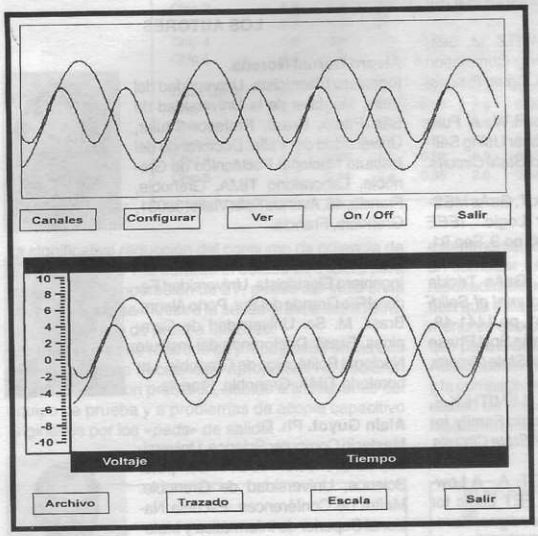
Software para monitoreo y registro de señales eléctricas mediante PC



❒ **Edinson Franco Mejía,** M.Sc.
Ingeniero Electricista
Profesor Asociado
Universidad del Valle

❒ **Carlos Felipe Rengifo**
Ingeniero Electricista
Asistente de Docencia
Universidad del Valle

Los autores pertenecen al Grupo de Investigación en Aplicaciones Industriales de la Teoría de Control (G.I.A.I.C).

* Este texto se recibió, para ser publicado, en octubre de 1997.

RESUMEN

En este artículo se presentan los resultados de la implementación de un software para monitoreo y registro de señales eléctricas por computador; se explican los posibles esquemas de monitoreo, la programación del controlador de interrupciones del PC, así como una descripción de la interfaz de usuario utilizada.

ABSTRACT

This paper present the results of a software implementation. This software is used to monitor and to register electrical signals by a personal computer. The possible monitor schemes, the controller PC interruption programming and an user interface are explained.

1. INTRODUCCIÓN

Dentro del control automático es indispensable el monitoreo y almacenamiento de señales asociadas con el proceso a controlar, por inconvenientes de costos, número limitado de canales, imposibilidad de almacenamiento masivo de información y dificultad para obtención de información impresa, por esto se implementó un software que permite a través de una tarjeta de adquisición de datos realizar el monitoreo, registro e impresión de señales eléctricas en un computador.
El software se implementó en C++ bajo D.O.S, utilizando el enfoque orientado a objetos, el cual está conformado por archivos independientes, clasificados según su función y compilados en un proyecto.

2. ESQUEMAS PARA MONITOREO DE SEÑALES

Los esquemas de monitoreo a utilizar en un software de adquisición de datos son:

Poolling: es la técnica más simple de implementar, consiste, en enviar la señal de inicio de conversión a la tarjeta, esperar a que el dato sea convertido, pasarlo a la rutina de graficación y esperar hasta que se complete un periodo de muestreo.

Acceso Directo a Memoria: los datos son enviados directamente de la tarjeta de adquisición a la memoria del PC; durante la transferencia de información el procesador está inhabilitado, por consiguiente, los datos deben ser graficados después de que han sido transferidos en su totalidad; cuando se está graficando no puede realizarse adquisición. Aunque esta técnica es la más rápida, en cuanto rata de muestreo, tiene el inconveniente que los datos deben ser leídos en bloques y no es posible realizar un monitoreo continuo.

Interrupciones: Cuando un dato está listo para ser leído, la tarjeta de adquisición genera una señal de interrupción que detiene la ejecución del programa encargado de graficar, la rutina de servicio a la interrupción lee el dato de la tarjeta y lo almacena en un vector, para ser graficado posteriormente. Cuando la rutina termina, el programa continúa su ejecución.

Las señales de inicio de conversión son dadas por temporizadores que se programan al inicio de la adquisición, permitiendo un control más preciso del tiempo de muestreo. Con interrupciones se pueden adquirir y graficar datos de una manera más rápida que con Poolling, además tiene la ventaja del monitoreo continuo con respecto a la técnica del DMA. **El software desarrollado utilizó el esquema de interrupciones.**

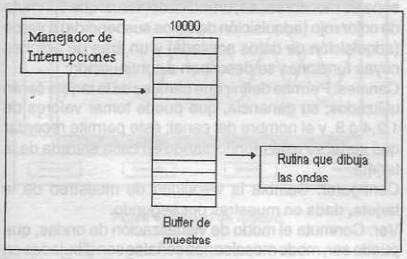


Figura 1. Esquema de monitoreo con interrupciones.

3. PROGRAMACIÓN DEL CONTROLADOR DE INTERRUPCIONES DEL PC

El controlador de interrupciones es un chip programable que se encarga de priorizar e informar al procesador cuál de las fuentes de interrupción solicitó atención. Un controlador de interrupciones (PIC: P-gramable C-ontrole I-nterrupt) tiene 8 líneas de interrupción disponibles, el procesador sólo tiene una línea de interrupción (dos si se tiene en cuenta el pin de solicitud de interrupción no enmascarable NMI). En un esquema con un solo PIC la conexión es la siguiente:

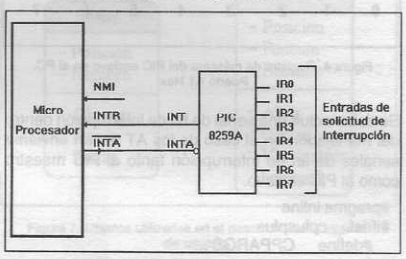


Figura 2. Conexión del PIC con el microprocesador

Cuando el PIC detecta una señal de interrupción, en cualquiera de sus pines (IR0,...IR7) envía una señal al microprocesador por el pin INTR (requerimiento de interrupción), éste le responde con la señal *INTA* (reconocimiento de interrupción) y el PIC le envía al procesador, por el bus de datos (líneas D0,...D7), el número de la interrupción.

La programación de un PIC, en general, incluye dos partes principales: la programación de los registros de inicialización y la programación de los registros de operación. En el ejemplo siguiente se programan los registros de operación del chip de la mother board, la parte de programación de los registros de inicialización la efectúa el B.I.O.S en el proceso de arranque del computador. La programación de los registros de operación, para habilitar una línea de interrupción es bastante simple, cada línea tiene asociado un bit que representa su estado, 0 habilitada y 1 deshabilitada, estos bits (8 en total) están contenidos en el registro de máscara.

Figura 3. Registro de máscara del PIC maestro. Puerto 21 Hex.

El primer PIC controla las interrupciones de la cero a la siete; el segundo, de la ocho a la quince.

Bits del registro de máscara							
IR8	IR9	IR10	IR11	IR12	IR13	IR14	IR15
0	1	2	3	4	5	6	7

Figura 4. Registro de máscara del PIC esclavo en el PC. Puerto A1 Hex.

Se debe incluir una señal de fin de interrupción dentro del manejador, en el caso de los AT deben enviarse señales de fin de interrupción tanto al PIC maestro como al PIC esclavo.

```
#pragma inline
#ifdef __cplusplus
  #define __CPPARGS ...
#else
  #define __CPPARGS
#endif

#include <conio.h>
#include <stdio.h>
#include <dos.h>

// Constantes
const int INTR = 0x72; //Interrupci¢n hardware 10 (IRQ10)
                       //Posicion 72 Hex del vector de interrupción.

//Anterior vector de interrupción.
void interrupt (*oldhandler) (__CPPARGS);

//Rutina manejadora de interrupción.
void interrupt handler (__CPPARGS)
 {
/*..... Escriba aquí su rutina de servicio a la interrupción .....*/
asm {
      mov al,32    //Señal de fin de interrupción.
      out 32,al    //EOI para PIC maestro.
      out 160,al   //EOI para PIC esclavo.
    }
 }
// EOI = End Of Interruption.
// PIC = Programable Interrupt Controler.

void main(void)
{
  unsigned char número;
  oldhandler = getvect(INTR); //Salvar el anterior vector de interrupción
  setvect(INTR,handler);      //Poner el nuevo vector. Dirección de mi manejador.
  //programación del controlador de interrupciones
  número = inportb(0xa1);     //Leo la máscara.
  número &= 0Xfb;             //Cambio del Bit 2 a Cero (Habilitar).0xfb = 1111-1011.
  outportb(0xa1,numero);      //Habilito la línea para IRQ 10.
 printf(«Presione una para desinstalar el manejador de Interrupción\n»);
  while(!kbhit());
 setvect(INTR,oldhandler);
}
```

4. DESCRIPCIÓN DEL SOFTWARE

El software diseñado consta de dos módulos principales: el de monitoreo, encargado de configurar el hardware, leer los datos de la tarjeta de adquisición y almacenarlos en disco; el de registro, encargado de la impresión y visualización de curvas a partir de datos provenientes de archivos; existe, además, un cursor manipulable por el usuario que permite hacer seguimiento de los puntos que componen un gráfico.
El software permite velocidades de muestreo programables de hasta 25.000 muestras/seg entre todos los canales (máximo ocho).

4.1 Diseño arquitectónico - interface de usuario (MMI)

La interface de usuario consta de un menú de entrada que permite al usuario elegir entre el módulo de monitoreo y el de registro, o salir del programa.

4.1.1 Módulo de monitoreo. El módulo de monitoreo está conformado por un área para la visualización de las señales monitoreadas, un indicador visual que conmuta de color rojo (adquisición de datos suspendida) a verde (adquisición de datos activada) y un área de botones cuyas funciones se describen a continuación:
Canales: Permite definir que canales de la tarjeta serán utilizados; su ganancia, que puede tomar valores de 1,2,4 ó 8, y el nombre del canal; éste permite recordar qué señal se está monitoreando en cada entrada de la tarjeta.
Configurar: Cambia la velocidad de muestreo de la tarjeta, dada en muestras por segundo.
Ver: Conmuta el modo de visualización de ondas, que puede ser, modo mosaico: las señales son dibujadas en ventanas independientes, o modo osciloscopio: todas las señales se dibujan en la misma ventana.
On/Off: Inicia la adquisición de datos, y se inhabilitan los otros botones; para suspender el monitoreo debe presionarse <Ctrl - Break>.
Salir: Regresa al menú principal.

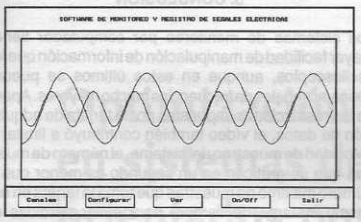


Figura 5. Módulo de monitoreo.

4.1.2. Módulo de registro. De manera similar al módulo de registro, el de monitoreo tiene un área de visualización donde se pueden observar las ondas provenientes de un archivo y un área de botones cuyas funciones son:

Archivo: Permite al usuario seleccionar el archivo de datos con que desea trabajar, su apariencia es similar a la de un menú de selección de archivos tipo windows.

Trazado: Dibuja en pantalla los datos provenientes del archivo seleccionado, habilita el cursor para seguimiento de puntos y permite realizar impresiones en papel.

Escala: Permite cambiar los valores máximo y mínimo de la escala vertical, los valores por defecto 10 y -10 voltios.

Salir: Regresa al menú principal.

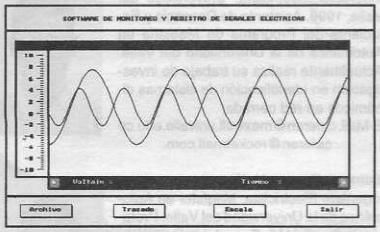


Figura 6. Módulo de registro.

4.2 Diseño interno

Del modelo orientado a los objetos, utilizado para el desarrollo del software, se extractó información (atributos y métodos) de los principales objetos utilizados, así como algunas de las estructuras de clasificación y ensamblaje utilizadas.

La mayoría de los objetos desarrollados tiene funciones relacionadas con la interface de usuario, tales como pantallas y botones, estos objetos tienen la ventaja de que una vez creados pueden ser reutilizados en aplicaciones futuras. Además de estos objetos, se desarrollaron muchos otros que no serán presentados.

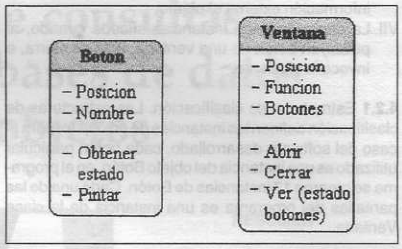


Figura 7. Objetos utilizados en el desarrollo de la interfaz de usuario.

Durante la fase de análisis de un proyecto de programación, cada objeto debe tener una tabla de especificación con un formato como el que se muestra a continuación; como ejemplo se ha tomado la clase Ventana de la figura 5.

I. Nombre del objeto: Ventana
II. Descripciones de atributos
   A. Posición, Botones, Función.
   B. El atributo Posición almacena la coordenadas (fila y columna) superior derecha e inferior izquierda de la ventana. Botones contiene la información de qué botones serán utilizados en la ventana. Función define la rutina encargada de pintar la ventana.
   C. El atributo Posición consta de cuatro números enteros. Botones es un arreglo de punteros a los objetos de tipo Botón que serán utilizados. Función es del tipo puntero a rutina.
III. El objeto no procesa ningún tipo de datos externo.
IV. El objeto no entrega ningún tipo de datos a otros objetos. Simplemente, de acuerdo con el botón pulsado, invoca la rutina correspondiente.
V. Descripciones de operaciones
   A. Abrir, Cerrar, Ver.
   B. La operación Abrir se encarga de que se dibuje la ventana y se activen sus botones. Cerrar libera la memoria asociada con los punteros a los botones y los desactiva. Ver verifica si un botón ha sido pulsado o no.
   C. Dentro de las restricciones y limitaciones de las operaciones del objeto puede mencionarse el hecho de que las ventanas son estáticas y de tamaño constante, esta restricción se debe, principalmente, a la limitación de asignación de memoria del DOS, como se sabe, para desplazar o

cambiar el tamaño de una ventana, debe almacenarse la información de su entorno.

VI. La conexión de mensajes va del objeto Botón al objeto Ventana, como Botón es parte de Ventana, esto no debe entenderse como una entrada de información externa al objeto.

VII. La conexión entre instancias sucede cuando, al pulsar un botón de una ventana, ésta se cierra, e invoca la siguiente.

4.2.1 Estructuras de clasificación. Las estructuras de clasificación definen las instancias de un objeto; para el caso del software desarrollado, cada botón particular utilizado es una instancia del objeto Botón, en el programa se crearon 17 instancias de Botón. Cada una de las pantallas del programa es una instancia de la clase Ventana.

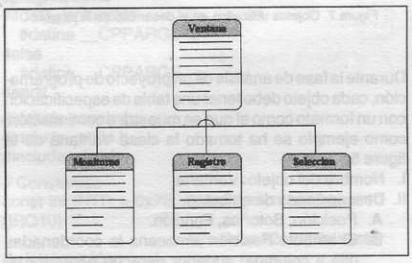


Figura 8. Instancias de la clase ventana.

4.3 Estructuras de ensamblaje

Un objeto puede estar constituido por otros objetos, dicha relación debe especificarse por medio de una estructura de ensamblaje; dentro de los componentes de una ventana se encuentran los botones que, a su vez, son otros objetos.

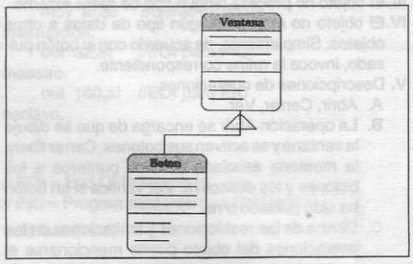


Figura 9. Composición de la clase ventana.

5. CONCLUSIÓN

Los sistemas de monitoreo por computador tienen mayor facilidad de manipulación de información que los osciloscopios, aunque en estos últimos se pueden observar señales de frecuencias mucho mayores. Aparte de las restricciones impuestas por la tarjeta de adquisición de datos, el vídeo también contribuyó a limitar la velocidad de muestreo del sistema, el número de muestras que se grafican en un segundo es menor que el número máximo de muestras que toma la tarjeta en ese mismo instante de tiempo; para evitar pérdidas de información, los datos se almacenan en un buffer hasta ser graficados, cuando este buffer se llena, el monitoreo se suspende; por tal razón, para algunas ratas de adquisición, el monitoreo no es continuo, ni en tiempo real.

6. BIBLIOGRAFÍA

[1.] Roger. S. P, Software Engineering. McGrawHill, Inc. 1993.

[2.] Barry. B. B, The advanced intel microprocessors 80286, 80386 and 80486. Macmillan, Inc. 1992.

[3.] Sharam. H, C++ guía para programadores en C. Prentice Hall, Inc. 1992.

[4.] Rengifo. C.F, Software para Monitoreo y Registro de señales eléctricas por computador. Universidad del Valle. 1996.

[5.] User Guide of AT-MIO 16D National Instruments

AUTORES

Carlos Felipe Rengifo Rodas.

Ingeniero Electricista, Universidad del Valle, 1996. Asistente de Docencia y Estudiante del Programa de Maestría en Automática de la Universidad del Valle. Actualmente realiza su trabajo de investigación en Identificación de sistemas dinámicos en red cerrada.
E-Mail: caferen@maxwell.univalle.edu.co
caferen@rocketmail.com.

Edinson Franco Mejía.

Ingeniero Electricista, Magíster en Automática de la Universidad del Valle. Profesor Asociado de la Escuela de Ingeniería Eléctrica y Electrónica de la Universidad del Valle. Integrante de la Cátedra de Automática. Director de la Maestría en Automática y especialización en Automatización Industrial. Sus áreas de interés son: el control inteligente (adaptativo, redes neuronales, lógica difusa, algoritmos genéticos) para el control de procesos industriales, identificación de sistemas y el desarrollo de hardware y software para aplicaciones en instrumentación y automatización.
E-Mail: efm@maxwell.univalle.edu.co
efm@eiee.univalle.edu.co