6

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

DISEÑO DE UN SISTEMA DE ADQUISICIÓN DE DATOS EN TIEMPO REAL PARA CONVERTIDORES ANALÓGICOS DIGITALES

T E S I S

QUE PARA OBTENER EL TÍTULO DE

INGENIERO EN COMPUTACIÓN

P R E S E N T A

MARTHA CECILIA AYALA HERNÁNDEZ

DIRECTOR: ING. GABRIEL CASTILLO HERNÁNDEZ

MÉXICO, D. F. OCTUBRE 2001

Agradecimientos

A Dios

Por haberme permitido llegar hasta aquí

A mi madre

Por quererme, guiarme y apoyarme todo este tiempo

A mi amor

Por que contigo aprendí lo maravilloso que es compartir la vida

A mi mejor amiga

Por que contigo aprendí a ser más fuerte y me ayudaste a acercarme y conocer más a Dios

A mi familia

Por su cariño y ayuda en los momentos difíciles

A mis amigos

Por haber estado siempre a mi lado y compartir conmigo esta alegría

Al Ing. Castillo

Por haber sido un buen maestro y guía

A mis maestros

Por enseñarme y ayudarme a crecer como ser humano

TABLA DE CONTENIDO

CAPÍTULO 1 ANTECEDENTES **1**
Introducción 1
Evaluación de las necesidades 2
Hardware disponible 3
Computadora Personal (PC) *3*
PCL-812PG de Advantech C.O. *3*
AT-MIO-64F-5 de National Instruments *3*

CAPÍTULO 2 ANÁLISIS **5**
Planteamiento de los requerimientos 5
Requerimientos del front-end *5*
Requerimientos de back-end *5*
Consideraciones y expectativas en la elaboración del sistema 7
Diagrama jerárquico funcional del sistema 7
Configuración *8*
Adquisición *8*
Graficación *9*
Almacenamiento *10*
Calibración *11*
Escalamiento *11*
Investigación de las funciones asociadas a las tarjetas de adquisición soportadas por el sistema. 12
Funciones de la tarjeta PCL-812PG *12*
Funciones de Entrada Analógica *12*
Funciones de Alta Velocidad *13*
Funciones clasificadas por objetivo PCL-812PG *14*
Funciones de la tarjeta AT-MIO-64F-5 ..... *16*
Funciones clasificadas por objetivo AT-MIO-64F-5 ..... *17*
Operaciones mínimas necesarias en un convertidor genérico 20
Operaciones necesarias para cada convertidor específico 21
Interacción entre el código específico y el genérico 21

CAPÍTULO 3 DISEÑO **22**
Jerarquía de herencia 22
Diagrama de Clases 24
La Clase TConvertidor 25
Diseño de las clases para cada tarjeta, derivadas de TConvertidor 25
Plan para la adquisición de muestras y la clase TCircular 26
El método de graficación y la Clase TSmallGraph 28
Comportamiento General del Sistema 29

Módulo Principal 29
Módulo Configuración 30
Módulo de Escalamiento 30
Módulo de Graficación 31
Módulo de Calibración 31
Diagramas de Transición de Estados 33

CAPÍTULO 4 IMPLANTACIÓN 34
Programación del sistema utilizando Delphi como herramienta 34
Configuración de pantallas 35
Pantalla Principal 35
Iniciar la adquisición 36
Guardar en archivo 36
Elección de canales 37
Pantalla de configuración 38
Pantalla de Escalamiento 38
Pantalla de Graficación 39
Pantalla de Calibración 39
Formato de los archivos 41
Elaboración del archivo de ayuda 42

CAPÍTULO 5 PRUEBAS AL SISTEMA BAJO DIFERENTES CONDICIONES DE OPERACIÓN 43
Pruebas 43
Tiempos de adquisición y tamaño de los archivos 44
Carga Ligera 44
Carga Mediana 44
Alta Carga 45
Comportamiento bajo condiciones adversas 45

CONCLUSIONES 46

GLOSARIO 47

BIBLIOGRAFÍA 51
Libros 51
Sitios en Internet 51

APÉNDICE A CONVERTIDORES ANALÓGICOS-DIGITALES 52
Características básicas de los convertidores 52
Tipos de Convertidores Analógicos-Digitales 54
Convertidores A/D de rampa de escalera 54
Convertidores A/D de aproximaciones sucesivas 55
Convertidores A/D de doble rampa 55
Convertidor de voltaje a frecuencia 56
Convertidor en paralelo (o instantáneo) 57
Aplicaciones de los convertidores 58

APÉNDICE B TEORÍA ORIENTADA A OBJETOS 59
Conceptos 59
Objeto 59
Tipo de Objeto 59
Métodos 60
Encapsulado 60

Mensajes 61
Clase 61
Herencia 61
Eventos 62
Tipos de Eventos 62
Diseño de la estructura y comportamiento de un objeto. 63
Diferencia entre operación y método 63
Herencia de Clase. 64
Herencia Múltiple. 64
Selección del Método. 64
Polimorfismo. 65
Metodologías de desarrollo de software orientado a objetos 65
UML (Unified Modeling Language) 66
Enfoque General 67
Tipos de Diagramas UML 68
Diagramas de Clase 68
Diagramas de Paquete 68
Diagramas de Objeto 68
Diagramas de casos de Uso 69
Diagramas de Secuencia 69
Diagramas de colaboración 69
Diagramas de estado 70
Diagramas de actividad 70
Diagramas de componentes 70
Diagramas de plataformas o despliegue 71
OOD (Object Oriented Design) 71
Enfoque General 71
Tipos de Diagramas OOD 72
Diagrama de Clases 72
Diagramas de Objetos. 73
Diagramas de transición de estados. 73
Diagramas de Interacción 73
Diagramas de módulos. 74
Diagramas de procesos. 74
OOSE (Object-Oriented Software Engineering) 74
Enfoque general 74
Object Modeling Technique (OMT) 75
Enfoque General 75
Yourdon & Coad's Object-Oriented Analysis and Design (OOA/OOD) 76
Enfoque General 76
Fusion Method 76
Enfoque General 77

APÉNDICE C REFERENCIA DE CLASES 78
TConvertidor 78
Propiedades 78
Running 79
GainStrings 79
StartChanel 79
EndChanel 80
Gain 80

GainStr....80
GainFactor....80
MaxChanel....81
Frec....81
MaxFrec....81
Métodos....82
SetStartChanel....82
SetEndChanel....83
SetGain....83
SetGainStr....83
GetGainStr....84
GetIndexGain....84
SetFrec....84
StartHardware....84
StopHardware....85
Create....85
Destroy....85
Start....86
Stop....86
Reset....86
GetGainStrings....86
Volts....87
Scaled....87
Sample....87
nSample....87
VSample....88
nVSample....88
sSample....88
TConvAT_MIO_64F5....89
Métodos....89
StartHardware....89
StopHardware....89
Create....90
Destroy....90
GetGainStrings....90
Volts....90
Scaled....91
TConvPCL_812PG....92
Métodos....92
StartHardware....92
StopHardware....93
Create....93
Destroy....94
GetGainStrings....94
Volts....94
Scaled....94
TCircular....95
Propiedades....95
Distancia....95
Tail....95
Head....95

Vuelta....96
Overflow....96
Métodos....*96*
Create....96
Destroy....97
Reset....97
Add....97
Get....98
nGet....98
TGetEvent....99
Execute....99
AdOverrunEvent....99
AdHalfReady....100
Create....101
TSmallGraph....102
Propiedades....*102*
Serie1Color....102
Serie2Color....102
Serie3Color....103
Serie4Color....103
GridColor....103
AxisColor....103
Métodos....*103*
SetSerie1Color....104
SetSerie2Color....104
SetSerie3Color....104
SetSerie4Color....105
SetGridColor....105
SetAxisColor....105
Create....105
Destroy....106
Prepare....106
PlanoOnPaint....106
PlotPoint....106
PlotPoint1....107
PlotPoint2....107
PlotPoint3....107
PlotPoint4....108
Scroll....108
Clean....108

CAPÍTULO 1 ANTECEDENTES

Introducción

En el Laboratorio de Hidromecánica del Instituto de Ingeniería era necesario contar con un sistema que pudiera interactuar con tarjetas de conversión analógicas-digitales para registrar las diferentes variables físicas que intervienen en los experimentos ahí realizados. Por ejemplo, se realizan mediciones de presión hidráulica, gasto y velocidad de giro en la descarga de bombas que sufren fallas en el suministro de energía. Las variables físicas se registran mediante transductores, estos dispositivos convierten cada una de las variables físicas en los niveles lógicos de tensión, dichos valores entran al convertidor analógico digital y a la salida son valores binarios que pueden utilizarse para un registro y análisis posteriores.

La información binaria proporcionada por la tarjeta de adquisición debe ser codificada nuevamente para presentar los valores correspondientes de tensión de entrada, una aplicación apropiada debe realizar dicha transformación.

La presente tesis expone el diseño y desarrollo de un sistema que registre y presente dichas muestras mediante la computadora, usando archivos para almacenar la información y graficando los resultados en pantalla, debido a que en el laboratorio no se utiliza una tarjeta específica, se requiere que este sistema sea capaz de escalarse de manera sencilla para utilizarse con diversos tipos de tarjetas.

Para una mejor visión del sistema se plantearan los puntos más importantes en cuanto a sus requisitos, una vez establecidos se procederá a encontrar la mejor manera de llevarlos a cabo.

Evaluación de las necesidades

Se requiere de un sistema que sea capaz de proporcionar información confiable y veraz en la adquisición de muestras durante fenómenos y pruebas hidráulicas para poder analizarlos. Esto ayudará a la toma de decisiones y mejoramiento de la operación del sistema que está sometido a estudio y pruebas.

El sistema deberá cumplir con las siguientes expectativas:

- **Sencillez y Transparencia**

  El uso debe ser lo más transparente posible para el usuario, contando con opciones claras para su operación. El sistema debe de operar con el mínimo de acciones, ya que debido a la naturaleza de los experimentos, éstos deben realizarse periódicamente, eliminando la posibilidad de un proceso complicado y tardado en la adquisición de muestras.

- **Flexibilidad en su operación**

  El sistema deberá de permitir al usuario muestrear, almacenar y generar gráficas en diferentes formatos. Así el usuario podrá almacenar sus archivos en modo texto o en formato xls, graficar la información de una o varias señales usando escalas lineales o logarítmicas, presentar los datos en voltaje o transformados a presiones, gastos, etc.

- **Escalabilidad**

  La meta propuesta consiste en utilizar la misma interfaz para varias tarjetas de adquisición de datos disponibles en el mercado, apoyándose en los archivos VxD *(virtual device drivers)* que vienen incluidos en la distribución de las tarjetas, dichos archivos permiten al programador establecer un mecanismo de adquisición conveniente para sus necesidades.

- **Especificaciones adicionales**

  Los datos deben ser guardados en un archivo y visualizados gráficamente en tiempo real. También se desea que el sistema tenga características adicionales tales como escalamiento y calibración de respuestas.

Hardware disponible

COMPUTADORA PERSONAL (PC)

Con las siguientes características mínimas:
Procesador: Intel Pentium
Memoria: 32 MB en RAM
Espacio en disco duro: 1 GB

Las tarjetas soportadas por el sistema se presentan a continuación.

PCL-812PG DE ADVANTECH C.O.

La PCL-812PG es una tarjeta de adquisición de datos con alto desempeño, velocidad y multifunction para computadoras IBM PC/XT/AT compatibles. Es utilizada para una gran variedad de aplicaciones en ambientes industriales y de investigación en laboratorios.

Características

- 16 canales de entrada analógica.
- Convertidor industrial estándar de aproximaciones sucesivas de 12 bits (HADC574Z) para la conversión de entradas analógicas. La frecuencia máxima de muestreo en modo DMA (Direct Access Memory) es de 30 KHz.
- Rangos de entrada analógica programables por software:
  Bipolar : +/- 5V, +/- 2.5V, +/- 1.25V, +/- 0.625V, +/- 0.3125V.
- Tres modos de disparo A/D:
  - *Por Software.*
  - *Programmable pacer trigger.*
  - *Pulso Externo.*

AT-MIO-64F-5 DE NATIONAL INSTRUMENTS

La AT-MIO-64F-5 es una tarjeta de adquisición de datos con alto desempeño y multifunción analógica y reloj I/O para computadoras personales.

Características

- 64 canales de entrada analógica.

- Resolución analógica de 12 bits, de 1 a 4096. Frecuencia máxima de muestreo de 200 kHz.
- Impedancia de entrada de100 G. en paralelo con 100 pF
- Rangos de +/-50mV, +/-100mV, +/-0.25V, +/-0.5V, +/-1V, +/-2.5V, +/-5V y +/-10V, seleccionadas por software.

	PCL-812PG	AT-MIO-64F-5
Número de Canales	16	64
Frecuencia máxima de muestreo	30 KHz	200 KHz
Rangos	+/- 5[V], +/- 2.5[V], +/- 1.25[V], +/- 0.625[V] y +/- 0.3125[V]	+/-50[mV], +/-100[mV], +/-0.25[V], +/-0.5[V], +/-1[V], +/-2.5[V], +/-5[V] y +/-10[V]

Tabla 1.1 Tabla comparativa de las tarjetas de adquisición

CAPÍTULO 2 ANÁLISIS

Planteamiento de los requerimientos

La meta consiste en crear un sistema que sea capaz de soportar distintos tipos de tarjetas de adquisición bajo la misma interfaz, el sistema tendrá la configuración siguiente:

REQUERIMIENTOS DEL FRONT-END

Se debe desarrollar una interfaz gráfica basada en las características generales de una tarjeta de conversión. Esta interfaz deberá ofrecer al usuario la posibilidad de elegir las opciones de conversión tales como número de canales a muestrear, frecuencia y ganancia de muestreo. El usuario podrá ver en pantalla la graficación en tiempo real de los valores de voltajes, presiones, gastos, etc en función del tiempo.

Otro aspecto importante es el concerniente almacenamiento de muestras, debe ofrecerse la opción de salvar los resultados de la adquisición en un archivo de datos.

También debe contarse con una opción de escalamiento de señales, dicha función se refiere a la posibilidad de representar la señal en las unidades de la variable física medida.

La calibración del transductor antes de un experimento es otro aspecto importante, que debe contemplarse en el desarrollo del proyecto.

REQUERIMIENTOS DE BACK-END

Por otro lado el sistema debe tener una estrategia interna en donde cada tarjeta de conversión de datos deberá ser manipulada dependiendo de sus propias funciones y

secuencias correspondientes a la adquisición multicanal. Ya que cada una es diferente, el manejo debe contemplar sus características generales e individuales. Por lo que se hace necesario contar con una capa que conecte la interfaz gráfica con las funciones provenientes de cada tarjeta las cuales son proporcionadas como DLL's por cada fabricante. El diagrama muestra como los archivos DLL's son obtenidos mediante VxDs (virtual device drivers). Los *virtual device drivers* son archivos que sirven esencialmente para correr sin problemas en Windows. La razón por la que Windows utiliza los *virtual device drivers* descansa en la función primaria de Windows, ahora casi olvidada: corriendo varias aplicaciones a la vez o mejor dicho haciendo capaz al usuario de intercambiar rápidamente entre varias aplicaciones. Sin *virtual device drivers*, las aplicaciones pueden causar problemas cuando se quiera usar un recurso de sistema que ya esta siendo utilizado por otra aplicación. Los VxDs manejan el uso de estos recursos (principalmente en dispositivos de hardware)

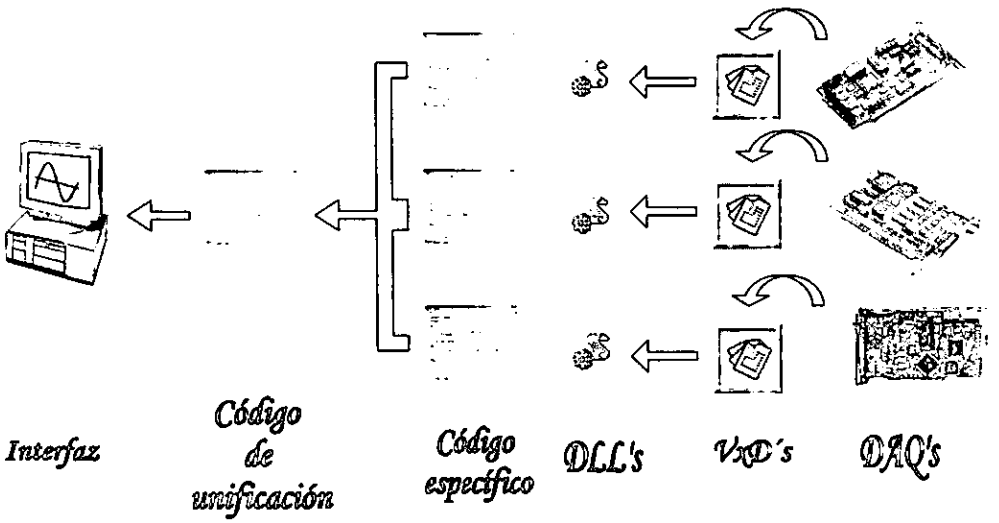


Figura 2.1 Esquema inicial del sistema

En el denominado código de unificación se contempla incluir las características comunes entre todos los convertidores mientras que en el código específico se encuentran las características individuales.

Consideraciones y expectativas en la elaboración del sistema

Es importante puntualizar las siguientes condiciones de operación:

- ✓ Una vez iniciada la adquisición de datos no es posible modificar las opciones, ni elegir un archivo para guardar hasta que dicha adquisición concluya.
- ✓ Mientras se estén adquiriendo muestras no puede ser cerrada la aplicación, para llevar un manejo adecuado de las tarjetas de conversión.
- ✓ La frecuencia que elija el usuario no debe rebasar la frecuencia máxima soportada por la tarjeta y el sistema en conjunto. (La de la tarjeta se ve reducida debido al manejo del archivo donde se guarda las muestras, la graficación y presentación de las mismas)
- ✓ Si la tarjeta no se encuentra presente en el sistema se debe enviar un mensaje de error, de otra forma el usuario no sabrá si la tarjeta se encuentra presente o no.
- ✓ El sistema debe ofrecer la posibilidad de recuperar las muestras nuevamente en modo gráfico, una vez que ha sido llevado a cabo el experimento.
- ✓ Las muestras deberán guardarse en el archivo a medida que van siendo adquiridas esto con el fin de no perder los datos del experimento ante una posible falla de energía.

Diagrama jerárquico funcional del sistema

El diagrama general nos permite conocer los módulos principales del sistema y dentro de cada uno de éstos se definen las distintas operaciones que son necesarias para dicho módulo.

Figura 2.2 Módulos principales del sistema

CONFIGURACIÓN

Este módulo permitirá al usuario elegir las opciones de conversión, tales como:

Número de canales.- esta opción deberá presentar una lista de selección con el número de canales a muestrear desde el primero hasta el último canal disponible para cada tarjeta.

Factor de ganancia .- el factor de ganancia se refiere a la resolución que tiene la conversión.

Frecuencia de muestreo .- medida en Hz., es el número de muestras que son adquiridas por cada segundo para cada canal.

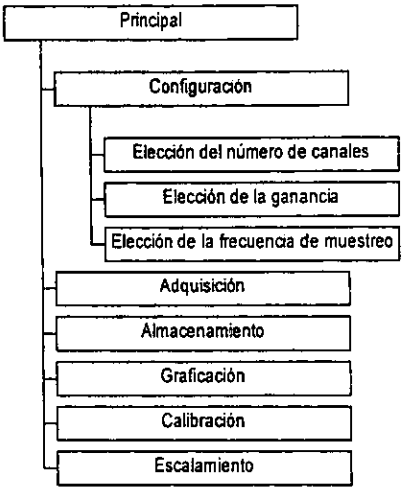


Figura 2.3 Configuración

ADQUISICIÓN

La adquisición deberá estar organizada de manera que las muestras no se pierdan durante la transición de éstas desde el buffer de la tarjeta de adquisición hasta el despliegue y almacenamiento de información.

Para el inicio de adquisición muestras se debe contar con un botón de **Start/Stop** de esta forma con el mismo botón podrá iniciar y terminar un experimento.

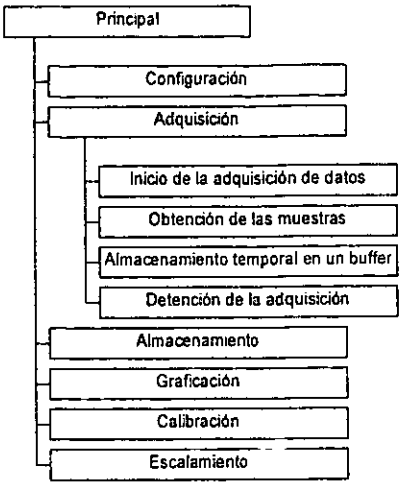


Figura 2.4 Adquisición

GRAFICACIÓN

El módulo de graficación deberá presentar las muestras que están siendo adquiridas, podrán ser visualizados hasta cuatro canales en tiempo real. El usuario podrá ver en pantalla la graficación de los valores voltajes en función del tiempo. Para llevar un monitoreo el sistema debe ser capaz de presentar el valor del tiempo transcurrido y de manera simultánea los valores correspondientes a los canales muestreados. Para facilitar la visualización dependiendo del usuario debe permitirse la selección del color de trazo para cada canal. Se debe contar con un botón de inicio y fin de adquisición de datos.

En principio se encontrarán disponibles solamente aquellos canales que fueron seleccionados en la etapa de configuración. Y mediante un mecanismo de selección podrá escogerse para ser visualizado en la pantalla, esto es podrán estarse adquiriendo las muestras para dicho canal y su visualización será opcional.

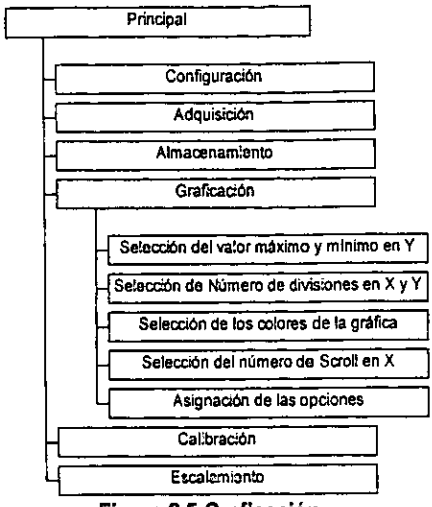


Figura 2.5 Graficación

ALMACENAMIENTO

Este módulo será el encargado de salvar los resultados de la adquisición en un archivo de datos. El usuario puede especificar el nombre del archivo, y el formato del archivo puede ser de texto o en formato Excel (.xls) organizado por columnas de canales y registros de tiempo en segundos. El usuario puede especificar el nombre del archivo.

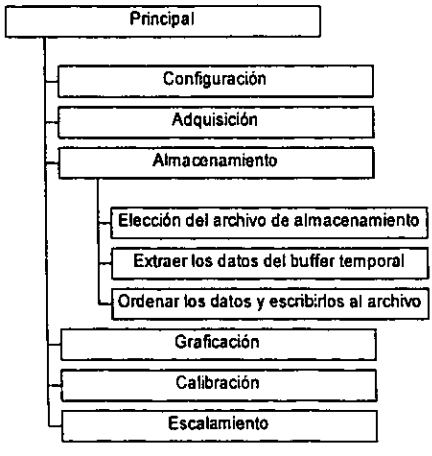


Figura 2.6 Almacenamiento

CALIBRACIÓN

Dado que los experimentos tienen un valor de referencia, los valores adquiridos deben sujetarse a dicha condición. Es necesario que el usuario tenga la posibilidad de introducir introducir una serie de valores que serán ajustados a una recta por medio del método de mínimos cuadrados o bien de ingresar una función, de esta manera los datos adquiridos serán convertidos mediante dicha transformación.

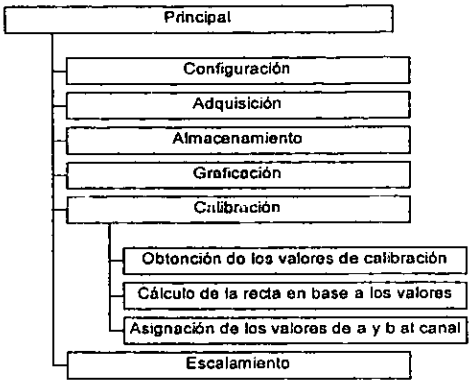


Figura 2.7 Calibración

ESCALAMIENTO

Esta función se refiere a la posibilidad de representar la señal en las unidades de la variable física medida, es deseable que existan unidades de uso frecuente que puedan seleccionarse, así como también la posibilidad de ingresar nuevas unidades.

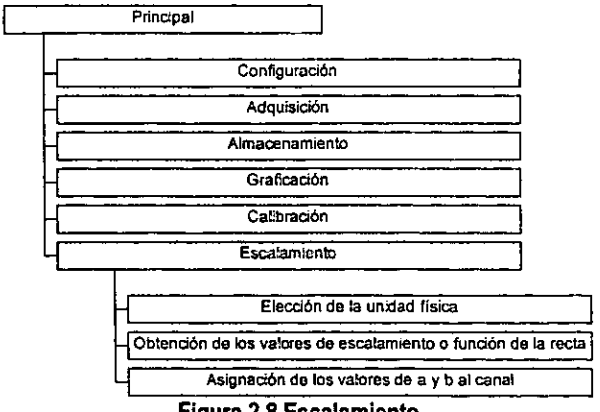


Figura 2.8 Escalamiento

Investigación de las funciones asociadas a las tarjetas de adquisición soportadas por el sistema.

FUNCIONES DE LA TARJETA PCL-812PG

Las funciones para inicializar y configurar la tarjeta de adquisición PCL-812PG son las siguientes:

- ***DRV_DeviceOpen***
  Recibe los parámetros que pertenecen a la operación del dispositivo del registro o archivo de configuración, y reserva memoria para almacenar la configuración y utilizarla de referencia rápida.

- ***DRV_DeviceClose***
  Libera la memoria que fue reservada en la función ***DRV_DeviceOpen***.

- ***DRV_DeviceGetFeatures***
  Esta función recibe una dirección de almacenamiento como argumento, y regresa las características específicas del dispositivo en dicha dirección.

- ***DRV_SelectDevice***
  Esta función muestra una lista de dispositivos para el número de dispositivo seleccionado.

- ***DRV_DeviceGetList***
  Regresa el número y tipo de dispositivos instalados.

- ***DRV_DeviceGetNumOfList***
  Regresa el número de dispositivos instalados.

- ***DRV_DeviceGetSubList***
  Regresa una lista de dispositivos instalados conectados a un Puerto COM o CAN específico.

- ***DRV_GetErrorMessage***
  Obtiene el mensaje de error para un código de error específico cuando éste ha sido obtenido de una llamada previa a otra función.

FUNCIONES DE ENTRADA ANALÓGICA

Estas funciones se utilizan para realizar una conversión A/D sencilla y recuperar los datos:

- ***DRV_AIConfig***
  Configura el rango de entrada para el canal de entrada analógica especificado.

- ***DRV_AIGetConfig***
  Lee la configuración de entrada analógica almacenada en el Registro o archivo de configuración. La aplicación llama a esta función pasándole una dirección de memoria de los datos de configuración.

- ***DRV_AIBinaryIn***
  Lee un canal de entrada analógica y regresa el resultado sin escalar (datos binarios)

- ***DRV_AIScale***
  Convierte el resultado binario de ***DRV_AIBinaryIn*** al voltaje actual de entrada.

- ***DRV_AIVoltageIn***
  Lee un canal de entrada analógica y regresa el resultado escalado a un voltaje (unidades = volts).

- ***DRV_AIVoltageInExp***
  Lee un canal de entrada analógica con una tarjeta de expansión y regresa el resultado escalado a un voltaje.

- ***DRV_MAIConfig***
  Configura los rangos de entrada para los canales de entrada analógica especificados.

- ***DRV_MAIBinaryIn***
  Lee múltiples canales de entrada analógica y regresa los resultados sin escalar (datos binarios)

- ***DRV_MAIVoltageIn***
  Lee múltiples canales de entrada analógica y regresa los resultados escalados a voltajes (unidades = volts).

- ***DRV_MAIVoltageInExp***
  Lee múltiples canales de entrada analógica con una tarjeta de expansión y regresa los resultados escalados a voltajes.

FUNCIONES DE ALTA VELOCIDAD

Estas funciones son utilizadas para el establecimiento, la inicialización y el monitoreo de las operaciones asíncronas de adquisición de datos.

- ***DRV_FAIIntStart***
  Inicia una adquisición asíncrona, de un solo canal con interrupción por transferencia y almacena los datos adquiridos en un arreglo.

- ***DRV_FAIDmaStart***
  Inicia una adquisición asíncrona, de un solo canal con DMA (*Direct Access*

Memory) y almacena los datos adquiridos en un arreglo.

- ***DRV_FAIIntScanStart***
  Inicia una adquisición de múltiple canal, con o sin intervalo de muestreo y almacena los datos adquiridos en un arreglo con interrupción por transferencia.

- ***DRV_FAIDmaScanStart***
  Inicia una adquisición de múltiple canal, con o sin intervalo de muestreo y almacena los datos adquiridos en un arreglo por transferencia DMA.

- ***DRV_FAICheck***
  Comprueba si la adquisición de datos actual ha sido completada y regresa el estatus.

- ***DRV_FAITransfer***
  Transfiere los datos del buffer que esta siendo usado por la operación de adquisición al buffer especificado.

- ***DRV_FAIStop***
  Cancela la operación de adquisición actual y reestablece el hardware y el software.

- ***DRV_AllocateDMABuffer***
  Reserva un buffer DMA para la adquisición de datos con transferencia DMA.

- ***DRV_FreeDMABuffer***
  Libera el buffer reservado por ***DRV_AllocateDMABuffer***.

- ***DRV_ClearOverrun***
  Limpia la bandera de sobrecarga.

- ***DRV_EnableEvent***
  Habilita un evento.

- ***DRV_CheckEvent***
  Limpia el evento y regresa alguna información del dispositivo.

- ***DRV_TimerCountSetting***
  Cambia el valor del interruptor contador/tiempo del dispositivo PCI dinámicamente.

FUNCIONES CLASIFICADAS POR OBJETIVO PCL-812PG

La secuencia lógica de operación y las funciones que lo llevan a cabo son las siguientes:

1. Verificar que existan dispositivos instalados y el tipo de éstos.

   DRV_DeviceGetNumOfList
   Obtiene el número de dispositivos instalados regresando: `SUCCESS (1)` si fue

exitoso.

DRV_DeviceGetList
Obtiene la lista de dispositivos instalados, dicha lista contiene el nombre de los dispositivos.

2. Obtener las características del dispositivo.

DRV_DeviceOpen
Recibe los parámetros que pertenecen a la operación del dispositivo del registro o archivo de configuración, y reserva memoria para almacenar la configuración y utilizarla de referencia rápida. Esta función debe ser llamada antes de otras funciones que usan el **DeviceHandle** como parámetro.

DRV_DeviceGetFeatures
Recupera las características del dispositivo especificado y las regresa en un buffer, dicho buffer contiene la siguiente información:

- ⇨ Versión del *device driver*
- ⇨ Nombre del dispositivo
- ⇨ Máximo número de canal para modo: diferencial, único, de salida analógica, salida digital, entrada digital, de alarma y contador/tiempo.
- ⇨ Número de bits para el convertidor A/D
- ⇨ A/D channel width in bytes
- ⇨ Número de bits para el convertidor D/A
- ⇨ D/A channel width in bytes
- ⇨ Número máximo para el código de ganancia
- ⇨ Lista de ganancias
- ⇨ Permutación

3. Configuración e inicialización de la adquisición de datos.

DRV_FAIIntScanStart
Inicia una adquisición de múltiple canal por interrupción y obtiene la configuración de entrada de un arreglo. Dicho arreglo contiene los siguientes elementos:

- ⇨ Fuente de disparo (triggering): externa (1), interna (0)
- ⇨ Frecuencia de muestreo en segundos
- ⇨ Número de canales
- ⇨ Canal inicial
- ⇨ Arreglo con el código de ganancias correspondientes a los canales
- ⇨ Buffer de datos reservado por el usuario
- ⇨ Contador de conversión
- ⇨ Contador para interrupción

4. Obtención de los datos adquiridos.

   DRV_FAITransfer
   Transfiere los datos del buffer que esta siendo utilizado por la operación de adquisición de datos al buffer definido por el usuario.

   Los parámetros que esta función recibe son:

   - ⇨ Buffer activo: 0 – buffer A, (o buffer sencillo) 1 – buffer B
   - ⇨ Espacio reservado para el buffer de datos definido por el usuario
   - ⇨ Tipo de dato: unsigned short (0), float-point(1)
   - ⇨ Punto inicial en el buffer de origen a ser copiado en el buffer destino.
   - ⇨ Número de puntos en buffer de origen a ser copiados al buffer de datos
   - ⇨ Status de sobreflujo: overrun (1), no overrun (0)

5. Detener la adquisición.

   DRV_FAIStop
   Cancela la operación de adquisición actual y reestablece el hardware y el software.

6. Liberación del dispositivo.

   DRV_DeviceClose
   Libera el espacio reservado por *DRV_DeviceOpen*

FUNCIONES DE LA TARJETA AT-MIO-64F-5

Las funciones para inicializar y configurar la tarjeta de adquisición PCL-812PG son las siguientes:

- ***Get_DAQ_Device_Info***
  Permite obtener información perteneciente al dispositivo.

- ***DAQ_DB_Transfer***
  Transfiere la mitad de los datos del buffer, que esta siendo usado durante una adquisición de doble buffer, a otro buffer, el cual es pasado a la función, y espera hasta que los datos transferidos se encuentren disponibles antes de regresar de la función.

- ***Config_DAQ_Event_Messsage***
  Notifica a las aplicaciones NI_DAQ cuando el status de una operación asíncrona de conversión contiene cierto criterio especificado por el usuario. La notificación es hecha a través del API PostMessage de Windows y/o una función de callback.

- ***DAQ_Clear***
  Cancela la operación actual de conversión (para un canal y múltiples canales) y reinicializa el sistema de circuitos eléctricos del convertidor.

- ***SCAN_Start***
  Inicia una operación de adquisición múltiple, con o sin intervalo de muestreo, y lo almacena su entrada en un arreglo.

- ***SCAN_Setup***
  Inicializa el sistema de circuitos eléctricos del convertidor para una operación de adquisición de datos. Dicha inicialización incluye el almacenamiento de una tabla que contiene la secuencia de canales y el factor de ganancia para cada canal a ser digitalizado.

- ***DAQ_DB_Config***
  Habilita o deshabilita las operaciones con doble buffer.

- ***DAQ_Rate***
  Convierte la tasa de transferencia de conversión analógico-digital en los valores de base de tiempo e intervalo de muestreo necesarios para obtener la velocidad de muestreo deseada.

- ***Timeout_Config***
  Establece una pausa que será utilizada por las funciones asíncronas para asegurar que dichas funciones regresaran eventualmente el control a la aplicación.

FUNCIONES CLASIFICADAS POR OBJETIVO AT-MIO-64F-5

La secuencia lógica de operación y las funciones que lo llevan a cabo son las siguientes:

1. Verificar que existan dispositivos instalados y el tipo de éstos.

   Get_DAQ_Device_Info Permite obtener información perteneciente al dispositivo

Tipo	Nombre	Descripción
Entrada	deviceNumber	asignada por la utilería de configuración
	infoType	tipo de información que se quiere obtener
Salida	infoValue	información obtenida

Tabla 2.1 Argumentos y valor regresado por Get_DAQ_Device_Info

ejemplo para AT-MIO-64F-5.*:*
infoType: ND_DEVICE_TYPE_CODE **infoValue:** 20

2. Obtener las características del dispositivo.

   No existe una función asociada a esta operación.

3. Configuración e inicialización de la adquisición de datos.

   SCAN_Setup Inicializa el sistema de circuitos eléctricos del convertidor para una operación de adquisición de datos. Dicha inicialización incluye el almacenamiento de una tabla que contiene la secuencia de canales y el factor de ganancia para cada canal a ser digitalizado.

Tipo	Nombre	Descripción
Entrada	deviceNumber	asignado por la utilería de configuración
	numChans	número de canales
	chanVector	vector que contiene la secuencia del muestreo de canales
	gainVector	ganancia a ser utilizada por cada canal en el vector de canales

Tabla 2.2 Argumentos y valor regresado por SCAN_Setup

SCAN_Start Inicia una operación de adquisición múltiple, con o sin intervalo de muestreo, y lo almacena su entrada en un arreglo.

Tipo	Nombre	Descripción
Entrada	deviceNumber	asignado por la utilería de configuración
	count	número de muestras
	sampTimebase	resolución utilizada para el contador de muestreo
	sampInterval	longitud del intervalo de muestreo
	scanTimebase	resolución utilizada para el contador de adquisición
	scanInterval	longitud del intervalo de adquisición
Salida	buffer	resultados de la adquisición

Tabla 2.3 Argumentos y valor regresado por SCAN_Start

4. Obtención de los datos adquiridos.

   DAQ_DB_Transfer Transfiere la mitad de los datos del buffer, que esta siendo usado durante una adquisición de doble buffer, a otro buffer, el cual es pasado a la función, y espera hasta que los datos transferidos se encuentren disponibles antes de regresar de la función.

Tipo	Nombre	Descripción
Entrada	deviceNumber	asignado por la utilería de configuración
Salida	halfBuffer	arreglo de enteros en el cual los datos van a ser transferidos.
	PtsTfr	número de puntos transferidos

	daqStopped	Indica que se completó una adquisición en modo pre-trigger.

Tabla 2.4 Argumentos y valor regresado por DAC_DB_Transfer

5. Detener la adquisición.

DAQ_Clear Cancela la operación actual de conversión (para un canal y múltiples canales) y reinicializa el sistema de circuitos eléctricos del convertidor.

Tipo	Nombre	Descripción
Entrada	deviceNumber	asignado por la utilería de configuración

Tabla 2.5 Argumentos y valor regresado por DAC_Clear

6. Liberación del dispositivo.

No existe una función asociada a esta operación.

Objetivos	PCL-812PG	AT-MIO-64F-5
1. Verificar que existan dispositivos instalados y el tipo de éstos.	DRV_DeviceGetNumOfList DRV_DeviceGetList	Get_DAQ_Device_Info
2. Obtener las características del dispositivo	DRV_DeviceOpen DRV_DeviceGetFeatures	No existe una función asociada a esta operación.
3. Configuración e inicialización de la adquisición de datos.	DRV_FAIIntScanStart	SCAN_Setup SCAN_Start
4. Obtención de los datos adquiridos.	DRV_FAITransfer	DAQ_DB_Transfer
5. Detener la adquisición.	DRV_FAIStop	DAQ_Clear
6. Liberación del dispositivo.	DRV_DeviceClose	No existe una función asociada a esta operación.

Tabla 2.6 Tabla comparativa de las funciones por objetivo de las tarjetas de adquisición

Operaciones mínimas necesarias en un convertidor genérico

Start Inicia la adquisición de datos.

Stop Detiene la adquisición de los datos.

Reset Establece los valores por omisión, además detiene el proceso de conversión si lo hubiera.

Sample Extrae una muestra del buffer destinado al almacenamiento temporal de muestras.

nSample Obtiene una cantidad específica de muestras del buffer destinado al almacenamiento temporal de muestras.

VSample Extrae una muestra transformada a voltaje del buffer destinado al almacenamiento temporal de muestras.

nVSample Extrae una cantidad específica de muestras y las transforma a voltajes.

SSample Obtiene el valor del canal convertido y luego lo escala mediante la función **Escalar**.

setStartChanel
getStartChanel La primera establece el canal inicial a muestrear y la segunda lo obtiene.

setEndChanel
getEndChanel Establecen y obtienen el canal final a muestrear.

setGain Establece el factor de ganancia.

setGainStr
getGainStr Establece y obtiene la ganancia respectivamente.

setFrec
getFrec Establece la frecuencia de muestreo y obtiene la frecuencia de muestreo.

getIndexGain Obtiene el índice asociado a la ganancia

Operaciones necesarias para cada convertidor específico

Las siguientes funciones deberán ser diseñadas para cada una de las tarjetas de adquisición, aunque deben pertenecer también al genérico.

startHardware	Esta función es la encargada de iniciar la conversión de datos para cada tarjeta de conversión específica. Configuración de canales, ganancias y frecuencias, etc.
stopHardware	Esta función es la encargada de detener la conversión de datos para cada tarjeta de conversión específica y la liberación de recursos.
getGainStrings	La función obtiene las cadenas correspondientes a las ganancias, que soporta la tarjeta de conversión.
Volts	Convierte una muestra en un valor en voltaje con base en la ganancia elegida.
Scaled	Convierte una muestra en un valor dado un escalamiento.

Interacción entre el código específico y el genérico

La función denominada **start** en el convertidor genérico será la encargada de llamar a **starthardware**, y la función de **stop** llamará a **stophardware**. De esta manera las operaciones generales serán efectuadas y el funcionamiento de la tarjeta estará controlado vía el convertidor genérico, tomando en cuenta las diferencias de hardware mediante las funciones de código específico.

La función **getGainStrings** será la encargada de obtener las ganancias particulares que maneja la tarjeta, para que las funciones generales tales como **getGainStr** sean capaces de retraer valores de ganancias válidas.

Dado que las ganancias son distintas para cada tarjeta, la función **Volts** también debe ser diseñada de manera individual.

CAPÍTULO 3 DISEÑO

Jerarquía de herencia

De acuerdo con los requerimientos del sistema, se decide utilizar la metodología orientada a objetos, para poder utilizar el sistema para varias tarjetas de conversión. Se requiere una clase convertidor que tenga las características generales de un convertidor analógico-digital, los convertidores específicos heredaran las características de la clase general o padre, implantando las características individuales en su diseño.

La metodología orientada a objetos que se elige es la de Booch, ya que provee de los elementos necesarios para la realización del proyecto:

- Los requerimientos se puede plasmar de manera sencilla con este método.
- Los modelos de objetos son fáciles de representar y de comprender.
- Las notaciones ayudan grandemente en el paso de diseño a la implantación.
- Puede encontrarse gran información y ejemplos relacionados con este método.

Un breve estudio de las metodologías (UML, OMT, Booch, OSSE, Fusion, Coad-Yourdon) esta contenido en el apéndice B de este trabajo.

Aunque UML esta siendo usado actualmente nuestro sistema de adquisición no es de grandes proporciones y muchos de los elementos adicionales como son diagramas de casos de uso y actividad, no ofrecen la utilidad que pudiera ofrecer a sistemas más complejos.

En base a los planteamientos hechos durante el análisis, la jerarquía de herencia es la siguiente:

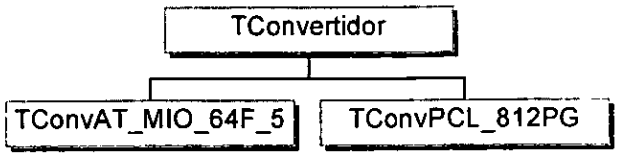


Figura 3.1 Jerarquía de herencia

La clase ***TConvertidor*** será la base para el desarrollo de las subclases y clases relacionadas, ésta tendrá las características esenciales que debe poseer un convertidor. A partir de esta heredarán las clases ***TConvAT_MIO_64F_5*** y ***TConvPCL_812PG***, con las modificaciones necesarias para que cada una de ellas pueda operar correctamente con su respectiva tarjeta de adquisición de datos.

Durante la implantación de la tarjeta PCL-812PG nos encontramos con la necesidad de utilizar un thread, el cual se creó con el nombre de clase ***TGetEvent***, para que efectúe la transferencia del buffer del convertidor de la tarjeta de adquisición al buffer circular perteneciente a la clase ***TConvertidor***.

Para la graficación de muestras es necesario que éstas se efectúen de una manera rápida y confiable. Es por ello que se creará una clase que permita el despliegue gráfico de las muestras, denominada ***TSmallGraph***, esta clase será diseñada para adaptarse totalmente a nuestras necesidades.

Las muestras serán almacenadas en un buffer circular, para controlar esta operación se diseño la clase ***TCircular***, esta clase se encarga de recibir las muestras de la tarjeta de adquisición y las clase ***TPrincipal*** va recogiendo las muestras para graficar y guardar en el archivo.

Todas las clases anteriores tienen que ser utilizadas de manera apropiada, para ello se tiene una clase encargada de proveer la interfaz con el usuario, a esta clase la llamaremos ***TPrincipal***. En esta clase enlazaremos cada una de las clases anteriores siguiendo la relaciones de uso y de herencia correspondientes.

Diagrama de Clases

El diagrama de clases se utiliza para mostrar la existencia y relaciones entre clases y de esta manera asignar las tareas y responsabilidades de cada una de las entidades que constituyen el sistema.

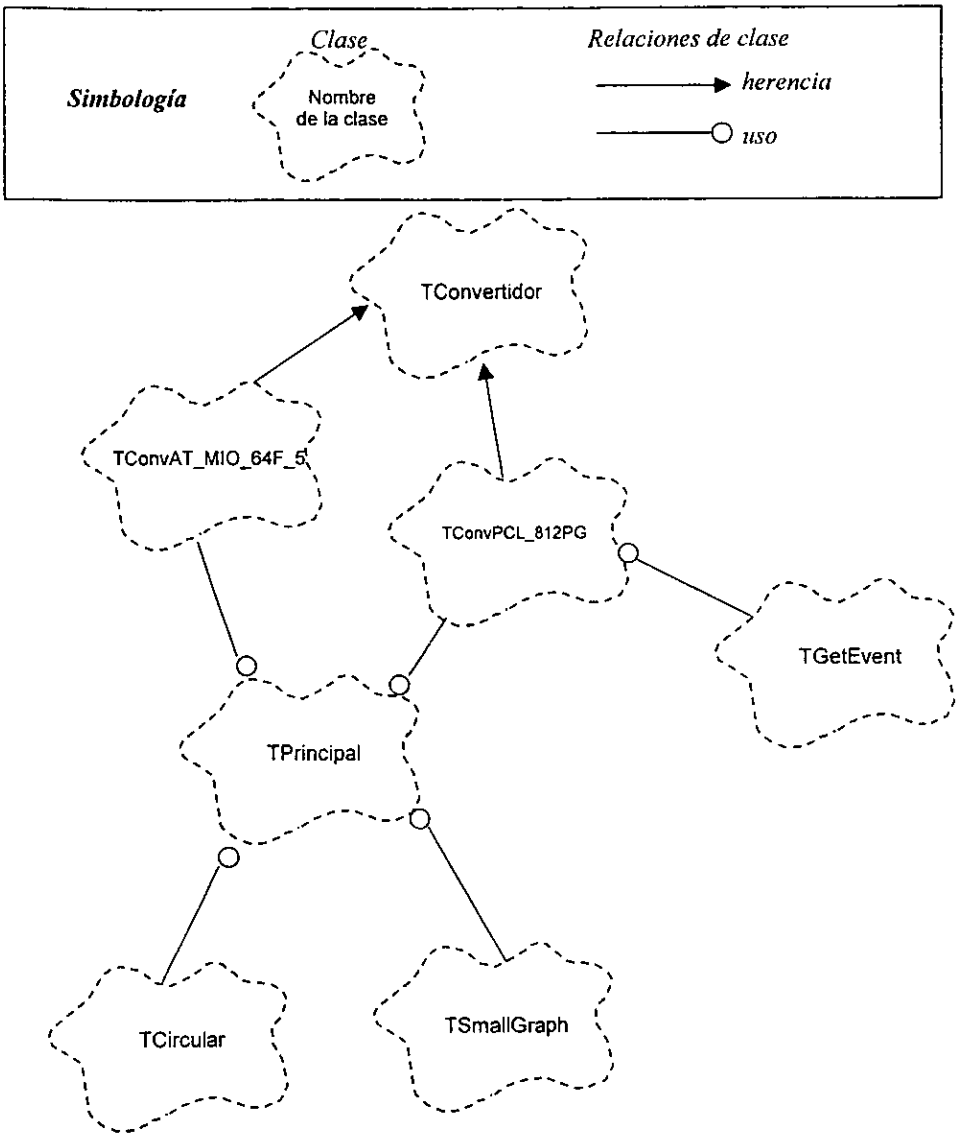


Figura 3.2 Diagrama de clases del Sistema

La Clase TConvertidor

Dentro de los elementos más importantes que debe incluir el convertidor genérico se encuentran las siguientes:

- ***Número de canales a muestrear.***
  El sistema debe de ser capaz de adquirir muestras de 4 canales diferentes simultáneamente, el número de canales permite especificar un canal inicial y un canal final.

- ***Canal máximo.***
  Dependiendo de la tarjeta de adquisición de la que se trate, se tiene un número fijo de canales. El canal máximo tiene el valor del último canal accesible para la tarjeta especificada.

- ***Frecuencia máxima de muestreo.***
  Indica el número máximo de muestras por segundo que el sistema soporta por cada canal.

- ***El factor de ganancia.***
  Indica la relación entre la entrada analógica y la señal digital transformada. Esto es la razón entre una palabra de "n" bits representada por un número entero y cada uno de los niveles lógicos de tensión de entrada.

- ***El arreglo de ganancias asociadas***
  Dependiendo de la tarjeta de adquisición se tienen un número determinado de ganancias, siendo ejemplos de estas ±5[V], ±2.5[V]

- ***La ganancia y la frecuencia a la que está trabajando.***
  Ya que la adquisición se efectúa en tiempo real, la frecuencia de muestreo debe de ser la misma para todos los canales involucrados, aunque la ganancia puede variar de canal en canal, los requerimientos del sistema plantean una misma ganancia para todos los canales.

Diseño de las clases para cada tarjeta, derivadas de TConvertidor

Los convertidores específicos son dos para este caso y las funciones que son propias de cada uno de ellos se listan a continuación:

La función **getGainStrings** asignara las cadenas y los valores de ganancia para cada una de las tarjetas

Para ***TConvAT_MIO_64F_5*** las ganancias son:

+/-50mV, +/-100mV, +/-0.25V, +/-0.5V, +/-1V, +/-2.5V, +/-5V y +/-10V

Para ***TConvPCL_812PG*** las ganancias son:

+/- 5V, +/- 2.5V, +/- 1.25V, +/- 0.625V, +/- 0.3125V.

La función **Volts** convertirá el valor de muestra adquiridos a un valor de voltaje

Para ***TConvAT_MIO_64F_5*** la forma de obtener el valor de voltaje a partir de la muestra es:

*Volts = Muestra * FactorDeGanancia*

Para ***TConvPCL_812PG*** la forma de obtener el valor de voltaje a partir de la muestra es:

Volts = (*Muestra*-2048) * *FactorDeGanancia*

La función **Scaled** transformará el valor mediante un valor de escala y un valor de offset, los cuales son distintos para cada canal.

Para ***TConvAT_MIO_64F_5*** la forma de obtener el escalamiento a partir de la muestra del canal X es:

*ValorEscalado = Muestra * FactorDeEscalaCanalX + FactorDeOffsetCanalX*

Para ***TConvPCL_812PG*** la forma de obtener el escalamiento a partir de la muestra del canal X es:

ValorEscalado = (Muestra-2048) FactorDeEscalaCanal + FactorDeOffsetCanal*

Plan para la adquisición de muestras y la clase TCircular

Cada tarjeta puede tener un método propio de almacenamiento de muestras, ya sea en un medio interno propio y dentro de la tarjeta o bien, recurriendo al almacenamiento en disco en la máquina donde se esta corriendo el programa.

Por lo anterior es necesario diseñar una estrategia para la adquisición de muestras. TCircular es una clase encargada de obtener la muestras que son generadas por el convertidor.

Se cuenta con un arreglo que forma una lista circular con el máximo de puntos configurables en la creación del objeto. Como buffer circular contiene un inicio (head) y un fin (tail), debe de ser capaz de indicar cuando ha ocurrido una sobreescritura, y llevando

el control de la distancia entre extremos. De esta manera las principales propiedades que se requieren son:

- ***Distancia***
  Valor que indica el número de enteros almacenados en la lista circular.
- ***Cola o Fin***
  Valor del extremo posterior de la lista circular.
- ***Cabeza o Inicio***
  Valor del extremo anterior de la lista circular.
- ***Vuelta.***
  Bandera que indica si se ha completado una vuelta en la lista circular.
- ***Sobreflujo***
  Bandera que indica si ha ocurrido una sobre-escritura de datos

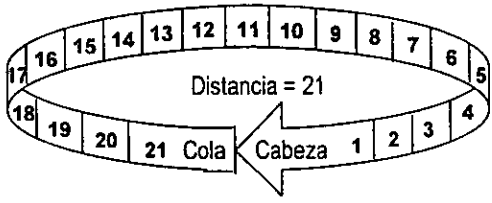


Figura 3.3 Buffer circular

La secuencia para el almacenamiento de datos es la siguiente: La tarjeta de adquisición transfiere la información al buffer circular, entonces las funciones de graficación y almacenamiento de archivos, acuden al buffer circular para extraer los datos. El siguiente diagrama ejemplifica la operación.

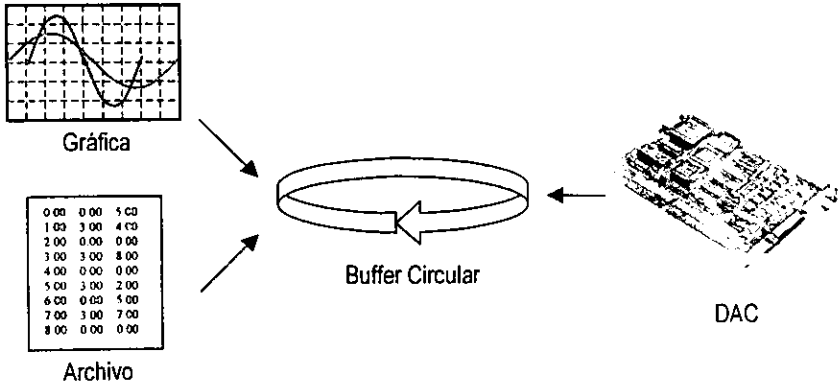


Figura 3.4 Interacción del buffer circular con los elementos del Sistema

El método de graficación y la Clase TSmallGraph

Se requiere una clase encargada de realizar la presentación en pantalla de las muestras adquiridas en una gráfica para los distintos canales. Como se había especificado en el análisis, se requieren 4 canales simultáneos en la presentación, el usuario podrá elegir los colores para facilitar la visibilidad.

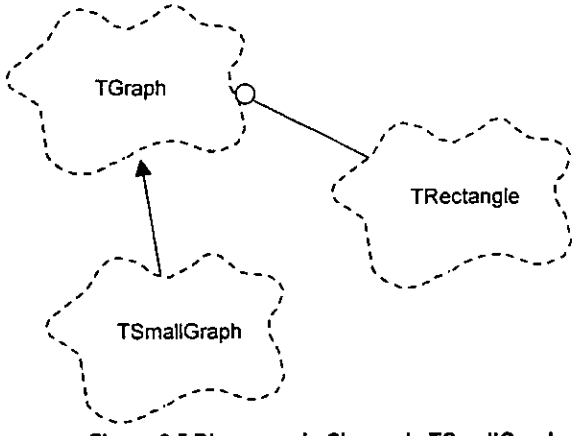


Figura 3.5 Diagrama de Clases de TSmallGraph

Dentro de las propiedades que debe incluir la clase de graficación se encuentran las siguientes:

- ***Rango Máximo.***
  En el eje Y debe especificarse un rango máximo, este valor nos dará el límite superior en el eje Y para dibujar la gráfica.

- ***Rango Mínimo.***
  En el eje Y debe especificarse un rango mínimo, este valor nos dará el límite inferior en el eje Y para dibujar la gráfica.

- ***Intervalo***
  El rango elegido en Y será dividido en n partes iguales, y cada parte tendrá el valor de Intervalo.

- ***Número de divisiones en X***
  De acuerdo a las necesidades, el eje X varía en un intervalo unitario, teniendo su valor mínimo en 0 y el máximo es asignado mediante el número de divisiones.

- ***Número de Scroll***
  A medida que el tiempo transcurre la gráfica debe recorrerse al llegar al final, esta propiedad indica los recuadros a ser desplazados.

- ***Borde***
  Se define como el margen superior, inferior, izquierdo y derecho de la gráfica

- ***Color de los trazos.***
  Cada canal deberá poseer un color para que pueda ser diferenciado de los otros.

- ***Color de la malla y ejes***
  Este es el color de la malla y ejes X, Y en la gráfica.

- ***Color de fondo.***
  Este es el color correspondiente al fondo de la gráfica.

Comportamiento General del Sistema

De acuerdo con los planteamientos hechos durante el análisis es necesario un módulo principal que controle a los demás. A continuación se presentan los módulos con sus respectivos prototipos de pantalla.

MÓDULO PRINCIPAL

Este módulo consta de un menú de opciones de donde serán llamados los demás. Se deberá mostrar la gráfica, un botón de encendido/apagado, así como la información más importante concerniente a la adquisición de datos: ganancia, frecuencia, archivo en donde serán guardados los datos y el status de escalamiento de los canales elegidos. En la parte inferior se podrán especificar 4 de los canales a ser graficados, y el tiempo transcurrido en la adquisición.

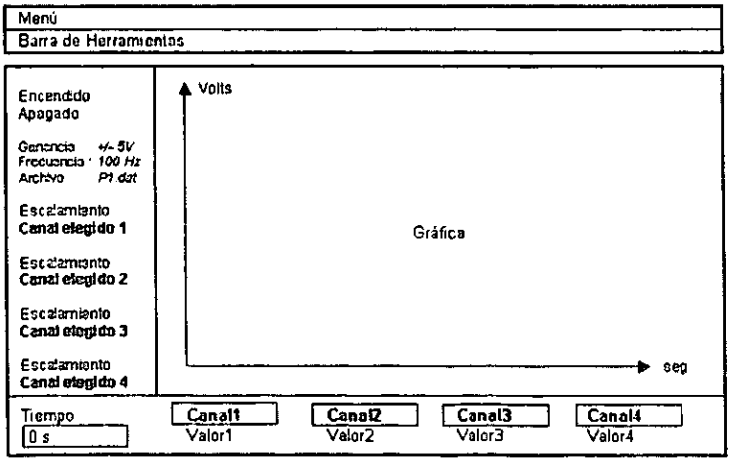


Figura 3.6 Prototipo del Módulo Principal

MÓDULO CONFIGURACIÓN

Para este módulo deben tomarse en cuenta las características de la adquisición, canal inicial, canal final, frecuencia y ganancia. También se debe incluir información sobre la frecuencia máxima. El prototipo de la pantalla para configuración es la siguiente:

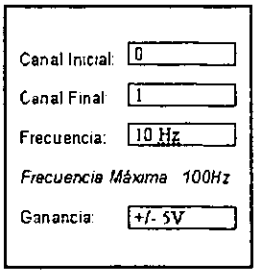


Figura 3.7 Prototipo del Módulo de Configuración

En caso de que el canal inicial sea mayor que el final o el final menor que el inicial, se debe enviar un mensaje de error al usuario. También si el producto del número de canales por la frecuencia excede la máxima frecuencia permitida.

MÓDULO DE ESCALAMIENTO

El escalamiento se refiere a una transformación que será aplicada al valor adquirido de un determinado canal. Cada canal puede tener un escalamiento distinto. Este módulo deberá obtener un valor de pendiente y ordenada para convertir dichas muestras o bien ofrecer la alternativa de que el usuario pueda ingresar pares de puntos, para después ajustarlos a una recta por medio del método de Mínimos Cuadrados. Existirá también la posibilidad de asociar estos escalamientos a una unidad determinada. Las unidades pueden ser añadidas o eliminadas y éstas se conservaran para que el usuario pueda hacer uso de ellas posteriormente. Además de que se debe indicar el status del canal como *Escalado* o *Sin escalar*.

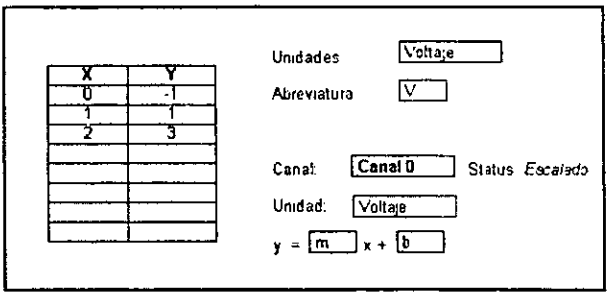


Figura 3.8 Prototipo del Módulo de Escalamiento

MÓDULO DE GRAFICACIÓN

Este módulo consta de tres botones en la parte izquierda, que contienen los colores actuales de malla, fondo y ejes de la gráfica, permitiendo que el usuario cambie dichos colores; del lado derecho se encuentran el número de divisiones en X, número de scroll, número de divisiones en Y, valor máximo en Y, valor mínimo en Y. El prototipo de pantalla para graficación es el siguiente:

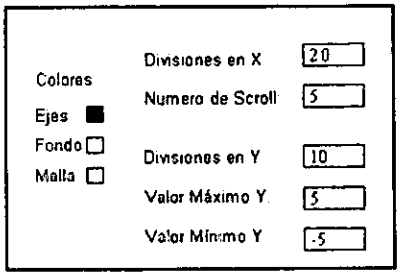


Figura 3.9 Prototipo del Módulo de Graficación

MÓDULO DE CALIBRACIÓN

Con la finalidad de escalar los canales por medio de eventos experimentales, se diseña este módulo. El usuario accionara una adquisición de 5 segundos a 10 Hz, que será graficada en el lado izquierdo y al finalizar ésta se efectuará un *promedio de las muestras adquiridas* que junto con el *valor real sensado* forma un par de puntos que servirá para ajustar la recta. A medida que los puntos cambien la gráfica correspondiente al lado derecho se actualizará automáticamente junto con la ecuación de la recta en la parte inferior.

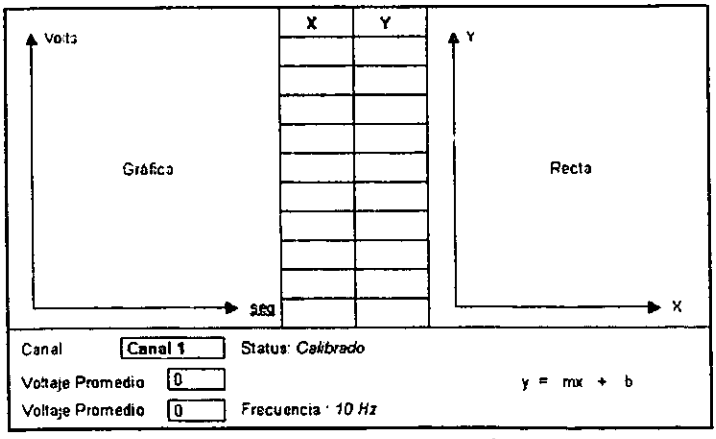


Figura 3.10 Prototipo del Módulo de Calibración

Para este módulo en la recta de ajuste es deseable que se le permita al usuario cambiar el rango de los ejes. Para esto se diseña también una pequeña pantalla para la configuración de dicha recta.

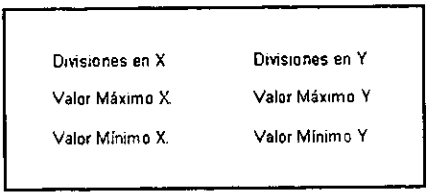
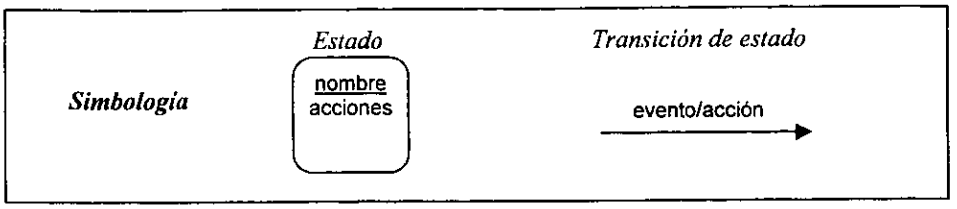


Figura 3.11 Prototipo de Pantalla para opciones de gráfica

Diagramas de Transición de Estados

En términos operativos la clase ***TPrincipal*** se encarga de coordinar el trabajo conjunto de las demás clases, por ello es importante describir es espacio de estados para esta clase con sus respectivas transiciones. De esta manera puede apreciarse claramente el funcionamiento y la secuencia seguida en la adquisición.



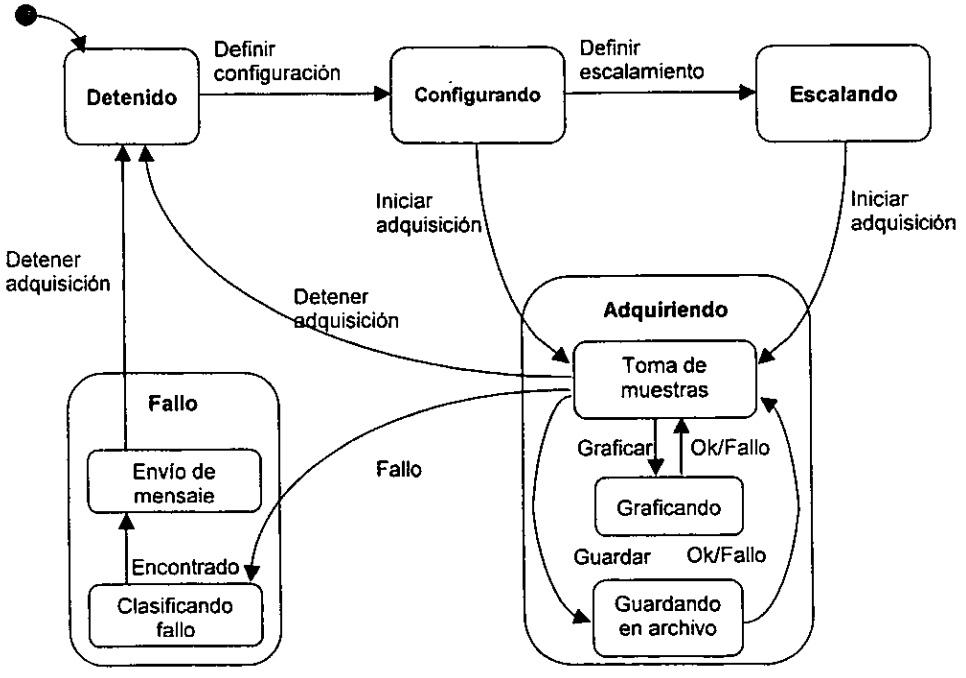


Figura 3.12 Diagrama de transición de estados de la clase principal

CAPÍTULO 4 IMPLANTACIÓN

Programación del sistema utilizando Delphi como herramienta

Delphi es una potente herramienta de desarrollo de programas que permite la creación de aplicaciones para Windows. Delphi dispone del Object Pascal, un lenguaje de programación muy poderoso. Este lenguaje surge a partir del desarrollo del Borland Pascal 7.0, un lenguaje que ocupa un lugar muy importante en la programación.

Las razones para utilizar Delphi para la implantación de este sistema son:

- Las aplicaciones pueden colocarse de forma muy sencilla en la pantalla según el principio de módulos. Para ello se dispone de una paleta que contiene una gran variedad de componentes. Esta paleta es denominada por Borland **VCL** *(Visual Component Library)*, o biblioteca de componentes visuales. Así la programación se realiza con los componentes de Delphi y no con las complejas llamadas al sistema de Windows. Esto simplifica enormemente la programación bajo Windows. Además, de requerirse, Delphi permite llamadas al sistema de Windows **API** *(Application Programming Interface)*. Para nuestro sistema serán necesarias algunas llamadas API, para graficación y manejo de memoria.

- A diferencia de otras herramientas de desarrollo visuales (Visual Basic, Toolbook, etc) con Delphi es posible crear nuevos componentes que pueden entonces incorporarse en la paleta con los componentes ya existentes y que pueden ser utilizados de la misma forma. La **VCL** puede estructurarse libremente y así adaptarse totalmente a las situaciones propias de programación. Gracias a esta característica podrá desarrollarse un componente gráfico independiente.

- Las aplicaciones terminadas quedan disponibles como archivos ejecutables (.EXE) que pueden utilizarse solos y sin bibliotecas adicionales. Consecuentemente la velocidad con la que pueden ejecutarse los programas creados es muy alta. Si se incluyen llamadas a DLLs, éstas se deben incluir junto con el ejecutable, en nuestro

caso el sistema utiliza estos DLLs, que son provistos por los fabricantes de las tarjetas.

- Delphi es una "Two-Way-Tool", es decir, una herramienta de dos direcciones, porque permite crear el desarrollo de programas de dos formas: una de forma visual en la pantalla, por medio de las funciones de Drag & Drop (Arrastrar y colocar) y la otra a través de la programación convencional, escribiendo el código. Ambas técnicas pueden utilizarse de forma alternativa o simultánea.

Configuración de pantallas

PANTALLA PRINCIPAL

De acuerdo al diseño la pantalla principal contiene un menú y una barra de herramientas. La otra parte corresponde a la gráfica con sus opciones, en la cual se agregó solo un botón para salvar, de esta forma el usuario puede tener o no habilitada la escritura al archivo durante la adquisición, además se incluyó un indicador de colores verde y rojo, que significan encendido y apagado respectivamente.

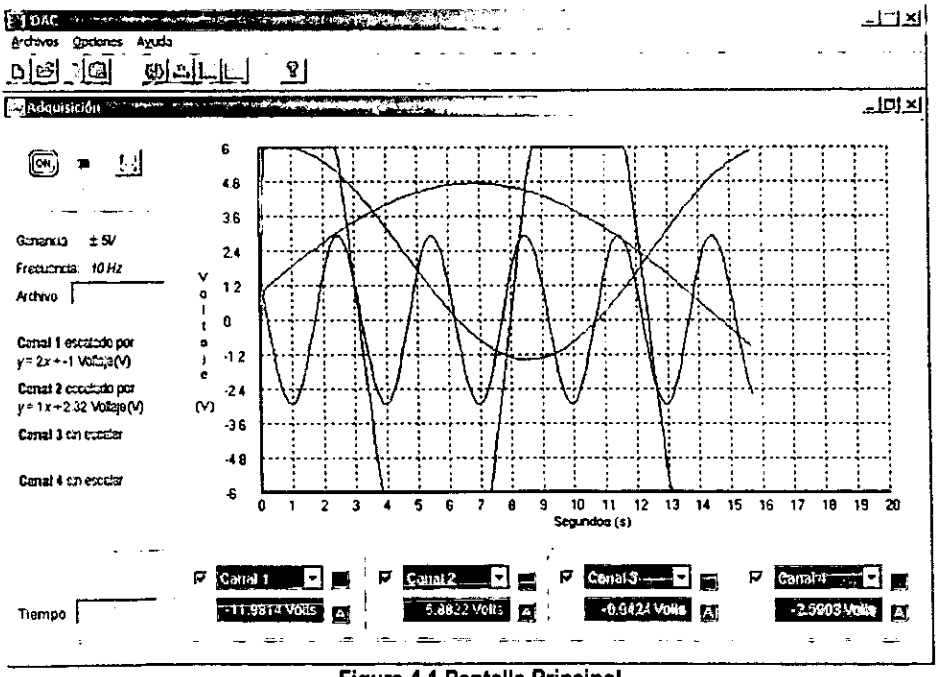


Figura 4.1 Pantalla Principal

El menú tiene su correspondiente elemento en la barra encontrándose las siguientes opciones:

Archivo Nuevo
Esta opción crea un archivo nuevo de datos, aunque tiene la extensión (.dat) el formato es texto plano.

Abrir Archivo
Para abrir resultados de experimentos anteriores el formato aceptado es de datos (.dat)

Guardar en formato Excel (.xls)
Esta opción se habilita una vez que los resultados son analizados y manipulados frecuentemente en esta hoja de cálculo. Esta opción se habilita sólo después de que ya existe el archivo de datos (.dat)

Copia la gráfica al portapapeles
Al colocarse la imagen el portapapeles, ésta puede ser pegada en otros programas tales como Word, Excel, PowerPoint, Paint, etc.

Configura la adquisición

Escala los canales

Opciones de gráfica

Calibra los canales

Ayuda

Existe una configuración inicial por default, en ésta sólo se muestrea el primer canal a 100Hz con la máxima ganancia.

Iniciar la adquisición

Al presionar botón **ON** iniciamos la adquisición de los datos, al presionarlo cambia a **OFF** y el indicador que se encuentra a la derecha de este botón cambia de color rojo a verde para indicar que el convertidor esta trabajando.

Guardar en archivo

Para guardar en un archivo éste primero deberá seleccionarse mediante Archivo Nuevo o Abrir Archivo . Una vez que se ha seleccionado un archivo la caja de archivo muestra el nombre.

Ahora el botón se muestra como  lo que indica que esta listo para guardar a archivo durante la adquisición. Para deshabilitar el almacenamiento de muestras en el archivo se presiona el botón y se verá de la siguiente manera 

Elección de canales

Al lado izquierdo de cada una de las cajas de selección de canales  se encuentra una casilla de verificación  para habilitar el trazo un canal en la gráfica presiónelo y cambiará a  . Una vez hecho estos podrá seleccionarse el canal que se desea graficar en este trazo, por ejemplo en el segundo trazo de la siguiente imagen se puede seleccionar que canal se quiere graficar.

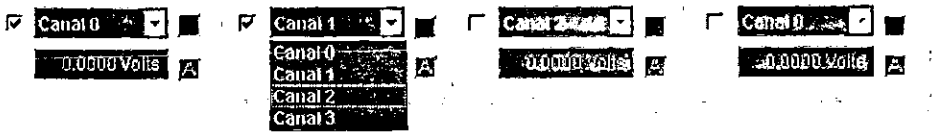


Figura 4.2 Selección de canales

Al lado derecho de la caja de selección de canales se encuentra el botón  , el cual al presionarlo abre una pantalla para cambiar el color del trazo para el canal. El botón  cambia el color de la fuente para los cajones de **blanco** a **negro** o viceversa.

Por último el cajón en la parte inferior  muestra el valor en voltaje adquirido del canal.

PANTALLA DE CONFIGURACIÓN

La pantalla de configuración mantiene los elementos propuestos en el diseño utilizando cajas de selección en los canales y la ganancia, también se añadieron botones para **Aceptar** o **Cancelar** la configuración.

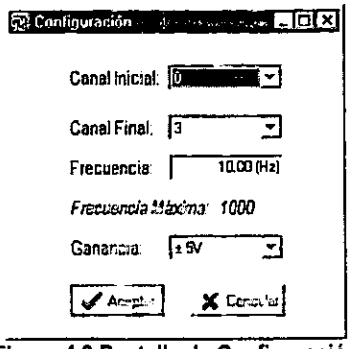


Figura 4.3 Pantalla de Configuración

PANTALLA DE ESCALAMIENTO

En la pantalla de escalamiento la unidades son controladas mediante botones de **Agregar** y **Eliminar**. Una vez ingresadas dichas unidades pueden seleccionarse para un canal en particular y junto con la a–pendiente y la b–ordenada son asignadas al canal mediante el botón **Escalar**, si se desea eliminar el escalamiento se selecciona el canal deseado y se oprime el botón **Eliminar Escalamiento.**

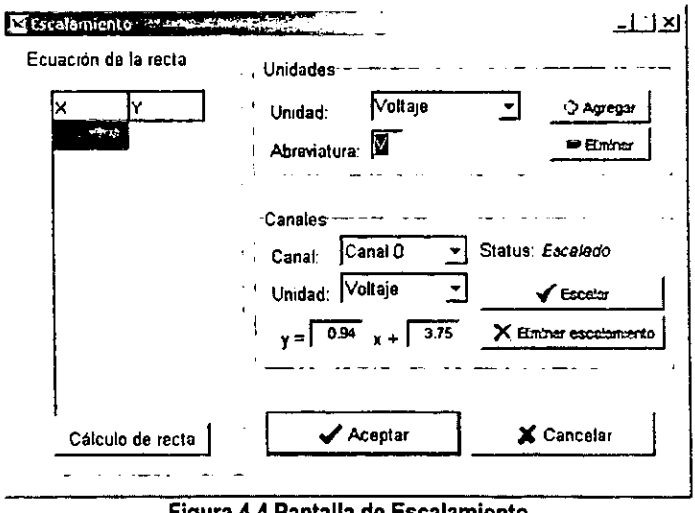


Figura 4.4 Pantalla de Escalamiento

PANTALLA DE GRAFICACIÓN

La pantalla de graficación respeta todos los elementos del diseño. Únicamente se añadieron botones para **Aceptar** o **Cancelar** los cambios.

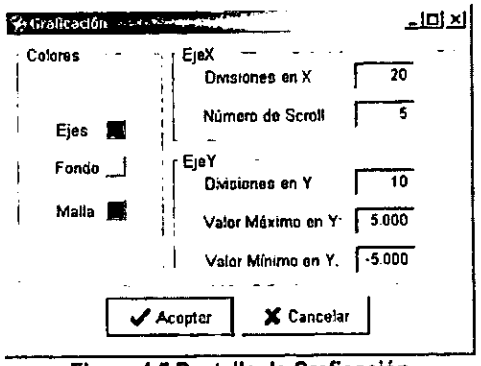


Figura 4.5 Pantalla de Graficación

PANTALLA DE CALIBRACIÓN

En la pantalla de calibración se agregaron los botones que controlan las operaciones.

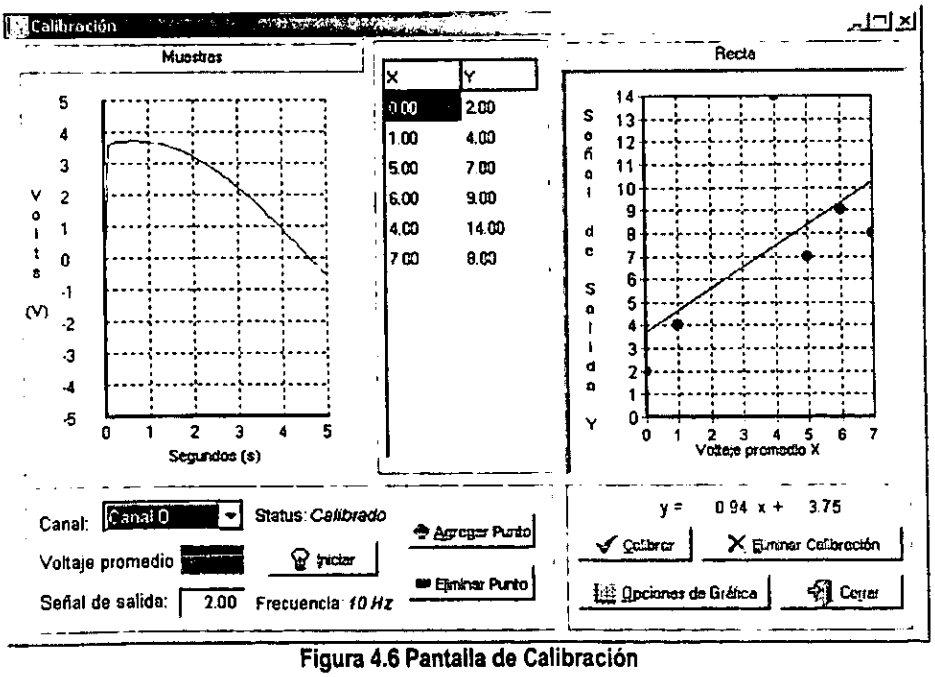


Figura 4.6 Pantalla de Calibración

A continuación se indica la tarea de cada uno de ellos:

Iniciar	Inicia una adquisición de 5 segundos a 10 Hz, al finalizar ésta se efectuará un promedio de las muestras adquiridas
Agregar Punto	Agrega un punto para el ajuste de la recta, tomando los valores del voltaje promedio y la señal de salida o valor sensado.
Eliminar Punto	Elimina el punto seleccionado.
Calibrar	Asigna la calibración dada por la ecuación de la recta al canal seleccionado.
Eliminar Calibración	Elimina la calibración del canal seleccionado.
Opciones de Gráfica	Abre una pantalla de configuración para la gráfica de la recta.
Cerrar	Cierra la pantalla.

La siguiente pantalla aparece cuando se oprime el botón Opciones de Gráfica, también tiene botones para **Aceptar** o **Cancelar** los cambios.

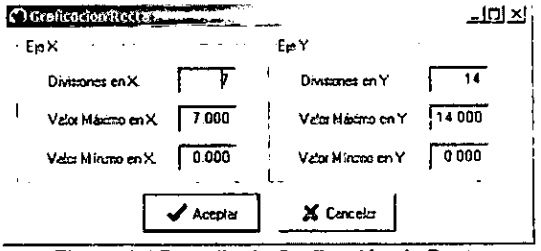


Figura 4.7 Pantalla de Graficación de Recta

Formato de los archivos

Existen dos formatos en los que se puede escribir la información.

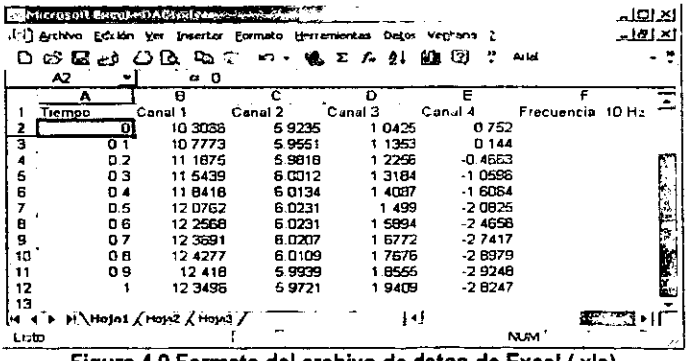
- Archivo de datos (.txt). La información escrita en él es la siguiente:

```
Numero de canales y Canales:
4 1 2 3 4
Frecuencia:
1.00000000000000E+0001
0.0000  10.3086  5.9235  1.0425  0.7520
0.1000  10.7773  5.9551  1.1353  0.1440
0.2000  11.1875  5.9818  1.2256 -0.4663
0.3000  11.5439  6.0012  1.3184 -1.0596
0.4000  11.8418  6.0134  1.4087 -1.6064
0.5000  12.0762  6.0231  1.4990 -2.0825
0.6000  12.2568  6.0231  1.5894 -2.4658
0.7000  12.3691  6.0207  1.6772 -2.7417
0.8000  12.4277  6.0109  1.7676 -2.8979
0.9000  12.4180  5.9939  1.8555 -2.9248
1.0000  12.3496  5.9721  1.9409 -2.8247
```

Número de canales : 4
Canales muestreados: 1, 2, 3 y 4
Frecuencia : 10 Hz
Primera Columna : Tiempo
Segunda Columna : Canal inicial elegido
: :
Última Columna : Canal final elegido

Figura 4.8 Formato del archivo de datos (.txt)

- Archivo de excel (.xls). El contenido es el siguiente:



	A	B	C	D	E	F
1	Tiempo	Canal 1	Canal 2	Canal 3	Canal 4	Frecuencia 10 Hz
2	0	10 3086	5 9235	1 0425	0 752	
3	0 1	10 7773	5 9551	1 1353	0 144	
4	0 2	11 1875	5 9818	1 2256	-0 4663	
5	0 3	11 5439	6 0012	1 3184	-1 0596	
6	0 4	11 8418	6 0134	1 4087	-1 6064	
7	0 5	12 0762	6 0231	1 499	-2 0825	
8	0 6	12 2568	6 0231	1 5894	-2 4658	
9	0 7	12 3691	6 0207	1 6772	-2 7417	
10	0 8	12 4277	6 0109	1 7676	-2 8979	
11	0 9	12 418	5 9939	1 8555	-2 9248	
12	1	12 3496	5 9721	1 9409	-2 8247	
13						

Figura 4.9 Formato del archivo de datos de Excel (.xls)

Elaboración del archivo de ayuda

A fin de que el usuario tenga una referencia del sistema disponible en cualquier momento, se incluyó en el programa una ayuda que contempla cada uno de los módulos y su operación. Esta ayuda es fácilmente accesible cuando se esta ejecutando el sistema, presionando el botón de Ayuda que se encuentra en el menú.

La ayuda se dividió en 5 partes utilizando los módulos ya explicados anteriormente.

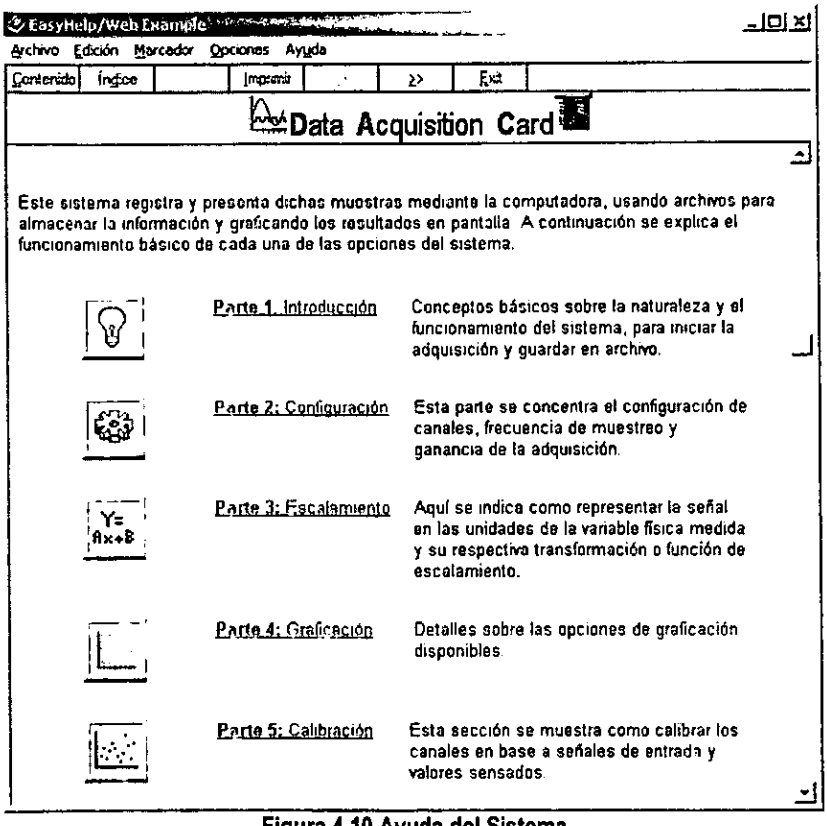


Figura 4.10 Ayuda del Sistema

En cada uno de estos tópicos se explica el funcionamiento y propósito de cada módulo. También se especifica el formato y tipo de archivos.

CAPÍTULO 5
PRUEBAS AL SISTEMA BAJO DIFERENTES CONDICIONES DE OPERACIÓN

Pruebas

La fase de prueba constituye una parte muy importante en la funcionalidad, ya que la adquisición tiene que ser en tiempo real y generalmente en pruebas que difícilmente se pueden realizar otra vez, por lo tanto dicha información resulta sumamente valiosa.

Las pruebas se realizaron a lo largo de la elaboración del sistema, ya que ante los cambios el tiempo de operación experimentaba variaciones. En un inicio no se tenía contemplado el desarrollo de un componente gráfico, sin embargo durante la realización de pruebas iniciales se observó que al despliegue gráfico limitaba fuertemente el desempeño del sistema, fundamentalmente por la utilización de un componente gráfico de propósito general y por lo tanto con un alto costo en recursos del sistema.

Los archivos tipo texto ocupan un espacio mayor que los de tipo binario, es por eso que el tamaño de los archivos en estas pruebas son extremadamente grandes cuando se trabaja a frecuencias altas y se muestrea un largo tiempo, sin embargo el tiempo requerido para los experimentos de hidráulica a altas frecuencias es pequeño (10 minutos o menos).

Las pruebas fueron hechas en una máquina con las siguientes características:

Procesador: **Intel Pentium II**
Memoria: **64 MB en RAM**
Espacio en disco duro: **6 GB**
Sistema Operativo: **Microsoft Windows 2000**
Tarjeta de adquisición: **PCL-812PG**

Tiempos de adquisición y tamaño de los archivos

CARGA LIGERA

*Un Canal a **400 Hz***

El sistema adquirió y desplegó un total de 4750.2560 segundos. La prueba se concluyó en este tiempo, estimando que podía seguir trabajando así, tanto como se requiera, sólo limitado por la capacidad de almacenamiento de la computadora y del sistema operativo.

El tamaño del archivo es de 30,323 ***KB*** = 29.61 ***MB***

*Dos Canales a **200 Hz***

El sistema adquirió y desplegó un total de 4680.9350 segundos. La prueba se concluyó en este tiempo, estimando que podía seguir trabajando así, tanto como se requiera, sólo limitado por la capacidad de almacenamiento de la computadora y del sistema operativo.

El tamaño del archivo es de 21,934 ***KB*** = 21.41 ***MB***

*Cuatro Canales a **100 Hz***

El sistema adquirió y desplegó un total de 4960.3760 segundos. La prueba se concluyó en este tiempo, estimando que podía seguir trabajando así, tanto como se requiera, sólo limitado por la capacidad de almacenamiento de la computadora y del sistema operativo.

El tamaño del archivo es de 18,642 ***KB*** = 18.2 ***MB***

CARGA MEDIANA

*Un Canal a **2000 Hz***

El sistema adquirió y desplegó un total de 2560.4320 segundos. La prueba se concluyó en este tiempo, con el conocimiento de que, el tiempo requerido para los experimentos a esta frecuencia es menor.

El tamaño del archivo es de 80,240 ***KB*** = 78.35 ***MB***

*Dos Canales a **1000 Hz***

El sistema adquirió y desplegó un total de 2467.8420 segundos. La prueba se concluyó en este tiempo, con el conocimiento de que, el tiempo requerido para los experimentos a esta frecuencia es menor.

El tamaño del archivo es de 58,250 ***KB*** = 56.88 ***MB***

Cuatro Canales a 500 Hz

El sistema adquirió y desplegó un total de 2897.5643 segundos. La prueba se concluyó en este tiempo, con el conocimiento de que, el tiempo requerido para los experimentos a esta frecuencia es menor.

El tamaño del archivo es de 56,280 ***KB*** = 54.96 ***MB***

ALTA CARGA

*Un Canal a **4000 Hz***

No se experimentaron pérdidas hasta 686.2440 segundos. El sistema se detuvo en este punto debido a sobrecarga. Cumpliendo con el tiempo de adquisición requerido para los experimentos de hidráulica.

El tamaño del archivo es de **11,618 *KB*** = **11.3 *MB***

*Dos Canales a **2000 Hz***

No se experimentaron pérdidas hasta 1462.2520 segundos. El sistema se detuvo en este punto debido a sobrecarga. Cumpliendo con el tiempo de adquisición requerido para los experimentos de hidráulica.

El tamaño del archivo es de **20,043 *KB*** = **19.57 *MB***

*Cuatro Canales a **1000 Hz***

No se experimentaron pérdidas hasta 3054.9170 segundos. El sistema se detuvo en este punto debido a sobrecarga. Cumpliendo con el tiempo de adquisición requerido para los experimentos de hidráulica.

El tamaño del archivo es de **125,696 *KB*** = **122 *MB***

Comportamiento bajo condiciones adversas

Es importante que los datos sean almacenados inmediatamente después de adquirirlos, ya que ante una falla el archivo debe contener los datos hasta donde ésta ocurrió. Algunos experimentos requieren condiciones que resultan difíciles de repetir.

De esta forma, al ocurrir un error como sobrecarga o falta súbita energía, los datos adquiridos hasta ese momento se encuentran ya almacenados en el archivo.

CONCLUSIONES

El sistema final, permite disponer de un proceso de adquisición de datos confiable flexible y oportuno.

Disponer de un sistema que permitiera contar información precisa y oportuna de las variables medidas y escalamientos establecidos por el usuario en fenómenos hidráulicos, así como también el registro en archivos de la información arrojada en cada fenómeno, constituye la consecución de los objetivos de esta tesis. El sistema obtenido ayudará a una mejor evaluación y modelado de los fenómenos hidráulicos, y por lo tanto auxiliará en la toma de decisiones en la operación de los sistemas hidráulicos estudiados.

Además, se realizaron las pruebas pertinentes para determinar la eficacia y precisión y confiabilidad del almacenamiento de datos. En estas pruebas se obtuvieron resultados adecuados que satisfacen las necesidades del grupo de hidromecánica del Instituto de Ingeniería de la UNAM.

La programación orientada a objetos como base del sistema, ayudó a la obtención de un sistema escalable y flexible que ofrece la posibilidad de incorporar nuevos tipos de tarjetas con sólo definir sus respectivas unidades de operación, integrándose inmediatamente al sistema desarrollado. Esto evita que se genere un sistema con mayor susceptibilidad de errores y más fácil de actualizar y mantener. Además este sistema cumple con la flexibilidad necesaria para que un usuario no especializado en computación lo pueda utilizar.

El sistema es lo suficientemente robusto para responder adecuadamente a la falta súbita de energía. En la actualidad, el sistema se encuentra liberado y ha beneficiado a la comunidad del Laboratorio de Hidráulica.

GLOSARIO

Analógico(a) - La principal característica de las representaciones analógicas es que estas son continuas, es decir, que entre dos puntos que representan valores existe una infinidad de valores intermedios.

Aplicación - O programa de aplicación. Un programa o grupo de programas diseñados para usuarios finales. Las aplicaciones se encuentran en la cima de los sistemas de software a causa de que son incapaces de correr sin el sistema operativo o las utilerías del sistema.

Apuntador - En programación, un apuntador es un tipo especial de variable que contiene una dirección de memoria (esto es, apunta a una localidad de memoria).

Archivo - Una colección de datos o información que tiene un nombre, llamado nombre de archivo. Diferentes tipos de archivo almacenan diferente tipo de información.

Archivo Binario - Archivo que contiene datos o instrucciones en formato binario.

Bandera - Una marca de software que señaliza una condición o estado particular. Una bandera es como un interruptor que puede estar apagado o encendido.

Bit - Unidad mínima de información en sistemas computacionales, sólo puede tomar el valor de 0 o 1.

Bloque - Conjunto de datos organizados de manera subsecuente en el orden de direcciones.

Booleano - Referencia a una expresión o a una variable que puede tener solamente un valor verdadero o falso.

Buffer - Es un área de almacenamiento temporal, usualmente en RAM. El propósito de la mayoría de los buffers es el actuar como un área de retención, permitiendo que la

unidad de procesamiento pueda manipular los datos antes de transferirlos a un dispositivo.

Byte - Se refiere a un tamaño específico de datos, siendo un byte equivalente a un conjunto de ocho bits. Un byte es la unidad de datos básica.

Código Fuente - Instrucciones de programa en su forma original. Inicialmente, un programador escribe un programa en un lenguaje de programación en particular. Esta forma del programa se llama programa fuente, o de manera más general, código fuente.

CPU - Abreviación de Unidad de Procesamiento Central. Algunas veces referido simplemente como el procesador o el procesador central, el CPU es donde la mayoría de los cálculos toman lugar. En términos de poder de cómputo, el CPU es el elemento más importante de un sistema de cómputo.

DAC - Abreviación de Adquisición de Datos, en su acepción más frecuente se refiere a la adquisición de datos en formato digital desde señales de tipo analógico.

Digital - Describe cualquier sistema basado en datos o eventos discontinuos. Las representaciones digitales consisten de valores medidos a intervalos discretos.

Escalamiento - Operación que implica aumentar o disminuir la proporción de una variable en un factor determinado.

Hardware - De "ferretería" en inglés. En sistemas de cómputo se refiere a los objetos físicos que conforman el sistema, tales como tarjetas de circuito impreso, procesadores, monitores, teclado, etc.

Hexadecimal - Se refiere al sistema numérico de base 16, el cual consiste de 16 símbolos únicos: los números del 0 al 9 y las letras A hasta la F. Los números hexadecimales tienen ya sea un prefijo 0x o un sufijo h.

Interfaz - Medio que permite la conexión de dos entidades separadas.

Interfaz Gráfica - Un programa de interfaz que toma ventajas de las capacidades gráficas de una computadora para hacer al programa más sencillo de utilizar.

Interrupción - Señal que informa a un programa o un dispositivo de que un evento ha ocurrido. Cuando un programa recibe una señal de interrupción éste toma una acción específica (la cual puede ser ignorar la señal). Las interrupciones pueden causar que un programa sea suspendido para dar servicio a esa interrupción.

IRQ - Abreviación de Interrupt Request Line, o Línea de solicitud de interrupción. Las IRQ son generalmente líneas de hardware en un bus de expansión sobre las cuales dispositivos pueden enviar señales de interrupción a un microprocesador.

Muestra - Se refiere a la adquisición de un valor de una señal continua por medio de un sistema discreto de tiempo.

Multitarea - La habilidad de ejecutar más de una tarea al mismo tiempo, siendo la tarea un programa. En multitarea, sólo se involucra un CPU, pero alterna tan rápido de un programa a otro que da la impresión de ejecutar todos los programas al mismo tiempo.

Palabra - Se refiere al tamaño de datos utilizado, en este caso una palabra es equivalente a dos bytes (16 bits).

PC - (1)Abreviación del inglés computadora personal o PC de IBM. La primera computadora personal producida por IBM fue llamada PC, e incrementalmente el término PC paso a significar computadora personal de IBM o computadora personal compatible con IBM. (2) Abreviación de circuito impreso.

Programa - Lista organizada de instrucciones, que cuando son ejecutadas, causan que la entidad de procesamiento se comporte de un modo predeterminado.

Registro - Un área especial de almacenamiento de alta velocidad dentro del CPU. Cualquier dato debe ser presentado en un registro antes de ser procesado.

Reset – Reposicionamiento. Señal que obliga a un dispositivo a entrar en procedimiento de inicialización, parecido al hecho de eliminar la alimentación y reactivarla.

Sistema Operativo - El programa más importante que corre en una computadora. Cualquier computadora de propósito general debe tener un sistema operativo que corra otros programas. Los sistemas operativos realizan tareas básicas, tales como reconocer entrada desde teclado, mandar salida a la pantalla de despliegue, llevar registro de archivos y directorios en el disco, y controlar dispositivos periféricos tales como manejadores de disco e impresoras. Los sistemas operativos proveen una plataforma de software sobre la cual pueden correr otros programas, llamados programas de aplicación. Los programas de aplicación deben ser escritos para correr encima de un sistema operativo en particular.

Software - Se refiere a instrucciones o datos de computadora, existe como ideas conceptos y símbolos, pero carece de substancia, cualquier cosa que pueda ser almacenada electrónicamente es software.

Tarjeta de adquisición de datos - Subsistema cuya tarea es convertir señales de voltaje o corriente acondicionadas a un formato digital el cual puede ser interpretado por la computadora personal.

Thread - Es un flujo secuencial de control dentro de un programa. Un **thread** tiene un principio, un final, una secuencia, y en un momento dado durante el tiempo de ejecución del **thread** sólo hay un punto de ejecución. Un **thread** por si mismo no es un programa. No puede ejecutarse por sí mismo, pero si con un programa. Lo

importante de un **thread** es la posibilidad de que un solo programa ejecute varios **threads** a la vez y que realicen diferentes tareas.

Tiempo Real - De ocurrencia inmediata. "Un sistema computacional de tiempo real puede ser definido como uno que controla un medio ambiente al recibir datos, procesarlos, y regresar los resultados suficientemente rápido como para afectar el medio ambiente en ese tiempo". "Perteneciente al procesamiento de datos por una computadora en conjunción con algún proceso fuera de la misma, en concordancia con requerimientos de tiempo impuestos por el proceso externo."

BIBLIOGRAFÍA

Libros

- *Grady Booch*, **Object-Oriented Analysis and Design With Applications**, Addison-Wesley.
- *Robert L. Boylestad, Louis Nashelsky*, **Fundamentos de electrónica**, Cuarta edición, Prentice Hall.
- *Advantech,* **Function Reference PCL812-PG**, 1999
- *National Instruments,* **NI-DAQ Documentation Set,** 1998
- *Tom Swam,* **Delphi 4 Bible,** IDG Books.
- *Peter Coad, Edward Yourdon*, **Object-Oriented Analysis**, Yourdon Press Computing Series.
- *David E. Johnson, John L. Hilburn, Johnny R. Johnson, Peter D. Scott,* **Análisis Básico de Circuitos Electrónicos,** Quinta Edición, Prentice Hall.
- *Ivar Jacobson,* **Object-Oriented Software Engineering**, Addison-Wesley Object Technology Series

Sitios en Internet

- Rational Software http://www.rational.com/uml/

  Rational Software es la compañía perteneciente a los desarrolladores de Unified Modeling Languaje y que soporta de forma completa la especificación del UML.

APÉNDICE A
CONVERTIDORES ANALÓGICOS-DIGITALES

Los convertidores A/D son dispositivos electrónicos que establecen una relación biunívoca entre el valor de la señal en su entrada y la palabra digital obtenida en su salida. La relación se establece en la mayoría de los casos, con la ayuda de un valor de referencia.

La conversión analógica a digital tiene su fundamento teórico en el teorema de muestreo y en los conceptos de cuantificación y codificación.

Definición;
"Un convertidor Analógico/Digital (A/DC), es un elemento que recibe información de entrada analógica y la transforma a señal digital, en forma de una palabra de "n" bits, cada uno de los niveles lógicos de tensión de entrada es convertida en combinaciones binarias de salida".

Un convertidor analógico digital transfiere información expresada en forma analógica a una forma digital, para ubicar la función de este dispositivo conviene recordar que un sistema combina y relaciona diversos subsistemas que trabajan diferentes tipos de información analógica, como son; magnitudes eléctricas, mecánicas, etc., lo mismo que un micrófono, un graficador, o un motor y estos deberán interactuar con subsistemas que trabajan con informaciones digitales, como una computadora, un sistema lógico, un sistema con microprocesador, con microcontrolador o con algún indicador numérico.

Características básicas de los convertidores

Las características básicas que definen un convertidor analógico digital son en primer lugar, su resolución que depende del número de bits de salida del convertidor, otra característica

básica es la posibilidad de conversión unipolar ó bipolar, una tercera característica la constituye el código utilizado en la información de salida, generalmente los convertidores analógicos digitales operan con el código binario natural ó con el decimal codificado en binario (BCD), el tiempo de conversión es otra característica que definen al convertidor necesario para una aplicación determinada, y se define como el tiempo que necesita para efectuar el máximo cambio de su tensión con un error mínimo en su resolución, otras características que definen al convertidor son; su tensión de referencia, que puede ser interna o externa, si es externa puede ser variada entre ciertos márgenes, la tensión de salida vendrá afectada por este factor, constituyéndose éste a través de un convertidor multiplicador, así mismo deberá tenerse en cuenta, la tensión de alimentación, el margen de temperatura y su tecnología interna.

Cualquier convertidor analógico digital, debe proporcionar una salida digital proporcional a la magnitud expresada por la entrada analógica.

(palabra digital) = (constante proporcional) Vent

La entrada analógica puede ser voltaje ó corriente, la salida digital puede ser cualquier código digital, el código mas común es el binario, aunque puede usarse otro código cualquiera.

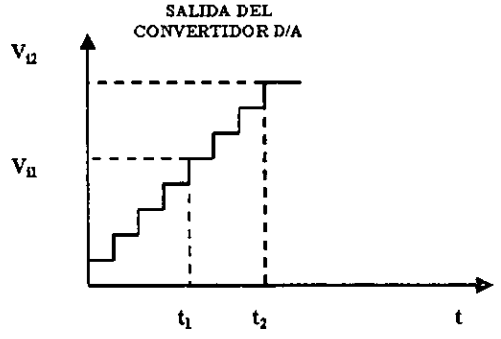


Figura A.1 Gráfica de la salida del convertidor D/A

Tipos de Convertidores Analógicos-Digitales

Se usan un gran número de métodos para convertir señales analógicas a la forma digital.

Los que más se emplean en los circuitos convertidores A/D disponibles en el mercado son cinco:

1. Rampa de escalera
2. Aproximaciones sucesivas
3. Doble rampa
4. Voltaje a frecuencia
5. Paralelo o instantáneo

CONVERTIDORES A/D DE RAMPA DE ESCALERA

Los convertidores más sencillos son de este tipo. Cuando se aplica un comando de inicio o arranque la lógica de control, el voltaje analógico de entrada se compara con una salida de voltaje de un convertidor D/A. Esta salida comienza en cero y se incrementa en un bit menos significativo con cada pulso del reloj.

Siempre que el voltaje de entrada sea mayor que el voltaje de salida del convertidor D/A, el comparador producirá una señal de salida que continúa permitiendo que los pulsos del reloj se alimenten al contador. Sin embargo, cuando el voltaje de salida de ese convertidor es mayor que el voltaje de entrada, la salida del comparador cambia y esta acción evita que los pulsos del reloj lleguen al contador. El estado del contador en ese instante representa el valor de voltaje de entrada en forma digital.

La desventaja de este tipo de convertidores es que, no obstante su simplicidad, es bastante lento y el tiempo de conversión depende de la amplitud de voltaje de entrada.

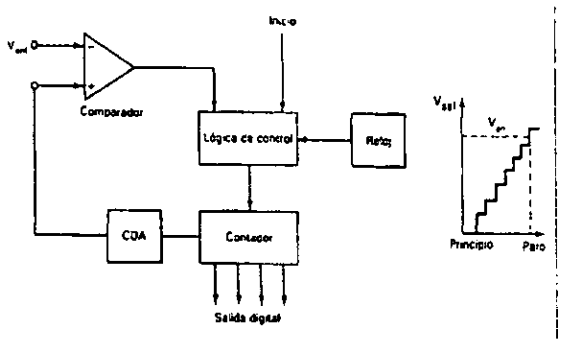


Figura A.2 Diagrama de bloques del convertidor analógico a digital en rampa de escalera

CONVERTIDORES A/D DE APROXIMACIONES SUCESIVAS

Se utilizan ampliamente debido a su combinación de alta resolución y velocidad, ya que pueden efectuar conversiones entre 1 y 50 m s. Sin embargo, son más caros.

La lógica de este convertidor prueba varios códigos de salida y los alimenta al convertidor D/A y a un registro de almacenamiento y compara el resultado con el voltaje de entrada a través del comparador. La operación es análoga a la acción de pesar una muestra en una balanza de laboratorio con pesos estándar en una secuencia binaria. El procedimiento correcto es comenzar con el mayor peso estándar y proseguir en orden hasta el menor. La muestra se coloca en un platillo y el peso mayor se coloca en el otro; si la balanza no se inclina, se deja el peso, y se coloca el siguiente con menor peso. Si la balanza se inclina, se quita el peso mayor y se agrega el siguiente menos pesado. Se usa el mismo procedimiento para el siguiente valor menos pesado y así se prosigue hasta el menor. Después de que se ha probado el enésimo peso y se ha tomado una decisión, se dan por terminadas las mediciones de peso. El total de las pesas que se encuentran en el platillo es la aproximación más cercana al peso de la muestra.

En el convertidor de aproximaciones sucesivas, se implementa el procedimiento de medición de pesos mediante un convertidor D/A, un comparador, un registro de almacenamiento y una lógica de control.

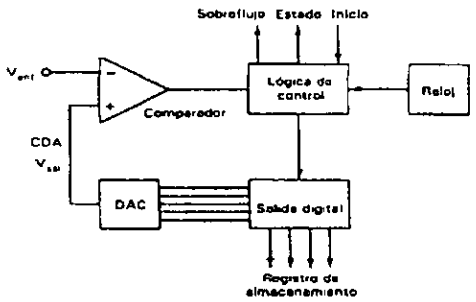


Figura A.3 Diagrama de bloques de un convertidor analógico a digital de aproximaciones sucesivas.

CONVERTIDORES A/D DE DOBLE RAMPA

Se emplean ampliamente en aplicaciones en donde la mayor importancia estriba en la inmunidad al ruido, gran exactitud y economía.

Los convertidores de doble rampa pueden suprimir la mayor parte del ruido de la señal de entrada debido a que emplean un integrador para efectuar la conversión. El rechazo del ruido puede ser infinito para una frecuencia específica del ruido si el primer periodo de integración del convertidor se iguala al periodo del ruido. Por lo tanto, para rechazar el

ruido prevaleciente debido a las líneas de alimentación de 60 Hz, se necesita que T_1 sea de 16.667 ms. Sin embargo, esta ventaja también conduce a tiempos de conversión muy largos. Sin embargo las ventajas de los convertidores de doble rampa los hacen muy adecuados para aplicaciones en las que no sean necesarios tiempos breves de conversión.

Se emplean mucho, en especial en aplicaciones de instrumentos de precisión tales como voltímetros digitales.

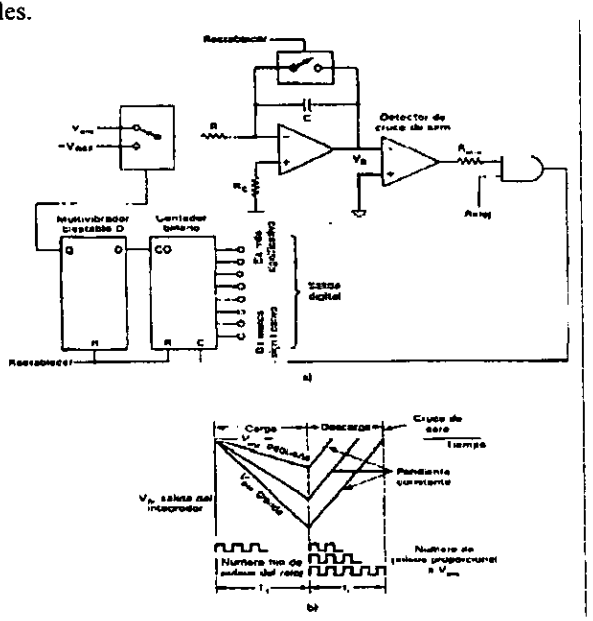


Figura A.4 Convertidor analógico a digital de doble rampa.

CONVERTIDOR DE VOLTAJE A FRECUENCIA.

En este tipo de convertidores, el voltaje de CD de entrada se convierte en un conjunto de pulsos cuya velocidad de repetición (o frecuencia) es proporcional a la magnitud del voltaje de alimentación. Los pulsos se cuentan mediante un contador electrónico en forma semejante al de contar las longitudes de onda con el contador de intervalo de tiempo en el voltímetro digital de doble rampa. Por lo tanto, la cuenta es proporcional a la magnitud del voltaje de entrada.

La parte primordial de esos convertidores es el circuito que transforma el voltaje de CD de entrada a un conjunto de pulsos. Se emplea un integrador para llevar a cabo esta tarea. Las frecuencias típicas del convertidor de voltaje a frecuencia (V/F) quedan en el rango de 10 kHz a 1 kHz. El convertidor muy utilizado de 10 kHz necesita un intervalo de compuerta de 0.025 s para una conversión A/D de 8 bits.

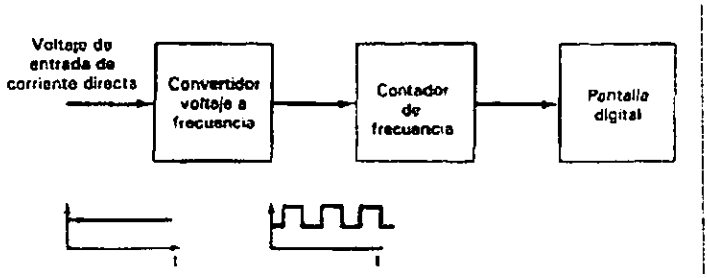


Figura A.5 Diagrama de bloques de un multímetro digital tipo integrador voltaje a frecuencia.

CONVERTIDOR EN PARALELO (O INSTANTÁNEO)

Estos convertidores llevan a cabo las más rápidas conversiones A/D. En esta técnica, el voltaje de entrada se alimenta simultáneamente a una entrada de cada uno de los P comparadores. La otra entrada de cada comparador es un voltaje de referencia. El comparador recibe un valor distinto del voltaje de referencia, comenzando en V_{Rmax}. Empleando el principio del divisor de voltaje y valores iguales de R, el valor del voltaje de referencia V_{Rp} en cada comparador estará dado por:

$$V_{Rp} = V_{Rmax}\, P/Q$$

Siendo

p = número del comparador (de 1 a P)
P = número total de comparadores
Q = número total de resistencias = P + 1

Así, el voltaje de entrada se compara de manera simultánea con valores de voltaje, igualmente espaciados (de 0 a V_{Rmax}).

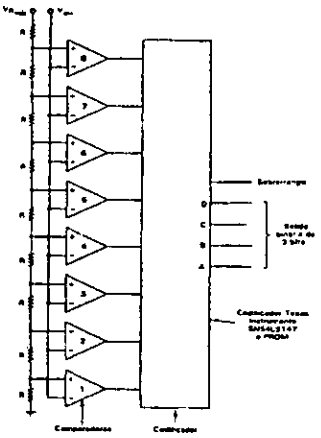


Figura A.6 Convertidor analógico a digital paralelo de tres bits.

Aplicaciones de los convertidores

Las aplicaciones mas significativas del DAC son;

1. ***En instrumentación y control automático,*** son la base para implementar diferentes tipos de convertidores analógico digitales, así mismo, permiten obtener, de un instrumento digital, una salida analógica para propósitos de graficación, indicación o monitoreo, alarma, etc.

2. ***El control por computadora de procesos ó en la experimentación***, se requiere de una interfase que transfiera las instrucciones digitales de la computadora al lenguaje de los ejecutores del proceso que normalmente es analógico.

3. ***En comunicaciones,*** especialmente en cuanto se refiere a telemetría ó transmisión de datos, se traduce la información de los transductores de forma analógica original, a una señal digital, la cual resulta mas adecuada para la transmisión.

APÉNDICE B
TEORÍA ORIENTADA A OBJETOS

Conceptos

OBJETO

Un objeto puede definirse como un concepto adquirido que nos permite sentir y razonar acerca de las cosas del mundo. Un objeto puede ser real o abstracto, por ejemplo una organización, una factura, una figura en un graficador, una pantalla de usuario, un avión, un vuelo de avión, etc.

En el análisis y diseño orientados a objetos (OO), interesa el comportamiento del objeto. Si se construye software, los módulos de software OO se basan en los tipos de objetos. El software que implanta el objeto contiene estructuras de datos y operaciones que expresan dicho comportamiento. Las operaciones se codifican como métodos. Las representación en software OO del objeto es entonces una colección de tipos de datos y objetos.

Entonces, dentro del software orientado a objeto, *un objeto es cualquier cosa, real o abstracta, acerca de la cual almacenamos datos y los métodos que controlan dichos datos.*

Un objeto puede estar compuesto por otros objetos. Estos últimos a su vez también pueden estar compuestos por otros objetos. Esta intrincada estructura es la que permite construir objetos muy complejos.

TIPO DE OBJETO

Los conceptos que poseemos se aplican a tipos determinados de objetos. Por ejemplo, **empleado** se aplica a los objetos que son personas empleadas por alguna organización.

Algunas instancias de empleado podrían ser Juan Pérez, José Martínez, etc. En el análisis orientado a objetos, estos conceptos se llaman tipos de objetos; las instancias se llaman objetos.

Así, un tipo de objeto es una categoría de objeto, mientras que un objeto es una instancia de un tipo de objeto.

Sin embargo, el término objeto tiene diferencias fundamentales con el término entidad, ya que la entidad sólo se refiere a los datos, mientras que objeto se refiere a los datos y a los métodos mediante los cuales se controlan a los propios datos. En OO, la estructura de datos y los métodos de cada tipo de objeto se manejan juntos. No se puede tener acceso o control de la estructura de datos excepto mediante los métodos que forman parte del tipo de objeto.

MÉTODOS

Los métodos especifican la forma en que se controlan los datos de un objeto. Los métodos en un tipo de objeto sólo hacen referencia a la estructura de datos de ese tipo de objeto. No deben tener acceso directo a las estructuras de datos de otros objetos. Para utilizar la estructura de datos de otro objeto, deben enviar un mensaje a éste. El tipo de objeto empaca juntos los tipos de datos y su comportamiento.

En un objeto sus propiedades están representadas por tipos de datos y su comportamiento por métodos.

Un método asociado con el tipo de objeto factura podría ser aquel que calcule el total de la factura. Otro podría transmitir la factura a un cliente. Otro podría verificar de manera periódica si la factura ha sido pagada y, en caso contrario, añadir cierta tasa de interés.

ENCAPSULADO

El empaque conjunto de datos y métodos se llama encapsulado. El objeto esconde sus datos de los demás objetos y permite el acceso a los datos mediante sus propios métodos. Esto recibe el nombre de ocultamiento de información. El encapsulamiento evita la corrupción de los datos de un objeto. Si todos los programas pudieran tener acceso a los datos de cualquier forma que quisieran los usuarios, los datos se podrían corromper o utilizar de mala manera. El encapsulado protege los datos del uso arbitrario y no pretendido.

El encapsulado oculta los detalles de su implantación interna a los usuarios de un objeto. Los usuarios se dan cuenta de las operaciones que puede solicitar del objeto, pero desconocen los detalles de cómo se lleva a cabo la operación. Todos los detalles específicos de los datos del objeto y la codificación de sus operaciones están fuera del alcance del usuario.

Así, encapsulado es el resultado de ocultar los detalles de implantación de un objeto respecto de su usuario.

El encapsulado, al separar el comportamiento del objeto de su implantación, permite la modificación de ésta sin que se tengan que modificar las aplicaciones que lo utilizan.

MENSAJES

Para que un objeto haga algo, le enviamos una solicitud. Ésta hace que se produzca una operación. La operación ejecuta el método apropiado y, de manera opcional, produce una respuesta. El mensaje que constituye la solicitud contiene el nombre del objeto, el nombre de una operación y, a veces, un grupo de parámetros.

La programación orientada a objetos es una forma de diseño modular en la que con frecuencia el mundo se piensa en términos de objetos, operaciones, métodos y mensajes que se transfieren entre tales objetos. Un mensaje es una solicitud para que se lleve a cabo la operación indicada y se produzca el resultado.

CLASE

El término clase se refiere a la implantación en software de un tipo de objeto.

El tipo de objeto es una noción de concepto. Especifica una familia de objetos sin estipular la forma en que se implanten. Los tipos de objetos se especifican durante el análisis OO.

Así, una clase es una implantación de un tipo de objeto. Especifica una estructura de datos y los métodos operativos permisibles que se aplican a cada uno de sus objetos.

HERENCIA

Un tipo de objeto de alto nivel puede especializarse en tipos de objeto de bajo nivel. Un tipo de objeto puede tener subtipos. Por ejemplo, el tipo de objeto **persona** puede tener subtipos **estudiante** y **empleado**. A su vez, el tipo de objeto **estudiante** puede tener como subtipo **estudiante de licenciatura** y **estudiante de postrado**, mientras que **empleado** puede tener como subtipo a **académico** y **administrativo**. Existe de este modo una jerarquía de tipos, subtipos, etc.

Una clase implanta el tipo de objeto. Una subclase hereda propiedades de su clase padre; una sub-subclase hereda propiedades de las subclases; etc. Una subclase puede heredar la estructura de datos y los métodos, o algunos de los métodos, de su superclase. También tiene sus métodos e incluso tipos de datos propios.

EVENTOS

El mundo está lleno de eventos: un cliente solicita un préstamo, el servidor se cae, se termina la tarea, etc.

En el análisis orientado a objetos el mundo se describe en términos de los objetos y sus estados, así como los eventos que modifican esos estados.

Un evento produce un cambio en el estado de un objeto. Los eventos sirven como indicadores de los instantes en que ocurren los cambios de estado.

Para saber de los cambios y reaccionar adecuadamente ante ellos, debemos entender y modelar los eventos.

TIPOS DE EVENTOS

El analista no necesita conocer cada evento que ocurra en una organización: sólo los tipos de eventos.

Por ejemplo, el tipo de evento reservación en lista de espera confirmada es la colección de eventos donde un objeto cambia de una reservación en lista de espera a una reservación confirmada.

Los tipos de eventos indican los cambios sencillos en el estado de un objeto; por ejemplo, cuando se deposita dinero en una cuenta bancaria o se actualiza el sueldo de un trabajador. Básicamente, los tipos de eventos describen las siguientes formas de cambios de estado:

- Un objeto se crea o destruye.
- Un objeto se clasifica o desclasifica como una instancia de un tipo de objeto.
- Un atributo de un objeto cambia.

Los objetos pueden asociar un objeto con otro. Por ejemplo, en la mayoría de las organizaciones, cuando un objeto se clasifica como **empleado**, debe estar asociado con un **departamento**. Un evento clasificará al objeto como empleado. Otro evento creará una asociación entre el objeto empleado y un objeto Departamento (las asociaciones son objetos como los demás).

Algunos eventos requieren que antes ocurran otros. Por ejemplo, antes de cerrar un departamento, todos los empleados deben ser asignados a otra parte, las oficinas que ocupaban deben tener otro uso, etc.

Algunas veces, un evento puede provocar la reacción encadena de otros eventos. Por ejemplo, el cambio de circuito a las conexiones de un avión, puede exigir cambios a varios otros objetos. Una operación hace que los eventos ocurran.

Diseño de la estructura y comportamiento de un objeto.

En el diseño de la estructura y comportamiento de objetos se identifican los componentes siguientes:

- Clases que se implantarán.
- Estructuras de Datos que utilizará cada clase. Se puede hacer un diagrama para representar la estructura de datos.
- Operaciones que ofrecerá cada clase y cuáles serán sus métodos. Se enumeran las operaciones y se especifican los métodos.
- Forma de Implantación de la herencia de clases y efecto sobre las especificaciones de los datos y operaciones.

En el análisis de estructura de objetos, se identifican los tipos de objetos; el diseño de estructura de objetos se centra en la implantación de esos tipos de objetos.

Clase es la implantación de un tipo de objeto. Especifica la estructura de datos y los métodos operativos permitidos que se aplican a cada uno de sus objetos.

La clase especifica la estructura de datos de cada uno de sus objetos y las operaciones que se utilizan para tener acceso a los objetos. La especificación de cómo se llevan a cabo las funciones de una clase se llama método. Los objetos se pueden utilizar exclusivamente con métodos específicos.

Una instancia de una clase, almacena sus datos dentro de él. Se tiene acceso a los datos y se les modifica sólo mediante operaciones permisibles. Esta restricción al acceso se debe al encapsulado. El encapsulado protege los datos del uso arbitrario o no permitido. El acceso o la actualización directa de los datos de un objeto por parte del usuario violaría el encapsulado.

Los usuarios observan el "comportamiento" del objeto en términos de las operaciones que se pueden aplicar a los objetos, así como los resultados de tales operaciones. Estas operaciones forman la interfaz del objeto con sus usuarios.

DIFERENCIA ENTRE OPERACIÓN Y MÉTODO.

Las operaciones son procesos que se pueden solicitar como unidades. Los métodos son especificaciones del procedimiento de una operación dentro de una clase. Es decir, la operación es el tipo de servicio solicitado y el método es su código de programación.

Por ejemplo una operación asociada con la clase pedido podría ser aquella que calcule el total del pedido. El método especificaría la forma de calcular el total. Para esto, el método

podría obtener el precio de cada artículo del pedido al enviar una solicitud a los objetos artículo asociados. A su vez, cada objeto artículo regresaría su precio al método pedido mediante un método de la clase artículo.

Los métodos de una clase controlan solamente a los objetos de esa clase. No pueden tener acceso directo a las estructuras de datos de un objeto en una clase distinta. Para utilizar las estructuras de datos en una clase diferente, deben enviar una solicitud a ese objeto.

HERENCIA DE CLASE.

La generalización es una noción conceptual. La herencia de clase (que sólo se conoce como herencia) es una implantación de la generalización. La generalización establece que las propiedades de un tipo se aplican a sus subtipos.

La herencia de clase hace que la estructura de datos y operaciones sean disponibles para su reutilización por parte de sus subclases. La herencia de las operaciones de una superclase permite que las clases compartan código. La herencia de la estructura de datos permite la reutilización de la estructura.

HERENCIA MÚLTIPLE.

En la herencia múltiple, una clase puede heredar estructuras de datos y operaciones de más de una superclase.

Por ejemplo supóngase que existe un tipo de objeto cuenta, que tiene como subtipos a los tipos de objetos cuenta de cliente y cuenta vencida. A su vez, cuenta de cliente tiene como a subtipo a cuenta de cliente vencida y cuenta vencida también tiene como subtipo a cuenta de cliente vencida.

SELECCIÓN DEL MÉTODO.

Cuando se envía una solicitud a un objeto, el software selecciona los métodos por utilizar. El método no se almacena en el objeto, pues esto causaría una réplica múltiple y pérdida de espacio. En vez de esto, el método se asocia con la clase. El método puede no estar en la clase de la que el objeto es una instancia, sino en una superclase.

En ese caso, el mecanismo de selección buscará la operación en su superclase y en todas las superclases de la jerarquía hasta que lo encuentre, nivel por nivel. Si la encuentra, selecciona la operación. Si la operación no se encuentra en ningún nivel de la superclase, se considera inválida la fuente de la solicitud.

De esta forma, los usuarios sólo deben especificar lo que se debe hacer, dejando que sea el mecanismo de selección el que determine la forma de localizar la operación y la ejecute. El mecanismo de selección deja en manos de la aplicación OO el problema de localizar la operación y la ejecute.

POLIMORFISMO.

Uno de los objetivos principales de las técnicas OO es utilizar otra vez el código. Sin embargo, algunas de las operaciones requieren adaptación para resolver necesidades particulares.

Esta necesidad, se da generalmente entre superclases y subclases, donde una subclase es una especialización de su superclase, y puede requerir alcanzar los mismos objetivos, pero con distintos mecanismos. Por ejemplo, una superclase rectángulo podría tener una operación área cuyo objetivo es calcular el área del rectángulo, definida como la multiplicación de los largos de dos lados contiguos. A su vez, la clase cuadrado es una subclase de rectángulo que también tiene una operación área cuyo objetivo es calcular el área del cuadrado, pero que está definida especialmente para los objetos del tipo cuadrado como la multiplicación del largo de uno de sus lados por sí mismo.

El fenómeno recién descrito se conoce como polimorfismo, y se aplica a una operación que adopta varias formas de implantación según el tipo de objeto, pero cumple siempre el mismo objetivo.

Una de las ventajas del polimorfismo es que se puede hacer una solicitud de una operación sin conocer el método que debe ser llamado. Estos detalles de la implantación quedan ocultos para el usuario; la responsabilidad descansa en el mecanismo de selección de la implantación OO.

ABSTRACCIÓN

La abstracción es una de las formas fundamentales con la que los humanos hacemos frente a la complejidad. La abstracción se origina de un reconocimiento de similitudes entre ciertos objetos, situaciones o procesos en el mundo real, y la decisión de concentrar sobre dichas similitudes e ignorar por el momento las diferencias. Definiremos la abstracción como sigue:

Una abstracción denota las características esenciales de un objeto que se distingue de los otros tipos de objetos y de esta forma proveer de forma breve los límites conceptualmente definidos, relativos a la perspectiva del observador.

Metodologías de desarrollo de software orientado a objetos

La Orientación a Objetos, inicialmente fue un conjunto de técnicas de programación soportadas en el uso de lenguajes especiales (orientados a objetos), pero ha ido más allá de la propia programación hasta convertirse en una metodología genérica y de gran potencia para construir modelos de sistemas, que puede ser aplicada en todas las fases del desarrollo de aplicaciones: análisis, diseño, programación y mantenimiento.

Una metodología de desarrollo de software Orientado a Objetos consta de

1. Conceptos y diagramas
2. Etapas y definición de entregas en cada una de ellas
3. Actividades y recomendaciones

Las metodologías más sobresalientes se listan a continuación:

Metodología	Autores	Año
Unified Modeling Language (**UML**)	Booch, Rumbaugh y Jacobson	1996
Object Oriented Design (**OOD**)	Grady Booch	1991
Object-Oriented Software Enginnering (**OOSE**)	Ivar Jacobson	1992
Object Modeling Technique (**OMT**)	James Rumbaugh	1991
Yourdon & Coad's Object-Oriented Analysis and Design (**OOA/OOD**)	Peter Coad y Edward Yourdon	1991
Fusion Method	Derek Coleman	1993

Tabla B.1 Tabla de metodologías OO actuales

Existen muchas aproximaciones de desarrollo de software que utilizan modelos orientado a objetos, pero algunas de estas carecen de suficientes abstracciones y tienen una baja relación en los detalles de implantación.

Otros métodos de programación orientados ponen un escaso énfasis en la estructura de datos y constantes, que son muy importantes para aplicaciones de base de datos.

De las metodologías listadas se hará una breve revisión de las cuatro primeras por tener mayor impacto en la actualidad, de las últimas dos solo se hará una descripción general.

UML (Unified Modeling Language)

Esta notación de sistemas orientados a objetos ha involucrado el trabajo de Grady Booch, James Rumbaugh, Ivar Jacobson, y de Rational Software Corporation. Estos renombrados personajes de la computación fusionaron sus respectivas tecnologías en un modelo estandarizado único.

ENFOQUE GENERAL

UML se quiere convertir en un lenguaje estándar con el que sea posible modelar todos los componentes del proceso de desarrollo de aplicaciones. No pretende definir un modelo estándar de desarrollo, sino únicamente un lenguaje de modelado. Otros métodos de modelado como OMT o Booch OOD sí definen procesos concretos.

En UML los procesos de desarrollo son diferentes según los distintos dominios de trabajo. Las diferencias son muy marcadas y afectan a todas las fases del proceso. El método del UML recomienda utilizar los procesos que otras metodologías tienen definidos.

En la especificación del UML podemos comprobar que una de las partes que lo componen es un *metamodelo* formal. Un *metamodelo* es un modelo que define el lenguaje para expresar otros modelos. Un modelo en OO es una abstracción cerrada semánticamente de un sistema y un sistema es una colección de unidades conectadas que son organizadas para realizar un propósito específico. Un sistema puede ser descrito por uno o más modelos, posiblemente desde distintos puntos de vista.

Además de los conceptos extraídos de métodos OMT, OOD y OOSE se han añadido otros nuevos que vienen a suplir carencias antiguas de la metodología de modelado. Estos nuevos conceptos son los siguientes:

- Definición de estereotipos: un estereotipo es una nueva clase de elemento de modelado que debe basarse en ciertas clases ya existentes en el metamodelo y constituye un mecanismo de extensión del modelo.
- Responsabilidades.
- Mecanismos de extensibilidad: estereotipos, valores etiquetados y restricciones.
- Tareas y procesos.
- Distribución y concurrencia
- Patrones/Colaboraciones.
- Diagramas de actividad
- Clara separación de tipo, clase e instancia.
- Refinamiento, para manejar relaciones entre niveles de abstracción.
- Interfaces y componentes.

TIPOS DE DIAGRAMAS UML

UML define nueve tipos de diagramas: clase (paquete), objeto, caso de uso, secuencias, colaboración, diagrama de estados, actividades, componentes y despliegue. Los diagramas se describen brevemente a continuación:

Diagramas de Clase

Los diagramas de clase son el soporte de casi todo método orientado a objetos. Este describe la estructura estática del sistema.

Representada por un rectángulo con tres divisiones internas, son los elementos fundamentales del diagrama. Una clase describe un conjunto de objetos con características y comportamiento idéntico.

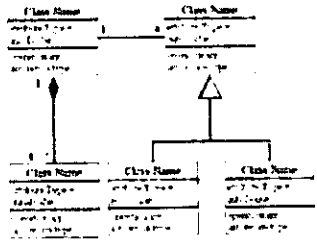


Figura B.1 Diagramas de clase

Diagramas de Paquete

Un paquete es una forma de agrupar clases (u otros elementos en otro tipo de diagramas) en modelos grandes. Pueden tener asociaciones de dependencia o de generalización entre ellos.

Los diagramas de paquete organizan elementos de un sistema en grupos relacionados para minimizar la dependencia entre paquetes.

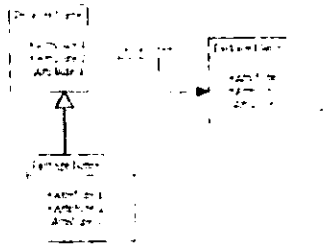


Figura B.2 Diagramas de paquete

Diagramas de Objeto

Los diagramas de objeto describen la estructura estática del sistema en un momento determinado. Muestra el conjunto de clases y objetos importantes que hacen parte de un sistema, junto con las relaciones existentes entre estas clases y objetos. Muestra de una manera estática la estructura de información del sistema y la visibilidad que tiene cada una de las clases, dada por sus relaciones con las demás en el modelo.

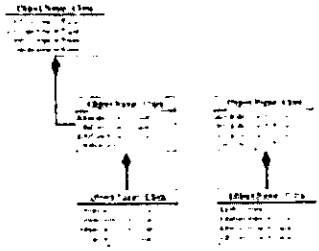


Figura B.3 Diagramas de objeto

Diagramas de casos de Uso

Los Casos de Uso son servicios o funciones que son suministradas por el sistema a los usuarios.

Un diagrama de Casos de Uso muestra las distintas operaciones que se esperan de una aplicación o sistema y cómo se relaciona con su entorno (usuarios u otras aplicaciones).

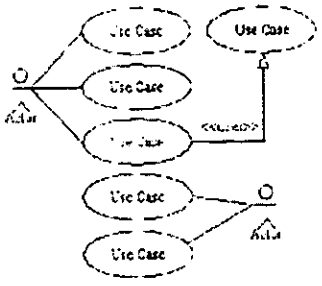


Figura B.4 Diagramas de casos de uso

Diagramas de Secuencia

Un diagrama de secuencia muestra la interacción de un conjunto de objetos en una aplicación a través del tiempo. Esta descripción es importante porque puede dar detalle a los casos de uso, aclarándolos al nivel de mensajes de los objetos existentes, como también muestra el uso de los mensajes de las clases diseñadas en el contexto de una operación.

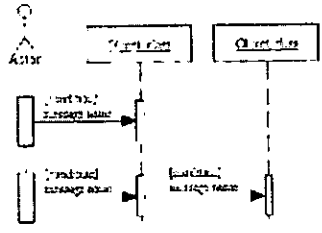


Figura B.5 Diagramas de secuencia

Diagramas de colaboración

Un diagrama de colaboración es una forma de representar interacción entre objetos, alterna al diagrama de secuencia. A diferencia de los diagramas de secuencia, pueden mostrar el contexto de la operación (cuáles objetos son atributos, cuáles temporales, ...) y ciclos en la ejecución.
Los diagramas de colaboración representan una combinación de la información obtenida de los diagramas de clase, secuencia y casos de uso que describe la estructura estática y el comportamiento dinámico del sistema.

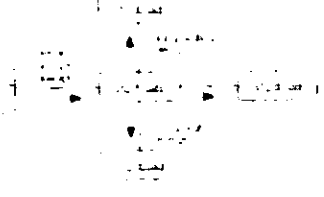


Figura B.6 Diagramas de colaboración

Diagramas de estado

Los diagramas de estado describen el comportamiento dinámico del sistema en respuesta a estímulos externos. Los diagramas de estados son especialmente útiles en el modelado de objetos reactivos, cuyos estados son activados por eventos específicos.

Muestra el conjunto de estados por los cuales pasa un objeto durante su vida en una aplicación, junto con los cambios que permiten pasar de un estado a otro.

Un estado puede descomponerse en subestados, con transiciones entre ellos y conexiones al nivel superior. Las conexiones se ven al nivel inferior como estados de inicio o fin, los cuales se suponen conectados a las entradas y salidas del nivel inmediatamente superior.

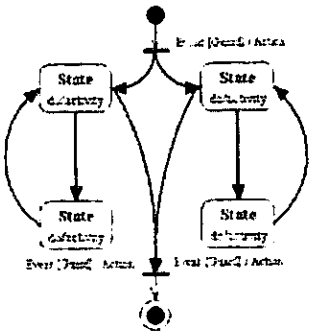


Figura B.7 Diagramas de estado

Diagramas de actividad

Un diagrama de actividad ilustra la naturaleza dinámica de un sistema derivado del modelado del flujo de control de actividad en actividad. Una actividad representa una operación sobre alguna clase del sistema que da como resultado un cambio en el estado del sistema. Típicamente, los diagramas de actividad son usados para modelar el flujo de trabajo o procesos del negocio y operaciones internas. Ya que el diagrama de actividad es un tipo especial de diagrama de estados, éste utiliza algunas convenciones de modelado similares.

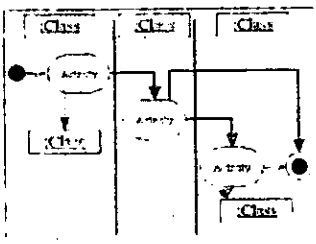


Figura B.8 Diagramas de actividad

Diagramas de componentes

Los diagramas de componentes describen la organización de los componentes físicos de software.

Un componente es un fragmento de código software (un fuente, binario o ejecutable) que se utiliza para mostrar dependencias en tiempo de compilación.

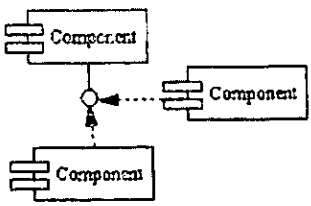


Figura B.9 Diagramas de componentes

Diagramas de plataformas o despliegue

Los diagramas de despliegue muestran la configuración de los componentes hardware, los procesos, los elementos de procesamiento en tiempo de ejecución y los objetos que existen en tiempo de ejecución. En este tipo de diagramas intervienen nodos, asociaciones de comunicación, componentes dentro de los nodos y objetos que se encuentran a su vez dentro de los componentes. Un *nodo* es un objeto físico en tiempo de ejecución, es decir una máquina que se compone habitualmente de, por lo menos, memoria y capacidad de procesamiento, a su vez puede estar formado por otros componentes.

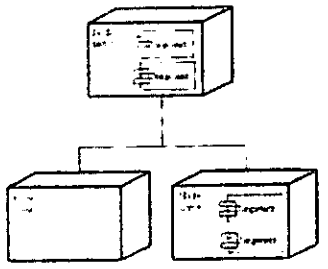


Figura B.10 Diagramas de plataformas o despliegue

OOD (Object Oriented Design)

El Análisis y diseño orientado a objetos de Grady Booch, es uno de los precursores del Lenguaje Unificado de Modelado (**UML**). La metodología de Booch usa los siguientes tipos de diagramas para describir las decisiones de análisis y diseño, tácticas y estratégicas, que deben ser hechas en la creación de un sistema orientado por objetos: clase, objeto, transición de estados, interacción, módulos y procesos.

ENFOQUE GENERAL

El libro de Booch comienza con una discusión de los conceptos de la teoría orientada a objetos, la complejidad y los atributos de un sistema con estructura compleja. La construcción del modelo es muy importante en un sistema complejo. Booch propone 4 modelos de desarrollo orientado a objetos: lógico, físico, estático y dinámico.

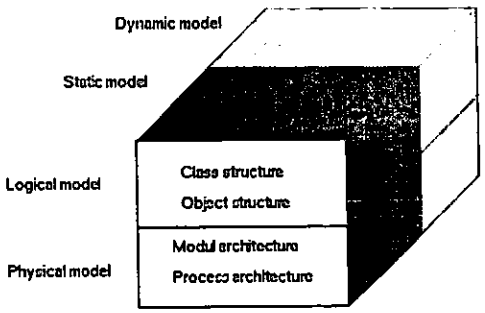


Figura B.11 Modelos del desarrollo orientado a objetos según Booch

Después el plantea el modelo de objeto y muestra que los sistemas modelo con objetos se derivan de los atributos de los sistemas complejos bien estructurados. La parte más difícil del análisis orientado a objetos de acuerdo con Booch, es la identificación de las clases y los objetos. Una clasificación apropiada es una cuestión fundamental en el análisis y diseño orientados a objetos.

El OOD esta dividido en micro y macro procesos. El micro-proceso básicamente representa las actividades cotidianas de los desarrolladores y consiste en cuatro pasos principales (no secuenciales):

- Identificar las clases y objetos con un cierto nivel de abstracción
- Identifica la semántica de los objetos y las clases
- Identifica las relaciones entre clases y objetos
- Implantación de las clases y objetos

El macro-proceso es utilizado para controlar al micro-proceso. Este dirige las actividades para el equipo entero de desarrollo con una escala de tiempo basada en meses o semanas. Planteando cinco actividades:

- **Conceptualización**, en la cual los requerimientos centrales son establecidos.
- **Análisis**, en el que el modelo del comportamiento deseado es elaborado.
- **Diseño**, en el que la arquitectura es creada.
- **Evolución**, que involucra la implantación.
- **Mantenimiento**, en la cual gestiona la evolución postventa, es decir se realizan cambios al sistema a medida que se añaden nuevos requisitos o se eliminan errores persistentes.

TIPOS DE DIAGRAMAS OOD

Diagrama de Clases

Un diagrama de clases representa una vista de la estructura de clases del un sistema. Durante el análisis se utilizan para indicar las misiones y las responsabilidades comunes de las entidades que caracterizan el comportamiento de un sistema.

Los elementos esenciales que componen un diagrama de clases son las clases y sus relaciones básicas

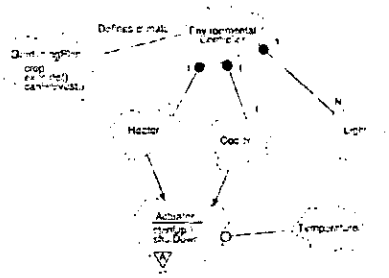


Figura B.12 Diagramas de clases

Diagramas de Objetos.

Muestra objetos en el sistema y su relación lógica. Pueden ser diagramas de escenario, donde se muestra cómo colaboran los objetos en cierta operación; o diagramas de instancia, que muestra la existencia de los objetos y las relaciones estructurales entre ellos.

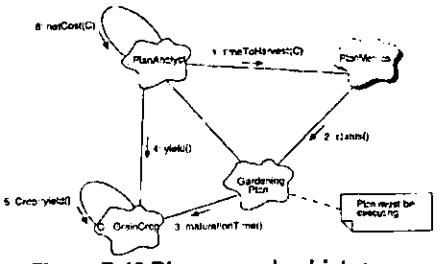


Figura B.13 Diagramas de objetos

Diagramas de transición de estados.

Un diagrama de transición de estados se utiliza para mostrar el espacio de estados de una clase determinada, los eventos que provocan una transición de un estado a otro y las acciones que resultan en ese cambio de estado. Los dos elementos esenciales de un diagrama de transición de estados son los estados y las transiciones entre estados.

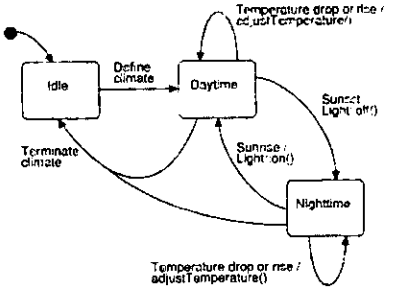


Figura B.14 Diagramas de transición de estados

Diagramas de Interacción

Un diagrama de interacción se utiliza para realizar una traza de la ejecución de un escenario en el mismo contexto que un diagrama de objetos. Un diagrama de interacción es otra forma de representar un diagrama de objetos, pero estos son mejores que los diagramas de objeto para capturar la semántica de los escenarios en un momento temprano del ciclo de vida del desarrollo.

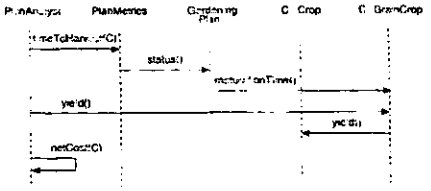


Figura B.15 Diagramas de interacción

Diagramas de módulos.

Muestra la localización de objetos y clases en módulos del diseño físico de un sistema. Un diagrama de módulos representa parte o la totalidad de la arquitectura de módulos del sistema. Los dos elementos esenciales de un diagrama de módulos son los módulos y sus dependencias.

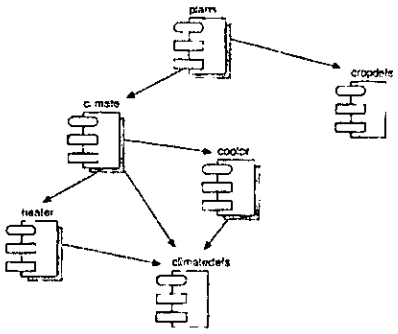


Figura B.16 Diagramas de módulos

Diagramas de procesos.

Se usa un diagrama de procesos para mostrar la asignación de procesos a procesadores en el diseño físico de un sistema. Durante el desarrollo se usan diagramas de proceso para indicar la colección física de procesadores y dispositivos que sirven como plataforma de ejecución del sistema. Los tres elementos esenciales de un diagrama de procesos son los procesadores, los dispositivos y sus conexiones.

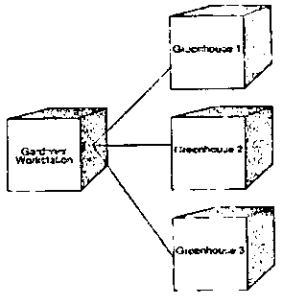


Figura B.17 Diagramas de procesos

OOSE (Object-Oriented Software Engineering)

OOSE combina tres diferentes técnicas que han sido utilizadas por mucho tiempo. La primera técnica es la programación orientada a objetos, usando principalmente conceptos de encapsulamiento, herencia y relaciones entre clases e instancias.

ENFOQUE GENERAL

En OSSE los modelos sirven para comprender el sistema y obtener una arquitectura de sistema bien definida. Aquí ser utilizan los modelos de casos de uso que sirve como modelo

central mediante el cual todos los otros sistemas se derivan. Un modelo de caso de uso describe la funcionalidad completa del sistema identificando todo lo exterior que interactúa con el sistema.

- El modelo de caso de uso es la base en la fase del análisis, construcción y pruebas.
- El propósito del análisis es comprender el sistema de acuerdo a sus requerimientos funcionales.
- Los objetos son encontrados, organizados y las interacciones entre objetos son descritas. Las operaciones de objetos y las vistas internas son descritas también durante el análisis.
- La construcción abarca el diseño y la implantación en código fuente.
- En las pruebas el sistema es verificado, esto significa que las correcciones al sistema son hechas de acuerdo a las especificaciones.

Object Modeling Technique (OMT)

OMT pone énfasis en la importancia del modelo y uso de modelo para lograr una abstracción , en el cual el análisis esta enfocado en el mundo real para un nivel de diseño, también pone detalles particulares para modelado de recursos de la computadora. Esta Tecnología puede ser aplicado en varios aspectos de implementación incluyendo archivos, base de datos relacionales, base de datos orientados a objetos. OMT esta construido alrededor de descripciones de estructura de datos, constantes, sistemas para procesos de transacciones.

ENFOQUE GENERAL

Desde que la comunidad de programación orientada a objetos tuvo la noción de incorporar el pensamiento de que los objetos son entidades coherentes con identidad estado y conducta, estos objetos pueden ser organizados por sus similitudes y sus diferencias, puestas en uso en herencia y polimorfismo.

Desde el modelado de información, tuvo que ser adoptada la noción de entidades que son conectadas con entidad relación, los modelos de relación son declarativas, imperativas.

OMT pone énfasis en especificaciones declarativas de la información, para capturar limpiamente los requerimientos, especificaciones imperativas para poder descender prematuramente en el diseño, declaraciones que permiten optimizar los estados, además provee un soporte declarativo para una directa implementación de DBMS.

Yourdon & Coad's Object-Oriented Analysis and Design (OOA/OOD)

Coad y Yourdon describen un método de Análisis Orientado a Objetos basado en cinco actividades principales:

- Definición de las clases y objetos
- Identificación de estructuras
- Identificación de temas
- Definición de atributos
- Definición de servicios

Estas actividades son usadas para construir cada capa de un modelo de objetos de *cinco niveles*. Los objetos existen en el ámbito del problema. Las clases son abstracciones de objetos. Los objetos son instancias de clases. La primera tarea del método es identificar las clases y los objetos.

La segunda tarea del método es identificar las estructuras. Se reconocen dos tipos de estructuras: *estructuras de generalización-especialización* y *estructuras globales*. El primer tipo de estructura es como un árbol genealógico: es posible la herencia entre los miembros de una estructura. El segundo tipo de estructura es utilizado para modelar relaciones de entidades. Los modelos grandes y complejos pueden necesitar una organización *temática*, con cada tema dedicado a un aspecto particular del problema.

ENFOQUE GENERAL

Los atributos caracterizan a cada clase. Cada objeto tendrá un valor para ese atributo. Los servicios definen lo que los objetos hacen. Definir los servicios es equivalente a definir las funciones del sistema.

La importancia fundamental del método de Coad y Yourdon es su descripción breve y concisa, así como el uso de textos generales como fuentes para las definiciones; de modo que las definiciones se enmarcan dentro del sentido común y reducen el empleo de modismos.

La debilidad principal del método es su notación compleja, la cual es difícil de utilizar sin el apoyo de una herramienta.

Fusion Method

Este método está considerado como uno de los métodos de desarrollo de "Segunda Generación". Proporciona elementos de desarrollo para reutilización y reingeniería. Los siguientes métodos o técnicas han influido en Fusión:

- **OMT** (Rumbaugh et al., 1991): El modelo Objeto es casi similar que en OMT. El modelo operacional es análogo al modelo funcional en OMT; los diagramas de flujo de datos de OMT no son apropiados de acuerdo con Coleman y han sido formalizados con pre-condiciones y post-condiciones.

- **Métodos formales**: Pre y post-condiciones son adoptados para describir formalmente qué es lo que hace un sistema.

- **CRC**: CRC extendido con información de comunicación ha influenciado en gráficas de interacción de objetos.

- **Diseño OO con Aplicaciones** (Booch, 1991):La visibilidad de las gráficas son influenciadas por información de visibilidad en los diagramas Objeto de Fusión está basado en un pequeño pero comprensivo conjunto de técnicas de diagramación bien definidas para la descripción de las etapas de análisis y diseño. Fusión consiste de 3 fases: análisis, diseño e implementación. Fusión también proporciona reglas para verificar la consistencia e integridad del desarrollo de los modelos.

Enfoque General

En la fase de análisis se define el comportamiento propuesto del sistema. Los modelos en esta fase describen las clases de objetos, las relaciones entre clases, las operaciones que pueden ser ejecutadas en el sistema y permite la realización de secuencias de esas operaciones. En la fase de diseño, los modelos producidos muestran la forma en que las operaciones del sistema son implementadas por objetos interactivos, referencias entre clases, relaciones de herencia, atributos de clases y operaciones en clases.

APÉNDICE C
REFERENCIA DE CLASES

TConvertidor

Unidad: UConvertidor

TObject • TConvertidor

TConvertidor es la clase genérica que hace posible el manejo de diferentes convertidores, con la misma interfaz grafica. Cada una de los elementos necesarios para esta clase fueron adecuados para formar la siguientes propiedades

Las propiedades son las encargadas de proteger a las variables internas de una mal manejo externo, es por ello que varias de estas son manipuladas mediante algunos de los métodos listados después de esta sección.

PROPIEDADES

Las siguiente tabla lista las propiedades de TConvertidor:

Propiedad	Visibilidad
Running	public, read-only
GainStrings	public, read-only
StartChanel	public
EndChanel	public
Gain	public
GainStr	public
GainFactor	public, read-only
MaxChanel	public, read-only
Frec	public

Propiedad	Visibilidad
MaxFrec	public, read-only

Tabla C.1 Propiedades de TConvertidor

Running *TConvertidor*

Running es una bandera que indica si el convertidor se encuentra en operación.

```
property Running : boolean
```

Descripción Coloca un valor de falso si el convertidor no se encuentra adquiriendo y un valor de verdadero en caso contrario. Esta propiedad puede usarse para leer el valor de dicha bandera.

GainStrings *TConvertidor*

GainStrings es una arreglo de registros que contienen la información de las ganancias.

```
TGanancia = record
    s : string;
    min : double;
    max : double;
end;

GainArray = array of TGanancia;

property GainStrings  : GainArray
```

Descripción Esta propiedad es utilizada para leer el arreglo de Ganancias correspondientes al convertidor. La ganancia consta de un registro con los siguientes elementos:

s	cadena que contiene la ganancia en formato +/- *voltaje*
min	valor mínimo de ganancia –*voltaje*
max	valor máximo de ganancia +*voltaje*

StartChanel *TConvertidor*

StartChanel es el canal inicial para la adquisición de datos.

property StartChanel :integer

Descripción Esta propiedad es utilizada para leer o escribir el canal inicial de la adquisición en curso. Junto con el canal final sirve para controlar el número y la cantidad de canales a ser muestreados.

EndChanel *TConvertidor*

EndChanel es el canal final para la adquisición de datos.

property EndChanel :integer

Descripción Esta propiedad es utilizada para leer o escribir el canal final de la adquisición en curso.

Gain *TConvertidor*

Gain es la ganancia.

property Gain :integer

Descripción Es un índice de escala que permite seleccionar el factor de ganancia. Esta propiedad es utilizada para leer o escribir la ganancia.

GainStr *TConvertidor*

GainStr es la ganancia en formato de cadena.

property GainStr :integer

Descripción Para un valor de ganancia existe su valor de cadena correspondiente, con esta propiedad el valor de cadena asociado a la ganancia de la adquisición actual puede ser leído y escrito.

GainFactor *TConvertidor*

GainFactor es el factor de ganancia.

property GainFactor : double

Descripción Indica la relación entre la entrada analógica y la señal digital transformada. Esto es la razón entre una palabra de "n" bits representada por un número

entero y cada uno de los niveles lógicos de tensión de entrada. Esta propiedad es utilizada para leer el factor de ganancia.

MaxChanel *TConvertidor*

MaxChanel es el canal máximo.

property MaxChanel : integer

Descripción Dependiendo de la tarjeta de adquisición de la que se trate, se tiene un número fijo de canales. El canal máximo tiene el valor del último canal accesible para la tarjeta especificada. Esta propiedad es utilizada para leer o escribir el canal máximo.

Frec *TConvertidor*

Frec es la frecuencia.

property MaxFrec : double

Descripción Indica el número de muestras por segundo a la que están trabajando los canales del convertidor, con esta propiedad el valor de frecuencia de la adquisición actual puede ser leído y escrito. La frecuencia debe de ser la misma para todos los canales.

MaxFrec *TConvertidor*

MaxFrec es la frecuencia máxima.

property MaxFrec : double

Descripción Indica el número máximo de muestras por segundo que el sistema soporta por cada canal. Esta propiedad es utilizada para leer o escribir la frecuencia máxima.

MÉTODOS

Los métodos son los encargados de llevar a cabo las acciones fundamentales del convertidor, detectando los errores cuando estos ocurren, previniendo así un mal funcionamiento.

Las siguiente tabla lista los métodos de TConvertidor:

Método	Visibilidad
SetStartChanel	protected
SetEndChanel	protected
SetGain	protected
SetGainStr	protected
GetGainStr	protected
GetIndexGain	protected
SetFrec	protected
StartHardware	protected
StopHardware	protected
Create	protected
Destroy	protected
Start	protected
Stop	protected
Reset	protected
GetGainStrings	public
Volts	public
Scaled	public
Sample	public
NSample	public
Vsample	public
NVSample	public
Ssample	public
GetStrError	public

Tabla C.2 Métodos de TConvertidor

Los métodos que se plantean para el objeto convertidor son los siguientes:

SetStartChanel *TConvertidor*

Establece el canal inicial a ser muestreado.

Descripción Este método verifica que el canal de entrada (valor entero) no sea negativo, mayor que el canal final o mayor que el canal máximo permitido y posteriormente lo asigna. Si una de las condiciones no se cumple genera un

código de error. Este método modifica la propiedad ***StartChanel***, la cual podrá ser empleada sólo en operaciones de lectura y en ningún caso podrá modificarse directamente.

Parámetros *Entrada* - valor entero – *número del canal inicial.*

SetEndChanel *TConvertidor*

Establece el canal final a muestrear.

Descripción Este método verifica que el canal de entrada (valor entero) no sea negativo, menor que el canal inicial o mayor que el canal máximo permitido y posteriormente lo asigna. Si una de las condiciones no se cumple genera un código de error. Este método modifica la propiedad ***EndChanel***, la cual podrá ser empleada sólo en operaciones de lectura y en ningún caso podrá modificarse directamente.

Parámetros *Entrada* - valor entero – *número del canal final.*

SetGain *TConvertidor*

Establece la ganancia, el factor de ganancia.

Descripción Recibe en valor entero el número de la ganancia asociada y en base a éste establece el factor de ganancia con la siguiente operación:

FactorDeGanancia = abs(VoltajeMáximo-VoltajeMínimo)/4096

Parámetros *Entrada.*- valor entero – *número de ganancia*

SetGainStr *TConvertidor*

Establece la ganancia en formato de cadena.

Descripción El valor de entrada lo envía a la función **getIndexGain** para obtener el índice asociado a la ganancia, una vez hecho esto se procede a asignar dicho valor a la ganancia actual válida.

Parámetros *Entrada.*- valor de cadena - *Ganancia.*

GetGainStr *TConvertidor*

Obtiene la ganancia como cadena.

Descripción En base a la ganancia actual válida obtiene la cadena de ganancia correspondiente.

Parámetros *Salida.*- valor de cadena - *Ganancia.*

GetIndexGain *TConvertidor*

Obtiene el índice asociado a la ganancia.

Descripción Este método recibe una cadena de ganancia cuando esta es encontrada dentro del arreglo de ganancias el índice de dicha ganancia es entregado.

Parámetros *Entrada.*- valor de cadena - *Ganancia.*
Salida.- valor entero – *número de ganancia.*

SetFrec *TConvertidor*

Establece la frecuencia de muestreo.

Descripción Verifica si la frecuencia máxima es menor que cero, si es así la frecuencia recibida es asignada automáticamente; además de que se verifica que el valor recibido no sea negativo y que no exceda la frecuencia máxima permitida. Una vez que esto ha sido verificado el valor de frecuencia es asignado. Este método modifica la propiedad ***Frec***, la cual podrá ser empleada sólo en operaciones de lectura y en ningún caso podrá modificarse directamente.

Parámetros *Entrada.*- valor de punto flotante – *Frecuencia.*

StartHardware *TConvertidor*

Comienza la adquisición de muestras a nivel físico o de hardware.

Descripción Método abstracto que debe ser definido en las subclases.

StopHardware *TConvertidor*

Detiene la adquisición de muestras a nivel físico o de hardware.

Descripción Método abstracto que debe ser definido en las subclases.

Create *TConvertidor*

Crea una instancia de la clase TConvertidor.

Descripción Haciendo una llamada a Create logra instanciar un Convertidor en tiempo de ejecución. Inicializa las siguientes propiedades:

- ***Running*** al valor de *falso*.
- ***MaxChanel*** al valor de –1 que significa *no definido*.
- ***StartChanel*** al valor de 0.
- ***EndChanel*** al valor de 0.
- ***MaxFrec*** al valor de –1 que significa *no definido*.
- ***Frec*** al valor de 100 (frecuencia por default).
- ***Gain*** al valor de –1 que significa *no definido*.
- ***GainFactor*** al valor de 2*5.0/4096.
- ***ScaleFactor*** al valor de 1.
- ***OffsetFactor*** al valor de 1.

Y también crea una instancia de TCircular y la asigna a la variable buffer.

Parámetros *Entrada* - ninguno.
Salida - ninguno.

Destroy *TConvertidor*

Destruye una instancia de TConvertidor.

Descripción Hace una llamada a **stop**, para después liberar la instancia de TCircular llamada buffer.

Parámetros *Entrada* - ninguno.
Salida - ninguno.

Start *TConvertidor*

Comienza la adquisición de muestras a nivel software.

Descripción Se encarga de hacer una llamada a **starthardware** la cual inicia la adquisición de muestras en la tarjeta, también deberá habilitar la bandera de carga de muestras, para llevar el control de la adquisición en caso de fallo.

Parámetros *Entrada* - ninguno.
Salida - ninguno.

Stop *TConvertidor*

Detiene la adquisición de los datos a nivel software.

Descripción Será la encargada de hacer una llamada a **stopthardware** quien está encargada de iniciar la adquisición de muestras en la tarjeta, también deberá deshabilitar la bandera de carga de muestras.

Parámetros *Entrada* - ninguno.
Salida - ninguno.

Reset *TConvertidor*

Detiene el proceso de conversión si lo hubiera y limpia el buffer de datos.

Descripción Al igual que **stop,** hace una llamada a **stophardware**, con la diferencia de que esta función también se encargará de limpiar el buffer de datos adquiridos.

Parámetros *Entrada* - ninguno.
Salida - ninguno.

GetGainStrings *TConvertidor*

Obtiene las cadenas correspondientes a las ganancias de la tarjeta.

Descripción Método abstracto que debe ser definido en las subclases.

Parámetros *Entrada* - ninguno.
Salida - ninguno.

Volts *TConvertidor*

Convierte una muestra en un valor en voltaje con base en la ganancia elegida.

Descripción Método abstracto que debe ser definido en las subclases.

Scaled *TConvertidor*

Convierte una muestra en un valor con base en un escalamiento.

Descripción Método abstracto que debe ser definido en las subclases.

Sample *TConvertidor*

Extrae una muestra del buffer destinado al almacenamiento temporal de muestras

Descripción Comprueba que el convertidor se encuentre en operación y obtiene la muestra del buffer colocando esta en el valor entero recibido entregando el valor de verdadero. Si el convertidor no se encontrara operando entregaría un valor de falso.

Parámetros *Entrada* - valor entero – *variable que contendrá la muestra adquirida.*
Salida - valor booleano – *status del convertidor.*

nSample *TConvertidor*

Obtiene una cantidad específica de muestras del buffer destinado al almacenamiento temporal de muestras.

Descripción Este método al igual que **Sample** comprueba que el convertidor se encuentre en operación si es así regresa un valor de verdadero, y de otra manera regresa falso; con la diferencia de que obtiene n número de muestras, donde n es un entero. Las muestras adquiridas son puestas en un arreglo.

Parámetros *Entrada* - valor entero – *número de muestras a ser adquiridas.*
arreglo de enteros – *contendrá las muestras adquiridas.*
Salida - valor booleano – *status del convertidor.*

VSample *TConvertidor*

Extrae una muestra transformada a voltaje del buffer destinado al almacenamiento temporal de muestras.

Descripción Obtiene la muestra del buffer y a continuación manda llamar a la función **Volts,** dando como resultado el valor convertido en punto flotante si el convertidor se encuentra adquiriendo regresa un valor de verdadero, de otra manera regresa falso.

Parámetros *Entrada* - valor de punto flotante – *guardará la muestra en voltaje obtenida.*
Salida - valor booleano – *status del convertidor.*

nVSample *Tconvertidor*

Extrae una cantidad específica de muestras y las transforma a voltajes.

Descripción Obtiene n muestras del buffer y a continuación manda llamar a la función **Volts** en un ciclo, dando como resultado el valor convertido en punto flotante si el convertidor se encuentra adquiriendo regresa un valor de verdadero, de otra forma regresa falso.

Parámetros *Entrada* - valor entero – *número de muestras en voltaje a ser adquiridas.*
arreglo de valores de punto flotante – *guardará las muestras en voltaje obtenidas.*
Salida - valor booleano – *status del convertidor.*

sSample *TConvertidor*

Obtiene el valor del canal convertido y luego lo escala.

Descripción Comprueba que el convertidor se encuentra en operación si es así retorna verdadero, y de otra manera regresa falso. El canal convertido es obtenido del buffer y a continuación el valor es pasado a la función **Scaled**, de lo que se obtiene el valor adquirido y escalado en punto flotante.

Parámetros *Entrada* – valor entero –*muestra a ser escalada.*
valor entero – *número del canal al que pertenece la muestra.*
Salida - valor de punto flotante – *muestra escalada.*

TConvAT_MIO_64F5

Unidad: UConvAT_MIO_64F5

TObject • TConvertidor • TConvAT_MIO_64F5

MÉTODOS

Las siguiente tabla lista los métodos de TConvAT_MIO_64F5:

Método	Visibilidad
StartHardware	protected
StopHardware	Protected
encodeGain	Protected
Create	Public
Destroy	Public
GetGainStrings	Public
Volts	Public
Scaled	Public

Tabla C.3 Métodos de TConvAT_MIO_64F5

StartHardware *TConvAT_MIO_64F5*

Comienza la adquisición de muestras a nivel físico o de hardware.

Descripción Determina el valor más adecuado de velocidad de transferencia a fin de que pueda muestrear los canales solicitados, para esto utiliza la función **DAQ_Rate**. Habilita además el modo de doble buffer con **DAQ_DB_Config**. Utilizando **SCAN_Setup** configura el número de canales a ser muestreados así como la ganancia utilizada. Por último con la función **SCAN_Start** inicia la adquisición múltiple.

Parámetros *Entrada.*-ninguno .
Salida.- ninguno.

StopHardware *TConvAT_MIO_64F5*

Detiene la adquisición de muestras a nivel físico o de hardware.

Descripción Cancela la operación actual de conversión utilizando la función **DAQ_Clear**.Deshabilita el modo de doble buffer con **DAQ_DB_Config.** Se asegura de que las funciones regresaran eventualmente el control a la aplicación con **Timeout_Config** y por último remueve todos los mensajes asociados con el dispositivo mediante **Config_DAQ_Event_Message.**

Parámetros *Entrada.*- ninguno.
Salida.- ninguno.

Create *TConvAT_MIO_64F5*

Crea una instancia de la clase TconvAT_MIO_64F5.

Descripción Inicializa las siguientes propiedades:

- ***MaxFrec*** al valor de 3000
- ***MaxChanel*** al valor de 16

Parámetros *Entrada* - ninguno.
Salida - ninguno.

Destroy *TConvAT_MIO_64F5*

Este método se hereda de la clase padre.

GetGainStrings *TConvAT_MIO_64F5*

Obtiene las cadenas correspondientes a las ganancias de la tarjeta.

Descripción Este método utiliza un arreglo dinámico de registros, que es alimentado con las siguientes ganancias: de +/-50mV, +/-100mV, +/-0.25V, +/-0.5V, +/-1V, +/-2.5V, +/-5V y +/-10V

Parámetros *Entrada.*- ninguno.
Salida.- ninguno.

Volts *TConvAT_MIO_64F5*

Convierte una muestra en un valor en voltaje con base en la ganancia elegida.

Descripción Recibe la muestra en un valor entero y lo multiplica por su factor de ganancia para obtener el sus valor de voltaje en formato de punto flotante. El factor de ganancia se refiere a la relación entre Volts/bit.

$$Volts = ValorEntero * FactorDeGanancia$$

Parámetros *Entrada.*- valor entero – muestra recibida.
Salida.- valor de punto flotante – valor de voltaje entregado.

Scaled *TConvAT_MIO_64F5*

Convierte una muestra en un valor con base en un escalamiento.

Descripción Al recibir el valor entero lo multiplica por un factor de escala y le suma un offset específico, obteniendo como resultado el valor escalado en formato de punto flotante.

*ValorEscalado = ValorEntero * FactorDeEscalaCanal + FactorDeOffsetCanal*

Parámetros *Entrada* – valor entero –*muestra a ser escalada.*
valor entero – *número del canal al que pertenece la muestra.*
Salida - valor de punto flotante – *muestra escalada.*

TConvPCL_812PG

Unidad: UConvPCL_812PG

TObject • TConvertidor • TConvPCL_812PG

MÉTODOS

Las siguiente tabla lista los métodos de TConvPCL_812PG:

Método	Visibilidad
StartHardware	protected
StopHardware	protected
encodeGain	protected
Create	public
Destroy	public
GetGainStrings	public
Volts	public
Scaled	public

Tabla C.4 Métodos de TConvPCL_812PG

StartHardware — *TConvPCL_812PG*

Comienza la adquisición de muestras a nivel físico o de hardware.

Descripción Abre el dispositivo con **DRV_DeviceOpen**

Reserva memoria para la conversión, el cálculo del espacio en memoria se hace tomando en cuenta la frecuencia de muestreo y el número de canales, cada muestra es almacenada en una localidad del tamaño de una palabra.

Configura el evento. Para configurar el evento se le indicó que éste podía ser del tipo interrupción, cambio de buffer, terminación de evento y sobre-escritura, utilizando **DRV_EnableEvent**

Creación de la lista de ganancia Ya que para los propósitos de este programa de adquisición, la ganancia es la misma para todos los canales, ésta es asignada a todos los canales activos.

Inicia el muestreo Para esto se configura la estructura **PT_FAIIntScanStart**. Inicio de la adquisición **DRV_FAIIntScanStart**

Habilita el thread (clase **TGetEvent**)

Parámetros *Entrada.*-ninguno .
Salida.- ninguno.

StopHardware *TConvPCL_812PG*

Detiene la adquisición de muestras a nivel físico o de hardware.

Descripción Detiene el thread (**TGetEvent**) La instancia de clase que transfiere los datos debe ser detenida, si ésta se encuentra aún activa, a continuación debe procederse a:

- Detener la adquisición con **DRV_FAIStop.**
- Recolectar los datos de la última adquisición. con la función **DRV_FAITransfer.**
- Guardar los datos en el buffer circular.
- Cerrar el dispositivo con **DRV_DeviceClose** y liberar el buffer de datos.

Parámetros *Entrada.*- ninguno.
Salida.- ninguno.

Create *TConvPCL_812PG*

Crea una instancia de la clase TConvPCL812_PG.

Descripción Inicializa las siguientes propiedades:

- ***MaxFrec*** al valor de 3000
- ***MaxChanel*** al valor de 64

1. Verifica que existan dispositivos instalados. **DRV_DeviceGetNumOfList**
2. Abre el dispositivo para obtener su información **DRV_DeviceOpen**
3. Obtiene las características del dispositivo **DRV_DeviceGetFeatures**
4. Después de la llamada a la función de creación del padre TConvertidor, se asignó el canal máximo y la frecuencia máxima de muestreo
5. Cierre del dispositivo **DRV_DeviceClose**

Parámetros *Entrada* - ninguno.
Salida - ninguno.

Destroy *TConvPCL_812PG*

Este método se hereda de la clase padre.

GetGainStrings *TConvPCL_812PG*

Obtiene las cadenas correspondientes a las ganancias de la tarjeta.

Descripción Este método utiliza un arreglo dinámico, que es alimentado con las siguientes ganancias: de +/- 5V, +/- 2.5V, +/- 1.25V, +/- 0.625V, +/- 0.3125V

Parámetros *Entrada.*- ninguno.
Salida.- ninguno.

Volts *TConvPCL_812PG*

Convierte una muestra en un valor en voltaje con base en la ganancia elegida.

Descripción Recibe la muestra en un valor entero le resta 2048 y al resultado lo multiplica por su factor de ganancia para obtener el sus valor de voltaje en formato de punto flotante. El factor de ganancia se refiere a la relación entre Volts/bit.

$$Volts = (ValorEntero\text{-}2048) * FactorDeGanancia$$

Parámetros *Entrada.*- valor entero – muestra recibida.
Salida.- valor de punto flotante – valor de voltaje entregado.

Scaled *TConvPCL_812PG*

Convierte una muestra en un valor con base en un escalamiento.

Descripción Al recibir el valor entero le resta 2048 y al resultado lo multiplica por un factor de escala y le suma un offset específico.

$$ValorEscalado = (ValorEntero\text{-}2048) * FactorDeEscalaCanal + FactorDeOffsetCanal$$

Parámetros *Entrada* - valor entero –*muestra a ser escalada.*
valor entero – *número del canal al que pertenece la muestra.*
Salida - valor de punto flotante – *muestra escalada.*

TCircular

Unidad: UCircular

TObject • TCircular

PROPIEDADES

Las siguiente tabla lista las propiedades de TCircular:

Propiedad	Visibilidad
Distancia	public, read-only
Tail	public, read-only
Head	public, read-only
Vuelta	public, read-only
Overflow	public, read-only

Tabla C.5 Propiedades de TCircular

Distancia *TCircular*

Distancia es el valor que indica el número de enteros almacenados en la lista circular, o en otras palabras la distancia de la cabeza hasta la cola.

property Distancia : Integer

Descripción Esta propiedad puede usarse para leer el valor de la distancia.

Tail *TCircular*

Tail es el valor del extremo posterior de la lista circular.

property Tail : integer

Descripción Esta propiedad es utilizada para leer el número del último elemento de la lista circular.

Head *TCircular*

Head es el valor del extremo anterior de la lista circular.

property Head : integer

Descripción Esta propiedad es utilizada para leer el número del primer elemento de la lista circular.

Vuelta *TCircular*

Vuelta es una bandera que indica si se ha completado una vuelta en la lista circular.

property Vuelta : boolean

Descripción Esta propiedad es utilizada para leer la bandera *Vuelta*, esta bandera tiene el valor de verdadero cuando se ha completado una vuelta en la lista circular y regresa al valor de falso una vez que la lista ha sido vaciada.

Overflow *TCircular*

Overflow es una bandera que indica si ha ocurrido una sobre-escritura de datos.

property Overflow : boolean

Descripción Esta propiedad es utilizada para indicar si se han perdido datos, por una sobre-escritura.

MÉTODOS

Las siguiente tabla lista los métodos de TCircular:

Método	Visibilidad
Create	Protected
Destroy	Protected
Reset	Protected
Add	Protected
Get	Protected
nGet	Protected

Tabla C.6 Métodos de TCircular

Los métodos que se plantean para el objeto lista circular son los siguientes:

Create *TCircular*

Crea una instancia de la clase TCircular.

Descripción Con el valor de entrada más 10 asigna el tamaño a un arreglo de valores enteros. Especificando el número máximo de puntos para la lista. A continuación efectúa una llamada al método **reset**.

Parámetros *Entrada* - valor entero – *número de puntos que contendrá la lista.*
Salida - ninguno.

Destroy *TCircular*

Destruye una instancia de TCircular.

Descripción Libera el arreglo de puntos utilizado para la lista circular.

Parámetros *Entrada* - ninguno.
Salida - ninguno.

Reset *TCircular*

Inicializa la lista circular.

Descripción Inicializa las siguientes propiedades:

- ***Vuelta*** al valor de *falso*.
- ***OverFlow*** al valor de *falso*.
- ***Tail*** al valor de 0.
- ***Head*** al valor de 0.
- ***Distancia*** al valor de 0.

Parámetros *Entrada* - ninguno.
Salida - ninguno.

Add *TCircular*

Agrega un valor a la lista circular.

Descripción Este método primero verifica que el inicio de la lista no coincida con el fin y a demás que no se haya completado una vuelta. Si es así el valor recibido es insertado al inicio de la lista la distancia es incrementada y si el inicio de la lista no coincide con el máximo de puntos el apuntador al inicio es incrementado también, de otra forma el inicio es puesto a cero y el valor de vuelta es puesto a verdadero.

Parámetros *Entrada* - valor entero – *elemento de la lista a ser agregado.*

Get *TCircular*

Obtiene un valor de la lista circular.

Descripción Este método primero verifica que el inicio de la lista no coincida con el fin y a demás que no se haya completado una vuelta. Si el inicio coincide con el fin retorna un valor de falso, de otra forma retrae el valor de la lista, decrementa la distancia. Si el fin es menor que el número máximo de elementos incrementa el apuntador de fin, de otra manera inicializa el fin al valor de cero y la bandera de vuelta es puesta al valor de *Falso*. Si sucedió que el inicio y el fin coincidían además de que una vuelta fue completada el valor status de la obtención es puesto a *falso*, y la bandera de overflow es puesta a *verdadero.*

Parámetros *Entrada* - valor entero – *elemento de la lista a ser obtenido.*
Salida - valor booleano – *status de la obtención de la muestra.*

nGet *TCircular*

Obtiene n número de valores de la lista circular.

Descripción Este método verifica que la bandera de overflow no este activada y que el valor entero recibido sea mayor que cero, si las condiciones no se cumplen retorna el valor de *falso.* Una vez verificadas las condiciones se procede a checar que el número de muestras que se desean obtener no exceda el número de muestras almacenadas en la lista, , si la condición no se cumple retorna el valor de *falso.* Por último se hace una llamada iterativa a **get** para obtener todas las muestras.

Parámetros *Entrada* - valor entero – *número de muestras a ser obtenidas.*
arreglo de enteros – *contendrá las muestras obtenidas.*
Salida - valor booleano – *status de la obtención de las muestras.*

TGetEvent

Unidad: UConvPCL_812PG

TObject • TThread • TGetEvent

TgetEvent es una clase del tipo thread que es creada para que efectúe la transferencia del buffer del convertidor de la tarjeta de adquisición PCL_812PG al buffer circular perteneciente al convertidor genérico.

Las siguiente tabla lista los métodos de TGetEvent:

Método	Visibilidad
Execute	protected
AdOverrunEvent	protected
AdHalfReady	protected
Create	public

Tabla C.7 Métodos de TGetEvent

Execute — *TGetEvent*

Ejecuta el thread.

Descripción Este método primero realiza la preparación de la estructura para el chequeo del evento.

Después efectúa el siguiente ciclo a lo largo de la adquisición de datos:

1. Obtiene el evento del dispositivo **DRV_CheckEvent** Si no obtiene el evento cierra el thread
2. Verifica el contenido del evento Overrun.
3. Verifica si se ha llenado la mitad del buffer. Si el ciclo concluye envía un mensaje de error y termina el thread.

Parámetros *Entrada* - valor de punto flotante – *número de dispositivo.*
Apuntador a estructura – *handler del dispositivo.*
Salida - ninguno.

AdOverrunEvent — *TGetEvent*

Se ejecuta cuando ocurrió un sobreflujo de información.

Descripción Limpia el sobreflujo con la función **DRV_ClearOverrun**, obteniendo el código de error. Con el código de error y la función **DRV_GetErrorMessage** obtiene el mensaje de error que es enviado al usuario.

Parámetros *Entrada* - ninguno.
Salida - ninguno.

AdHalfReady *TGetEvent*

Se ejecuta cuando se llena la mitad del buffer interno.

Descripción Este método es el encargado de transferir las muestras del buffer interno al buffer circular. El proceso se lleva en dos etapas primero transfiere la primera mitad en una llamada, en la siguiente invocación de este método se transfiere la segunda mitad. El proceso es el siguiente:

1. Modifica la bandera *LoadingSamples* del convertidor poniéndole el valor de *verdadero*. Esto es con el fin de indicar que esta cargando las muestras al buffer circular.

2. Verifica la variable local *CanEnter*, esto es con el propósito de que no interfiera con otra ejecución del mismo thread.

3. Verifica que mitad del buffer interno debe ser transferida en esta llamada.

4. Prepara la estructura de datos para la adquisición, dependiendo de si se encontraba en la primera o segunda mitad del buffer, de la siguiente forma:

```
overrun := @wOverrun
bandera que indica si existe sobre-escritura del buffer
count := giConvCount div 2
numéro de muestras a transferir
start := startCount
Inicia desde el principio del buffero a la mitad
DataType := 0
El tipo de datos es enter, la escala corresponde a 0-4096
ActiveBuf:=0
Un solo buffer
space := SizeOf(word); GetMem(DataBuffer, space * count )
Reserva memoria par los datos, cada dato ocupa una palabra.
```

5. Inicia la transferencia de las últimas muestrascon la función **DRV_FAITransfer.** Si existe un error en este punto el dispositivo se cierra y el espacio en memoria se libera.

6. Guarda los datos en el buffer circular.

7. Modifica la variable local *CanEnter* poniéndole el valor de *verdadero.*

8. Modifica la bandera *LoadingSamples* del convertidor poniéndole el valor de *falso.* Para indicar que dejo de cargar las muestras.

Parámetros *Entrada* - ninguno.
Salida - ninguno.

Create *TGetEvent*

Crea una instancia de la clase TGetEvent

Descripción Se le indica que debe ejecutarse en el momento de ser creado, con el parámetro *falso*, además de que en el término de su ejecución deberá liberar su espacio. Asignando el parámetro recibido como el manejador de dispositivo a una variable interna de la clase.

Parámetros *Entrada* - valor entero – *número de puntos que contendrá la lista.*
Salida - ninguno.

TSmallGraph

Unidad: USmallGraph

TObject • TPersistente • TComponent • TControl • TGraphicControl • TGraph • TSmallGraph

El componente TChart que incluye la versión 5 de Delphi, experimento problemas en el momento de la graficación. La memoria era consumida en virtud del gran número de muestras recibidas.

El componente fue heredado en base a una clase TGraph de propósito general desarrollado previamente.

PROPIEDADES

Las siguiente tabla lista las propiedades de TSmallGraph:

Propiedad	Visibilidad
Serie1Color	published
Serie2Color	published
Serie3Color	published
Serie4Color	published
GridColor	published
AxisColor	published

Tabla C.8 Propiedades de TSmallGraph

Serie1Color *TSmallGraph*

Serie1Color es el color correspondiente a la serie o canal número 1.

property Serie1Color:TColor

Descripción Esta propiedad es utilizada para leer o escribir el color de la serie 1.

Serie2Color *TSmallGraph*

Serie2Color es el color correspondiente a la serie o canal número 2.

property Serie2Color:TColor

Descripción Esta propiedad es utilizada para leer o escribir el color de la serie 2.

Serie3Color *TSmallGraph*

Serie3Color es el color correspondiente a la serie o canal número 3.

property Serie3Color:TColor

Descripción Esta propiedad es utilizada para leer o escribir el color de la serie 3.

Serie4Color *TSmallGraph*

Serie4Color es el color correspondiente a la serie o canal número 4.

property Serie4Color:TColor

Descripción Esta propiedad es utilizada para leer o escribir el color de la serie 4.

GridColor *TSmallGraph*

GridColor es el color correspondiente a la malla de la gráfica.

property GridColor:TColor

Descripción Esta propiedad es utilizada para leer o escribir el color de la malla.

AxisColor *TSmallGraph*

AxisColor es el color correspondiente a los ejes de la gráfica.

property AxisColor:TColor

Descripción Esta propiedad es utilizada para leer o escribir el color de los ejes en la gráfica.

MÉTODOS

Las siguiente tabla lista los métodos de TSmallGraph:

Método	Visibilidad
SetSerie1Color	private
SetSerie2Color	private
SetSerie3Color	private
SetSerie4Color	private
SetGridColor	private

Método	Visibilidad
SetAxisColor	private
Create	public
Destroy	public
Prepare	public
PlanoOnPaint	public
PlotPoint	public
PlotPoint1	public
PlotPoint2	public
PlotPoint3	public
PlotPoint4	public
Scroll	public
Clean	public

Tabla C.9 Métodos de TSmallGraph

SetSerie1Color *TSmallGraph*

Asigna el color a la Serie 1

Descripción Este método asigna un color al trazo del canal 1, revisando antes que ésta no tenga el mismo color asignado previamente.

Parámetros *Entrada* - TColor – *Color para asignar a la serie 1.*
Salida - ninguno.

SetSerie2Color *TSmallGraph*

Asigna el color a la Serie 2

Descripción Este método asigna un color a la serie 2 o canal 2, revisando antes que ésta no tenga el mismo color asignado previamente.

Parámetros *Entrada* - TColor –*Color para asignar a la serie 2.*
Salida - ninguno.

SetSerie3Color *TSmallGraph*

Asigna el color a la Serie 3

Descripción Este método asigna un color al trazo del canal 3, revisando antes que ésta no tenga el mismo color asignado previamente.

Parámetros *Entrada* - TColor –*Color para asignar para la serie 3.*
Salida - ninguno.

SetSerie4Color *TSmallGraph*

Asigna el color a la Serie 4

Descripción Este método asigna un color a la serie 4 o canal 4, revisando antes que ésta no tenga el mismo color asignado previamente.

Parámetros *Entrada* - TColor –*Color para asignar para la serie 4.*
Salida - ninguno.

SetGridColor *TSmallGraph*

Asigna el color a la malla.

Descripción Este procedimiento asigna un color a la malla, revisando previamente que esta no tenga el mismo color asignado con anterioridad.

Parámetros *Entrada* - TColor –*Color para asignar a la malla.*
Salida - ninguno.

SetAxisColor *TSmallGraph*

Asigna el color a los ejes.

Descripción Este método asigna un color a los ejes, revisando antes que ésta no tenga el mismo color asignado previamente.

Parámetros *Entrada* - TColor –*Color para asignar a los ejes.*
Salida - ninguno.

Create *TSmallGraph*

Crea una instancia de la clase TSmallGraph

Descripción En este constructor se inicializo el color de cada una de las series de la graficación, de los ejes y de la malla. También se liga una función personalizada con el evento de re-dibujo del componente gráfico.

Parámetros *Entrada* - TComponent –*Propietario del objeto.*
Salida - ninguno.

Destroy *TSmallGraph*

Destruye una instancia de TSmallGraph.

Descripción Libera el espacio en memoria utilizado por el componente.

Parámetros *Entrada* - ninguno.
Salida - ninguno.

Prepare *TSmallGraph*

Calcula las dimensiones de los cuadros y el espacio que ocupará el área de graficación.

Descripción Inicializa el contador de punto actual para cada una de las series. Y llena el vector de valores en el eje de las X para el área especificada. Asigna el corrimiento a 0, así como también define los rectángulos en que se divide el área de graficación.

Parámetros *Entrada* – valor entero – *Total de cuadros*
valor entero – *Número de cuadros*
valor de punto flotante – *Frecuencia a la que esta trabajando el convertidor*
Salida - ninguno.

PlanoOnPaint *TSmallGraph*

Asigna los valores para el espacio de la gráfica.

Descripción En este procedimiento se asignan los valores de incremento para los ejes X y Y. Manipulando las coordenadas máximas y mínimas del espacio de trabajo. Se dibujan los ejes, la malla y las etiquetas para indicar las escalas en los ejes.

Parámetros *Entrada* – TCanvas – *lienzo sobre el que se va a dibujar.*
Salida - ninguno.

PlotPoint *TSmallGraph*

Dibuja una línea para cualquier serie.

Descripción Este método dibuja una línea entre el punto anterior y el actual, cualquier serie. Recibe el punto a ser dibujado y también el número de la serie.

Parámetros *Entrada* – valor real – *punto a ser dibujado.*
Valor entero – *número de la serie.*
Salida - ninguno.

PlotPoint1 *TSmallGraph*

Dibuja una línea para la serie 1.

Descripción Este método dibuja una línea entre el punto anterior y el actual, para la serie1. La propiedad ***SerieColor1*** es utilizada para cambiar el color actual de pluma de windows, y obtener el color de la serie..

Parámetros *Entrada* – valor real – *punto a ser dibujado.*
Salida - ninguno.

PlotPoint2 *TSmallGraph*

Dibuja una línea para la serie 2.

Descripción Este método dibuja una línea entre el punto anterior y el actual, para la serie2. La propiedad ***SerieColor2*** es utilizada para cambiar el color actual de pluma de windows, y obtener el color de la serie..

Parámetros *Entrada* – valor real – *punto a ser dibujado.*
Salida - ninguno.

PlotPoint3 *TSmallGraph*

Dibuja una línea para la serie 3.

Descripción Este método dibuja una línea entre el punto anterior y el actual, para la serie3. La propiedad ***SerieColor3*** es utilizada para cambiar el color actual de pluma de windows, y obtener el color de la serie..

Parámetros *Entrada* – valor real – *punto a ser dibujado.*
Salida - ninguno.

PlotPoint4 *TSmallGraph*

Dibuja una línea para la serie 4.

Descripción Este método dibuja una línea entre el punto anterior y el actual, para la serie4. La propiedad ***SerieColor4*** es utilizada para cambiar el color actual de pluma de windows, y obtener el color de la serie..

Parámetros *Entrada* – valor real – *punto a ser dibujado.*
Salida - ninguno.

Scroll *TSmallGraph*

Efectúa un corrimiento en la gráfica..

Descripción En este método se efectúa el corrimiento o scroll de la gráfica. Los puntos a recorrerse se especificaron en la función prepare. Los rectángulos asignados previamente son manipulados para copiar la parte que se conserva de la gráfica y preparar la parte del corrimiento para agregar nuevos puntos. También se toma en cuenta el repintado de las etiquetas y los ejes.

Parámetros *Entrada* - ninguno.
Salida - ninguno.

Clean *TSmallGraph*

Limpia la gráfica.

Descripción Este método inicializa los puntos a ser dibujados en primera instancia a cero y manda llamar un repintado del componente. Es necesario para efectos de inicialización para la adquisición de las siguientes muestras.

Parámetros *Entrada* - ninguno.
Salida - ninguno.