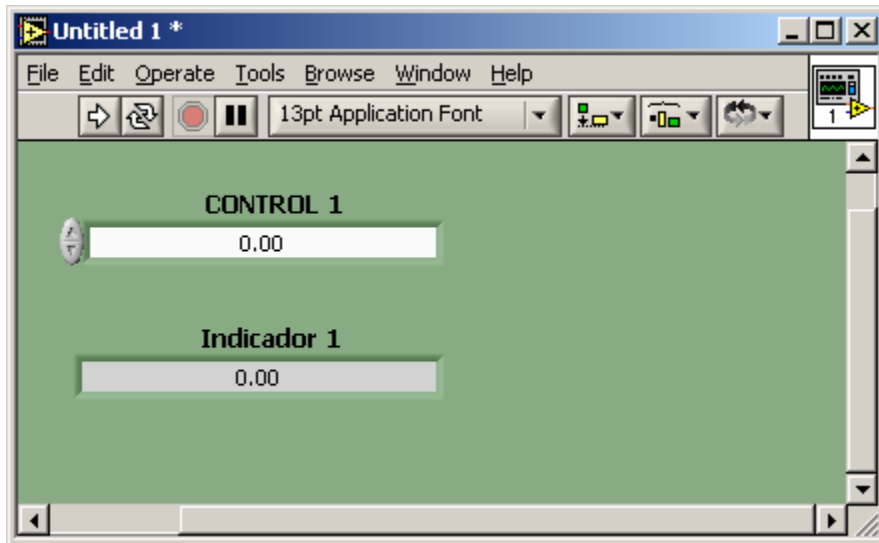




Figura 1.56. Nuevo tamaño de los objetos.

1.16.6 Ejecución de una aplicación

LabVIEW permite ejecutar una aplicación de dos formas.

 **RUN**: Permite ejecutar la aplicación en modo normal.

 **RUN CONTINUOSLY**: Permite ejecutar la aplicación una y otra vez.

Este último método es utilizado únicamente en el proceso de edición y depuración.

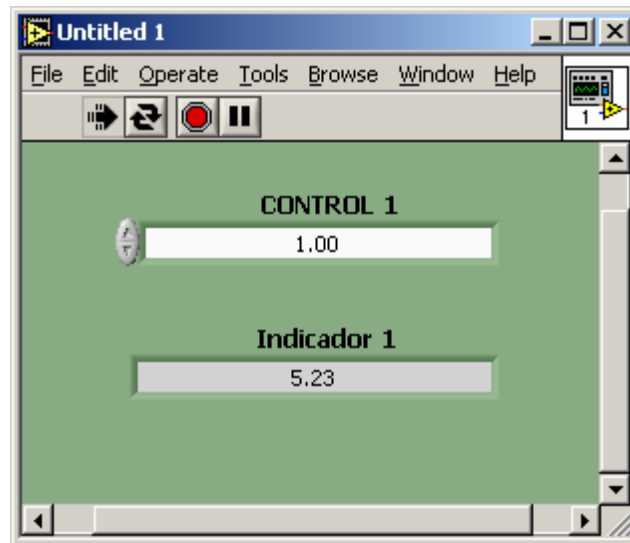


Figura 1.57. Ejecución de un VI.

 **ABORT EXUCUTION**: Aborta la ejecución del programa.

Cuando un VI se ejecuta en modo continuo se puede abortar su ejecución con **ABORT EXUCUTION**. Sin embargo, lo más conveniente es cambiar al modo de ejecución normal presionando de nuevo **RUN CONTINUOSLY** para que la ejecución se detenga normalmente.

1.16.7 Guardar un VI

Desde cualquiera de las dos ventanas se puede seleccionar **File>>Save**. Se podrá entonces ver la ventana de la figura 1.58.

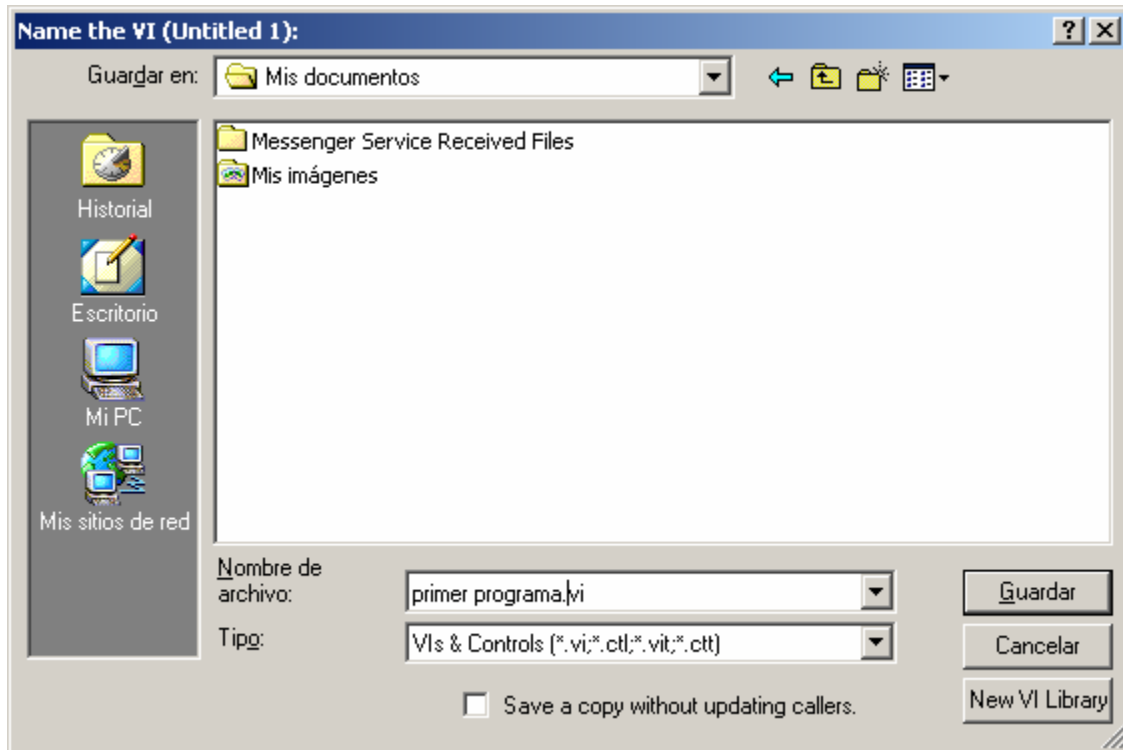


Figura 1.58. Cuadro de diálogo para guardar un VI.

Dependiendo del sistema operativo, cambiará esta ventana y las condiciones para asignar nombre a los VIs. No se debe guardar trabajos personales en las carpetas de instalación de LabVIEW. Se recomienda utilizar otras carpetas para almacenar el trabajo.

Si no se escribe una extensión al archivo en el momento de guardar, LabVIEW utilizará VI como extensión por defecto.

Es posible guardar varios VI en un solo archivo. Esto se denomina una librería de VIs. Para crear una librería de VI se selecciona "New VI Library" de la ventana guardar. Las librerías de VIs poseen extensión llb.

EJERCICIO 1.1 CREACIÓN DE UN NUEVO VI

Consideré la realización de un programa que permita obtener la corriente que circula a través de un circuito eléctrico resistivo dado, para diferentes valores de resistencias y alimentación, figura 1.59.

- En un editor gráfico como *paint*, haga el siguiente dibujo y guárdelo en un archivo tipo *bmp* o *wmf* del tamaño deseado. También puede copiarlo en el portapapeles.

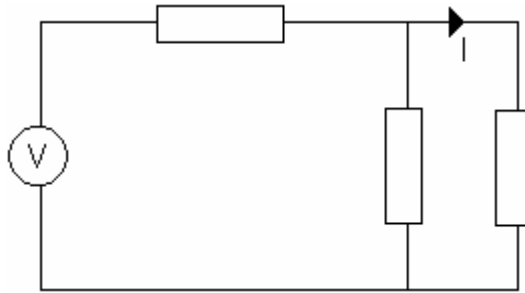


Figura 1.59. Circuito propuesto del ejercicio 1.1.

- Ejecute LabVIEW.
- Seleccione *New VI*, para crear un VI nuevo.
- Pegue el dibujo que realizó en el panel frontal del VI de la siguiente forma:

Si lo copió en el portapapeles debe seleccionar del menú principal **Edit>>Paste**.

Si tiene el dibujo en un archivo, antes de pegar deberá seleccionar **Edit>>Import Picture from File**.

La figura 1.60 muestra el dibujo realizado ya ubicado en el panel frontal de un VI.

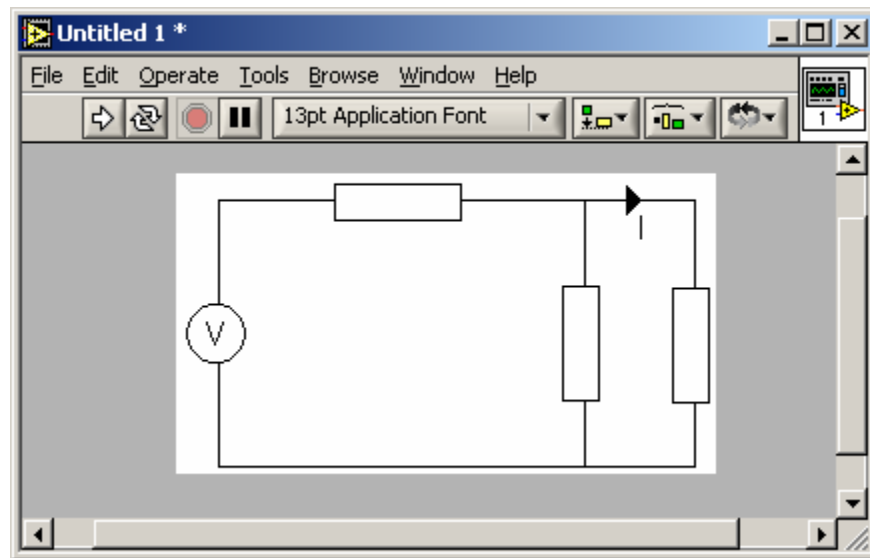


Figura 1.60. Dibujo en el panel frontal.

- De la paleta de controles, obtenga un indicador y cuatro controles numéricos. Póngalos de forma similar al panel de la figura 1.61.

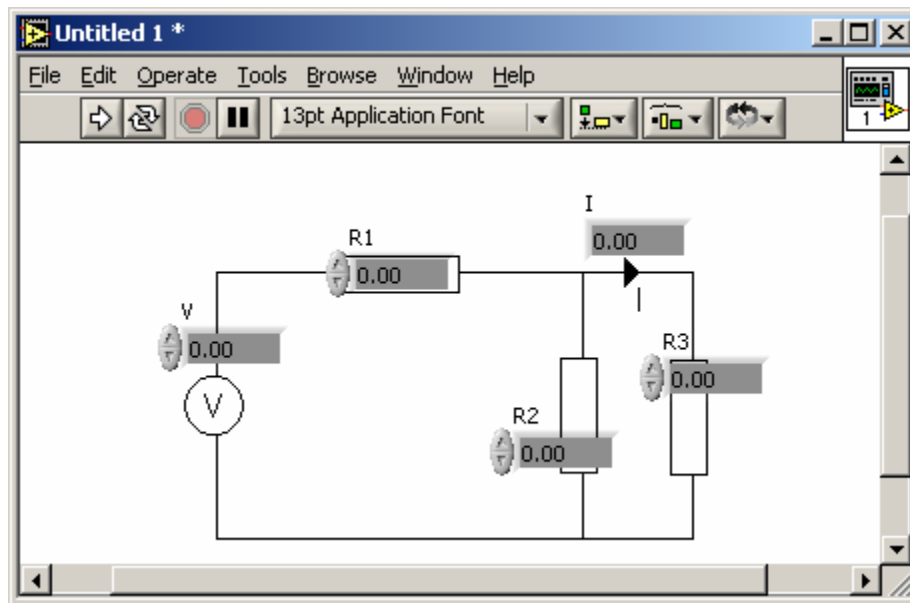


Figura 1.61. Controles e indicadores sobre el dibujo en el panel frontal.

- Obtenga el valor de I , en función de las demás variables.

El valor de I en función de V , $R1$, $R2$ y $R3$, es:

$$I = \frac{\frac{V}{\frac{R2R3}{R2+R3} + R1} \left(\frac{R2R3}{R2+R3} \right)}{R3}$$

Ecuación 1.1. I en función de las demás variables.

- Ponga las funciones aritméticas necesarias en el diagrama para lograr evaluar la función obtenida en la ecuación 1.1. Figura 1.62.

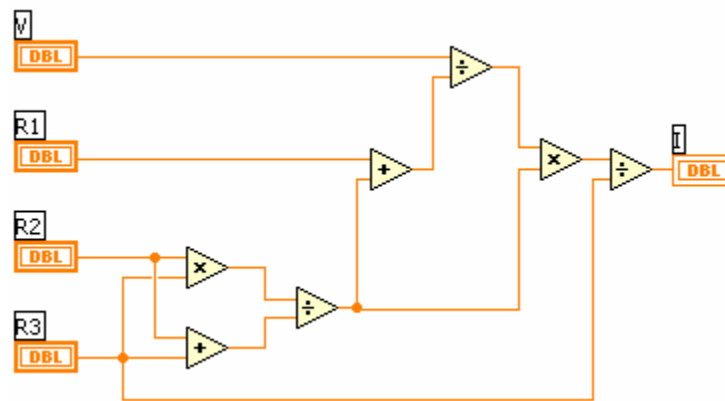




Figura 1.62. Codificación en G de la ecuación 1.1.




- Ejecute el programa:

En la barra de herramientas aparece el icono de **RUN** , que permite ejecutar el programa una vez.

- Varíe los valores de las resistencias o de la fuente y observe como cambia la respuesta “ I ” cada vez que corre el programa.

- Corra continuamente el programa utilizando **RUN CONTINUOSLY** .

Correr continuamente el programa permitirá observar como cambia la salida a medida que se realicen cambios en la entrada.

- Presione nuevamente **RUN CONTINUOSLY**  para detener la ejecución.
- En el diagrama de bloques presione **Highlight Execution**  y corra el programa utilizando **RUN** .

Esto permitirá que observe una animación de cómo fluyen los datos durante la ejecución del VI.

Ahora se desea reemplazar la fuente de voltaje DC por una de AC y las resistencias por impedancias.

Para hacer un análisis en régimen permanente del nuevo circuito se deben reemplazar los números reales por números complejos.

Esto se permite mediante el cambio de representación de cada control e indicador, haciendo clic con el botón derecho del ratón en cada uno y seleccionando la opción **Representation>>CDB** del menú de cada objeto.

La figura 1.63 muestra el menú de un control y la forma de cambiar su representación.

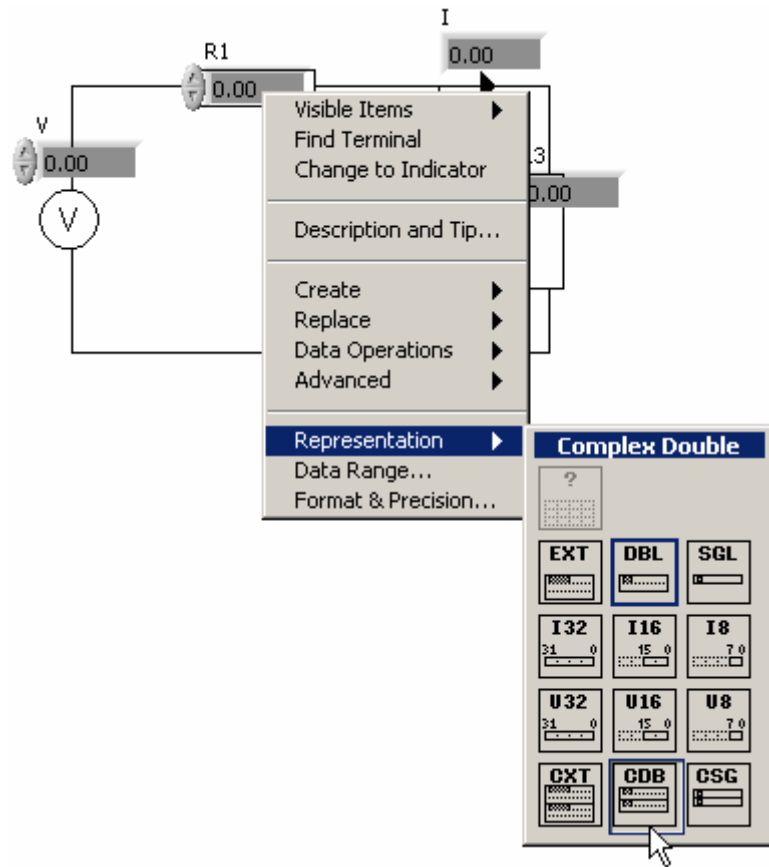


Figura 1.63. Cambio de representación del control R1.

Una vez realizados todos los cambios se podrá observar el panel de la figura 1.64.

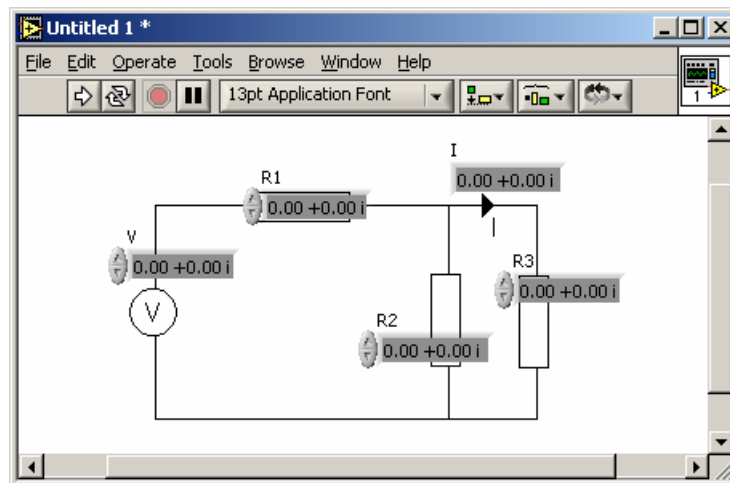


Figura 1.64. Indicadores y controles complejos.

La única diferencia en el diagrama de bloques es la representación de los terminales. Figura 1.65.

Este cambio se hace automáticamente al cambiar la representación de los controles e indicadores en el panel frontal.

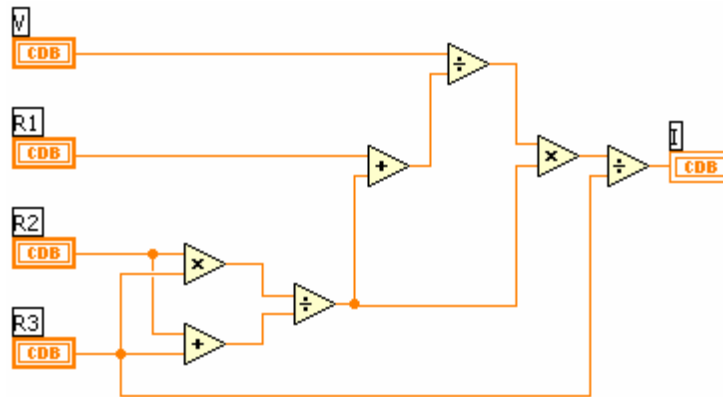


Figura 1.65. Diagrama con terminales complejos.

No es necesario hacer ninguna modificación para que el programa opere con números complejos. Esto se debe a las características de polimorfismo y sobrecarga de las funciones de LabVIEW.

Para obtener más información acerca de las funciones, como por ejemplo la forma de operación y las características de polimorfismo y sobrecarga se puede utilizar la ventana de ayuda.

La ventana de ayuda se obtiene con el método de teclado <CONTROL + H> y tiene la apariencia de la figura 1.66.

Esta ventana mostrará ayuda acerca de la función sobre la que se encuentre el cursor en cada instante.

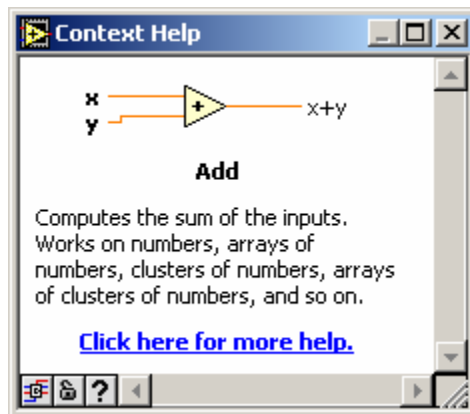


Figura 1.66. Ventana de ayuda.

- Corra el programa de nuevo y observe los resultados.
- Guarde el VI en su carpeta de trabajo y ciérrelo.

FIN EJERCICIO 1.1

1.17 EJERCICIOS PROPUESTOS

- 1) Crear un panel de control idéntico al de la figura 1.67.

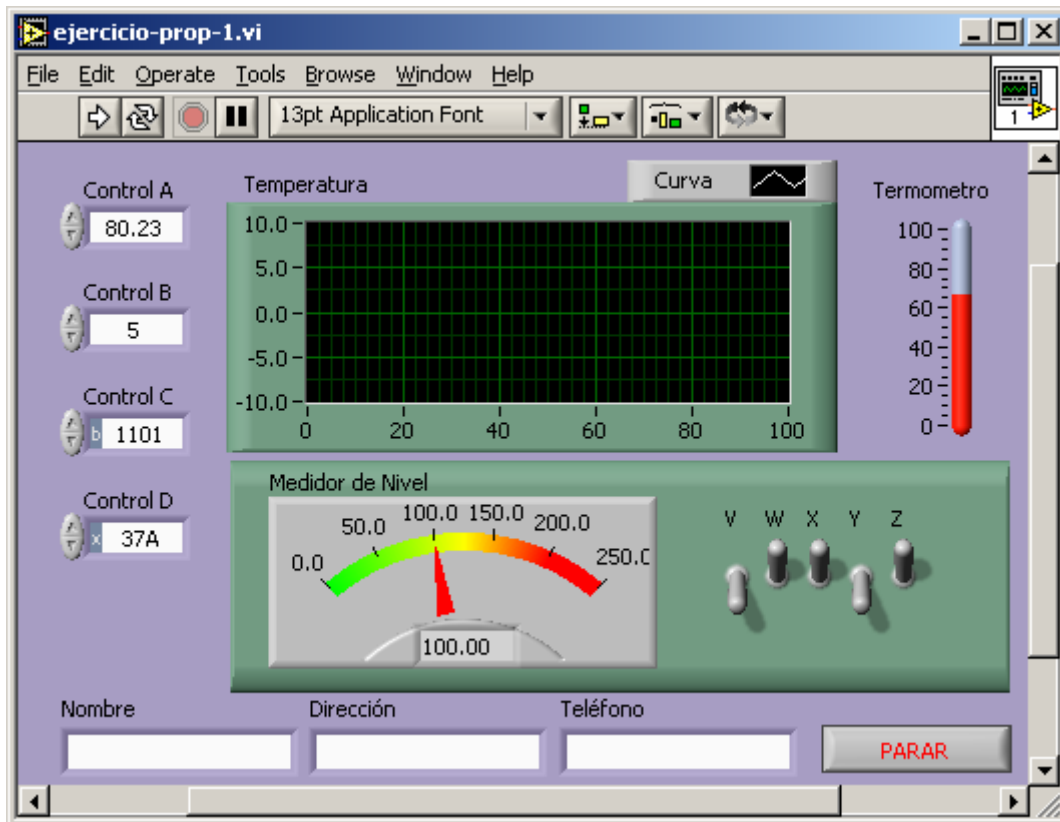


Figura 1.67. Ejercicio propuesto 1.

- 2) Hacer un programa en LabVIEW que evalúe la función $y = 3x^2 + 4x - 9$ en cualquier valor de x .
- 3) Codificar en G la expresión booleana $S = \overline{A}B.CD + AB.\overline{C}D + \overline{A}B\overline{C} + \overline{A}C.B\overline{D}$.
- 4) Crear un VI que genere un número aleatorio entre 1 y 100.

2. ESTRUCTURAS.

2.1 OBJETIVO

Estudiar las estructuras de programación utilizadas por LabVIEW para definir secuencias, decisiones y ciclos. Además, estudiar el nodo de fórmula del lenguaje G.

2.2 DESCRIPCIÓN

Una estructura en general es un nodo que controla el flujo de los datos de un programa en G. LabVIEW cuenta, en orden, con las siguientes estructuras.

1. Sequence.
2. Case.
3. For Loop.
4. While Loop.
5. Formula Node.

Las estructuras se encuentran en la paleta de funciones como se muestra en la figura 2.1.

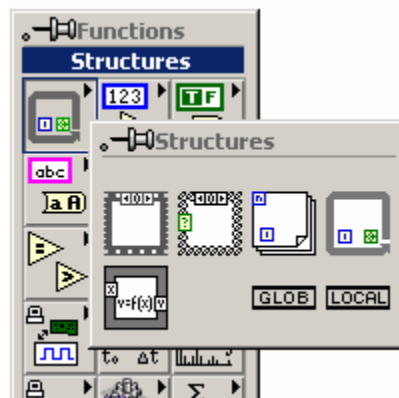


Figura 2.1. Estructuras.

Para efectos de estudio se comenzará con las estructuras cíclicas que son *While Loop* y *For Loop*.

2.3 ESTRUCTURA “WHILE LOOP”

La estructura *While Loop* es un ciclo que repite el subdiagrama que contiene hasta que una condición determinada se cumpla. En G está representada por el marco que se muestra en la figura 2.2.

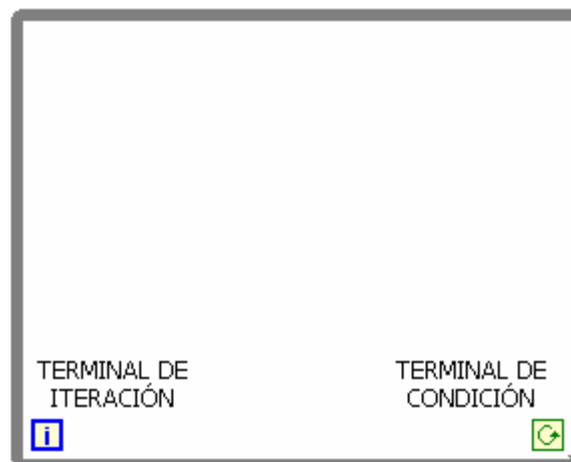


Figura 2.2. Estructura *While Loop*.

Por defecto las instrucciones contenidas en el ciclo se repetirán mientras que al terminal de condición llegue un valor verdadero. Si se desea cambiar la lógica del terminal de condición, es decir, que el ciclo se repita mientras que a este llegue un valor falso, basta con hacer clic derecho en dicho terminal y seleccionar la opción **Stop If True** como se muestra en la figura 2.3. Si el terminal de condición no se cablea, el VI no se podrá ejecutar.



Figura 2.3. Terminal de condición de la estructura *While Loop*.

El terminal de iteración determina el número de veces que se ha ejecutado el ciclo y puede ser utilizado para visualización o para alguna operación dentro de la estructura. Este terminal varía desde 0 hasta N-1 donde N es el número de iteraciones realizadas por el ciclo.

La estructura *While Loop* del lenguaje G es equivalente a **DO..WHILE** en C/C++ o a **REPEAT..UNTIL** en Pascal.

Se debe tener en cuenta que el subdiagrama contenido en la estructura *While Loop* se ejecutará por lo menos una vez.

EJERCICIO 2.1 SIMULACIÓN DE LA LECTURA DE UNA TEMPERATURA

La figura 2.4 muestra el panel frontal y el diagrama de bloques para una simulación simple de la lectura de una temperatura.

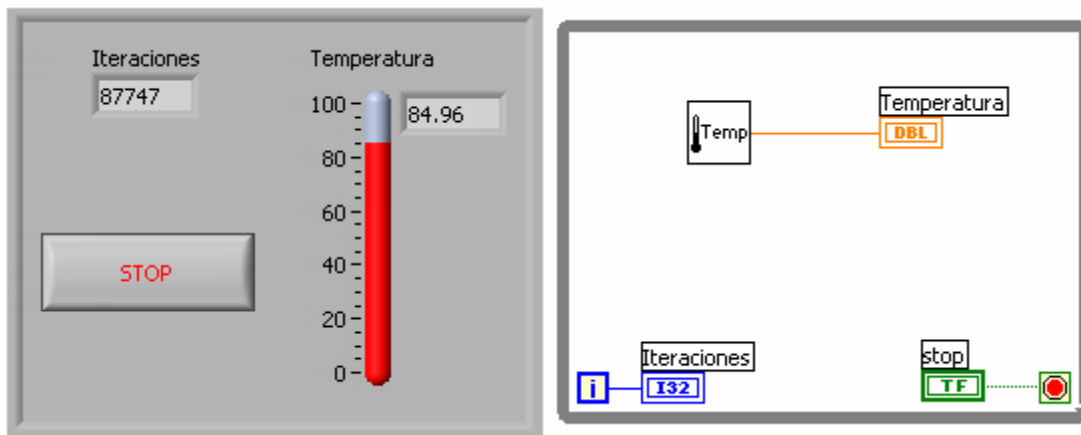


Figura 2.4. Simulación de lectura de temperatura.

El programa se ejecutará hasta que el usuario presione el botón de paro.

Como se desea que el ciclo se detenga sólo cuando se presione el botón de paro, entonces se debe cambiar la lógica por defecto del terminal de condición.

El subVI que simula la lectura de temperatura se encuentra en la paleta de funciones en el submenú *Tutorial* y se llama “*Digital Thermometer.vi*”.

Si se desea agregar una espera para el VI no se ejecute tan rápido, se puede utilizar la función *Wait*, que se encuentra en la paleta de funciones en el submenú *Time&Dialog*. Esta función produce la espera especificada en milisegundos.

La figura 2.5 muestra su utilización en el diagrama para lograr una espera de 250 milisegundos en cada iteración.

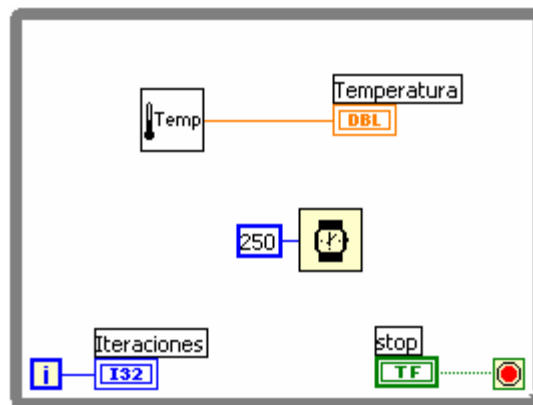


Figura 2.5. Función *Wait* en un ciclo.

Una forma fácil de obtener una gráfica de los valores de temperatura leídos por el subVI de simulación, es con el indicador “*Waveform chart*” localizado en la paleta de controles en el submenú *Graph* como se muestra en la figura 2.6.

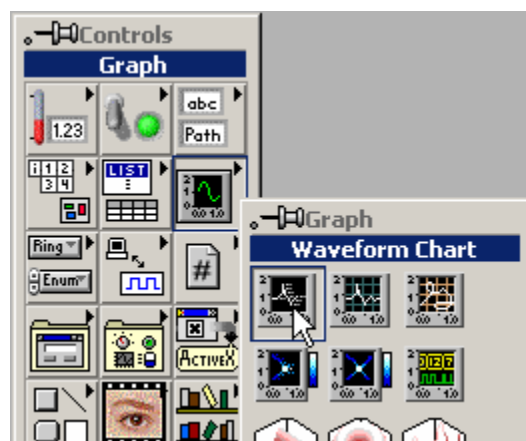


Figura 2.6. *Waveform Chart*.

El estudio detallado de este graficador será realizado más adelante. Por ahora basta decir que este control permite acumular datos escalares y graficarlos.

Reemplazando el termómetro en el panel de la figura 2.4 por la gráfica se obtiene el panel de la figura 2.7.

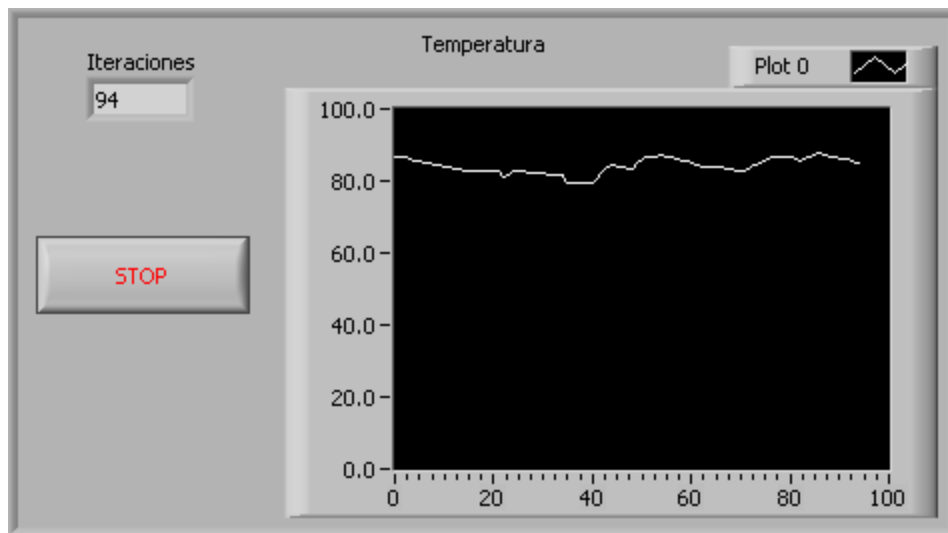


Figura 2.7. Gráfica de datos de temperatura.

El diagrama conserva el mismo aspecto ya que el terminal para la gráfica es igual al del termómetro.

FIN EJERCICIO 2.1

EJERCICIO 2.2 GENERACIÓN DE UNA ONDA SENO

Se desea generar continuamente una onda seno y verla en un graficador *Chart* hasta que se presione el botón de paro.

Las figuras 2.8 y 2.9 muestran el panel frontal y el diagrama de esta aplicación.

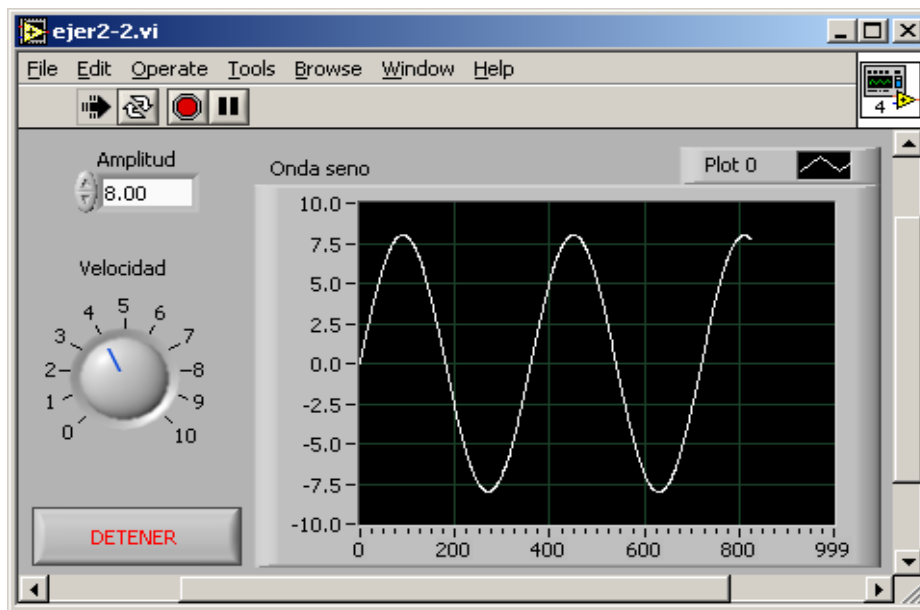


Figura 2.8. Panel frontal para generar onda seno.

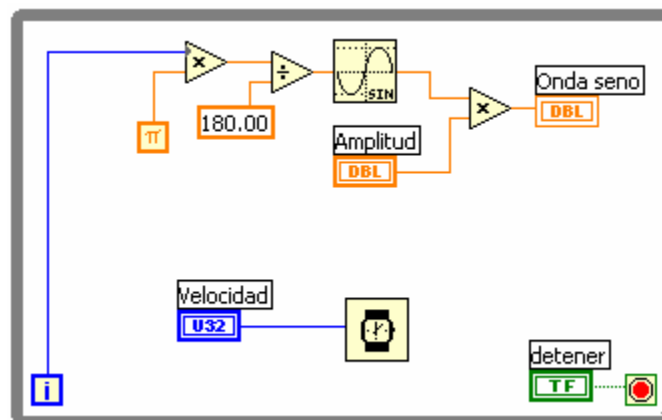


Figura 2.9. Diagrama para generar onda seno.

La función seno tomada de la paleta de funciones numéricas recibe el argumento en radianes, por tanto, el terminal de iteraciones utilizado para generar un punto cada grado debió ser multiplicado por $\pi/180$.

La velocidad con que se genera los datos es controlada por la función *Wait*.

FIN EJERCICIO 2.2

2.4 ESTRUCTURA “FOR LOOP”

La estructura *For Loop* es un ciclo que repite el subdiagrama que contiene un número definido de veces. En G está representada por el marco que se muestra en la figura 2.10.

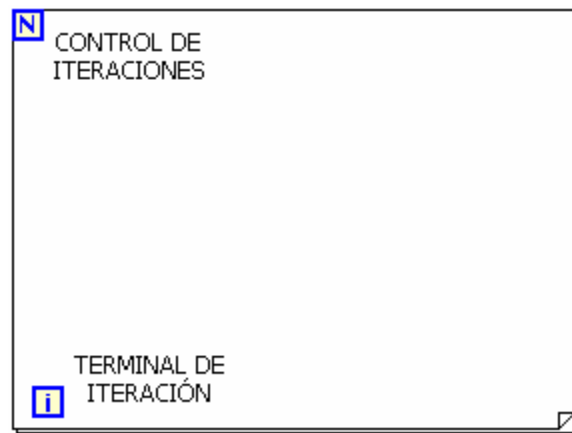


Figura 2.10. Estructura *For Loop*.

El terminal de iteración indica el número de veces que se ha ejecutado el ciclo. Varía desde 0 hasta $N-1$ donde N es el número total de iteraciones que realiza el ciclo.

El control de iteraciones contiene el número de veces que se ejecutará el subdiagrama contenido en el ciclo.

En C/C++ es análogo a

```
for(i=0, i<N, i++)
```

```
{  
}
```

```
end
```

EJERCICIO 2.3 GRÁFICA DE 100 NÚMEROS ALEATORIOS ENTRE 10 Y 50

Como se conoce el número exacto de veces que se debe repetir la tarea se utiliza una estructura *For Loop*.

La función *random* genera un número aleatorio entre 0 y 1. Por tanto se debe ajustar su rango al solicitado. Una forma fácil de hacerlo es multiplicar el número generado por 40 y sumarle 10. En general si el rango solicitado es [a,b] entonces el número aleatorio es $N = R(b-a)+a$, donde R es la salida de la función *random*.

Para graficar los 100 números generados se utiliza una gráfica *Chart*.

La figura 2.11 muestra el panel y el diagrama que dan solución a este ejercicio.

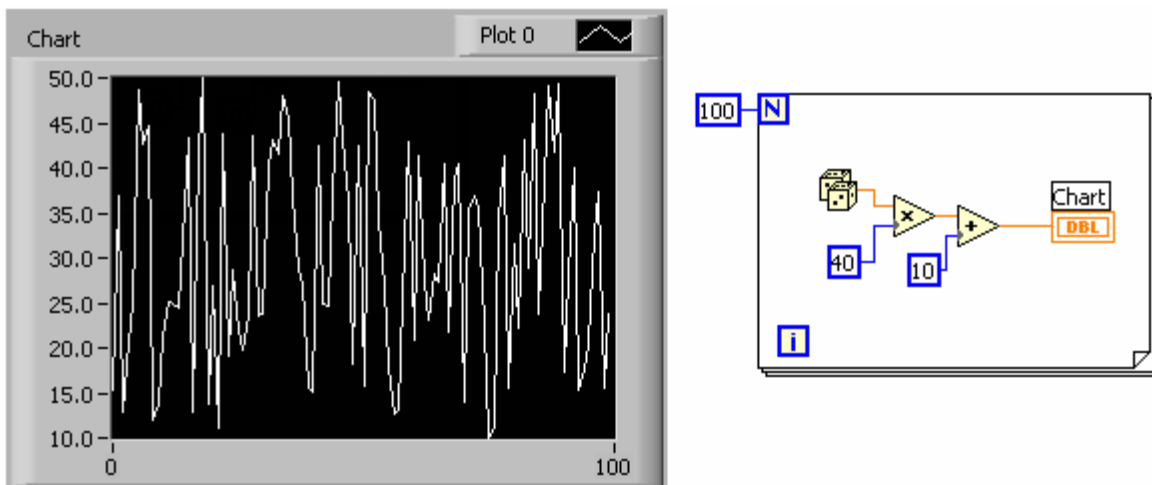


Figura 2.11. Gráfica de 100 números aleatorios.

De nuevo, si se desea que el VI se ejecute más lento, se debe adicionar la función *Wait*.

FIN EJERCICIO 2.3

Es muy frecuente que en las estructuras *While Loop* y *For Loop* sea necesario pasar datos entre iteraciones. Para ello se utilizan los “*shift registers*”.

Éstos se encuentran en el menú de las estructuras y se obtienen como se muestra en la figura 2.12.

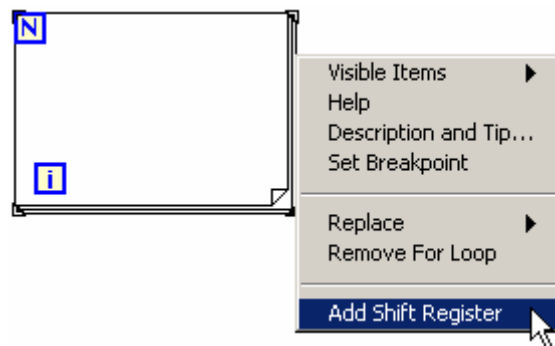


Figura 2.12. Adición de un *Shift register*.

Los “*shift registers*” o registros de desplazamiento están formados por un par de terminales que se adaptan a cualquier tipo de dato y que están localizados a cada lado de los bordes de la estructura como se muestra en la figura 2.13.

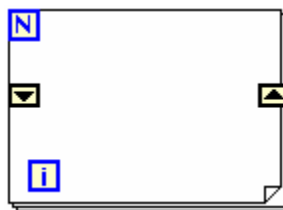


Figura 2.13. Localización de un registro de desplazamiento.

La secuencia de comportamiento de un registro de desplazamiento se muestra en la figura 2.14.

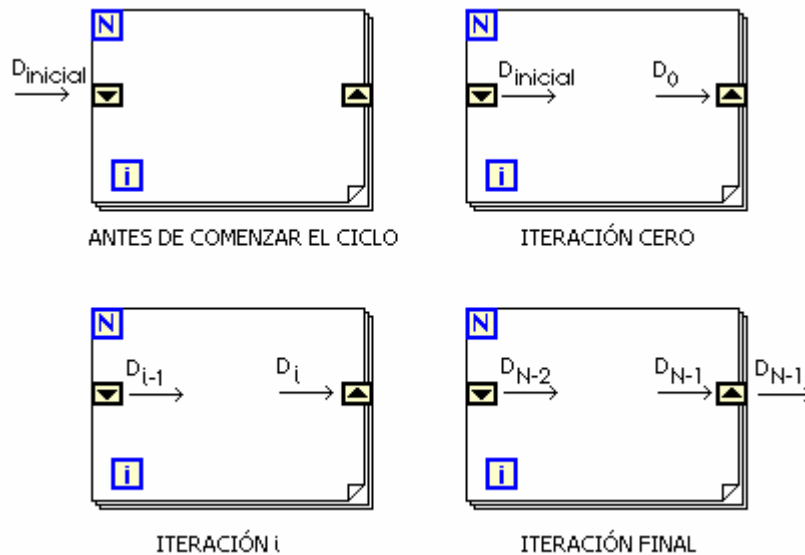


Figura 2.14. Comportamiento de un registro de desplazamiento.

El terminal derecho almacena el dato una vez concluya la iteración y le entrega el dato al terminal de la izquierda para que sea utilizado en la próxima iteración.

En la primera iteración el sistema podría tomar un número no deseado, por tanto se debe inicializar, desde afuera, con un valor conveniente del mismo tipo de la variable utilizada.

EJERCICIO 2.4 SUMAR LOS NÚMEROS ENTEROS ENTRE 1 Y 100

Este ejercicio busca encontrar un valor $S = 1 + 2 + 3 + \dots + 98 + 99 + 100$.

Para realizar esta suma se utiliza una estructura *For Loop*.

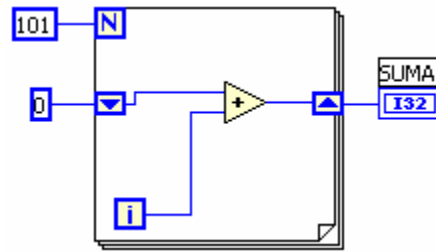


Figura 2.15. Sumatoria de 1 a 100.

Como el terminal de iteración comienza desde 0, se debe realizar 101 iteraciones para que el último número de la sumatoria sea el 100.

Esto significa que en realidad la suma realizada por este ejercicio es:

$$S = 0 + 1 + 2 + 3 + \dots + 98 + 99 + 100$$

Lo que no afecta el resultado deseado.

FIN EJERCICIO 2.4

Para poder tener acceso a iteraciones anteriores se debe adicionar elementos al registro de desplazamiento. Para adicionarlos, se hace clic derecho sobre uno de los terminales y se selecciona *Add Element*, como se muestra en la figura 2.16.

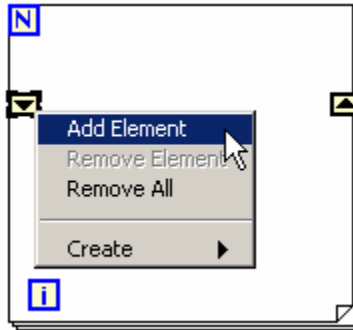


Figura 2.16. Adición de elementos al registro de desplazamiento.

El comportamiento del registro de desplazamiento con elementos adicionados se ilustra en la figura 2.17.

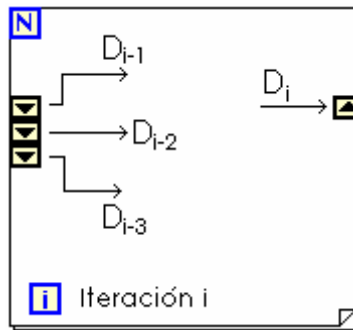


Figura 2.17. Comportamiento de elementos en los registros de desplazamiento.

Todos los elementos que se adicionen al registro de desplazamiento también deben ser inicializados, de lo contrario se correrá el riesgo de tener valores no deseados.

EJERCICIO 2.5 PROMEDIAR LOS ÚLTIMOS DOS DATOS ALEATORIOS

Para el ejercicio 2.3, se requiere calcular el promedio de los últimos 2 datos aleatorios que se han generado.

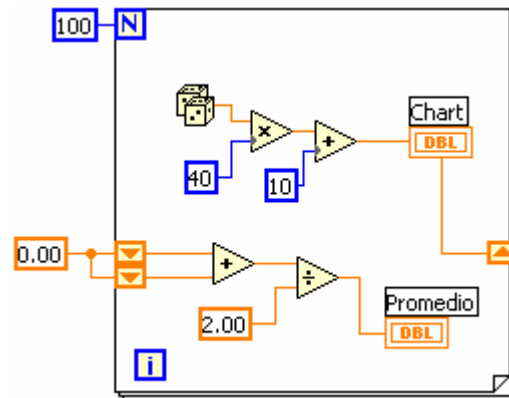


Figura 2.18. Promedio de últimos 2 datos generados.

Se han sumado y dividido por 2 los datos correspondientes a las últimas dos iteraciones.

FIN EJERCICIO 2.5

2.5 ESTRUCTURA “SEQUENCE”

La estructura *sequence* tiene la apariencia de la figura 2.19 y permite ejecutar varios subdiagramas de manera ordenada y controlada por el programador. En los lenguajes de programación convencionales basados en código de líneas no se requiere y por lo tanto no existe una estructura análoga.

Esta estructura posee varios subdiagramas denominados “*frames*” que se ejecutan en estricto orden y sólo es visible uno a la vez.

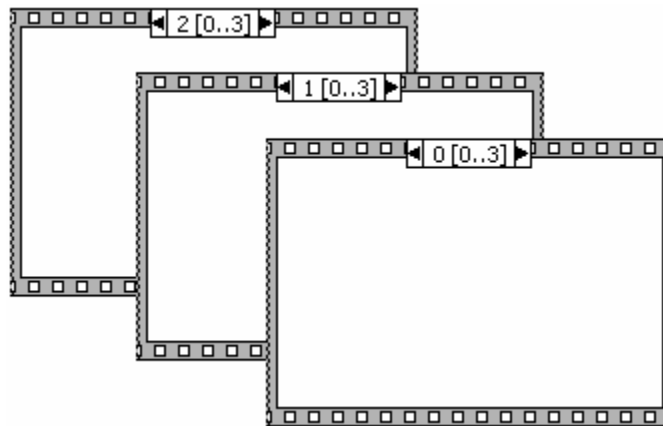


Figura 2.19. Apariencia de estructuras *sequence*.

En la parte superior del marco de cada estructura se encuentra el identificador de diagrama que es utilizado para navegar entre *frames*. Ver la figura 2.20.

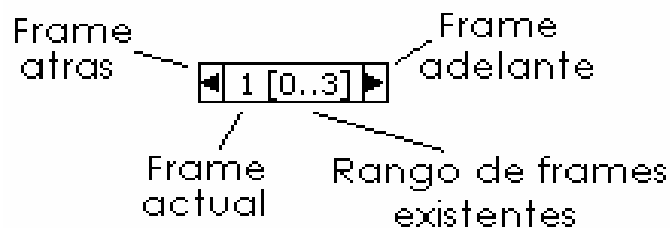


Figura 2.20. Identificador de diagrama.

Por defecto la estructura *sequence* posee un solo *frame* y no tiene identificador de diagrama. Para adicionar, borrar, mover o duplicar *frames* se puede acceder al menú de la estructura desde alguno de sus lados. Ver figura 2.21.

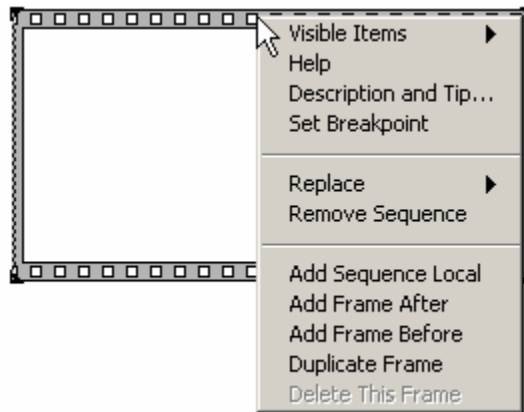


Figura 2.21. Menú de la estructura *sequence*.

En este menú es posible seleccionar alguna de las siguientes opciones:

Add Sequence Local: Utilizado para pasar un dato desde un *frame* a otro posterior. Esta opción genera un terminal en el borde de la estructura como en la figura 2.22.

Add Frame After: Adiciona un *frame* después del actual.

Add Frame Before: Adiciona un *frame* antes del actual.

Duplicate Frame: Genera una copia exacta del *frame* actual en un nuevo *frame*.

Delete Frame: Elimina el *frame* actual. Sólo se habilita cuando la secuencia posee dos o más *frames*.

Remove Sequence: Se utiliza para remover la estructura *sequence*. Este procedimiento eliminará todos los objetos de los *frames* que no estén visibles.

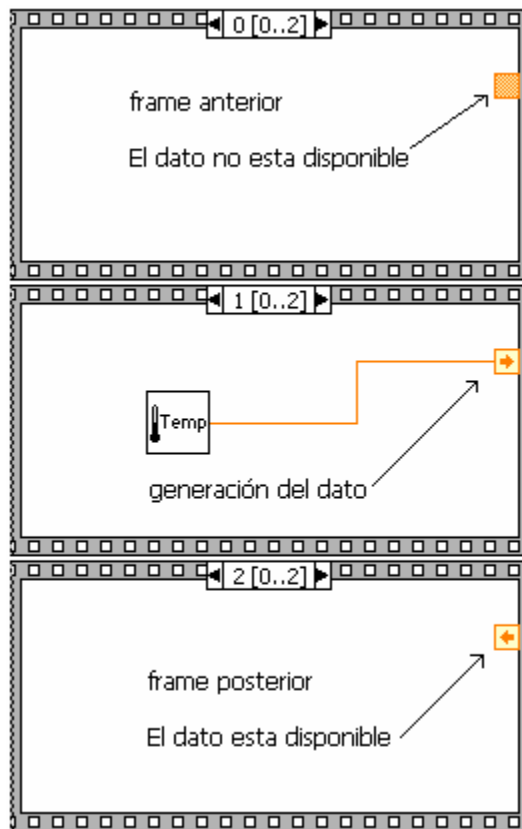


Figura 2.22. Elementos de la estructura *SEQUENCE*.

EJERCICIO 2.6 MEDIR EL TIEMPO QUE EL PC TARDA EN GENERAR 10000 DATOS ALEATORIOS

Para la solución de este ejercicio se utiliza una sola estructura *sequence* con tres *frames* como se enseña en la figura 2.23.

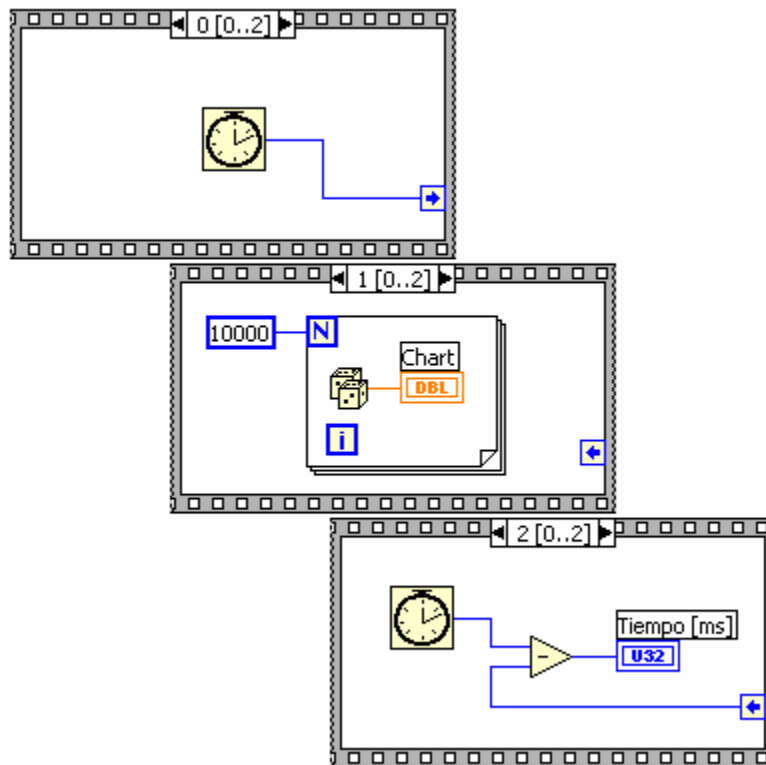


Figura 2.23. Frames 0, 1 y 2.

En el *frame* cero es necesario tomar un tiempo de referencia y enviarlo al *frame* dos donde se tomará un nuevo valor para restarle el tomado en el *frame* cero. Esta diferencia será el tiempo que tomó el computador en realizar la tarea.



La función *Tick Count* se encuentra en la paleta de funciones en el submenú *Time&Dialog* y devuelve un valor de tiempo en milisegundos contados a partir de un instante de referencia no controlable por el usuario. El valor devuelto por esta función es U32, por lo tanto se debe tener cuidado al usarla porque saltará de $2^{32}-1$ a 0.

Nótese que el dato entregado por la función *tick count* en el *frame* cero es enviado al *frame* uno a través de una secuencia local.

Para poder utilizar un dato después de la ejecución de la estructura basta con alambrarlo a una de las paredes como se puede ver en la figura 2.24.

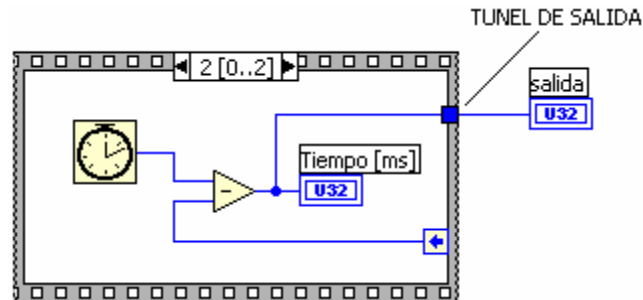


Figura 2.24. Salida de datos de un *sequence*.

El paso de un dato entre la parte interna y externa de una estructura se denomina **túnel** y puede ser de entrada o de salida dependiendo del sentido del flujo de los datos.

FIN EJERCICIO 2.6

2.6 ESTRUCTURA “CASE”

La estructura *Case* posee varios subdiagramas denominados casos (*cases*) de los cuales sólo se ejecuta uno.

Esta estructura tiene la apariencia de la figura 2.25.

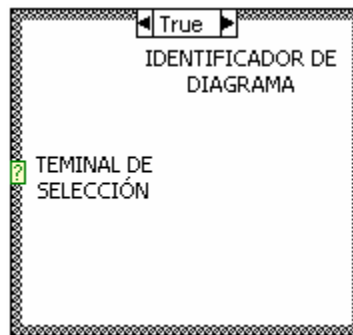


Figura 2.25. Estructura case.

Al igual que en la estructura *sequence* sólo es visible un subdiagrama a la vez.

Esta estructura es una fusión de las estructuras **IF** y **CASE** o **SWITCH** de los lenguajes de programación convencionales.

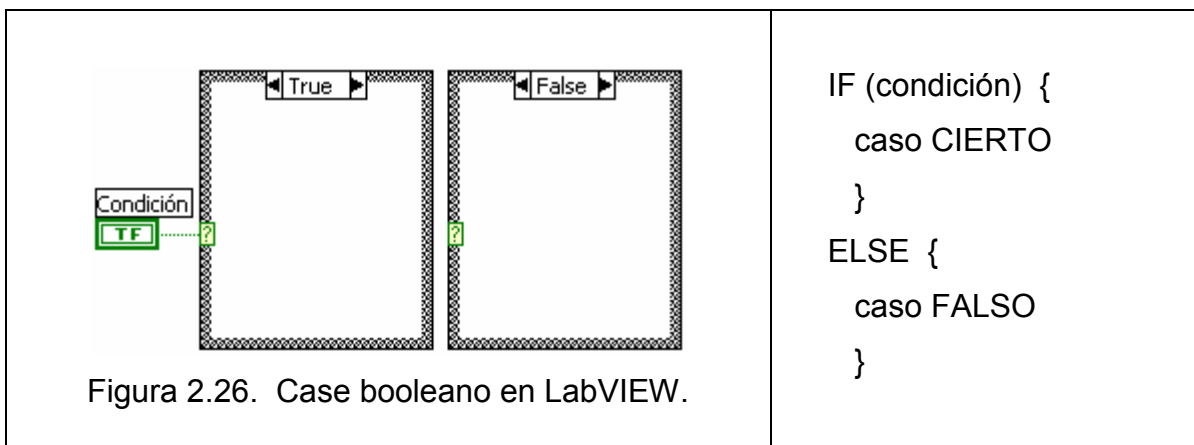
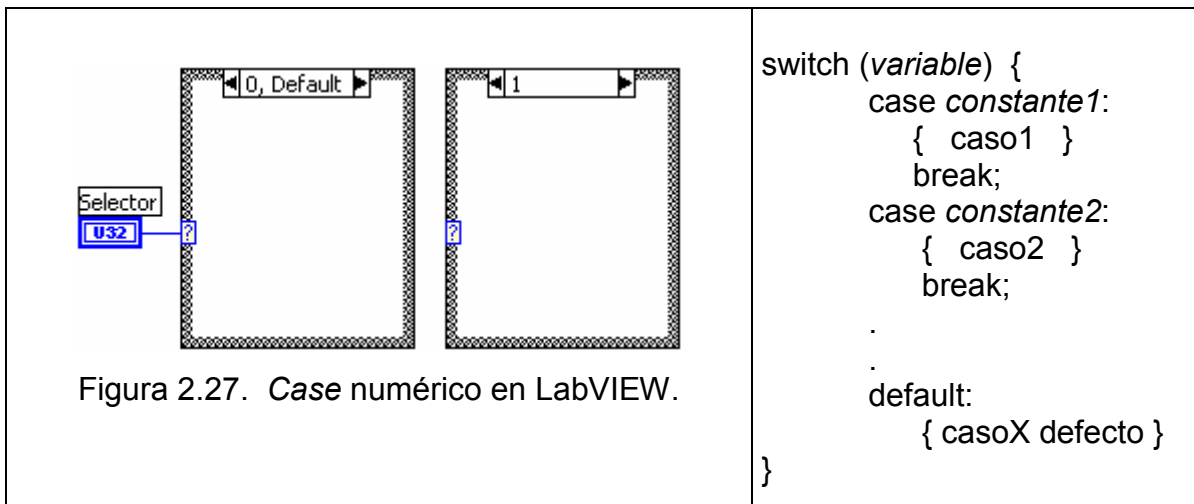


Figura 2.26. Case booleano en LabVIEW.

Dependiendo del tipo de variable asociada al terminal de selección la estructura se comportará como un **IF** o como un **CASE**. Si el valor cableado es booleano la estructura tendrá dos casos *FALSE* y *TRUE*, pero si es numérico o cadena la estructura podrá tener desde 2 hasta $2^{16}-1$ casos.



La estructura CASE también posee un menú que se obtiene haciendo clic derecho sobre uno de sus bordes. En este menú se encuentran opciones como añadir, eliminar, duplicar, mover y organizar casos.

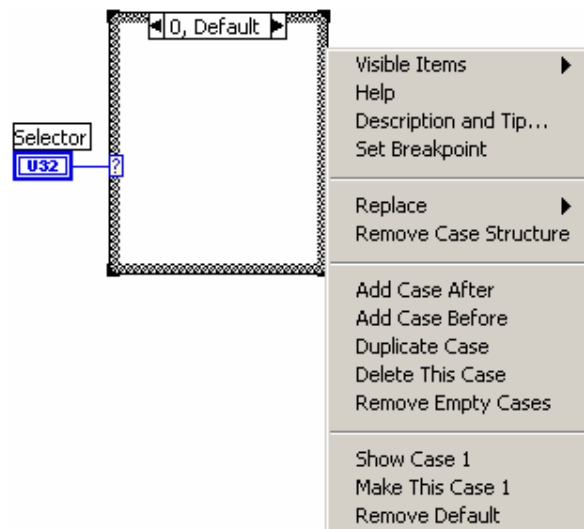


Figura 2.28. Menú de la estructura case.

Cuando la variable de selección no es booleana LabVIEW exige que alguno de los casos de la estructura *case* sea definido como el caso por defecto.

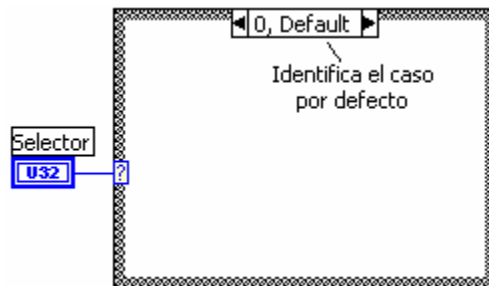


Figura 2.29. Caso declarado por defecto.

EJERCICIO 2.7 MENÚ DE OPCIONES

A partir de dos entradas numéricas y un control tipo menú con las opciones suma, resta, multiplicación y división, se busca generar una salida que enseñe su resultado.

En el Panel Frontal:

Para resolver este ejercicio es necesario utilizar un control tipo menú, que son comúnmente utilizados para seleccionar una opción entre varias posibles.

Los controles tipo *menu ring* se encuentran en la paleta de controles en el submenú *Ring&Enum* y son de tipo numérico U16. Para editar un control de menú se debe tomar la herramienta de texto.

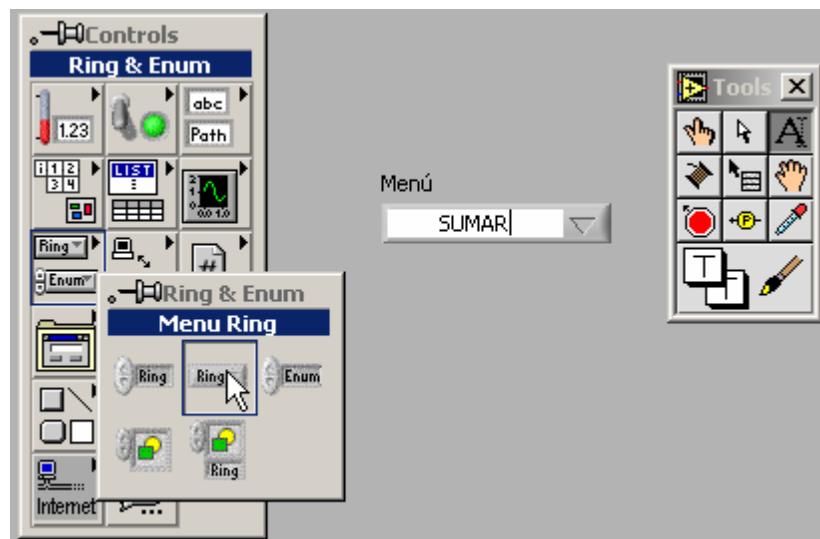


Figura 2.30. Selección y edición de un control *menu ring*.

El control *menu ring* posee también un menú al que se accede haciendo clic derecho sobre éste. En él se pueden seleccionar acciones como adicionar, remover o deshabilitar ítems. El menú de este control se muestra en la figura 2.31.

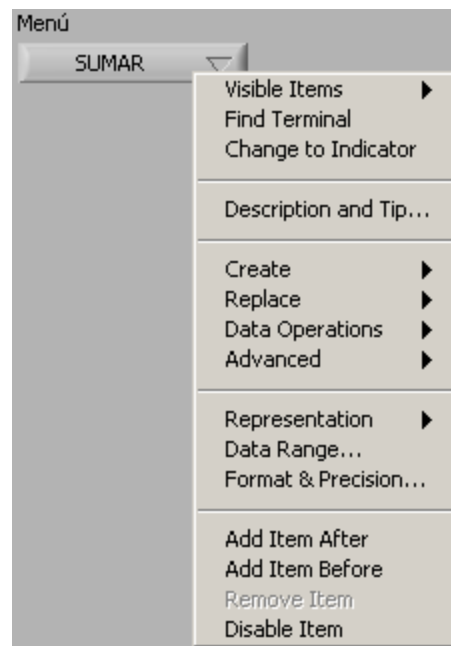


Figura 2.31. Menú del control *menu ring*.

Se adicionan los ítems Sumar, Restar, Multiplicar y Dividir. El aspecto del control *menu ring* desplegado se muestra en la figura 2.32.

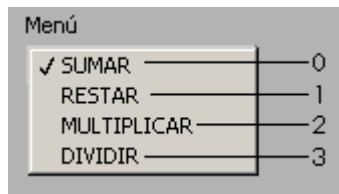


Figura 2.32. Menú desplegado.

De acuerdo con la figura 2.32 este control tomará el valor de 0 cuando se seleccione SUMAR, 1 para RESTAR, 2 para MULTIPLICAR y 3 para DIVIDIR.

Ahora se puede crear los otros elementos. Dos controles numéricos y un indicador numérico.

El panel frontal terminado se muestra en la figura 2.33.



Figura 2.33. Panel frontal completo.

En el diagrama de bloques:

El terminal del control de menú se utilizará para cablear el terminal de selección de una estructura *case* como se muestra en la figura 2.34.

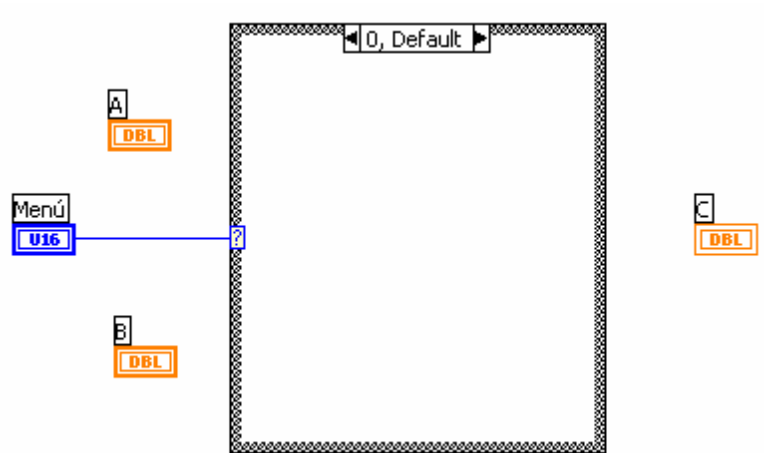


Figura 2.34. Estructura Case controlado por un *menu ring*.

Como el control posee 4 posibles opciones, se debe adicionar los casos 2 y 3. El caso 0 será el caso por defecto.

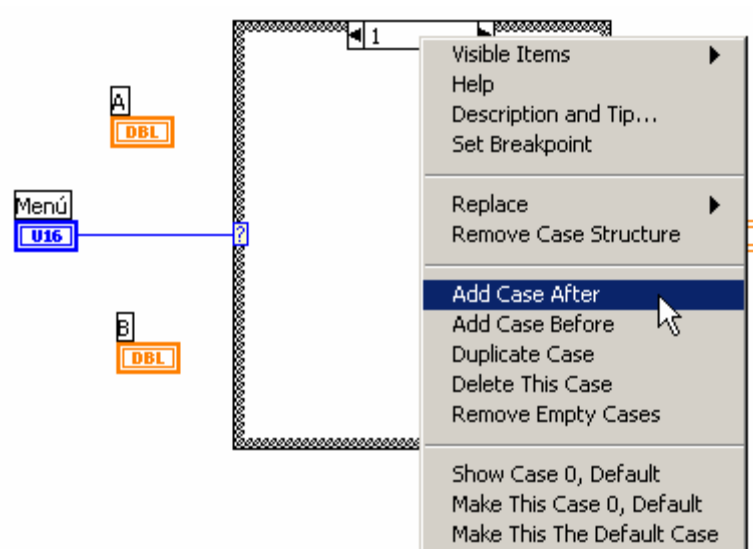


Figura 2.35. Adición de casos.

Ahora se debe colocar las funciones en los respectivos casos, por ejemplo, la función suma en el subdiagrama del caso cero, la resta en el uno y así sucesivamente.

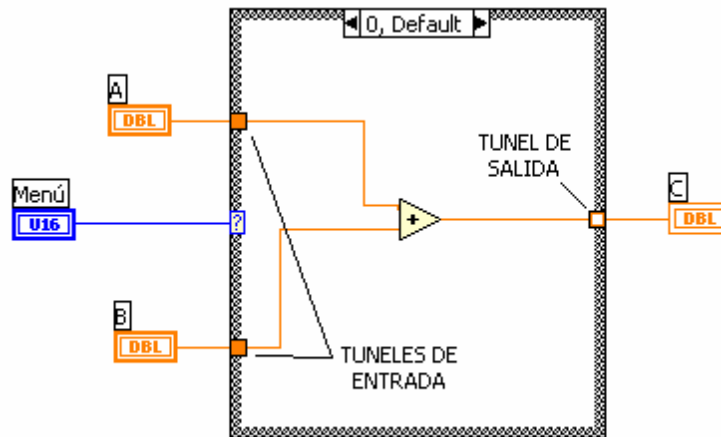


Figura 2.36. Subdiagrama cero.

Los datos en todos los túneles de entrada y en el terminal de selección pueden ser accedidos por todos los subdiagramas de casos.

Para que un túnel de salida este completo se le debe alimentar desde todos los casos que posea la estructura. La figura 2.37 muestra dos túneles de salida en una misma estructura case uno de los cuales no está completo.

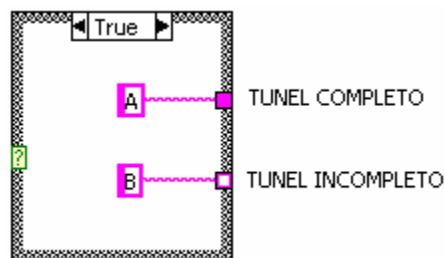


Figura 2.37. Túneles de salida de un case.

Observe en la figura 2.37 que aún existe un túnel de salida incompleto. Esto significa que en alguno o en varios de los casos no visibles aún éste no ha sido alimentado. Esto se ve por ejemplo en la figura 2.38.

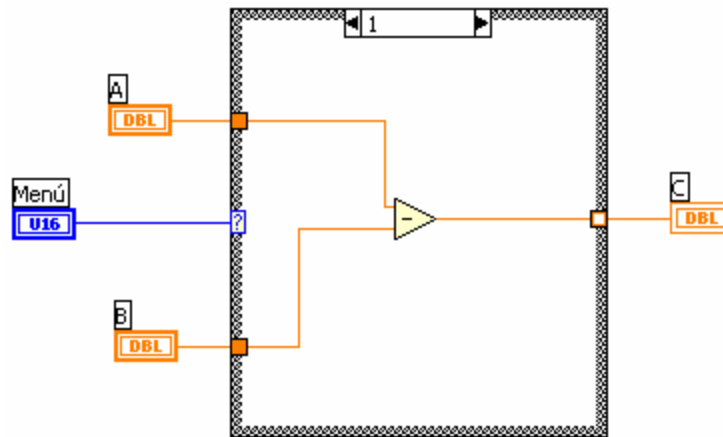


Figura 2.38. Caso 1 con túnel incompleto.

Se debe tener cuidado al completar los túneles, pues frecuentemente sucede que el cableado no se realiza exactamente sobre el túnel sino que se generan túneles nuevos que podrían estar superpuestos y aparentar ser uno sólo.

Se recomienda asegurarse que el túnel al que cablea sea el correcto. Ver técnicas de cableado en el capítulo uno.

La figura 2.39 muestra el caso 3 con el túnel ya completo.

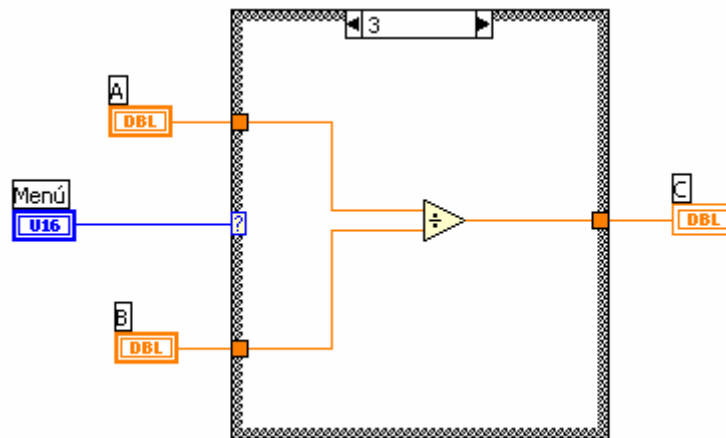


Figura 2.39. Caso 3 con túnel de salida completado.

Por último, se debe recordar que el programa debe correr por si mismo sin la ayuda del botón de correr continuamente. Para ello se anexa una estructura *While Loop* que encierre toda la operación.

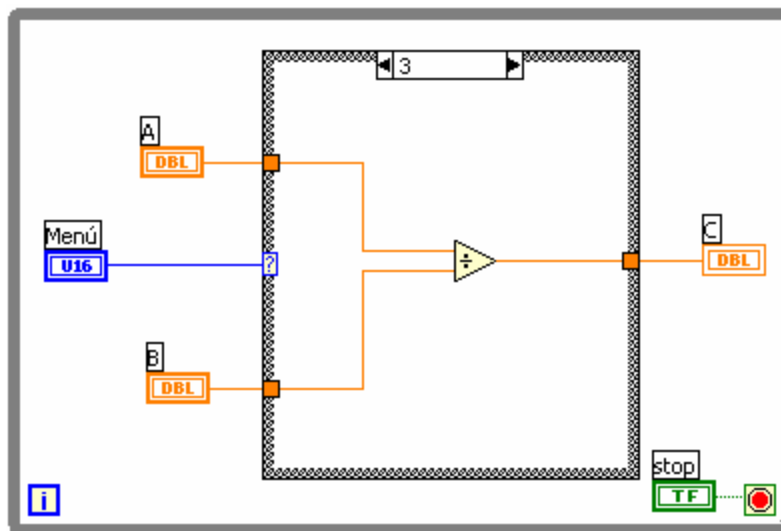


Figura 2.40. Diagrama completo.

FIN EJERCICIO 2.7

2.7 ESTRUCTURA “FORMULA NODE”

Un nodo de fórmula es una caja redimensionable donde se alojan fórmulas matemáticas y lógicas para su evaluación. Los nodos de fórmula tienen la apariencia de la figura 2.41.

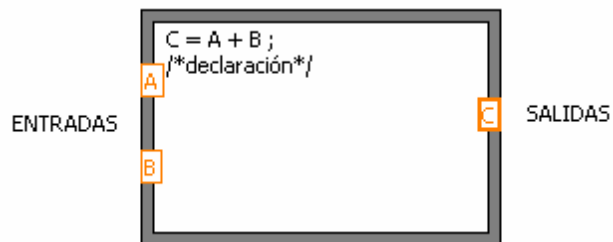


Figura 2.41. Nodo de Fórmula.

Los terminales de entrada y salida son variables numéricas, escalares y reales que se generan adicionándolas desde el menú de la estructura como se muestra en la figura 2.42.

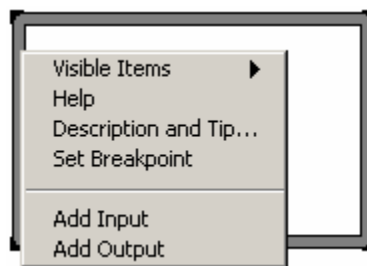


Figura 2.42. Menú de los nodos de fórmula.

La sintaxis interna de los nodos de fórmula es similar a la de C.

Asignación =

Condición ? :

or ||

xor ^
 and &&
 not !
 Relacional == != > < >= <=
 Aritméticas + - * / **

Las funciones aceptadas dentro del nodo de fórmula son:

abs acos acosh asin asinh atan atanh ceil cos cosh cot csc exp
 expm1 floor getexp getman int intrz ln ln2 log log2 max min mod
 rand rem sec sign sin sinh sqrt tan tanh

Dentro del nodo de fórmula se pueden adicionar comentarios encerrándolos dentro de un par slash-asterisco así: `(/*comentario*/)` como se muestra en la figura 2.41.

EJERCICIO 2.8 UTILIZACIÓN DE LOS NODOS DE FÓRMULA

Se realizará en forma convencional y con un nodo de fórmula el cálculo de

$$Z = \frac{X^2}{Y} + Y$$

suponiendo conocidos X y Y. Esta operación en forma convencional

tendría la apariencia de la figura 2.43.

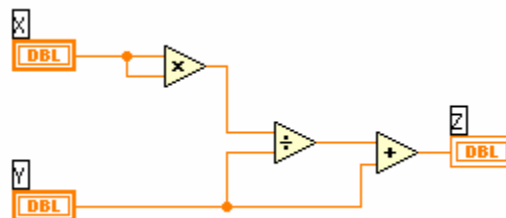


Figura 2.43. Cálculo de Z en forma convencional.

Con la utilización del nodo de fórmula la apariencia del diagrama sería como en la figura 2.44.

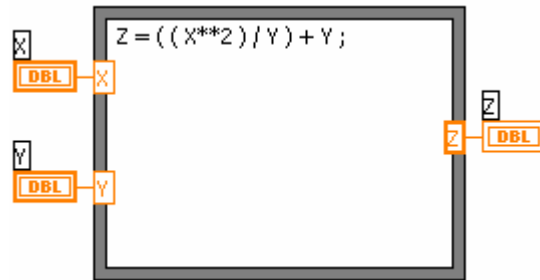


Figura 2.44. Cálculo de Z con un nodo de fórmula.

CUIDADO: El operador (^) que en versiones previas de LabVIEW era el operador de la exponenciación ahora es el operador booleano XOR. El nuevo operador de la exponenciación es (**), como se muestra en la figura 2.44.

En los casos que se requiera evaluar fórmulas matemáticas extensas el nodo de formula es evidentemente más apropiado.

FIN EJERCICIO 2.8

EJERCICIO 2.9 REALIZAR EL EJERCICIO 1.1 CON UN NODO DE FÓRMULA

La figura 1.62 muestra la codificación en G de la ecuación 1.1.

Esta misma codificación se puede realizar con un nodo de fórmula como se muestra en la figura 2.45.

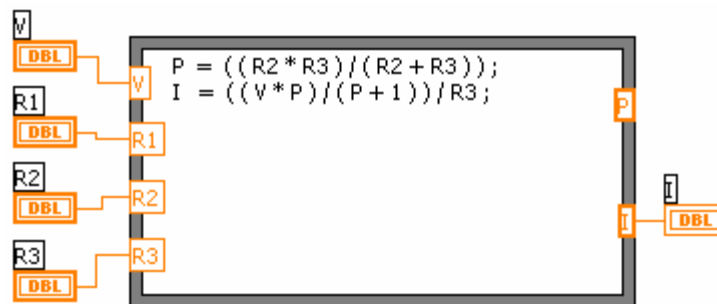


Figura 2.45. Ecuación 1.1 con nodo de fórmula.

Nótese que aunque es posible realizar cálculos intermedios para simplificar la sintaxis de las ecuaciones se debe agregar una salida por cada asignación que se realice. Por ejemplo, en la figura 2.45 se utiliza P para realizar un cálculo intermedio pero se debe agregar como salida así no se utilice.

FIN EJERCICIO 2.9

Los nodos de fórmula se pueden utilizar también con expresiones de decisión. En programación es común encontrar la estructura *IF THEN ELSE*:

C/C++	PASCAL
<pre> IF (condición) { (caso verdadero) } ELSE { (caso falso) } </pre>	<pre> IF (condición) THEN BEGIN (caso verdadero) END; ELSE BEGIN (caso falso) END; END; </pre>

Sin embargo, y aunque la estructura *case* ofrece una solución, en algunos lenguajes como en C es posible utilizar el operador *?*, cuando se desea realizar la asignación de un valor, para simplificar la estructura así:

```
salida = (condición) ? (caso verdadero) : (caso falso) ;
```

Al igual que la estructura *IF* el uso del operador *?* también es anidable.

```
salida=(condición1)?(caso1):(condición2)?(caso2):(condicion3)?(caso3):(caso4);
```

El nodo de fórmula de LabVIEW acepta la sintaxis del operador *?*.

EJERCICIO 2.10 EJERCICIO 2.7 CON UN NODO DE FÓRMULA

Se puede escribir con sólo una línea de código dentro del nodo de fórmula así:

```
C = (M = 0) ? A + B : (M = 1) ? A - B : (M = 2) ? A * B : A / B ;
```

El aspecto que tendría en G se muestra en la figura 2.46.

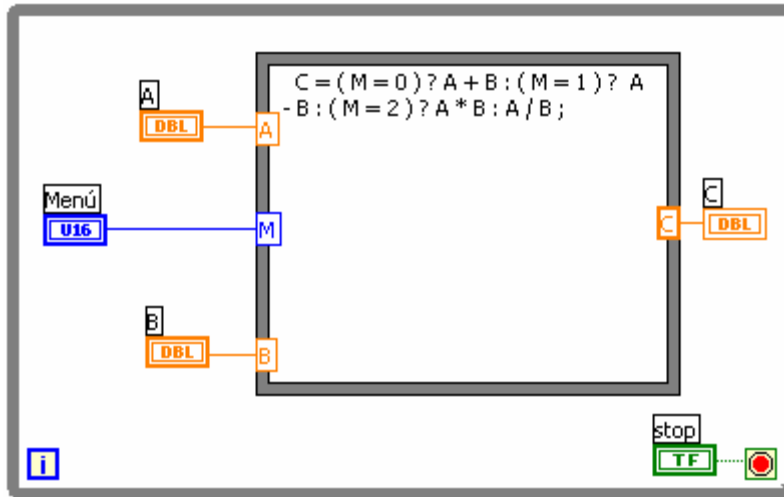


Figura 2.46. Ejercicio 2.7 con nodo de fórmula.

FIN EJERCICIO 2.10

LabVIEW también ofrece una alternativa para estructuras tipo *IF* sin anidamiento que consistan simplemente en determinar si se utiliza un dato u otro. Esto es realizado por la función *Select*, que se muestra en la figura 2.47 y que se puede encontrar en la paleta de funciones en el submenú *comparition*.

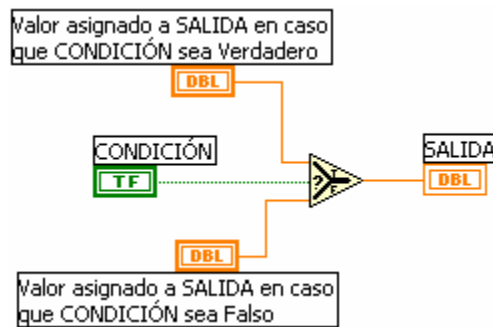


Figura 2.47. Función *Select*.

Obsérvese que “CONDICIÓN” es booleano y determina que dato será asignado a la salida. La salida y los datos a asignar pueden ser cualquier tipo de variable válida de LabVIEW.

EJERCICIO 2.11 LA ECUACIÓN CUADRÁTICA

Para resolver la ecuación $a X^2 + b X + c = 0$, se requiere de tres controles numéricos reales para a , b y c , y de dos indicadores complejos para mostrar las raíces encontradas.

El panel frontal se muestra en la figura 2.48.

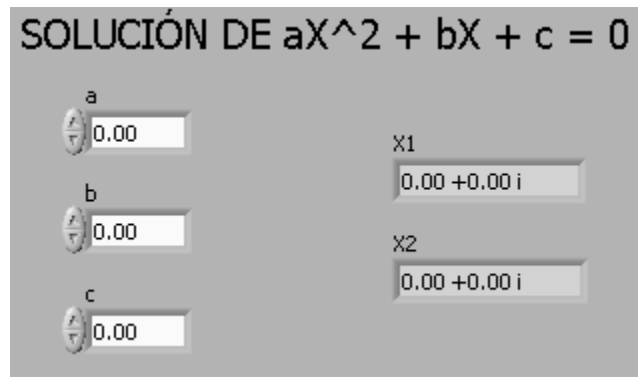


Figura 2.48. Panel frontal para la ecuación cuadrática.

Se sabe que X1 y X2 están dadas por la ecuación 2.1.

$$X_{1,2} = -\frac{b}{2a} \pm \sqrt{\frac{b^2 - 4ac}{4a^2}}$$

Ecuación 2.1. Raíces de la ecuación cuadrática.

Una solución en G para esta ecuación se muestra en la figura 2.49.

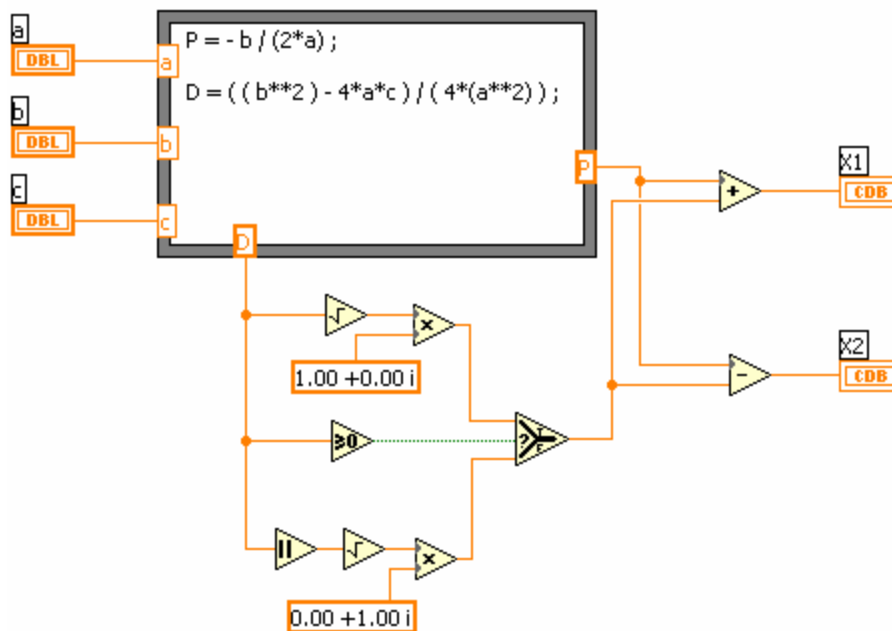


Figura 2.49. Solución de la ecuación cuadrática.

Se utiliza un nodo de fórmula para calcular los términos más útiles.

En LabVIEW la función raíz cuadrada no permite un argumento negativo. Por lo tanto se utiliza la función ≥ 0 para determinar el signo del discriminante y luego la función *Select* para escoger el resultado corregido de la raíz cuadrada.

FIN EJERCICIO 2.11

2.8 ACCIONES MECÁNICAS DE LOS CONTROLES BOOLEANOS

Los controles booleanos pueden ser personalizados con seis diferentes tipos de acciones mecánicas, permitiendo así obtener mayor control sobre las tareas a desarrollar y lograr que los paneles frontales de los instrumentos virtuales puedan parecerse más a los instrumentos físicos.

Para cambiar la acción mecánica de un control booleano se debe seguir los siguientes pasos:

1. Seleccionar el control booleano apropiado de acuerdo con su aplicación.
2. Del menú del control, seleccionar “*Mechanical Action*”.
3. Escoger la acción mecánica requerida.

En la figura 2.50 se observa las acciones mecánicas de un control booleano.

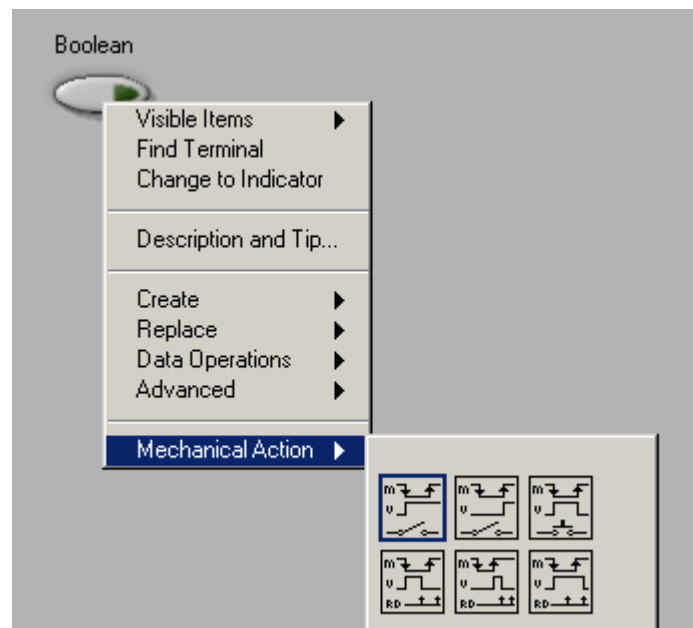


Figura 2.50. Acciones Mecánicas.

Algunos controles booleanos vienen predefinidos por LabVIEW con algún tipo de acción mecánica, ya que fueron diseñados para desempeñar una tarea específica. Los controles “STOP” y “CANCEL” pueden ser un buen ejemplo de ello. La figura 2.51 muestra el control booleano “STOP” con su respectiva acción mecánica.

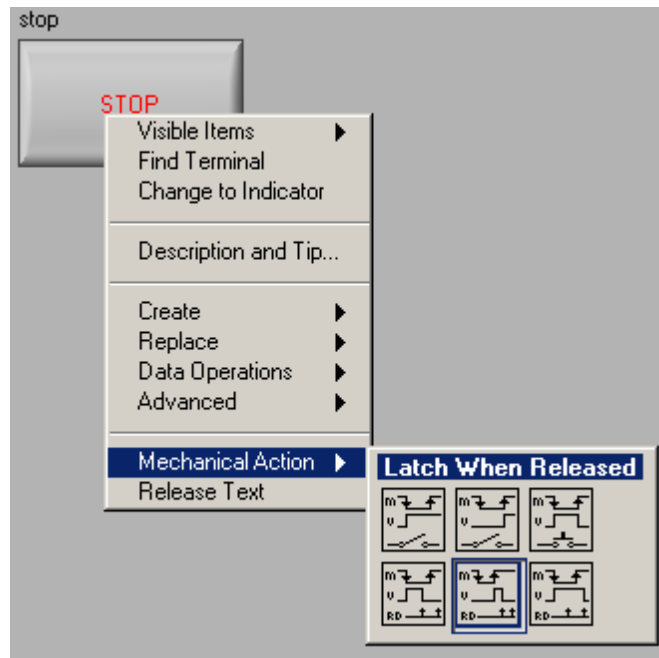


Figura 2.51. Botón STOP con acción mecánica predefinida.

Las acciones mecánicas se dividen en dos tipos: *Switch* y *Latch*.

Los *Switch* son mecanismos que retornan al estado inicial una vez el usuario lo decida. Se muestran en la figura 2.52.



Figura 2.52. Acciones Mecánicas tipo *Switch*.

Los *Latch* son mecanismos que retornan al estado inicial cuando el valor sea leído por el VI o cuando el usuario así lo determine. Son mostrados en la figura 2.53.

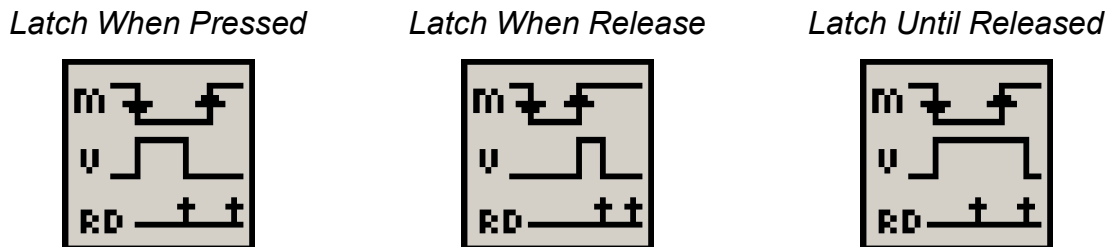


Figura 2.53. Acciones Mecánicas tipo *Latch*.

2.8.1 Switch When Pressed

Este mecanismo permite que los controles booleanos se comporten de manera similar al interruptor de control de una lámpara. Es decir la variable booleana sólo cambia de estado cuando el control es presionado por el usuario. Ver figura 2.54.

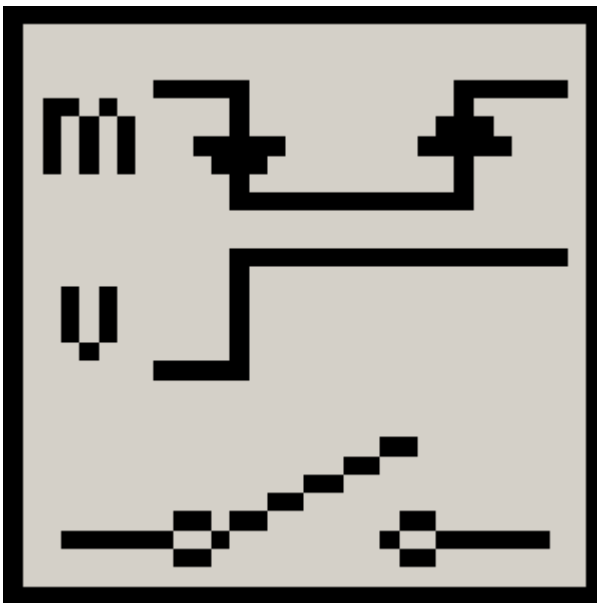


Figura 2.54. *Switch When Pressed*.

En la figura 2.54 la acción del clic del ratón se muestra a través de la letra “m” y la respuesta del control o variable booleana es “v”.

Nótese que cuando en ratón es presionado la variable booleana cambia de estado y permanece así aunque el botón del ratón sea liberado.

La figura 2.55 muestra un VI donde el interruptor está funcionando como un *Switch When Pressed*. Al ejecutar este VI se podrá observar como la salida sólo cambia de estado cuando el usuario lo decida.

Se debe recordar que el botón *STOP* posee una acción mecánica predefinida.

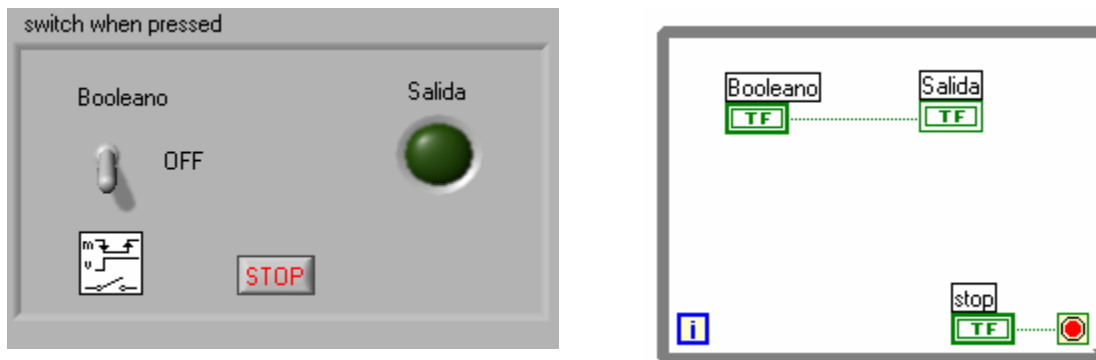


Figura 2.55. Control booleano trabajando como un *Switch When Pressed*.

2.8.2 *Switch When Released*

Este mecanismo actúa sobre la variable sólo cuando el clic del ratón es liberado. El control volverá a su estado anterior sólo cuando el usuario decida volver a accionarlo.



Figura 2.56. *Switch When Released*.

Nótese que la variable “v” NO cambia de estado cuando se presiona el botón de ratón. Sólo lo hace cuando el botón es liberado y permanece en su nuevo estado hasta que el usuario decida volver a presionar y soltar el botón del ratón.

En las figuras 2.57 a 2.59 se muestra la manera de actuar de *Switch When Released* al tratar de controlar una salida.

La figura 2.57 muestra el control booleano configurado con acción mecánica *Switch When Released*. Su estado inicial es *OFF*.

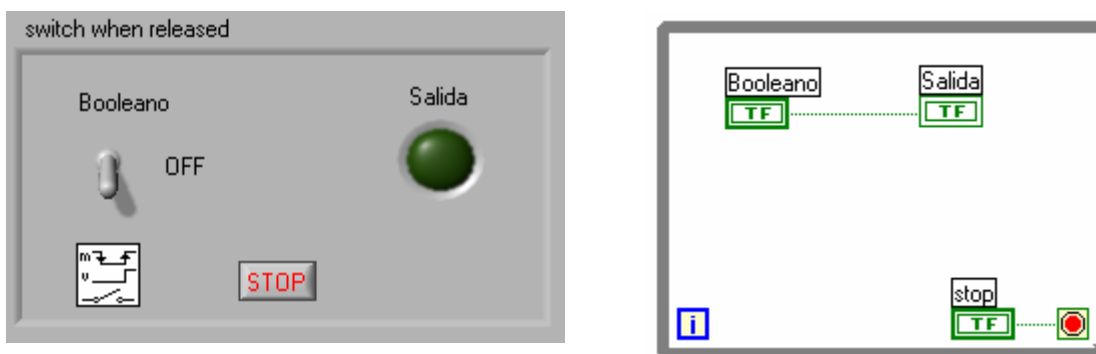


Figura 2.57. Control booleano trabajando como un *Switch When Released*.

En la figura 2.58 el control ha sido accionado a través de un clic, pero el botón del ratón no ha sido liberado, por lo anterior, la salida no ha cambiado su estado. Nótese la posición del control.

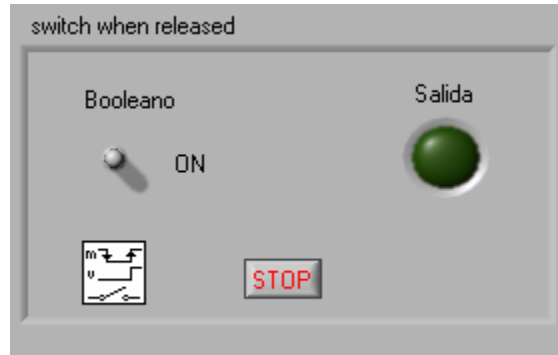


Figura 2.58. Control accionado a través de un clic del ratón.
El botón del ratón no ha sido liberado.

En la figura 2.59 se ha liberado el botón del ratón y por lo tanto la salida ha cambiado su estado.



Figura 2.59. Clic del ratón liberado.

2.8.3 *Switch Until Released*

Cambia el valor de control mientras el botón del ratón es presionado. La figura 2.60 muestra el comportamiento de un control booleano configurado con esta acción mecánica.

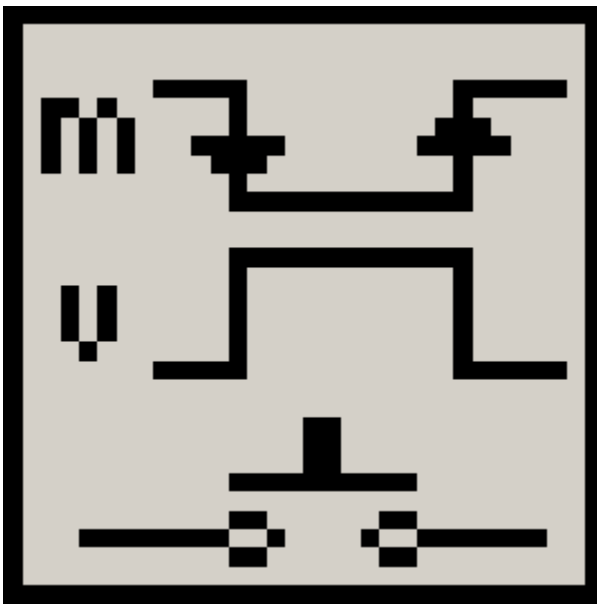


Figura 2.60. *Switch Until Released*.

Al presionar el botón “m” del ratón la variable “v” cambia de estado, pero sólo permanecerá así hasta que el botón del ratón “m” sea liberado.

Esta acción mecánica simula el funcionamiento de un pulsador. Por ejemplo el pulsador que controla un timbre.

La figura 2.61 muestra un VI que al ejecutarse permite observar el funcionamiento de la acción mecánica *Switch Until Released*.

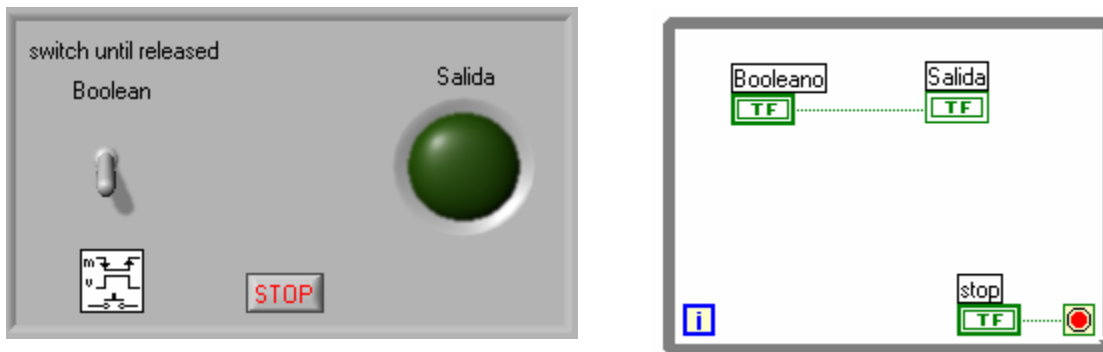
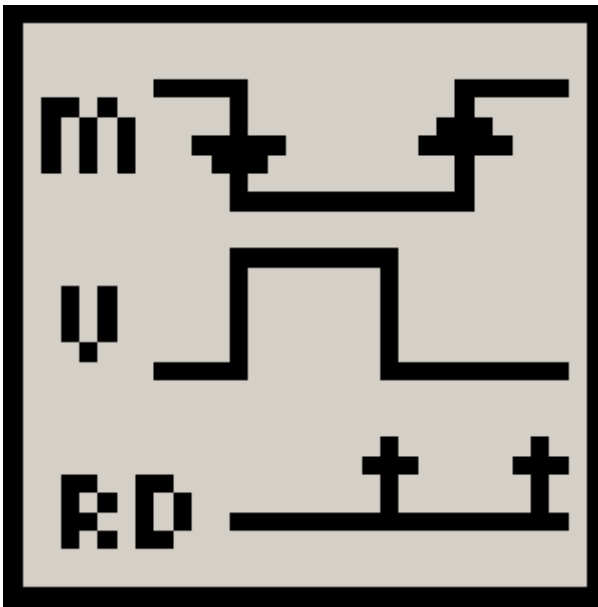


Figura 2.61. Control booleano trabajando como un *Switch Until Released*.

2.8.4 *Latch When Pressed*

Este mecanismo cambia el valor del control cuando el usuario hace un clic sobre el control booleano, este retiene su valor hasta que el VI lo lea, una vez sea leído el control vuelve a su valor por defecto, incluso si el usuario todavía tiene presionado el botón del ratón. Esta acción es especial para truncar la acción de un *While Loop*.

La figura 2.62 muestra el comportamiento de la acción.



“m” muestra el comportamiento del botón del ratón.

“v” muestra el comportamiento de la variable booleana.

“RD” muestra las lecturas que el VI hace a la variable booleana.

Figura 2.62. *Latch When Pressed.*

En la figura 2.63 se muestra un VI que al ejecutarse permite observar el comportamiento de esta acción mecánica.

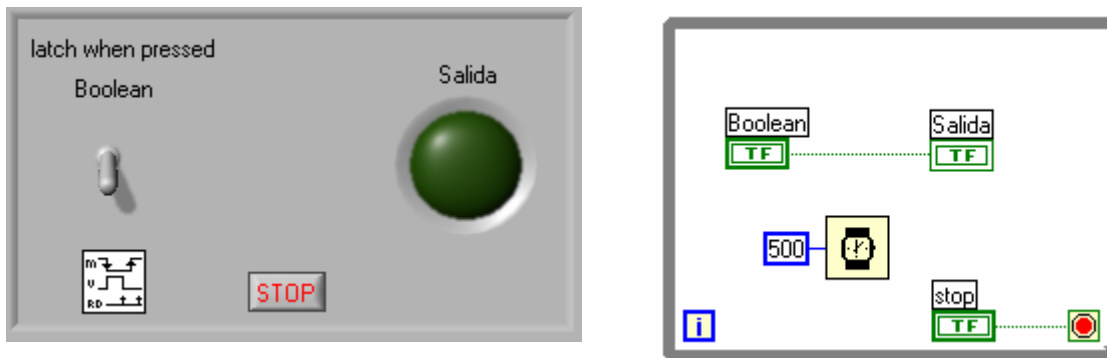


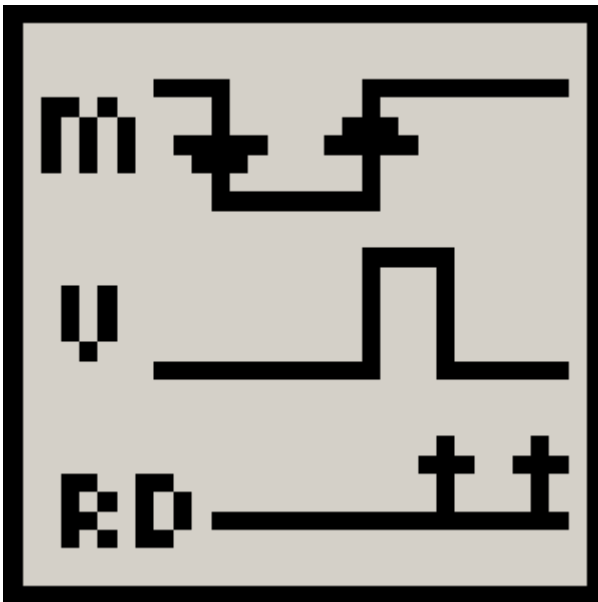
Figura 2.63. Control booleano trabajando como un *Latch When Pressed.*

2.8.5 *Latch When Released*

Este mecanismo de acción trabaja de manera similar al *Latch When Pressed*, con la diferencia de que el control actúa únicamente cuando el usuario libera el clic del ratón. El objeto booleano retendrá su valor hasta que el VI lo pueda leer.

Esta acción trabaja de manera similar a las cajas de botones o sistemas de botones utilizados en algunos sistemas de control. También es muy utilizado para detener ciclos *While* y para los botones en cuadros de diálogo.

La figura 2.64 muestra el comportamiento de la acción.



“m” muestra el comportamiento del botón del ratón.

“v” muestra el comportamiento de la variable booleana.

“RD” muestra las lecturas que el VI hace a la variable booleana.

Figura 2.64. *Latch When Released*.

En la figura 2.65 se muestra un VI que al ejecutarse permite observar el comportamiento de esta acción mecánica.

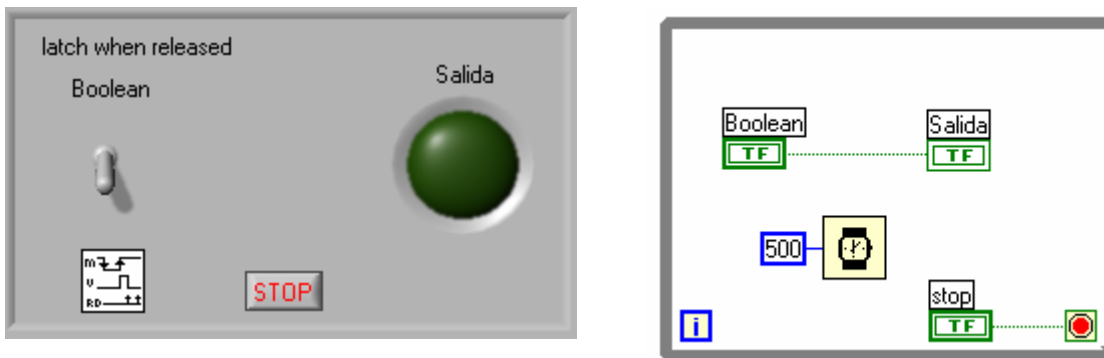
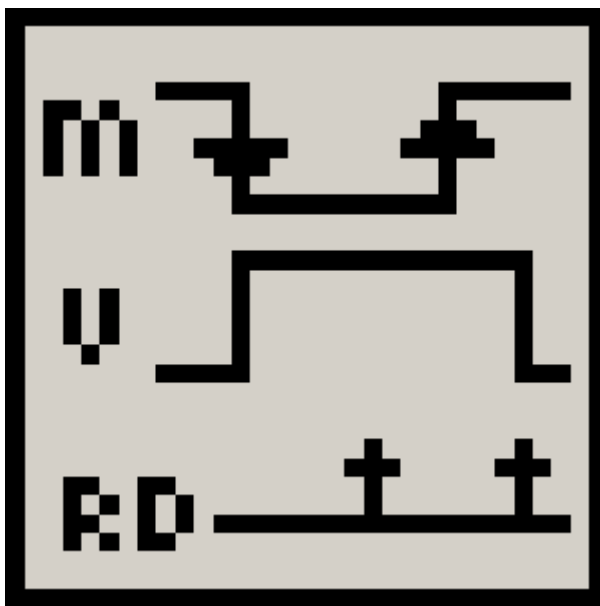


Figura 2.65. Control booleano trabajando como un *Latch When Released*.

2.8.6 Latch Until Released

El mecanismo de acción cambia el valor del control haciendo un clic sobre éste y retiene el valor hasta que el VI lo lee o el usuario lo libera, dependiendo de que fuera lo último que ocurrió.

La figura 2.66 muestra el comportamiento de la acción.



“m” muestra el comportamiento del botón del ratón.

“v” muestra el comportamiento de la variable booleana.

“RD” muestra las lecturas que el VI hace a la variable booleana.

Figura 2.66. *Latch Until Released*.

En la figura 2.67 se muestra un VI que al ejecutarse permite observar el comportamiento de esta acción mecánica.

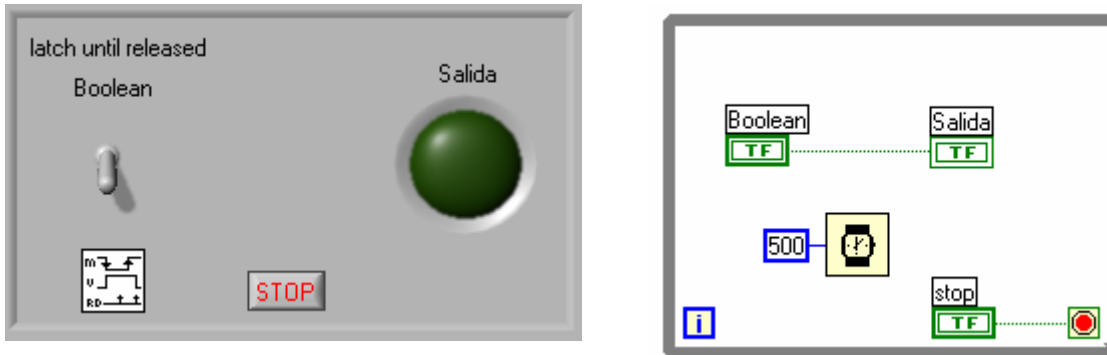


Figura 2.67. Control booleano trabajando como un *Latch Until Released*.

Importante: Los controles booleanos que utilizan las acciones tipo *Latch* no pueden generar variables locales.

EJERCICIO OPCIONAL 2.12 APLICACIÓN DE UNA ESTRUCTURA DE SECUENCIA A LA MEDICIÓN DE ENERGÍA

Un transductor de potencia activa instantánea está conectado a una carga eléctrica. Se requiere calcular la energía consumida por la carga en un intervalo de tiempo definido por el operador a través de un botón de paro.

La energía esta definida según la ecuación 2.2.

$$E(t) = \int_{t1}^{t2} P(t)dt$$

$$E = \sum_{i=0}^{N-1} P_i * (T_i - T_{i-1})$$

Ecuación 2.2. Energía.

La Figura 2.68 muestra el diagrama de bloques que realiza la tarea solicitada.

Del identificador de diagrama de la estructura *sequence* se puede observar que esta posee tres *frames* (0, 1 y 2) de los cuales sólo es visible el cero. Los otros dos *frames* se ven en las figuras 2.70 y 2.71 respectivamente.

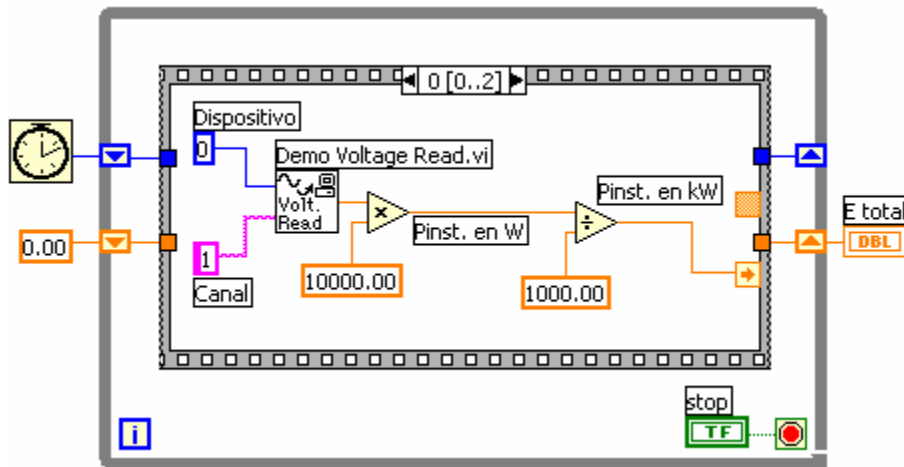


Figura 2.68. Diagrama de bloques.

La estructura *while* es utilizada únicamente para dar continuidad a la ejecución y esperar la orden de PARO.

Para calcular la energía es necesario tomar un tiempo de referencia entregado por la función *Tick Count* antes de realizar cualquier otra tarea.

La función que simula la tarea de recoger los datos enviados por un dispositivo de adquisición se muestra en la figura 2.69 y se puede encontrar en la paleta de funciones en el submenú *Tutorial*.

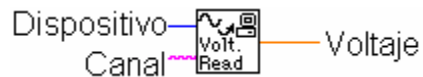


Figura 2.69. Función que lee el sensor.

La salida de esta función está en Voltios y se supondrá que el transductor conectado al canal 0 del dispositivo de adquisición 1 especifica un factor de 10000 W/V, por lo que se debe multiplicar por 10000 para llevar la salida a W y luego dividir por 1000 para llevarla a kW. Desde luego, también se puede simplemente multiplicar por 10.

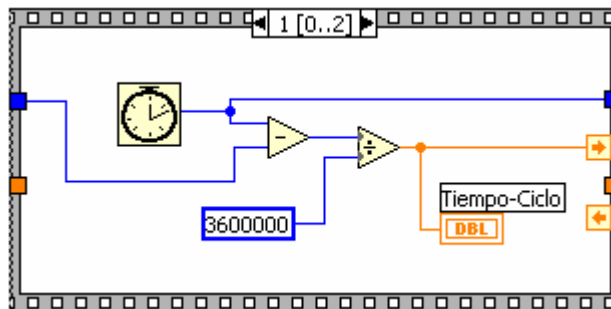


Figura 2.70. Segundo *frame* de la secuencia.

De la ecuación 2.2 se sabe que el consumo de energía en la primera iteración es $E_0 = P_0 \cdot (T_0 - T_{-1})$, donde T_{-1} es el tiempo de referencia de la rutina y T_0 el tiempo tomado para el cálculo de la energía en la primera iteración. La diferencia $(T_{i+1} - T_i)$ es el intervalo de tiempo transcurrido entre iteraciones (Tiempo-Ciclo). Como *Tick Count* tiene su salida en milisegundos se debe dividir por 1000 para llevarla a segundos y por 3600 para llevarla a horas. Esto se puede realizar con una sola división por 3600000.

El tiempo de referencia para cada nuevo ciclo es el tiempo calculado en la iteración anterior, por lo que es necesario generar un *shift register* para calcular la variable Tiempo-Ciclo.

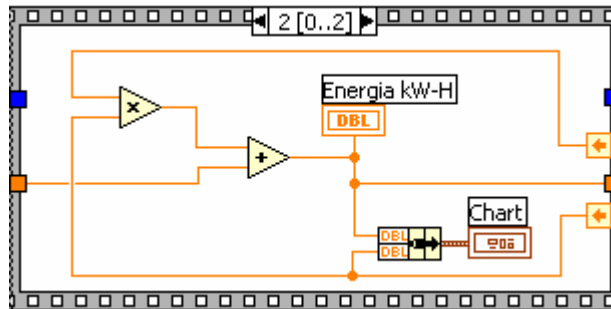


Figura 2.71. Tercer *frame* de la secuencia.

Con el fin de poder acumular los valores de energía de cada lapso de tiempo, es decir la suma de los E_i , es necesario contar con un segundo *shift register*, que se debe inicializar en cero. Obsérvese lo necesario de estos elementos para contar con una historia acumulativa de esta variable energía.

Cuando se desea que un graficador Chart muestre varias curvas distintas, simultáneamente, se debe generar un *Cluster* para lo cual es necesaria la función *bundle*. Esta función se explicará en detalle más adelante y se puede encontrar en **Functions>>Cluster>>Bundle**.

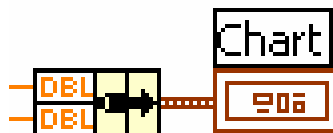


Figura 2.72. Varias gráficas en un mismo chart.

El panel frontal de esta aplicación se observa en la figura 2.73.

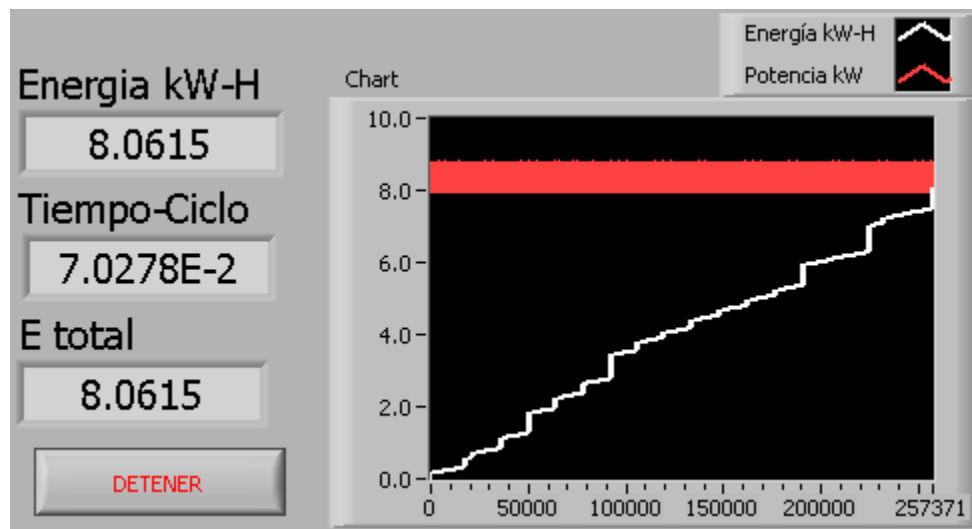


Figura 2.73. Panel Frontal para el ejercicio 2.12.

Obsérvese que el indicador de Energía total sólo se actualizará hasta que se presione el botón de detener, ya que se encuentra localizado por fuera del ciclo.

Para efectos de simulación es conveniente modificar la escala de tiempo para observar resultados más rápidamente. Por ejemplo, se puede reemplazar la constante 3600000 del *frame* 1 por 3600 para que 1 hora real sea 1 segundo de simulación.

FIN EJERCICIO OPCIONAL 2.12

2.9 EJERCICIOS PROPUESTOS

1. Se requiere una variable de iteración para una estructura *For Loop* que se comporte según:
 - a) For (i = 0; i <= 50; i = i+2)
 - b) For (i = 7; i < 10; i = i+0.01)
 - c) For (i = 100; i >= 10; i = i-5)
2. Utilizando “*Digital Thermometer.vi*”, escribir un programa que genere una alarma cuando la temperatura leída esté por encima de un valor máximo permitido o por debajo de un valor mínimo permitido.
3. Hacer que un *While* se comporte como un *For*. Es decir que se detenga en un número definido de iteraciones o cuando el usuario presione parar.
4. Resolver la ecuación cuadrática utilizando únicamente un nodo de fórmula. Optimizar el código del nodo a sólo 4 instrucciones. Recordar que el nodo de formula no puede operar con complejos, por lo que las salidas serán las partes reales e imaginarias de las raíces.
5. Hacer que una estructura *While Loop* se comporte como un **WHILE..DO** y no como un **DO..WHILE**. Esto es, que la condición de ejecución del ciclo sea revisada al inicio de cada iteración y no al final.

3. ARREGLOS Y CLUSTERS.

3.1 OBJETIVO

Estudiar los arreglos y los clusters de LabVIEW, su utilidad y las principales herramientas para su utilización y manejo.

3.2 ARREGLOS

Un arreglo es una colección ordenada de variables del mismo tipo. Puede tener una o varias dimensiones y hasta $2^{31}-1$ elementos por dimensión.

Para crear un control o indicador tipo arreglo se debe seguir estos pasos:

1. Poner en el Panel Frontal un contenedor de arreglos. Este se encuentra en la paleta de controles en el submenú **Array&Cluster >>Array**, como se observa en la figura 3.1.
2. Poner dentro del contenedor un control o un indicador del tipo de dato que se desee el arreglo. No es posible poner dentro del contenedor ni gráficas ni contenedores de arreglos.



Figura 3.1. Paleta de controles *Array&Cluster*.

La figura 3.2 muestra la secuencia descrita para crear un arreglo de controles numéricos.

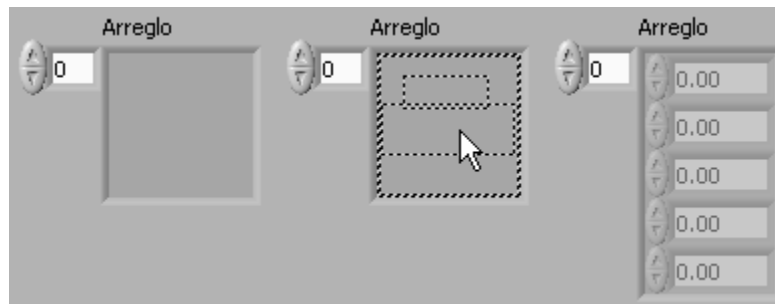


Figura 3.2. Secuencia de creación de un arreglo numérico.

Los elementos de los arreglos están ordenados desde la posición 0 hasta la posición $N-1$ donde N es el tamaño del arreglo.

La figura 3.3 muestra un arreglo booleano de 6 elementos.



En general :

$$A[i] = [a_0, a_1, a_2, \dots, a_i, \dots, a_{N-1}];$$

Figura 3.3. Organización de los elementos.

En la esquina superior izquierda del contenedor de arreglos se observa un control numérico denominado **indicador de índice** (*index display*). Con él se puede controlar qué elemento del arreglo se visualizará en la primera posición del contenedor. Además con la herramienta de posición se puede redimensionar el número de elementos visibles. La figura 3.4 muestra un mismo arreglo numérico de 4 elementos en diferentes presentaciones.

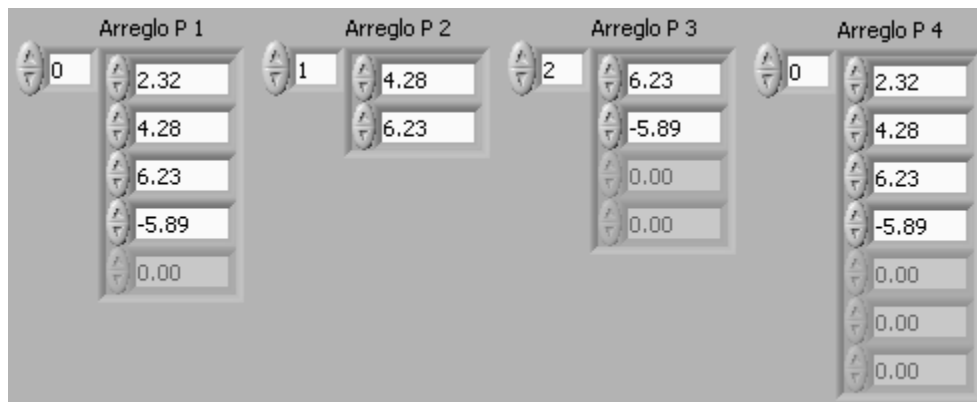


Figura 3.4. Arreglo numérico en diferentes presentaciones.

Como se puede observar, el tamaño del arreglo es independiente de la presentación que el usuario dé a este. Cuando el tamaño visualizado es mayor

que el tamaño del arreglo, o que el número de elementos restantes, las posiciones no utilizadas estarán deshabilitadas y se mostrarán atenuadas (*grayed-out*), como en los casos P1, P3 y P4 de la figura 3.4.

Con la herramienta de posición también es posible redimensionar el *index display* con el objetivo de agregar dimensiones al arreglo. Por defecto los arreglos son de una dimensión pero es posible adicionar múltiples dimensiones.

La figura 3.5 muestra la secuencia para crear un arreglo numérico de dos dimensiones. En este caso el primer *index display* es para las filas y el segundo para las columnas.

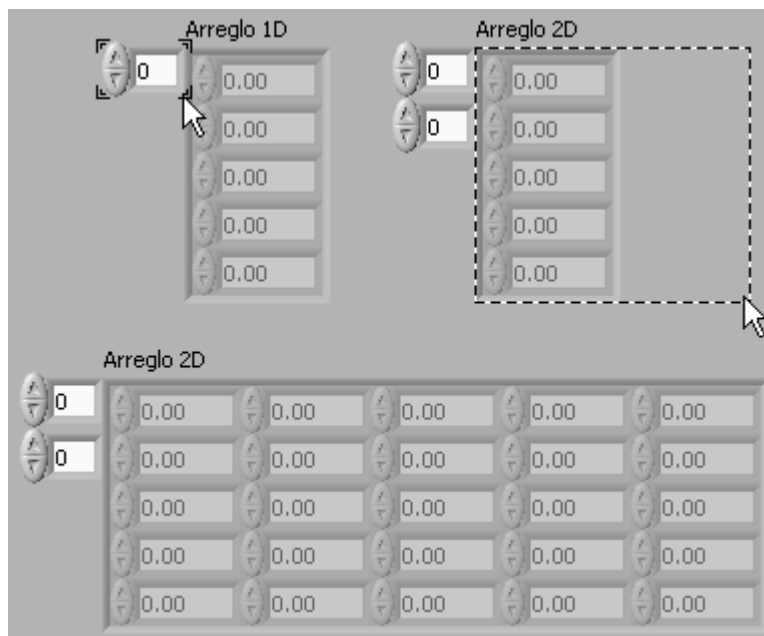


Figura 3.5. Adición de una dimensión al arreglo.

El solo hecho de adicionar un elemento al *index display* es suficiente para obtener un arreglo de dos dimensiones. Luego, si se desea, se puede ampliar el área visible del arreglo con la herramienta de posición como se muestra en la figura 3.5.

La figura 3.6 muestra dos posibles presentaciones de un arreglo numérico de dos dimensiones de tamaño 3x4 (3 filas y 4 columnas).

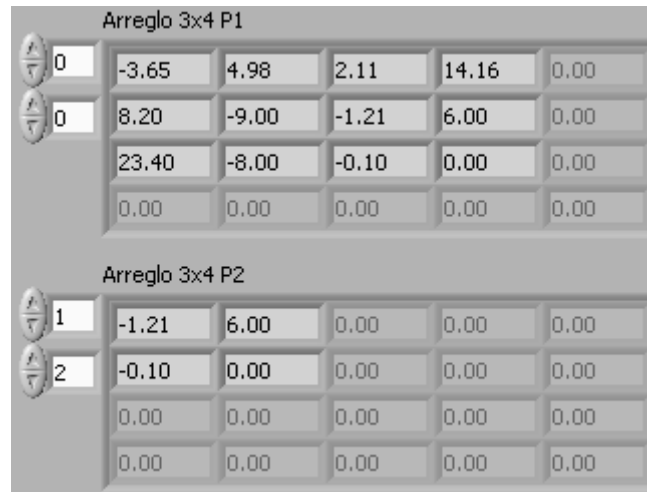


Figura 3.6. Dos presentaciones de un arreglo 2D 3x4.

En la ventana de diagramación un arreglo posee un terminal único. Se distinguen de los terminales de escalares en los corchetes que encierran el tipo de dato. Estos corchetes se hacen más gruesos para arreglos de 2 dimensiones. La figura 3.7 muestra terminales de arreglo para varios tipos de datos.

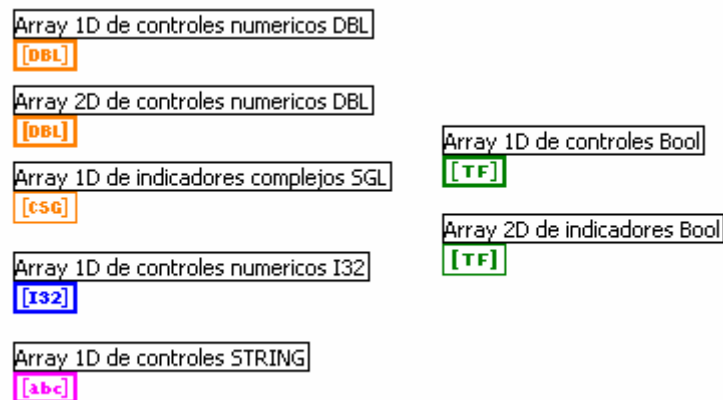


Figura 3.7. Terminales de arreglo.

En el menú *Array* de la paleta de funciones se puede encontrar todas las funciones necesarias para manipular los arreglos. La figura 3.8 enseña la paleta de funciones de arreglos.

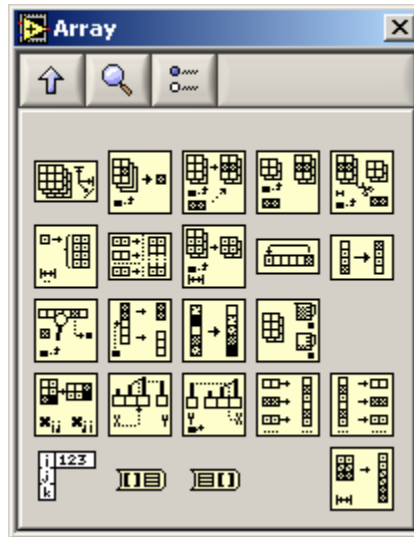


Figura 3.8. Funciones de arreglos.

A continuación se describe las funciones de esta paleta.

Array Size:

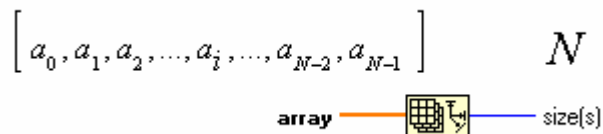


Figura 3.9. Función *Array Size*.

Retorna el tamaño N del arreglo de entrada. Si este es de n dimensiones, la salida será un vector de n elementos donde cada uno mostrará el tamaño de cada dimensión.

Index Array:

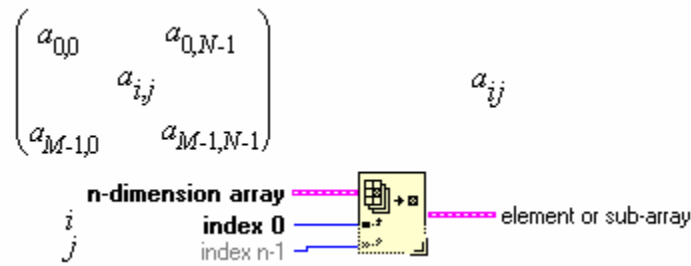


Figura 3.10. Función *Index Array*.

Retorna el elemento indicado de un arreglo. Se debe adicionar tantos elementos de índice como dimensiones tenga el arreglo.

La salida de esta función también puede ser un arreglo cuando se cablean sólo algunos de los índices.

Replace Array Subset:

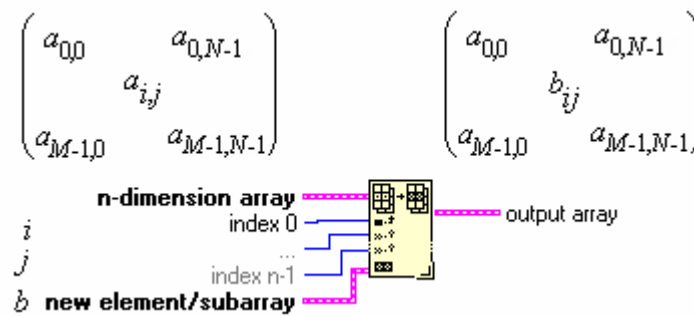


Figura 3.11. Función *Replace Array Subset*.

Reemplaza el elemento indicado en los terminales de índice por el elemento que este cableado al terminal *new element/subarray*. Se debe agregar tantos terminales de índice como dimensiones tenga el arreglo.

Si se desea reemplazar varios elementos, es decir una porción de arreglo, basta con cablear el nuevo arreglo al terminal *new element/subarray* y establecer los índices correspondiente a la porción de arreglo a reemplazar.

Si no se cablea una dimensión, la función reemplazará todos los elementos en esta dimensión comenzado en la posición 0.

El nuevo elemento o arreglo debe ser del mismo tipo y dimensión del arreglo inicial.

Al igual que la mayoría de funciones en LabVIEW, es polimórfica y puede trabajar con arreglos de cualquier tipo y dimensión.

Insert Into Array:



Figura 3.12. Función *Insert Into Array*.

Inserta un arreglo o un elemento en la posición especificada por el terminal de índice. Cuando no se cablea este terminal, el arreglo o elemento se inserta al final del arreglo de entrada.

Delete From Array:

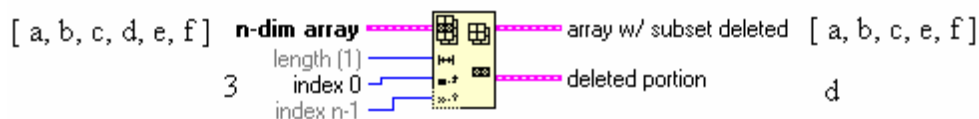


Figura 3.13. Función *Delete From Array*.

Elimina un elemento o arreglo del arreglo de entrada. Además de devolver el arreglo editado, también devuelve la porción de arreglo eliminada.

Initialize Array:

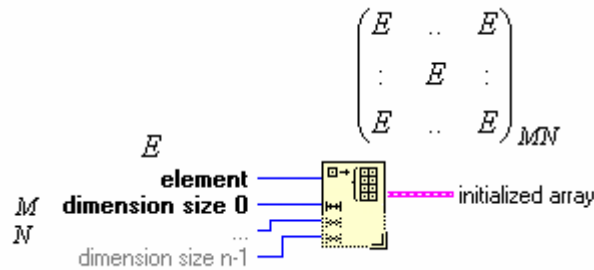


Figura 3.14. Función *Initialize Array*.

Retorna un arreglo de n dimensiones, donde todos los elementos serán inicializados con el valor y tipo de dato cableado en *element*.

Se debe adicionar tantos elementos de índice como dimensiones se desee en la salida.

Build Array:

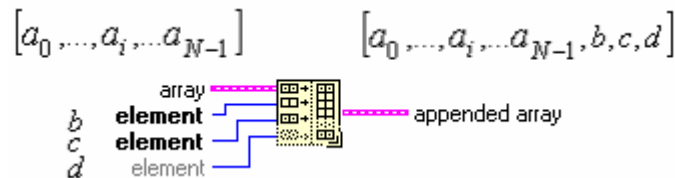


Figura 3.15. Función *Build Array*.

Construye un arreglo de n dimensiones con los elementos de entrada que pueden ser de n o de $n-1$ dimensiones.

Cuando todas las entradas poseen dimensión n , la salida será de dimensión $n+1$. Ahora bien, si se desea que la salida sea un arreglo de dimensión n conformado por la concatenación de todas las entradas, entonces se debe seleccionar *Concatenate Inputs* del menú de la función.

Array Subset:

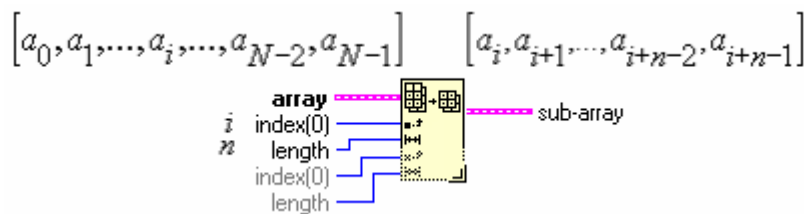


Figura 3.16. Función *Array Subset*.

Retorna una porción del arreglo de entrada (sub-arreglo).

index(0) indica la posición de inicio y *length* el número de elementos que tendrá el sub-arreglo. LabVIEW añadirá tantos elementos de índice como dimensiones tenga el arreglo de entrada.

Rotate 1D Array:

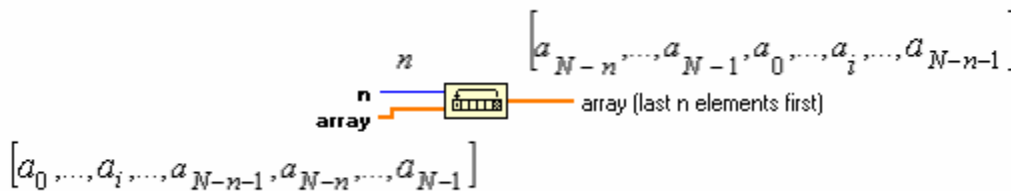


Figura 3.17. Función *Rotate 1D Array*.

Envía los últimos n elementos de un arreglo al principio del mismo. Si n es un número negativo, se rotarán los primeros n elementos al final del arreglo. Si n es 0, N ó $-N$ el vector de salida es igual al de entrada.

Reverse 1D Array:

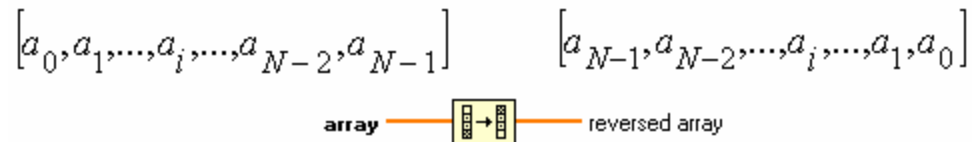


Figura 3.18. Función *Reverse 1D Array*.

Reordena todas las posiciones del arreglo desde el último elemento hasta el primero.

Search 1D Array:

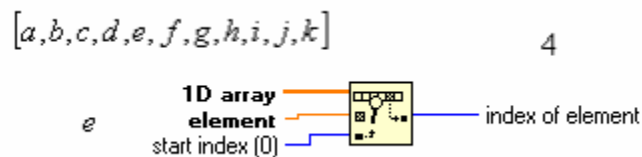


Figura 3.19. Función *Search 1D Array*.

Busca *element* dentro de un arreglo de una dimensión comenzando desde *start index* y retorna la posición donde se encontró. De no hallar ninguna coincidencia retorna "- 1".

Split 1D Array:

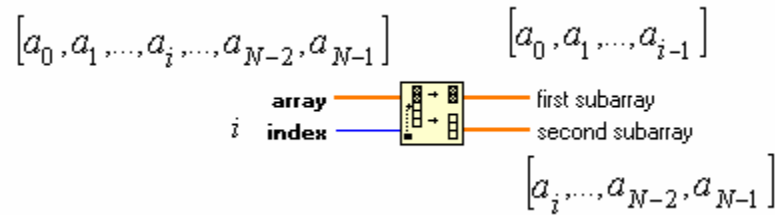


Figura 3.20. Función *Split 1D Array*.

Divide el arreglo de entrada en el elemento *index* regresando dos sub-arreglos como se muestra en la figura 3.20.

Sort 1D Array:

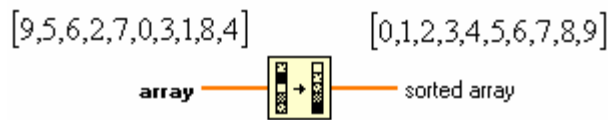


Figura 3.21. Función *Sort 1D Array*.

Ordena un arreglo de forma ascendente si es numérico o alfabéticamente si es tipo cadena.

Array Max & Min:

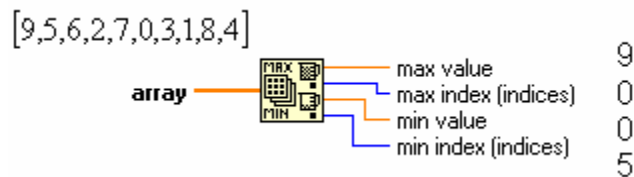


Figura 3.22. Función *Array Max & Min*.

Retorna los valores máximo y mínimo de un arreglo numérico con sus respectivas posiciones.

Transpose 2D Array:

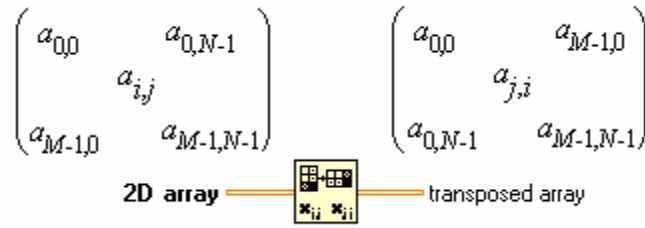


Figura 3.23. Función *Transpose 2D Array*.

Retorna la transpuesta de una matriz o arreglo de dos dimensiones.

Interpolate 1D Array:

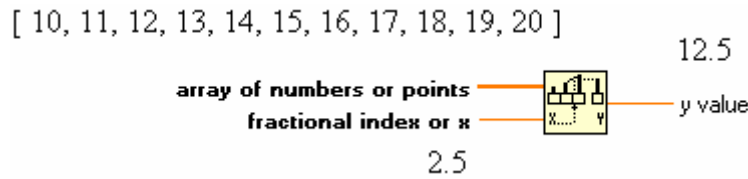


Figura 3.24. Función *Interpolate 1D Array*.

Retorna un valor Y como resultado de la interpolación sobre el arreglo de entrada de tamaño N en el punto X.

Los elementos de las abscisas son los números naturales desde 0 hasta N-1, y X el número real donde se desea evaluar el arreglo.

Threshold 1D Array:

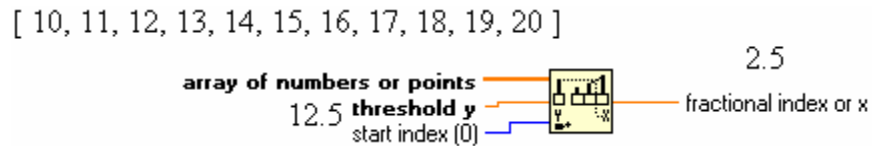


Figura 3.25. Función *Threshold 1D Array*.

Retorna un valor X que es igual al índice fraccionario equivalente a un valor Y del arreglo de entrada.

Interleave 1D Arrays:

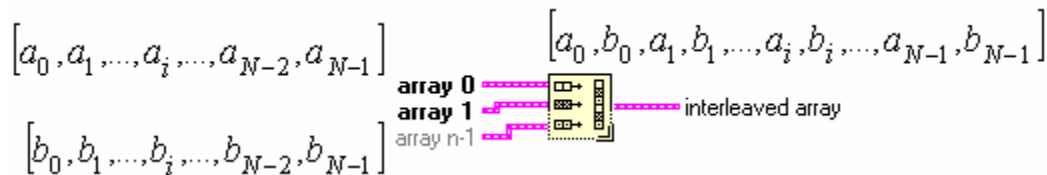


Figura 3.26. Función *Interleave 1D Arrays*.

Forma un arreglo compuesto por los elementos intercalados de los arreglos de entrada como se muestra en la figura 3.26.

Decimate 1D Arrays:

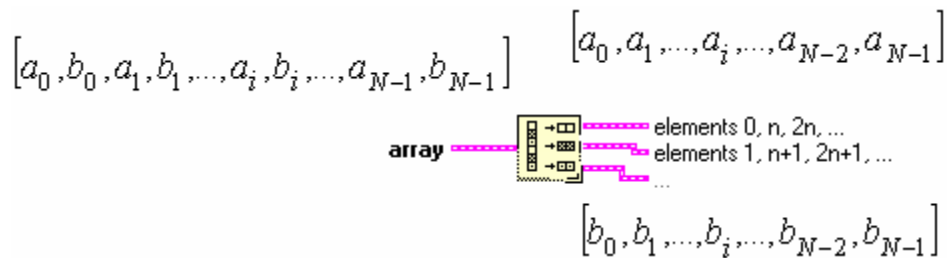


Figura 3.27. Función *Decimate 1D Array*.

Distribuye los elementos de un arreglo de entrada en n arreglos de salida, poniendo los primeros n elementos del arreglo de entrada como primera posición de cada arreglo de salida y así sucesivamente. Figura 3.27. Su funcionamiento es inverso al de la función *Interleave 1D Arrays*.

Array Constant:

Figura 3.28. Arreglo Constante.

Es un contenedor de arreglos al que se puede agregar cualquier tipo de constante a excepción de otros contenedores de arreglo y puede ser de una o varias dimensiones. Sus propiedades de presentación son idénticas a las de los arreglos, controles o indicadores, del panel frontal.

EJERCICIO 3.1 DE UN ARREGLO 1D, GENERAR LAS SALIDAS REQUERIDAS

- Tamaño del Arreglo.
- Valor máximo y la posición en que se encuentra.
- Valor mínimo y la posición en que se encuentra.
- Arreglo en orden ascendente.
- Arreglo en orden descendente.
- Cuarto elemento del arreglo, es decir la posición 3.

Para realizar esta tarea se ha construido el panel frontal de la figura 3.29.



Figura 3.29. Panel frontal del ejercicio 3.1.

Los marcos de fondo que separan la entrada de las salidas en este panel son decoraciones obtenidas del menú *Decorations* de la paleta controles.

El diagrama de bloques de este ejercicio es una simple aplicación de las funciones de arreglo anteriormente descritas. Se puede observar en la figura 3.30.

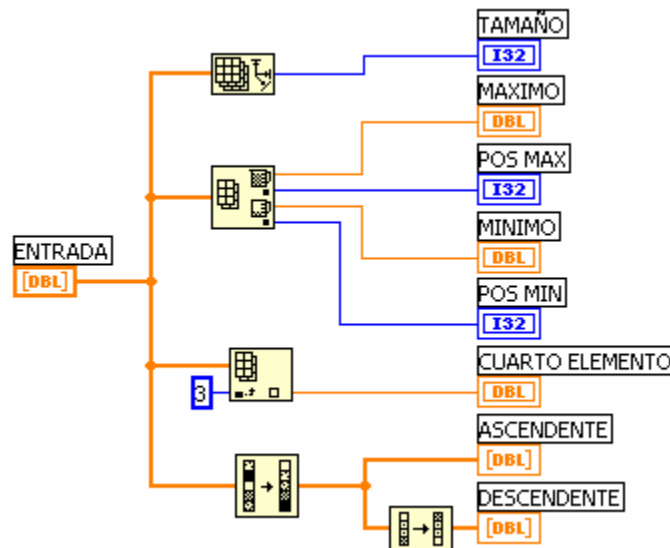


Figura 3.30. Diagrama de bloques del ejercicio 3.1.

Se puede observar que el terminal del arreglo “Entrada” alimenta cuatro funciones tomadas de *Functions>>Array* que cumplen las tareas específicas solicitadas.

En el caso de *Sort 1D Array* se obtiene el arreglo en orden ascendente y luego por medio de la función *Reverse 1D Array* se puede obtener el arreglo en orden descendente.

FIN EJERCICIO 3.1

EJERCICIO 3.2 EXTRAER DATOS DE UN ARREGLO 2D

Dado un arreglo 2D de 1000x2 donde cada columna posee 1000 datos de una señal, obtener una gráfica de los datos de una de las columnas en una *waveform chart*.

El panel frontal se observa en la figura 3.31.

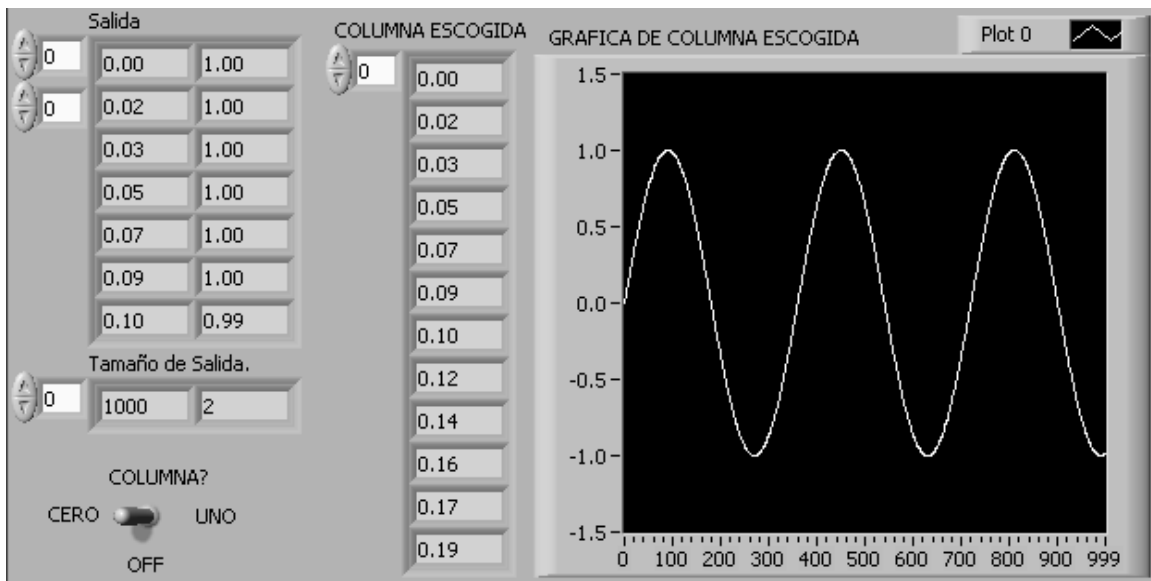


Figura 3.31. Panel Frontal del ejercicio 3.2.

El diagrama de bloques se observa en la figura 3.32.

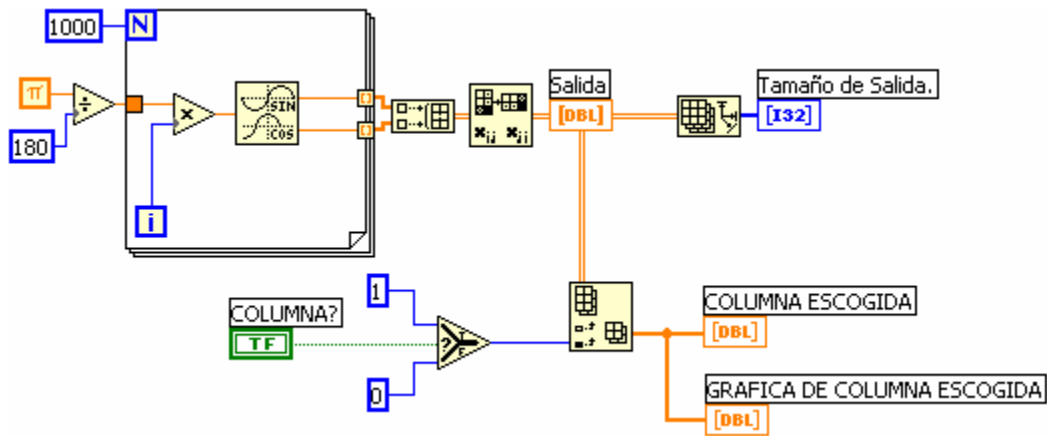


Figura 3.32. Diagrama de bloques del ejercicio 3.2.

En la figura 3.32 se puede observar tres diferentes formas de cables, logrando así establecer una diferencia en cuanto al número de dimensión que posee el dato respectivo.

La figura 3.33 muestra el aspecto de los cables según el número de dimensiones de cada dato.

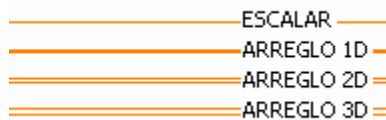


Figura 3.33. Tipos de cables según la dimensión del dato.

La tarea de extraer una columna de la matriz “Salida” se realiza con la función *Index Array*. Para extraer toda una columna no se cablea el índice de filas.

Obsérvese con atención la salida de los datos de la estructura *FOR*. Estos túneles tienen la característica especial de ordenar los datos de cada iteración del ciclo en un arreglo de salida con todos los datos generados en cada iteración. Esta opción es válida también en los ciclos *WHILE*. Sin embargo, la opción allí esta

deshabilitada por defecto. Para habilitarla se debe escoger la opción *Enable Indexing* del menú de la estructura, tal como la muestra la figura 3.34.

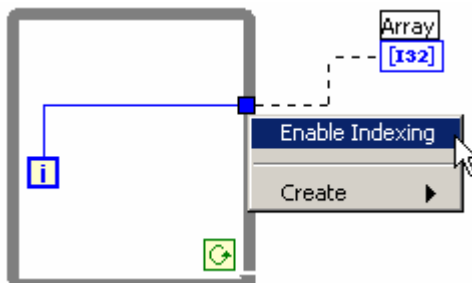


Figura 3.34. Habilitación del indexado en un ciclo *WHILE*.

En los túneles de entrada, la opción también es válida. Es muy útil cuando se requiere disponer de cada elemento del arreglo en la respectiva iteración.

En el caso del ciclo *FOR*, el *indexing* está habilitado por defecto y permite obviar el terminal N del ciclo *FOR*. Sin embargo, si se cablea, el ciclo se detendrá cuando se terminen los datos del arreglo o cuando se cumpla N, lo que ocurra primero.

FIN EJERCICIO 3.2

3.3 CLUSTERS

Un *CLUSTER* es una colección ordenada de variables que pueden ser de diferentes tipos.

Es análogo a los registros de PASCAL o las estructuras de C/C++.

Para crear un control o indicador tipo *clusters* se debe:

- 1) Poner en el panel frontal un contenedor de *clusters* localizable en la paleta de controles en el submenú **Array&Cluster>>Cluster**. Este es una caja redimensionable donde se puede alojar otros elementos.
- 2) Poner dentro del contenedor los controles o indicadores deseados. Sin embargo todos los elementos del cluster serán controles o indicadores de acuerdo con la naturaleza del primer objeto que se ponga.



Figura 3.35. Cluster vacío y con controles.

Los *clusters* poseen un único terminal en la ventana de diagramación. La figura 3.36 muestra el terminal correspondiente a un cluster y las funciones para

manipularlos que se pueden encontrar en el menú *Clusters* de la paleta de funciones.



Figura 3.36. Terminal de un *Cluster* y funciones para *clusters*.

Las principales funciones de esta paleta son *Unbundle* y *Bundle*.

Unbundle:

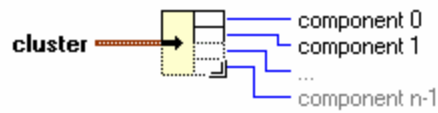


Figura 3.37. Función *Unbundle*.

Permite separar cada una de las variables de un *cluster* para poderlas utilizar independientemente dentro de un diagrama.

Bundle:

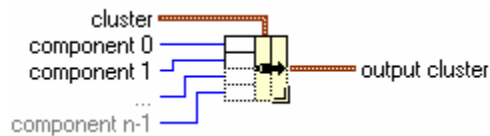


Figura 3.38. Función *Bundle*.

Realiza la tarea contraria a *Unbundle*, es decir, crea un *cluster* a partir de varios componentes independientes. Esta función también se utiliza para reemplazar componentes de un *cluster* existente.

La figura 3.39 muestra cómo la función *Unbundle* separa todas las variables de un *cluster* y cómo la función *bundle* puede crear un *cluster*. Sin embargo para estos procedimientos se requiere conocer el orden en que se encuentran las variables dentro del *cluster*.

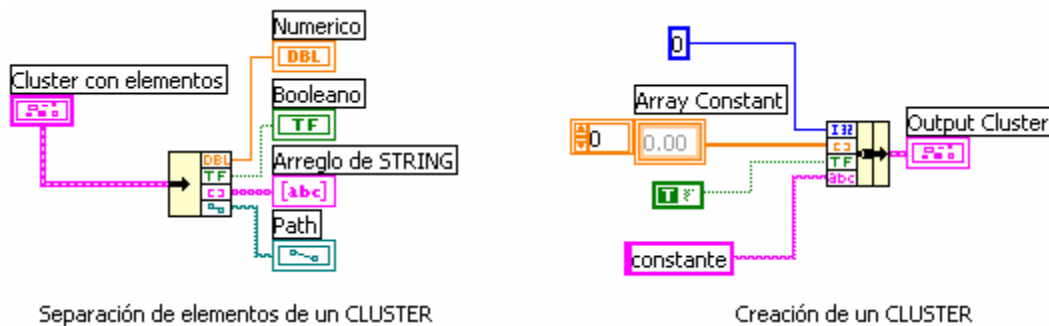


Figura 3.39. Uso de las funciones *Unbundle* y *Bundle*.

Dentro de un *cluster* las variables están numeradas. Este orden se requiere en las funciones *Bundle* y *Unbundle* para conocer cual es el terminal al que corresponde cada variable.

El orden del *cluster* se puede acceder en el panel frontal por el menú del objeto seleccionando la opción *Reorder Controls In Cluster*. Figura 3.40.

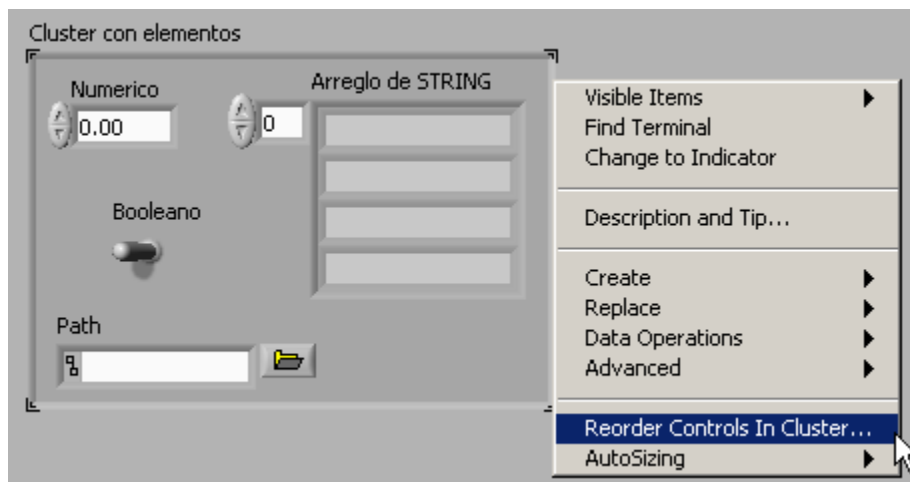


Figura 3.40. Opción para ver y editar el orden del *cluster*.

Esta acción mostrará el orden de cada elemento y permitirá modificarlo como se muestra en la figura 3.41.

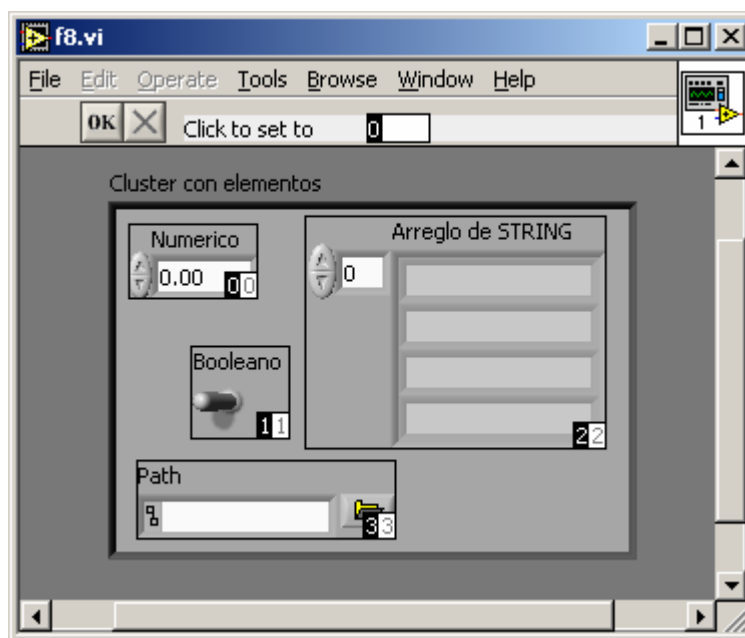


Figura 3.41. Edición del orden de un *cluster*.

Los números en fondo blanco muestran la posición actual de los elementos, mientras los de fondo negro muestran la nueva posición que tomarán al presionar “OK”.

Para editar el índice de cada elemento se escribe el valor deseado y luego se hace un clic sobre el objeto que se desea reordenar. Al terminar se hace clic en “OK” para aceptar los cambios o en “X” para cancelar.

Si un *cluster* contiene sólo elementos del mismo tipo puede ser convertido a un arreglo de 1D por medio de la función “*Cluster To Array*”.



Figura 3.42. Función *Cluster To Array*.

Ahora bien, cualquier arreglo de 1D puede ser convertido en un *cluster* por medio de la función “*Array To Cluster*”.



Figura 3.43. Función *Array To Cluster*.

En este caso es necesario definir el tamaño del cluster a crear. Esto se logra seleccionando *Cluster Size* del menú de la función.

El índice de cada elemento dentro del arreglo será equivalente a su orden dentro del *cluster* y viceversa.

EJERCICIO 3.3 UTILIZACIÓN DE LOS CLUSTERS

Un control tipo *cluster* contiene los siguientes elementos:

- Un control numérico DBL llamado A.
- Dos controles booleanos llamados C y D.
- Una matriz cuadrada llamada M.

Se desea obtener un *cluster* indicador que contenga:

- Un indicador numérico igual al cuadrado de A.
- Un indicador booleano igual a “C or D”
- Una matriz T igual a la transpuesta de M.

El panel frontal de la aplicación tendrá la forma de la figura 3.44.

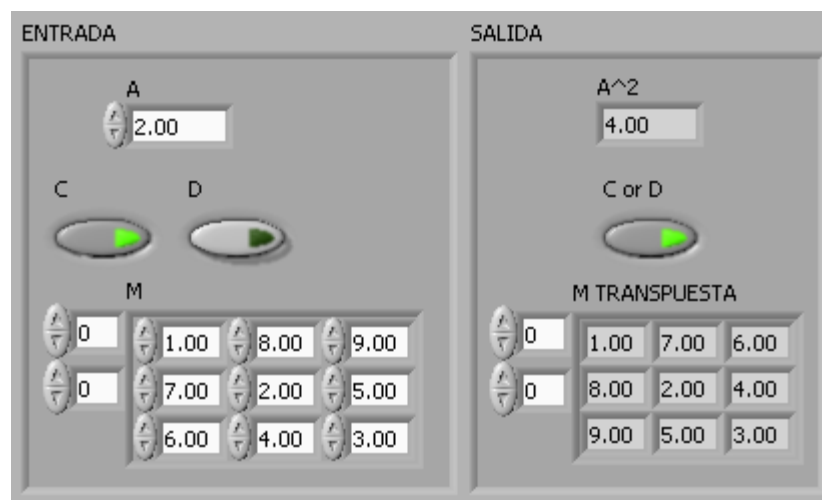


Figura 3.44. Utilización de los *Clusters*.

Lo primero que se debe realizar es separar las variables del *cluster* de entrada. Esto se logra con la función *Unbundle*.

Luego, como ya se dispone de las variables en el diagrama se realizan las tareas necesarias. En este caso se utilizan las funciones *Multiply*, *Or* y *Transpose 2D Array* para tal fin. Por último se debe formar el nuevo *cluster* de salida con la función *Bundle*. Todo el procedimiento se observa en la figura 3.45.

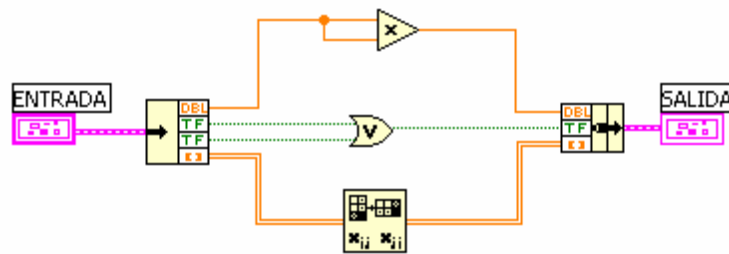


Figura 3.45. Diagrama de bloques del ejercicio 3.3.

Es importante recordar el orden de cada *cluster*.

Existe además dos funciones análogas a *Unbundle* y *Bundle* pero que funcionan de acuerdo a los nombres de las variables. Estas funciones son “*Unbundle by name*” y “*bundle by name*” respectivamente y se encuentran en el mismo menú de la paleta de funciones. La ventaja de utilizar dichas funciones radica en poder observar las variables por sus etiquetas y no por el tipo de dato. La figura 3.46 muestra una solución al ejercicio reemplazando *Unbundle* por la función *Unbundle by name*.

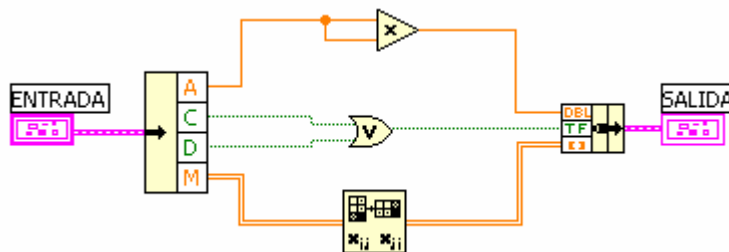


Figura 3.46. Utilización de la función *Unbundle by Name*.

La función *Bundle by name* sin embargo sólo puede ser utilizada para reemplazar directamente los elementos de un *cluster* previamente existente. La figura 3.47 muestra como se reemplaza en el *cluster* de entrada las variables A y D por los valores 3.25 y *True* respectivamente.

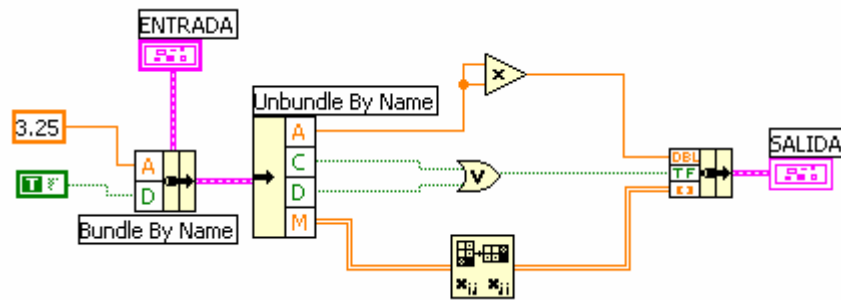


Figura 3.47. Uso de la función *Bundle by Name*.

FIN EJERCICIO 3.3

3.4 EJERCICIOS PROPUESTOS

1. Construya un VI que genere un arreglo de 1000 números aleatorios entre 0 y 100 con 3 cifras decimales. Organice este arreglo en orden ascendente y descendente.
2. A partir de dos arreglos A y B de 1D y cinco elementos genere:
 - a. Un arreglo C de 2D donde A y B sean las filas de C.
 - b. Un arreglo D de 2D donde A y B sean las columnas de D.
 - c. Un arreglo E de 1D formado por [A B].
 - d. Un arreglo F de 1D formado por [B A].
3. Dado un arreglo de 2D de 10x8 elementos numéricos aleatorios, extraiga el subarreglo 2D de 3x3 que se forma a partir de la posición (3,3).
4. Genere un VI que permita al usuario introducir la información básica de un cliente (nombre, edad, sexo, teléfono, pazysalvo?) en un arreglo de clusters.
5. Identifique el cluster de error de la paleta de controles y explique su funcionamiento y para que sirven cada uno de sus componentes. Puede utilizar la opción <Control + H> para esta tarea.

4. GRAFICADORES.

4.1 OBJETIVO

Estudiar los métodos y procedimientos necesarios para generar gráficos utilizando LabVIEW.

4.2 DESCRIPCIÓN

Un concepto importante dentro de la instrumentación virtual es poder reproducir todos los datos capturados de manera gráfica tratando de ayudar a la conceptualización y el análisis por parte del operador, para esto LabVIEW cuenta con 9 diferentes tipos de graficadores que se encuentran en la paleta de **functions>>graph**. Ver figura 4.1.

En esta paleta se puede observar:

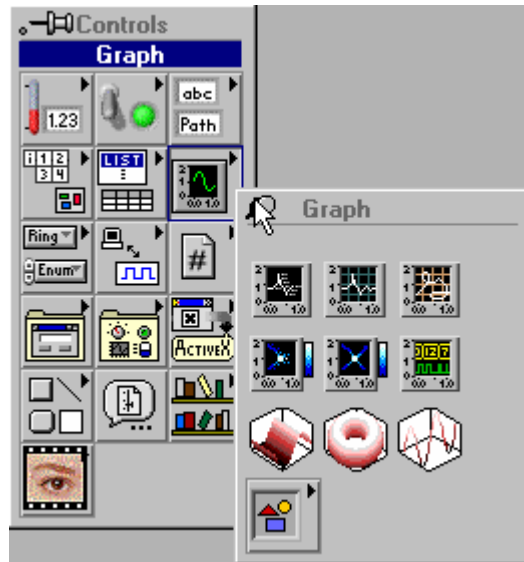


Figura 4.1. Paleta *Graph*.

1. Cuatro graficadores básicos para la elaboración de tareas en dos dimensiones,
 - *Waveform chart*
 - *Waveform graph*
 - *XY Graph*
 - *Digital waveform graph*

2. Dos herramientas para gráficos de arreglos de intensidad.
 - *Intensity chart*
 - *Intensity graph*

3. Tres herramientas para gráficas en tres dimensiones
 - *3D Surface graph*
 - *3D Parametric graph*

- *3D Curve graph*

Y una sub-paleta para el gráfico de polares y otros tipos de gráficos no convencionales.

En esencia los graficadores son un tipo especial de indicadores numéricos en dos o tres dimensiones que permiten visualizar la información de manera gráfica, pero como todos los indicadores de LabVIEW, en algún momento del diseño, éstos pueden ser convertidos a controles.

Cada graficador tiene asociado un terminal de tipo dinámico que sólo responderá a algunas configuraciones especiales de datos definidas por LabVIEW.

Por ejemplo, el terminal de un graficador tipo *waveform chart*, mostrado en la figura 4.2, puede tomar diferentes formas al ser alambrado con diferentes tipos de configuraciones de datos válidas para él. Ver figura 4.3.

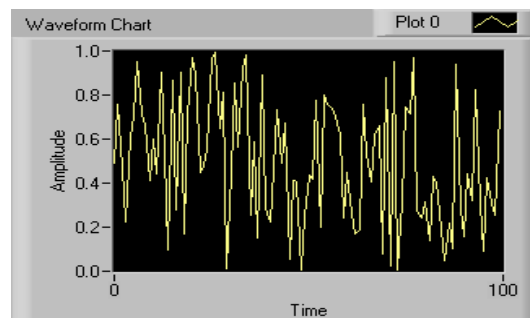


Figura 4.2. Graficador *Waveform chart*.

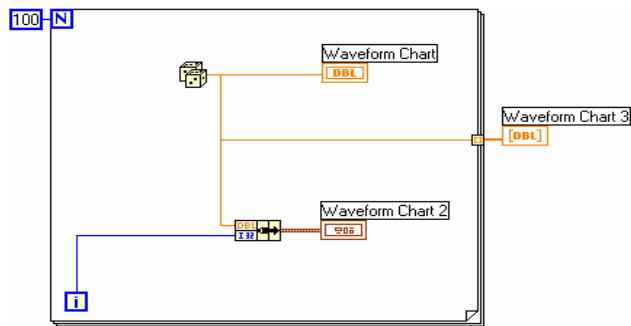


Figura 4.3. Diferentes aspectos del terminal del graficador *waveform chart*.

4.3 GRAFICADOR *WAVEFORM CHART*

Como se señaló, un graficador tipo *waveform chart*, mostrado en la figura 4.4, es un tipo de graficador que únicamente atiende a un grupo determinado de tipo de datos tales como:

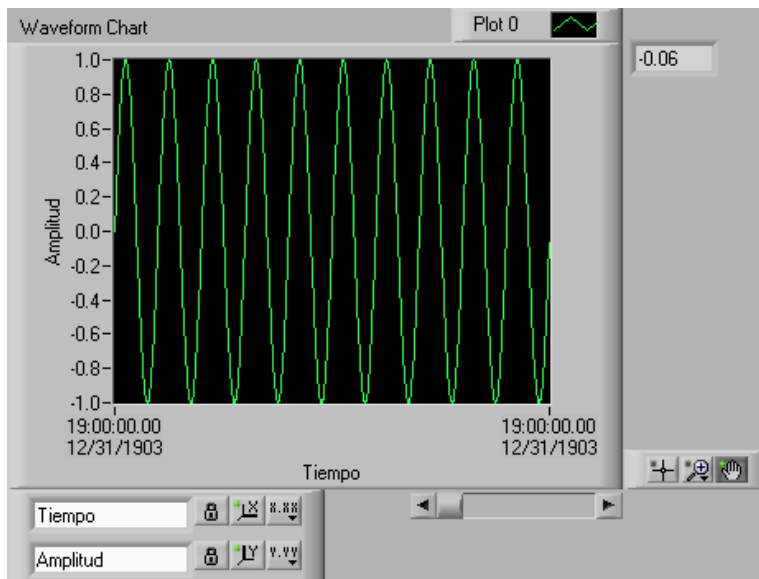
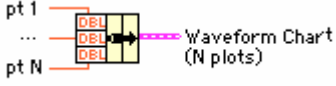


Figura 4.4. Graficador *waveform chart*.

Tabla 4.1. Tipos de Datos de *waveform chart*.

Tipo de dato	Resultado del dibujo
Escalar.	Una simple gráfica. Un punto.
Un vector (1D).	Una simple gráfica. Uno o más puntos.
<i>Waveform Data Type – WDT</i> Puede incluir información en el tiempo.	Una simple gráfica. Uno o más puntos.
Un arreglo (2D).	Múltiples gráficas. De uno o más puntos.
Una combinación de puntos a través de un nodo generador de registros (<i>cluster de escalares</i>). Figura 4.5.	 <p data-bbox="711 779 1300 846">Figura 4.5. Formato para generar un tipo de dato compatible con <i>waveform chart</i>.</p>

EJERCICIO 4.1 GRÁFICO DE ESCALARES *WAVEFORM CHART*

Generar una señal seno y graficarla punto a punto a través de un generador *waveform chart*.

Las figuras 4.6 y 4.7 muestran el panel frontal y la ventana de diagramación del ejercicio respectivamente.

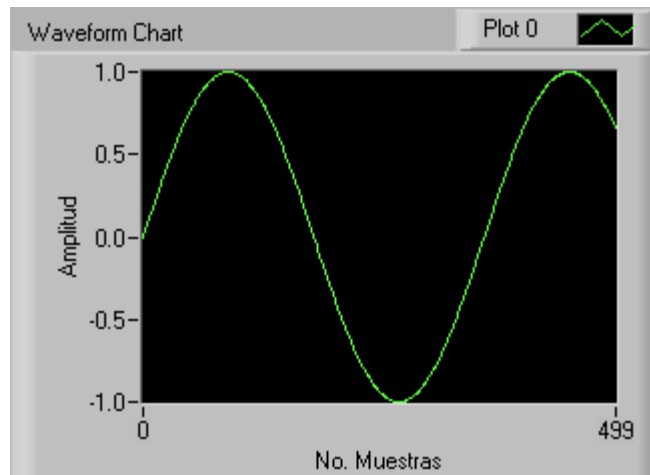


Figura 4.6. Panel frontal del ejercicio 4.1.

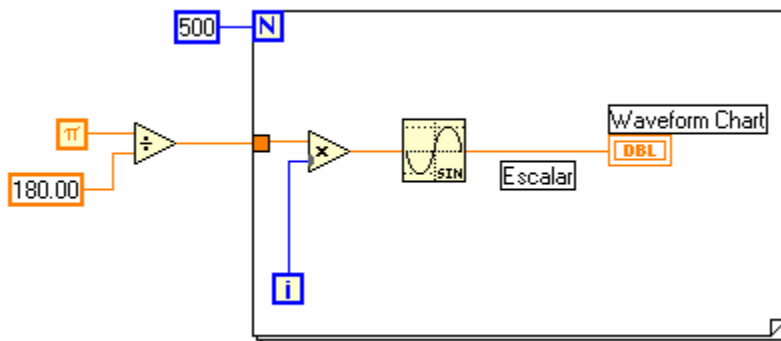


Figura 4.7. Diagrama del ejercicio 4.1.

Para observar mejor la generación de puntos, al diagrama de la figura 4.7 se le provocará un retraso de 70 ms a través de la función *Wait*, localizada en la paleta de funciones en el submenú *time & Dialog*.

Antes de correr nuevamente la aplicación se debe borrar el dibujo anterior. Esto se logra a través del menú del graficador, seleccionando **Data Operation>>Clear Chart**. Ver figura 4.8. Esta opción cambiará de nombre para los graficadores **Waveform Graph** y **XY Graph** por **Clear Graph**.

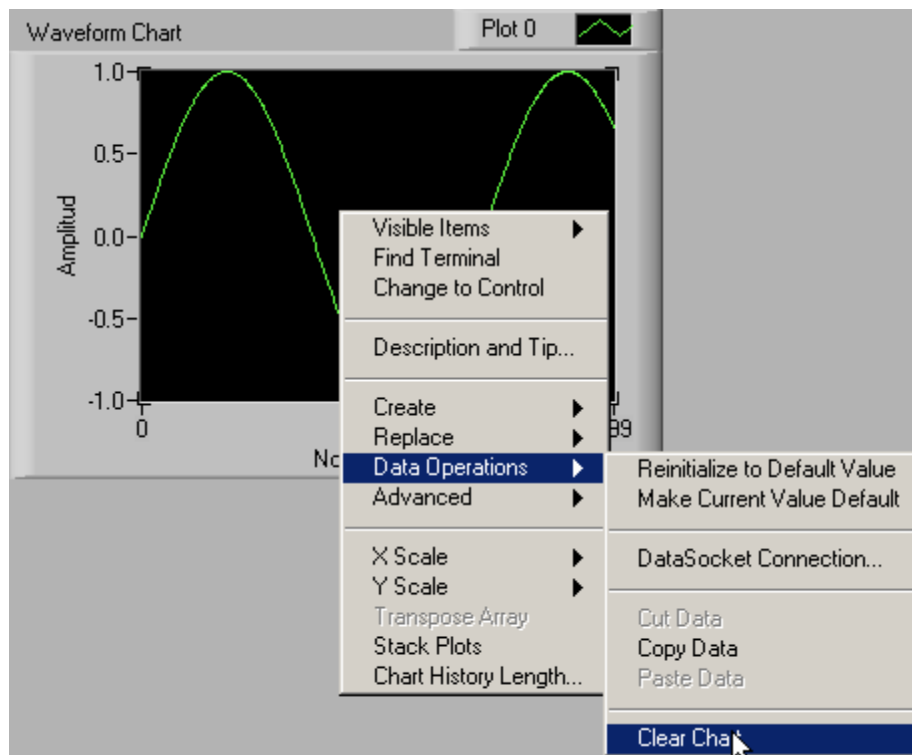


Figura 4.8. Modo de operar *Clear Chart*.

Este graficador cuenta con tres tipos de barrido de pantalla:

Strip Chart

Scope Chart

Sweep Chart

La figura 4.9 muestra como seleccionarlos.

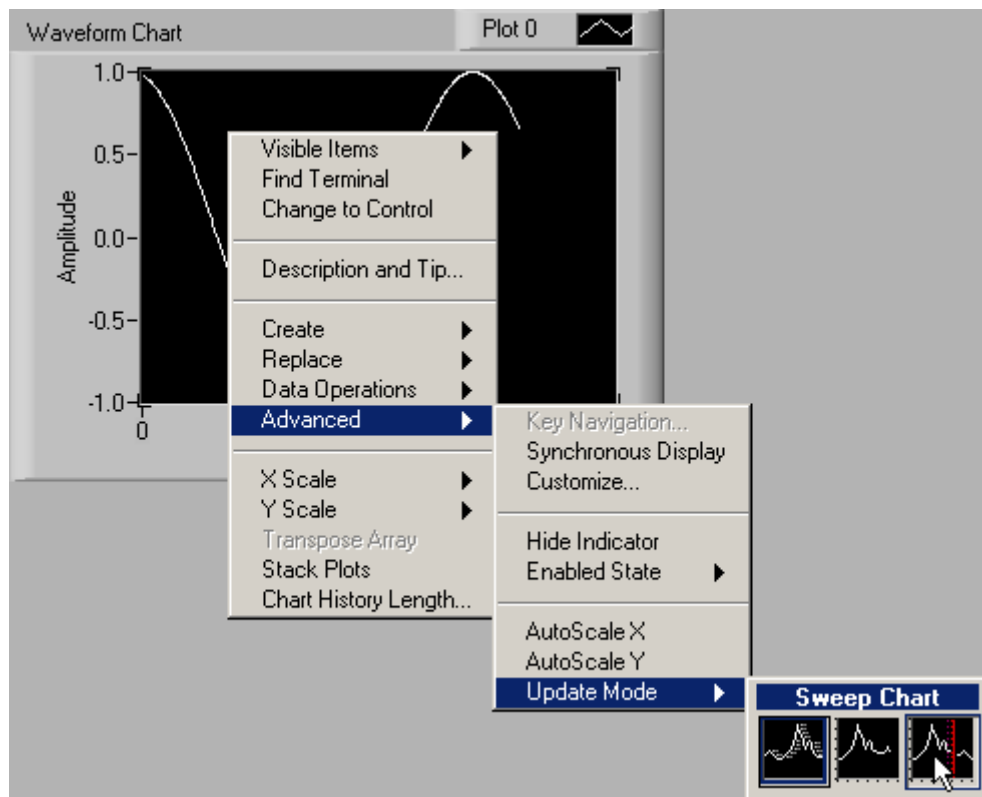


Figura 4.9. Modos de barrido de pantalla para el monitor *waveform chart*.

Al correr la aplicación se puede observar el comportamiento de cada modo.

FIN EJERCICIO 4.1

EJERCICIO 4.2 GRÁFICO DE VECTORES CON WAVEFORM CHART

a. Generar la misma señal del ejercicio 4.1, pero a partir de un vector.

La figura 4.10 muestra la ventana de diagramación.

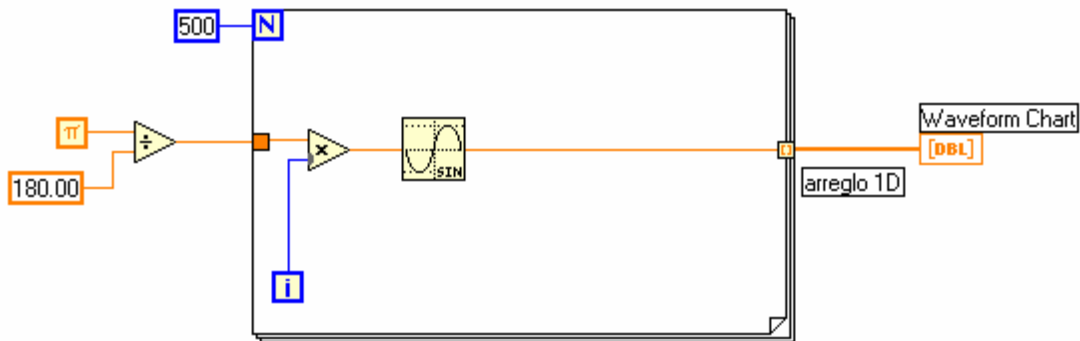


Figura 4.10. Ventana de diagramación para graficar un arreglo.

b. En la estructura *FOR* de la figura 4.10, cambiar el número de iteraciones a 524, y correr nuevamente la aplicación.

Obsérvese que gran parte de la información aparentemente se encuentra perdida. Para poder ver los datos no presentes existe dos métodos:

1. Cambiar la escala del eje horizontal del graficador.
2. Generar un deslizador que pueda ver la información a voluntad del operador. Ver figuras 4.11 y 4.12.

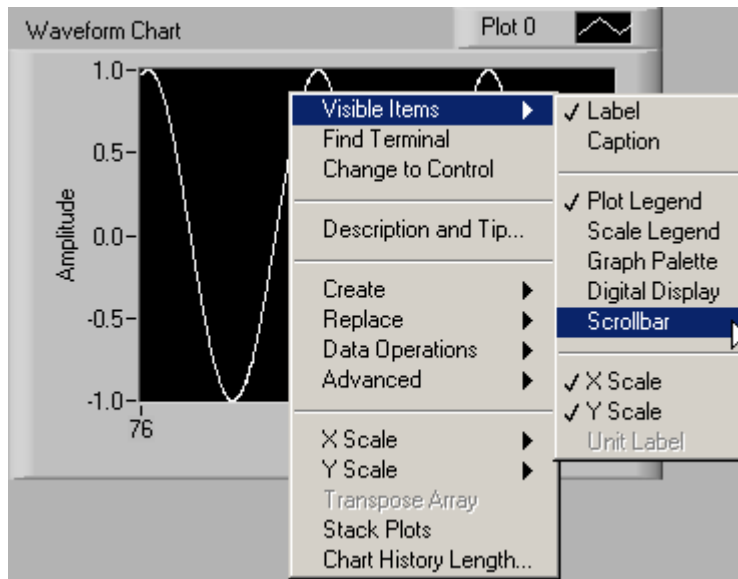


Figura 4.11. Creación de un *Scrollbar* para observar datos históricos.

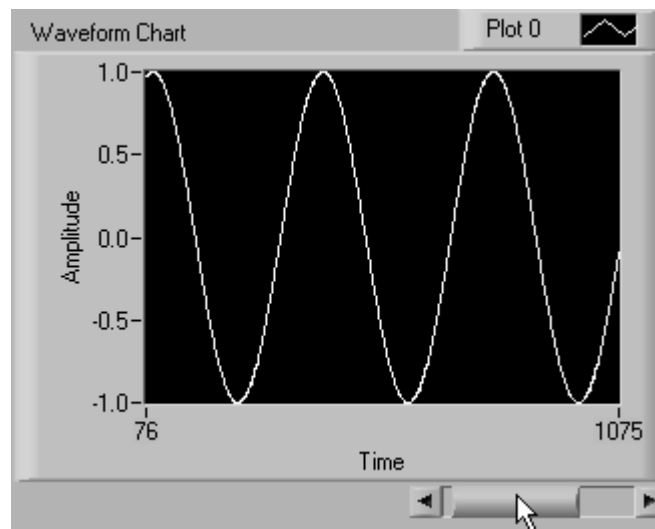


Figura 4.12. Graficador donde se enseña el deslizador.

De todas formas para que el graficador no pierda datos, catalogados como históricos por el operador, es necesario configurar el *buffer* de éste. La figura 4.13 enseña como.

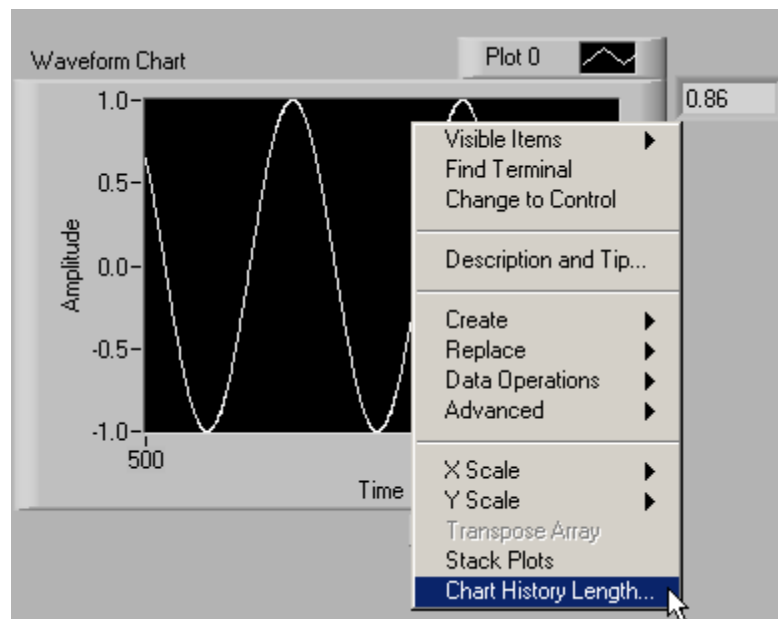


Figura 4.13. *Buffer* del graficador *waveform chart*.

El resultado de la acción anterior es la ventana de diálogo de la figura 4.14.

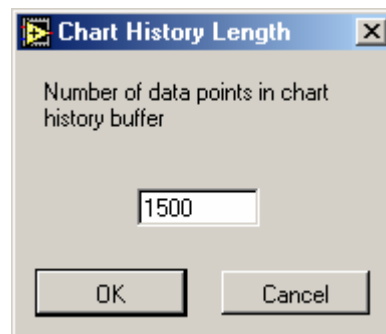


Figura 4.14. Cambio del *buffer* de un *waveform chart*.

FIN EJERCICIO 4.2

4.4 TIPO DE DATO *WDT*

Los datos *WDT* son un registro especial generado por LabVIEW, que facilita el manejo matemático, el análisis y la adquisición de señales. La figura 4.15, muestra el contenido de un registro *WDT*.

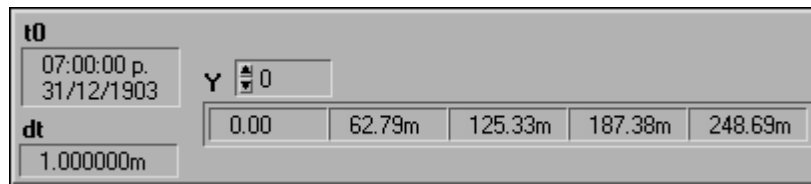


Figura 4.15. Registro tipo *WDT*.

Los datos contenidos en el registro *WDT* son:

t_0 : Valor inicial de tiempo de la señal.

dt : Intervalo de tiempo entre dos puntos de la señal.

[Y]: Vector de datos que contiene la señal.

Aunque aparentemente los datos *WDT* son un *cluster* común, su configuración es diferente, **y no pueden configurarse manualmente desde el panel frontal.**

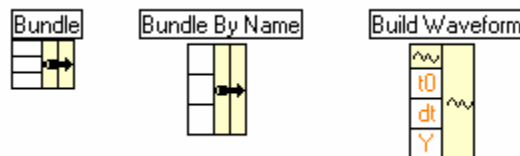


Figura 4.16. Funciones para *clusters* y para datos *WDT*.

Para la generación de un registro tipo *WDT* es necesario utilizar la función *Build Waveform* localizada en la paleta de funciones en el submenú *waveform*. Esta se enseña en la figura 4.17 y tiene las siguientes funciones.

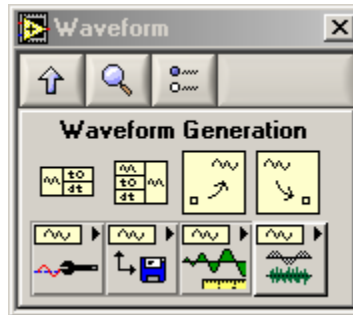


Figura 4.17. Paleta de funciones *waveform*.

Get Waveform Components

Descompone un registro *WDT* en sus componentes t_0 , dt , y $[Y]$.

Build Waveform

Herramienta que permite crear un dato *WDT* a partir de: un valor inicial de tiempo t_0 , un intervalo de tiempo entre puntos dt y los valores de la forma de onda contenidos en un arreglo $[Y]$.

Set waveform Attribute

Añade o cambia atributos a una *WDT*.

Get waveform Attribute

Captura los nombres y todos los atributos de un *WDT*. Los atributos pueden ser por ejemplo el nombre de los canales.

Waveform Operations

Conjunto de herramientas matemáticas aplicables a datos *WDT*.

Waveform File I/O

Conjunto de herramientas para el manejo de archivos aplicables a datos tipo *WDT*.

Waveform measurements

Conjunto de herramientas para el análisis de los datos tipo *WDT*.

Waveform Generations

Conjunto de herramientas que se utilizan para generar señales tipo *WDT*. Esta subpaleta se muestra en la figura 4.18.

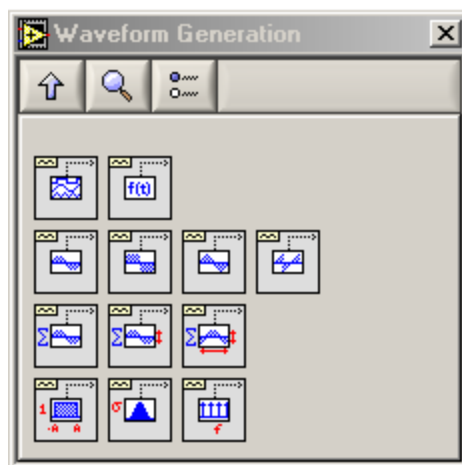


Figura 4.18. Subpaleta *Waveform Generation*.

Esta subpaleta permite generar diversos tipos de señales (multitono, funciones y de ruido), requeridas para el desarrollo de aplicaciones donde se necesita simular procedimientos, su salida es siempre del tipo *WDT*. Una de las más utilizadas es

“*Basic Function Generation*”. La figura 4.19 muestra los terminales de entrada y salida para esta función.

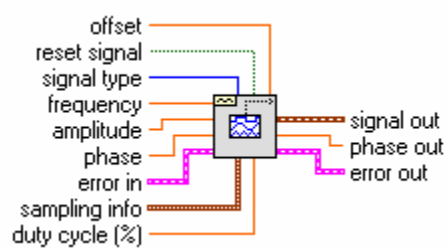


Figura 4.19. SubVI *Basic Function Generation*.

EJERCICIO 4.3 WAVEFORM DATA TYPE CON WAVEFORM CHART

Graficar en un *waveform chart*: $v(t) = 5 \cos(377t)$ utilizando un dato tipo *WDT*

Las figuras 4.20 y 4.21 enseñan el panel frontal y el diagrama requeridos.

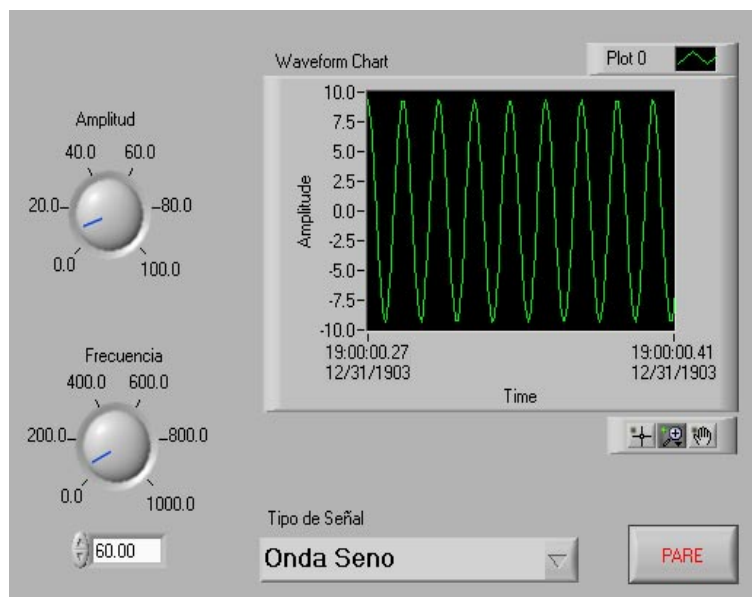


Figura 4.20. Panel frontal del ejercicio 4.3.

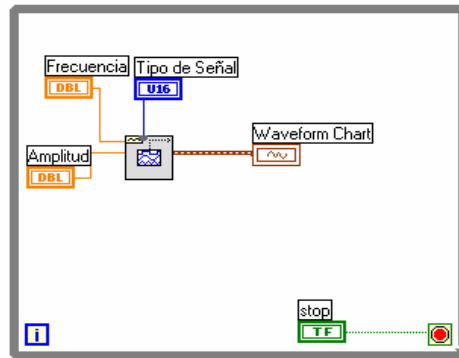


Figura 4.21. Diagrama del ejercicio 4.3.

FIN EJERCICIO 4.3

EJERCICIO 4.4 MÚLTIPLES GRÁFICAS EN UN WAVEFORM CHART

Graficar en un mismo *waveform chart* dos señales. Una seno y otra coseno.

Las figuras 4.22 y 4.23 muestran la solución al ejercicio.

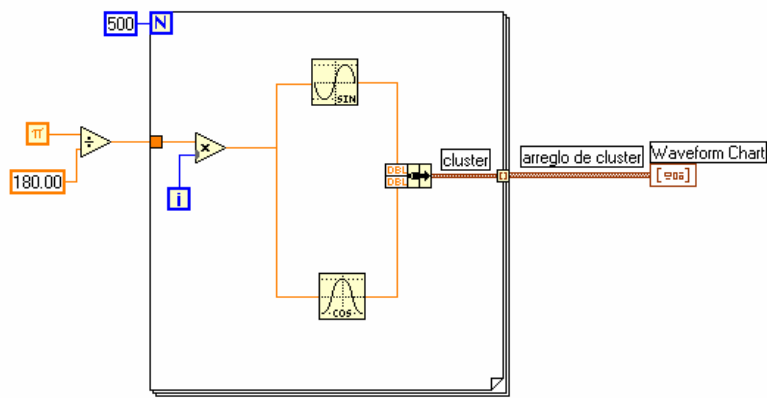


Figura 4.22. Diagrama del ejercicio 4.4.

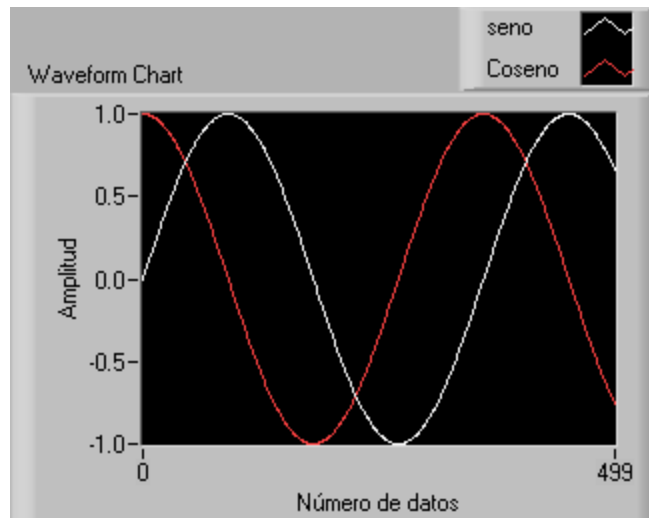


Figura 4.23. Panel frontal del ejercicio 4.4.

Cuando un *waveform chart* maneja varios gráficos de manera simultánea puede subdividirse por señales. Haciendo clic derecho sobre el graficador se selecciona la opción *stack plots*. Ver figura 4.24.

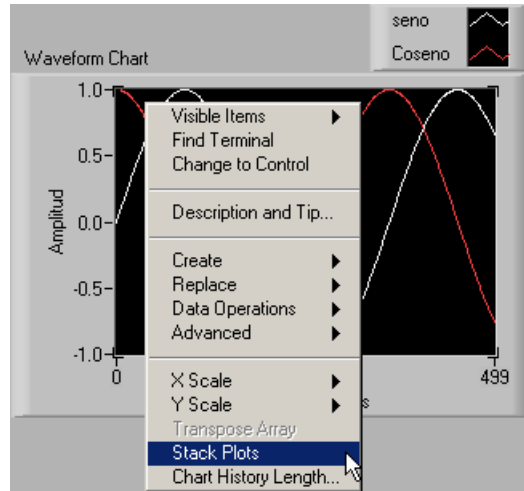


Figura 4.24. Selección de la propiedad *Stack Plots*.

Una vez seleccionada la opción *Stack Plots* el graficador tendrá la apariencia de la figura 4.25.

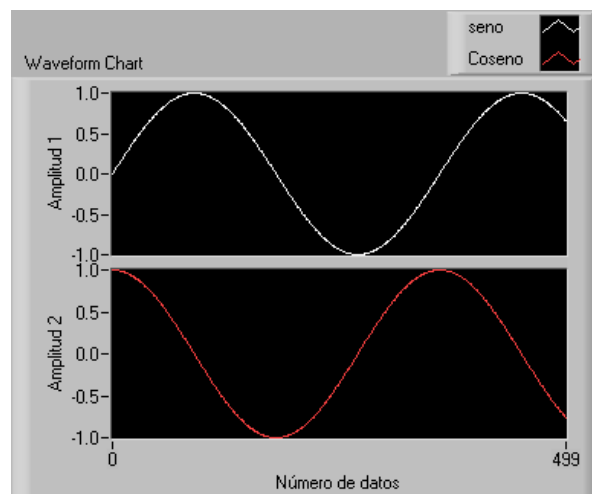


Figura 4.25. *Waveform chart* dividido de acuerdo con el número de señales.

FIN EJERCICIO 4.4

EJERCICIO 4.5 CLUSTER DE ESCALARES EN UNA WAVEFORM CHART

Generar las formas de onda del ejercicio anterior con un barrido de pantalla *Sweep Chart* y un control de retardo.

La solución al ejercicio 4.5 se muestra en las figuras 4.26 y 4.27.

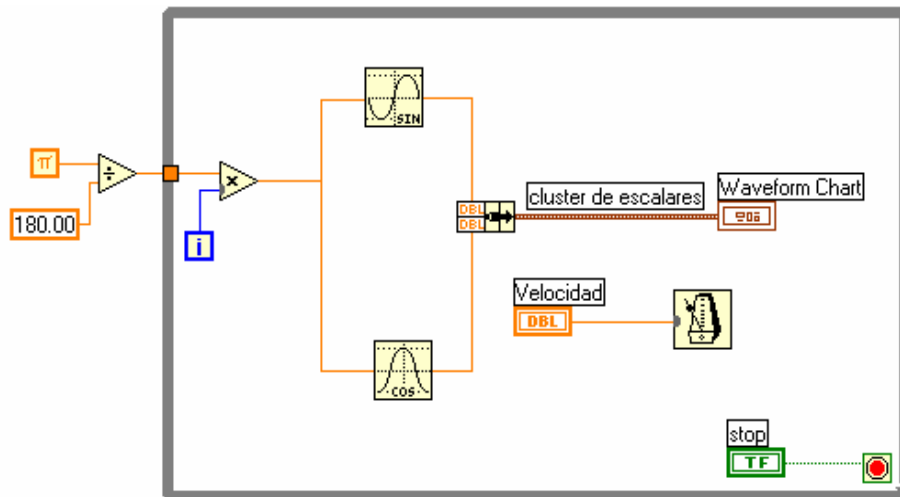


Figura 4.26. Diagrama de bloques del ejercicio 4.5.

Nótese que el formato, esta vez, para el graficador tipo *waveform chart* es un registro de escalares (*cluster* de escalares).

La figura 4.27 muestra el panel frontal de este ejercicio. Las opciones *stack Plots* y *sweep chart* están habilitadas.

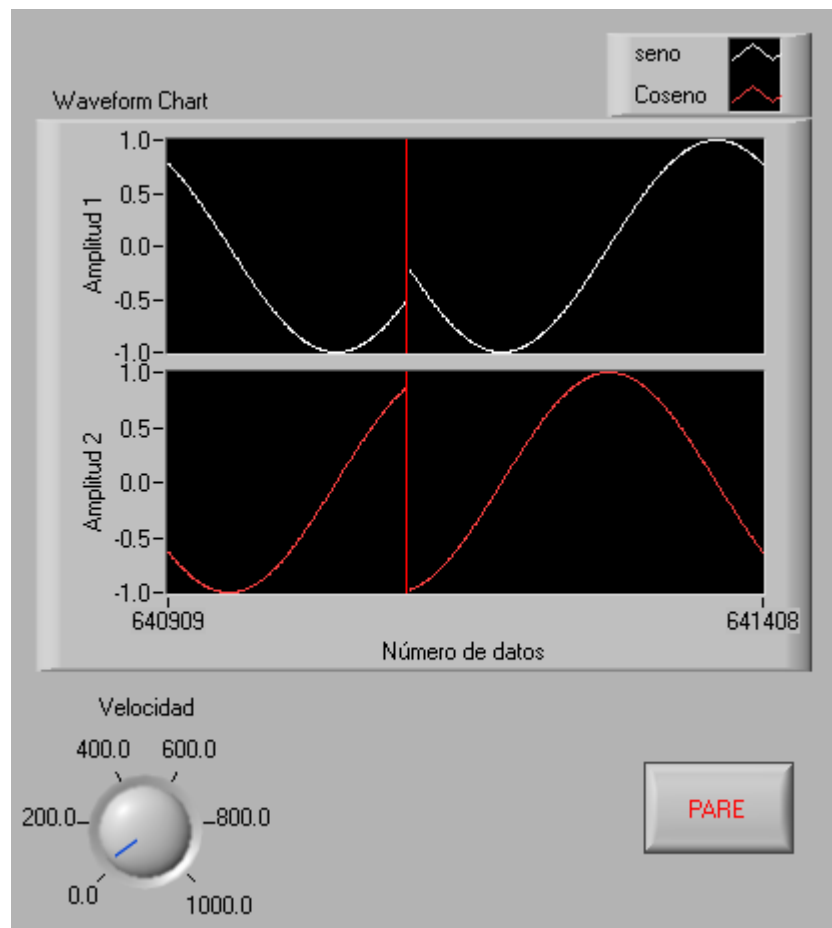


Figura 4.27. Panel frontal del ejercicio 4.5.

FIN EJERCICIO 4.5

4.5 GRAFICADOR WAVEFORM GRAPH

Este tipo de graficador está diseñado especialmente para graficar señales muestreadas, los datos estarán siempre referidos al eje X de manera continua.

La figura 4.28 muestra este graficador con sus respectivas herramientas.

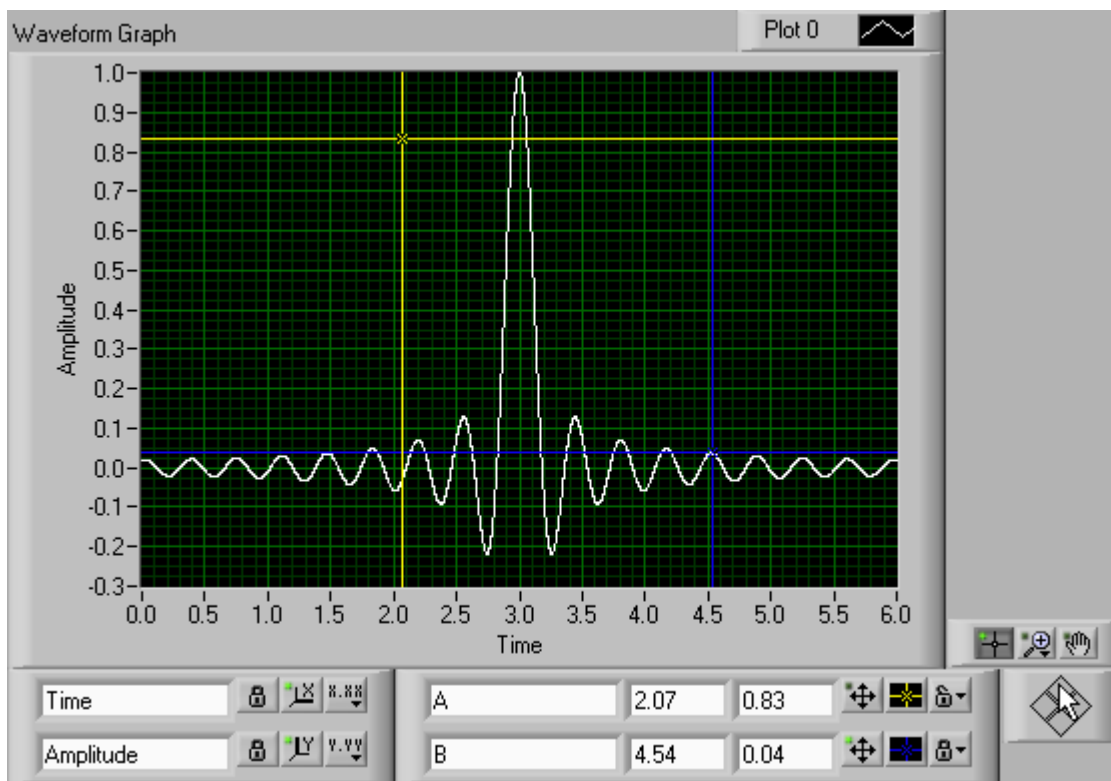
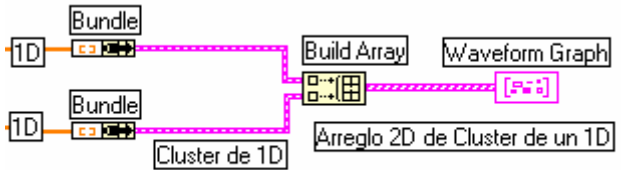
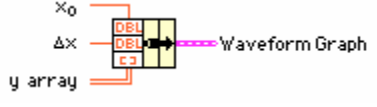
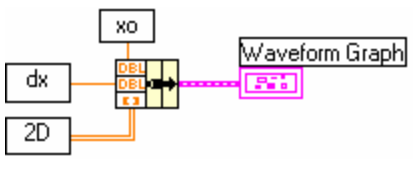
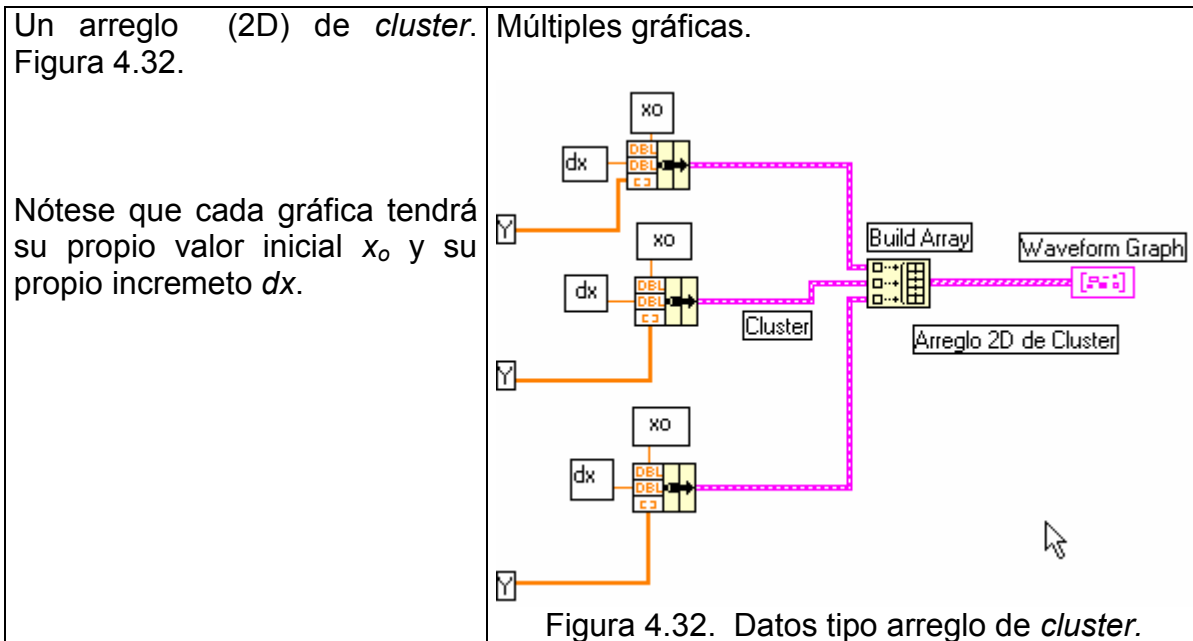


Figura 4.28. *Waveform graph* y sus respectivas herramientas.

Los tipos de datos aceptados para este tipo de graficador son:

Tabla 4.2. Tipos de dato para *Waveform Graph*.

Tipo de dato	Resultado
Un vector (1D).	Una simple gráfica.
<i>WDT</i>	Una simple gráfica.
Un arreglo (2D)	Múltiples gráficas.
<p>Un arreglo (2D) de <i>cluster</i>, figura 4.29.</p> <p>El <i>cluster</i> contendrá como único elemento un vector. La señal.</p>	<p>Múltiples gráficas.</p>  <p>Figura 4.29. Arreglo de <i>cluster</i>.</p>
<p>Un <i>cluster</i> con la siguiente información. Figura 4.30.</p> <p>x_0: Valor inicial del eje X. dx: Intervalo del eje X entre dos puntos de la señal $[Y]$: Vector de datos.</p>	<p>Una simple gráfica con formato.</p>  <p>Figura 4.30. <i>Cluster</i> aceptado por <i>waveform graph</i>.</p>
<p>Un <i>cluster</i> con la siguiente información. Figura 4.31.</p> <p>x_0: Valor inicial del eje X. dx: Intervalo del eje X entre dos puntos de la señal $[Y]$: Vector 2D con los datos de las señales a graficar.</p> <p>Todas las curvas graficadas contarán con el mismo valor x_0 y dx.</p>	<p>Múltiples gráficas.</p>  <p>Figura 4.31. Múltiples gráficas con el mismo formato.</p>



EJERCICIO 4.6 WAVEFORM GRAPH UTILIZANDO DATOS WDT

Calcular la potencia $p(t)$ y la potencia activa de una carga cuyo voltaje y corriente son:

$$v(t) = 138 \text{ sen } (377t)$$

$$i(t) = 13 \text{ sen } (377t - 30^\circ)$$

El programa debe tener la capacidad de cambiar los valores de $v(t)$ e $i(t)$ a voluntad del operador.

El panel frontal del ejercicio 4.6 se observa en la figura 4.33.

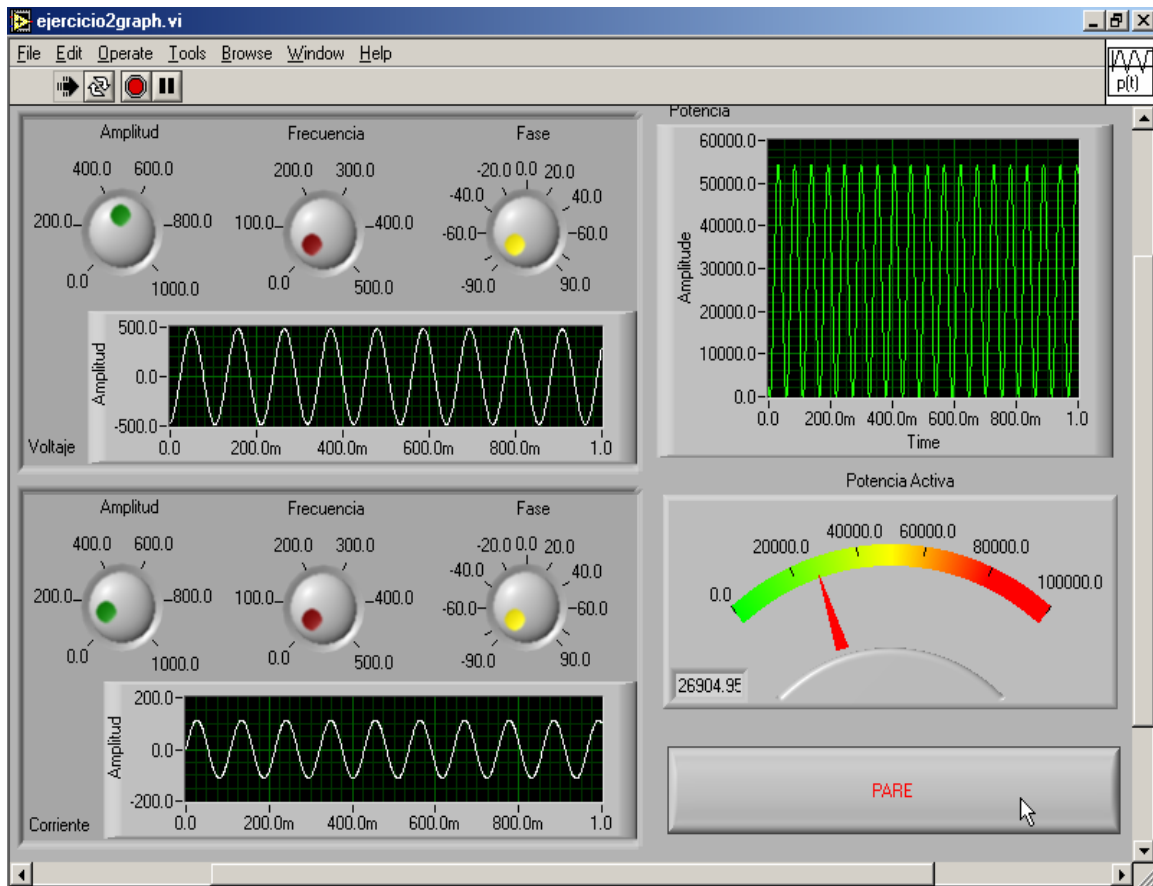


Figura 4.33. Panel frontal del ejercicio 4.6.

El valor de la potencia es $p(t) = v(t) \cdot i(t)$ y la potencia activa es el promedio de $p(t)$.

El diagrama se muestra en la figura 4.34.

Las funciones utilizadas para la generación de señales y la aritmética fueron tomadas del submenú *waveform* de la paleta de funciones.

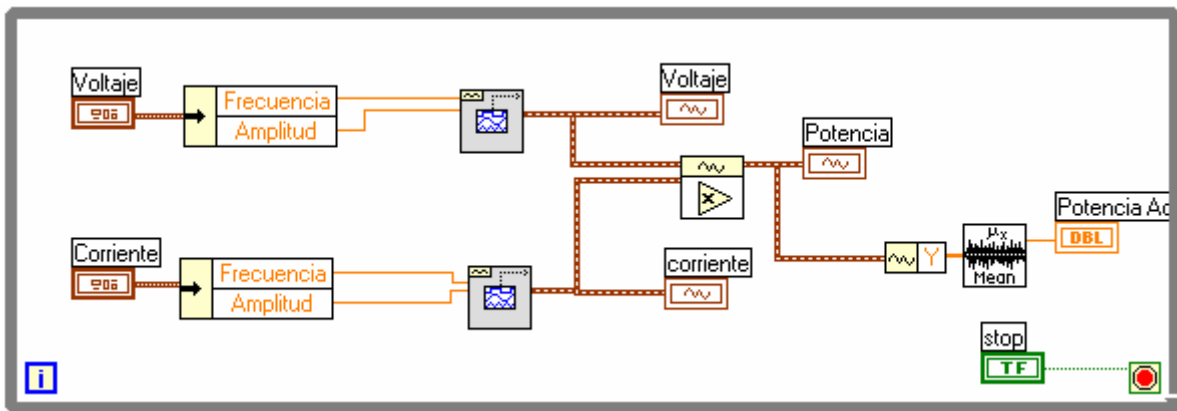


Figura 4.34. Ventana de diagramación del ejercicio 4.6.

FIN EJERCICIO 4.6

EJERCICIO 4.7 WAVEFORM GRAPH A PARTIR DE UN CLUSTER

A partir de una estructura *FOR* graficar la función de la figura 4.28.

El diagrama para el ejercicio 4.7 se observa en la figura 4.35.

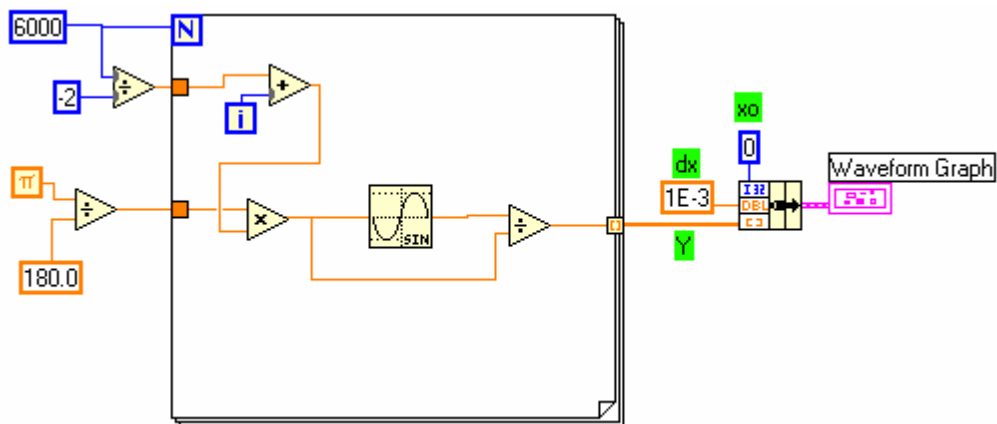


Figura 4.35. *Waveform graph* a partir de un *cluster*.

FIN EJERCICIO 4.7

EJERCICIO 4.8 MÚLTIPLES GRÁFICOS

Generar tres señales con las siguientes características.

Tipo de señal	Amplitud	X_0	dx
Seno	1	0 ms	1 ms
Cuadrada	2	10 ms	0.5 ms
Triangular	1.5	15 ms	2 ms

La respuesta del ejercicio se observa en las figuras 4.36 y 4.37.

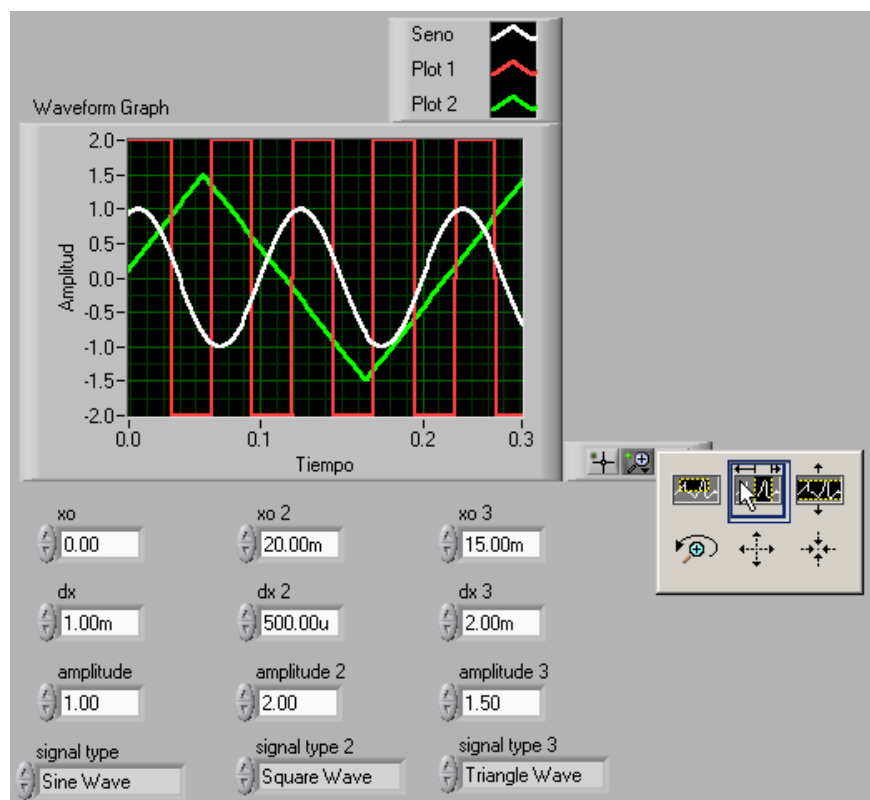


Figura 4.36. Panel frontal del ejercicio 4.8.

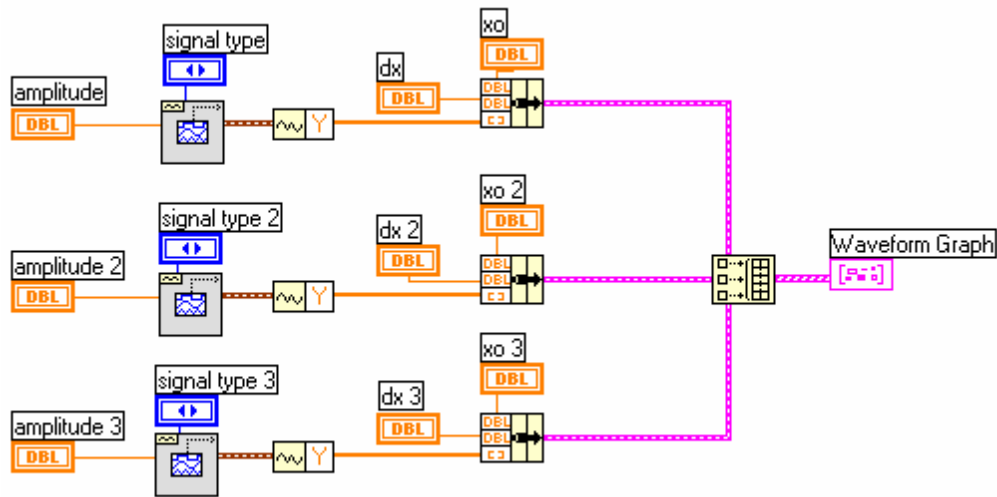


Figura 4.37. Panel frontal del ejercicio 4.8.

Una mejor manera de generar el diagrama es utilizando las herramientas *WDT* como se muestra en la figura 4.38.

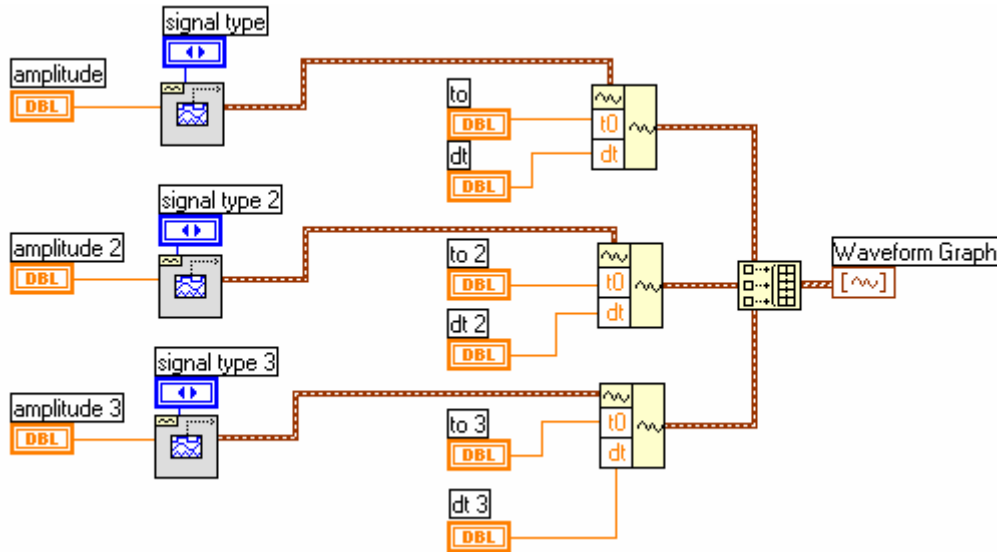


Figura 4.38. Diagrama utilizando las herramientas de *WDT*.

FIN EJERCICIO 4.8

4.6 GRAFICADOR XY GRAPH

El graficador *XY Graph*, es un graficador cartesiano de propósito general. La figura 4.39 muestra el graficador y sus respectivas herramientas.

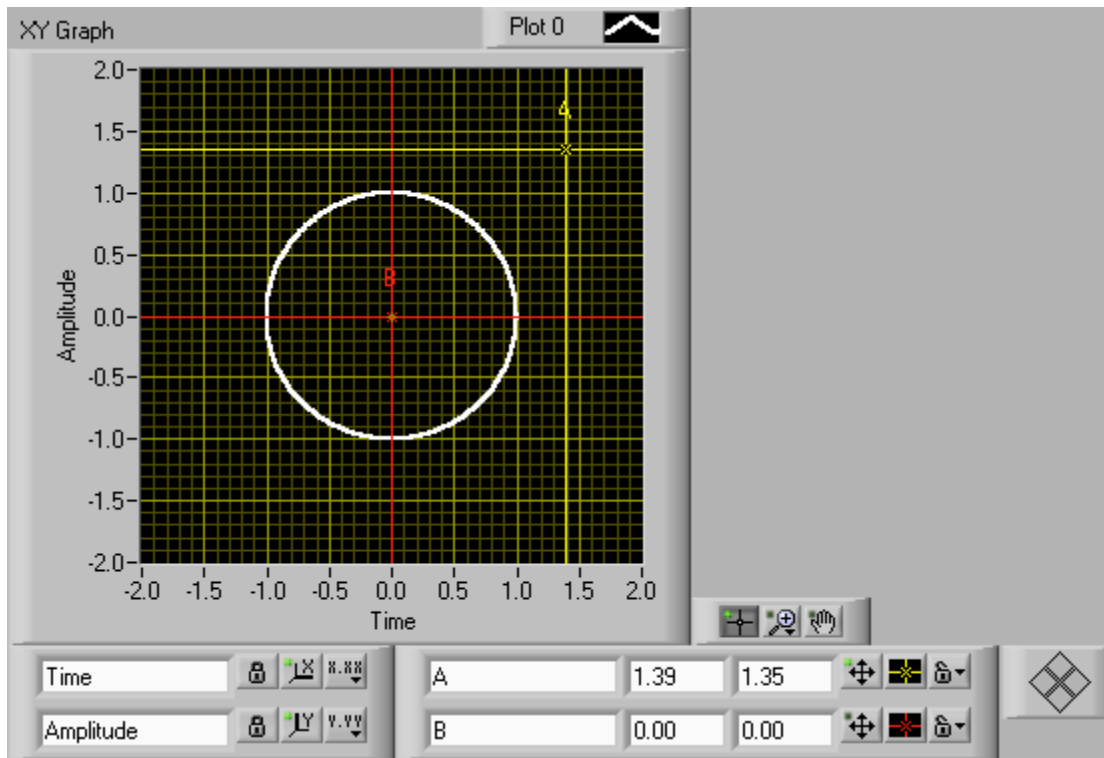
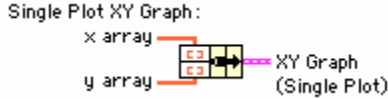
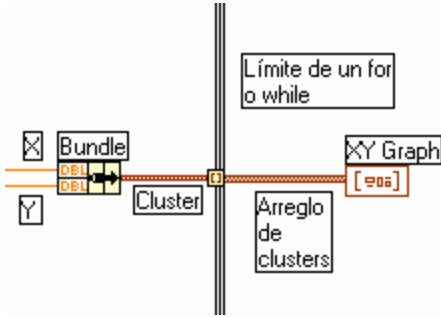
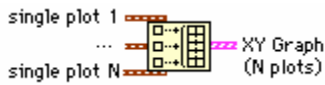


Figura 4.39. *Graph* y sus correspondientes herramientas.

Los tipos de datos aceptados por este graficador son:

Tabla 4.3. Tipos de datos para *XY Graph*.

Tipo de dato	Resultado del dibujo
Un registro conformado por dos vectores, el vector uno con los datos de X y el vector dos con los datos de Y. Ver figura 4.40.	Una sola gráfica.

	 <p>Single Plot XY Graph: x array y array XY Graph (Single Plot)</p> <p>Figura 4.40. Tipo de dato válido para un graficador <i>XY Graph</i>.</p>
<p>Tipo de dato</p>	<p>Resultado del dibujo</p>
<p>Un arreglo de registros. Cada registro esta conformado por un valor X Escalar y un valor Y Escalar, ver figura 4.41.</p>	<p>Una sola gráfica</p>  <p>Límite de un for o while Arreglo de clusters XY Graph</p> <p>Figura 4.41. Dato válido para un <i>XY Graph</i>.</p>
<p>Un arreglo de clusters. Cada cluster está conformado por los siguientes elementos. Ver figura 4.42.</p> <p>Arreglo de datos [X] Arreglo de datos [Y]</p>	<p>Múltiples gráficos en un <i>XY Graph</i></p>  <p>single plot 1 ... single plot N XY Graph (N plots)</p> <p>Figura 4.42. Tipo de dato válido para un graficador <i>XY Graph</i>.</p>

EJERCICIO 4.9 XY GRAPH

Generar dos señales A y B y obtener a partir de ellas la curva de Lissajouss y los valores rms de cada una.

La figura 4.43 muestra el panel frontal utilizado para dar solución a este ejercicio.

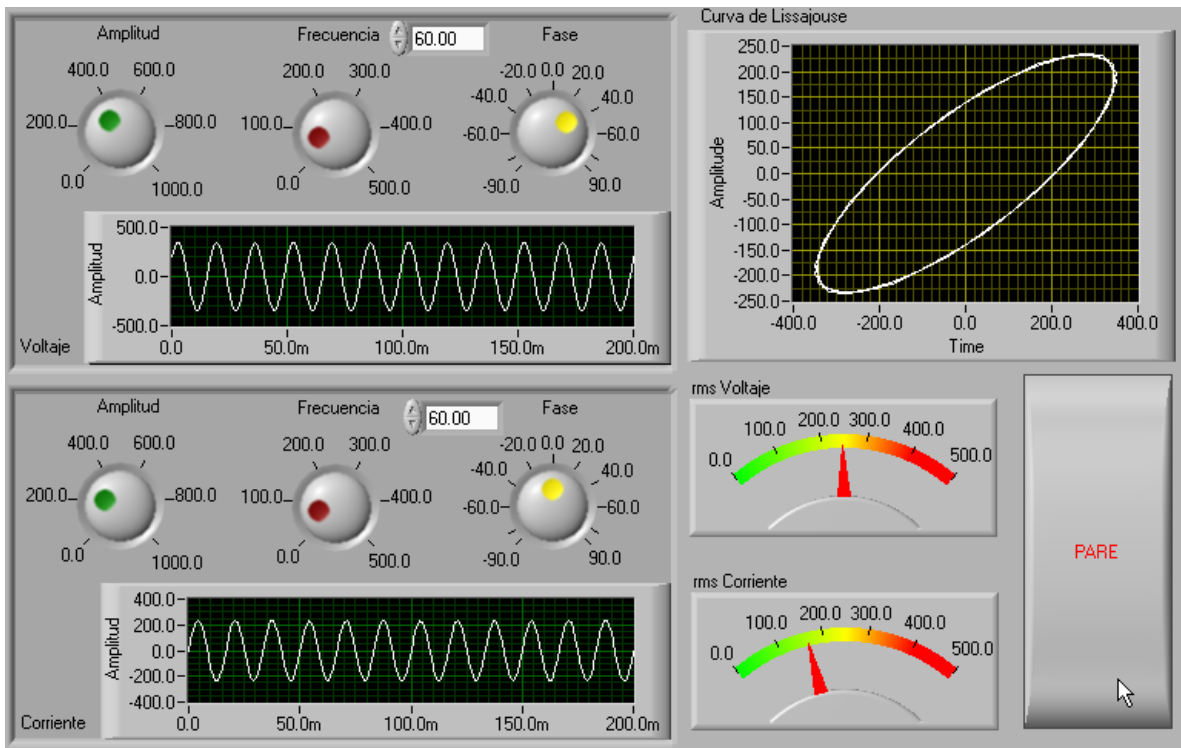


Figura 4.43. Panel frontal del ejercicio 4.9.

La figura 4.44 muestra el diagrama.

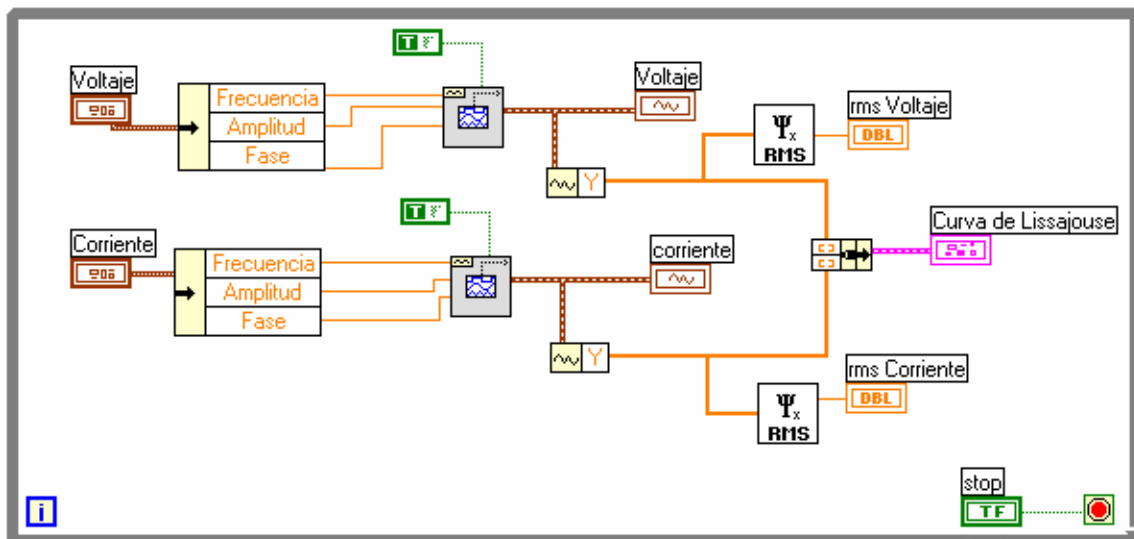


Figura 4.44. Diagrama del ejercicio 4.9.

Si se desea medir el desfase a través de la figura de Lissajous, en la figura 4.45 se muestra la manera de cargar la ventana *cursor legend*, esto permitirá generar dos cursores y tomar medidas en las curvas correspondientes.

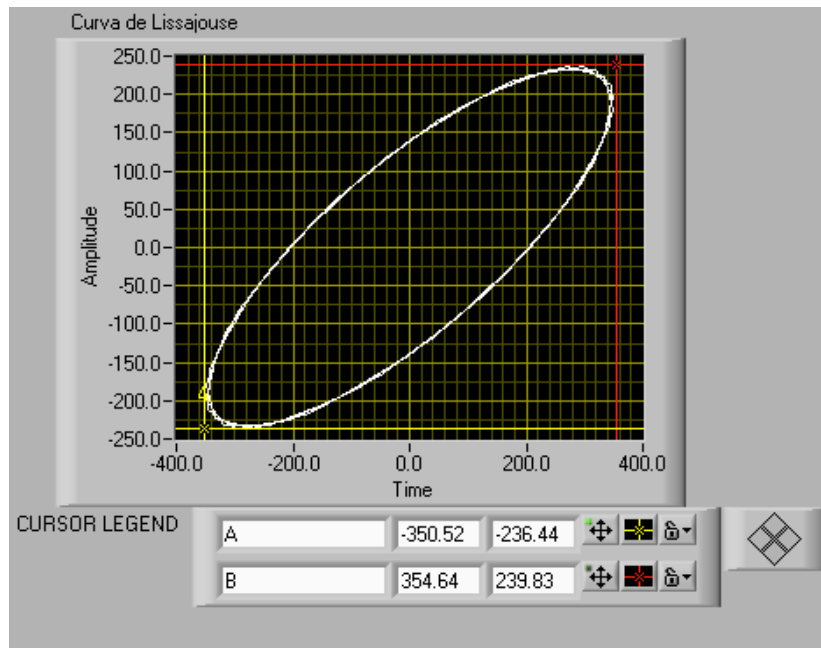


Figura 4.45. *XY Graph* con los cursores activos.

FIN EJERCICIO 4.9

EJERCICIO 4.10 MÚLTIPLES GRÁFICOS EN UN XY GRAPH

Generar dos curvas a partir de tres funciones $f(t)$ y construir arreglos de *clusters*. Cada *cluster* contendrá los vectores $[x]$ y $[Y]$. La figura 4.46 muestra el panel frontal del VI.

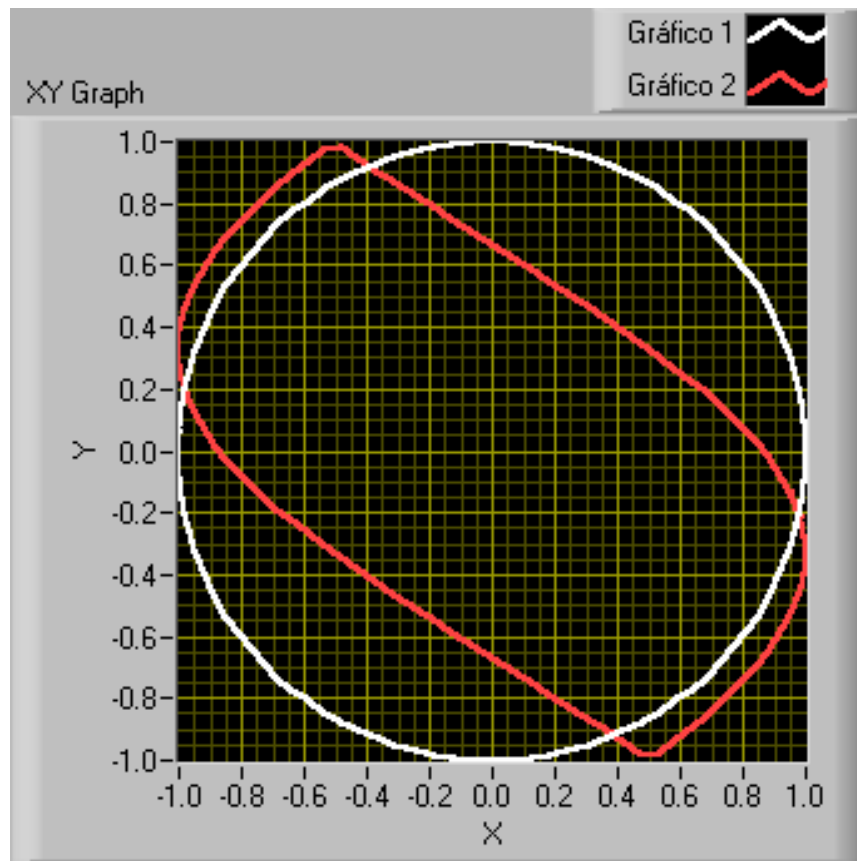


Figura 4.46. Panel frontal del ejercicio 4.10.

El diagrama correspondiente se muestra en la figura 4.47.

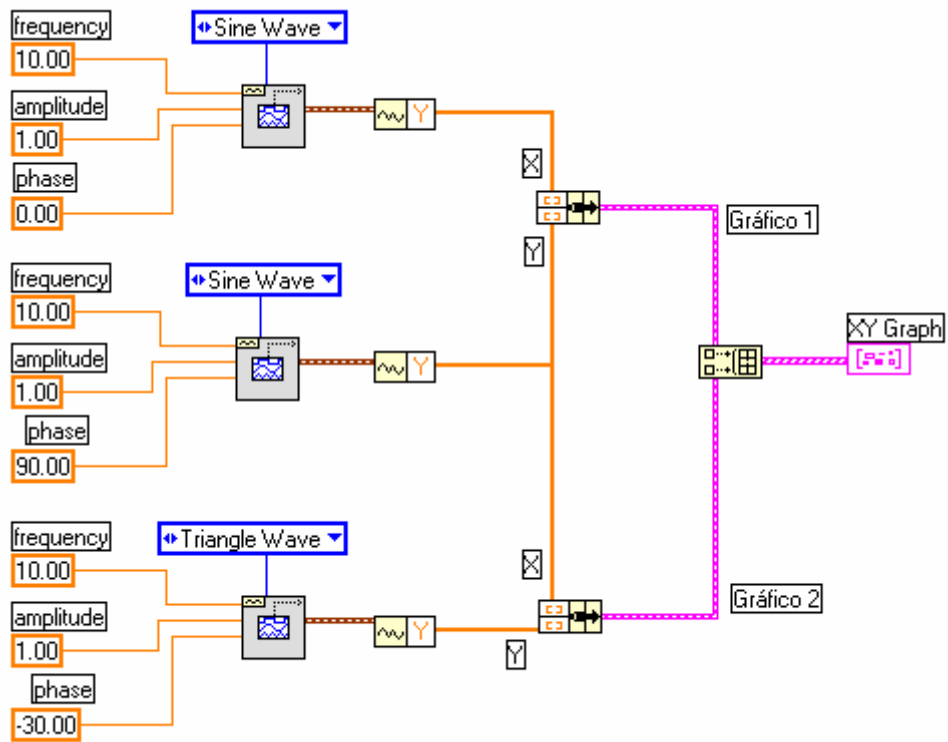


Figura 4.47. Panel frontal del ejercicio 4.10.

FIN EJERCICIO 4.10

4.7 EJERCICIOS PROPUESTOS

1. Utilizando las funciones “*digital thermometer.vi*” y “*Demo Voltaje Read.vi*”, localizadas en el submenú **tutorial** de la paleta de funciones, generar dos señales y graficarlas de todas las formas posibles utilizando varios *waveform chart*.
2. Crear un panel frontal con dos graficadores *waveform graph*. En el primero graficar cinco señales independientes en amplitud, frecuencia y fase. En el segundo graficar la suma aritmética de éstas.
3. La ventana de diagramación de la figura 4.48 tiene un *XY Graph*. Determinar cuál es el resultado de esta rutina.

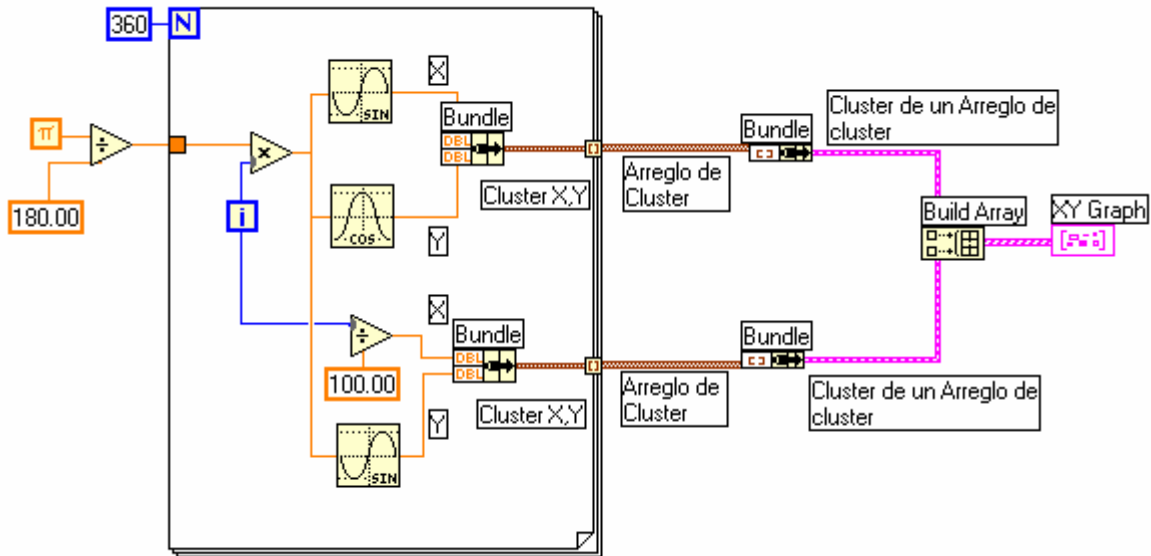


Figura 4.48. Diagrama para analizar.

4. Generar un cubo a partir de ocho puntos dados por el operador.

5. SUBVIS, VARIABLES LOCALES Y GLOBALES.

5.1 OBJETIVOS

Estudiar el procedimiento para generar subrutinas conocidas en LabVIEW como subVIs. Analizar otras herramientas importantes, como las variables de tipo local y global y los nodos de atributo, que permiten controlar dinámicamente el diseño y comportamiento de los programas realizados. Estudiar como alterar las propiedades de un VI.

5.2 SUBVIS

Un subVI es un VI que esta siendo utilizado dentro de otro VI y por tanto tiene asociado un icono y conectores de entrada y salida de datos.

Los subVIs permiten modular una aplicación en tareas más simples permitiendo que aplicaciones extensas puedan ser divididas en varias tareas pequeñas, las que a su vez pueden ser divididas en otras tareas más pequeñas y así sucesivamente.

Los subVIs son equivalentes a las subrutinas en lenguajes de programación convencionales basados en texto.

La figura 5.1 muestra la implementación en G de la función “operar” y su código equivalente.

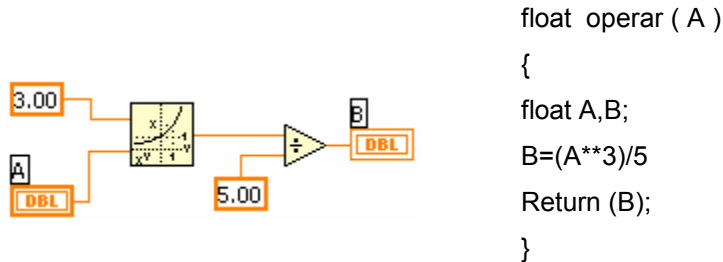


Figura 5.1. Función “operar”.

La figura 5.2 muestra como puede ser utilizada la función “operar” dentro de otra función.

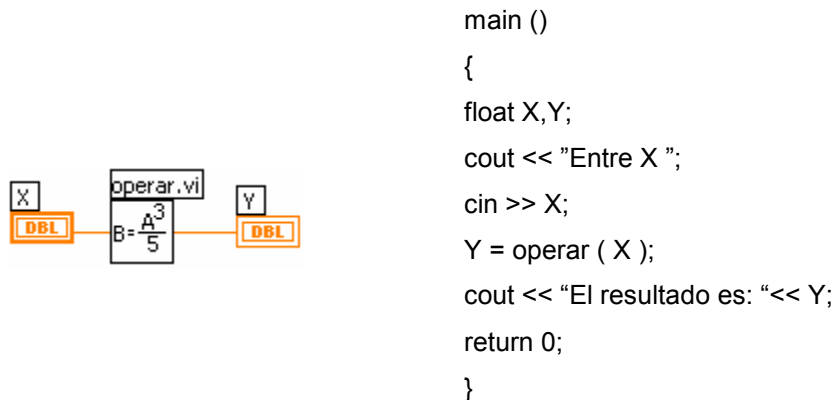


Figura 5.2. Utilización de “operar” dentro de otra función.

Puede observarse que el VI operar.vi ahora está siendo utilizado dentro de otro VI, por lo tanto es un subVI.

5.3 EDICIÓN DEL ICONO

El icono que identifica un VI, está ubicado en la parte superior derecha del panel frontal.

Para editarlo se debe hacer clic derecho en él y seleccionar *Edit Icon* como se observa en la figura 5.3.

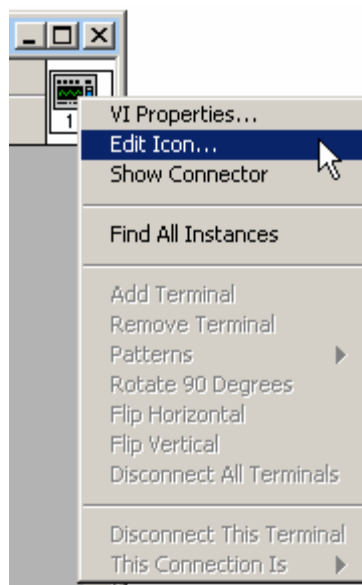


Figura 5.3. Menú para editar el Icono.

Esta acción mostrará la ventana de diálogo de la figura 5.4 donde se puede editar el icono.

Esta ventana funciona de forma similar a un programa de dibujo y cuenta al lado izquierdo con las siguientes herramientas.

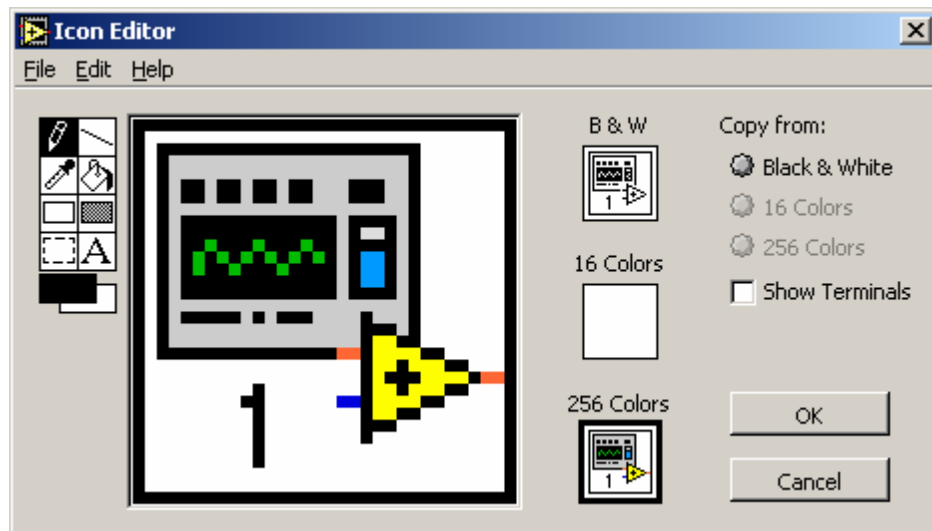


Figura 5.4. Editor de Icono.



Pencil: Dibuja o borra píxel por píxel.



Line: Dibuja líneas continuas, con la opción de <shift> restringe el dibujo a líneas horizontales, verticales o diagonales.



Dropper: Copia el color del píxel seleccionado.



Fill bucket: Llena áreas delimitadas del color seleccionado en *foreground*.



Rectangle: Dibuja bordes rectangulares del color del *foreground*. Haciendo doble clic en este icono se generará una ventana del color seleccionado.



Filled Rectangle: Dibuja áreas rectangulares con el color del *foreground*. Si se hace doble clic llenará la ventana con el color de *background*.



Select: Selecciona áreas del icono para moverlo, copiarlo o borrarlo. Si se hace doble clic en esta área se podrá seleccionar todo el icono para borrar su contenido.



Text: Edita texto dentro del icono. Si se hace doble clic en este icono se podrá seleccionar diferentes fuentes.



Foreground & Background: Muestra los colores de trabajo y de fondo. Haciendo un clic en cada paleta se puede seleccionar los colores respectivos.

Las opciones a la derecha de la ventana son:

Copy From: Herramienta que permite copiar un icono realizado en otro formato de color (B&N, 16 colores o 24 colores) al formato seleccionado.

Show Terminal: Si se hace clic en esta opción se podrá observar los terminales del conector.

OK: Para guardar el dibujo y retornar a la ventana del panel frontal.

CANCEL: Para retornar a la ventana del panel frontal sin guardar los cambios.

La barra de menú cuenta además con opciones típicas de cualquier editor gráfico: *Undo*, *Redo*, *Cut*, *Copy*, *Paste* y *Delete*. Para permitir mayor compatibilidad con los formatos de colores se recomienda siempre hacer un icono para los tres formatos disponibles. La figura 5.5 muestra el icono editado para operar.vi.

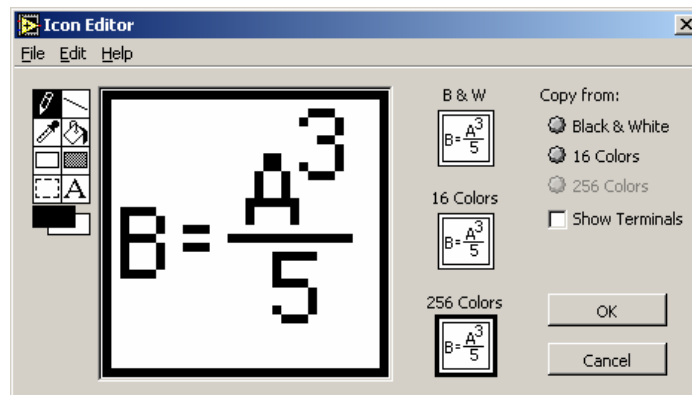


Figura 5.5. Icono de operar.vi.

5.4 LOS CONECTORES

Los conectores de un VI permiten enviar y recibir datos desde un VI de mayor jerarquía cuando se este utilizando como subVI. Para editar los conectores se selecciona la opción *Show Connectors* del menú del icono del VI.

El icono elaborado será entonces reemplazado por los terminales que LabVIEW espera que sean cableados. La figura 5.6 muestra los conectores esperados para *operar.vi*.

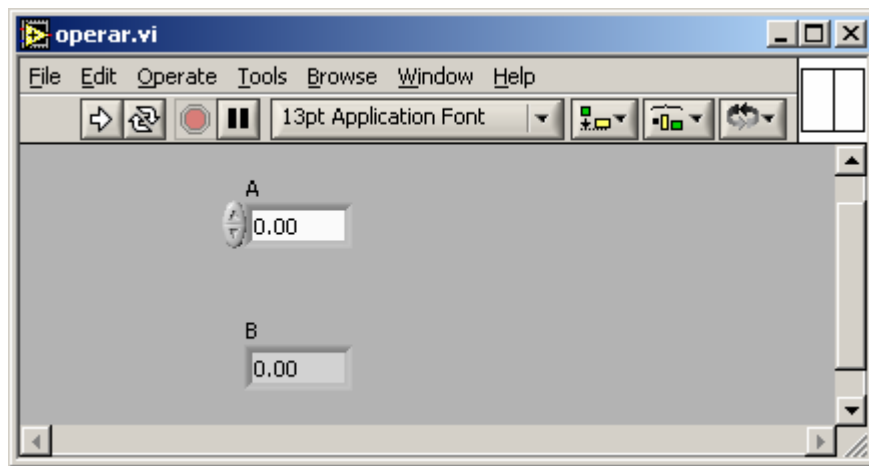


Figura 5.6. Conector de un VI.

LabVIEW de manera automática selecciona la herramienta de cableado con el fin de que el programador asocie que controles o indicadores corresponden a las áreas que definen los conectores.

Una vez asociado un conector con algún control o indicador, este tomará el color del tipo de representación de la variable seleccionada.

En caso de ser necesario modificar el número de terminales, mover o borrar la configuración, LabVIEW cuenta con una serie de opciones que permiten hacerlo.

La figura 5.7 muestra el menú de los conectores.

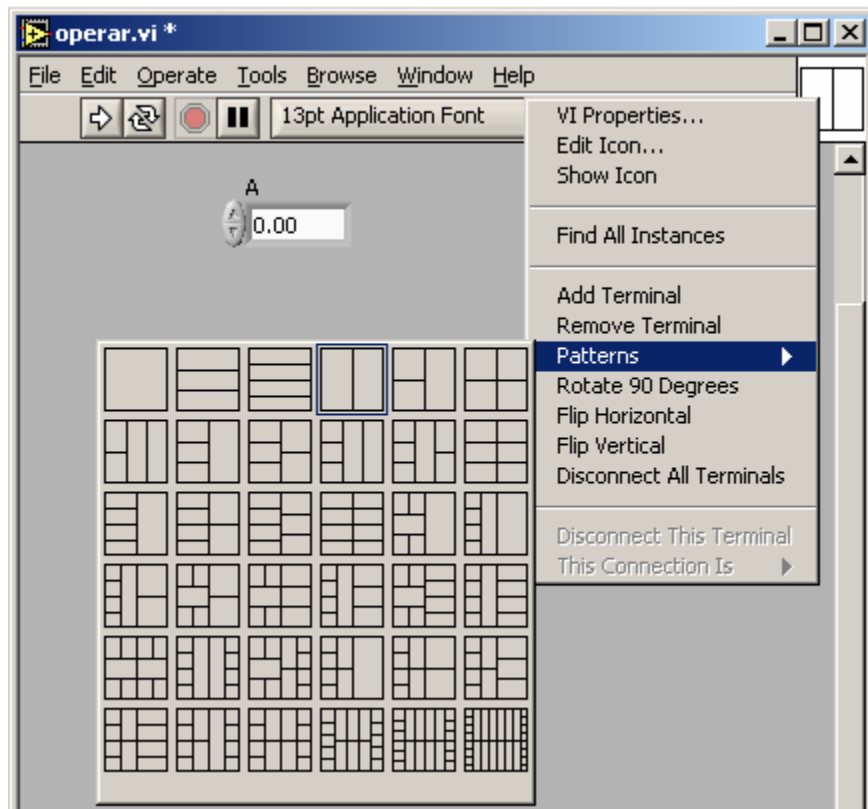


Figura 5.7. Menú del Icono cuando se muestran los conectores.

Para asignar los terminales a los controles e indicadores se procede de la siguiente manera:

1. Se selecciona el área correspondiente a un conector con la herramienta de cableado.

Este tomará el color negro para mostrar que se encuentra seleccionado. Figura 5.8.

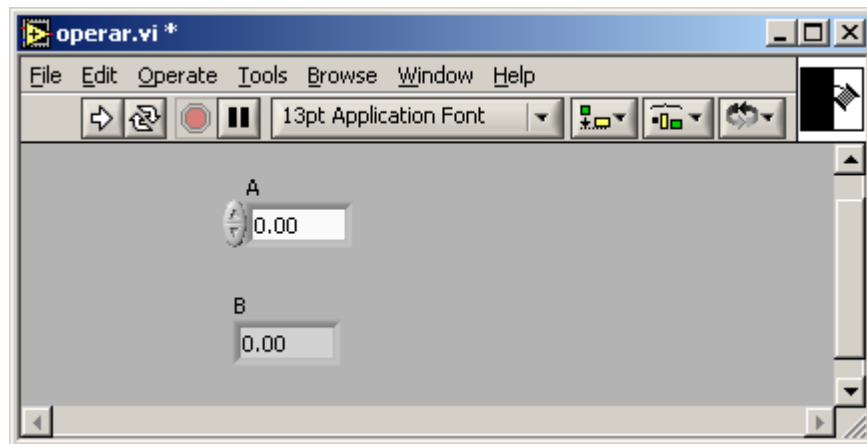


Figura 5.8. Conector Seleccionado.

2. Con la herramienta de cableado se selecciona el control o el indicador que se desea asignar a ese conector.

Automáticamente el área tomará el color correspondiente a la representación del control o indicador seleccionado. La figura 5.9 muestra como se asigna el Control A al conector seleccionado.

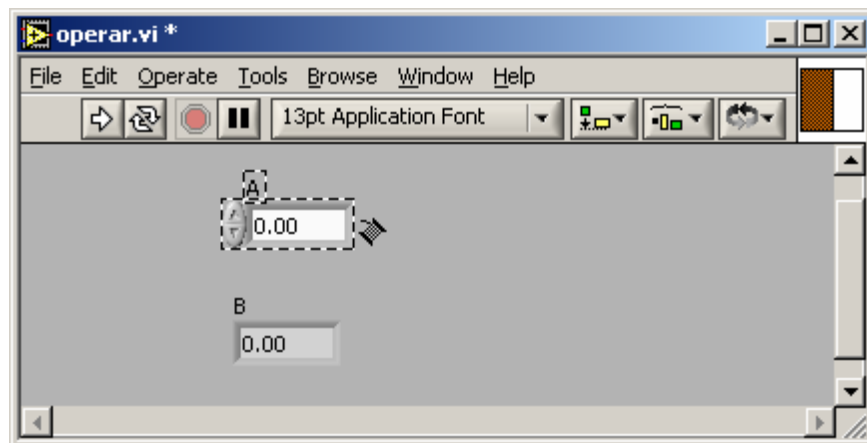


Figura 5.9. Control A asignado a un conector.

3. Se repite 1 y 2 para todos los conectores que se desee asignar.

Se recomienda que los controles se asocien a conectores del lado izquierdo del icono y los indicadores a conectores del lado derecho del icono.

4. Se puede clasificar las conexiones de entradas y salidas de un VI de acuerdo a la necesidad de ser conectadas en VIs de mayor jerarquía. Para la respectiva clasificación, del icono se selecciona la opción *This Connection Is*.

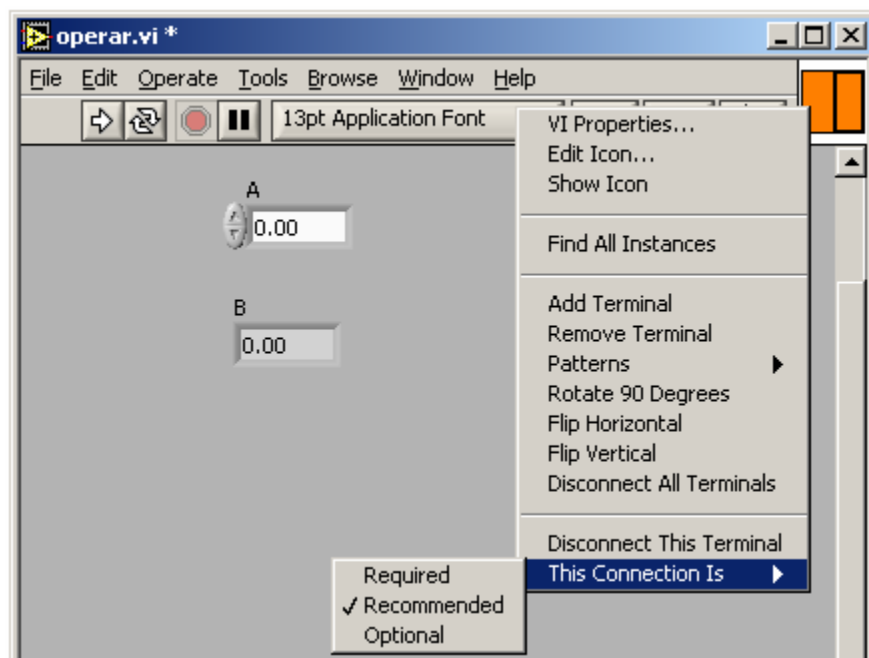


Figura 5.10. Clasificación de los conectores.

Required: La aplicación no podrá correr si no se alambra el respectivo control.

Recommended: La aplicación podrá correr, pero existirá una serie de prevenciones en la lista de errores reportados por LabVIEW.

Optional: La aplicación puede correr normalmente. Si no se cablea el control será cargado con su valor por defecto.

5. Del icono del panel se selecciona la opción “*show Icon*” para regresar a la vista de icono.
6. Se graba la aplicación con la opción <CTRL+S> o del menú **File>>Save**.

5.5 UTILIZACIÓN DE UN SUBVI

Para adicionar un subVI en el diagrama de un VI, se sigue los siguientes pasos:

1. Del diagrama del VI seleccione el menú **Select a VI** de la paleta de funciones. Ver figura 5.11.

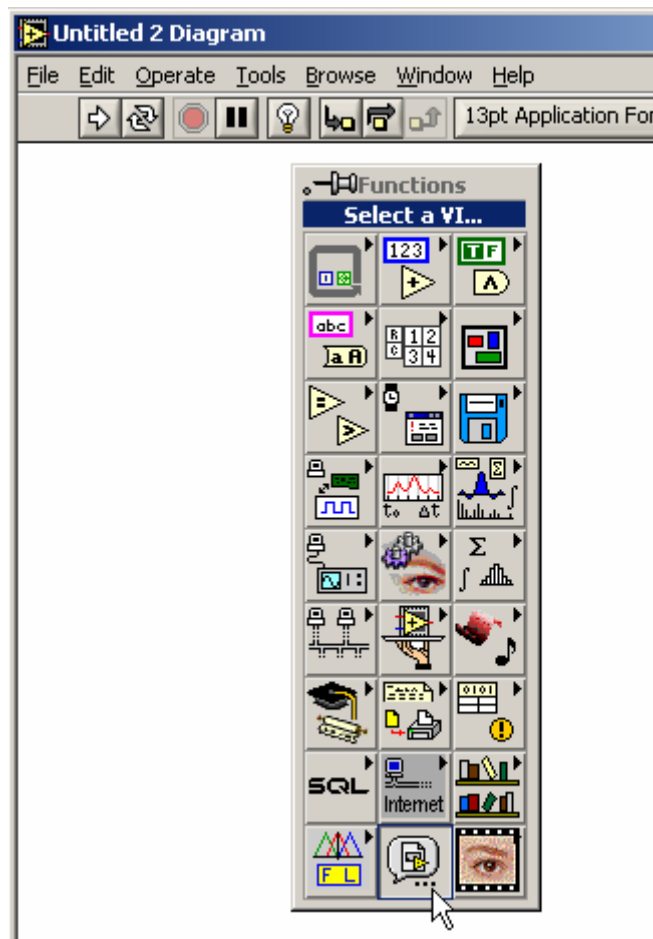


Figura 5.11. Menú *Select a VI* de la paleta de funciones.

2. La opción **Select a VI**, abrirá una ventana de diálogo donde se puede seleccionar el VI que se requiere utilizar como subVI. Ver figura 5.12.

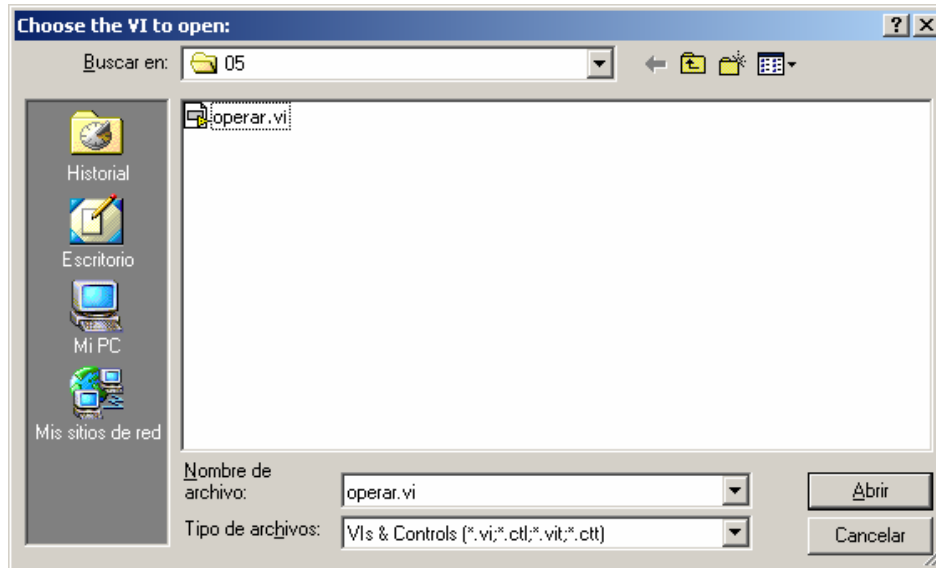


Figura 5.12. Ventana de diálogo de *Select a VI*.

3. Se busca y selecciona el VI deseado y luego se presiona Abrir.
4. Se ubica el cursor en el lugar del diagrama donde se desea localizar el subVI. Luego se hace un clic. Se observará el icono del subVI en el diagrama del VI en construcción.

Con la herramienta de cableado es posible obtener los nombres de los terminales del subVI sólo con acercarse a estos como se muestra en la figura 5.13.

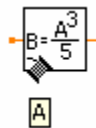


Figura 5.13. Terminales de un subVI.

Ahora el subVI está listo para ser utilizado como una función más dentro del diagrama.

En LabVIEW no está permitida la recursividad en el llamado de los subVIs. Es decir, un subVI no puede llamarse a si mismo.

EJERCICIO 5.1 FILTRADO DE UNA SEÑAL SENO CON TRES COMPONENTES ARMÓNICAS

Dada la señal:

$$V(t) = 5 \text{ sen } (\omega t) + 1.3 \text{ sen } (3 \omega t) + 0.3 \text{ sen } (5 \omega t)$$

Con $f = 60 \text{ Hz}$.

Y suponiendo que la única información conocida de las muestras obtenidas es el dt :

1. Utilizar un filtro para dejar pasar una banda definida por el usuario.
2. Graficar:
 - a) Señales original y filtrada.
 - b) Espectro de potencia de las señales original y filtrada.
3. Calcular para las señales original y filtrada:
 - a) Amplitudes y frecuencias armónicas.
 - b) THD.
 - c) Valor True-RMS

El panel frontal de esta aplicación podría tener el siguiente aspecto.

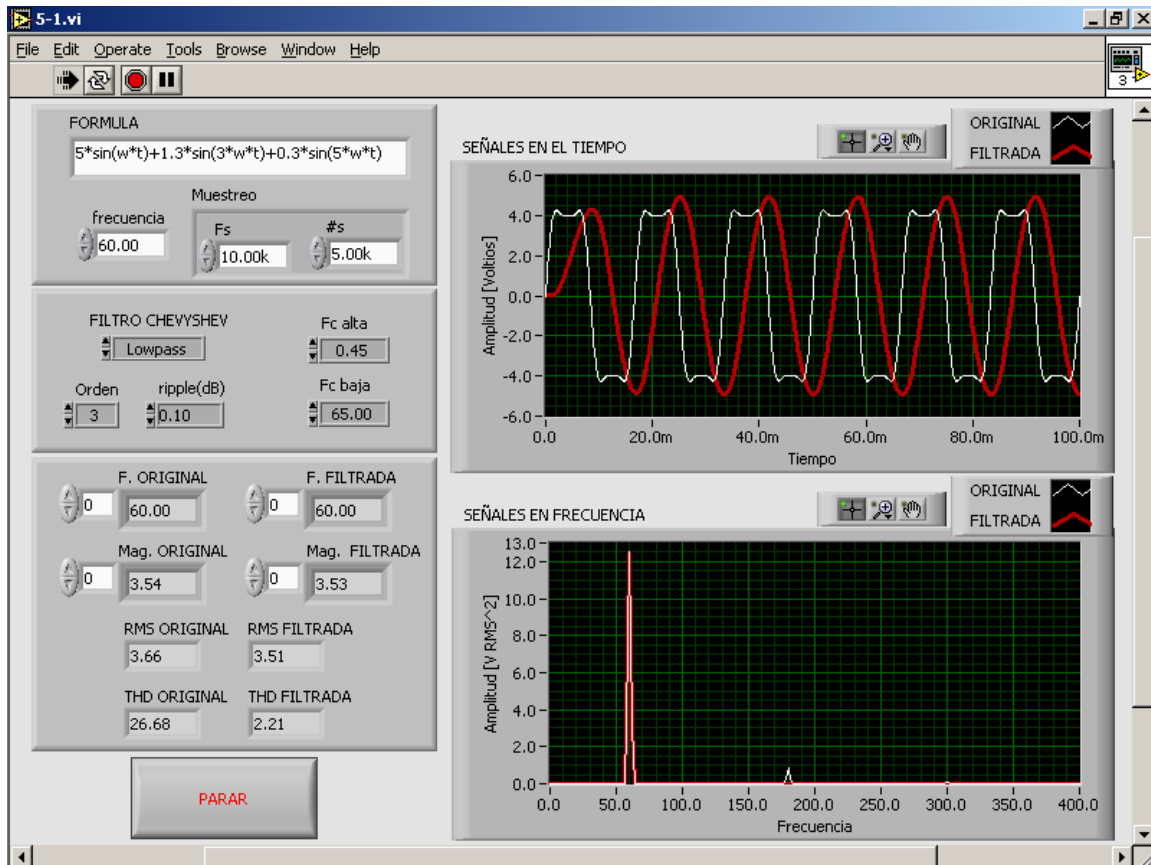


Figura 5.14. Panel frontal para el ejercicio 5.1.

Del enunciado del problema se puede observar que se requiere algunos datos dos veces, una vez para la señal original y otra para la señal filtrada.

Por consiguiente, lo más recomendable en este caso es construir una sola función que realice los cálculos requeridos para cualquier señal de entrada y luego aplicarla a cada señal solicitada.

El primer paso es entonces construir la función de cálculos que va a ser llamada "calculos.vi".

Para cualquier señal de entrada, conocido su Δt , se debe calcular:

- Espectro de potencia.
- Amplitudes armónicas.
- Frecuencias armónicas.
- Distorsión armónica total.
- Verdadero valor RMS.

LabVIEW posee herramientas de procesamiento digital de señal que pueden ser utilizadas para realizar los cálculos requeridos.

En la paleta de funciones en el menú **Analyze>>Signal Processing>>Frequency domain**, se puede encontrar las funciones necesarias para dicha tarea. La figura 5.15 muestra el diagrama de bloques que calcula los valores solicitados.

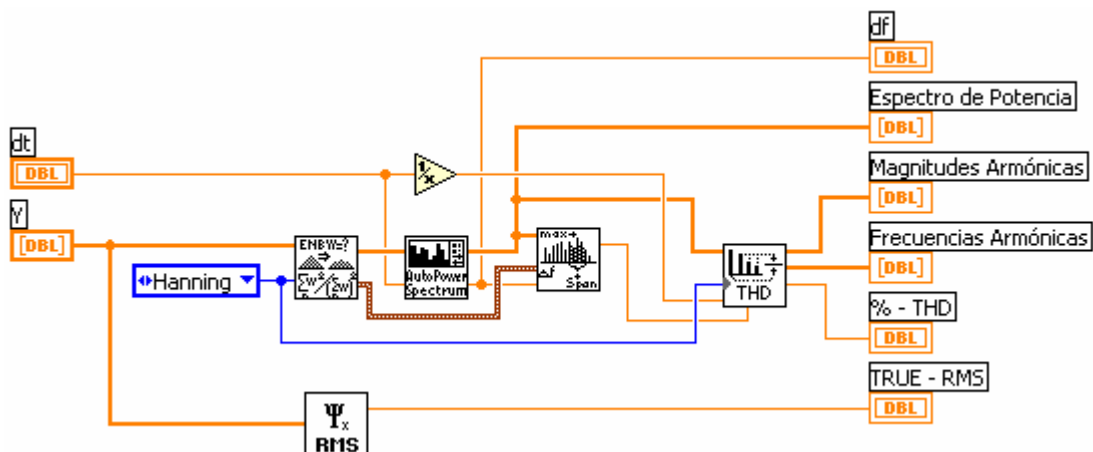


Figura 5.15. Diagrama de bloques del subVI "calculos.vi".

La primera función utilizada es "Scaled Time Domain Windows" y se encuentra en **Analyze>>Signal Processing>>Windows**. Es encargada de aplicar la ventana seleccionada a una señal en el tiempo.

Las siguientes tres funciones utilizadas se encuentran en **Analyze>>Signal Processing>>Frequency domain** y son en su orden:

Auto Power Spectrum: Calcula el espectro de potencia de una señal en el tiempo.

Power & Frequency Estimate: Calcula la potencia de la señal y la frecuencia fundamental.

Harmonic Analyzer: Calcula entre otros la THD y las amplitudes y frecuencias de las componentes armónicas de una señal.

Por último la función que calcula el valor RMS se encuentra en **Analyze>>Mathematics>>Probability and Statistics** y se denomina **RMS**.

Si se desea obtener más información acerca de las funciones de procesamiento de señal, usted puede consultar la ayuda rápida de LabVIEW, presionando <CONTROL + H> o en la ayuda en línea en el menú **HELP**.

Para este diagrama de bloques habrá un panel frontal sencillo con los controles e indicadores necesarios como se muestra en la figura 5.16.

Para que este nuevo VI pueda ser utilizado como un subVI dentro de otro de mayor jerarquía se debe definir los conectores que permitirán la entrada y salida de datos.

Para definir los conectores se debe seguir los pasos descritos en el ítem 5.4 de este mismo capítulo. La figura 5.17 muestra el icono con los conectores asignados.

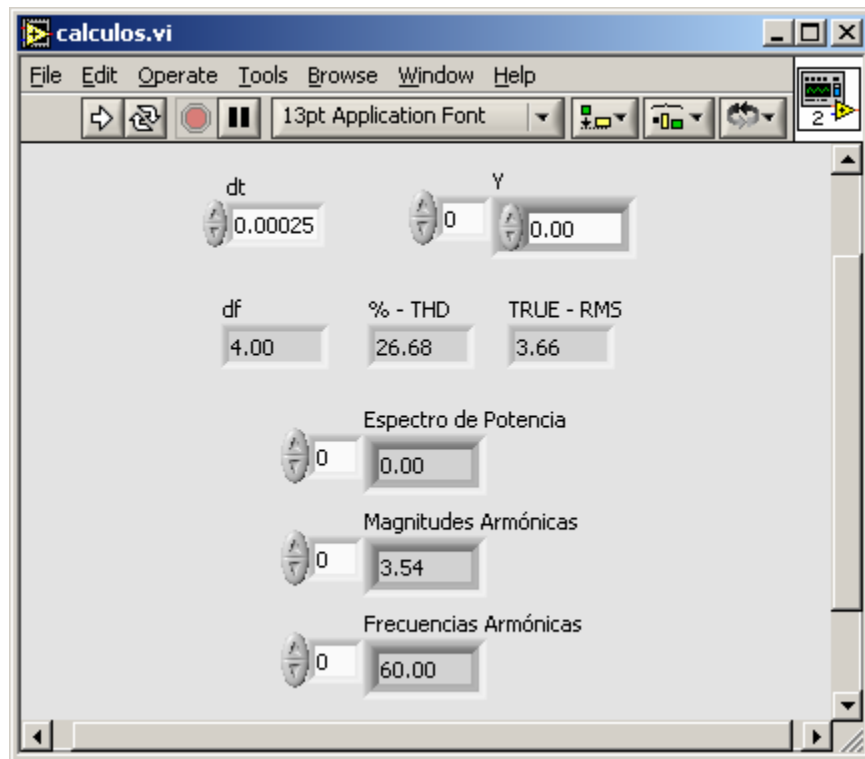


Figura 5.16. Panel frontal de “calculos.vi”.

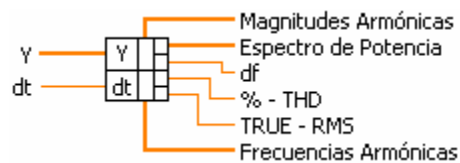


Figura 5.17. Icono conectado de “calculos.vi”.

Ahora este VI está listo para ser guardado y llamado desde un VI de mayor jerarquía.

El diagrama de bloques de la aplicación general del ejercicio 5.1 se muestra en la figura 5.18.

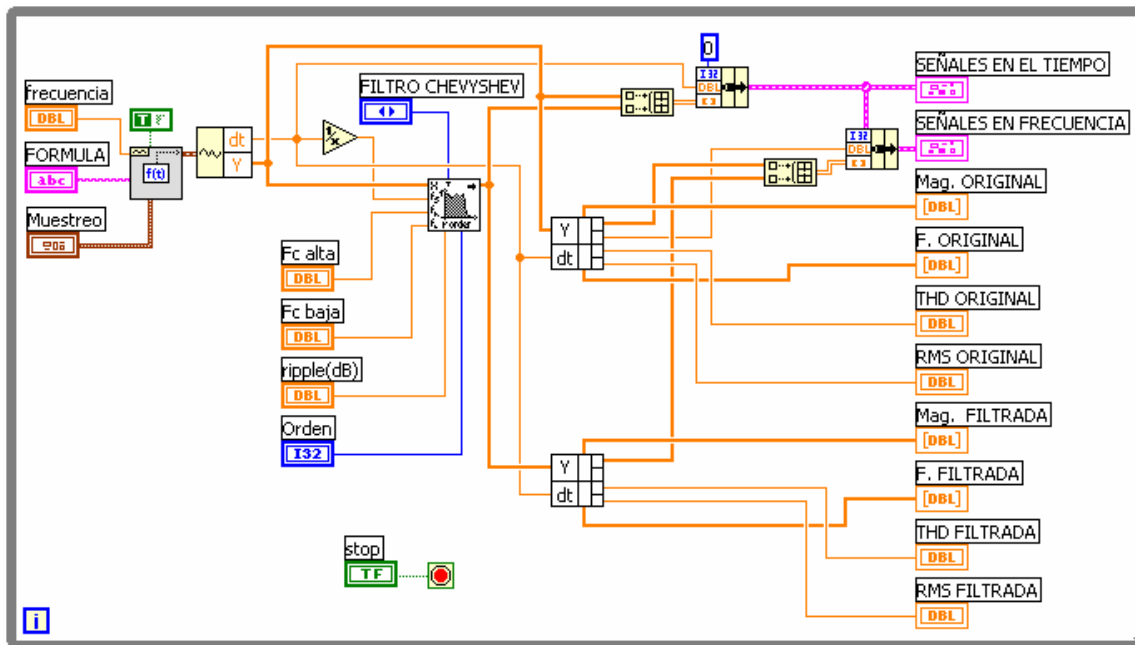


Figura 5.18. Diagrama de bloques del ejercicio 5.1.

La función que implementa un filtro chebyshev se puede encontrar en la paleta de funciones en el submenú **Analyze>>Signal Processing>>Filters** y se denomina *chebyshev filter*.

Este VI al que se le ha llamado “5-1.vi”, es el de mayor jerarquía dentro de la estructura del programa.

Si se desea obtener la estructura jerárquica del VI, se debe buscar la opción *Show VI Hierarchy* del menú *Browse* de la barra de menú. La jerarquía para este caso se ve en la figura 5.19.

El VI de mayor jerarquía es denominado “*TOP VI LEVEL*”, todos los demás son los subVIs con los que está programado el *Top VI Level*, y a su vez, estos subVIs están formados por otros subVIs y así sucesivamente.

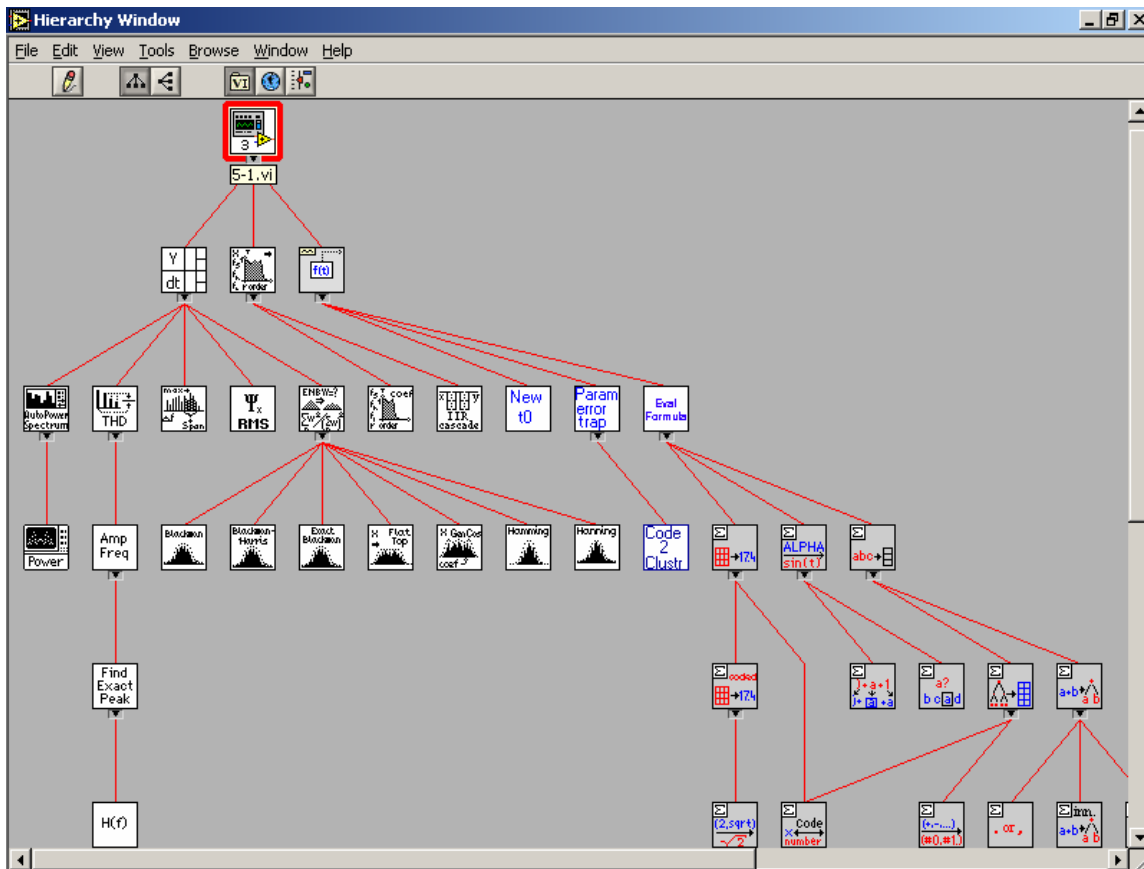


Figura 5.19. Ventana de jerarquía para “5-1.vi”

FIN EJERCICIO 5.1

Es posible alterar la forma como un VI se ejecuta o se comporta, su apariencia y algunas otras propiedades generales.

Las propiedades de un VI pueden ser obtenidas en el menú *FILE*, en el icono del VI haciendo clic derecho y seleccionando *VI Properties*, o con el método de teclado <CONTROL + I>.

La figura 5.20 muestra la ventana que se obtiene al seleccionar *VI Properties*.

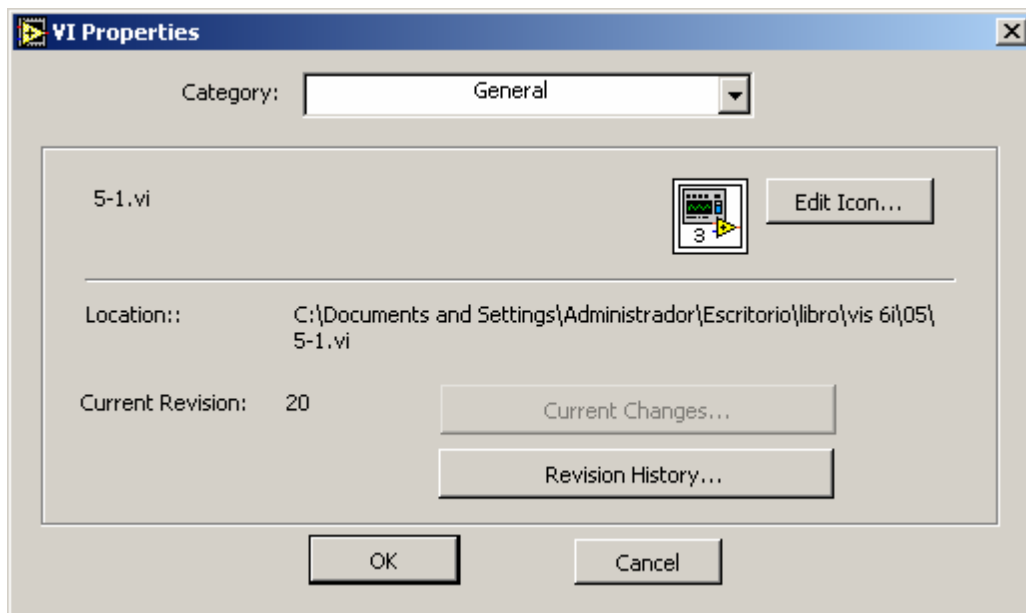


Figura 5.20. Ventana de *VI Properties*.

La opción *Category*, permite navegar entre las diferentes opciones de configuración como General, Seguridad, Ejecución, Ventana, Impresión, entre otras.

Se puede consultar una referencia completa acerca de cada una de las opciones de propiedad en la ayuda en línea de LabVIEW, *Contents and Index* en el menú *HELP*.

EJERCICIO 5.2 MENÚ EN LOS PANELES FRONTALES

Se desea que el usuario pueda seleccionar una opción entre un conjunto de ellas. Para eso se debe construir un panel frontal como el de la figura 5.21:

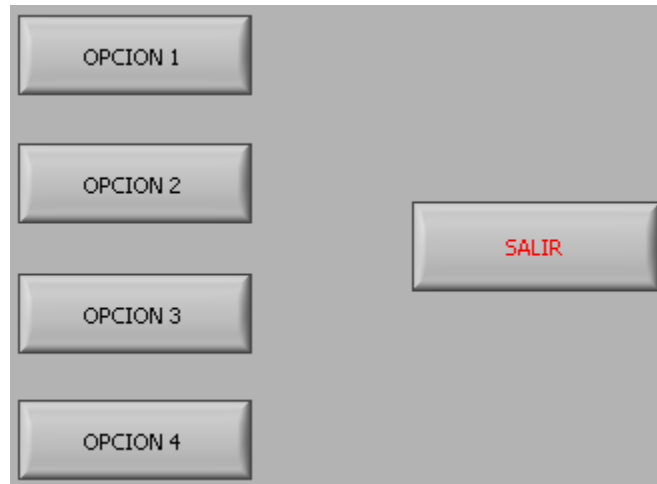


Figura 5.21. Panel frontal ejercicio 5.2.

Para los botones de opción se escogió *OK Button* de la paleta de controles del menú *Boolean* y como botón SALIR se escogió *STOP Button* de la misma paleta. Las etiquetas y textos de los botones se pueden cambiar con la herramienta de texto.

Se puede hacer uso también de las herramientas de alineación y distribución para dar un aspecto ordenado al panel.

El usuario tiene la posibilidad de escoger entre alguna de las cuatro opciones disponibles. El diagrama correspondiente a esta parte se muestra en la figura 5.22.

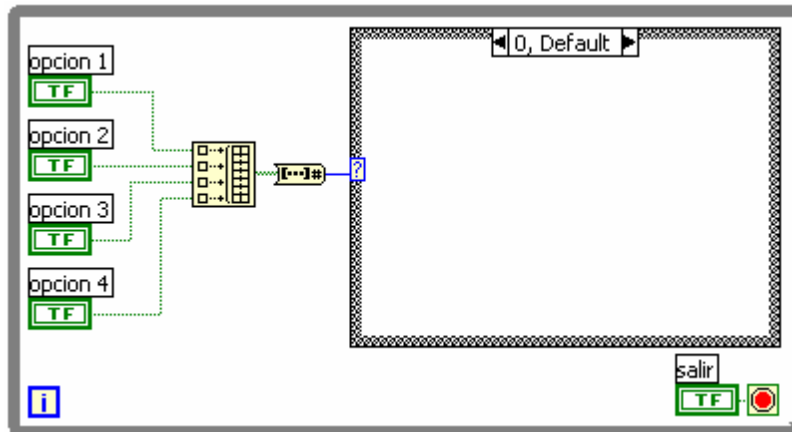


Figura 5.22. Diagrama del ejercicio 5.2.

Los cuatro terminales de los controles booleanos son llevados a la función *Build Array* para construir un arreglo booleano. Luego dicho arreglo es convertido en un número utilizando la función *Boolean Array To Number* que se encuentra en la paleta de funciones en el submenú *Boolean*.

Un arreglo booleano convertido en un número es la representación decimal del equivalente binario del arreglo booleano. Esto quiere decir que este número sólo tomará valores potencias de 2 (0, 1, 2, 4, 8, 16, 32, ...).

Así, cuando no hay ningún control presionado se formará el arreglo booleano 0000 y al selector de la estructura *CASE* llegará un cero (0). Si se presiona el botón OPCION 1, se formará el arreglo booleano 1000 donde el primer bit es el menos significativo, por lo tanto equivale al número decimal 1 que llegará al selector del *CASE*. Si se presiona OPCION 4, se formará 0001, donde el primer bit es el menos significativo, por lo tanto el decimal 8 llegará al selector del *CASE*.

Los otros tres *frames* de la estructura *CASE* se muestran en la figura 5.23.

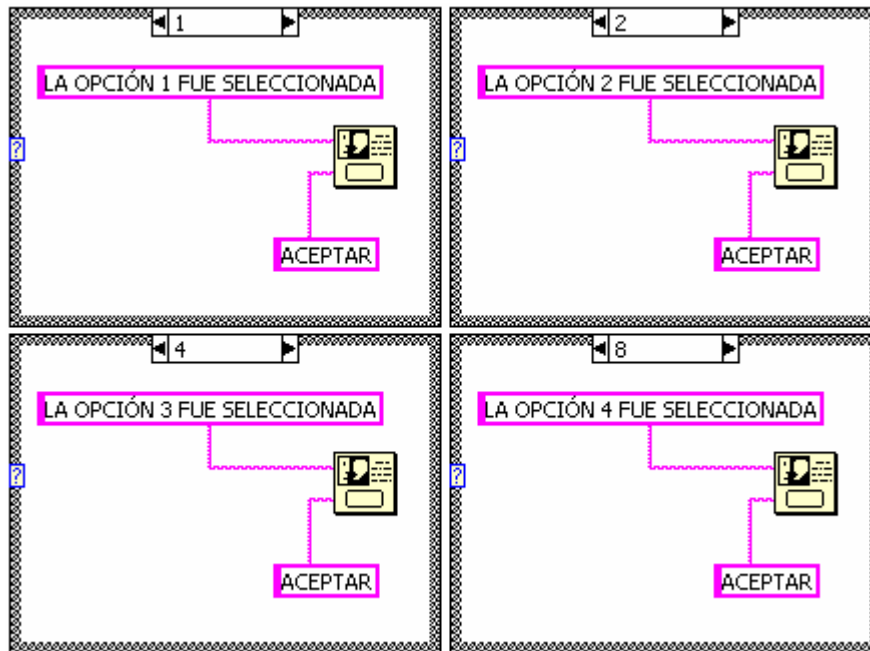


Figura 5.23. Casos de la estructura CASE en el ejercicio 5.2.

Para mostrar un cuadro de diálogo simple, se puede utilizar la función *One Button Dialog* ubicada en el menú *Time&Dialog* de la paleta de funciones. Las constantes tipo cadena se pueden encontrar en la paleta de funciones en el submenú *String*.

Nótese que cada opción puede ser un subVI encargado de realizar alguna tarea específica.

FIN EJERCICIO 5.2

EJERCICIO 5.3 CARGAR EL PANEL FRONTAL DE UN SUBVI DURANTE LA EJECUCIÓN

Si la OPCION 1 del menú anterior tuviera como tarea solicitarle al usuario que ingrese sus datos, se requeriría de un subVI que muestre su panel frontal cuando es llamado.

Lo primero que se debe hacer es un nuevo VI que solicite los datos del usuario.

El panel sugerido tiene la apariencia de la figura 5.24.

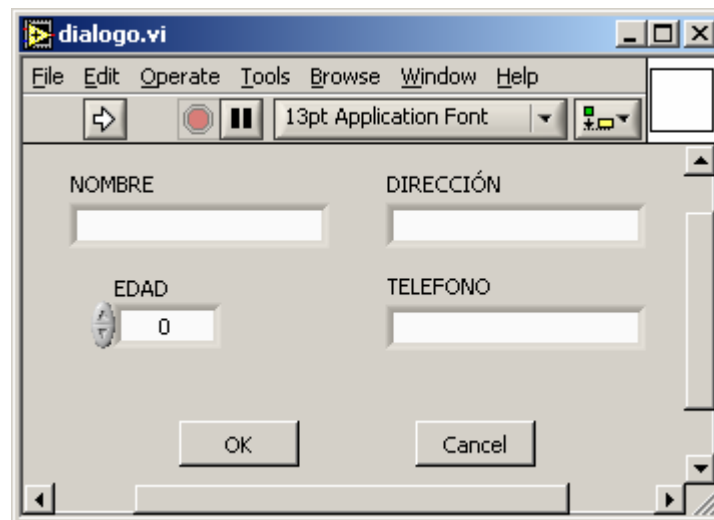


Figura 5.24. Panel de diálogo del ejercicio 5.3.

Los botones *OK* y *CANCEL* son obtenidos de la paleta de controles en el submenú *Boolean*. Nombre, Dirección y Teléfono son controles tipo *STRING* ubicados en la paleta de controles en el submenú *String&Path*.

El diagrama correspondiente se muestra en la figura 2.25.

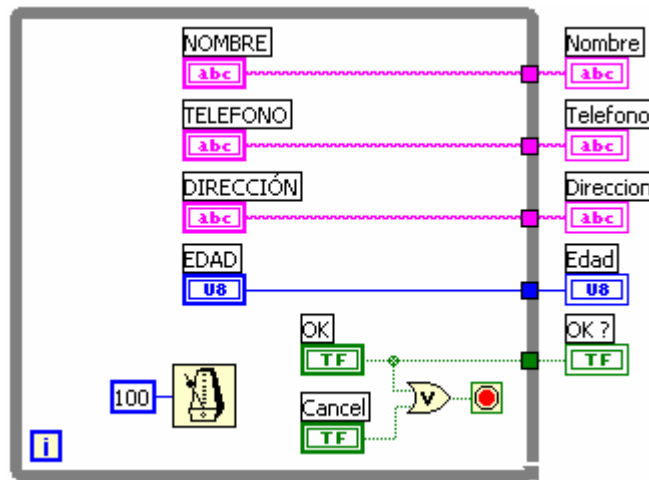


Figura 5.25. Diagrama para el VI de diálogo del ejercicio 5.3.

Los controles, son ubicados dentro del ciclo *WHILE*, con el fin que puedan ser manipulados por el usuario hasta que se presione *OK* o *CANCEL*.

Los indicadores cumplen la función de pasar los datos a otro subVI. Como no se requieren observar en el panel, del menú de cada uno de los terminales de indicador se selecciona la opción *Hide Indicator*.

Terminado el VI debe crearse su icono y sus conectores según el procedimiento descrito en 5.3 y 5.4.

En este caso, sólo es necesario crear conectores para los indicadores. La figura 5.26 muestra el icono creado y los conectores.

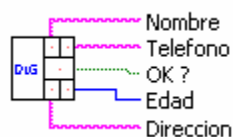


Figura 5.26. Icono del VI de diálogo del ejercicio 5.3.

Como propone este ejercicio, se debe reemplazar el *frame 1* de la estructura *CASE* del ejercicio 5.2 por el VI de diálogo.

Además se agregará un indicador al panel frontal para determinar cual ha sido la última operación realizada.

El panel frontal modificado para el ejercicio 5.3 se muestra en la figura 5.27.

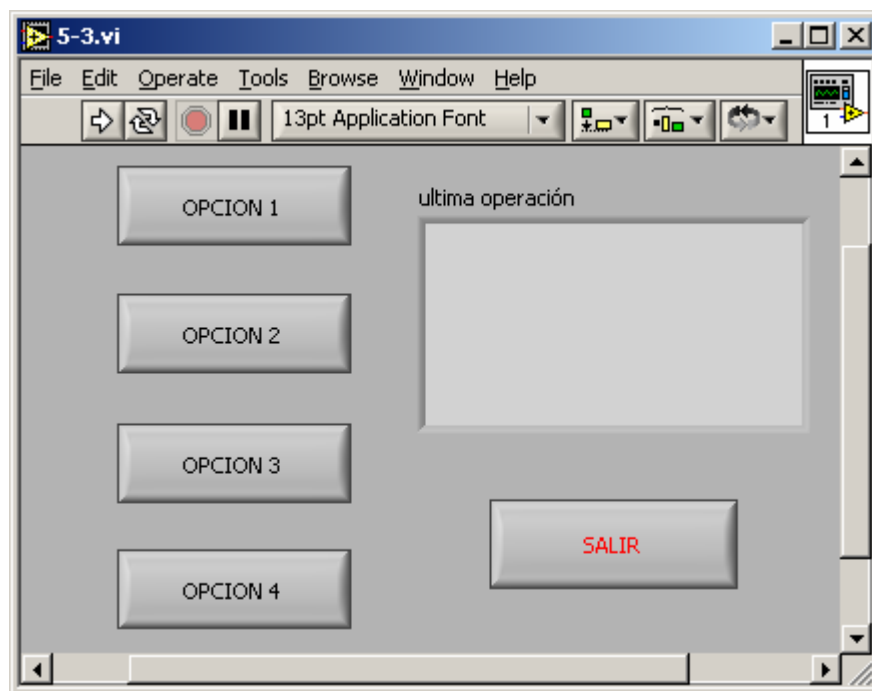


Figura 5.27. Panel frontal del ejercicio 5.3.

El diagrama de bloques modificado se muestra en la figura 5.28.

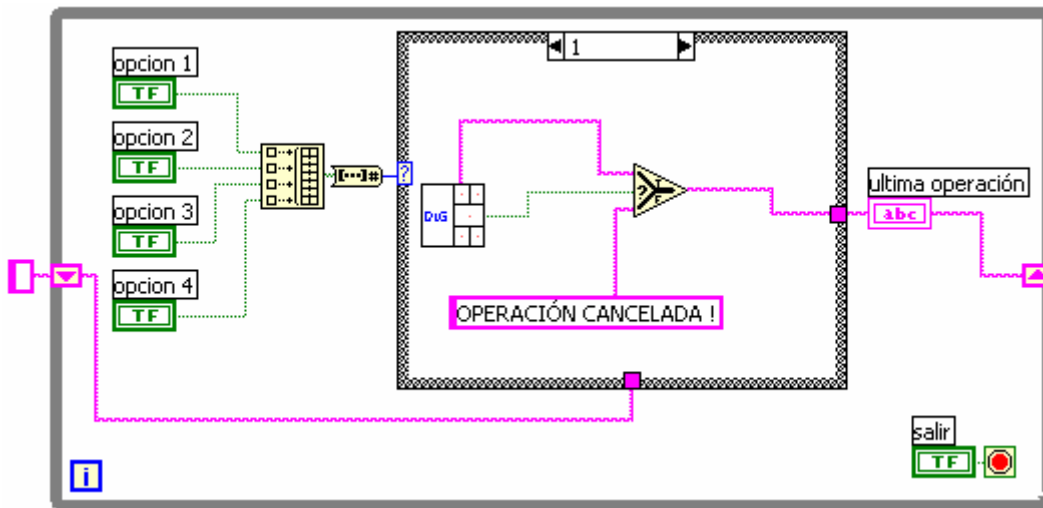


Figura 5.28. Diagrama del ejercicio 5.3.

La salida *OK?* del subVI de diálogo se utiliza para determinar si la operación de introducir los datos fue o no cancelada. En caso de serlo, al terminal "ultima operación" que es un indicador tipo *STRING*, llegará la constante "OPERACIÓN CANCELADA !", en caso contrario se cablea el nombre escrito por el usuario.

El *shift register* se utiliza para almacenar el último valor del indicador tipo *STRING*.

El túnel que se genera a la salida de la estructura *CASE* se debe alimentar desde todos los casos. Como el ejemplo busca indicar la última operación entonces se puede cablear la misma constante usada para el cuadro de diálogo. Un ejemplo de esto para el *frame 2* se muestra en la figura 5.29.

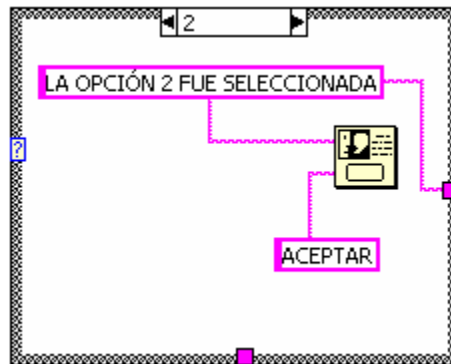


Figura 5.29. Túnel cableado desde los otros casos.

Sin embargo el VI todavía no está listo para correr. Si se intenta correr el VI, se observará que las opciones 2, 3 y 4 funcionan normalmente, pero la opción 1 no realiza su trabajo y hace que el VI deje de funcionar correctamente.

Esto se debe a que el subVI “dialogo.vi” comienza su trabajo pero no muestra su panel frontal para que el usuario pueda ingresar sus datos.

La forma de lograr este comportamiento en un VI es con el menú de configuración de nodo (*SubVI Node Setup*), que se encuentra en el menú del icono de la función como se muestra en la figura 5.30.

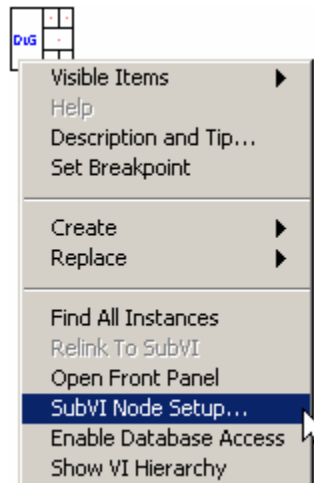


Figura 5.30. *SubVI Node Setup*.

Esta acción mostrará la ventana de la figura 5.31. Se debe activar las opciones de “Mostrar el panel cuando sea llamado” y “Cerrar el panel si originalmente es cerrado”. Esto permitirá que el subVI muestre su panel cuando sea llamado y lo cierre cuando termine de ejecutarse.

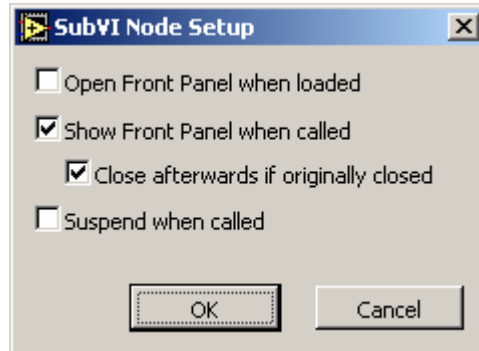


Figura 5.31. Ventana de *SubVI Node Setup*.

Es recomendable que el usuario final no pueda abortar la ejecución del programa mientras el cuadro de diálogo este activo. Para ello el VI se puede configurar como cuadro de diálogo en el menú *VI Properties* en la categoría *Window Appearance* del VI de diálogo.

FIN EJERCICIO 5.3

5.6 VARIABLES LOCALES

Las variables locales en LabVIEW son copias del terminal de un control o indicador que se pueden utilizar en cualquier lugar del diagrama para leer o escribir datos a ese control o indicador.

Son locales, porque su alcance se limita a un mismo VI. Para obtener una variable local, se debe buscar la opción **Create>>Local Variable** del menú del objeto como se muestra en la figura 5.32.

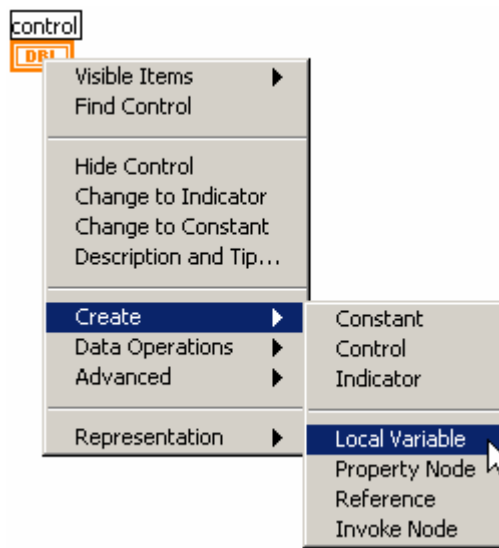


Figura 5.32. Creación de una variable local.

Sólo es posible crear una variable local de un objeto si este tiene etiqueta. El aspecto de una variable local es similar al de un terminal, toma el color representativo del tipo de variable y muestra en su interior la etiqueta del control o indicador al que pertenece. Una variable local del control numérico de la figura 5.32 es mostrada en la figura 5.33.



Figura 5.33. Variable Local.

Por defecto, las variable locales se crean en modo de escritura. Es decir, operan como si fueran un indicador, pues únicamente reciben datos.

Para cambiar el sentido del flujo de los datos es necesario hacer clic derecho y cambiar a modo lectura, o a escritura si la variable esta en modo lectura. La figura 5.34 muestra la forma de cambiar el sentido de los datos para la variable local de la figura 5.33 y el aspecto que toma cuando está en modo lectura.

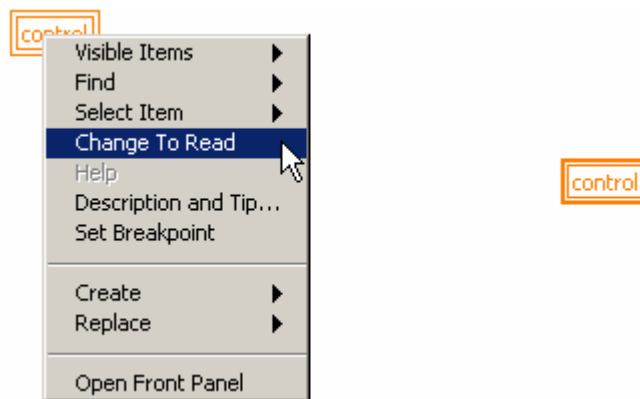


Figura 5.34. Cambio entre modos Lectura/Escritura.

De esta manera, es posible leer datos de un indicador y actualizar el valor de un control.

EJERCICIO 5.4 GRAFICAR DATOS DE DISTINTA FUENTE EN UN MISMO CHART

Se requiere un programa en LabVIEW que cumpla la siguiente secuencia de tareas:

- a) Generar 50 datos aleatorios entre 0 y 1 con intervalos de 20ms y graficarlos a medida que se generan.
- b) Mostrar un cuadro de diálogo con un mensaje que diga “50 datos entre 0 y 1 terminados”
- c) Generar 50 datos aleatorios entre 1 y 2 con intervalos de 15ms y graficarlos a medida que se generan en el mismo *CHART* utilizado en a.

Este es un problema que no puede resolverse sin utilizar una variable local.

Evidentemente se requiere de una estructura *SEQUENCE* para realizar las tres etapas del programa.

La solución al ítem a se muestra en la figura 5.35.

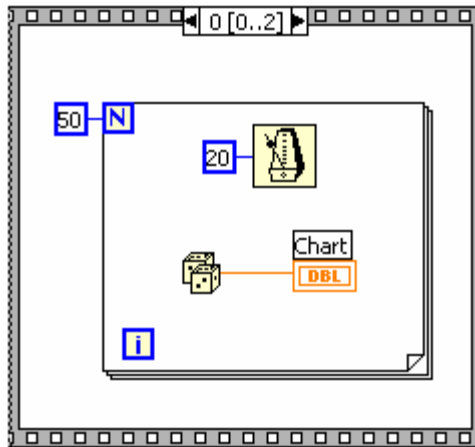


Figura 5.35. *FRAME 0* de la estructura *SEQUENCE*.

El panel frontal de este ejercicio únicamente cuenta con un graficador tipo *CHART*, escalado para mostrar 100 datos entre 0 y 2. La figura 3.36 muestra el panel frontal.

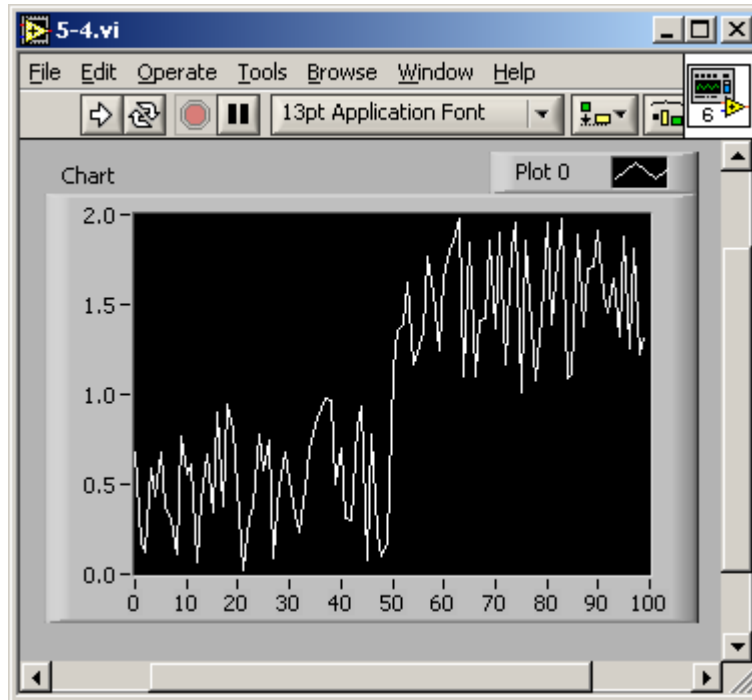


Figura 5.36. Panel frontal del ejercicio 5.4.

La solución al ítem b constituye el *FRAME 1* de la estructura de secuencia y se observa en la figura 5.37.

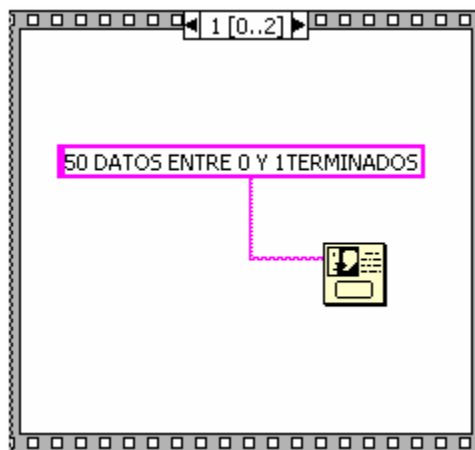


Figura 5.37. *FRAME 0* de la estructura *SEQUENCE*.

El problema aparece al intentar construir el *FRAME 2* para dar solución al ítem c del presente ejercicio.

Es posible generar los 50 datos entre 1 y 2 con intervalos de 15ms, pero para graficarlos en el mismo *CHART* se requiere una variable local de *CHART* en modo escritura para poder enviarle datos desde el *FRAME 2* de la secuencia.

La figura 5.38 muestra el *FRAME 2* utilizando una variable local de *CHART*.

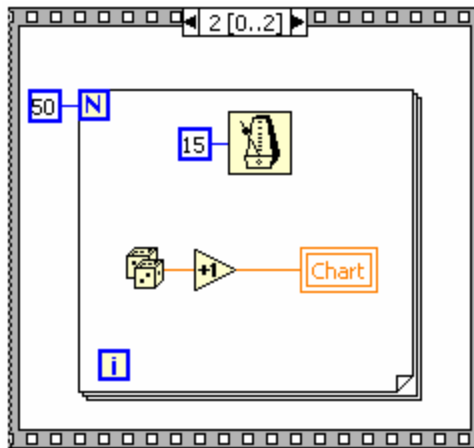


Figura 5.38. *FRAME 0* de la estructura *SEQUENCE*.

FIN EJERCICIO 5.4

En muchas ocasiones se tiene tareas distribuidas en dos ciclos *WHILE* independientes y se requiere que ambos ciclos se detengan a la misma orden de *PARO*.

La figura 5.39 muestra dos ciclos *WHILE* que deben ser detenidos simultáneamente.

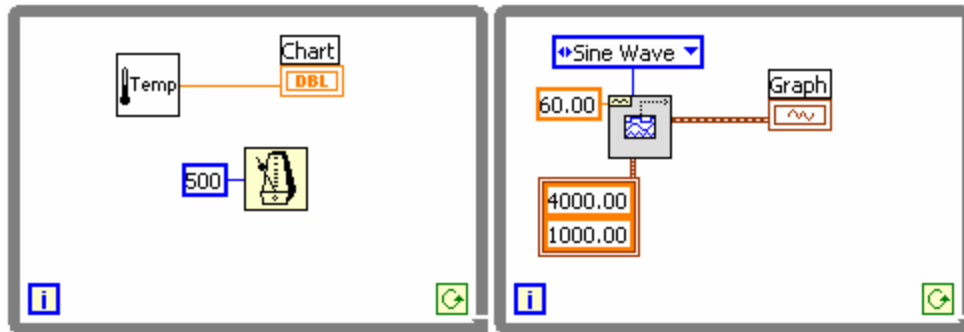


Figura 5.39. Dos ciclos *WHILE* simultáneos.

La tarea del ciclo de la izquierda es recoger un dato de temperatura cada 500 ms, mientras el de la derecha genera continuamente una señal seno de 60 Hz. Se requiere que las dos tareas se realicen simultáneamente. La figura 5.40 muestra una forma INCORRECTA para detener los dos ciclos.

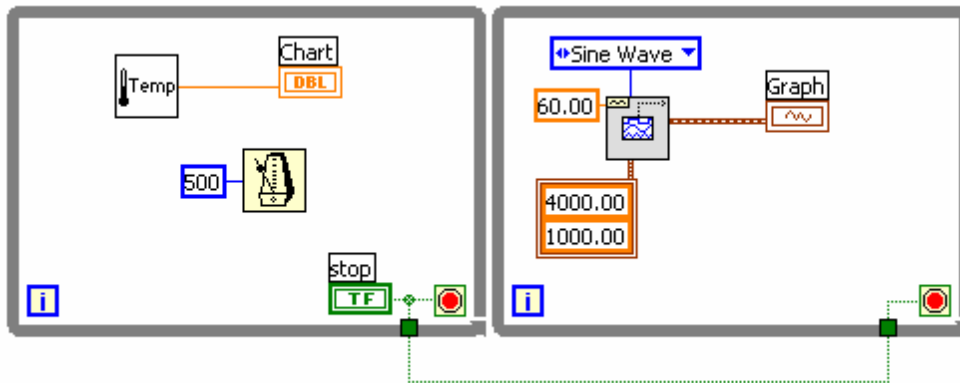


Figura 5.40. Método incorrecto para detener dos ciclos simultáneos.

Aunque no hay errores de cableado, la operación de este VI será anormal, ya que el ciclo de la derecha no comenzará su trabajo hasta que se presione el botón de paro y sólo se ejecutará una vez. Una segunda opción se muestra en la figura 5.41.

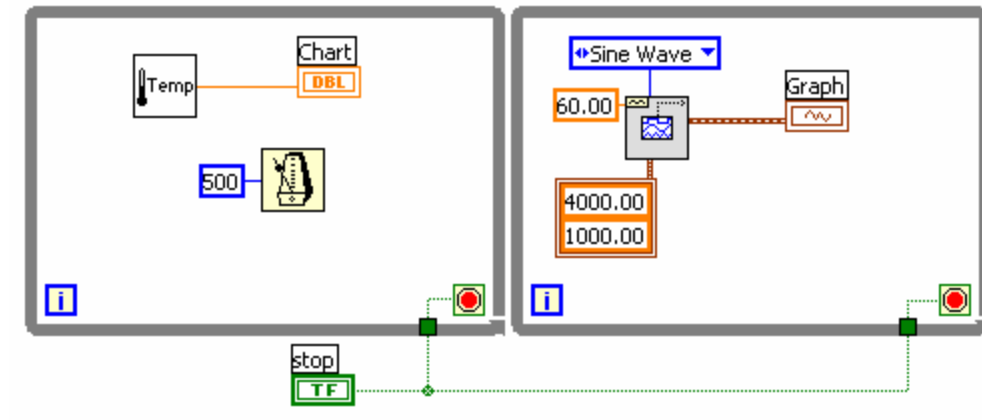


Figura 5.41. Segunda opción para detener dos ciclos.

Evidentemente esta opción es también INCORRECTA, pues aunque los dos ciclos se ejecutarán simultáneamente, el botón de paro sólo será leído una vez, haciendo que el VI caiga en un ciclo infinito.

La forma de detener este programa sería con el botón "Abortar" en la barra de herramientas. Si esta herramienta está deshabilitada o el VI está compilado, será necesario finalizar LabVIEW con el método respectivo del sistema operativo utilizado.

La única forma posible de cumplir con la tarea solicitada es utilizando una variable local del botón de paro como se muestra en la figura 5.42.

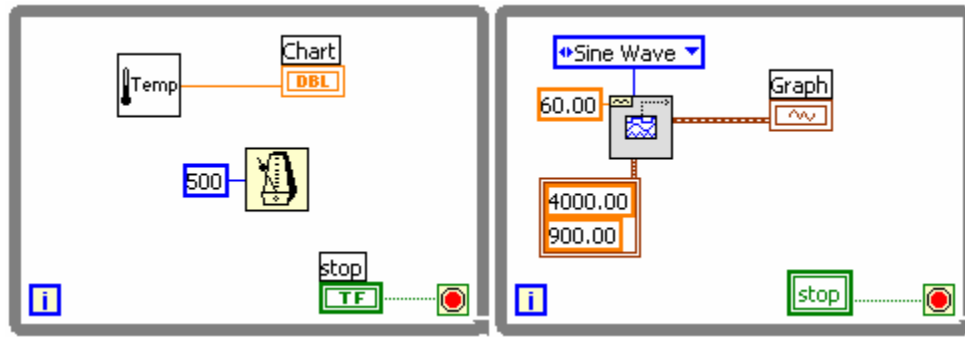


Figura 5.42. Forma correcta de detener dos ciclos simultáneos.

Esta configuración permite que los dos ciclos se ejecuten simultáneamente y puedan ser detenidos con un solo control de paro. Sin embargo, es necesario añadir, que cuando se crea una variable local a un booleano, sólo se puede utilizar las tres acciones mecánicas básicas (*switch*). Por tanto, se debe fijar un valor FALSO para el botón de paro tanto al inicio como al final del programa. Esto se muestra en la figura 5.43.

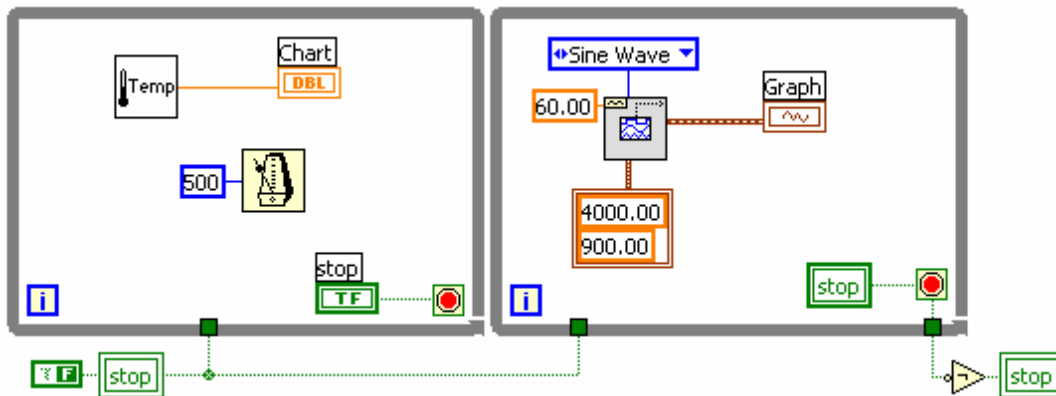


Figura 5.43. Diagrama terminado del ejercicio 5.5.

5.7 VIARIABLES GLOBALES

Las variables globales de LabVIEW son VIs que únicamente poseen panel frontal. Su utilidad se similar a la de las variables locales, pero su rango de aplicación es más amplio ya que pueden operar entre subVIs.

Las variables globales también se pueden configurar como lectura o escritura dependiendo de si se desea obtener o actualizar el dato que ellas almacenan.

Las variables globales se pueden crear seleccionando la opción **New...** del menú **File** o desde la paleta de funciones seleccionando *global* del submenú *structures*. La figura 5.44 muestra esta última opción.

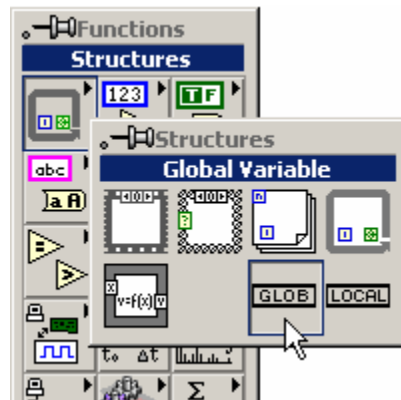


Figura 5.44. Creación de una variable Global.

Al colocar la variable global en el área de trabajo del diagrama ella toma el aspecto de la figura 5.45.



Figura 5.45. Variable global sin definir.

Las variables globales pueden estar agrupadas en un solo archivo que puede tener extensión .VI o .GLB. Para asignar los datos que formarán parte de un grupo de variables globales basta con hacer doble clic sobre el icono de la figura 5.45 para que se muestre un panel frontal como el de la figura 5.46.

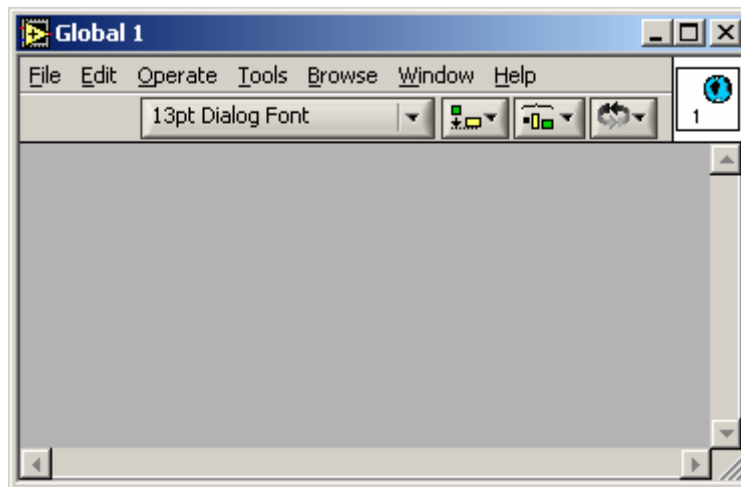


Figura 5.46. Panel de una variable global.

En este panel podrán ser colocadas todas las variables que se desee. Por ejemplo la figura 5.47 muestra la variable global con varios datos asignados.



Figura 5.47. Variables globales creadas.

El panel se debe guardar para posteriormente ser llamado desde un subVI. Generalmente no es necesario editar el icono de una variable global, ya que es poco utilizado.

Después de realizado este procedimiento se debe seleccionar cual ítem será asignado al diagrama del subVI. Esto se logra con el menú del objeto como se muestra en la figura 5.48.

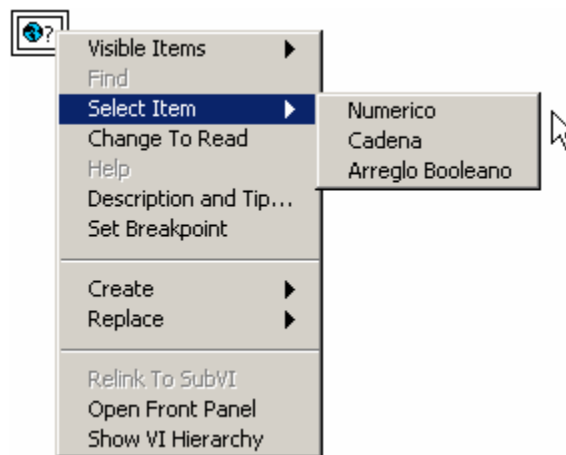


Figura 5.48. Selección de un ítem.

La variable global está definida ahora por el ítem seleccionado y su aspecto dependerá de la configuración en lectura o escritura tal como se muestra en la figura 5.49.



Figura 5.49. Variables globales en modo lectura y escritura respectivamente.

Es recomendable no exceder la utilización de variables globales ya que se podría ver decrementado el desempeño del VI.

EJERCICIO 5.5 VARIABLES GLOBALES

Se desea realizar una aplicación dividida en dos VIs.

El primero debe generar un número aleatorio cada determinado intervalo de tiempo. El segundo debe gráficar los datos generados por el primero a medida que se generan. El panel frontal y el diagrama de los dos VIs se muestran en las figuras 5.50 y 5.51 respectivamente.

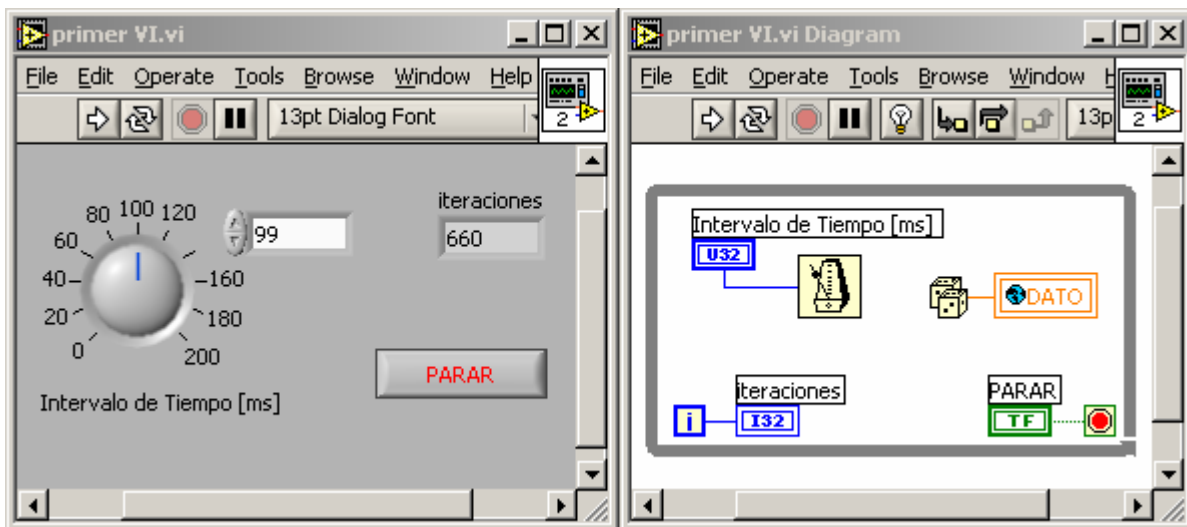


Figura 5.50. Panel y diagrama del primer VI.

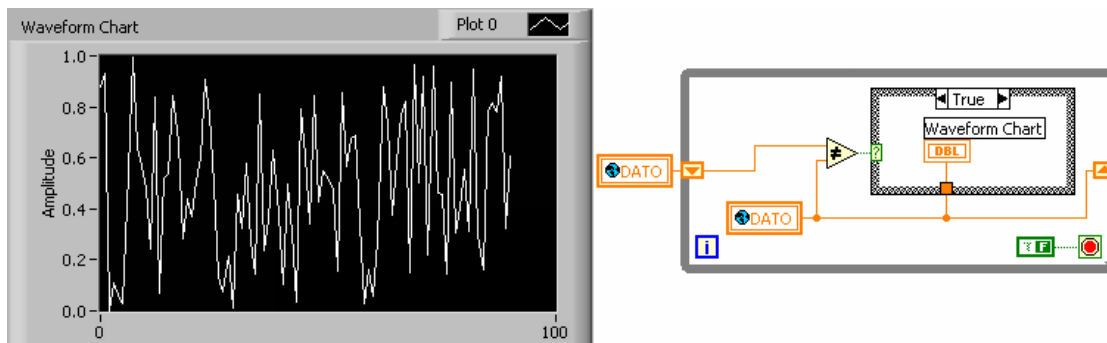


Figura 5.51. Panel y diagrama del segundo VI.

En este caso en particular se ha querido que el graficador sólo muestre un dato si este es diferente al anterior. Para esto se utiliza un *shift register* y un comparador de diferencia.

La variable global “DATO” esta creada en un archivo llamado “global.vi” cuyo panel se muestra en la figura 5.52.

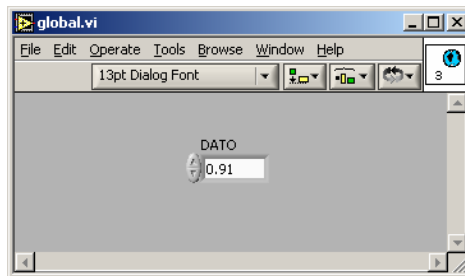


Figura 5.52. Variable global.

Para ver funcionando los dos VIs simultáneamente se pueden mostrar sobre el escritorio como en la figura 5.53.

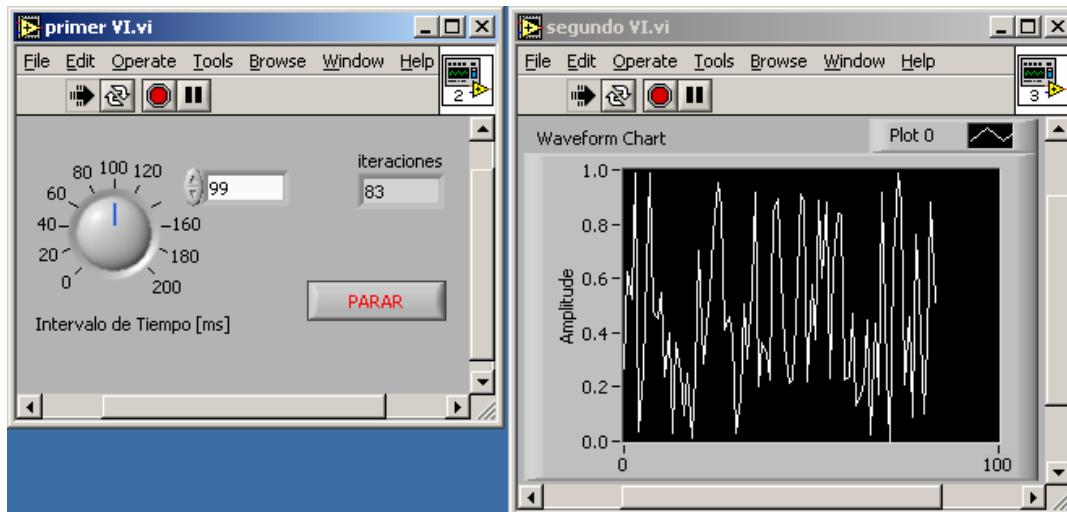


Figura 5.53. Ejecución simultanea de los dos VIs.

FIN EJERCICIO 5.5

5.8 EJERCICIOS PROPUESTOS

1. Construya un subVI que encuentre las raíces de la ecuación de segundo grado. Utilícelo en un VI que solucione tres ecuaciones cuadráticas simultáneamente.
2. Construya un panel frontal principal que posea un menú donde se puedan seleccionar 4 opciones además de SALIR. Cada opción deberá abrir un nuevo panel para realizar una de las cuatro operaciones básicas. Debe impedirse que después de seleccionar una opción el usuario regrese al menú principal sin antes cerrar la ventana que esta utilizando.
3. Genere un VI que grafique la temperatura de algún proceso utilizando Digital Thermometer. VI. Cuando la temperatura supere un límite permitido el color de la grafica debe cambiar de amarillo a rojo.
4. Analice la opción “reentrant execution” que se encuentra en las propiedades del VI. Utilice el primer ejercicio propuesto como ejemplo. Utilice la ayuda de LabVIEW para obtener la información necesaria.

6. CADENAS Y ARCHIVOS.

6.1 OBJETIVO

Estudiar el manejo de las variables tipo *String* o cadena y las funciones para el manejo de archivos que ofrece LabVIEW.

6.2 CADENAS

Una variable tipo *String* o cadena es una colección ordenada de caracteres ASCII.

Es muy común utilizar cadenas de datos o convertir datos numéricos en cadenas para posteriormente almacenarlas en disco, presentarlas en pantalla, enviarlos por el puerto serial o a través de la red.

Se puede encontrar controles e indicadores tipo cadena en la paleta de controles, en el submenú *String & Path*, como se observa en la figura 6.1.



Figura 6.1. Submenú *Strings & Path*.

Se puede diferenciar los controles y los indicadores por su color de fondo, ya que los primeros utilizan por defecto un fondo blanco mientras que los segundos uno gris.

Las funciones que permiten manipular cadenas se encuentran en la paleta de funciones en el submenú *Strings* como lo indica la figura 6.2.

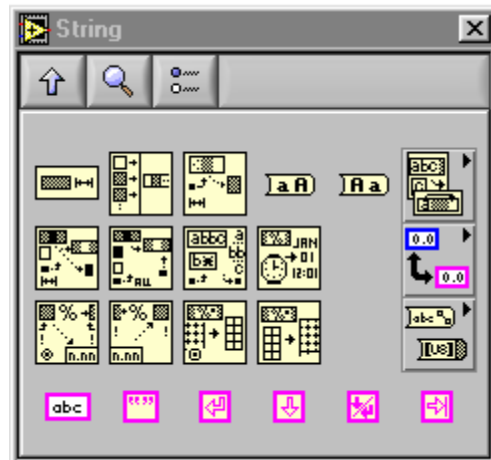


Figura 6.2. Funciones para manejo de cadenas.

Estas funciones permiten realizar tareas como determinar longitudes de cadena, agrupar varias cadenas, encontrar caracteres en una cadena, conversión de datos numéricos a tipo cadena y viceversa, agregar cadenas constantes y caracteres especiales, entre otras.

Las funciones de esta paleta se describen a continuación:

String Length

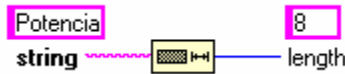


Figura 6.3. Función *String Length*.

Esta función es utilizada para determinar la longitud de una cadena de caracteres.

Concatenate Strings

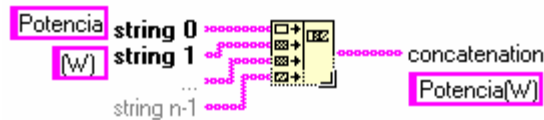


Figura 6.4. Función *Concatenate Strings*.

Enlaza n cadenas en estricto orden (desde *string 0* hasta *string n-1*) en una sola cadena de salida.

String Subset

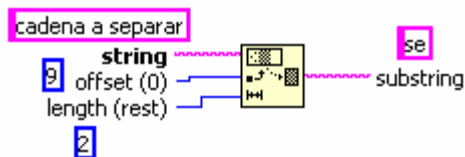


Figura 6.5. Función *String Subset*.

Esta función retorna una subcadena de la cadena de entrada, cuyo primer caracter esta determinado por la entrada *offset* y su longitud es indicada en *length*. Si esta entrada no es cableada se tomará como longitud el resto de la cadena.

To Upper Case

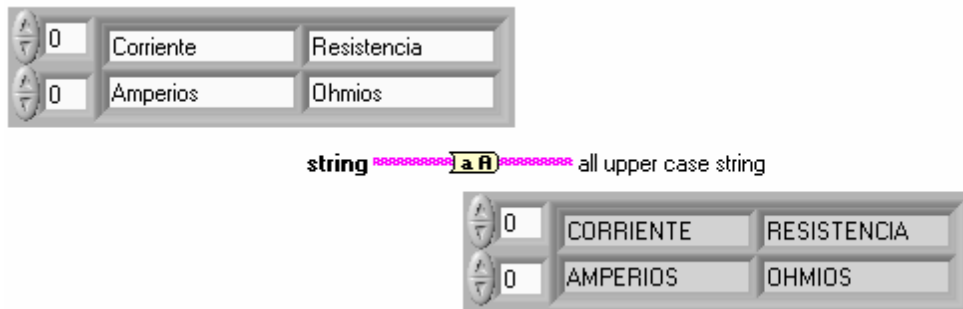


Figura 6.6. Función *To Upper Case*.

Convierte a mayúsculas todos los caracteres alfabéticos de una cadena o arreglo de cadenas. La función **To Lower Case** realiza el proceso inverso.

Replace Substring



Figura 6.7. Función *Replace Substring*.

Reemplaza una porción de tamaño *length* de una cadena por una subcadena a partir del *offset*. Si no se especifica una longitud se tomará por defecto el tamaño de la subcadena.

Search and Replace String

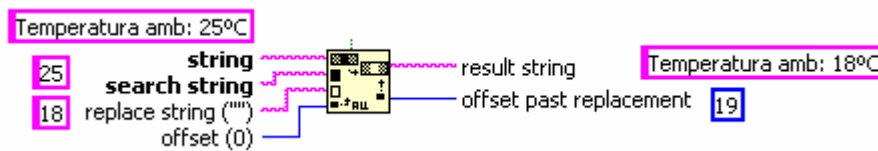


Figura 6.8. Función *Search and Replace String*.

Esta función busca una subcadena de una cadena y la reemplaza por otra subcadena establecida.

Match Pattern

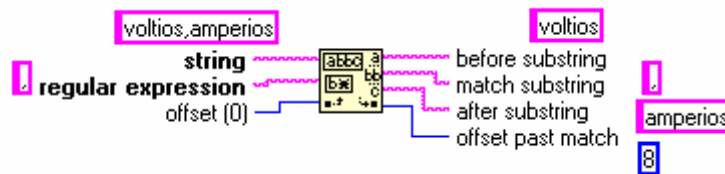


Figura 6.9. Función *Match Pattern*.

Esta función busca en la cadena patrones o subcadenas incluyendo expresiones tales como *, ?, ., +, \$, \ entre otras.

Format Date/Time String

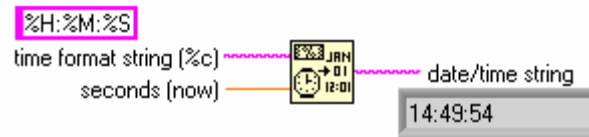


Figura 6.10. Función *Format Date/Time String*.

De acuerdo al formato que se le especifique captura la fecha y la hora del sistema.

Scan From String

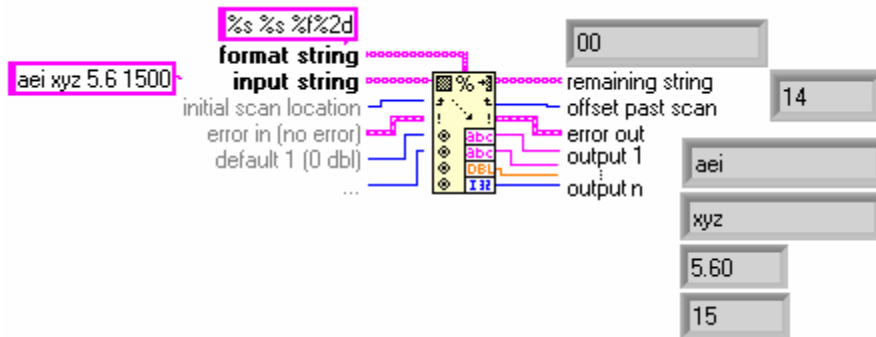


Figura 6.11. Función *Scan From String*.

Esta función rastrea la cadena de entrada buscando datos de acuerdo al formato definido en *format string*. Esta función es redimensionable, pues se puede obtener varias salidas de acuerdo a lo especificado en *format string*. Para definir el formato de conversión simplemente se hace clic derecho sobre el VI y se selecciona del menú desplegable la opción *Edit Format String*. La ventana que se despliega se muestra en la figura 6.12.

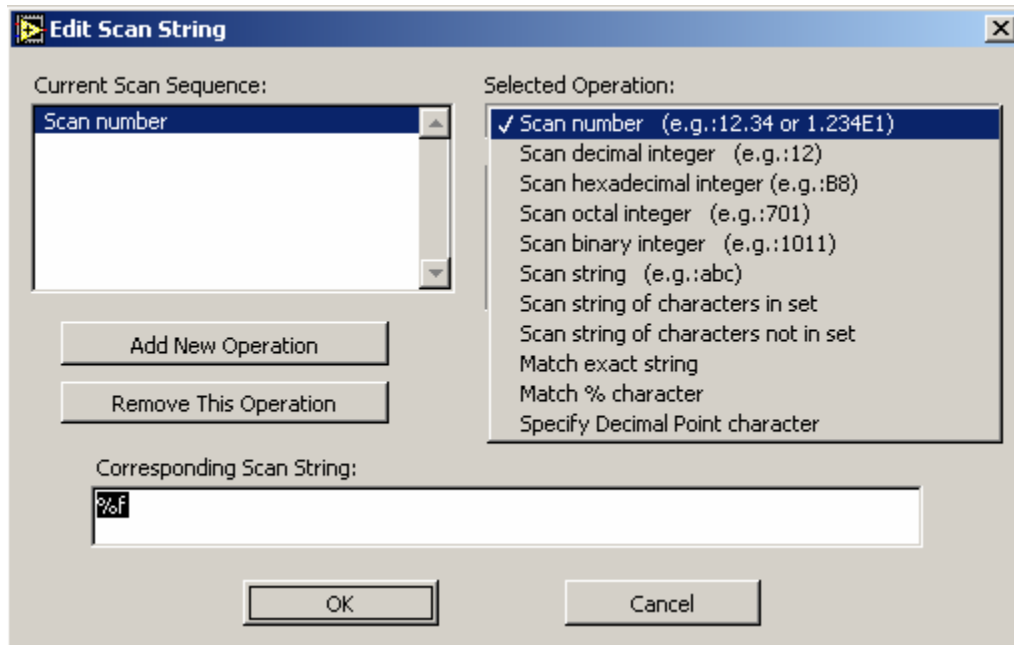


Figura 6.12. Cuadro de diálogo *Edit scan String*.

Allí se determina los tipos de datos que se manejarán posteriormente y el orden en que se ejecutarán.

Entre los formatos más utilizados se encuentran:

%s	Formato <i>string</i>
%f	Formato de punto flotante
%d	Formato decimal entero
%b	Formato binario

El formato de la conversión también puede ser definido manualmente cableando una variable *String* en el terminal *Format String*.

Format Into String

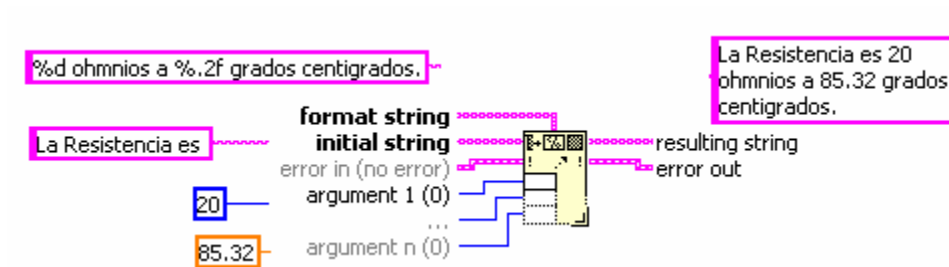


Figura 6.13. Función *Format Into String*.

Convierte los argumentos de entrada en una cadena, cuyo formato es determinado por *format string*.

Array to Spreadsheet String

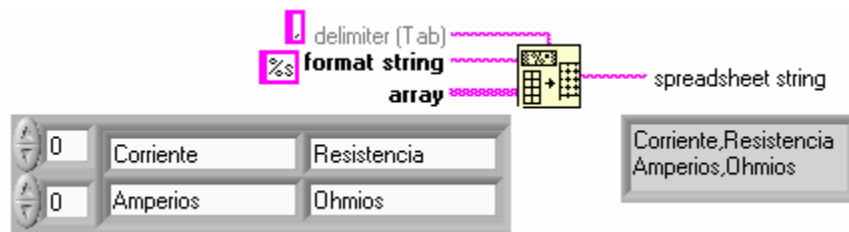


Figura 6.14. Función *Array to Spreadsheet String*.

Convierte un arreglo de cualquier dimensión a una cadena que delimita las columnas del arreglo por el caracter que se especifica en *delimiter*.

Constantes de cadena

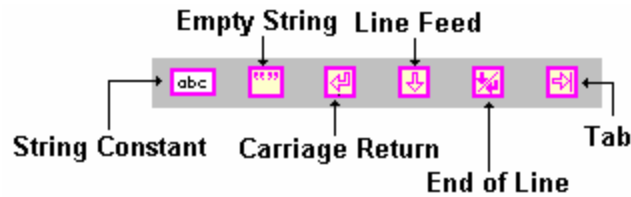


Figura 6.15. Caracteres especiales.

Se encuentran en la parte inferior del submenú *string* de la paleta de funciones. En su orden son:

String Constant: Constante tipo cadena.

Empty String: Consiste de una cadena que está vacía. Su longitud es cero.

Carriage Return: Consiste de una cadena constante que contiene el valor ASCII CR.

Line Feed: Consiste de una cadena constante que contiene el valor ASCII LF.

End of Line: Consiste de una cadena constante que contiene un fin de línea dependiendo de la plataforma utilizada.

Tab: Consiste de una cadena constante que contiene el valor ASCII correspondiente al tab horizontal.

Existe otra paleta que contiene las funciones de cadena adicionales como lo indica la figura 6.16.

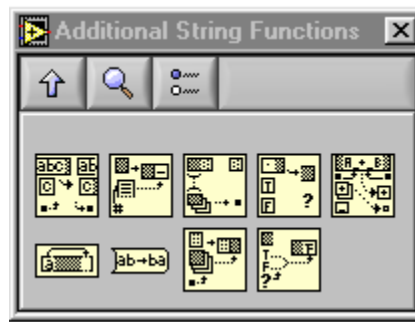


Figura 6.16. Funciones adicionales para cadenas.

Estas funciones son:

Search / Split String

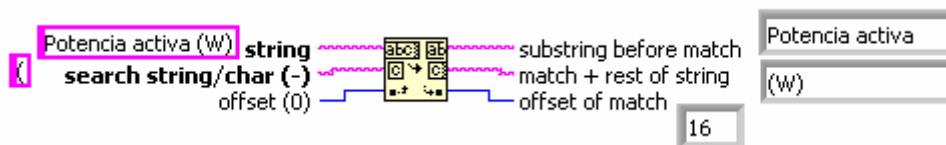


Figura 6.17. Función *Search / Split String*.

Esta función busca una subcadena o caracter en la cadena de entrada a partir de la posición *offset*. Retorna la subcadena anterior a la cadena o caracter buscado, la cadena o caracter buscado más el resto de la cadena de entrada y la posición en la cual se encontró.

Pick Line

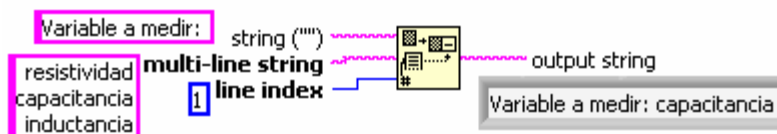


Figura 6.18. Función *Pick Line*.

Adiciona a una cadena existente una línea de una lista de subcadenas separadas por el retorno de carro, previamente definidas por el usuario en *multi-line string*. La entrada *line index* define cual subcadena se adiciona.

Match First String

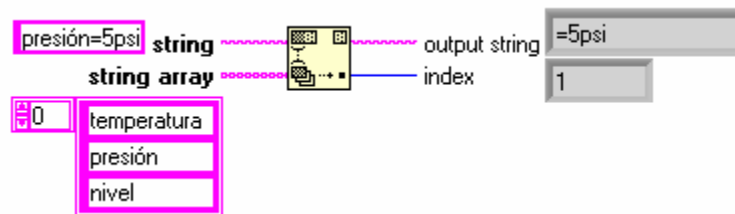


Figura 6.19. Función *Match First String*.

Esta función compara el principio de la cadena de entrada con cada uno de los elementos del arreglo cadena. Si encuentra coincidencia con alguno retorna la posición del arreglo en la cual lo encontró y como cadena de salida el resto de la cadena de entrada.

Match True / False String

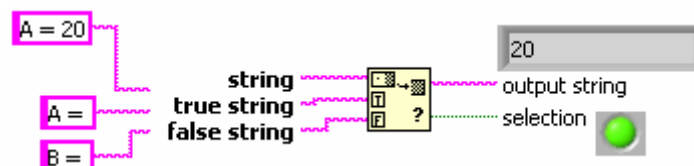


Figura 6.20. Función *Match True / False String*.

Compara el inicio de la cadena de entrada en *string* con las cadenas *true string* y *false string*. De acuerdo a la cadena que concuerde se entrega un valor booleano y la cadena de entrada sin el término similar.

Scan Strings for Tokens

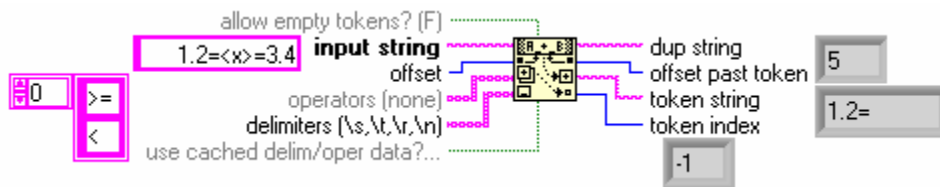


Figura 6.21. Función *Scan Strings for Tokens*.

Esta función busca *tokens* en la cadena de entrada. Un *token* puede ser un arreglo de cadenas con operadores (<, >, =) o una cadena delimitada por caracteres tales como (\s, \t, \r, \n).

Tiene como salidas la posición siguiente al *token* encontrado, la cadena *token* y el índice *token* que toma 2 valores: (-1 si *token string* no contiene valores *token* y -2 si ya ha terminado de recorrer la cadena de entrada y no encontró ningún *token*).

Index String Array

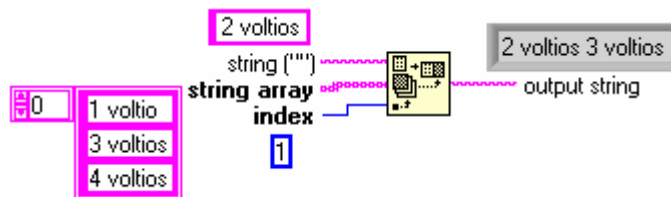


Figura 6.22. Función *Index String Array*.

Esta función adiciona a la cadena de entrada un elemento del arreglo de cadenas según lo especifique *index*.

Append True / False String

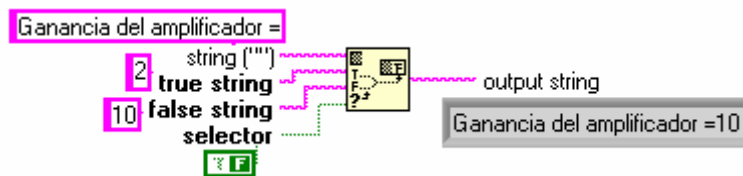


Figura 6.23. Función *Append True / False String*.

Adiciona a la cadena de entrada otra cadena de acuerdo al valor determinado en el selector booleano.

Rotate String



Figura 6.24. Función *Rotate String*.

Coloca el primer caracter de la cadena en la última posición.

Reverse String

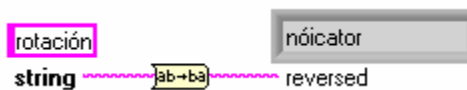


Figura 6.25. Función *Reverse String*.

Produce un reverso de orden de la cadena de entrada.

Otro gran número de funciones que presenta LabVIEW para la manipulación de cadenas son las de conversión. Con ellas se puede convertir cadenas decimales, hexadecimales, octales, fraccional/exponencial a representaciones numéricas. De igual forma los números hexadecimales, octales, decimales y con formato fraccional/exponencial se pueden convertir a cadenas.

En la figura 6.26 se presenta la paleta de conversiones.

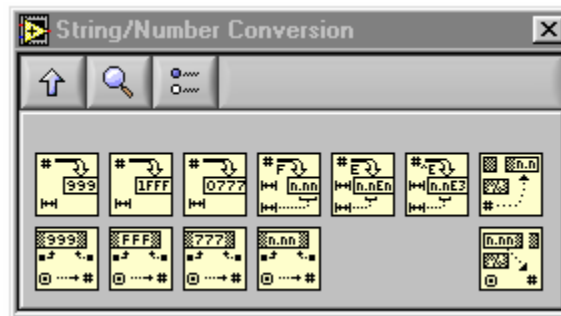


Figura 6.26. Funciones de conversión.

Algunas de estas funciones son polimórficas y sobrecargadas. Debido a ello están en condiciones de aceptar arreglos y clusters de números y convertirlos en arreglos y clusters de cadenas.

Otra de las herramientas utilizadas en el manejo de las cadenas son las conversiones string/array/path. Es así como se puede partir de una cadena o arreglo de cadenas y convertirlas en variables tipo *Path* o viceversa.

Estas funciones se encuentran en la paleta que se muestra en la figura 6.27.

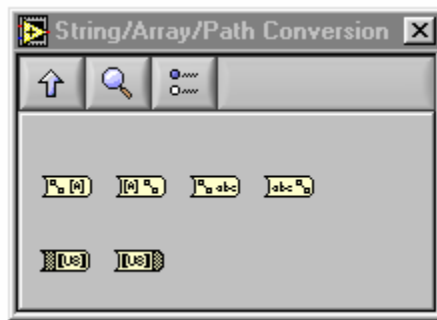


Figura 6.27. Paleta de conversiones string/array/path.

EJERCICIO 6.1 CONCATENACIÓN, CONVERSIÓN Y BÚSQUEDA

Se desea construir un VI que informe en un Mensaje Final el estado de una variable. El nombre de la variable y sus unidades están en forma de cadena, mientras el valor se encuentra de forma numérica. Se desea además conocer la longitud total del mensaje, la posición de dato en el mensaje y el mensaje al revés.

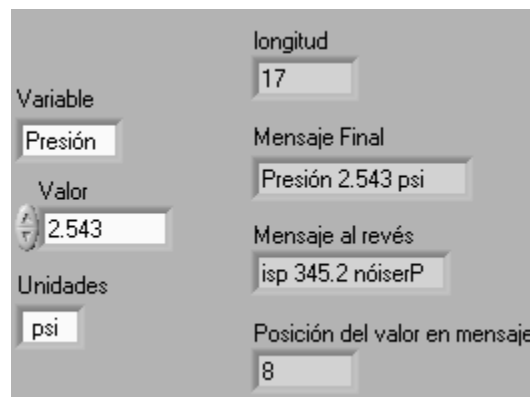


Figura 6.28. Panel frontal para el ejercicio 6.1.

La ventana de diagramación de la figura 6.29 muestra la solución al ejercicio 6.1.

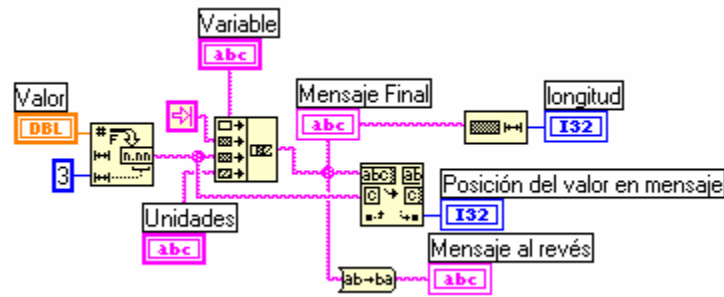


Figura 6.29. Ventana de diagramación para el ejercicio 6.1.

Para realizar la conversión número/cadena se utiliza la función *Number to Fractional String* a la cual se le define el número de dígitos de precisión a utilizar.

Una vez se tiene las 3 cadenas se concatenan utilizando la función *Concatenate String*.

Obsérvese que se ha utilizado un caracter tab en la concatenación de las 2 primeras cadenas, esto para separar con un espacio la cadena del valor numérico. También se hubiera conseguido con dejar un espacio al inicio de la segunda cadena o al final de la primera.

Cuando se tiene la cadena final se calcula su longitud con la función *String Length*, se obtiene su inversa y se encuentra la posición del valor numérico con las funciones *Search Split /String* y *reverse string*.

FIN EJERCICIO 6.1

EJERCICIO 6.2 DATOS DE UN MEDIDOR

Se desea presentar los datos de un medidor que puede leer tres variables (Corriente, Voltaje y Resistencia) seleccionando una a la vez.

El panel frontal del instrumento de medida se muestra en la figura 6.30.

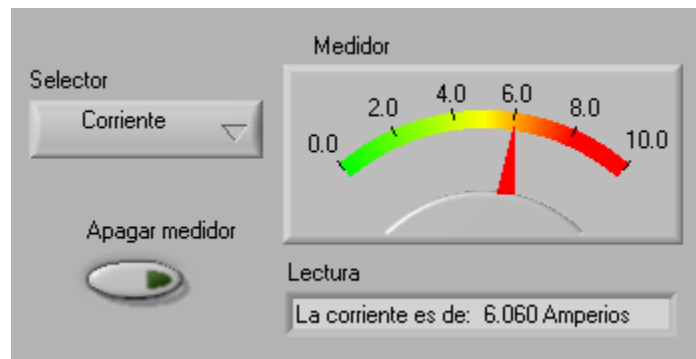


Figura 6.30. Panel frontal del ejercicio 6.2.

La figura 6.30 ilustra el medidor cuando se ha puesto en modo corriente.

Una forma para conseguir que la cadena "Lectura" varíe su contenido de acuerdo al menú "Selector" es construir una estructura *CASE* que entregue a la función *Format into String* los parámetros adecuados a cada caso.

En las figuras 6.31, 6.32 y 6.33 se muestra el diagrama para este ejercicio.

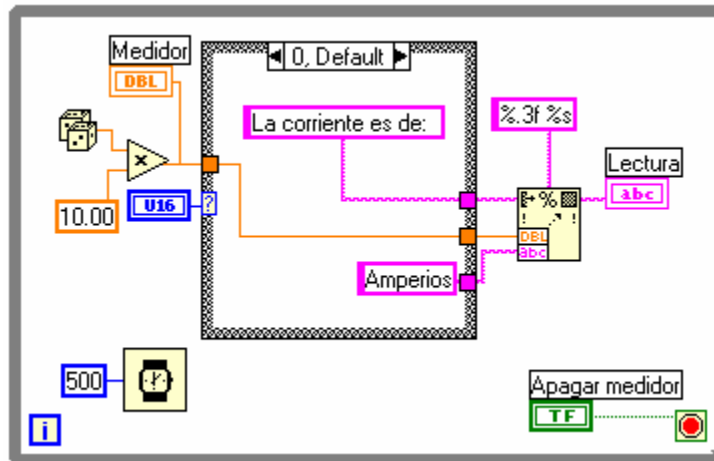


Figura 6.31. Diagrama del ejercicio 6.2. Caso 0.

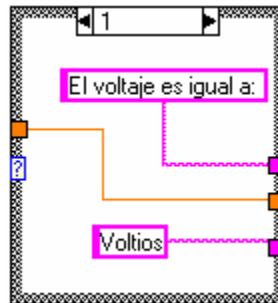


Figura 6.32. Diagrama del ejercicio 6.2. Caso 1.



Figura 6.33. Diagrama del ejercicio 6.2. Caso 2.

A la función *format into String* se le definen 4 entradas:

- Una cadena inicial: en la cual va el texto de acuerdo a la variable escogida. Por ejemplo: "La resistencia es de: ".
- El valor numérico generado aleatoriamente por el *Random Number*.
- Una cadena con las unidades correspondientes.
- El formato de las entradas. En este caso se define: formato fraccional con 3 dígitos decimales para el número y formato cadena para la segunda cadena de entrada.

El programa se ejecutará hasta que el usuario lo decida.

FIN EJERCICIO 6.2

6.3 ARCHIVOS

El manejo que se hace con archivos es esencialmente para realizar alguna de las siguientes operaciones:

Abrir y cerrar archivos.

Leer y escribir datos de un archivo.

Mover, copiar, borrar, renombrar archivos y directorios.

Las funciones que utiliza LabVIEW para estos fines se encuentran en el submenú *File I/O* de la paleta de funciones. Existe específicamente tres pasos relacionados con el manejo de un archivo:

- Crear o abrir un archivo. Aquí se especifica el nombre del archivo a crear o la ruta de un archivo existente.
- Leer o escribir en el archivo. De acuerdo a la operación que se esté realizando.
- Cerrar el archivo. Una vez se ha efectuado las operaciones el archivo debe cerrarse.

Una clasificación de los VIs que realizan estas operaciones se muestra en la figura 6.34.

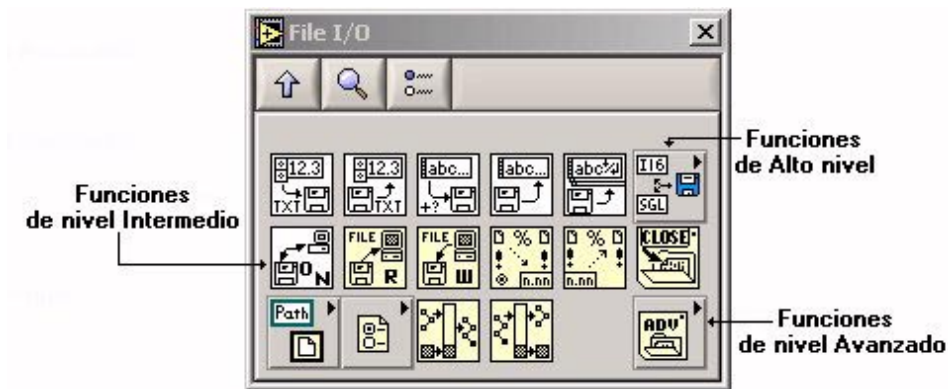


Figura 6.34. Funciones para manejo de archivos.

Esta división puede explicarse como se muestra en la tabla 6.1.

Tabla 6.1. Funciones para manejo de archivos.

Nivel	Funciones	Característica
Alto	Funciones básicas para archivos ASCII y binarios.	Utilizan como subVIs las funciones del nivel intermedio. El manejo de funciones de bajo nivel es transparente al usuario.
Intermedio	Funciones para abrir, crear, leer, escribir y cerrar archivos.	Pueden realizar todas las tareas de lectura y escritura de archivos desde LabVIEW.
Avanzado	Funciones para mover, copiar, borrar y listar archivos, entre otras.	Permiten operar sobre la estructura de directorios del sistema.

La figura 6.35 muestra la paleta de funciones en la que se encuentran las constantes para el manejo de archivos.

Entre estas constantes se encuentra la constante tipo *PATH* la cual permite identificar la ubicación de un archivo en disco.



Figura 6.35. Constantes de archivo.

Dependiendo del sistema operativo, los *Path* deben tomar la forma adecuada.

Windows: `c:\windows\escritorio\archivo.txt`

MacOS: `hard disk:\mi carpeta:archivo.txt`

Unix: `/home/users/usuario/archivo.txt`

Figura 6.36. Rutas en sistemas operativos.

Otras constantes están encargadas de enviar una ruta vacía, mostrar el directorio por defecto, crear directorios temporales, entre otras.

Ya que todos los procesos de manipulación de archivos se obtienen con las funciones de nivel intermedio, éstas serán mostradas a continuación.

Open/Create/Replace Files

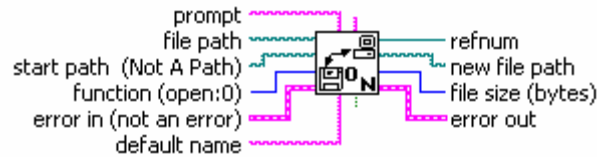


Figura 6.37. Función *Open/Create/Replace Files*.

Este VI abre, crea o reemplaza un archivo desde la ruta indicada en *file path*. Si lo encuentra entrega un valor de referencia en *refnum* que LabVIEW utilizará para identificar todas las tareas que se realicen sobre ese mismo archivo. Si a este VI no se le especifica la ruta del archivo, entonces en el momento de la ejecución el VI abrirá un cuadro de diálogo desde el cual se puede elegir el archivo.

Read File

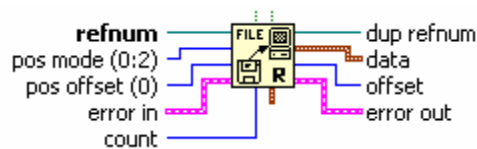


Figura 6.38. Función *Read File*.

Este VI recibe el *refnum* del VI anterior y lee el número de *bytes* que se le han especificado en *count* desde la posición definida por el *pos mode* y el *pos offset*, así:

Se puede leer desde el principio del archivo si *pos mode* esta en 0.

Con respecto al final del archivo si *pos mode* esta en 1.

Con respecto a la ubicación actual si *pos mode* esta en 2.

Si el *pos offset* es cableado, por defecto el *pos mode* valdrá 0, de lo contrario *pos mode* estará en 2.

Write File

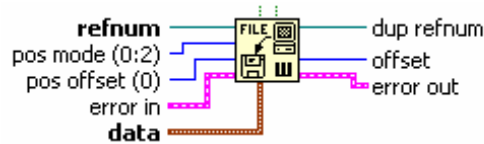


Figura 6.39. Función *Write File*.

Este VI permite escribir en el archivo referenciado previamente por *refnum*. Se comporta en forma idéntica con el *pos mode* y *pos offset* del VI anterior.

Close File



Figura 6.40. Función *Close File*.

Cierra el archivo especificado por *refnum*.

Scan from File

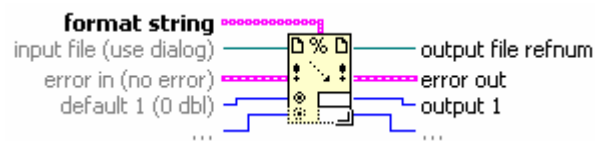


Figura 6.41. Función *Scan From File*.

Interpreta el archivo de acuerdo al formato definido en *format string* realizando las conversiones indicadas. Se puede tener más de una salida de acuerdo al número de conversiones que se especifiquen en el formato.

Format into File

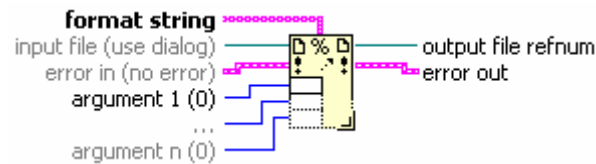


Figura 6.42. Función *Format into File*.

Convierte los argumentos de entrada de acuerdo al formato predefinido en *format string* y los adiciona al archivo de entrada.

En el manejo de los *path* o rutas se encuentran las funciones:

Build Path



Figura 6.43. Función *Build Path*.

Adiciona a una ruta existente un nombre o ruta relativa.

Strip Path



Figura 6.44. Función *Strip Path*.

Retorna el nombre del último componente de la ruta y la ruta que lo conduce.

6.3.1 Escribir Datos en un Archivo

La figura 6.45 muestra el diagrama general utilizado para escribir datos a un archivo nuevo o uno ya existente.

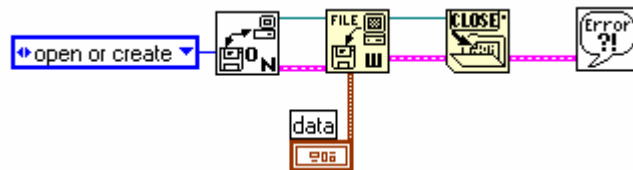


Figura 6.45. Secuencia utilizada al escribir datos a un archivo.

EJERCICIO 6.3 GUARDAR DATOS EN UN ARCHIVO ASCII

Generar un grupo de datos de temperatura y almacenarlos en un archivo.

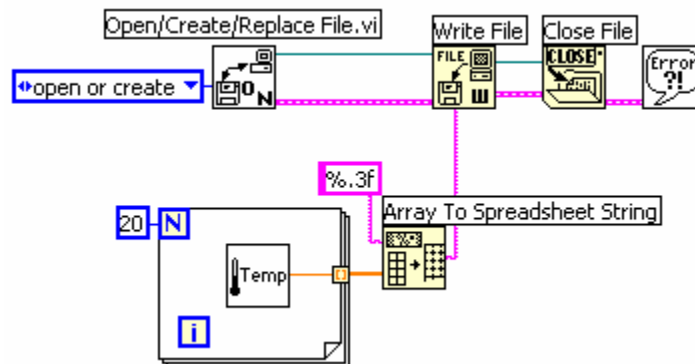


Figura 6.46. Ventana de diagramación.

Al ejecutar el anterior código, lo primero que se le presenta al usuario es un cuadro de diálogo donde se debe seleccionar el archivo en el cual se va a escribir o digitar el nombre del archivo si este no existe. En caso de querer especificar la ruta y el nombre del archivo se debe cablear al terminal de entrada *file path* una variable con estos datos.

Una vez se ha ejecutado el ciclo FOR se dispone de un arreglo de datos de temperatura arrojados por el *Digital Thermometer.vi*.

Posteriormente se realiza la conversión del arreglo a una cadena tipo *spreadsheet*. En este tipo de cadena los datos están tabulados. Por defecto el caracter delimitador es *tab*.

Otro de los parámetros de entrada que debe definírsele a este VI es el formato de los datos. Es así como se ha definido en este ejercicio en particular el formato flotante de 3 decimales. La cadena de datos se escribe al archivo mediante el *Write File.vi* y posteriormente se cierra el archivo con el *Close File.vi*.

FIN EJERCICIO 6.3

Los VIs utilizados para el manejo de archivos, generalmente poseen como salidas terminales de *refnum* y de error, los cuales además de utilizarse para el control de las tareas y para obtener información de procesos errados, proveen flujo de datos del programa en la forma esperada (abrir o crear un archivo, realizar operaciones sobre él y posteriormente cerrarlo). La figura 6.47 muestra un modelo alternativo a la figura 6.46.

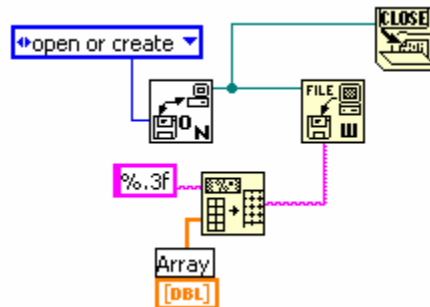


Figura 6.47. Error en flujo de datos.

En el anterior diagrama después de crear o abrir el archivo, no siempre se podrá asegurar cual operación se hará primero: si la de escribir en el archivo o la de cerrarlo, debido a que no existe un flujo de datos definido. Es posible incluso que se de el caso en que se cierre el archivo antes de poder escribir en él. Para establecer esto se acostumbra cablear el error de salida de un VI al error de entrada del siguiente como se indicó en la figura 6.45 y hacer uso de los terminales de *refnum* a las salidas.

EJERCICIO 6.4 CONSTRUIR UN ARCHIVO TIPO *TAB DELIMITER*

Los archivos *tab delimiter* son aquellos en los cuales un arreglo 2D utiliza caracteres especiales para la separación entre columnas y filas.

Este ejercicio busca crear un archivo de este tipo que contenga 2 columnas. Allí se almacenarán los datos arrojados por el *Digital Thermometer.vi* en una columna y en la otra columna el número correspondiente al dato generado. Se debe guardar el archivo con extensión .txt y abrirlo posteriormente en un procesador de texto u hoja de cálculo.

En el siguiente diagrama se observa el proceso.

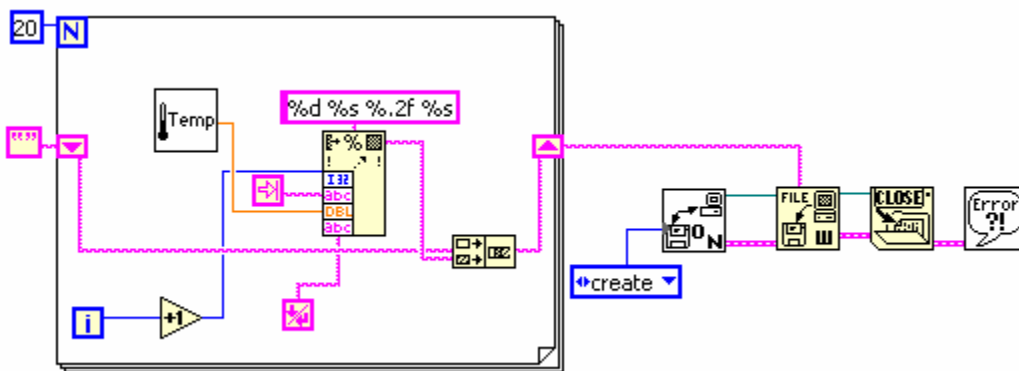


Figura 6.48. Ventana de diagramación.

A medida que se va generando cada uno de los 20 datos de temperatura se va construyendo una sola cadena con la ayuda del *Format into String* que solicita los formatos de cada uno de los datos de entrada. En este caso el formato especificado se muestra en la figura 6.49.

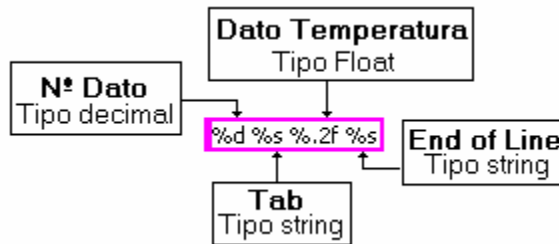


Figura 6.49. Formato de los datos.

Es muy importante recordar que cuando se trabaja con shift registers deben inicializarse con un valor de acuerdo al tipo de dato que se le cablea. En este ejercicio se ha utilizado una cadena vacía para inicializar el shift register.

Una vez se ha generado los datos se crea el archivo, se escribe los datos allí y luego se da por terminado el proceso con el Close File.VI.

FIN EJERCICIO 6.4

6.3.2 Leer un archivo

El procedimiento utilizado para recuperar la información de un archivo es similar al utilizado en escritura, se diferencian en la función *Read File.vi*, a la cual debe cablearse el número de *bytes* que van a ser leídos en el terminal de entrada *count*. La figura 6.50 ilustra el procedimiento mencionado.

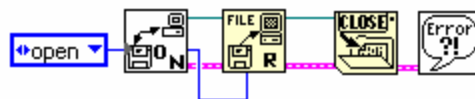


Figura 6.50. Leer de un archivo.

EJERCICIO 6.5 RECUPERACION DE DATOS DE UN ARCHIVO ASCII

Se busca recuperar el archivo creado en el ejercicio 6.4. La figura 6.51 detalla los pasos a seguir en la recuperación de la información.

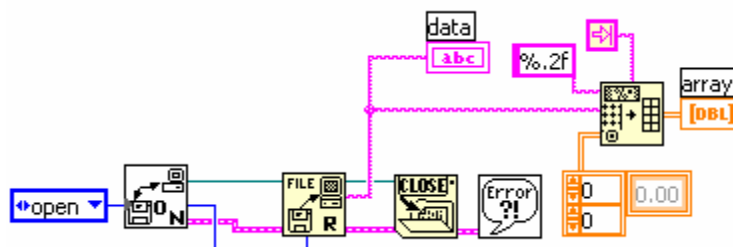


Figura 6.51. Diagrama del ejercicio 6.5.

Una vez se ha recuperado la información en formato cadena se convierte a formato arreglo para poder manipular los datos adecuadamente.

Observe que a la función *Spreadsheet to array* debe cablearse una constante con el tipo de datos a recuperar, por esta razón se creó una constante arreglo 2D con formato DBL.

FIN EJERCICIO 6.5

EJERCICIO 6.6 ESCRIBIR UN ARCHIVO BINARIO

Se creará un archivo binario con los datos generados por el usuario al interactuar con un control numérico tipo dial.

Nota: Se debe realizar una validación para que datos similares contiguos no sea guardados.

En la figura 6.52 se muestra los elementos utilizados en el panel frontal.

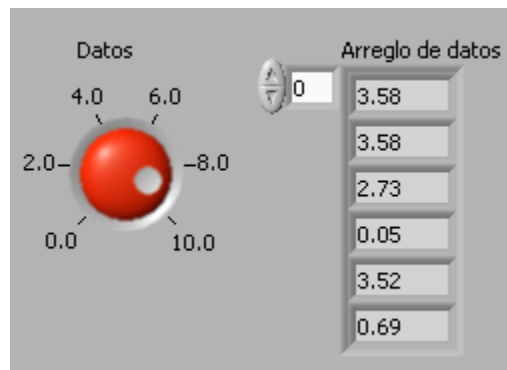


Figura 6.52. Panel frontal del ejercicio 6.6.

Se ha utilizado un indicador tipo arreglo para visualizar los datos que se han generado.

La ventana de diagramación de la figura 6.53 ilustra el proceso de generación y almacenamiento de datos.

El número de datos que genera el usuario se ha limitado como lo indica la estructura *FOR*.

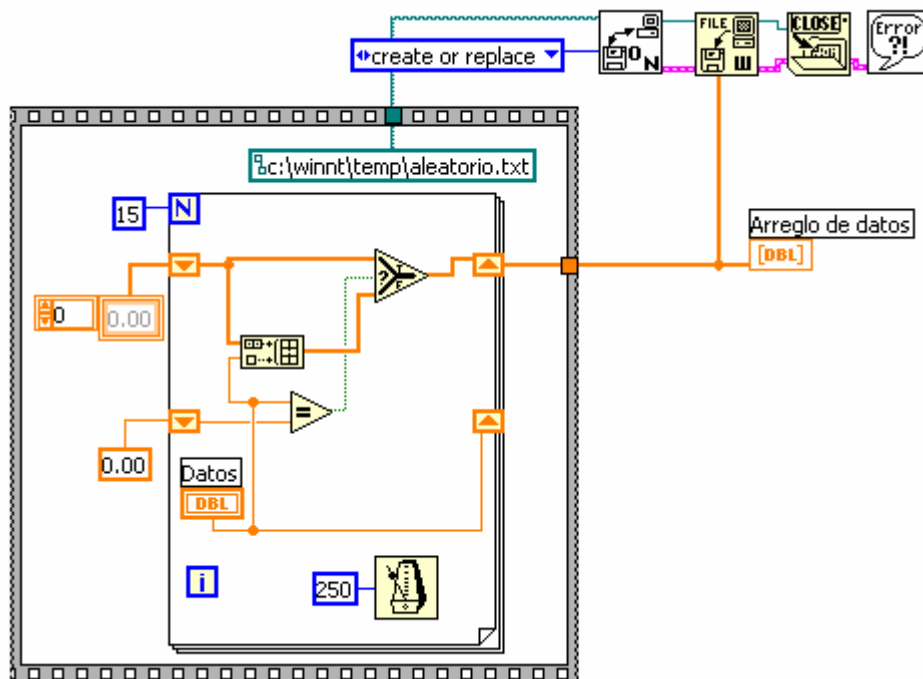


Figura 6.53. Ventana de diagramación del ejercicio 6.6.

A medida que se va generando cada dato se realiza la comparación con el dato inmediatamente anterior para evitar la duplicidad. Si el resultado de la comparación resulta negativo (valores diferentes) se agrega un nuevo dato al arreglo que se va creando, de lo contrario (valores iguales) no se adiciona el dato al arreglo.

Obsérvese que se ha utilizado una estructura *SEQUENCE* para contener el código anterior, además del *path* del *Open/Create/Replace File.vi*, esto con el fin de permitir que se genere los datos primero antes de escribirlos en el archivo. El usuario genera los datos y posteriormente el programa se encarga de almacenarlos en la ruta especificada.

FIN EJERCICIO 6.6

EJERCICIO 6.7 RECUPERACIÓN DE DATOS DE UN ARCHIVO BINARIO

Se pretende recuperar el archivo guardado en el ejercicio 6.6.

El diagrama que soluciona el ejercicio 6.7 se muestra en la figura 6.54.

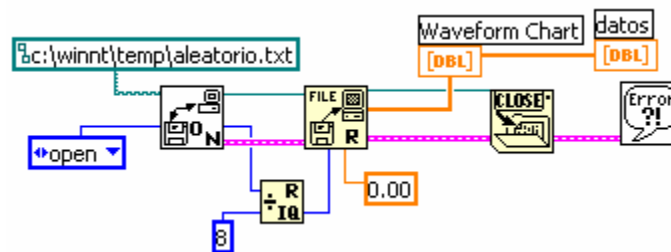


Figura 6.54. Ventana de diagramación del ejercicio 6.7.

Cuando se lee archivos binarios es necesario especificar el tipo y el número de datos a leer. El tipo de dato se puede especificar cableando una constante al terminal *byte stream type* del mismo tipo del dato. Por esta razón es muy importante contar con la suficiente información acerca de la naturaleza de los datos cuando se manipulan archivos binarios para poderlos recuperar satisfactoriamente.

Debido a que la representación DBL utiliza 8 *bytes*, es necesario especificarle al VI utilizado para lectura cuantos datos se recogerán. Por esta razón el número total de *bytes* del archivo abierto se divide por 8 para entregar al VI de lectura el número total de datos que debe leer.

Posteriormente se grafican los datos recuperados y se cierra el archivo.

FIN EJERCICIO 6.7

6.4 FUNCIONES DE ALTO NIVEL PARA EL MANEJO DE ARCHIVOS

LabVIEW posee funciones de alto nivel para realizar tareas sencillas de lectura y escritura de archivos tipo ASCII o binarios de forma directa.

El primer renglón de VIs de la figura 6.55 contiene las funciones mencionadas para archivos ASCII en formato *tab delimiter* o en formato de cadena y un submenú para manejar arreglos I16 (enteros a 16 bits) o SGL (precisión simple de punto flotante) en archivos binarios.

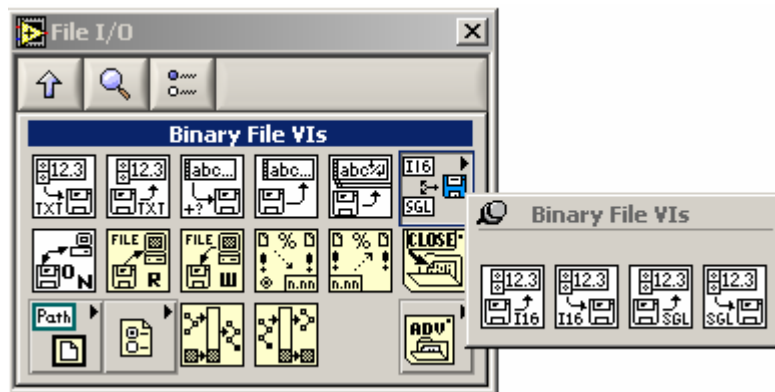


Figura 6.55. Funciones de archivos binarios.

Debido a que los archivos binarios son generalmente no portables a otras aplicaciones, es necesario detallar adecuadamente los tipos de datos utilizados cuando se desee recuperar el archivo.

Entre las funciones de la primera fila podemos destacar las utilizadas para manejo de *spreadsheet* tanto en lectura como en escritura, adicionar o recuperar caracteres y/o líneas.

En la tercera fila aparecen las funciones avanzadas, las cuales son utilizadas para operar sobre la estructura de directorios. En la figura 6.56 se presenta la paleta de este tipo de funciones.

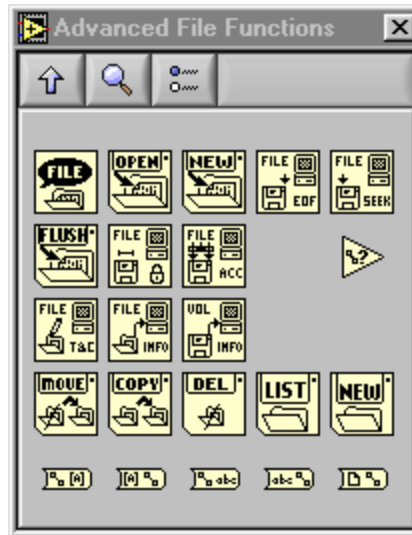


Figura 6.56. Funciones avanzadas de archivos.

Las características de estas funciones abarcan: obtención de información del archivo tal como permisos, tamaño, últimas modificaciones, mover archivos o directorios, borrarlos, listar archivos, entre otros.

Estos VIs pueden ser utilizados intuitivamente o con la ayuda presentada por LabVIEW para cada uno de sus VIs.

6.5 EJERCICIOS PROPUESTOS

1. Crear un archivo tipo *spreadsheet* de cuatro columnas con la siguiente información:

Valor X	Seno(x)	Cos(x)	Tan(x)

El valor de x debe ser el equivalente en radianes a cada grado desde 0 hasta 360. Los valores de seno, coseno y tangente deben ser guardados con 6 cifras decimales.

2. Recuperar sólo la información de la columna No.3 (Cos(x)) del ejercicio anterior y graficar los datos.

3. Crear un programa donde el usuario pueda decidir guardar los datos recuperados en el ejercicio 2 como archivo texto o archivo binario de acuerdo a la opción que seleccione de un menú.

4. Generar un archivo que contenga los valores de temperatura superiores a un valor establecido, con la fecha y hora en que se generaron. (**Nota:** Utilizar *Digital Thermometer.vi* y *Format Date/Time String*).

5. Recuperar el archivo del ejercicio anterior y representar los datos gráficamente.

BIBLIOGRAFÍA

1. LabVIEW User Manual, National Instruments Corporation, edición de Noviembre de 2001.
2. LabVIEW Measurement Manual, National Instruments Corporation, edición de Julio de 2000.
3. LabVIEW Development Guidelines, National Instruments Corporation, edición de Julio de 2000.
4. LabVIEW Tutorial, National Instruments Corporation, edición de Noviembre de 2001.
5. Getting Started with LabVIEW, National Instruments Corporation, edición de Noviembre de 2001.
6. LabVIEW Help, National Instruments Corporation, edición de Noviembre de 2001.
7. Analysis Concepts, National Instruments Corporation, edición de Julio de 2000.