

Memoria de Ampliación de Física

29 de Enero

2008

Memorias de las prácticas realizadas en el Laboratorio durante el
Curso 2007/08 de la asignatura Ampliación de Física.

Con Xilinx,
HPVVEE,
TestPoint y
LabView

*Rafael Vargas Sánchez (31.729.314-D),
Alfredo Michán Amado (32.068.875-K).*

*Curso 2007/08. Ampliación de Física. Dpto. Física Aplicada 1.
Universidad de Sevilla.*

Página intencionalmente en blanco.

Índice

Introducción	5
Material necesario.	7
Práctica 1: XLinx	9
Práctica 2: HPVEE	19
Práctica 3: TestPoint.....	35
Práctica 4: LabView	51
Conclusiones	69
Anexo A: Montaje de tarjeta de A/D.....	71
Anexo B: Comandos del K2700	75
Bibliografía	77
Contenido del CD	78

Introducción

A lo largo de esta memoria, vamos a realizar una serie de ejercicios propuestos con distintos entornos de desarrollo de aplicaciones para la adquisición de datos. Los programas que vamos a utilizar serán XLinx, HPVEE en su versión 5, TestPoint y LabView, ambos en su versión 6.

Dado que los comandos enviados a la unidad se repiten en todos los entornos, hemos escrito un anexo donde se muestran todas sus descripciones.

Todos estos ejercicios se han realizado durante las clases prácticas de la asignatura Ampliación de Física, impartida por el Dpto. de Física Aplicada 1 de la Universidad de Sevilla.

Material necesario

Para poder cumplimentar las prácticas descritas será necesario satisfacer los siguientes requisitos:

Materiales

- K2700.
- Tarjeta de adquisición de datos K7700.
- Fotorresistencia (LDR).
- PTC.
- NTC.
- PT100.
- Termopar tipo T ó K.

Software

- Keithley 2700 XLinx.
- HPVVEE 5.0
- TestPoint 6.0.
- National Instruments LabView 6i.

Práctica 1: Xlinx

Introducción

En esta práctica daremos los pasos para la correcta configuración del programa Xlinx.

Xlinx es un programa diseñado para mostrar de forma rápida y fácil graficas de los datos tomados por el K2700, tras unas sencillas configuraciones y sin necesidad de programar.

Objetivos

En esta práctica configuraremos el Xlinx para leer los sensores conectados a los canales del K2700. En el **Anexo A** se especifican que sensores utilizaremos y cómo se han de conectar.

También mostraremos la forma de crear gráficas de un dispositivo o de varios dispositivos (simultáneamente).

Por último guardaremos los datos mostrados en las graficas en distintos tipos de archivos (texto plano u hoja de cálculo en formato Microsoft Excel).

Práctica

Configuración de los canales

En este apartado explicaremos como configurar el XLinx para leer datos de los distintos sensores utilizados.

Para ello configuraremos el tipo de sensor conectado a cada canal del K2700 (Ilustración 1).

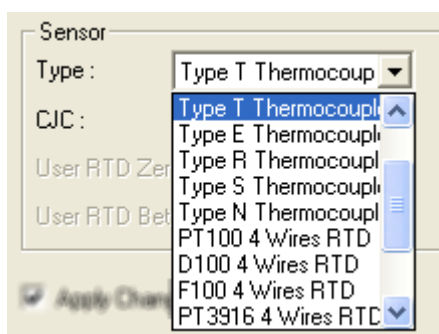


Ilustración 1: Tipo de sensores disponibles en el XLinx

Canal 101 LDR

La LDR es un sensor que manifiesta la alteración de la magnitud física (intensidad de la luz), variando su resistencia.

El K2700 no dispone de la función de equivalencia de Ohmios (Ω) a Lux. Por lo tanto, configuraremos este canal para mostrar el valor de la resistencia.

El tipo de sensor será pues un aparato de medida de resistencia con 2 vías.

Canal 102 PTC

La PTC aumenta su resistencia al sufrir un aumento de temperatura.

El K2700 tampoco dispone de la función de equivalencia en este caso, pero la función del termopar tipo K aproxima bastante el valor.

Por lo tanto configuraremos este canal como un termopar de tipo K.

Canal 103 NTC

La NTC también reacciona a los aumentos de temperatura con un cambio de resistencia, en este caso con una disminución de la misma.

En este caso el K2700 si dispone de la función de equivalencia.

La configuración que se le debe de aplicar es la de termistor de tipo 1950.

Canal 104 TC

El termopar traduce aumentos de temperatura en voltaje.

El K2700 dispone de la función de equivalencia de todos los tipos de termopares. Sin embargo, para una medición correcta debemos ajustar el sensor para que utilice el termómetro interno del aparato (Ilustración 2).

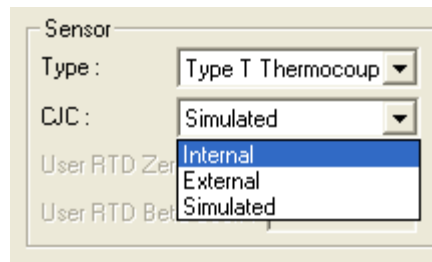


Ilustración 2: distintos tipos de unión fría (cold junction)

Entonces la configuración del canal será termopar tipo T y la CJC será interna (Ilustración 3).

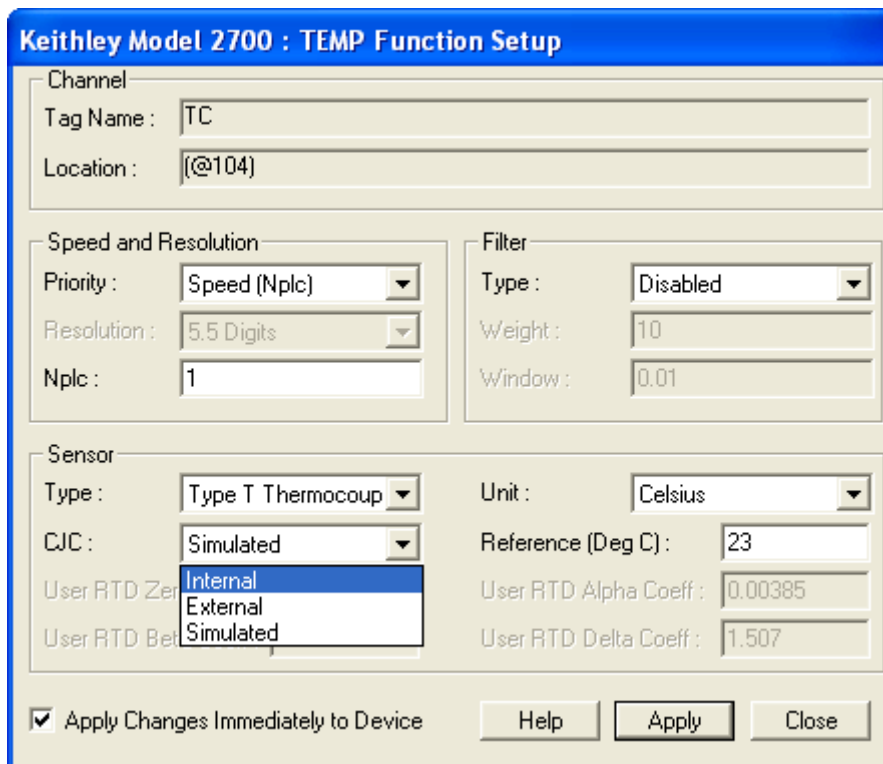


Ilustración 3: configuración del termopar tipo T

Canales 105 (y 115) PT 100

La PT100 es un sensor de temperatura por variaciones de resistencia, pero a diferencia de la PTC y la NTC utiliza 4 vías para funcionar.

El K2700 dispone de la función de equivalencia y al configurarlo como una PT100 sabrá automáticamente que el dispositivo es de 4 vías.

La configuración de este canal es una PT100 de 4 vías (Ilustración 4). No se requiere especificar más, pues por defecto se mide en grados Celsius.

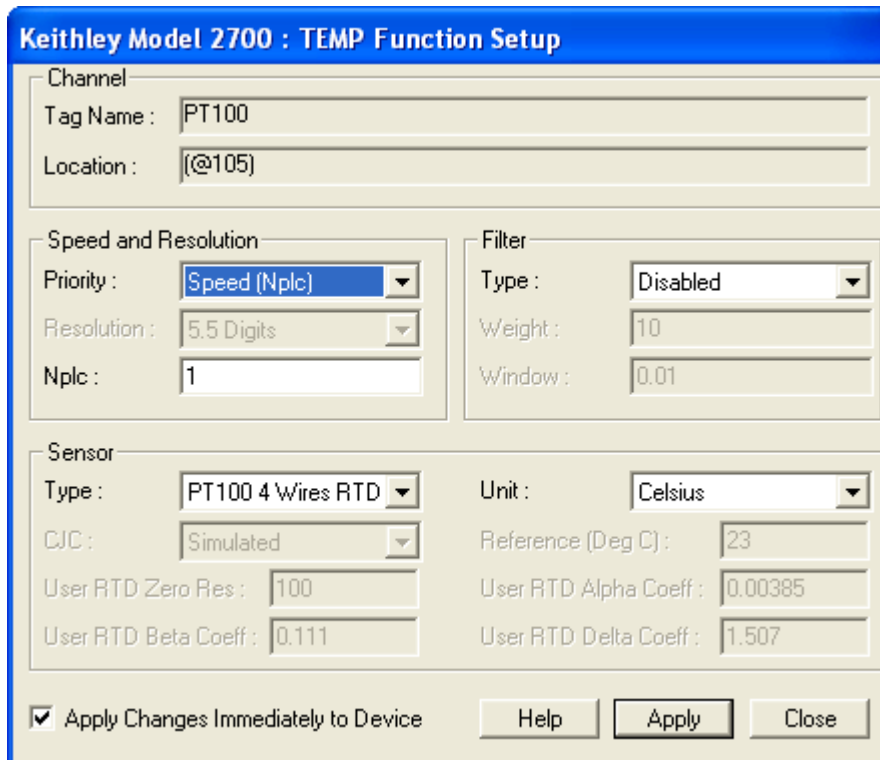


Ilustración 4: configuración de la PT100

Graficas y Exportación de datos

El Xlinx es un programa para realizar una serie de medidas rápidamente y mostrarlas en una gráfica, gracias a su fácil configuración.

Single Channel View

Para mostrar una gráfica, una vez configurado, basta con señalar el sensor que queremos medir. Por ejemplo: la LDR y pulsar el botón **Start** (Ilustración 5 e Ilustración 6).

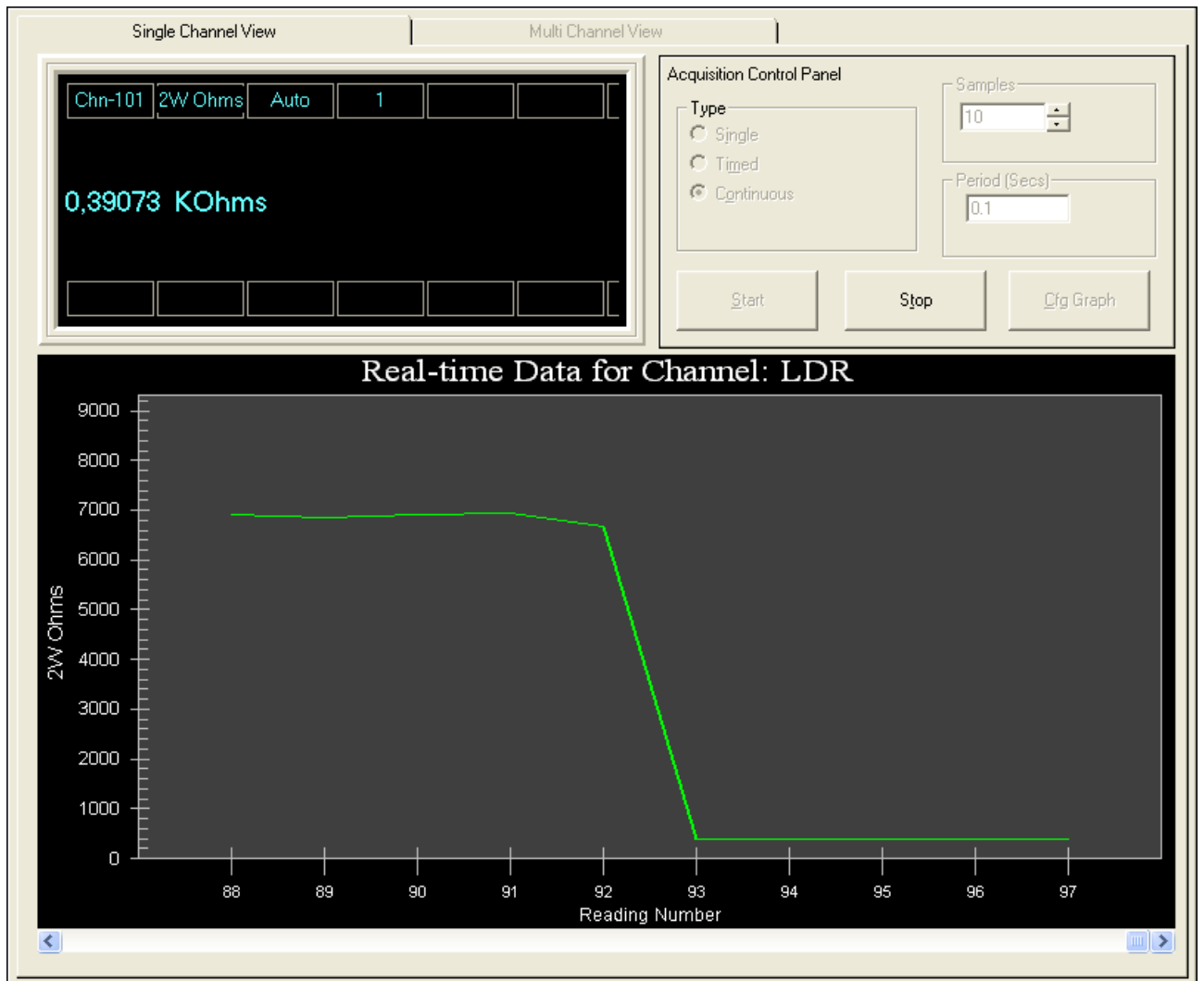


Ilustración 5: ejemplo de medida de un solo canal

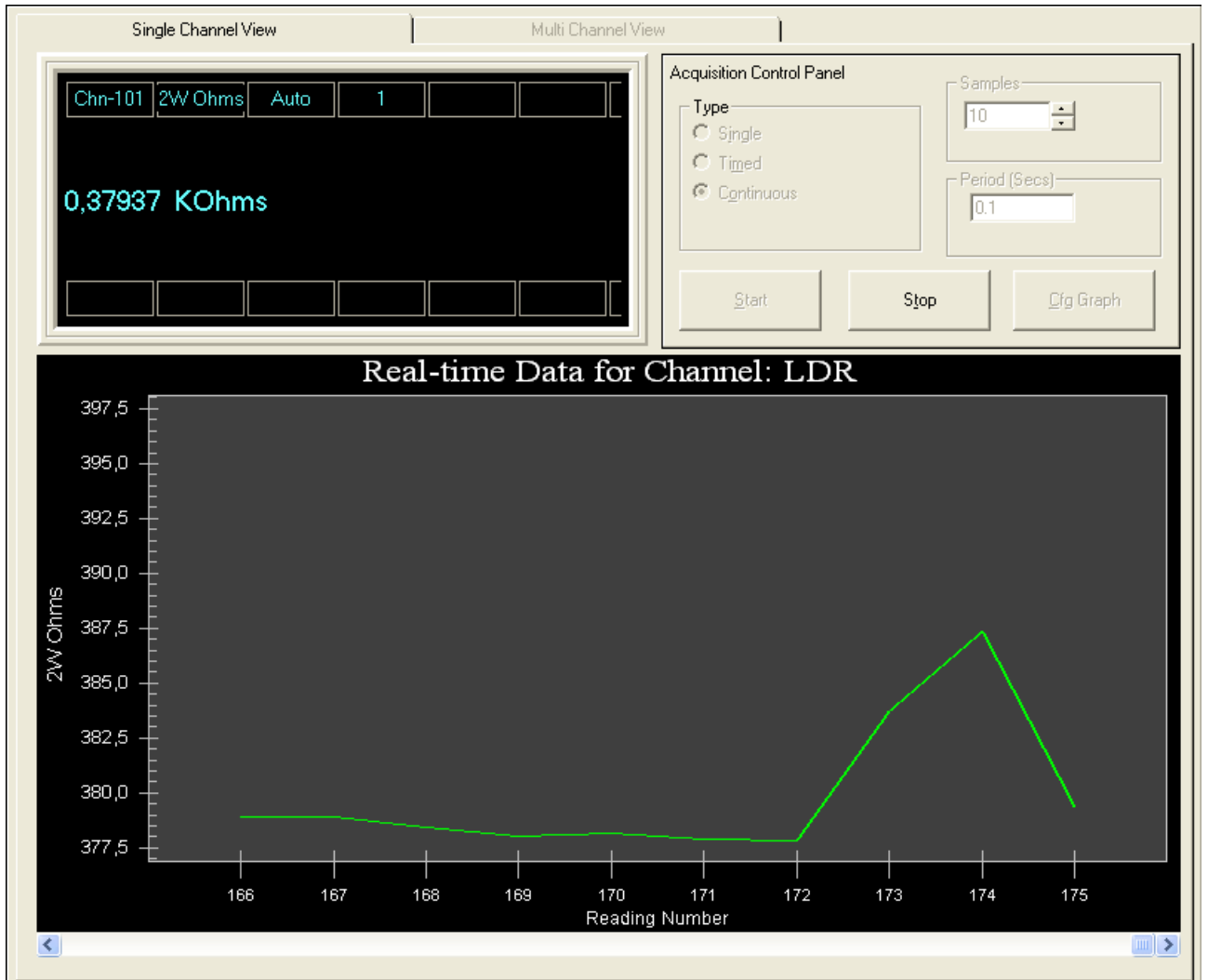


Ilustración 6: ejemplo de medida de un solo canal.

Existen tres modos de tomar datos en el Xlinx (Ilustración 7):

- **Single:** Toma un número exacto de medidas.
- **Timed:** Toma medidas durante el tiempo especificado.
- **Continuous:** Toma medidas hasta ser parado.

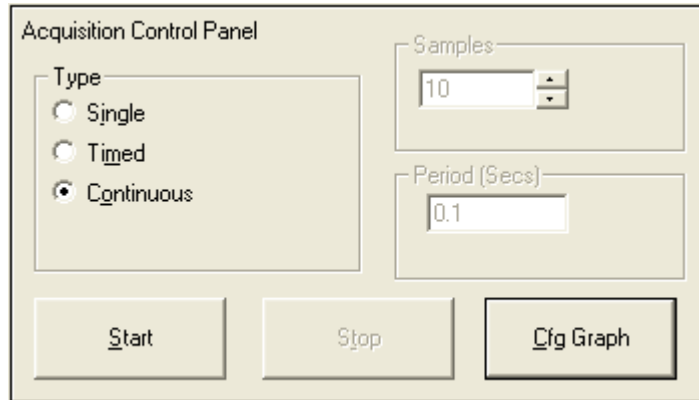


Ilustración 7: configuración de la medida

Las gráficas producidas se pueden configurar en el botón **Cfg Graph**. En la ventana que aparece se puede modificar el número de medidas que muestre, activar o desactivar la auto-escala del eje y restringir los valores mostrados en la Ilustración 8.

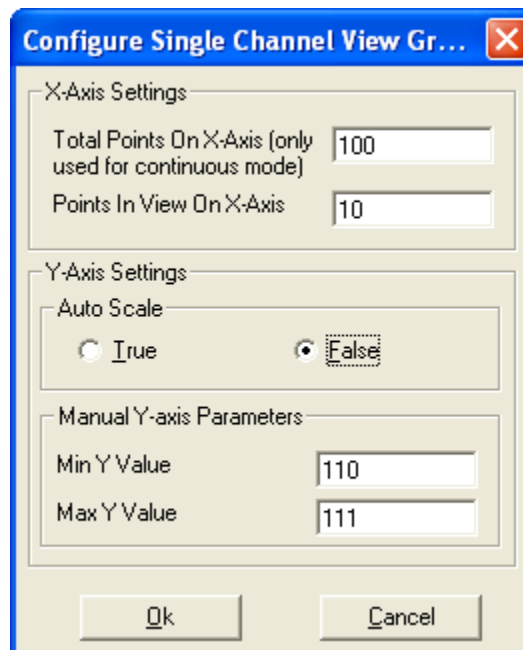


Ilustración 8: configuración de la gráfica

Multi Channel View

El Xilinx también puede realizar graficas de dos sensores enfrentados e incluso mostrar las graficas de 4 sensores a la vez (Ilustración 9).

Si queremos mostrar graficas de más de un dispositivo, sólo hay que pulsar la pestaña **Multi Channel View**.

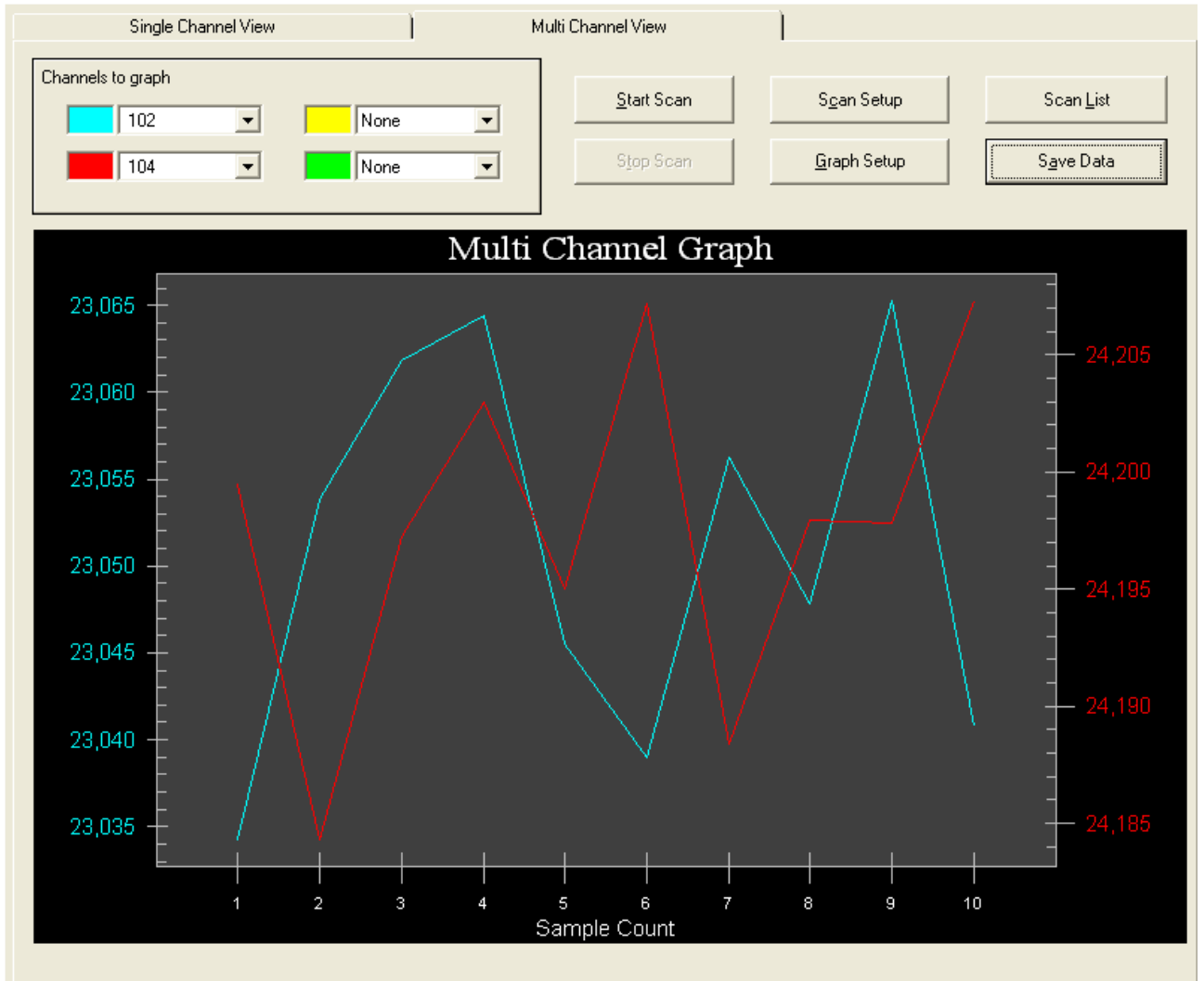


Ilustración 9: múltiples sensores simultáneamente

En la siguiente pantalla seleccionamos los canales que queremos medir.



También podemos cambiar la configuración en el botón **Scan Setup**

En la pantalla de configuración (Ilustración 10) podemos cambiar el numero de medidas, el tiempo entre las medidas de los distintos dispositivos y cada cuanto se actualiza la grafica.

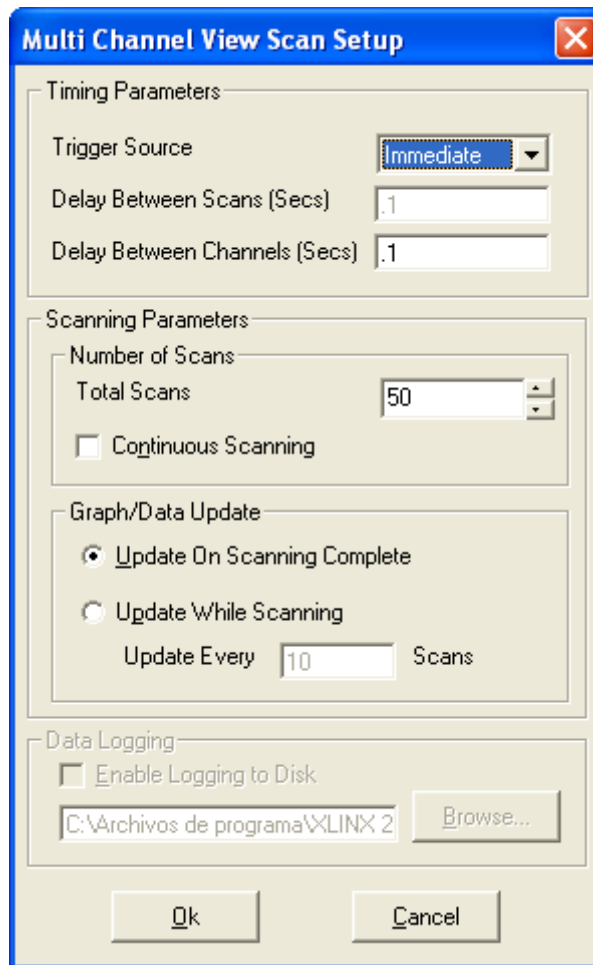


Ilustración 10: configuración de Multi Channel View

Una vez realizadas las graficas los datos de estas pueden ser guardadas pulsando el botón **Save Data**. Estos datos pueden ser guardados en texto plano o en una hoja de cálculo Microsoft Excel (*.xls).

Práctica 2: HP VEE

Índice

Introducción.....	19
Objetivos	19
Práctica.....	20
Programa 1: Medidas de un sensor.....	20
Programa 2: Medidas de varios sensores	25
Programa 3: Fichero.....	26
Programa 4: Gráficas.....	29
Programa 5: Tiempo	31

Introducción

En esta práctica realizaremos medidas con el K2700 utilizando el programa HPVEE.

Para ello realizaremos una serie de programas de complejidad creciente capaces de realizar medidas de todos los dispositivos, mostrarlas por pantalla, en una gráfica y guardarlas en un fichero con formato de tabla.

También realizaremos programas capaces de medir un número de medidas y programas capaces de medir durante un tiempo determinado.

Objetivos

Realizaremos cinco programas de propósito específico:

- **Programa 1:** leer de un solo dispositivo un número de medidas y mostrarlas por pantalla. Fichero **P02Prog01.vee**.
- **Programa 2:** leer de todos los dispositivos un número de medidas y mostrarlas por pantalla. Fichero **P02Prog02.vee**.
- **Programa 3:** leer de todos los dispositivos un número de medidas y las guardarlas en un fichero con formato de tabla. Fichero **P02Prog03.vee**.
- **Programa 4:** leer de todos los dispositivos un número de medidas y mostrarlas en una gráfica. Fichero **P02Prog05.vee**.

- **Programa 5:** leer de todos los dispositivos durante un tiempo y guardar las medidas en un fichero con formato de tabla junto con el tiempo en el que fueron tomadas. Fichero **P02Prog05.vee**.

Práctica

Programa 1: Medidas de un sensor

En este programa realizaremos diez medidas de la PT100 y las mostraremos por pantalla.

El primer paso es configurar la entrada y salida en el menú **I/O > Instrument Manager** (Ilustración 11)

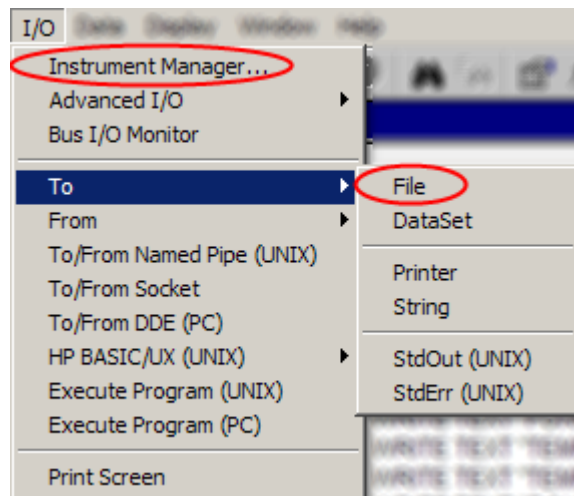


Ilustración 11

Dentro de esta ventana pulsamos **Add Instrument** (Ilustración 12)

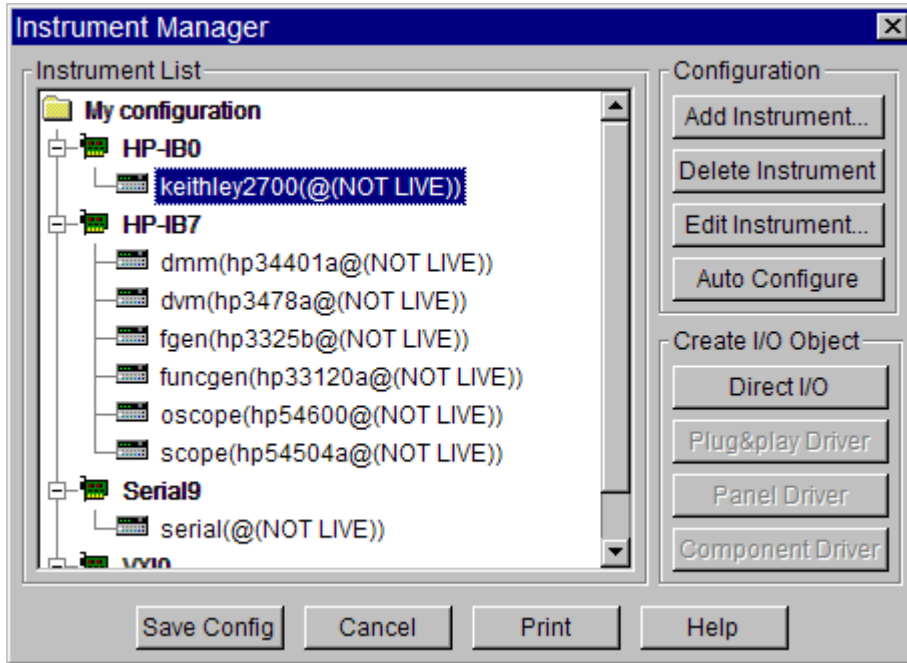
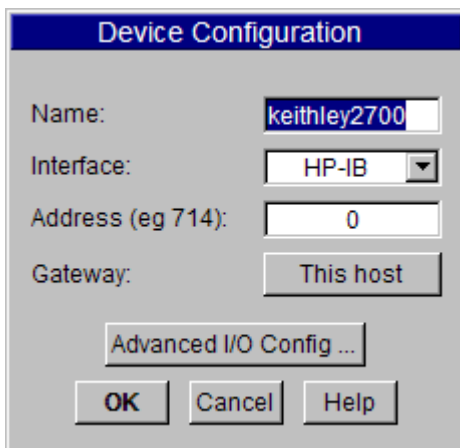


Ilustración 12

Configuramos los datos:



Name: Keithley2700
Interface: GPIB
Address: 16

Ilustración 13: configuración

Una vez configurado regresamos al menú **Instrument Manager** y esta vez pulsamos **Direct I/O**

Con esto crearemos un objeto para escribir instrucciones en el K2700 (Ilustración 15 y Ilustración 14).

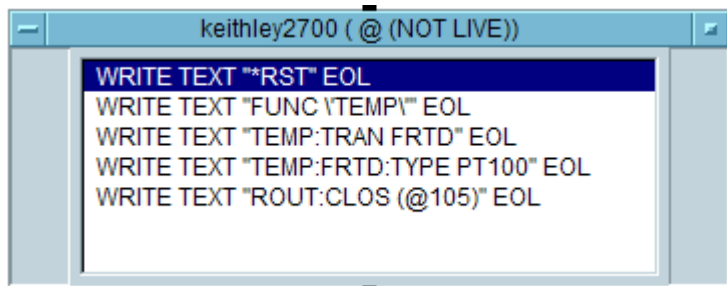


Ilustración 15: I/O Device con varias instrucciones



Ilustración 14: I/O device minimizado

Haciendo doble clic en cada línea se abre una ventana en la que podemos elegir lectura o escritura y en la que podemos escribir las instrucciones (Ilustración 16).

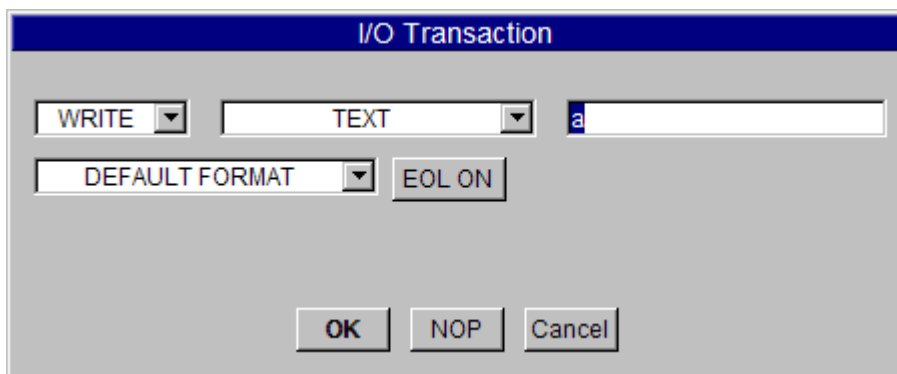


Ilustración 16: las instrucciones se deben escribir con el formato de los strings en C.

Escribimos las instrucciones necesarias para configurar el K2700 para leer datos de la PT100. Estas instrucciones se encuentran en el Anexo B.

Ahora necesitamos un dispositivo que nos permita leer secuencialmente las diez medidas: un objeto **For Range**. (Ilustración 17)

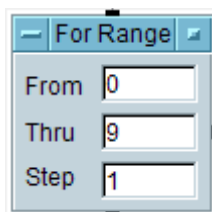






Ilustración 17

Este objeto se encuentra en el menú **Flow >Repeat**.

Conectamos el canal derecho del objeto I/O al canal superior del objeto **For Range**. Ahora creamos otro objeto I/O, le introducimos las instrucciones necesarias para leer un dato y conectamos su canal superior con el canal derecho del **For Range**. Si cometemos algún error al conectar estos dispositivos, hemos de utilizar el icono **Delete Line**  que se encuentra en la barra de herramientas.

Otros elementos útiles en la barra de herramientas son: el icono **Run** , que ejecuta el programa; el icono **Show Execution Flow** , que nos permite seguir el orden de ejecución de los objetos; o el icono **Show Data Flow** , que nos permite seguir el desplazamiento de los datos.

Para poder mostrar los datos por pantalla necesitaremos un **Logging Alphanumeric** (Ilustración 19 y Ilustración 20), objeto que se encuentra en el menú **Display**.



Ilustración 19

AlphaNumeric

Ilustración 18:
AlphaNumeric
minimizado

Creamos por lo tanto un **Logging Alphanumeric** que llamamos **PT100** y conectamos su canal izquierdo con el canal derecho del segundo objeto I/O.

El aspecto del programa terminado será similar a la Ilustración 21:

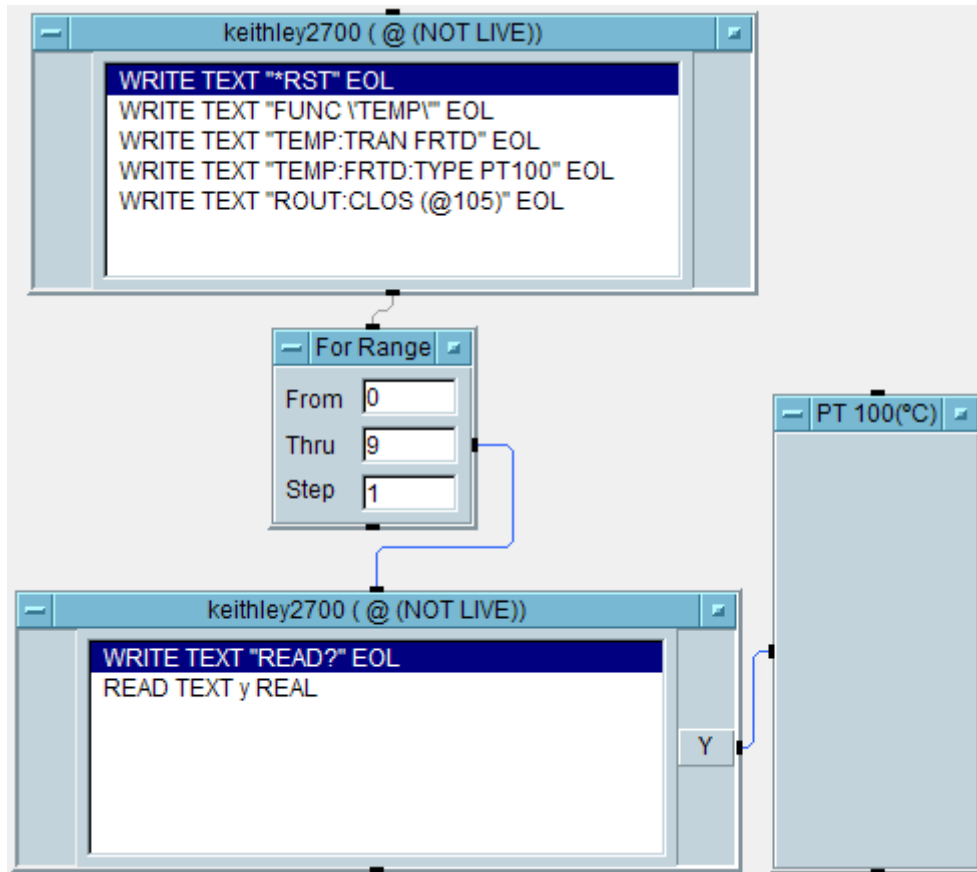


Ilustración 21: Programa 1

Programa 2: Medidas de varios sensores

En este programa realizaremos diez medidas de todos los dispositivos y las mostraremos en pantalla. Utilizando la configuración de entrada/salida del programa anterior, creamos un objeto I/O y le introducimos la instrucción de reseteo.

A continuación creamos un objeto **For Range**, y conectamos su canal superior con el canal inferior del objeto I/O.

Ahora necesitamos un objeto I/O para cada aparato, y en ellos colocaremos las instrucciones necesarias para medir en cada sensor [Ver Anexo B]. También necesitamos un **Logging Alphanumeric** para cada sensor.

Los objetos I/O se unirán los unos a los otros utilizando los canales superior e inferior, uniendo el superior al canal inferior del **For Range**. Los **Logging Alphanumeric** se unen por sus canales izquierdos con los canales derechos de sus objetos I/O respectivos.

Después de realizar estas uniones, el programa quedará como la Ilustración 22:

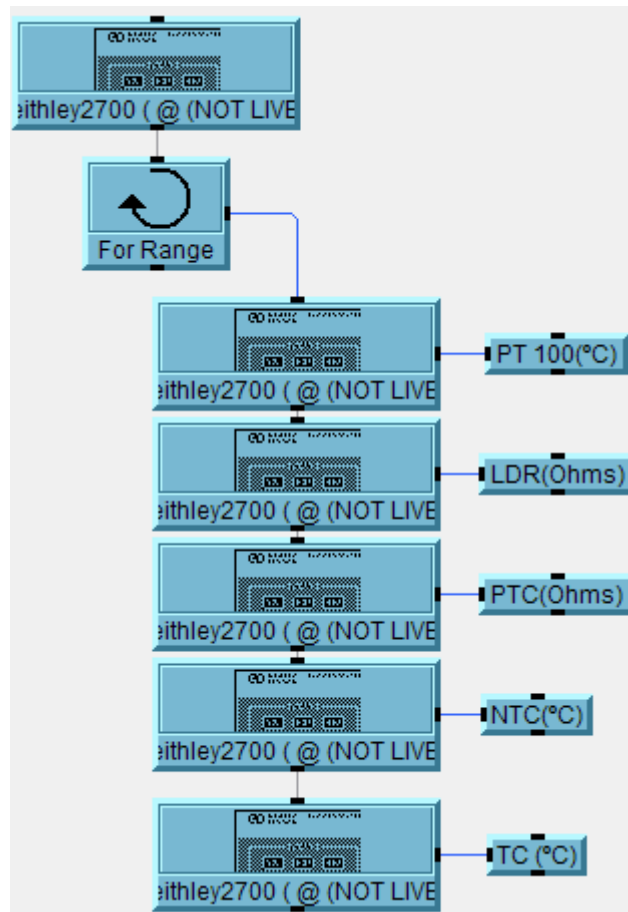


Ilustración 22: conexión en cascada

Programa 3: Fichero

Este programa lee diez medidas de todos los dispositivos y las guarda en un fichero con formato de tabla. Para la lectura de medidas crearemos seis objetos I/O, un objeto **For Range** y cinco objetos **Logging Alphanumeric**. Estos objetos los conectamos en el mismo orden del programa anterior e introducimos en los objetos I/O las mismas instrucciones que el programa anterior.

Con esto conseguimos una estructura capaz de leer los datos del K2700.

Para poder escribir en un fichero, necesitaremos un objeto **To File** (Ilustración 23), que se encuentra en el menú **I/O > To**.

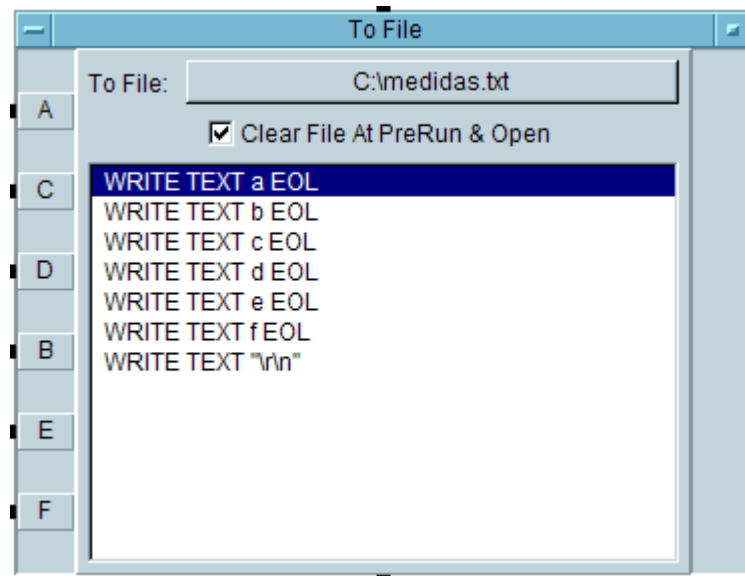


Ilustración 23: To File con varias entradas

Este objeto es similar al objeto I/O con el que enviamos instrucciones al K2700. En este caso también podemos enviar líneas, que en este caso se guardarán en el fichero. Además podemos seleccionar la ruta del mismo.

Unimos entonces el canal superior de un objeto **To File** en el que introducimos la cabecera del fichero al canal inferior del objeto I/O que manda el reseteo al K2700.

Ahora creamos otro objeto **To File** y le creamos seis entradas (Ilustración 24): una para cada sensor y una más para contar el número de medidas. Las conectamos con los canales derechos de los objetos I/O y del **For Range**.

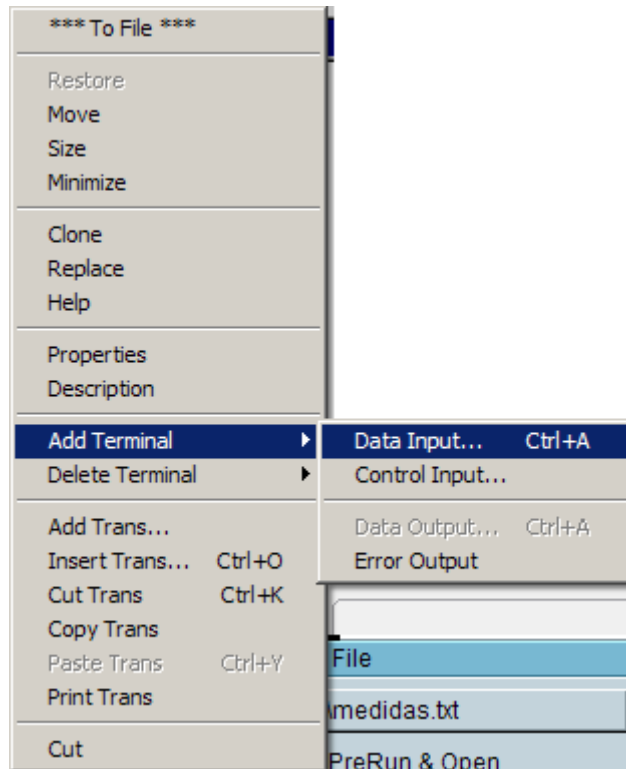


Ilustración 24

En ese objeto creamos las líneas para mandar los datos y un salto de línea en la última línea. Después modificaremos las propiedades (Menú contextual > Properties) el carácter de salto de línea, que lo modificaremos por “\t”. Al terminar el programa quedará como la Ilustración 25.

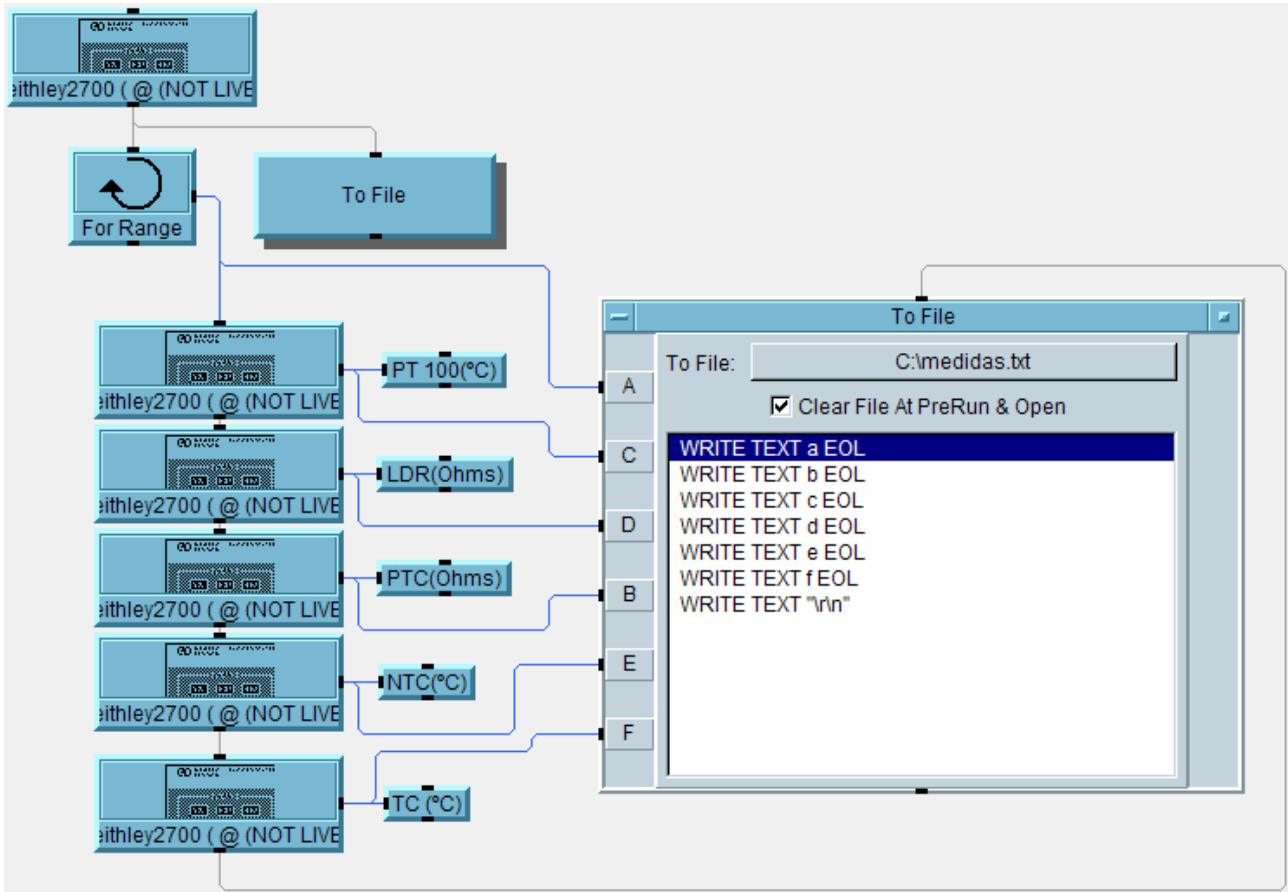


Ilustración 25: programa 3 finalizado

Programa 4: Gráficas

Este programa es similar al programa 2, pero en este caso los datos se representaran en gráficas en lugar de en **Logging Alphanumeric**. Por lo tanto, también crearemos 6 objetos I/O y un **For Range**. Estos objetos serán dispuestos y configurados de la misma manera que en dicho programa.

Las gráficas, objetos **X vs Y Plot** (Ilustración 16), se encuentran en el menú **Display**.

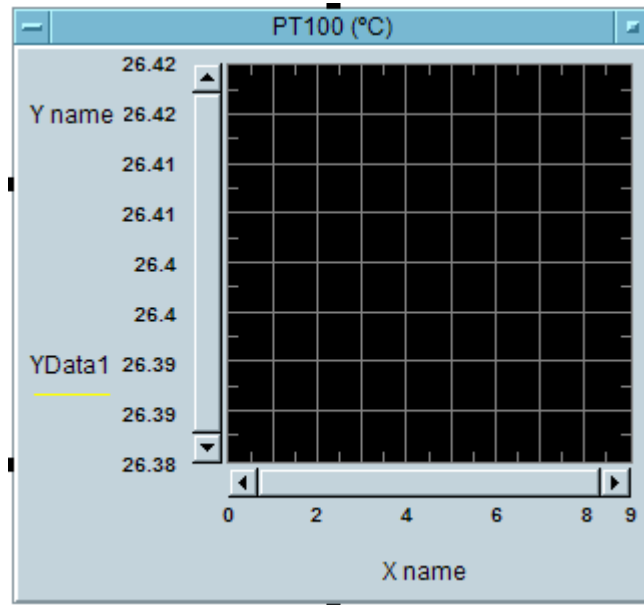


Ilustración 26

Creamos cinco objetos **X vs Y Plot** (uno para cada sensor) y conectamos la entrada izquierda superior de todas con la salida derecha del **For Range** y la salida izquierda inferior con la salida derecha del respectivo objeto I/O.

Como curiosidad, en este programa añadimos un último objeto I/O, con las instrucciones necesarias para que el K2700 muestre en su pantalla el texto “Finalizado”. Este objeto será conectado en su canal superior al canal inferior del **For Range**.

El programa resultante será igual a la Ilustración 27:

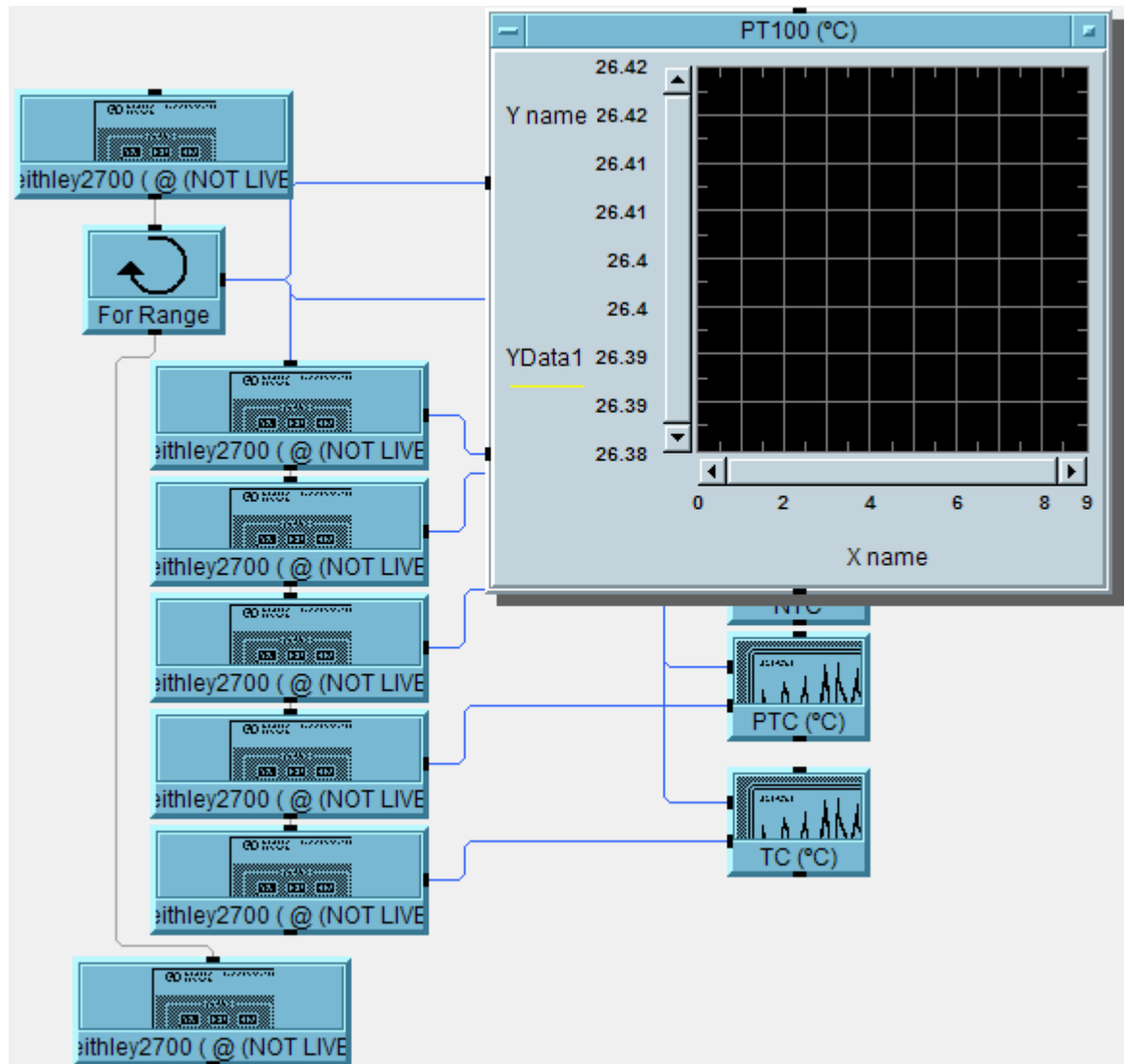


Ilustración 27

Programa 5: Tiempo

Este programa realizara medidas de los dispositivos durante 1800 segundos (media hora) y luego las guardara en un fichero de texto plano con formato de tabla.

Ya que es un programa dependiente del tiempo no nos servirá el objeto For Range. Necesitamos pues un objeto para medir el tiempo, el objeto **Timer** del menú **Device**(Ilustración 28).

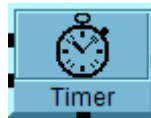


Ilustración 28

También necesitaremos objetos de control de flujo para leer datos mientras se cumpla una condición, en este caso que el número de segundos transcurridos $A < 1800$. Para controlar la condición utilizaremos un objeto **If/Then/Else** (Ilustración 29) que se encuentra en el menú **Flow > Repeat**. Si se cumple la condición introducida la ejecución del programa sigue por el camino **Then** y si no sigue por el camino **Else**.

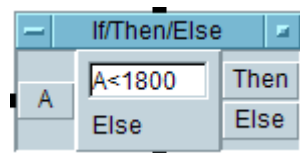


Ilustración 29

Para repetir las medidas utilizaremos objetos **Until Break** (Ilustración 32), **Next** (Ilustración 31) y **Break** (Ilustración 30), todos del menú **Flow**. Si el programa llega a un objeto **Next** regresará al objeto **Until Break** y continuara por la derecha. Si el flujo nos lleva a un **Break** regresará al objeto **Until Break** y continuara por abajo.

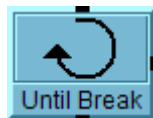


Ilustración 32



Ilustración 31



Ilustración 30

Necesitaremos también un objeto **Concatenator** (Ilustración 33), del menú **Data**, para concatenar todos los datos en una línea, es decir, es un array de una dimensión.

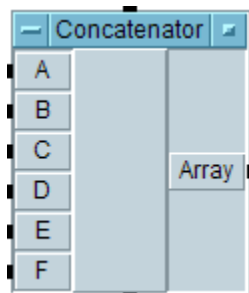


Ilustración 33

Por último necesitaremos un objeto para guardar los datos antes de escribirlos en el fichero, paso que se realiza al final de la ejecución del programa. Este será un objeto **Collector** (Ilustración 34) del menú **Data**.

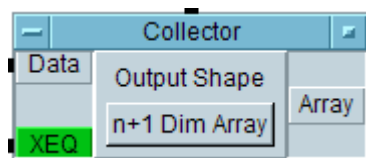


Ilustración 34

Ahora diseñaremos la parte del programa que realiza las medidas.

Creamos un objeto I/O para mandar una señal de **Reset** al K2700. Unimos el canal inferior de este objeto al canal superior del **Until Break**. Después creamos 5 objetos I/O, uno para cada sensor, e introducimos en ellos las instrucciones necesarias para leer del sensor correspondiente. Unimos los canales superiores e inferiores entre sí, el canal superior del primero con el canal derecho del **Until Break** y el canal inferior del último con el canal superior del **If/Then/Else**. En este bloque de objeto se realizarán las medidas.

Unimos ahora el canal izquierdo superior del **Timer** al inferior del objeto I/O que resetea, y el canal izquierdo inferior al canal derecho del **Until Break**. Con esto el **Timer** empezara a contar después de resetear el K2700 y devolverá el tiempo transcurrido cada vez que empecemos una serie de medidas.

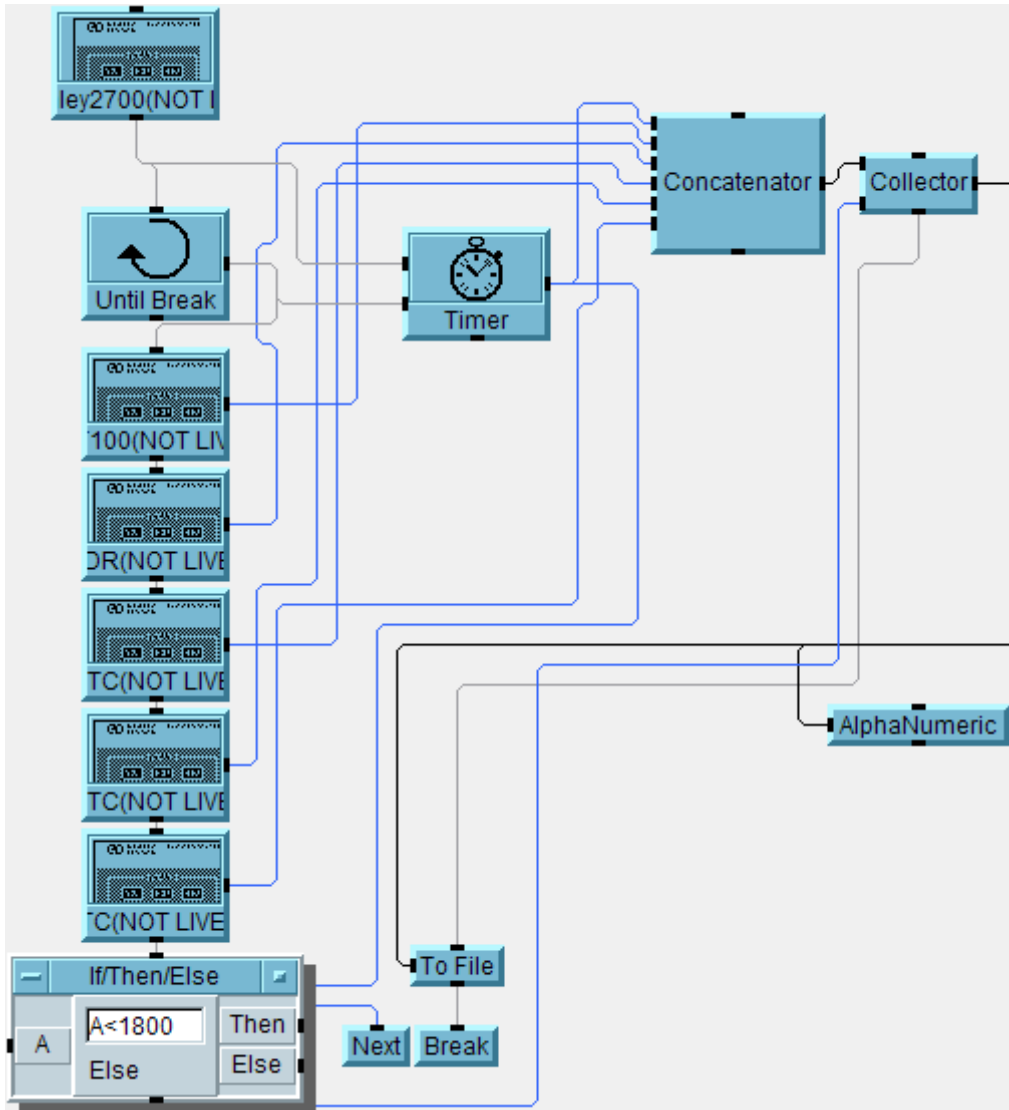
Creamos en el **Concatenator** seis entradas y las conectamos con los canales derechos de los objetos I/O de los sensores y del **Timer**. Así la salida del **Concatenator** será una cadena con el tiempo y todas las medidas, separadas con tabuladores.

Conectamos el canal derecho del **Concatenator** con el canal superior izquierdo del **Collector**, que ira guardando todas las cadenas con las medidas y el tiempo hasta el final del programa.

Creamos un objeto **To File**. Conectamos el canal izquierdo del mismo con el canal derecho del **Collector**, el canal superior con el canal inferior del **Collector** y el canal inferior con el **Break**. En la primera línea del objeto **To File** escribimos la cabecera del archivo y en la segunda escribimos el contenido del **Collector**. Este objeto, cuando se ejecute, escribirá el archivo entero en un solo paso (a diferencia del objeto **To File** del programa 3) y terminará la ejecución del programa.

Por último conectamos el canal derecho del **Timer** con el canal izquierdo del **If/Then/Else**, el canal **Then** con el **Next** y el canal **Else** con el canal izquierdo inferior del **Collector** para cerrar el bucle.

El aspecto final del programa será igual al siguiente:



Práctica 3: TestPoint 6.0

Introducción

En esta práctica aprenderemos a utilizar la herramienta TestPoint en su versión 6.0. TestPoint es una utilidad de creación de programas orientados a la adquisición de datos. TestPoint nos ofrece una gran facilidad para desarrollar interfaces gráficas de alta calidad y nos permite controlar un gran número de dispositivos existentes en el mercado.

Objetivos

En esta práctica desarrollaremos cinco programas en los que utilizaremos las características usadas con mayor frecuencia. Controlaremos el K2700 escribiendo directamente en el bus GPIB las ordenes necesarias para leer las medidas de los sensores y presentaremos la información de forma adecuada por la pantalla, bien sea en un cuadro de texto, una tabla o una gráfica. Por último, aprenderemos a volcar la información obtenida a un fichero de texto plano.


Los cinco programas que desarrollaremos son los siguientes:

- **Programa 1.** Lectura de un termopar tipo T en °C y presentación en una gráfica de esta. Fichero **P03Prog01.tst**.
- **Programa 2.** Igual que el anterior pero midiendo el tiempo transcurrido de cada medición. Fichero **P03Prog02.tst**.
- **Programa 3.** Igual que el anterior pero además añadiendo a una tabla (**Grid**) los valores obtenidos. Fichero **P03Prog03.tst**.
- **Programa 4.** Volcado a fichero de la tabla anterior. Fichero **P03Prog04.tst**.
- **Programa 5.** Lecturas de todos los dispositivos, almacenamiento en tabla de las mediciones, medida del tiempo transcurrido, presentación gráfica de tres de ellos y, por último, volcado a fichero de la tabla con formato y valores redondeados a las centésimas. Fichero **P03Prog05.tst**.


Práctica

Programa 1

Primero, crearemos un programa en blanco (**File > New**).





En TestPoint, antes de cualquier acción debemos definir los controles que vamos a utilizar. En este programa vamos a necesitar un **PushButton**  al que llamaremos **Adquirir**. Para añadir un nuevo PushButton debemos seleccionarlo desde la ventana **Stock** y soltarlo en la ventana **Panel** (que es donde definimos la interfaz gráfica de nuestro programa). Acto seguido, nos

aparecerá una ventana donde nos preguntarán qué nombre deseamos darle a nuestro nuevo control como la Ilustración 35.

Sabiendo esto, añadiremos también un control **GPIB**  al que llamaremos **K2700** y que será el encargado de la comunicación con el Keithley 2700. En el apartado **GPIB Address**, escribiremos 16, que es la dirección que estamos usando actualmente.

Obsérvese que cada vez que añadimos un control, aparece en la lista de la ventana **Objects**.

También añadiremos los siguientes controles:

- Un objeto **Loop**  al que nombraremos **Bucle** y que será el encargado de repetir el proceso de la medición.
- Un objeto **Graph**  que será el encargado de dibujar la gráfica y al que daremos el nombre de **Grafica**.
- Un objeto **Data-Entry**  que renombraremos por **Medidas** y será el encargado del número de lecturas que efectuará el bucle Loop.
- Un objeto **Display**  que denominaremos **Temperatura** y será el encargado de mostrar la temperatura medida en el termopar en cada instante.

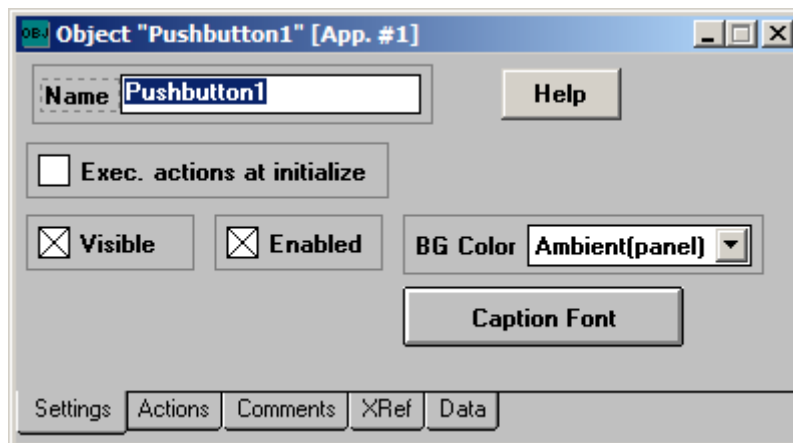


Ilustración 35: ventana Settings

Actualmente ya tenemos todos los objetos necesarios. Ahora aprenderemos a asignar las **Actions** o eventos a los controles.

En el menú contextual del PushButton “Adquirir” seleccionaremos **Action List**. En el Action List es donde escribiremos las instrucciones que se deben realizar cuando se haga clic en el botón.

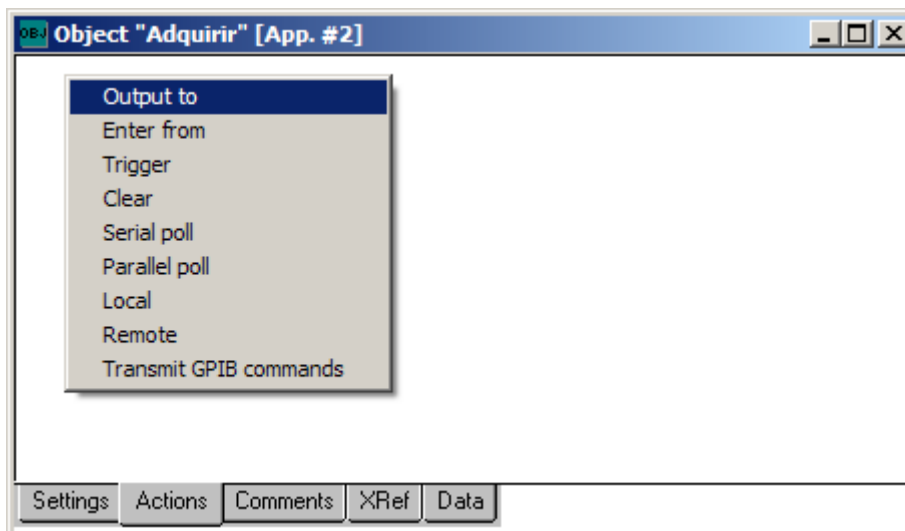


Ilustración 36: funciones de GPIB

Para crear una nueva instrucción, arrastraremos el objeto GPIB desde la ventana **Objects**, pinchando en el icono (no en la etiqueta de texto), hasta la Action List. Automáticamente, nos aparecerá un menú con las funciones disponibles del objeto GPIB. Seleccionaremos **Output To** y, a continuación, en el campo **Width**, escribiremos “*RST” para reiniciar el K2700.

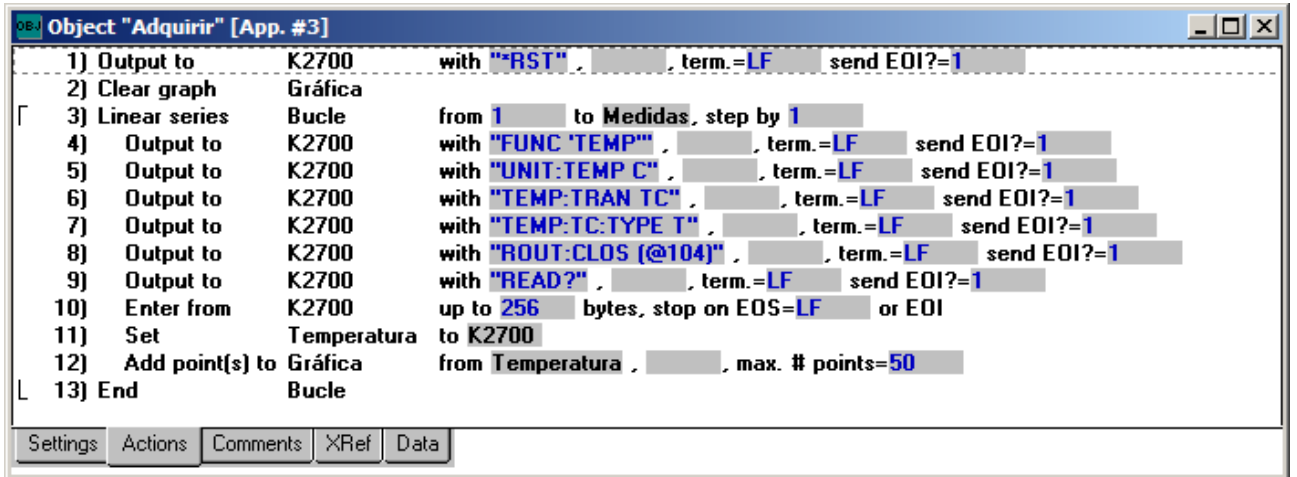
Lo siguiente será llamar a la función **Clear Graph** que seleccionaremos al arrastrar el objeto **Gráfica** en el Action List. Después, arrastraremos **Bucle** a la lista de acciones de **Adquirir** y seleccionaremos **Lineal Series**. Así se realizará incrementará de uno en uno el bucle hasta alcanzar el límite. En el campo **From** escribiremos el valor 1 y en el **To** arrastraremos el objeto **Medidas** directamente desde la ventana **Objects**.

Hecho esto, ahora escribiremos las instrucciones **interiores** del bucle. Puede observar la indentación que se produce al escribir nuevas instrucciones. De nuevo, usaremos en el K2700 con la instrucción Output To. Utilizaremos los comandos necesarios para la lectura de la temperatura de un termopar tipo T. Vea el anexo B para más detalles de las instrucciones.

Tras esto, utilizaremos la función **Enter From** del objeto GPIB gracias a la cual podremos leer los resultados de la medición. Ahora debemos mostrar por pantalla el resultado de esa lectura y para ello utilizaremos la opción **Set** del Display **Temperatura** y en el campo **To** especificaremos el K2700.

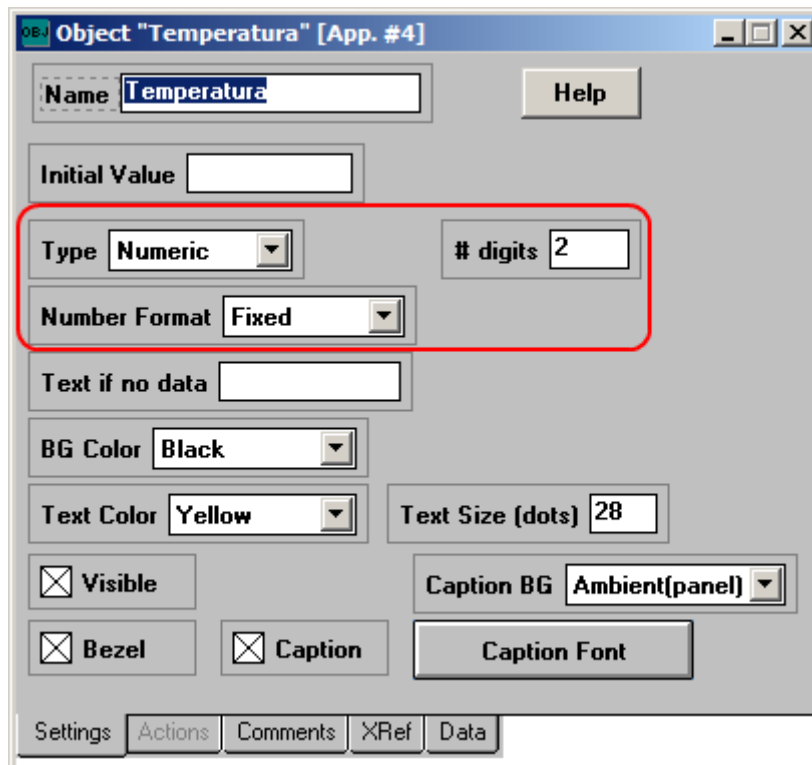
Finalmente, arrastramos el objeto **Gráfica** al Action List y elegimos la opción **Add Point(s) To**. En el campo **From** arrastramos el objeto K2700.

El Action List resultante debe ser como este.

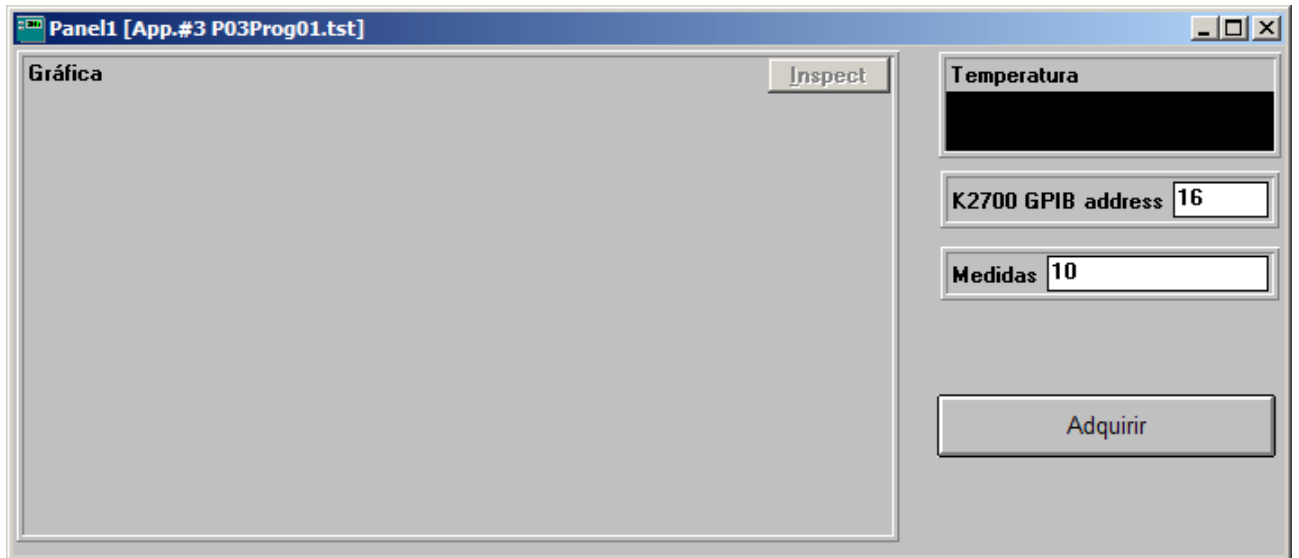


Antes de ejecutar nuestro primer programa, cambiaremos algunas propiedades del objeto Temperatura, ya que mostraría todo el contenido de la lectura y lo que a nosotros realmente nos interesa es el valor y dos cifras decimales. Para ello, hay que dirigirse a las Settings e indicar lo siguiente:

- **Type:** Numeric
- **# Digits:** 2
- **Number format:** Fixed



Este sería el aspecto de nuestro programa.



Ahora sólo nos queda hacer clic en el menú **Mode=Edit**, escribir un número de medidas mayor que 0 en **Medidas** y pulsar el botón **Adquirir**.

Nota: para no tener que escribir continuamente un número de medidas, en modo edición, rellene el campo **Initial Value** con un valor en las Settings de Medidas.

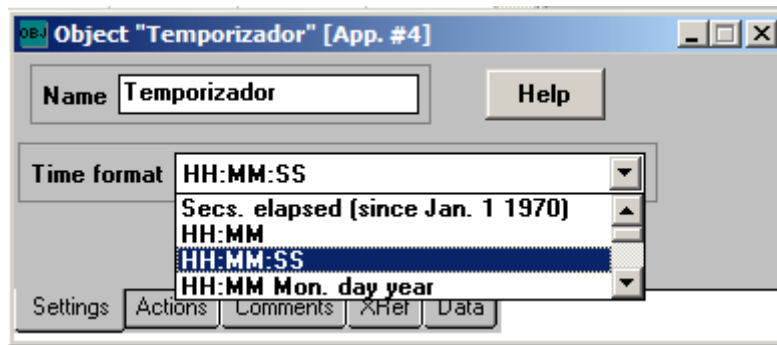
Programa 2

Partiendo del trabajo realizado en el programa 1. Ahora vamos a incorporar el objeto **Timer**



para medir el tiempo transcurrido entre cada medición. En concreto, lo que haremos será medir el tiempo transcurrido desde el comienzo de la ejecución hasta el momento en el que se realiza la medida.


Para ello, vamos a añadir un objeto **Timer** al que llamaremos **Temporizador**. En la opción **Time Format** seleccionaremos "Secs. elapsed...". Mediante este sistema después podremos realizar una diferencia aritmética entre el Tiempo Inicial y el actual.



Evidentemente, tendremos que almacenar el instante inicial. Para ello utilizaremos el objeto



Container. Los containers son variables que permiten almacenar todo tipo de datos. Le llamaremos **TiempoInicial**.

Para calcular la diferencia de tiempo, tendremos que utilizar la resta. Con el objeto **Math**  calcularemos la resta. Crearemos uno con la siguiente fórmula: "x - y" y le llamaremos **DiferenciaTiempo**.

También añadiremos un nuevo objeto Display al que llamaremos **TiempoTranscurrido** donde se mostrará la diferencia de tiempo.

Hecho esto, ahora sólo debemos modificar el Action List de Adquirir.

Arrastraremos el objeto **Temporizador** antes de entrar en el bucle y le asignaremos un **Interval** de 0.1. Así mismo, almacenaremos el instante inicial arrastrando **TiempoInicial** con la función **Store In** utilizando **Temporizador** en el campo **From**.

Después del bucle, calcularemos la Diferencia de tiempo. Para ello, utilizaremos la función **Calculate** del objeto DiferenciaTiempo y en los parámetros x e y colocaremos **Temporizador** y **TiempoInicial** respectivamente. Acto seguido, mostraremos el valor por pantalla con la función **Set** del objeto **TiempoTranscurrido**, donde el campo **From** tomará el valor del resultado de DiferenciaTiempo.

Finalmente, pararemos el Temporizador con la función **Stop**.

Consejo: si quiere ver la ejecución paso a paso utilice **Single Step Mode** en el menú **Debug**.

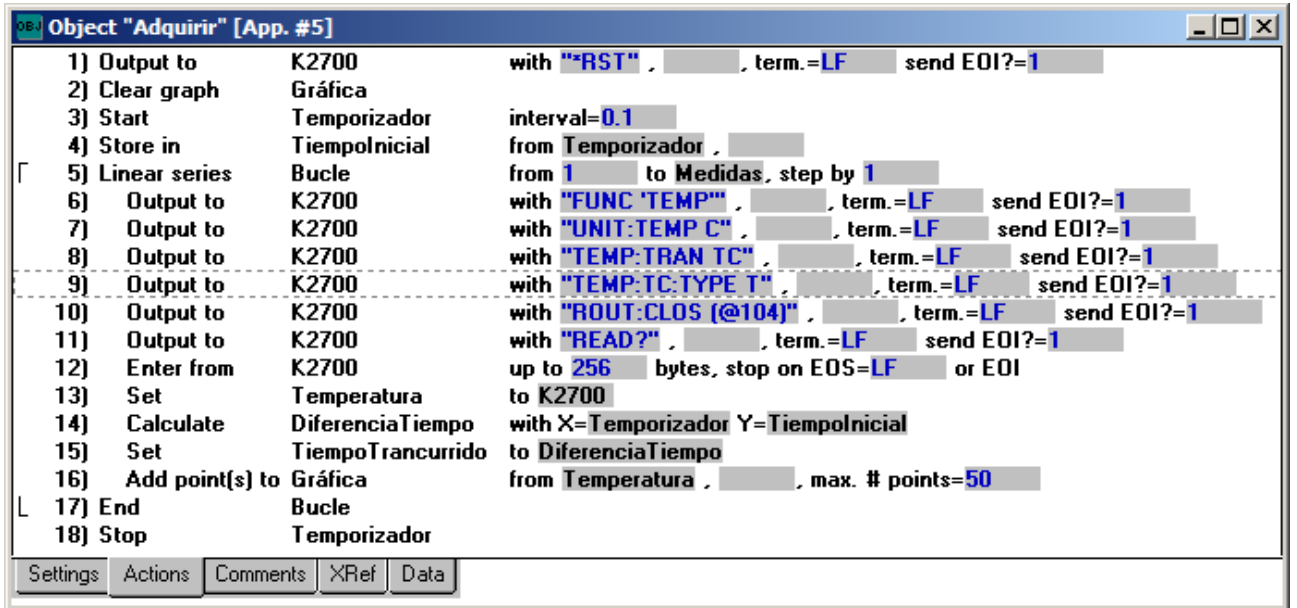


Ilustración 37: action list de Adquirir para el Programa 2

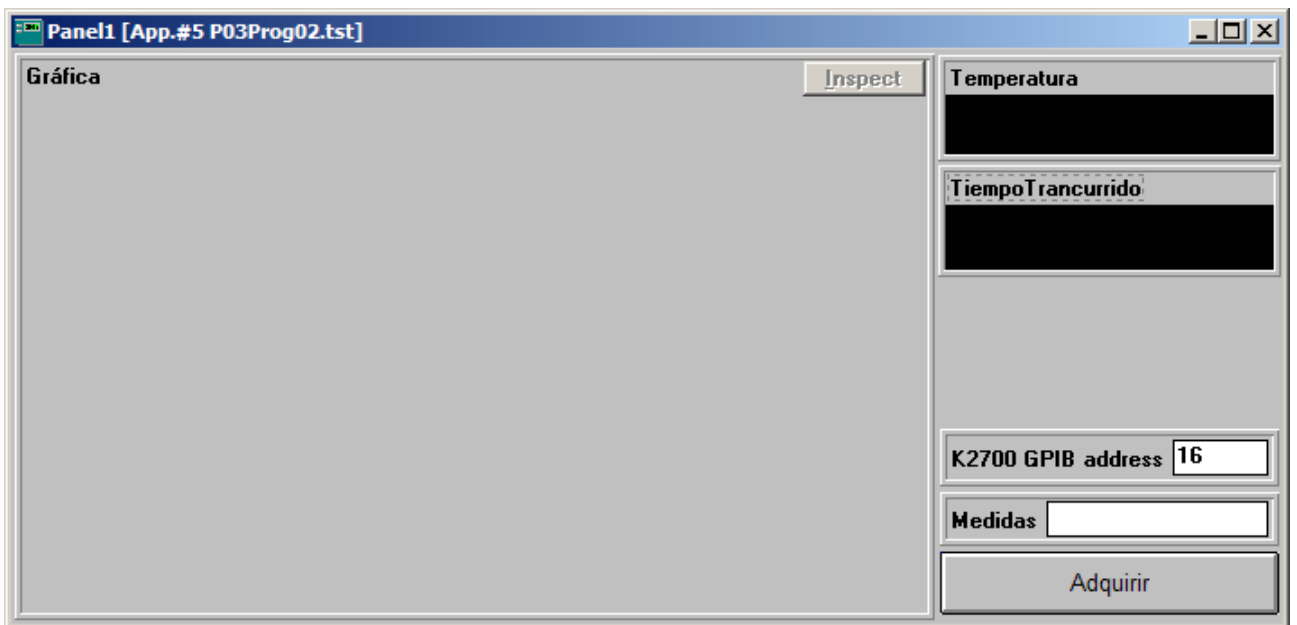



Ilustración 38: controles alineados en el Panel

Programa 3

En el programa 3 almacenaremos los resultados en una tabla (Grid ) . Así podremos consultar todas las mediciones realizadas durante la ejecución del programa.

Como ya hiciéramos antes, partiremos del programa anterior (Programa 2) y añadiremos varios controles más así como varias instrucciones en el Action List de Adquirir.

Añadiremos los siguientes objetos:

- Un objeto de tipo **Grid** al que denominaremos **Tabla**. En el campo **Max. Cols** insertaremos el valor 2 (número de columnas máximo).
- Un **Container** al que llamaremos **Muestra**.
- Un objeto **Math** al que llamaremos **Incremento** y será el encargado de incrementar en una unidad la variable Muestra. En él escribiremos la siguiente fórmula: “ $x + 1$ ”.

En el Action List de Adquirir escribiremos las siguientes sentencias:

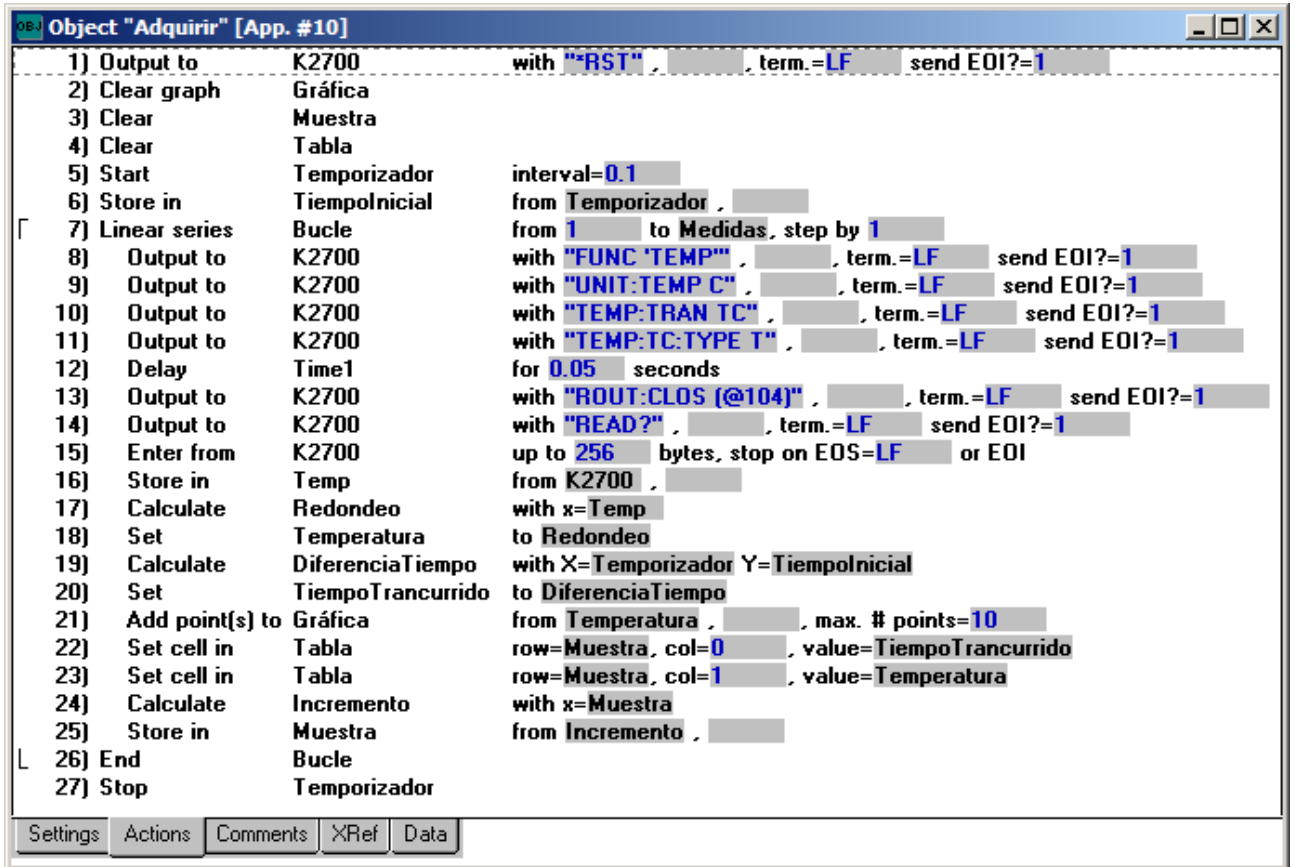
Después de la acción Clear Graph, añadiremos **Clear Muestra** para que siempre empiece a contar desde 0.

También haremos un Clear del objeto Tabla.

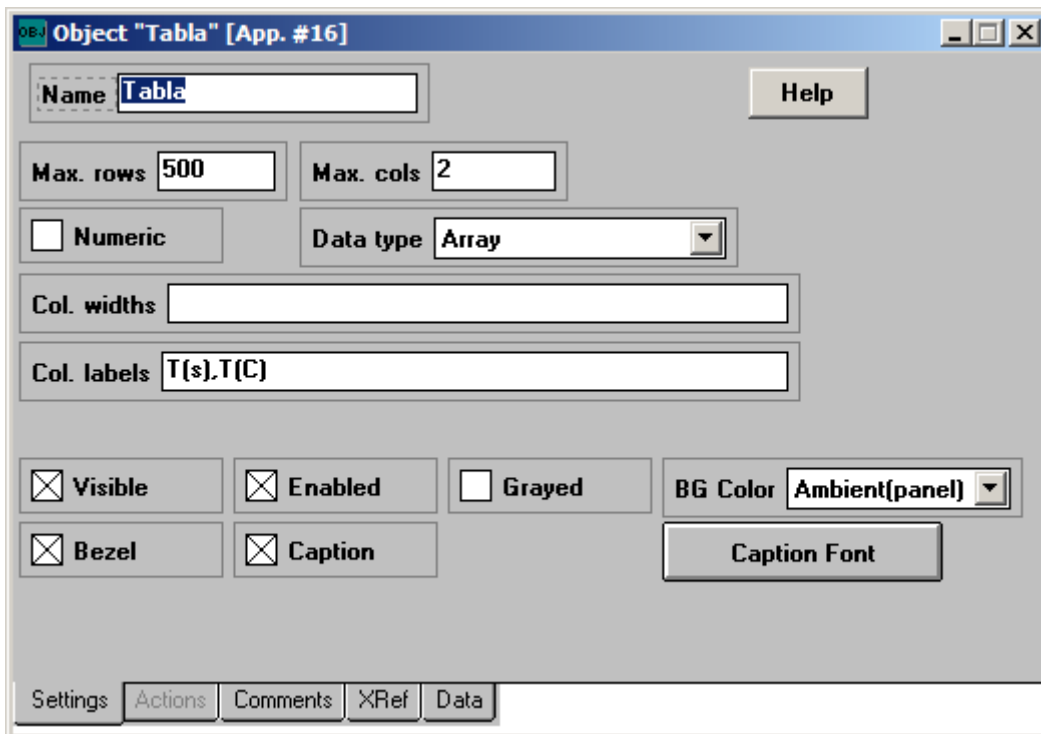
Antes de acabar el bucle, utilizaremos la función **Calculate** de **Incremento** con $x = \text{Muestra}$ y almacenaremos el resultado en **Muestra** con la función Store In.

Finalmente arrastraremos el objeto **Tabla** después de la línea **Add point(s) to** y seleccionaremos **Set Cell In**. En campo **row** (fila) utilizaremos el valor de **Muestra**, en **col** el valor **0** y en **value**, **DiferenciaTiempo**. Reiteramos el proceso nuevamente, salvo que esta vez podremos **col = 1** y **value = Temperatura**.

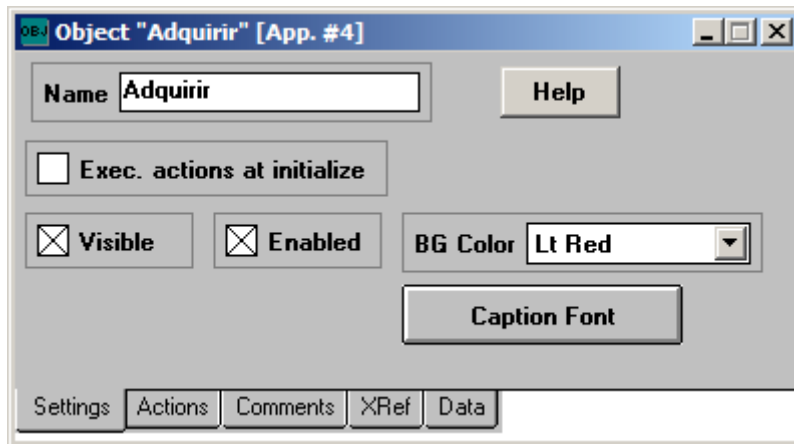
La lista de acciones de Adquirir debería ser como la siguiente.



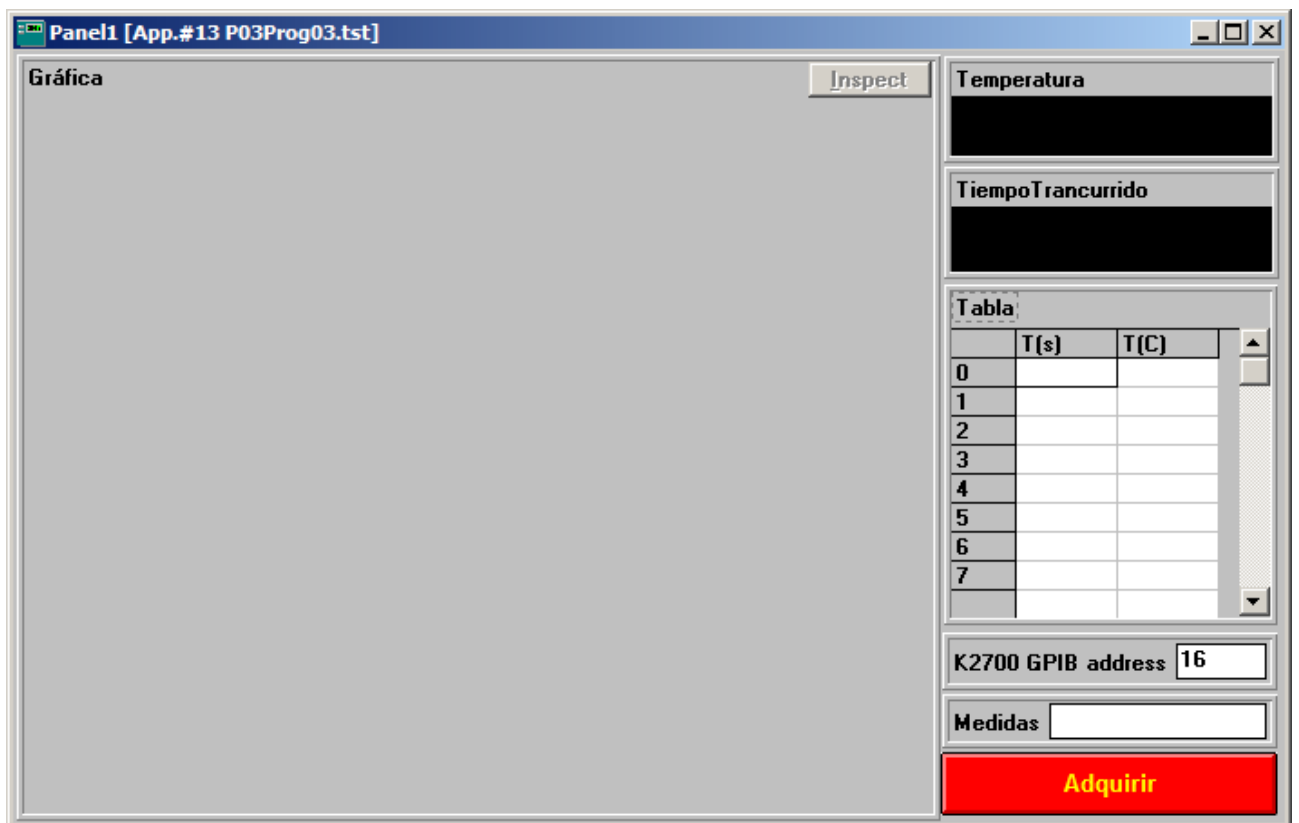
Ahora, haremos una pequeña modificación a las Settings de la tabla. En la propiedad **Col. Labels** vamos a separar con comas las etiquetas de las cabeceras.



También estableceremos un color personalizado para el botón Adquirir. Desde la pestaña Settings cambiaremos la propiedad **BG Color** a **Lt Red** y haciendo clic en **Caption Font**, cambiaremos el color de la fuente a **Yellow**.




Finalmente, nuestra interfaz debería tener la siguiente apariencia.



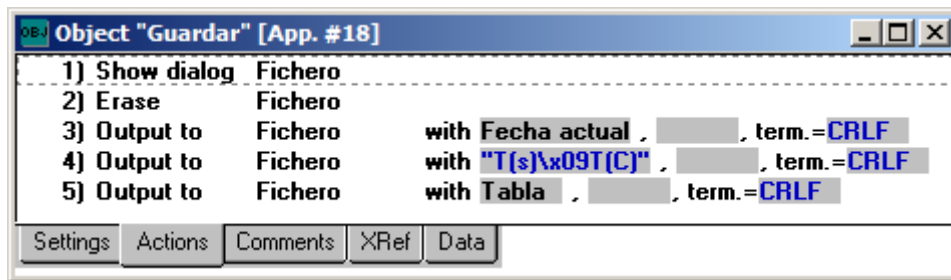
Programa 4

En el programa 4 vamos a escribir en un fichero de texto plano el contenido de la tabla que anteriormente hemos rellenado. Para llevar a cabo esta acción, al programa anterior, vamos a añadirle los siguientes objetos:

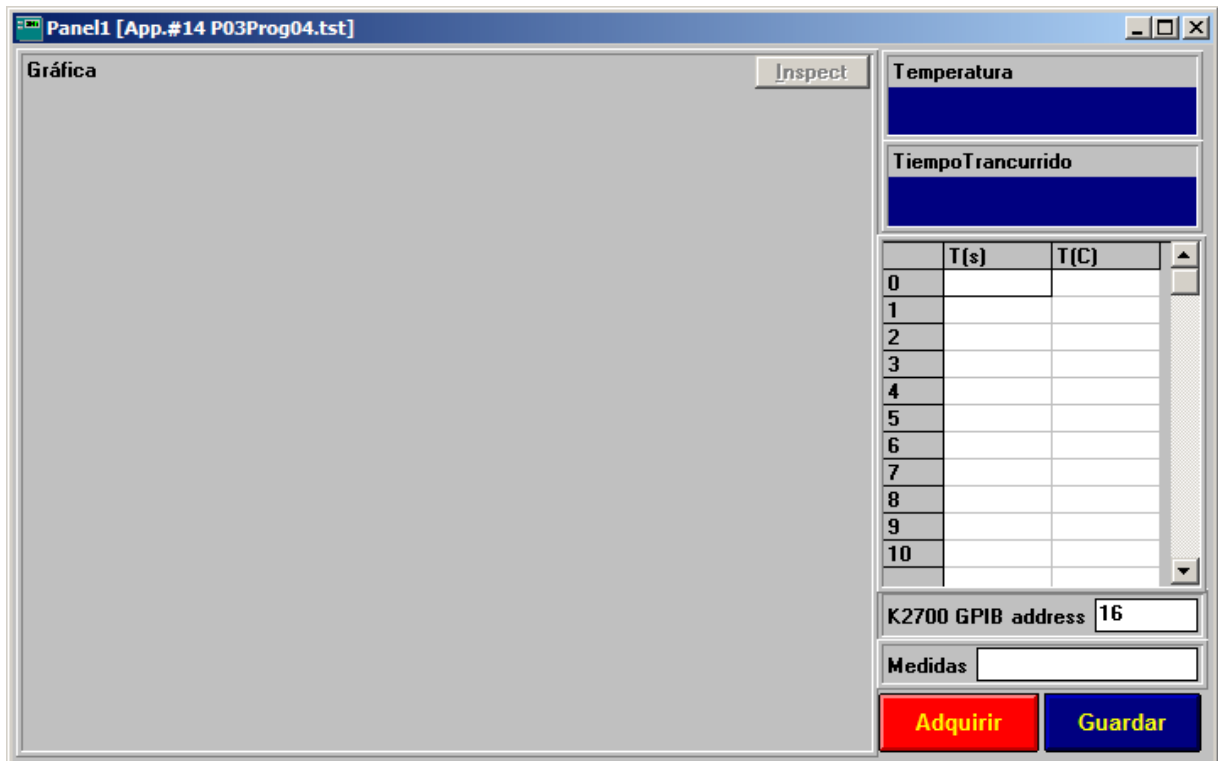
- Un **PushButton** al que llamaremos **Guardar**.
- Un objeto de tipo **File**  al que llamaremos **Fichero**.
- Un objeto **Timer** que llamaremos **FechaActual**. En sus propiedades elegir la opción **HH:MM:SS dd/mm/yy** en el apartado **Time Format**.

En el siguiente paso, lo que debemos es escribir las instrucciones de la lista de acciones de Guardar. Para ello, arrastraremos el objeto Fichero y elegiremos la opción **Show Dialog**. Acto seguido, utilizaremos la función **Erase** de Fichero.

Finalmente escribiremos en Fichero el Timer **FechaActual** y **Tabla** tal y como se muestra en la captura.



Después, modificaremos un poco la interfaz como ya hemos descrito anteriormente:



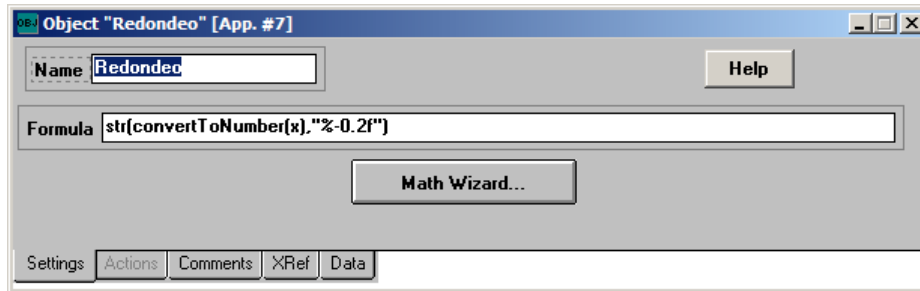
Ahora sólo queda ejecutar el programa, tomar un número de medidas con **Adquirir** y después **Guardar** para ver el resultado.

Programa 5

Finalmente, en el programa 5, haremos lo mismo que en el Programa 4 pero midiendo en todos los dispositivos simultáneamente.

En la gráfica añadiremos tres dispositivos: PT100, NTC y TC.

Además por cuestiones de redondeo, añadiremos varios containers a los que después calcularemos la función matemática siguiente:



Suponemos que hechos los anteriores programas, no debe entrañar ninguna dificultad entender la siguiente Action List de Adquirir.

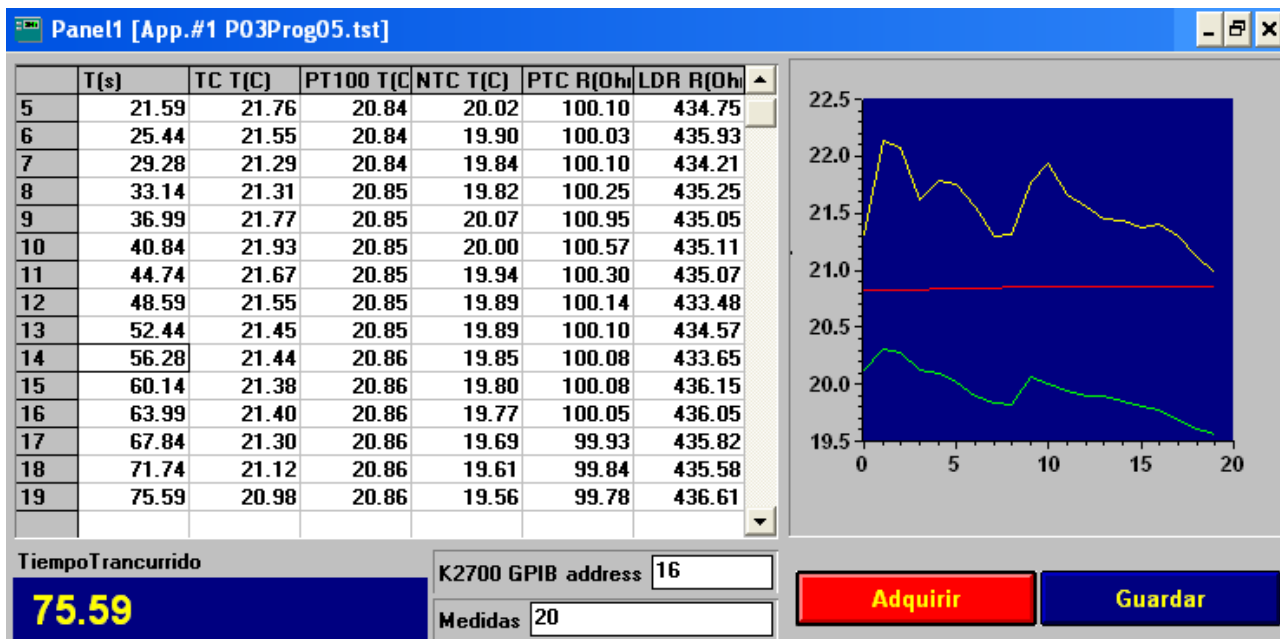


Object "Adquirir" [App. #1]

1) Output to	K2700	with "RST" , , term.=LF send EOI?=1
2) Clear graph	Gráfica	
3) Clear	Muestra	
4) Start	Temporizador	interval=0.1
5) Store in	Tiempolnicial	from Temporizador ,
6) Linear series	Bucle	from 1 to Medidas , step by 1
7) Output to	K2700	with "FUNC 'TEMP'" , , term.=LF send EOI?=1
8) Output to	K2700	with "UNIT:TEMP C" , , term.=LF send EOI?=1
9) Output to	K2700	with "TEMP:TRAN TC" , , term.=LF send EOI?=1
10) Output to	K2700	with "TEMP:TC:TYPE T" , , term.=LF send EOI?=1
11) Delay	Time1	for 0.2 seconds
12) Output to	K2700	with "ROUT:CLOS (@104)" , , term.=LF send EOI?=1
13) Output to	K2700	with "READ?" , , term.=LF send EOI?=1
14) Enter from	K2700	up to 256 bytes, stop on EOS=LF or EOI
15) Store in	TC	from K2700 ,
16) Calculate	Redondeo	with x=TC
17) Store in	TC	from Redondeo ,
18) Calculate	DiferenciaTiempo	with X=Temporizador Y=Tiempolnicial
19) Set	TiempoTrancurrido	to DiferenciaTiempo
20) Set cell in	Tabla	row=Muestra, col=0 , value=TiempoTrancurrido
21) Set cell in	Tabla	row=Muestra, col=1 , value=TC
22) Output to	K2700	with "FUNC 'TEMP'" , , term.=LF send EOI?=1
23) Output to	K2700	with "TEMP:TRAN FRTD" , , term.=LF send EOI?=1
24) Output to	K2700	with "TEMP:FRTD:TYPE PT100" , , term.=LF send EOI?=1
25) Delay	Time1	for 0.2 seconds
26) Output to	K2700	with "ROUT:CLOS (@105)" , , term.=LF send EOI?=1
27) Output to	K2700	with "READ?" , , term.=LF send EOI?=1
28) Enter from	K2700	up to 256 bytes, stop on EOS=LF or EOI
29) Store in	PT100	from K2700 ,
30) Calculate	Redondeo	with x=PT100
31) Store in	PT100	from Redondeo ,
32) Set cell in	Tabla	row=Muestra, col=2 , value=PT100
33) Output to	K2700	with "FUNC 'TEMP'" , , term.=LF send EOI?=1
34) Output to	K2700	with "TEMP:TRAN THER" , , term.=LF send EOI?=1
35) Output to	K2700	with "TEMP:THER 1950" , , term.=LF send EOI?=1
36) Delay	Time1	for 0.2 seconds
37) Output to	K2700	with "ROUT:CLOS (@103)" , , term.=LF send EOI?=1
38) Output to	K2700	with "READ?" , , term.=LF send EOI?=1
39) Enter from	K2700	up to 256 bytes, stop on EOS=LF or EOI
40) Store in	NTC	from K2700 ,
41) Calculate	Redondeo	with x=NTC
42) Store in	NTC	from Redondeo ,
43) Set cell in	Tabla	row=Muestra, col=3 , value=NTC
44) Output to	K2700	with "FUNC 'RES'" , , term.=LF send EOI?=1
45) Delay	Time1	for 0.2 seconds
46) Output to	K2700	with "ROUT:CLOS (@102)" , , term.=LF send EOI?=1
47) Output to	K2700	with "READ?" , , term.=LF send EOI?=1
48) Enter from	K2700	up to 256 bytes, stop on EOS=LF or EOI
49) Store in	Temp	from K2700 ,
50) Calculate	Redondeo	with x=Temp
51) Store in	Temp	from Redondeo
52) Set cell in	Tabla	row=Muestra, col=4 , value=Temp
53) Output to	K2700	with "FUNC 'RES'" , , term.=LF send EOI?=1
54) Delay	Time1	for 0.2 seconds
55) Output to	K2700	with "ROUT:CLOS (@101)" , , term.=LF send EOI?=1
56) Output to	K2700	with "READ?" , , term.=LF send EOI?=1
57) Enter from	K2700	up to 256 bytes, stop on EOS=LF or EOI
58) Store in	Temp	from K2700 ,
59) Calculate	Redondeo	with x=Temp
60) Store in	Temp	from Redondeo
61) Set cell in	Tabla	row=Muestra, col=5 , value=Temp
62) Add point(s) to	Gráfica	from TC , PT100 , NTC , , max. # points=10000
63) Calculate	Incremento	with x=Muestra
64) Store in	Muestra	from Incremento ,
65) End	Bucle	
66) Stop	Temporizador	

Settings Actions Comments XRef Data

Finalmente, un ejemplo de este programa en ejecución podría ser el siguiente:



Y con esto concluimos la práctica de TestPoint.

Práctica 4: NI LabView 6i

Índice

Introducción.....	51
Objetivos	52
Práctica.....	53
Programa 1. Reseteo del K2700.....	53
Programa 2. Lectura del K2700.....	55
Programa 3. Escritura a disco.	59
Programa 4. Medición simultanea temporizada con volcado a fichero.....	62
Creación de un Array de controles.....	62
Bucle For Loop.....	62
Array de medidas.....	63
Creación de la tabla de medidas (array 2D)	64
Calculo de tiempo transcurrido.....	64
Modificación del Array de medidas.....	65
Representación en una tabla.....	65
Formateo de Array 2D en texto.....	66

Introducción


En esta práctica desarrollaremos cuatro programas utilizando el entorno LabView para realizar medidas y almacenar los datos extraídos del K2700 en un fichero externo.

LabView es un entorno de desarrollo de programas parecido a otros entornos de programación estructurada. Sin embargo LabView utiliza un lenguaje de programación gráfico para crear programas en forma bloques de diagramas.

Además, LabView incluye las herramientas comunes de un entorno de desarrollo: *breakpoints*, traza de ejecuciones, etc. Haciendo más fácil desarrollar los programas.


Objetivos

Durante el transcurso de esta práctica diseñaremos cuatro programas:

- Un programa que envíe una sola instrucción al K2700. En el ejemplo concreto enviaremos la función Reset (*RST). Fichero **P04Prog01.vi**.
- Un programa para leer la medición de la LDR utilizando estructuras secuenciales ( Sequence). Fichero **P04Prog02.vi**.
- Un programa que escriba la lectura anterior en un fichero de texto plano. Fichero **P04Prog03.vi**.
- Un programa que escriba varias medidas simultáneas de todos los dispositivos (LDR, PTC, NTC, PT100, TC) en un fichero de texto con formato de tabla. Adicionalmente, esa tabla también se muestra en el programa. Fichero **P04Prog04.vi**.

Práctica

Programa 1. Reseteo del K2700.

El objetivo de este programa es resetear la unidad de adquisición de datos K2700. Para ello le enviaremos un comando *RST. Con este fin utilizaremos la función **GPIB Write**  en la ventana de Diagrama (la del fondo blanco). GPIB Write se encuentra en el menú **Functions > I/O Instruments > GPIB > GPIB Write** (Ilustración 39) y nos permite escribir directamente comandos a la unidad. El menú **Functions** se abre al hacer clic con el botón derecho del ratón sobre el fondo de la ventana.

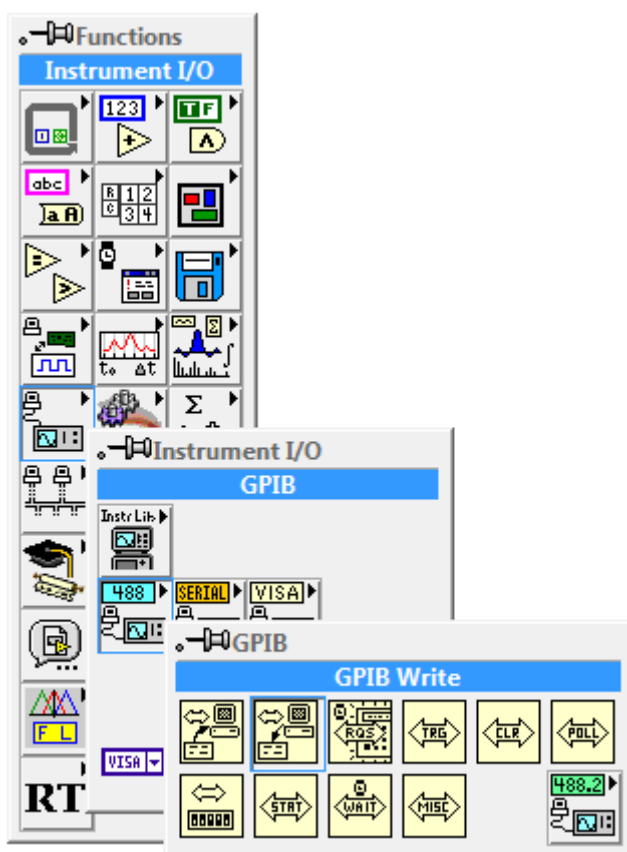




Ilustración 39

En la ventana de Panel (la del fondo gris) crearemos dos **String Control**  que llamaremos **Comando** y **Dirección**. Si no está visible la ventana de Panel, haga clic en el menú **Windows > Show Panel**. Cada vez que creemos un control en el Panel, un icono representativo aparecerá en la ventana de Diagrama. Para mover libremente los controles por el Panel, utilice la herramienta  **Position/Size/Select** en la paleta de herramientas.

Antes de proseguir, hay que señalar las diferencias entre **String Control** y **String Indicator**. Ambos objetos nos serán muy útiles a lo largo del desarrollo de la práctica, pero hay que saber que los controles marcados como **Control** son de sólo-escritura y los marcados como **Indicator** son de sólo-lectura.

Otra detalle que conviene recordar es que los iconos que aparecen en el Diagrama tienen un significado. El color además indica el tipo de datos (el color magenta indica un tipo String).

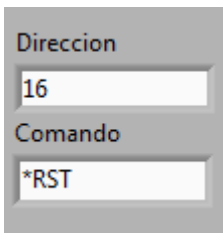



Ilustración 40: vista de Diagrama.

Teniendo en cuenta estas consideraciones básicas, podemos proseguir con el objetivo de esta práctica. Si hemos realizado los pasos anteriores correctamente, en la ventana de Panel deberíamos tener una interfaz semejante a la indicada en la Ilustración 40, sin los valores 16 ni *RST.

Para introducir los valores respectivos a cada String Control, vamos a utilizar la herramienta  **Edit Text**. Al hacer clic en el espacio designado para escribir con esta herramienta podremos introducir texto. En campo Dirección, indique en qué puerto GPIB se encuentra el K2700 (en nuestro caso el 16). En el campo Comando escriba el comando *RST que reiniciará la unidad a su estado inicial.

Mientras tanto, en la ventana de Diagrama, deberíamos tener algo similar a lo que se muestra en la ilustración 3. Si no se muestra la ventana de Diagrama, haga clic en el menú **Window > Show Diagram**. Los “hilos” que actualmente unen todos los objetos son el fundamento de LabView. El color de estos hilos muestra el tipo de datos que se está uniendo y el grosor la “cantidad” (un valor simple, un array, una matriz, etc.).

Para nuestro programa uniremos la entrada **Data** del objeto GPIB al objeto **Comando** y la entrada **Address String** al objeto **Direccion**. Tal y como se indica en la Ilustración 41.

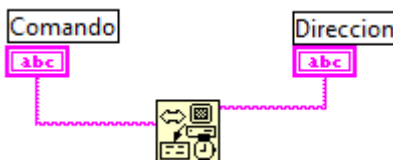







Ilustración 41



Ilustración 42

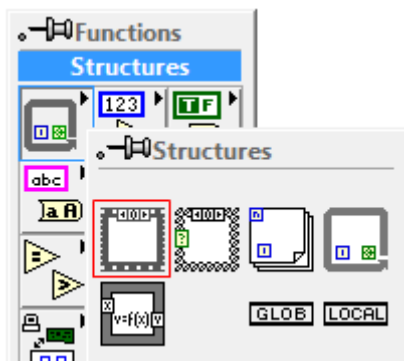
Para tal efecto, seleccionaremos la herramienta  **Connect Wire** desde la paleta (Ilustración 42) y clickeando respectivamente donde hemos indicado anteriormente tendremos configurada la escritura en el K2700.

Una alternativa a este programa podría ser poner los datos de entrada como constantes si sabemos que alguno de los datos no cambiará. Así pues, utilizaremos un objeto **String Constant**  del menú String, en sustitución del objeto String Control .

Si hemos seguido todos los pasos como se ha indicado, podremos hacer clic en el comando **Run**  en la barra de herramientas y se ejecutará nuestro programa. Si por algún motivo se nos hubiese olvidado conectar algún elemento esencial de la configuración, el icono de Run aparecería como en la siguiente imagen: .

Programa 2. Lectura del K2700.

En este programa leeremos una medida de la LDR y lo mostraremos por pantalla. Además ilustraremos el uso de la secuencia. Lo primero que haremos será insertar una secuencia en el diagrama desde el menú **Functions > Structures > Sequence**.



Después, pinche y arrastre hasta crear un objeto de un tamaño aceptable. Asegúrese de que la secuencia tiene al menos tres marcos. Verá el número de marcos en la parte superior central de la secuencia.

Si necesita añadir más marcos o eliminar los existentes, en el menú contextual encontrará **Add Frame After**, **Add Frame Before** y **Delete This Frame** que añadirán un marco antes del actual, después del actual o eliminará el marco seleccionado respectivamente.

En el primer marco de la secuencia (el número 0), creamos un objeto **String Constant** con el valor *RST y un objeto **GPIB Write** .

Crearemos un objeto **String Control** en el Panel y lo llamaremos **Dirección**. Conectaremos éste como hiciéramos en el apartado anterior y deberíamos tener un diagrama similar al mostrado en la Ilustración 43.

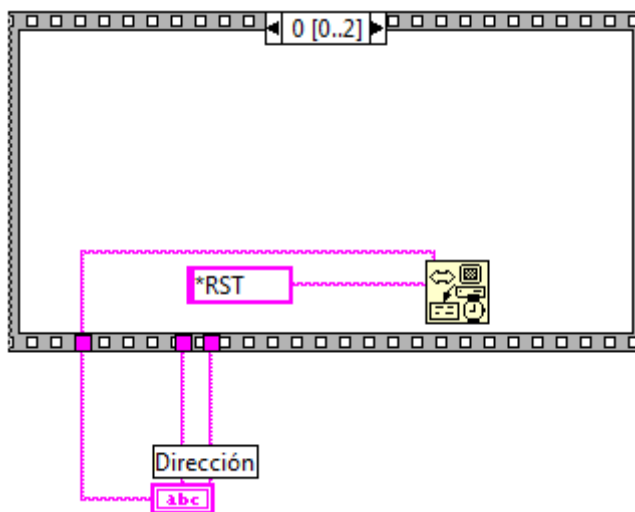



Ilustración 43

Como se puede observar, al conectar un elemento de fuera con otro de dentro del marco, un pequeño rectángulo se muestra en el borde.

En la segunda celda creamos otro objeto **GPIO Write** y un objeto **String Control**, al que llamaremos **Comando**. Este último objeto debe ser creado en el Panel. Si el icono del objeto aparece fuera del marco en el diagrama, muévelo hacia dentro. Para cambiar el marco mostrado por la secuencia, haga clic en el botón  de la parte central superior.

Al igual que hicimos antes, conectaremos la entrada **Data** del objeto GPIO Write con Comando y la entrada **Address String** con Dirección. El resultado se asemejará a lo mostrado en la Ilustración 44.

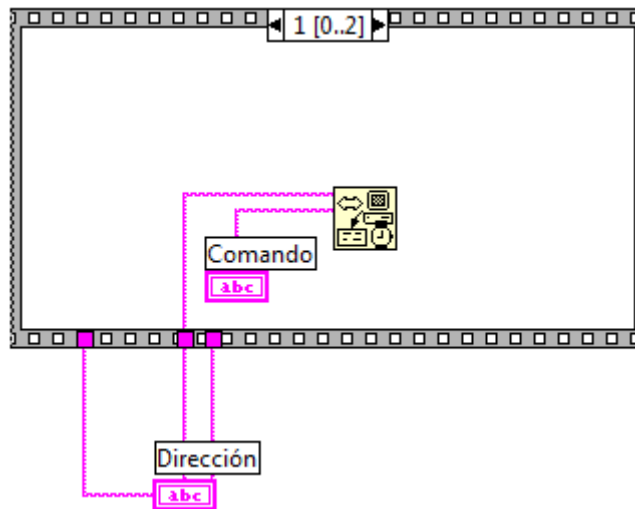





Ilustración 44

En este marco se enviara el comando guardado en el control Comando al K2700.

En el último marco (número 2) creamos un objeto **GPIO Read**  (**Functions > I/O Instruments > GPIO > GPIO Write**), un **Boolean False Constant**  (**Functions > Boolean > False Constant**), un **Fract/Exp String To Number**  (**Functions > String > String/Number Conversion > Fract/Exp String To Number**) y un **Numeric Constant** (Ilustración 45) de valor 50 que será el tamaño del buffer de lectura.

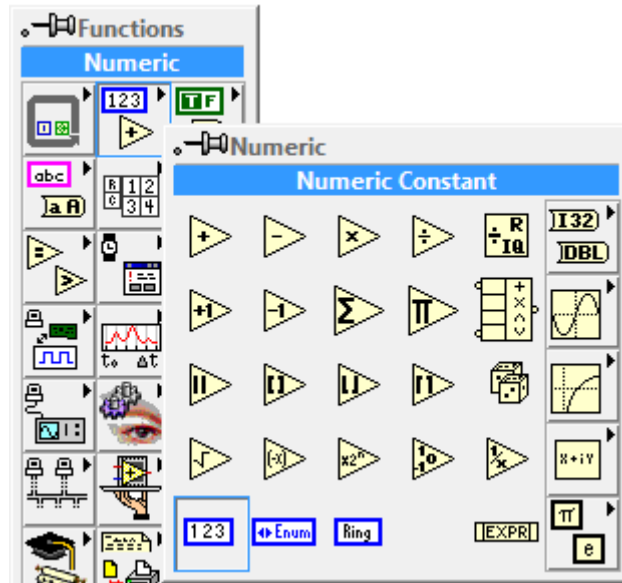




Ilustración 45

En el panel crearemos un **Error Out Cluster**  llamado **Error** (**Controls > Array & Cluster > Error Out 3D**), un objeto **String Indicator** llamado **Buffer** (**Controls > String & Path > String Indicator**), un objeto **Digital Indicator** (**Controls > Numeric > Digital Indicator**) llamado **Ohms**.

Al objeto GPIB Read le conectamos en la entrada Address String el objeto Dirección. En la entrada **Byte Count** la constante numérica 50 (para indicarle que se leerán como máximo 50 bytes). En la salida **Error Out** del GPIB Read le conectaremos el objeto **Error** (para que nos indique los errores que se han producido en el K2700, si es que se ha producido alguno) y en la salida **Data** conectamos el objeto **Buffer** (que mostrara todo el contenido del buffer de salida del K2700).



En el objeto **Fract/Exp String To Number**  conectamos en la entrada **String** al objeto **Buffer** (para convertir el dato a un tipo numérica), en la entrada **Use System Decimal Point** la constante **False** (para que utilice el carácter “.” en lugar de la coma “,” como separador decimal) y en la salida **Offset Past Number** el objeto **Ohms** (para que muestre el valor numérico en un formato entendible por nosotros).

Nos debería resultar un diagrama como el presentado en la Ilustración 46.

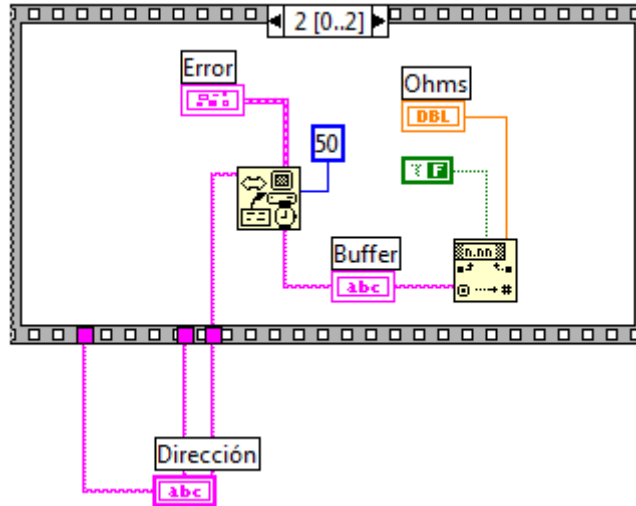



Ilustración 46

Ahora solo falta dale el valor 16 a Dirección y rellenar las sentencias necesarias en comando. En este caso, las necesarias para leer los Ω de una LDR.

```
FUNC 'RES'
ROUT:CLOS (@101)
READ?
```

Para que los valores introducidos se almacenen permanentemente en el programa, debemos seleccionar la opción **Make Current Values Default** del menú **Operate**.

Pulsamos **Run**  y debemos obtener unos resultados similares los presentados en la Ilustración 47.






Error		Dirección	Ohms
status	code	16	404,54
	d0	Comando	Buffer
source		FUNC 'RES' ROUT:CLOS (@101) READ?	+4.04535248E+02OH M,+905.496SECS, +00396RDNG#

Ilustración 47

En caso de que se produjese un error en la lectura, el **Error Out Cluster** nos indicaría la el código del error.

Programa 3. Escritura a disco.

El programa 3 es una variante del programa 2. En los marcos 0 y 1 no se realizará ninguna modificación.

En el marco 2 añadiremos los siguientes controles: un objeto **Open/Create/Replace File** , un **Write File** , un **Close File**  (menú **Functions > File I/O**); un **Numeric Constant** con valor 2, un **Number To Fractional String**  (**Functions > String > String/Number Conversion**).

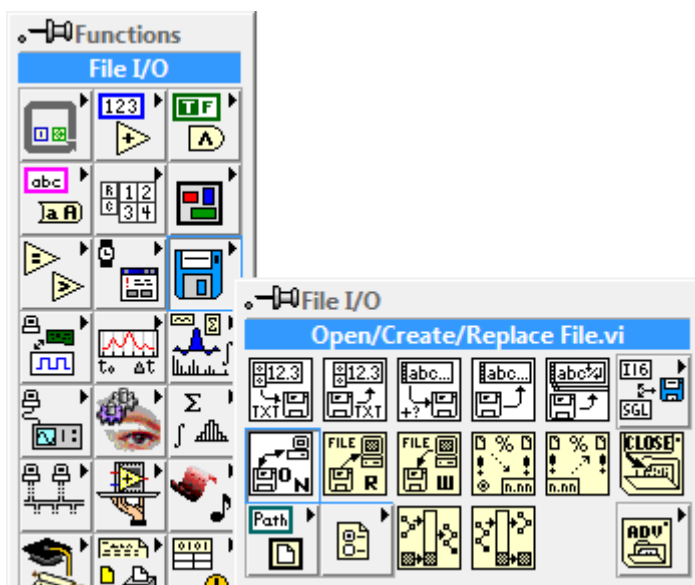




Ilustración 48: Menú File I/O

En el Panel añadiremos un **File Path Control**  al que llamaremos **Fichero** (**Controls > String & Path > File Path Control**) y un **Cluster Error Out**  al que llamaremos **Error de Fichero**.

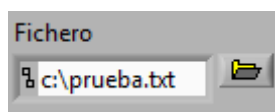


Ilustración 49: Ejemplo del File Path Control

Conectamos la entrada File Path de **Fichero** al **Open/Create/Replace File** y la constante numérica 2 a la entrada **Function** de éste. El modo 2 muestra un cuadro de diálogo en el que podremos seleccionar la ubicación del archivo y qué nombre quedemos darle. Si además el fichero ya existe, nos permitirá indicar si queremos reemplazar los contenidos del archivo existente.

En el objeto **Number To Fractional String** conectamos en la entrada **Number** al objeto **Ohms** que es el que almacena el valor de la medida. Este objeto convertirá el tipo numérico a tipo cadena para poder ser escrita en el archivo.

Ahora vamos a proceder a conectar la salida de errores de fichero. Los errores de fichero se pueden producir en cada una de las etapas (Abrir, Escribir, Cerrar). Por ello, debemos conectarlas en cascada para que se pueda propagar, es decir, la salida **Error Out** de Open/Create/Replace File se conectará a la entrada **Error In** de Write File. La salida **Error Out** de Write File se conectará a la entrada **Error In** de Close File. Por último, la salida Error Out de Close File se conectará al Error Out Cluster **Fichero**. Si se produjese un error durante alguna de esas fases, sería este control quien nos indicaría de ese error.

Al igual que hiciéramos con el Error, debemos propagar por cada uno de los objetos que trabajan con el fichero el **refnum**. El número de referencia de un fichero abierto es un valor unívoco que identifica a cada fichero abierto. Así la salida **dup refnum** de Open/Create/Replace File deberá ser conectada con la entrada del **refnum** del siguiente objeto. Ver Ilustración 50 para tener una idea gráfica.

Lo único que nos queda por conectar es la salida **F-format string** a la entrada **Data** de Write File.

Si hemos seguido todos los pasos correctamente, las conexiones deberían quedar como se ve en la Ilustración 50.

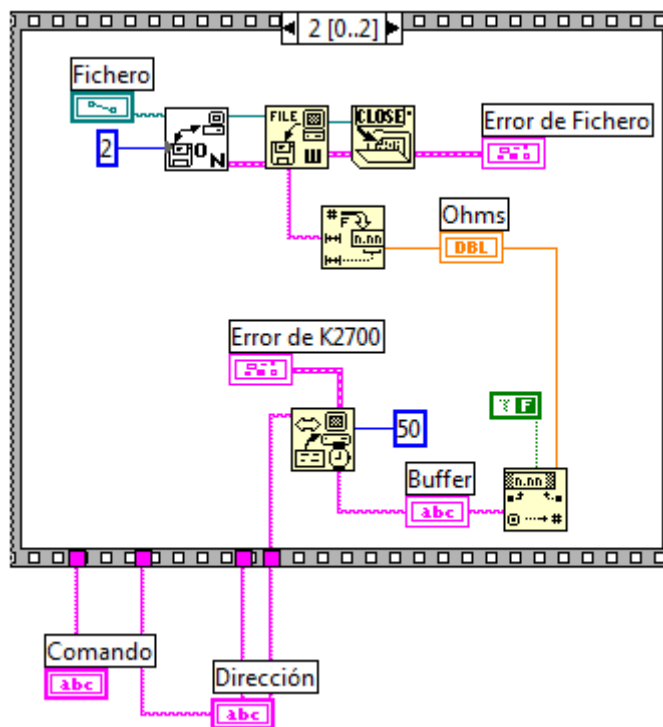


Ilustración 50: Conexiones de fichero

Si todo ha transcurrido correctamente al ejecutar, en el Panel, deberíamos tener algo parecido a lo mostrado en la Ilustración 51.

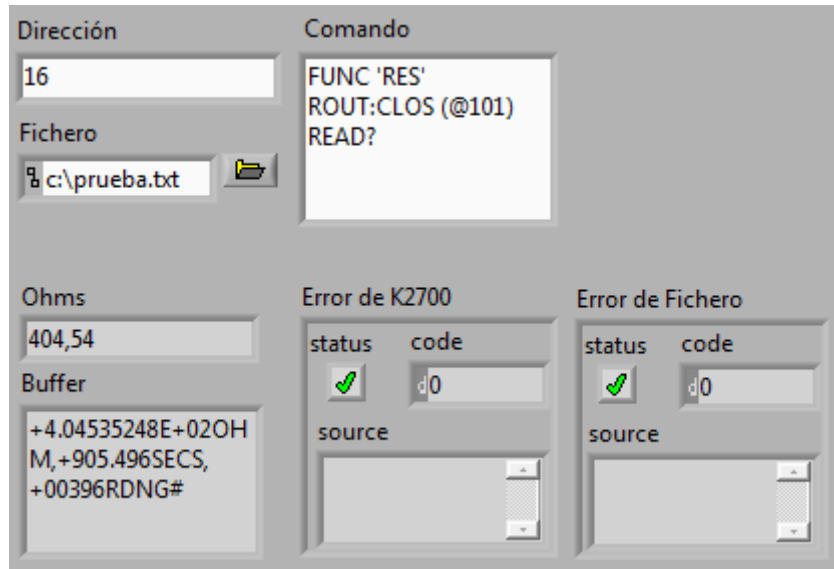


Ilustración 51: Ejecución satisfactoria del Programa 3

Programa 4. Medición simultánea temporizada con volcado a fichero.

Finalmente, en el programa cuatro, mediremos simultáneamente todos los dispositivos (e indicaremos el tiempo de cada medida). También haremos uso de los Arrays de datos y de Controles de LabView, así como uso del control Tabla (Table).

Creación de un Array de controles

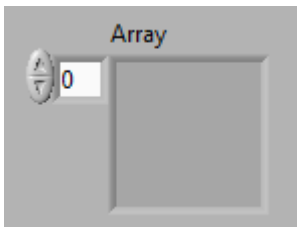



Ilustración 52: apariencia del Array en el Panel

Para crear un Array de controles lo primero que debemos hacer es crear un objeto Array en el Panel (**Controls > Array & Cluster > Array**). En el diagrama debería salirnos algo como esto  y en el Panel tendremos algo similar a la Ilustración 52.

Ahora debemos indicar de qué tipo será el Array. Dentro del recuadro más oscuro bajo la etiqueta, crearemos un elemento del tipo que necesitemos. En nuestro caso, utilizaremos un String Control. Además, renombraremos el control a **Comandos**.

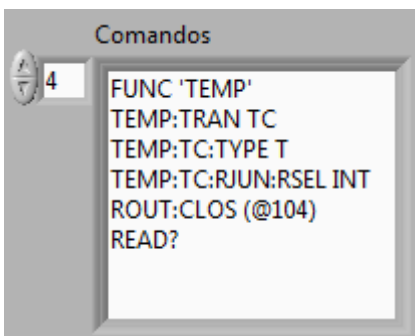




Ilustración 53: elemento del Array

Cambie el tamaño del String Control con la herramienta **Select** . Si se sitúa el puntero sobre alguna de las esquinas del control, aparecerán las esquinas marcadas. Pinche y arrastrar hasta obtener un tamaño adecuado.




Para cambiar de un elemento a otro del Array, utilizaremos la herramienta **Operate Value**  que nos permitirá movernos por el Array. Ahora escribiremos

cada elemento del Array con los comandos necesarios para realizar una medida de cada tipo. Los comandos se encuentran en el [Anexo B]. Debe quedar como algo parecido al Array de la Ilustración 53.

Bucle For Loop

Antes de nada, indicaremos lo siguiente: vamos a trabajar con una secuencia igual que la del Programa 2. Lectura del K2700. Por ello, aquí obviaremos cómo se construye. Lo único que indicaremos es que el control **Ohms**, pasa a llamarse **Lectura** y se encontrará fuera de la secuencia. Así como que el String Control Comando desaparece para dar paso al Array de String Control **Comandos**.

Ahora, insertaremos un **For Loop** (**Functions > Structures > For Loop**). Crearemos una Numeric Constant a la que le daremos el valor 5 (pues medimos 5 sensores) y la conectaremos con la **N**  del For Loop. N es el límite superior y nunca se alcanza. Las iteraciones van de 0 a N-1.

Llega el momento de conectar el array **Comandos** con la estructura. Dato que los tipos de dato no coinciden, utilizaremos una función que accede a uno y sólo uno de los elementos de un

array: **Index Array** (**Functions > Array > Index Array**).  Esta función accede a un elemento

de un array. Tiene dos entradas: **Index** e **Input Array**. Conectaremos la **i** del For Loop a **Index** y el Array **Comandos** a Input Array. La salida será lo que conectemos a la entrada Data del **GPIO Write** del marco 1 obteniendo un resultado como el de la Ilustración 54.

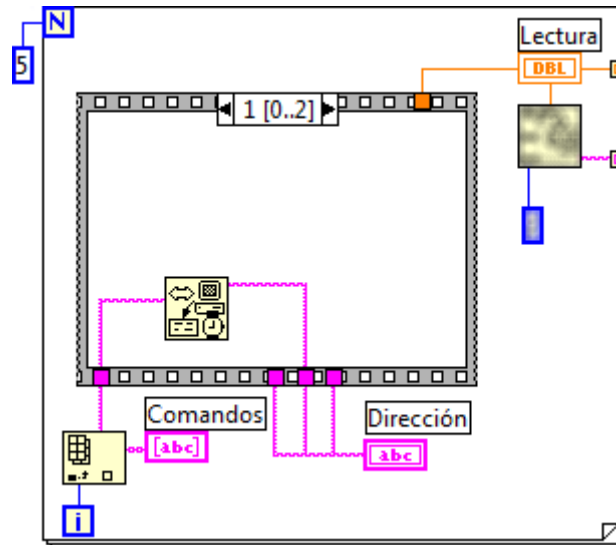


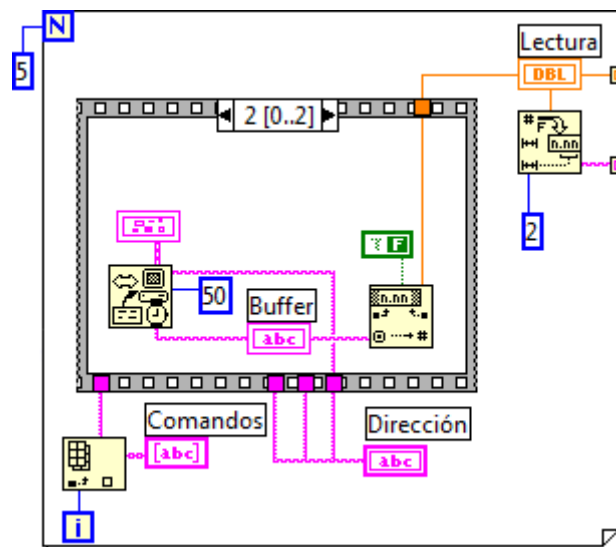
Ilustración 54: acceso al Array de comandos

Nota: obsérvese que Lectura permanece fuera de la secuencia.

Array de medidas

Si sacamos un cable desde lectura hacia fuera del For Loop, observaremos una pequeña marca en el borde. Esto quiere decir que automáticamente la lectura que se hace es a un array de medidas con 5 elementos, por lo que nosotros no necesitamos crear el array resultante.

Por otra parte, dado que vamos a rellenar una Tabla en este programa, necesitaremos crear una copia del valor como Texto. Lo cual ya sabemos realizar según vimos en el programa anterior.



Creación de la tabla de medidas (array 2D)

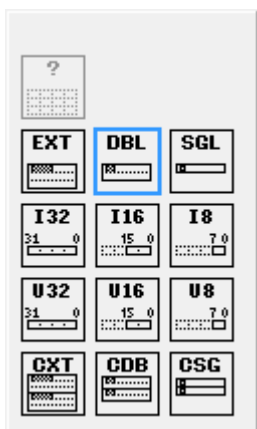
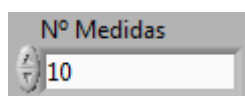


Ilustración 55: representation

Antes de proseguir, debemos especificar el número de mediciones que se realizará a los sensores. Para ello vamos a añadir un **Digital Control (Controls > Numeric > Digital Control)** al que llamaremos **Nº Medidas**.


Por defecto, el tipo numérico será **DBL** (Decimal), sin embargo, nosotros necesitamos que sea un entero. En el menú contextual del control, en la opción **Representation** encontraremos todos los tipos numéricos disponibles. En nuestro caso, dado que el número de medidas será un número mayor que 0 y natural, seleccionaremos alguno de los tipos U. En concreto **U16**, ya que el rango de U8 es de 0 a 255 y nos gustaría disponer de mayor rango (de 0 a 65535). Nuestro control debería tener la siguiente apariencia.




Hecho esto, ahora crearemos un nuevo **For Loop**. Conectaremos **Nº Medidas** con N. Pasemos a ver cómo calcular el tiempo transcurrido.

Calculo de tiempo transcurrido

En este apartado vamos a calcular el tiempo transcurrido entre cada medida y el tiempo inicial.

Para ello vamos a utilizar el objeto **Get Date/Time In Seconds**  (Functions > Time & Dialog > Get Date/Time In Seconds) que devuelve los segundos transcurridos desde el Viernes, 1 de Enero de 1904 a las 0.00 UTC.

Con dos de estos objetos colocados oportunamente y aplicando una resta  podemos obtener el número de segundos transcurridos entre dos lecturas como se muestra en la Ilustración 56.

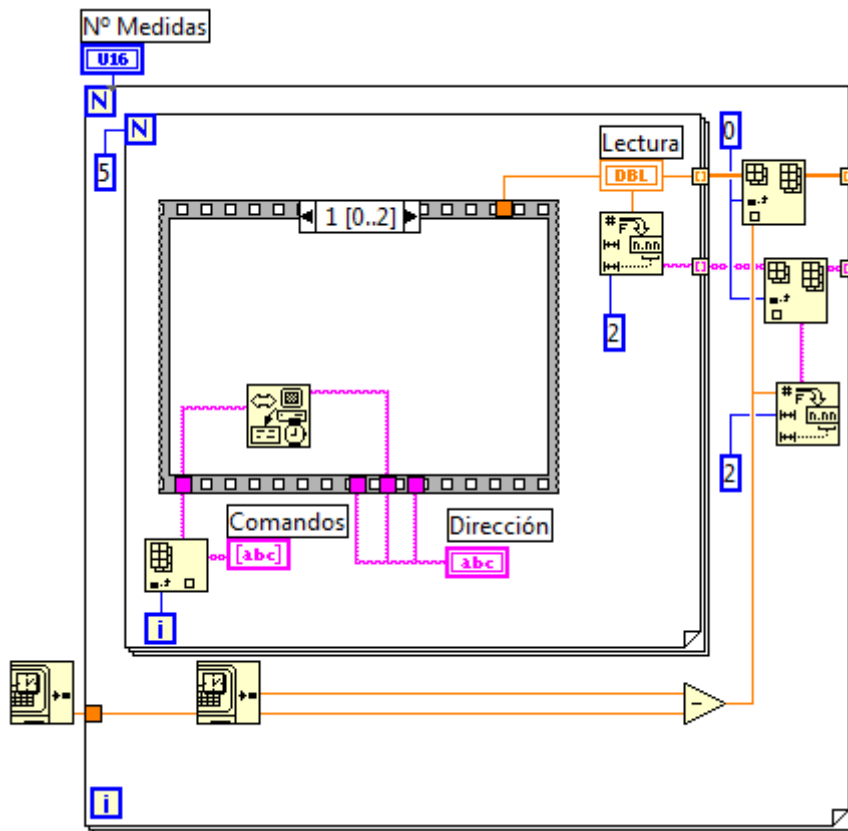



Ilustración 56: medida de tiempo

Modificación del Array de medidas

Dado que por cuestiones estéticas y por facilitar la lectura vamos a poner el tiempo transcurrido en la primera columna de la tabla, debemos modificar el array obtenido en el primer For Loop (el encargado de las medidas). Vamos a explicar cómo se consigue.

Lo único que debemos utilizar es un objeto **Insert To Array**  (Functions > Array > Insert To Array). Conectaremos el Array saliente del bucle For Loop interno a la entrada **Input Array** y una constante numérica de valor 0 a la entrada **Index**. Recordemos que los Arrays en LabView comienzan en 0. Después de esto, conectaremos el resultado del **Subtract** (la resta anterior de los tiempos) a la entrada **New Element/Subarray**

De la misma manera procederemos con la copia de tipo texto pero convirtiendo previamente el tipo a String.

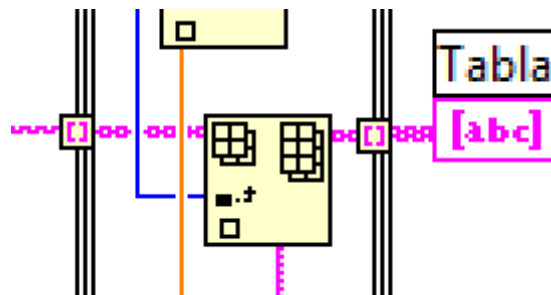
Representación en una tabla

Aunque LabView ya trae un control Tabla por defecto, tendremos que modificarlo para que se ajuste a nuestras necesidades. Lo primero será hacer visible las cabeceras de columna (**Menú Contextual > Visible Items > Column Headers**). Acto seguido, convertiremos el Control Tabla a Indicador (**Menú Contextual > Change To Indicator**). Finalmente escribiremos en las cabeceras de columna lo que representan tal como indica la Ilustración 57.

T	PT100	LDR	PTC	NTC	TC
0,00	23,77	526,29	105,81	22,79	23,95
10,64	23,79	526,71	107,21	22,90	24,12
21,08	23,79	527,63	105,95	22,71	23,81
31,64	23,81	527,28	105,32	22,52	23,56
42,09	23,81	527,47	105,25	22,41	23,33
52,53	23,81	527,33	105,00	22,30	23,12


Ilustración 57: cabeceras de columna


Ahora sólo nos queda unir la salida de bucle For Loop externo. Obsérvese que el grosor de los hilos conectores crece a medida que salimos del Array.



Formateo de Array 2D en texto

Finalmente, solo nos resta escribir el fichero con los datos obtenidos. Para poder formatear el contenido de un array de dos dimensiones como una tabla de texto debemos utilizar el objeto

Array To Spreadsheet  que está disponible en el menú Array.

En la entrada **delimiter** estableceremos cómo que carácter (o conjunto de estos) se utilizará como delimitante. Nosotros hemos utilizado el tabulador  que podemos encontrar en las constantes String. En la entrada **Format String** conectamos una String Constant con el siguiente contenido: “%.2f”. En LabView se utiliza el mismo tipo de formato que en C. Por tanto, hemos seleccionado redondear los decimales a centésimas. Finalmente, en la entrada **Array** enlazaremos el Array que sale del For Loop exterior.

Por último, sólo nos queda escribir en fichero la salida de Array To Spreadsheet. Pero antes vamos a añadir un encabezado. Crearemos una String Constant, haremos clic derecho en el mismo y seleccionaremos la opción **\ Codes Display** y luego escribiremos “T\tPT100\tLDR\tPTC\tNTC\tTC\r\n” que después escribiremos al comienzo del fichero. A continuación

Ahora sólo nos queda enlazar los objetos necesarios para la escritura de fichero como hiciéramos en el programa anterior.

Si hemos seguido todos los pasos, nuestro Diagrama y nuestro Panel después de una ejecución deberían ser como los siguientes.

Conclusiones

En este apartado realizaremos una lista de Pros y Contras sobre los programas con los que hemos realizado estas prácticas y la conclusión a la que llegamos una vez analizadas estas listas.

Xlinx

Pros

- Configuración rápida del programa.
- No se necesita programar para tomar medidas.
- Realiza gráficas automáticamente.
- Permite guardar los datos en hojas de cálculo.

Contras

- Posibilidades limitadas tales como realizar tablas, guardar el tiempo de las medidas o cambiar el formato de los ficheros.
- Requiere atención constante del usuario.

Conclusión

Es el mejor programa para realizar unas cuantas medidas rápidas y para probar que los dispositivos estén conectados correctamente.

HP VEE

Pros

- Facilidad de escritura en archivo (Transformación simple de tipos de datos e introducción de tabuladores y saltos de línea configurable)

Contras

- Poca usabilidad.
- Aspecto grafico paupérrimo.

Conclusión

No ha estado a la altura de las expectativas.

TestPoint

Pros

- Fácil de programar por su parecido a los lenguajes de programación tradicionales.

- Gran cantidad de opciones graficas que ayudan a realizar programas más agradables a la vista.

Contras

- Dificultad de escritura en los campos de la lista de acciones.
- Carece de redondeo real en los objetos (necesidad de redondear con formulas).

Conclusión

TestPoint es el programa más sencillo para crear programas completos.

LabView

Pros

- Opciones casi ilimitadas (capaz de realizar todo tipo de graficas, operaciones...).
- Muy utilizado, lo que aumenta el número de guías y documentación disponible en internet.

Contras

- Manejo poco intuitivos (cables que salen de bucle For Loop se convierten en Arrays).
- Exceso de operadores (son demasiado concretos, por lo que hay que buscar el que encaje exactamente para cada operación).

Conclusión

LabView tiene más potencial que el resto de programas, pero el uso es menos intuitivo.

Anexo A: Montaje de la tarjeta de A/D (Modelo 7700).

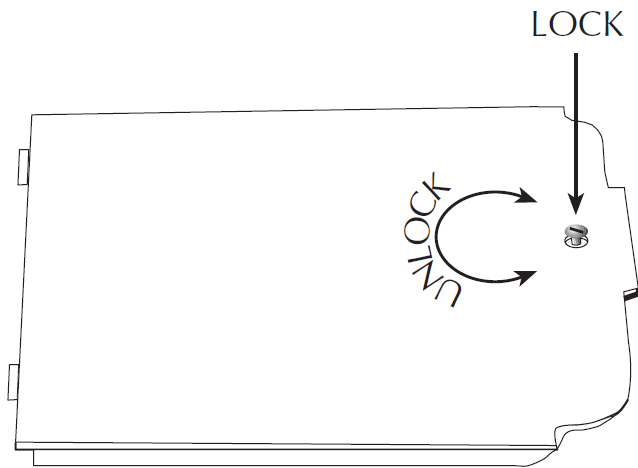


Ilustración 58: montaje y desmontaje del tornillo de la tarjeta de adquisición de datos.

Lo primero para montar la tarjeta de adquisición de datos será abrirla como indica el dibujo (Ilustración 58: montaje y desmontaje del tornillo de la tarjeta de adquisición de datos.). Gire el tornillo hasta que pueda levantar la tapa superior.

Al desmontar la tarjeta encontrará una serie de conectores en los que deberá enganchar las distintas conexiones de los sensores que se

utilizarán para la toma de mediciones (Ilustración 59).

Introduzca los cables por los huecos designados a tal efecto. Se encuentran dos huecos: uno para los conectores de la parte superior y otro para los conectores de la parte inferior. Para el caso de la tarjeta que nosotros utilizaremos a lo largo de las cuatro prácticas (El modelo 7700), tendremos 20 canales que podremos utilizar de la siguiente manera: como 20 canales de 2 terminales o bien como 10 canales de 4 terminales.

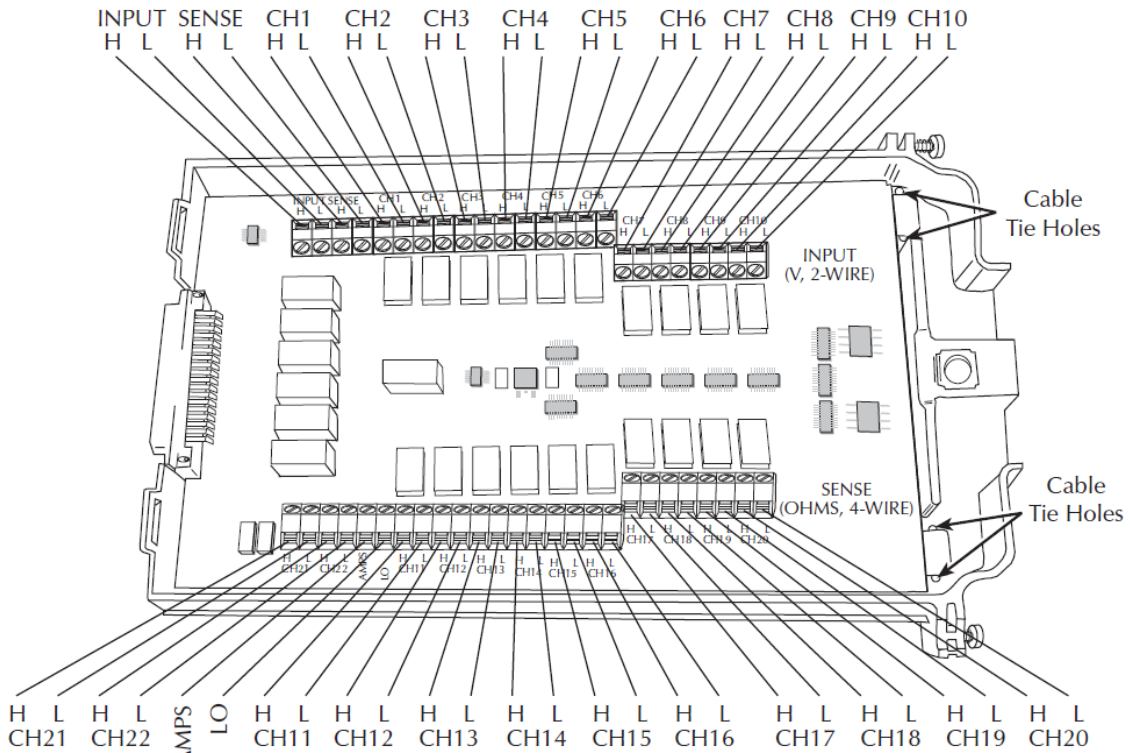


Ilustración 59

En la Ilustración 60 podemos ver al detalle cómo son los terminales. Cuando conectemos un sensor debemos tener precaución con la polaridad (ya que estamos trabajando con sensores semiconductores en la mayoría de los casos). De equivocarnos, podremos obtener medidas erróneas o incluso dañar el dispositivo sensor.

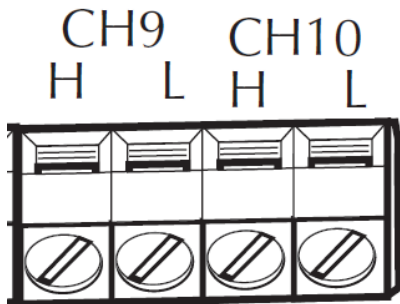


Ilustración 60: detalle de los terminales de cada canal.

Dicho esto, conectaremos nuestros sensores según la siguiente Tabla 1:

Canal	Color	Descripción
Ch01 (@101)	H Rojo	LDR
	L Negro	"
Ch02 (@102)	H Rojo	PTC
	L Negro	"
Ch03 (@103)	H Rojo	NTC
	L Negro	"
Ch04 (@104)	H Marrón	TC (Tipo T)
	L Blanco	"
Ch05 (@105)	H Rojo	PT100
	L Negro	"
Ch15 (@115)	H Rojo	PT100 (Fuente de Intensidad).
	L Negro	"

Tabla 1: conexiones

Salvo la PT100, los demás dispositivos son de dos terminales y no debería existir ninguna complicación para conectarlos al módulo de adquisición de datos. Para el caso de la PT100, conectaremos como se indica en la Ilustración 61.

Si por ejemplo, medimos la PT100 en canal Ch05, los terminales de intensidad deberían conectarse con un offset de +10, es decir, en el Ch15. El offset depende del número de canales, para el caso del modelo 7700 es 10.

Consideraciones

El termopar ha sido conectado directamente al módulo de adquisición de datos ya que el modelo 7700 posee un termómetro interno.

Las mediciones del valor de la PTC son en Ω (Ohm). Dado que el Keithley 2700 no viene preparado para leer temperatura de este sensor.

La tarjeta de adquisición de datos está insertada en el Slot 1.

Ω 4-Wire and RTD connections

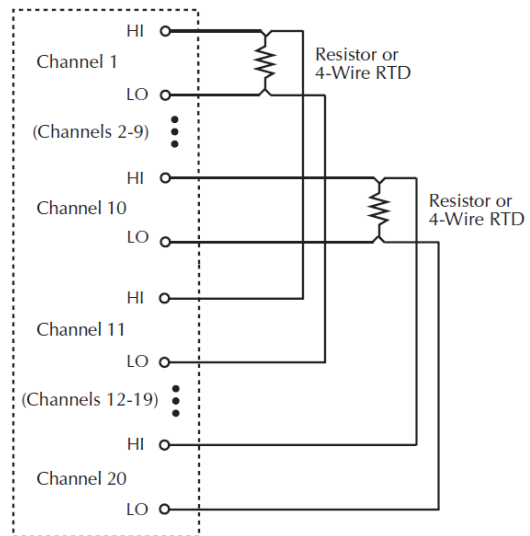


Ilustración 61: conexión para elementos de cuatro terminales.

Anexo B: Comandos del K2700

A lo largo de toda esta práctica, se han utilizado una serie de comandos del K2700 repetidos continuamente (ya que los comandos del K2700 son independientes del entorno usado). En cada entorno serán introducidos como sea necesario y utilizando las secuencias de escape necesarias si procede. Los canales se han dejado tal y como está montada la tarjeta de adquisición de datos.

Resetear la configuración

```
*RST
```

Lectura de LDR (o PTC) en Ω

```
FUNC 'RES'  
ROUT:CLOS (@101)  
READ?
```

Lectura de NTC en $^{\circ}\text{C}^1$

```
FUNC 'TEMP'  
TEMP:TRAN THER  
TEMP:THER 1950  
ROUT:CLOS (@103)  
READ?
```

Lectura de PT100 en $^{\circ}\text{C}$

```
FUNC 'TEMP'  
TEMP:TRAN FRTD  
TEMP:FRTD:TYPE PT100  
ROUT:CLOS (@105)  
READ?
```

Lectura de TC en $^{\circ}\text{C}$

```
FUNC 'TEMP'  
TEMP:TRAN TC  
TEMP:TC:TYPE T  
TEMP:TC:RJUN:RSEL INT  
ROUT:CLOS (@104)  
READ?
```

¹ Aunque no se especifica, el K2700 devuelve la temperatura en grados a no ser que se le indique explícitamente Fahrenheit o Kelvin.

Bibliografía

1. **MODEL 2700 MULTIMETER/SWITCH SYSTEM USER'S MANUAL**, ©2001, Keithley Instruments, Inc. Págs.: B-5 — B-9, SCPI Reference Tables.
2. **KEITHLEY 2700 XLINX MULTIMETER/DATA ACQUISITION SYSTEM HELP**. ©2001, Keithley Instruments.
3. **HPVEE HELP**. ©1998 Hewlett Packard Co.
4. **TESTPOINT HELP MANUAL**.
5. **NATIONAL INSTRUMENTS LABVIEW 6I HELP**. © 2000, National Instruments Corporation.

Contenido del CD

📁 01 – Xilinx		
📁 Medidas		
📄 Tabla 50 muestras temperatura.xls		Volcado de XLinX
📁 Programas		
📄 K2700.inx		Configuración canales
📁 02 – HPVEE		
📁 Medidas		
📄 Tabla Medidas 30 min.txt		Resultados del Programa 5
📁 Programas		
📄 P02Prog01.vee		Programa 1
📄 P02Prog02.vee		Programa 2
📄 P02Prog03.vee		Programa 3
📄 P02Prog04.vee		Programa 4
📄 P02Prog05.vee		Programa 5
📄 P02Prog05Alt.vee		Versión alternativa (guarda fila a fila)
📁 03 – TestPoint		
📁 Medidas		
📄 Tabla 14 medidas.txt		Resultados del Programa 5
📄 Tabla Resultado P03Prog04.txt		
📁 Programas		
📄 P03Prog01.tst		Programa 1
📄 P03Prog02.tst		Programa 2
📄 P03Prog03.tst		Programa 3
📄 P03Prog04.tst		Programa 4
📄 P03Prog05.tst		Programa 5
📁 04 – LabView		
📁 Medidas		
📄 Tabla 10 medidas.txt		Resultados del Programa 4
📁 Programas		
📄 P04Prog01.vi		Programa 1
📄 P04Prog02.vi		Programa 2
📄 P04Prog03.vi		Programa 3
📄 P04Prog04.vi		Programa 4
📄 Tabla.ctf		Modificación control Tabla
📁 10 – Recursos	Lea el archivo readme.txt para detalles del contenido de esta carpeta	