ESCUELA POLITECNICA NACIONAL
FACULTAD DE INGENIERIA ELECTRICA

TESIS DE GRADO

"ESTUDIO DEL SISTEMA DE ADQUISICION DE DATOS KEITHLEY 500A Y APLICACIONES"

TESIS PREVIA LA OBTENCION DEL TITULO DE INGENIERO
EN ELECTRONICA Y CONTROL

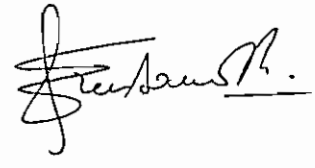
FAUSTO ALBERTO VASCO MONCAYO

MARZO, 1991

Agradecimiento

Mi más sincero y profundo agradecimiento al Ingeniero Patricio Burbano R. por su valiosa y acertada labor en la dirección del presente trabajo de tesis.

Certifico que el presente trabajo de tesis ha sido desarrollado en su totalidad por el Sr. Fausto Alberto Vasco Moncayo.



Ing. Patricio Burbano R.
Director

CONTENIDO

1.1.- INTRODUCCION GENERAL.

En esta introducción de tipo general, se hace una descripción de los objetivos que persigue la realización de este trabajo, el alcance y descripción del mismo.

Uno de los principales objetivos, es el de desarrollar una biblioteca de programas y rutinas, que permitan realizar tareas como adquisición de datos, salida de datos, graficación en tiempo real, procesamiento de la información y aplicaciones de estas tareas, utilizando el sistema de adquisición de datos KEITHLEY 500A, acoplado a un sistema personal de computación IBM PS/2 modelo 60.

La utilización acoplada de estos dos sistemas implica el uso de dos paquetes de software, el QUICK500, diseñado para la adquisición de datos y control mediante la estación KEITHLEY 500A. El segundo es el QuickBASIC 4.5 programa que proporciona el medio ambiente para el QUICK500, y permite la escritura de los programas que ejecutan las diferentes tareas que es posible realizar con el equipo de adquisición de datos y además compilarlos para sus futuras aplicaciones.

Estos programas se desarrollan en dos esquemas: FOREGROUND o trabajo dedicado, y en BACKGROUND o trabajo de fondo; en ambos esquemas se realizan todas las tareas antes enunciadas tanto para sistemas univariables, en los que se utiliza un solo canal de entrada y/o salida (unicanal), como para sistemas multivariables en los que utilizará más de un canal de entrada y/o salida (multicanal).

Los diferentes programas realizados se integran en tres paquetes independientes: el primero que contiene rutinas de adquisición y salida de datos

en los esquemas foreground y background, tanto unicanal como multicanal. Un segundo paquete realiza estas mismas tareas pero con la utilización de interrupciones, gráficos en tiempo real, procesamiento de datos, etc.. Un tercer paquete de aplicaciones que se refieren a control PID, ON-OFF, realimentación de estado, control de motores a pasos, etc..

Los programas desarrollados para la utilización del sistema de adquisición de datos están orientados a facilitar el uso de este equipo en los laboratorios tanto de instrumentación como de control discreto principalmente, y en general a todos los estudiantes y profesores del área, proporcionándoles además mediante la parte escrita de esta tesis un manual de uso del software y el hardware de la estación KEITHLEY 500A. Los usuarios tendrán además la posibilidad de encontrar de manera modular subrutinas que realicen tareas específicas de adquisición, salida, procesamiento, graficación de datos entre otras.

Siguiendo la tendencia actual se utilizan señales normalizadas, esto es, en las entradas y salidas análogas se utiliza valores que varíen dentro del rango de -10 a +10 voltios, mientras que en las salidas de tipo digital se trabaja con señales de amplitud 5 voltios (compatible con niveles lógicos TTL).

Las aplicaciones que se presentan en este trabajo requieren de toda una teoría detrás de ellas, sin embargo en esta tesis serán presentadas casi de manera directa, sin hacer énfasis en su desarrollo y teoría sino más bien en su aplicación misma al tiempo real. Como ya se dijo antes las aplicaciones se llevarán a cabo en los campos de la instrumentación, control lineal y control de motor a pasos mediante las salidas digitales.

El presente trabajo de tesis que tiene como objetivo principal el realizar una biblioteca de programas de aplicación de la estación KEITHLEY 500A y un manual de utilización del mismo se ha dividido en cuatro capítulos.

El capítulo uno ubica el tema que se ha de tratar a lo largo de esta tesis, para lo cual se presenta una introducción de tipo general que trata de los objetivos que se persiguen, el alcance del trabajo y una descripción del mismo; luego, se hace un estudio general del hardware del equipo de adquisición de datos, requerimientos de hardware del equipo de computación acoplado al mismo y del software necesario para el aprovechamiento de este sistema; aspecto en el que se aborda más bien la instalación del paquete, su configuración, requerimientos, modos de trabajo, convenciones, relación con el QuickBASIC, etc., dejando el estudio de los comandos para el segundo capítulo. Finalmente se enfoca un estudio de la adquisición de datos y el control en tiempo real, donde se hace una referencia al control digital directo.

El segundo capítulo centra su estudio en el software de adquisición y salida de datos. Este estudio se realizará en tres partes: en la primera se ocupa de la adquisición de datos análogos, en esquemas foreground y background, en ambos casos para tarea unicanal como multicanal. La segunda parte de este capítulo contempla el estudio de la salida de datos análogos, donde se sigue el mismo esquema de trabajo anteriormente descrito, y una tercera que enfoca el estudio de la salida de datos digitales, siguiendo así mismo, un esquema similar. Cada aspecto que se trate en este capítulo constará de una pequeña base teórica, el desarrollo del programa y una explicación detallada de todos los comandos que se vayan utilizando por primera vez.

El capítulo tercero centra su atención en un software un poco más avanzado, donde ya se realice operación multitarea, enfocando temas como período de muestreo, interrupciones, graficación en tiempo real, procesamiento de información. Cada uno de estos temas se desarrolla siguiendo un esquema donde se indica el objetivo y base teórica, programa y explicación detallada de cada uno de los comandos que se utilicen por primera vez.

En el cuarto capítulo se muestran en primer término los resultados obtenidos con cada una de las rutinas desarrolladas, para lo cual se realizarán pruebas de instrumentación, se presentarán gráficos, etc.. Una segunda parte de este capítulo contempla las aplicaciones del estudio realizado en esta tesis a los sistemas de control. A su vez se presentan los resultados obtenidos en cada una de estas aplicaciones y las conclusiones del trabajo realizado.

1.2.- EL SISTEMA DE ADQUISICION DE DATOS KEITHLEY 500A.

En esta sección se hace un estudio de lo que constituye el sistema de adquisición de datos y control KEITHLEY 500A, describiendo el hardware con que se cuenta en la actualidad, así como del software que permite realizar la interfaz entre el equipo y el usuario, desde sus requerimientos, y configuración, hasta su instalación.

La estación KEITHLEY 500A, está constituida de un módulo macstro, el cual acepta módulos de aplicación, los mismos que se instalan en los 10 slots con que cuenta la tarjeta matriz de este sistema, consta además de una fuente de poder y una interfaz lógica. Es compatible con más de 25 módulos especializados en acondicionamiento de señal y actuación, los cuales pueden utilizarse para adecuar señales medidas (adquiridas) y de control (salidas) para aplicaciones específicas. Entre los módulos de mayor aplicación se cuentan los de entrada y salida análoga, entrada y salida digital, utilización de termocuplas, strain gages, LVDT's y otros.

La estación KEITHLEY 500A está construida de acuerdo a la arquitectura de memoria de un computador IBM PC, PS/2, 80386 y computadores 100% compatibles con estos. El sistema es enlazado al computador, mediante una

interfaz que contiene un bus especializado de extensión, el cual reside en el computador "anfitrión".

A través de esta entrada, todas las funciones del sistema de adquisición de datos y control aparecen en localidades en la memoria del computador, pudiendo entonces realizarse transferencia de datos entre el KEITHLEY 500A y el computador anfitrión incluso a frecuencias altas.

En lo referente al software, se utiliza sea el SOFT500 o el QUICK500, paquetes que permiten al usuario una comunicación adecuada con el hardware del KEITHLEY 500A sea desde el BASIC o desde el QuickBASIC respectivamente. Estos dos paquetes mencionados son capaces de manejar todas las conversiones de unidades de ingeniería, manipulación de la memoria del computador, operación en foreground/background, manejo de arreglos de memoria especiales, gráficos en tiempo real en pantalla y rutinas de análisis de datos que se pueden incluir en los programas de aplicación.

El sistema de adquisición de datos KEITHLEY 500A puede utilizarse para la creación de una estación de trabajo o de laboratorio en la que un computador provée la inteligencia del sistema que incluye adquisición, salida y análisis de datos, sistema que se constituye en un laboratorio independiente, y además es capaz de adaptarse a cualquier tipo de planta o proceso de tipo industrial, para realizar control en tiempo real ON-LINE u OFF-LINE de acuerdo a las necesidades.

1.2.1.- ESTUDIO DEL HARDWARE DEL SISTEMA DE ADQUISICION DE DATOS KEITHLEY 500A.

Para iniciar el estudio del hardware se detalla algunas de las características principales del sistema de adquisición de datos y control en cuestión:

ARQUITECTURA.- Consta de 10 slots de expansión en torno al módulo maestro, acepta módulos de medición análoga y conversión análogo/digital y digital/análogo de 16 bits/50KHz. o de 12 bits/32KHZ (disponible en el equipo).

INTERFAZ.- La estación KEITHLEY 500A utiliza una interfaz común de mapéo de memoria de computador. Esto lo habilita para ser utilizado con computadores IBM PC y PS/2.

REQUERIMIENTOS DEL SISTEMA.- Se necesita un computador 100% compatible con IBM, que tengan al menos un slot de expansión de longitud media, o computadores que contengan al menos un microcanal completamente compatible con un slot de expansión.

REQUERIMIENTOS DE SOFTWARE.- Se necesita un MS-DOS versión 3.0 o más (3.3 para un PS/2). Requiere además del Soft500 v.6.0, Quick500 v.2.0 o mayores. La versión requerida de Soft500 se proporciona con el equipo.

CARCAZA.- Construida de acero rugoso/revestido de aluminio.

PANEL FRONTAL.- Interruptor de encendido (ON-OFF) y lámparas indicadoras del estatus del sistema.

PANEL POSTERIOR.- Entrada de poder, salida de una fuente de poder auxiliar, cable para conexión de la interfaz, fusible.

REQUERIMIENTOS DE ALIMENTACION.- 100/ 120/ 220/ 240 VAC +/- 10%, para lo cual tiene un switch de selección, 50-60 Hz., 90W. máx. Fusible de 1.0A @ 100/120V, 0.5A @ 220/240V.

POWER OUTPUT.- +5V @1A, +/-15V @0.8A disponible desde el panel posterior mediante un conector DB9.

TEMPERATURA DE OPERACION.- De 0 a 50 ^{o}C, con un RH de 80% (sin condensación).

DIMENSIONES Y PESO.- Alto: 11.76 cm, ancho: 32.72 cm, porfundidad: 28.11 cm. Su peso neto es de 5.5 Kg., es decir 12 lbs.

Un diagrama general de sistema de adquisición de datos se presenta en la figura 1.1

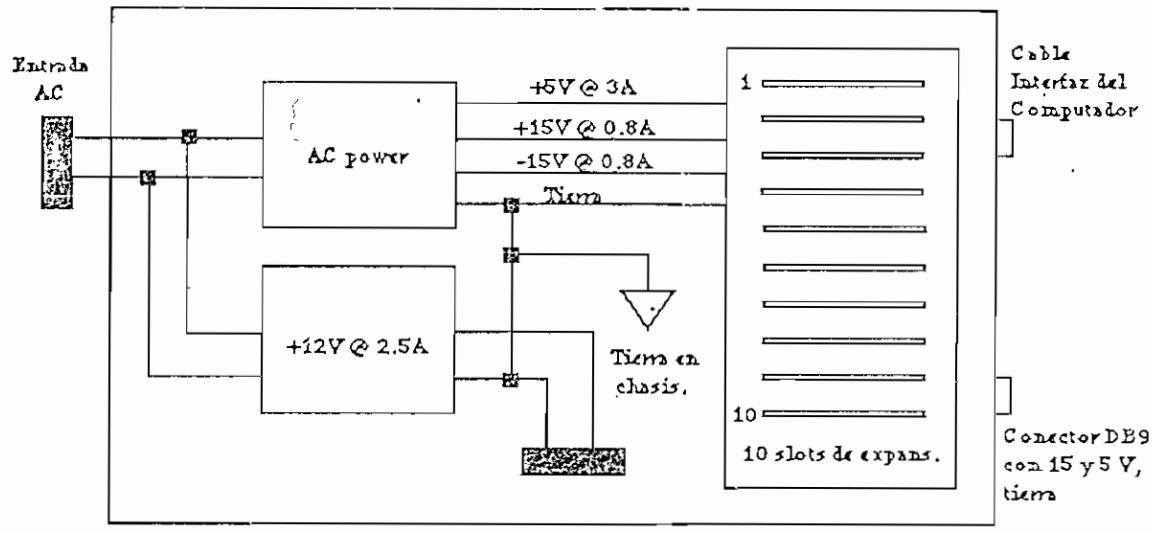


FIGURA 1.1.- Diagrama de Bloques general del KEITHLEY 500A.

La estación KEITHLEY 500A acepta una serie de módulos que se los denomina por la función que realizan y no por su código de identificación. Así se tiene: entradas análogas, que pueden ser entradas de voltaje DC de bajo nivel (disponible en el equipo), entradas de voltaje DC de alto nivel, amplificadores de corriente, entradas de termocuplas tipo B, E, J, K, R, S, T, strain gages, LVDT/RVDT's, sensores de temperatura AD590, entradas para corrientes de lazo normalizadas a 4-20 mA., conversión análoga digital (disponible en el equipo). salidas análogas con conversión digital análoga de 12 bits (disponible en el equipo) o 16 bits, corrientes de lazo de 0-20 mA. y fuentes programables de 0-10V, entradas y salidas digitales de 32 canales E/S, o de 16 canales de entrada o de 16 canales de salida (disponible en el equipo), aceleradores de gráficos, controladores, contadores de pulsos, sea de frecuencia de entrada o contadores de eventos, controladores de poder y controladores de motores a pasos.

Estas son todas las opciones de ampliación de la estación KEITHLEY 500A, sin embargo el equipo existente cuenta con tres módulos para adquisición de datos análogos, salida de datos análogos y salida de datos digitales, de los cuales a continuación se hace una descripción rápida que permite familiarizarse con estos tres módulos.

MODULO MAESTRO DE MEDICION ANALOGO (AMM1).- El módulo maestro de medición análogo (AMM1), es una tarjeta maestra de 12 bits/32 KHz. para realizar medidas análogas (adquisición de datos) con el KEITHLEY 500A. En todo sistema de adquisición de datos es indispensable tener un módulo AMM1 para entradas análogas. El módulo AMM1 combina dos funciones importantes del KEITHLEY 500A en un solo módulo: acondicionamiento y "switcheo" de señales análogas, y la conversión análoga digital A/D.

La sección análoga del AMM1 provée la posibilidad de selección de la señal y de una ganancia programable para señales análogas globales conectadas al sistema. Luego de su acondicionamiento las señales son enrutadas a la sección del conversor análogo digital A/D del módulo para su respectiva conversión.

El módulo AMM1 provée 8 canales de entrada, con ganancia local unitaria, las señales son aplicadas a través de una bornera de tornillo, y la ganancia global es manejada por software, pudiendo obtenerse ganancias de x1, x2 ,x5 y x10.

Para la conversión análoga digital A/D, el AMM1 utiliza un conversor de 12 bits de aproximaciones sucesivas que aseguran velocidad, exactitud en mediciones y conversión. El tiempo máximo que puede llevar una conversión puede ser de 25 uSeg., y una medición por muestreador retenedor puede tomar 5 uSeg. permitiendo muestrear a una frecuencia máxima de 10KHz. Para maximizar la resolución, el AMM1 tiene 5 rangos de conversión A/D (tres

bipolares y dos unipolares) que se pueden seleccionar mediante dip-switches en la tarjeta. El AMM1 cuyo esquema se muestra en la figura 1.2 está diseñado para utilizarse en el slot 1 de la tarjeta matriz del sistema.

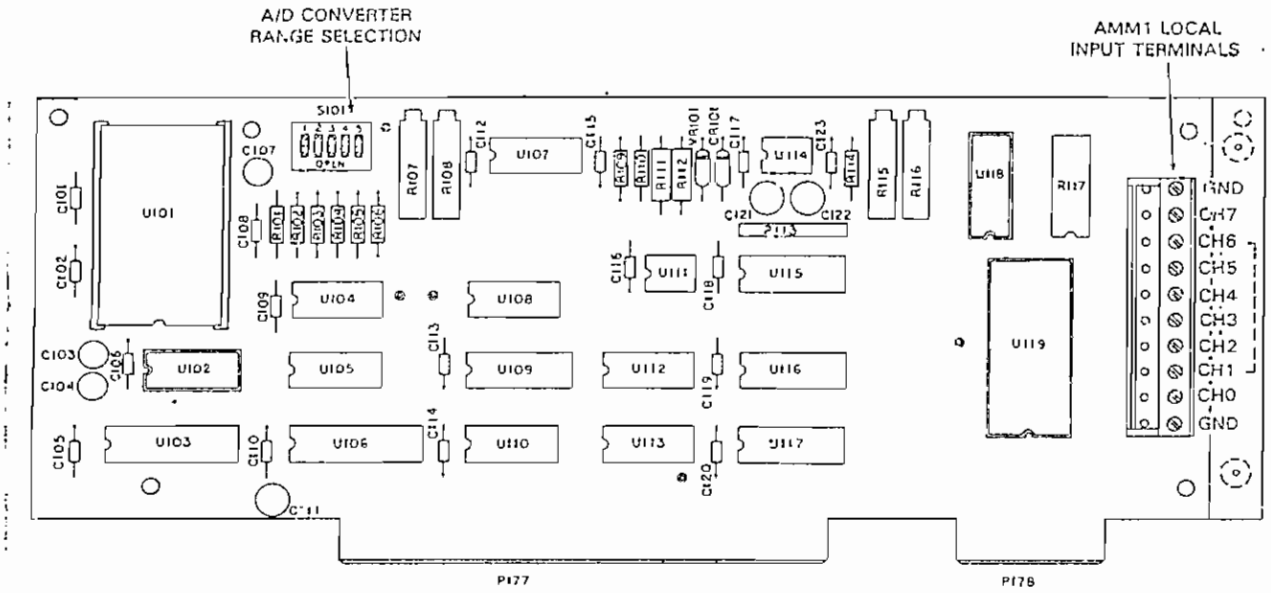


FIGURA 1.2.- Módulo maestro de medición analogo (AMM1).

A continuación se detallan algunas de las principales características del módulo maestro de medición análogo:

CANALES DE ENTRADA.-

Local: 8 canales simples.

Global: 9 entradas en los slots 2-10.

GANANCIA GLOBAL DE AMPLIFICACION PROGRAMABLE.-

Ganancia programable por dip-switches: x1 (seleccionado), x2, x5, x10.

Rangos de entrada: x1: +/-10V , x2: +/-5V, x5: +/-2V, x10: +/- 1V.

Exactitud: +/-(0.01% + 50 uV).

No Linearidad: +/-(0.005% para fondo de escala).

Coeficiente de temperatura: +/-(0.001% + 20 uV).

Ruido en voltaje de entrada: 30 uVp-p, de 0.1 Hz. a 10 KHz.

Exactitud de tiempo (al 0.01%): 6 uSeg.

CONVERSOR ANALOGO DIGITAL.-

Resolución: 12 bits, 1 parte en 4096.

Rangos de entrada: +/-2.5V, +/-5V, +/-10V(seleccionado), 0 a 5V, 0 a 10V.

No Linearidad: +/- 0.025% (+/- bit menos significat (bms)).

Exactitud (incluida la no linearidad): +/-(0.03% + 1 bms).

No linearidad en el coef. de temperatura: +/-0.003%/C.

Exactitud del coef. de tempetarura: +/-(0.0035% +/- 0.11 bms)/C.

Tiempo de adquisición por muestreador-retenedor: 5 uSeg.

Tiempo de conversión: 25 uSeg. máx.

Estas son las principales características de tipo técnico del módulo maestro de medición análogo AMM1, del cual se muestra un diagrama de bloques general en la figura 1.3.

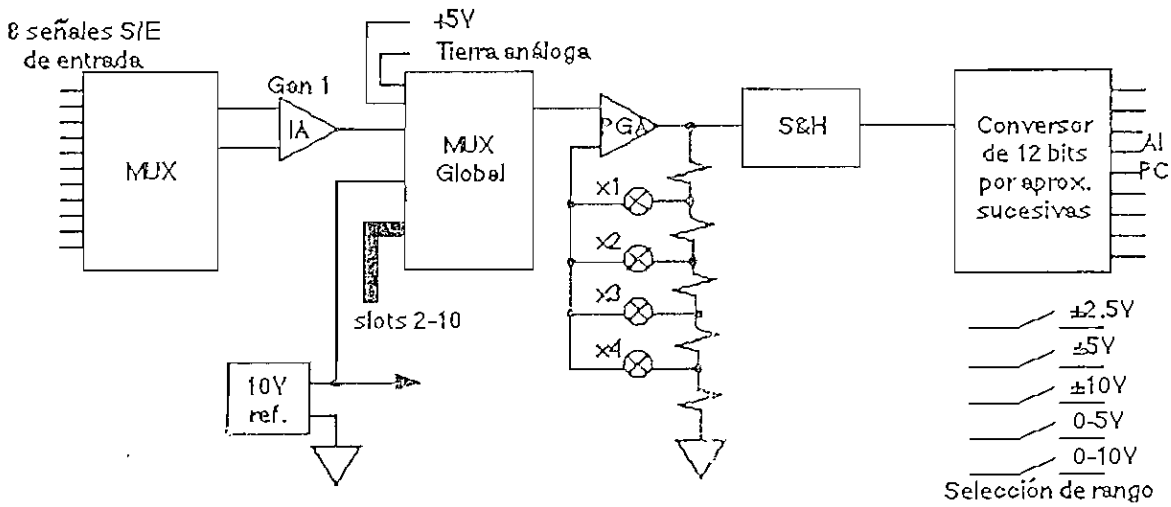


FIGURA 1.3.- Diagrama de bloques general del módulo AMM1.

MODULO DE SALIDA ANALOGO (AOM1).- El módulo de salida análogo, ofrece una resolución de 12 bits con una no linearidad máxima de +/-0.012%. El AOM1 es disponible en dos versiones: AOM1/2, que provée dos canales de salida análoga, y AOM1/5, que provée 5 canales de salida análoga. En ambos casos cada canal tiene un conversor digital a análogo D/A independiente con salida seleccionable mediante dip-swhitches en 0 a 10V y 0 a 5V unipolar, +/-10V, +/-5V, y +/-2.5V bipolar. La conexión de la señal es hecha a través de una bornera de tornillo.

Este módulo tiene la característica de soportar los dos niveles de congelamiento de datos en el conversor D/A, y permite sincronizar la actualización de todos los canales análogos de salida.

El AOM1 puede ubicarse en cualquiera de los 10 slots que contiene la tarjeta matriz del sistema (se encuentra en el slot Nº3). Un esquema de la tarjeta del AOM1/5, se presenta en la figura 1.4.

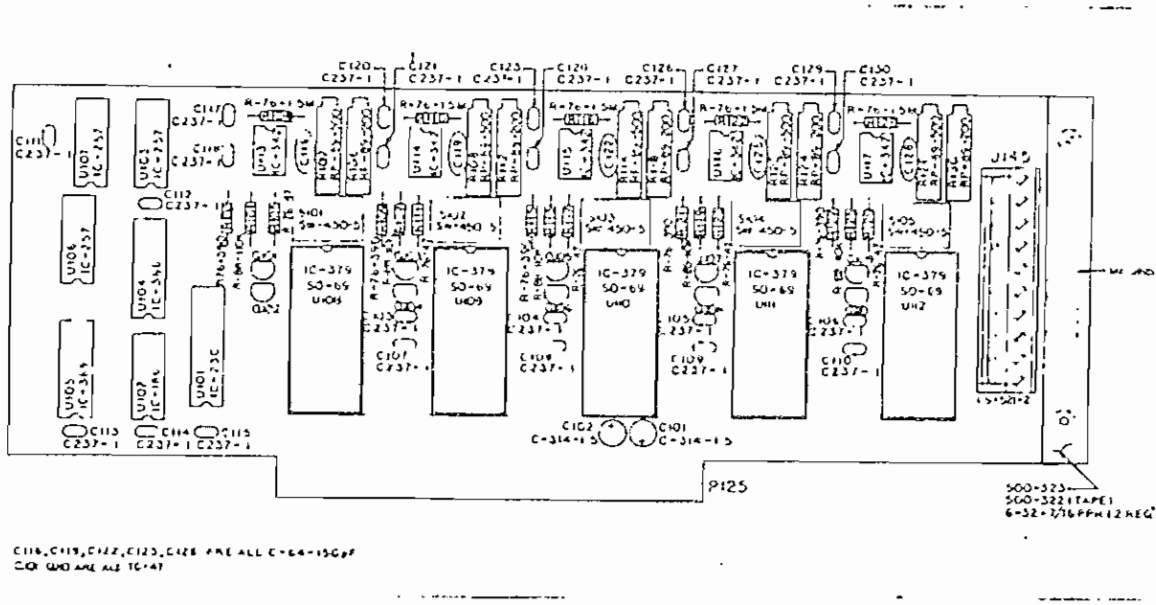


FIGURA 1.4.- Módulo de salida analogo (AOM1).

CARACTERISTICAS DE SALIDA.-

Canales de salida: 5 canales (AOM1/5).

Rangos de salida (selec. por canal): +/-2.5V, +/-5V, +/-10V (seleccionado), 0 a 5V, 0 a 10V.

Resolución: 12 bits.

No linearidad: +/-0.012% a fondo de escala (+/- 1 bms).

Impedancia de salida: 0.1 ohms.

Características de carga: 5 Kohms mín. 1000 pF máx, protección contra cortocircuito.

Estas son las principales características de tipo técnico del AOM1/5, del cual se muestra un diagrama en la figura 1.5.

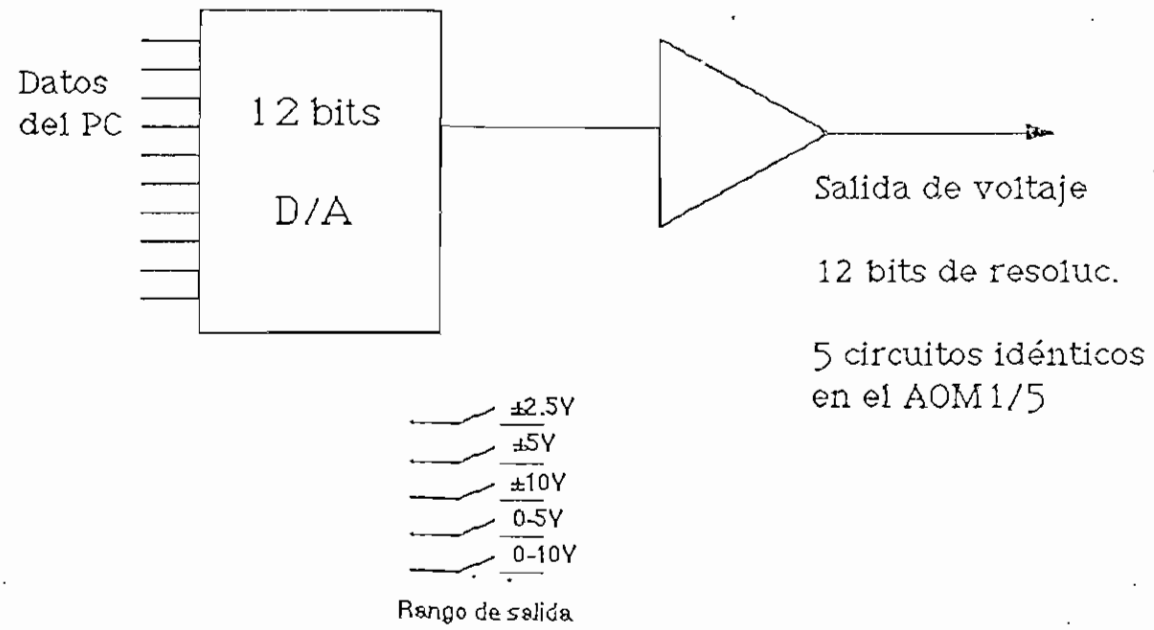


FIGURA 1.5.- Diagrama esquemático del AOM1/5.

MODULO DE SALIDA DIGITAL (DOM1).- El Módulo de salida digital, provée de 16 canales de salida, aislados ópticamente. Estos canales están configurados como salidas de voltaje TTL-compatibles (seleccionado en el equipo), aisladas

ópticamente cada una de la otra y del circuito principal. El usuario puede configurar cada canal de manera independiente para ser TTL-compatible, tener una salida de hasta +28V, o una salida de corriente, y con operación aislada o no.

Los canales digitales pueden ser accesados como 16 líneas independientes o como dos puertos de 8 canales cada uno, cada puerto es tratado como un simple byte en el software del Quick500. Este tipo de agrupamiento permite un acceso simultáneo de 8 canales, y habilita la comunicación con elementos TTL de 4 y 8 bits.

El módulo de salida digital incluye un reset al encendido (POWER-ON RESET), para asegurar una condición conocida al comenzar su operación. Este módulo puede ser ubicado en cualquier slot de la tarjeta matriz del sistema de adquisición de datos (se encuentra configurado en el slot Nº5). Un esquema de la tarjeta DOM1 se presenta en la figura 1.6. A continuación se detallan las principales características de tipo técnico del módulo de salida digital:

CANALES DE SALIDA.-

Locales: 16 canales.

CARACTERISTICAS DE SALIDA.-

Salida Lógica 0 compatible con TTL: 0.4V máx. @ 1.6mA.

Salida Lógica 1: 3.75V mín. @ 1mA.

Tiempo de transición: 100 uSeg.

AISLAMIENTO.-

Técnica: Optica.

Canal a tierra: 500V pico máx.

Canal a canal: 500V pico máx.

CARACTERISTICAS CONFIGURABLES POR EL USUARIO.-

Voltaje de salida: 28V máx.

Corriente de salida: 50mA máx.

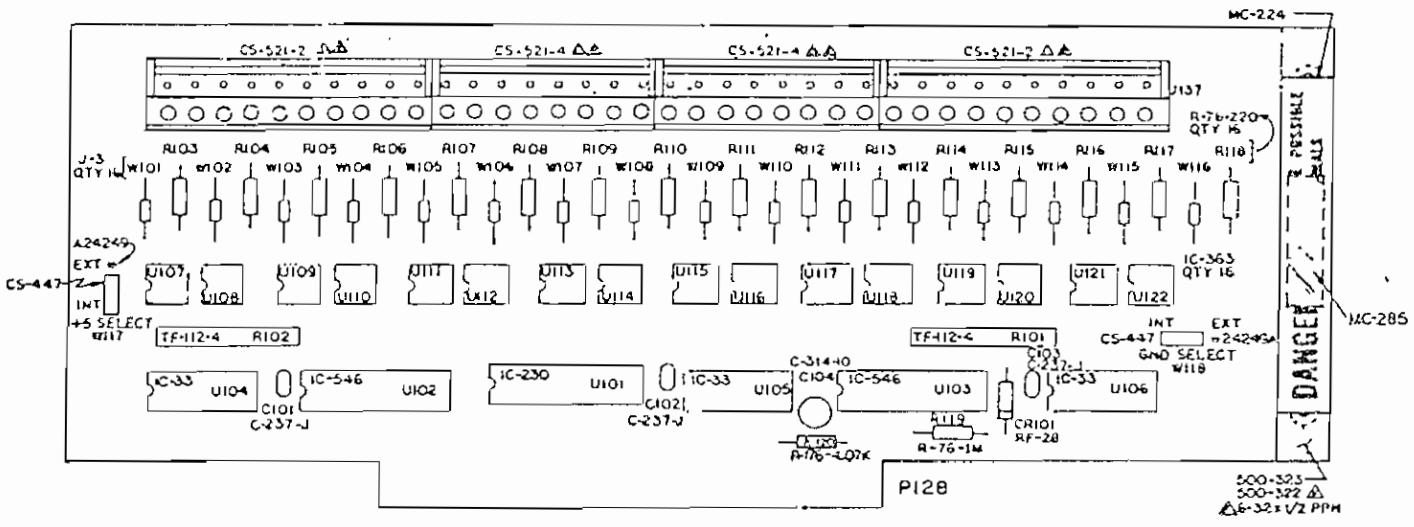


FIGURA 1.6.- Módulo de Salida Digital (DOM1).

Estas son las principales características de tipo técnico del DOM1, del cual se muestra un diagrama circuital elemental de la composición de cada canal en la figura 1.7.

1.2.2.- ESTUDIO DEL SOFTWARE DEL SISTEMA DE ADQUISICION DE DATOS KEITHLEY 500A.

En esta sección se describe el software del sistema, orientado al entendimiento de las necesidades y requerimientos del Quick500, para ser corrido, en lo que se refiere a la instalación del paquete, su configuración, requerimientos de hardware y software del sistema de computación, la relación directa del Quick500 con el QuickBASIC, convenciones para los comandos, etc.,

dejando el estudio mismo de los comandos para el segundo y tercer capítulos, donde se hace uso directo de los mismos.

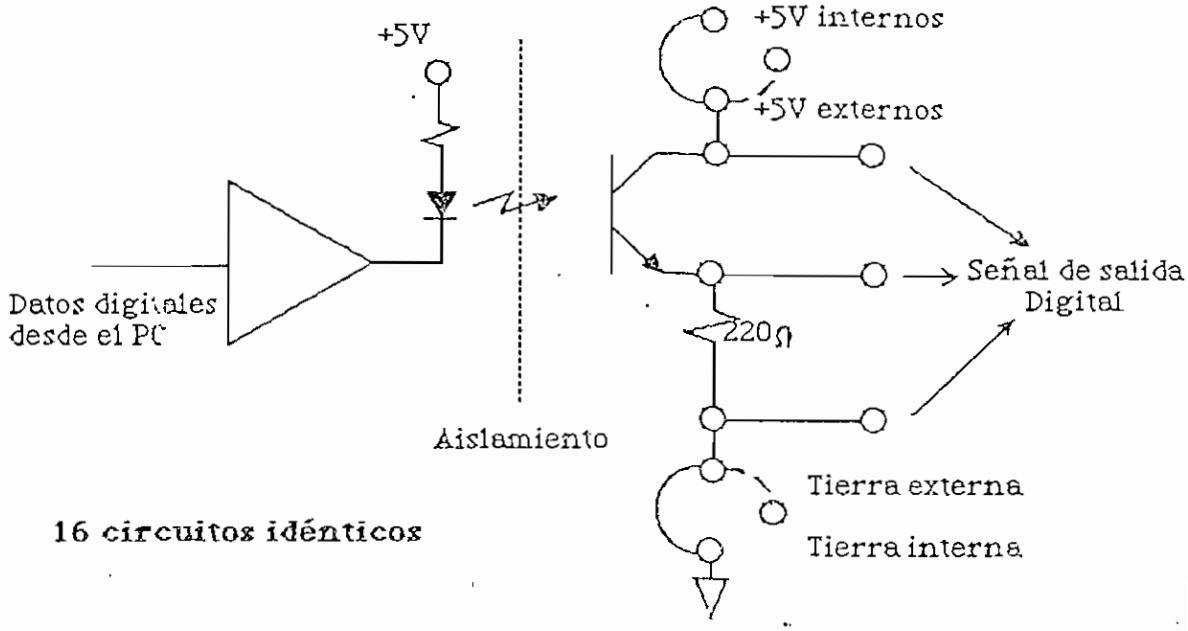


FIGURA 1.7.- Diagrama esquemático del DOMI por canal.

El QUICK500 Y SUS REQUERIMIENTOS.- El Quick500 es un paquete de software diseñado para adquisición de datos y procesos de control, el cual está escrito para ser utilizado por la estación KEITHLEY 500A, que es el equipo que nos ocupa. Este programa debe ser corrido en computadores IBM, Compaq, o computadores personales 100% compatibles.

El paquete de software Quick500 ha sido desarrollado para combinar la velocidad y la programación avanzada de un lenguaje estructurado y compilable, con las posibilidades y capacidad de un paquete de adquisición de datos no compilable como el Soft500, capaz de correr bajo varios interpretes de BASIC. El compilador seleccionado para el uso del Quick500 es el Microsoft QuickBASIC.

Este paquete además tiene algunos requerimientos de software y hardware para su adecuada utilización, que se enumeran a continuación:

MS-DOS.- La mínima versión de MS-DOS que se recomienda tener instalado en el computador es el MS-DOS 3.0, para un PC XT o AT, mientras que para un sistema PS/2, se recomienda instalar la versión MS-DOS 3.3 o mayores.

COMPILADOR.- El Quick500 ha sido diseñado para ser utilizado con el compilador QuickBASIC 4.0 o mayores.

COMPUTADOR.- El computador que se debe utilizar es un IBM PS/2, AT o XT, o computadores 100% compatibles. El programa INSTALL.EXE del Quick500, realiza un análisis profundo del computador, en cuanto a la versión del MS-DOS, espacio de trabajo en la memoria RAM, versión de memoria ROM y algunos otros parámetros en lo que a hardware se refiere. Con esta información obtenida , se crea entonces un archivo ".BAT", llamado SOFT500.BAT, adecuado al medio ambiente. Si el programa INSTALL, no se puede acomodar al hardware del computador, el programa es abortado y se muestra un mensaje de error, que indica la incompatibilidad del paquete con el hardware del computador.

Una vez superada esta prueba, se recomienda correr el Quick500 desde un disco duro, aunque podría correrse desde un drive de 1.2 MB., ya que una instalación completa del Quick500 utilizará al menos 1 MB. de memoria, por lo que resulta impráctico tratar de correrlo desde dos discos de 360 KB. o uno de 720.

El computador necesita al menos 640 KB. de memoria RAM, ya que menos memoria disminuye enormemente la cantidad de datos que se puede adquirir. El Quick500 no correrá desde un computador de 512 KB. o menos de memoria RAM disponible. Además para tener las ventajas gráficas del programa se debe tener tarjetas adaptadoras de gráficos como IBM CGA, Hércules COLOR, IBM EGA, Hércules monocromático, etc. El coprocesador matemático es un accesorio opcional, no necesario, que sin embargo, va a proveer una mayor velocidad en la ejecución de los comandos del Quick500.

Un punto importante que se debe señalar es que antes de poder instalar el software del Quick500, se debe instalar la tarjeta de interfaz en el computador para el sistema de adquisición de datos que se intente instalar. La tarjeta de interfaz puede ser seteada sea por software o por hardware, a una única dirección de memoria del computador.

INSTALACION DEL QUICK500.- Para iniciar la instalación del software se requiere de la instalación previa del hardware del sistema de adquisición de datos en el computador.

INSTALACION DEL HARDWARE.- Las tarjetas de interfaz entre el computador y el sistema de adquisición de datos son instaladas previa la corrida del programa INSTALL.EXE. Al correr este programa se desarrolla un proceso dinámico que chequea por las direcciones base, tipo y operación correcta de las tarjetas de interfaz instaladas. Si estas tarjetas no están presentes o no funcionan adecuadamente, el INSTALL es abortado.

INSTRUCCIONES ESPECIALES PARA USUARIOS DE PS/2.- . Provée soporte gráfico para CGA, EGA y VGA, este último en modo CGA solamente.

Con relación a los sistemas PS/2 modelos 50, 60 y 80, estos pueden utilizar el nuevo IBM Micro Channel Bus, para lo que requiere el interfaz 500-IBIN-PS/2. Estos computadores no aceptan el interfaz System 500.

El disco del Quick500 contiene un archivo llamado @6571.ADF, que se debe copiar en el disco de referencia antes de instalar el IBIN-PS/2. Luego de instalar este interfaz, se puede reconfigurar el sistema utilizando el utilitario "IBM Configure Utility", para seleccionar el espacio de memoria y nivel de interrupción del IBIN-PS/2 (que no pueden ser seteados por hardware). Se recomienda utilizar la dirección CFF80H e interrupciones de nivel 10, para no causar conflicto con el resto de hardware del sistema.

NOTAS PARA LA INSTALACION.- A modo de resumen se presenta un listado de los pasos que el programa INSTALL da para la instalación adecuada del Quick500:

1.- Chequear microprocesador para verificar si se trata de un 8086, 8088, 80286, etc.

2.- Chequear la cantidad de memoria RAM disponible en el sistema.

3.- Chequear que la versión de MS-DOS sea al menos la mínima requerida.

4.- Chequear la existencia de un AUTOEXEC.BAT, de existir, el INSTALL verificará la presencia del CLKSPD, que será instalado automáticamente de no estar presente. En último caso el AUTOEXEC.BAT será creado por el INSTALL.

5.- Especificar la cantidad de memoria que se desea reservar para la creación de los arreglos de datos del Quick500.

6.- Localizar las interfaces en el computador y verificar si son Series 500 o System 570.

7.- Crear un archivo llamado SOFT500.BAT en el directorio que contiene al Quick500. Este archivo arranca de manera indirecta al Quick500 y al QuickBASIC.

8.- Establecer tablas de configuración de nombres de documentos, por cada interfaz, y mostrarlas antes de terminar la instalación.

9.- El programa instruirá al usuario para presionar <ENTER>, tras lo cual, se reinicia automáticamente al computador, con lo cual la instalación queda completada.

TIPO DE COMPUTADOR.- Soporta diferentes relojes desde 4.77 MHz. a 12 MHz. con estados de espera 1 o 0. Durante la adquisición de datos, el Quick500 guarda estos datos como arreglos de memoria. El usuario está en capacidad de setear la cantidad de memoria deseada para este propósito.

Esto se lo hace mientras se corre el INSTALL.EXE, y si se instala el Quick500 con mucha o muy poca memoria, se va a obtener una serie de mensajes de error antes de que el programa INSTALL sea abortado.

INSTALACION DEL QUICK500 EN DISCO DURO.- El Quick500 debe ser instalado en el disco duro que contiene el MS-DOS con el que se arranca el computador al ser encendido (en caso de tenerse particiones o más de un disco duro). En este caso se va a requerir de un solo drive para la instalación.

La instalación del Quick500 se puede realizar en un subdirectorio separado (siempre es recomendable no cargar demasiado a la raiz del disco duro), una vez creado el directorio y ubicados en este se procede a copiar todos los archivos necesarios para la instalación adecuada del Quick500, entonces los pasos a darse son los siguientes:

```
A: <ENTER>
COPY A:*.* C: <ENTER>
```

Luego de esto se copian todos los archivos necesarios para que corra el QuickBASIC: BC.EXE, BQLB40.LIB, BRUN40.EXE, BRUN40.LIB, LINK.EXE, QB.EXE, QB.HLP, QB.LIB, QB.QLB, debiendo copiarse además todos los soportes necesarios para gráficos, mouse, etc.

SELECCION DE LOS NIVELES DE INTERRUPCION.- Normalmente el Quick500 utiliza interrupciones no mascarables para controlar la adquisición en background, debido a que tiene la más alta prioridad. Sin embargo suele suceder que algunos computadores o periféricos no son compatibles con este método de operación, pudiéndose entonces utilizar otros tres tipos de interrupción para la adquisición en background. Estos problemas son más suceptibles de presentarse en PS/2 modelo 30, AT o XT, puesto que los PS/2 modelo 50,60 y 80 utilizan la arquitectura de "bus de microcanal", la cual incluye un método propio para resolver los conflictos de interrupciones.

1.2.3.- CONFIGURACION DEL QUICK500.-

En este numeral se trata sobre la creación de una tabla de configuración del hardware "CONFIG.TBL", con el programa CONFIG.EXE del Quick500, donde se discutirá el propósito de este archivo CONFIG.TBL, explicando la importancia de este y una utilización tutorial del CONFIG.EXE, de una manera elemental que permita al usuario crear una tabla de configuración que presonalice el sistema de acuerdo a sus necesidades.

FUNCIONES DEL CONFIG.TBL.- El equipo de adquisición de datos KEITHLEY 500A es modular y expandible, por lo que las posibles combinaciones de módulos, rangos, ganancias y configuraciones de entrada y salida son prácticamente ilimitadas. El Quick500 debe conocer los nombres de los módulos presentes, la inicialización de switches y la información para controlar el hardware del sistema de adquisición de datos. Con el programa CONFIG.EXE, se crea una tabla de configuración de archivos, la que incluye toda esta información. Por defecto se va a poner un nombre a la primera tabla creada: CONFIG.TBL, pero cuando ya existan archivos previamente creados, el INSTALL sugiere nombres como CONFIG1.TBL, CONFIG2.TBL, etc.

MENUS, CONTROL DEL CURSOR Y TECLAS DE FUNCION.- Dentro del CONFIG.EXE se tienen básicamente dos pantallas de menú: HARDWARE SETUP y CHANNEL SETUP. Cada pantalla contiene diversas columnas o ventanas que muestran instrucciones y menús, que permiten elegir la configuración deseada. La selección se hace con las teclas de función <F1>....<F10>.

El cursor aparece titilando en video inverso, y es controlado por el teclado derecho de control del cursor en la pantalla. Se puede también dentro de una columna escoger opciones tipeando la letra inicial de esta.

CORRIENDO EL PROGRAMA CONFIG.EXE.- Para correr el este programa basta con tipear la palabra CONFIG <ENTER>, sin embargo se tinen una opción, la que

puede ser usada para cargar automáticamente la tabla de una archivo ya existente, omitiendo la extensión ".TBL" del archivo existente; el comando que se debe introducir es el siguiente:

CONFIG <Configuración existente> <ENTER>

ASIGNACION MANUAL DE MODULOS A SLOTS.- El usuario está en capacidad de asignar los diferentes módulos del KEITHLEY 500A en forma manual a los diferentes slots, esto siempre que no se utilice la opción SELF-ID, que identifica de manera automática los módulos y crea la tabla de configuración.

Esta técnica permite tener al usuario una configuración adecuada a sus necesidades, ubicando cada módulo de acuerdo a la conveniencia del usuario. A continuación se presenta los pasos de un ejemplo práctico de configuración, para lo cual se considera que ya se ha ingresado al CONFIG.EXE, tipeando desde el MS-DOS la línea.

CONFIG <ENTER>

Una vez que se ha ingresado al CONFIG, se presenta la pantalla de HARDWARE-SETUP sin ningún módulo de los que se dispone. Para el desarrollo de este ejemplo se toma la instalación de los tres módulos con que se cuenta y que fueron mencionados antes, los pasos que se deben seguir son los siguientes:

1.- El cursor se ubica sobre el número 1, en la columna de SLOTS.

2.- Presionando la tecla <F3>, se selecciona la opción MODULE, entonces el cursor aparece sobre la ventana de los módulos, donde utilizando las teclas de cursor se selecciona el módulo AMM1.

3.- Se presiona <ENTER>, con lo cual queda asignado el módulo AMM1 al SLOT #1, el cursor regresa de manera automática a la columna de SLOTS sobre el número 1.

4.- Con las teclas de cursor se mueve este hasta el número 3 en la columna de SLOTS, se presiona entonces <F3> y el cursor aparece en la ventana de los módulos. Se selecciona AOM1, y presiona <ENTER>, con lo que este módulo queda asignado al SLOT #3.

5.- Con las teclas de cursor se mueve este hasta el número 5 en la columna de SLOTS, se presiona entonces <F3> y el cursor aparece en la ventana de los módulos. Se selecciona DOM1, y presiona <ENTER>, con lo que este módulo queda asignado al SLOT #5.

De esta manera se completa la selección manual de los módulos para la tabla de configuración del hardware disponible, mediante el programa CONFIG.EXE.

PROGRAMACION DE LOS IONAMES COMO PARTE DE LA TABLA DE CONFIGURACION.- Esta opción que presenta el programa CONFIG.EXE es muy importante, ya que programar de manera directa los IONAMES (nombres que se pone para identificar a los diferentes canales y puertos de entrada y salida del sistema de adquisición de datos) en la tabla de configuración evita utilizar el comando CALL IONAME, cada vez que realiza un programa de prueba.

Para esta operación se utiliza la tecla de función <F5> para setear los nombres de los canales. Aquí se obtiene la posibilidad de nuevas opciones (estamos sobre la pantalla CHANNEL SETUP), que no son inmediatamente obvias para el usuario.

Proceso a seguirse:

IONAME. Presionar <ENTER>

CONFIG pregunta ahora el nuevo IONAME, se tipea el nombre y se presiona <ENTER>.

De esta forma se asignan los siguientes IONAMES:

Para canales análogos de entrada: ANLG0, ANLG1,.....,ANLG7.

Para canales análogos de salida: ANOUT0, ANOUT1,.....,ANOUT4.

Para canales digitales de salida: DOUT0, DOUT1,.....,DOUT15.

Adicionalmente se pueden realizar operaciones como copiar, borrar, renombrar, setear exactitud, ganancia local y global, etc. con los IONAMES.

SALIR O ABORTAR EL PROGRAMA CONFIG.- Para salir del programa CONFIG se presiona <Esc> tantas veces como sea necesario, o la tecla de función <F10> para retornar a la pantalla de HARDWARE SETUP.

Una vez allí, se presiona <F10> para salir. El programa pregunta entonces al usuario si quiere guardar los cambios. Si no se guardan los cambios y se sale del programa se presiona "Y"-<ENTER>. Si en cambio se quiere guardar los cambios se presiona "N"-<ENTER>, con lo cual sin embargo no se sale del programa. Para salir se vuelve a presionar <F10>-"Y"-<ENTER>.

Pero también se puede necesitar abortar el programa de manera directa aunque se pierdan todos los cambios realizados, en cuyo caso se tipea la combinación <ALT-F10>, lo que nos ubica directamente en el DOS.

1.2.4.- MODOS DE OPERACION DEL QUICK500.-

Para iniciar el estudio de los modos de operación del Quick500 se asume un buen nivel de familiarización con el QuickBASIC y haber creado ya la tabla de configuración de archivos (GONFIG.EXE) para el sistema de adquisición de datos instalado en el computador.

El Quick500 puede ser arrancado y corrido en diferentes modos de operación y para entenderlos, primero se va a revisar las funciones de algunos archivos del Quick500.

QUICK500.BAT.- Este archivo corre el SOFT500.BAT, el cual carga el editor del QuickBASIC. Para comenzar una sesión de trabajo con el Quick500, se debe ingresar únicamente el comando "QUICK500" desde el DOS, con lo cual se ingresa al editor del QuickBASIC, con el Quick500 completamente funcional.

SOFT500.BAT.- Este archivo carga el S500.EXE, seteando un área separada de memoria, carga además la tabla de configuración del hardware, e identifica el tipo de computador que se está utilizando. Este programa puede ser cargado desde otro archivo ".BAT" del Quick500. SOFT500.BAT es creado al correrse el programa INSTALL del Quick500.

QLIB.QLB y QLIB.LIB.- Son las librerías que entienden las llamadas (CALL) a comandos del Quick500. Estos comandos comunican al usuario con el sistema de adquisición de datos. QLIB.QBL debe estar en el disco desde el cual se ingresa al editor del QuickBASIC. El QLIB.LIB es un archivo de librería requerido para compilar los programas de prueba y producir un archivo estándar de extensión .EXE, si se intenta esta operación sin tener estas librerías se obtiene mensajes de error.

S500.EXE.- Programa residente en memoria que controla el hardware de adquisición de datos, debe estar siempre presente para correr un programa del Quick500. El archivo QUICK500.BAT lo carga en memoria antes de cargar el editor del QuickBASIC, lo cual permite correr programas del Quick500 dentro del editor de QuickBASIC. QRUN.BAT y QLOAD.BAT son cargados a memoria también por el S500.EXE, mismo que debe estar en el disco cuando se corran programas compilados por el QuickBASIC.

QRUN.BAT.- Se encarga de correr programas que han sido compilados a la forma ejecutable del MS-DOS (.EXE). Po lo tanto el QRUN no ingresa al editor del QuickBASIC, y al terminar el uso del programa, regresa directamente al MS-DOS.

Al correr el QRUN, este carga el S500.EXE en memoria de manera temporal, mientras corre el programa compilado .EXE del Quick500, al terminar la sesión el S500.EXE es descargado de memoria automáticamente. La sintaxis para este comando es:

QRUN <nombre del programa> <arg1> <arg2> .....

QLOAD.BAT.- Carga el S500.EXE en memoria y lo hace residente de manera permanente, luego el QLOAD retorna al DOS, donde corre los programas compilados simplemente ingresando el nombre del documento en el DOS. La sintaxis para este comando es simplemente QLOAD.

METODOS PARA CORRER EL QUICKBASIC Y QUICK500.- Las vías para correr el compilador del QuickBASIC y los archivos .BAT del Quick500, se describen en los siguientes párrafos. En cada caso QLIB.QLB y QLIB.LIB deberán estar accesibles en el disco duro.

1.- OBJETIVO: Correr el compilador del QuickBASIC desde el editor para crear un archivo ejecutable (.EXE).

METODO: a.- Desde el DOS se ingresa el comando: QUICK500 <ENTER>. El sistema corre el SOFT500.BAT y carga el S500.EXE en memoria. Este corre el QuickBASIC con el argumento "/L QLIB.QLB", y QLIB se convierte en la librería básica.

b.- Escribir y probar el programa en el editor, cuando este funcione adecuadamente, se recurre al menú de barra "RUN" y se seleccionar la opción "Make EXE".

c.- Completar los pasos restantes del QuickBASIC para producir un archivo ejecutable.

2.- OBJETIVO: Correr un programa compilado .EXE con el compilador del QuickBASIC, sin hacer permanentemente residente en memoria al S500.EXE.

METODO: a.- Ingresar el comando: QRUN <nombre del programa>.

b.- QRUN.BAT carga al S500.EXE y ejecuta el programa de prueba, al terminar la sesión aborta el S500.EXE de memoria.

3.- OBJETIVO: Correr un programa ejecutable compilado con QuickBASIC, tipeando solo el nombre del archivo.

METODO: a.- tipeando el comando QLOAD <ENTER> desde el DOS, hacer residente en memoria al S500.EXE. No se debe perder de vista que este método reserva memoria para el S500.EXE, lo cual reduce el monto de memoria disponible para el DOS.

b.- Ingresar el nombre delarchivo .EXE desde el DOS.

4.- OBJETIVO: Compilar un programa de Quick500 sin entrar al editor del QuickBASIC y sin el S500.EXE. Un programa de Quick500 puede ser compilado antes del cargar el S500.EXE, pero no puede ser corrido. La practicidad de este método se observa cuando el programa fuente, el S500 y el compilador son demasiado para la memoria del computador, por lo que puede ser deseable compilar un programa desde el DOS directamente, y sin el S500.EXE residente en memoria.

METODO: a.- Desde el DOS se tipea el siguiente comando: BC <programa.BAS> <ENTER>, lo cual produce un archivo llamado <programa.OBJ>, el cual es utilizado para crear un archivo ejecutable.

b.- Todavía en el MS-DOS se tipea el siguiente comando:

LINK <programa.OBJ> <programa.EXE>, ,QLIB.LIB

Con lo que se produce un archivo ejecutable llamado <programa.EXE> semejante al producido desde el editor del QuickBASIC.

1.2.5.- INICIALIZACION DEL HARDWARE.-

La inicialización del hardware se realiza mediante el programa HARDINIT.EXE incluido en el paquete del Quick500, que inicializa todos los módulos del sistema cuando el sistema es encendido. Se asegura además que dicha inicialización se realice una vez que se haya cargado y ejecutado el AUTOEXEC.BAT. Se deben observar las siguientes precauciones:

1.- Si bien el HARDINIT está dentro del AUTOEXEC, la inicialización del sistema no ocurrirá mientras este no se haya encendido.

2.- El HARDINIT utiliza las tablas de configuración creadas en el CONFIG para identificar e inicializar los módulos de salida. El formato del comando HARDINIT es el siguiente:

HARDINIT -c0xAAAA <nombre> -p ; donde:

-c: especifica que el siguiente parámetro es una dirección de interfaz y tabla de configuración de archivos.

0xAAAA: es la dirección de memoria de la interfaz. AAAA representa 4 dígitos más significativos de la dirección de memoria en hexadecimal.

<nombre>: el nombre completo de la tabla de configuración del CONFIG, incluyendo el path completo de localización.

-p: es una opción de pausa que advierte en pantalla que se debe encender el sistema de adquisición de datos. Se debe presionar <ENTER> antes de comenzar la inicialización.

La secuencia "-c0xAAAA <nombre>" debe ser ingresada para cada sistema de adquisición de datos presente en la inicialización. Si se utiliza la opción de pausa -p, esta se coloca una sola vez en el HARDINIT.

Un ejemplo de como debe ser el AUTOEXEC.BAT incluyendo el HARDINIT se presenta a continuación:

```
ECHO OFF
CLS
CLKSPD
HARDINIT -c0xAFF8 CONFIG1.TBL -p
DATE
TIME.
```

1.2.6.- QUICK500 Y QUICKBASIC.-

Antes del aparecimiento del Quick500, se utilizaba el Soft500, donde los comandos CALL ..., no podían ser compilados, debido a que la mayoría de compiladores son sumamente costosos. Sin embargo la introducción de Microsoft QuickBASIC, provée un compilador barato, capaz de compilar porgramas escritos en BASIC estándar. Sin embargo de esto el QuickBASIC no puede resolver los problemas de un CALL en Soft500, por lo que aparece el Quick500, el mismo que introduce una librería "QLIB.EXE", la que actúa como interfaz entre las sentencias CALL y el S500.EXE.

CONVENCIONES PARA COMANDOS DEL QUICK500.- Los comandos del Quick500 deben respetar algunas convenciones de forma de poder ser reconocidos como tales dentro del QuickBASIC, en este numeral se trata de describir las más importantes. Así se tiene que todos los comandos del Quick500 deben comenzar con la palabra CALL, sin ninguna excepción. Por ejemplo el comando CALL INIT, inicializa el hardware de adquisición y la memoria del computador.

Todos los comandos siguen este formato o sintaxis, pero algunos comandos permiten incluir información adicional acerca de su función mediante parámetros. Estos parámetros se enlistan y también deben seguir una convención que obliga a que esta lista vaya encerrada entre paréntesis, además cada parámetro debe estar presente, ninguno es opcional. Por ejemplo el

comando INTON, que habilita las interrupciones para trabajo de adquisición o salida de datos, tiene el siguiente formato:

CALL INTON (rate%, "mil")

La lista se compone de constantes, variables o valores mixtos, en el ejemplo todos los parámetros posibles del comando INTON se incluyen en la lista, no se asumen valores por defecto u omisión de parámetros.

Cuando un comando es especificado como entrada, se puede usar constantes o variables, pero si es de salida, solo se puede utilizar variables, puesto que el CALL retornará coloca datos en dichos parámetros, esto es importante de tomar en cuenta para la especificación correcta de ambos tipos de parámetros.

Parámetros mal ubicados o tipeados causan resultados impredecibles y probables fallas en el sistema. Existen algunas palabras reservadas en el Quick500 tales como "MAGNIFY", "PAGEO", "BT", "WBT", "WGO", "NT", etc. que no pueden ser utilizadas a no ser para el motivo que fueron predefinidas.

INICIALIZACION DEL PROGRAMA.- El Quick500 tiene dos comandos que ejecutan diferentes tareas de inicialización del sistema de adquisición de datos, computador y área de memoria, estos son INIT y SOFTINIT.

SOFTINIT.- Es importante indicar que este comando debe utilizarse antes de cualquier otro comando del Quick500. Además este comando es indispensable al comenzar cualquier programa realizado en Quick500.

El comando SOFTINIT no afecta las zonas de hardware y software afectadas por el comando CALL INIT. El comando SOFTINIT inicializa variables y parámetros internos que pueden ser seteados por el compilador del Quick500 para operar apropiadamente. Debido a esto SOFTINIT no es opcional y debe usarse antes de cualquier otro comando.

INIT.- Este comando si es opcional. Inicializa algunos elementos del sistema y resetea el hardware del sistema de adquisición de datos, tiene funciones específicas que se describen a continuación:

1.- Resetea todos los canales de salida análoga y digital a cero en el sistema de adquisición de datos. los relés conectados a los módulos de control son abiertos. CALL INIT carga y lee el CONFIG.TBL para determinar la ubicación de los módulos de salida y sus rangos de trabajo.

2.- Enciende todos los interruptores que deben estar encendidos.

3.- Remueve los IONAMES de memoria, los mismos que pueden ser creados por comandos del Quick500, cuando no es realizado esto por el CONFIG.TBL.

4.- Borra las áreas de memoria para datos. Todo dato guardado con anterioridad es perdido, por lo que se debe utilizar con cuidado y salvar los datos obtenidos antes de su utilización. Se utiliza frecuentemente al inicio del programa, a continuación del CALL SOFTINIT.

1.2.7.- CARACTERISTICAS Y CAPACIDAD DEL SISTEMA.-

Uno de los tópicos de mayor importancia es el manejo de memoria del Quick500 en un Computador Personal IBM o compatible, el mismo que es extremadamente sofisticado y flexible, acomodándose a las necesidades de un sistema de adquisición de datos y aplicaciones de control, por lo que debe cumplir con porpósitos básicos que son:

1.- Proveer áreas grandes para datos.

2.- Proveer estructuras de datos con características especiales.

3.- Liberar un espacio de memoria suficiente para el QuickBASIC.

4.- Permitir implementar 2 sistemas de manejo de memoria, para el Quick500 y el QuickBASIC, cada uno de estos con sus propias ventajas para hacer el mejor uso de ellas.

Debido a la naturaleza del trabajo que se ha de realizar con un sistema de adquisición de datos, resulta muy importante reservar la mayor cantidad de memoria para arreglos de datos, sin embargo la cantidad de memoria disponible para este propósito depende de: la versión del DOS, tamaño del programa desarrollado, programas residentes en memoria, etc.. Si se selecciona un espacio de memoria demasiado grande, se recibe una serie de mensajes de error del Quick500 y del QuickBASIC.

La vía más fácil para reservar el área de memoria es correr el INSTALL, aunque una alternativa es utilizar un procesador de palabra en ASCII, para modificar directamente el SOFT500.BAT.

Ahora bien una vez que se ha reservado el espacio de memoria para datos es importante saber de que forma se crean las áreas de memoria para utilización directa desde el Quick500.

CREACION DE AREAS DE MEMORIA PARA USO DEL QUICK500.- En general los comandos de entrada de datos (ANIN, DIGIN, ANINQ, etc.) crean áreas para el Quick500 automáticamente al ser ejecutados, no así los comandos de salida de datos (ANOUT, DIGOUT, etc.) en cuyo caso las áreas de memoria deben crearse con anterioridad mediante el uso del comando ARMAKE (ARray MAKE). Por ejemplo se crea un arreglo de nombre "arreglo%", donde el ("%" denota que cada elemento del área ocupará 2 bytes de memoria), el arreglo consta de 100 elementos con una anchura de 2; la sintaxis de este comando será:

```
CALL ARMAKE("arreglo%", 100!, 2, " ")
```

El nombre que se le da al área creada, permite un acceso fácil de otros comandos a esta área refiriéndose a ella por su nombre para recuperar, introducir datos en ella, salvar, cargar el área en memoria. Se tiene básicamente cuatro tipos de áreas creadas para el Quick500:

1.- BIT:&.- Area empaquetada por bits, donde los valores guardados son 1 o 0. Utilizado con entradas y salidas digitales del Quick500, cuando el acceso es por canales.

2.- BYTE:@.- Cada elemento tiene 8 bits, almacenando valores entre 0 a 255. Se utiliza especialmente por entradas y salidas digitales cuando se accesa a pórticos digitales.

3.- WORD:%.- Elementos de 2 bytes, almacena valores de 0 a 65535. Utilizado por entradas y salidas análogas.

4.- REAL:!.- Elementos que consumen 4 bytes, almacenados en el formato del punto flotante de simple presición del 8087.

Estas áreas de elementos reales no son soportadas por ningún comando de adquisición de datos del Quick500.

Siempre se debe especificar en el nombre del área o arreglo de que tipo es este, para que el Quick500 sepa cuanta información debe guardarse o accesarse para cada dato, así como el área total destinada para el arreglo.

El Quick500 ha previsto una serie de comandos para incrementar la utilidad de las áreas creadas, así estas pueden salvarse (ARSAVE), borrarse (ARDEL), cargarse a memoria (ARLOAD), pedir información para reportarla (ARSTATUS), verificar la ubicación del último elemento del arreglo al que se accesó (ARLASTP), etc. Finalmente con los datos de un arreglo se puede hacer análisis de datos, proceso de gráficos, etc. tanto en valores binarios como el unidades de ingeniería.

1.2.8.- BACKGROUND/FOREGROUND.-

El concepto de BACKGROUND/FOREGROUND es muy fácil de entender si se examina la tarjeta de interfaz entre el KEITHLEY 500A y el computador. Esta tarjeta contiene un generador de interrupciones, que trabaja junto con las interrupciones no mascarables (NMI) del computador. El comando INTON (INTerrupt ON) permite especificar la frecuencia de estas interrupciones.

CALL INTON(rate%, ut$)

Donde ut$ indica las unidades de tiempo utilizadas (seg, min, etc.), mientras que rate%, es un valor entero que indica el número de unidades de tiempo que dura la interrupción.

Cada interrupción el procesador salta desde el FOREGROUND, donde ejecuta la tarea principal, a ejecutar tareas de BACKGROUND, donde se realizan las tareas de fondo. Cuando esta tarea se completa, se regresa el control al foreground hasta la siguiente interrupción. Este traspaso del control del sistema se realiza a intervalos constantes, con el ánimo de maximizar la exactitud en las tareas de adquisición de datos

El sistema de adquisición de datos utiliza la interrupción NMI (no mascarable), por lo que tiene la mayor prioridad. Los comandos que no son de background y los comandos ordinarios del BASIC se llaman comandos de foreground, y son ejecutados durante el tiempo en que no se atiende a las interrupciones. Esta operación conjunta de los dos modos de trabajo se denomina multitarea, y si bien se ejecutan simultáneamente, el computador lo que hace en realidad es switchear la atención a ambas tareas.

Tanto foreground como background pueden actuar independientemente, donde además la programación de ambos es muy flexible. Mientras que en

background se ejecutan secuencias de adquisición de datos, el foreground se puede utilizar para monitorear estatus de las tareas del background, comunicación con periféricos, usuario, graficación, análisis, etc.

Dependiendo de la frecuencia de las interrupciones para adquisición o salida de datos y de la cantidad de memoria disponible en el computador se puede incluso correr otra aplicación.

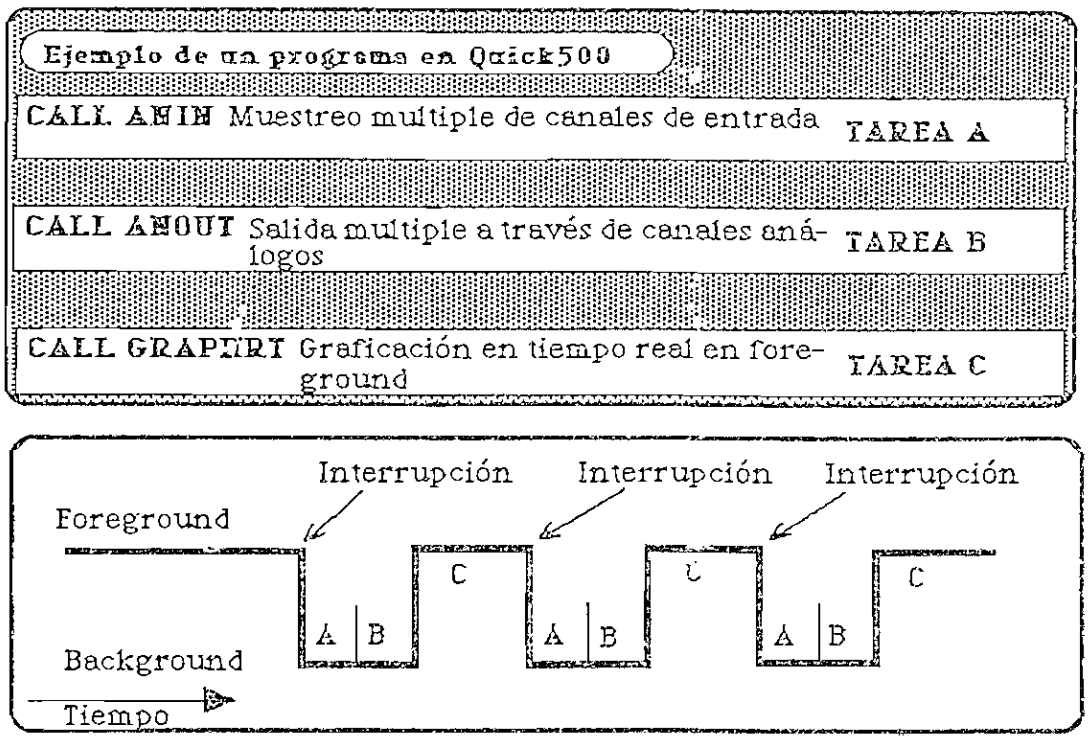


Figura 1.8.- Ejecución de tareas foreground/background en Quick500

COMUNICACION BACKGROUND/FOREGROUND.- Para comenzar, Quick500 ofrece una capacidad de disparo de sus comandos, de esta manera comandos de background pueden ser disparados por comandos del foreground. Existe también disparo desde background mismo, lo que permite una actividad independiente de este modo de trabajo para realizar actividades de adquisición y control.

Ciertos comandos del Quick500 (DIGINTRIG, SCHMITRIG), pueden actuar como disparo del background, con lo que se reacciona directamente frente a eventos del mundo real como entradas digitales o análogas. Estos comandos

eventos del mundo real como entradas digitales o análogas. Estos comandos pueden ser enlazados con comandos del background como el ANIN o DIGOUT mediante el parámetro "modo de disparo". Cuando el Quick500 encuentra este parámetro, la tarea es seteada en el background, pero no disparada, permaneciendo latente un tiempo más, después de lo cual, la tarea comienza sin necesidad de ser disparada por el foreground, detalles en el capítulo 3.

Una de las características del background más utilizadas es que este modo de trabajo una vez que ha establecido una secuencia de trabajo, no requiere de programas de control. Por ejemplo si se tiene un ARSAVE, automáticamente el background sabe que esta no se ha de ejecutar mientras no se termine la adquisición o control.

Así, se puede utilizar comandos como STATUS, que reporta si una tarea de background se está ejecutando, espera por un disparo o si ya se ejecutó.

Otro comando de gran valor es el ARLASTP, el mismo que recupera la ubicación del último punto del arreglo que fue accesado por un comando del Quick500. Su importancia radica en que permite sincronizar cálculos en el foreground con las adquisiciones o salidas del background. ARLASTP se ejecuta cuando se quiere realizar un análisis con parte de un arreglo, mientras el resto del arreglo sigue siendo adquirido, es usualmente utilizado junto con comandos de escritura o recuperación de datos en los arreglos como ARPUTVALI, ARPUTVALF, ARPUTI, ARPUTF, ARGETI, ARGETF, etc.

El background es habilitado por el comando INTON, el mismo que puede ser utilizado antes o después de que los comandos del background hayan sido seteados, esto permite arrancar un número de tareas simultáneamente ya que estas son sincronizadas. En forma similar se tiene el comando INTOFF, sea o no que las tareas estén corriendo. Cuando el INTOFF es utilizado, las tareas no son

limpiadas, ya que el background puede ser temporalmente parado para ser reiniciado más tarde desde el mismo lugar.Si se desea limpiar las tareas de background , se debe correr los comando BACKCLEAR, HALT.

TIMERS EN QUICK500.- Junto con la adquisición de datos y funciones de control. los timers del Quick500 pueden operar también en el background. Cada vez que ocurre una interrupción, todos los timers corren y se actualizan, estos timers requieren que el background esté en operación, pero si este es suspendido, los timers se paran y retienen el último valor hasta ser reiniciados. Una nueva habilitación del background resetea todos los timers, y estos comienzan su cuenta desde cero.

DISPARO DESDE BACKGROUND.- El propósito de un disparo desde background es vigilar la entrega de una condición sin detener el proceso del foreground. Cuando se cumple la condición, la tarea es activada o disparada y la tarea de disparo se desactiva. Para realizar un disparo desde el background solo se necesita cumplir con dos pasos, primero llamar un comando del Quick500 con el parámetro "modo de diparo" en "bt" (background trigger), este setea la información acerca del disparo. Un segundo paso es llamar un comando con el parámetro "modo de disparo" en "wbt" (wait a background trigger). Estos dos comandos pueden quedar separados por varias líneas de programa, pero que no contengan un "bt" adicional.

Al correr un programa, el comando "bt" no es activado si se lo encuentra primero, únicamente se setea la tarea sin ser comunicada al background, y el control pasa directamente al siguiente comando. Cuando es encontrado un comando "wbt", las tareas "bt" y "wbt" son comunicadas juntas al background, se activa la tarea "bt" que comienza por chequear la condición de disparo así como la siguiente interrupción. La condición de disparo se chequea cada

interrupción, la misma que al cumplirse dispara la tarea "wbt" mientras que la tarea "bt" se auto desactiva.

Son 8 los comandos que tienen la capacidad de tener disparo desde background ("bt"), estos son:

DIGINTRG	FREQIN	DIGIN	ANIN
SCHMITRIG	PULSEIN	DIGOUT	ANOUT

Mientras que 12 comandos tienen la capacidad de esperar un disparo desde BACKGROUND ("wbt"), estos son:

ANIN	FREQIN	GONOW	TIMERSTART
DIGOUT	ANOUT	HALT	PULSECOUNT
DIGIN	CLOCKREAD	PULSEIN	TIMERREAD

Las interrupciones no necesariamente deben estar activadas cuando comandos con "bt" y "wbt" son llamados, sin embargo si deben activarse para ser ejecutados estos comandos.

Este tipo de disparo obtiene una respuesta sumamente rápida, en el orden de los useg., ya que las tareas que esperan disparo "wbt", son ejecutadas inmediatamente después de las tareas "bt", siguiendo una secuencia de background.

1.2.9.- INTEGRACION DEL SOFTWARE Y HARDWARE.-

Cuando al sistema de adquisición de datos y control se conectan fuentes de voltaje, corriente y transductores compatibles con el mismo, existen ciertos requerimientos para la operación correcta del sistema. El KEITHLEY 500A está capacitado para manejar los valores binarios puros, o en su defecto unidades de ingeniería de acuerdo al módulo con el que esté trabajando, así se puede trabajar

con entradas y salidas de voltaje, termocuplas en cuyo caso se trabaja en grados celcius, iguales unidades para las RTD's y sensores semiconductores de temperatura, se puede trabajar con niveles de voltaje adecuados para strain gages y celdas fotovoltáicas, entradas de corriente de lazo (4-20 mA) normalizadas para sistemas de control industrial.

Dado que se cuenta con módulos que permiten trabajar unicamente con niveles de voltaje, se hace referencia a este tipo de relación con el mundo exterior de manera un poco más detallada.

1.- Valores Binarios (EUF% = -1).- El parámetro EUF% (Enginnering Unit Flag) debe siempre incluirse en la lista de parámetros. Los valores binarios son siempre positivos y enteros y su rango depende del número de bits de los conversores A/D o D/A, así: de 12 bits 0 - 4095, 14 bits 0 - 16383 y 16 bits 0 - 65535. Cuando se utilizan valores binarios, el Quick500 no tomará en cnenta ganancias o rango de información, porque los datos son simplemente proporcionales a cualquier rango usado.

2.- Entradas y Salidas de Voltaje (EUF% = 0, 1, 2).- El parámetro EUF% = 0, señala voltios, 1 señala mV y 2 señala uV, con lo que el resultado de un conversor A/D puede ser leido directamente en voltaje. De igual forma los valores de voltaje pueden ser escritos directamente en un canal de salida análoga durante una conversión D/A. El sistema toma en cuenta los rangos, amplificaciones y atributos seleccionados en software y resolución de conversión de manera automática.

1.3.- ADQUISICION DE DATOS Y CONTROL EN TIEMPO REAL.-

Una vez que se ha estudiado el software y el hardware del sistema de adquisición de datos y control Keithley 500A, se analizan ligeramente algunos

conceptos básicos y definiciones fundamentales que ayudarán a comprender los objetivos que se persiguen en el desarrollo de los siguientes capítulos.

Una de las tareas fundamentales para el control en es la **adquisición** de datos, que consiste en una recolección de información que describe una situación dada dentro de una planta o proceso. Desde el punto de vista del control automático, los datos adquiridos describen la dinámica de la planta y además reflejan lo que ha sucedido en la planta mientras se satisfacía una condición, la misma que suele estar definida por una base uniforme de tiempo, pero que puede ser controlada por cualquier evento.

Los sistemas de **tiempo real** se caracterizan por la habilidad de ejecutar la adquisición de datos y/o control en un intervalo de tiempo apropiado. La rapidez y la precisión con la que el sistema de adquisición de datos y control debe responder depende de los requerimientos de la aplicación (planta).

En general, la tendencia actual, es la de aplicar control en tiempo real a un **proceso** determinado, el cual consiste de una serie de cambios graduales, debidos a una secuencia de acciones de control que tiene como fin obtener un producto o productos afines terminales. Estos procesos pueden ser de diversa naturaleza tales como físicos, químicos, energéticos, biológicos, de manufactura, etc. y pueden estar afectados por perturbaciones, inclusive de tipo aleatorio.

Entonces si en base a datos de entrada (adquisición) se realiza un procesamiento de acuerdo con algún algoritmo, que va a generar unas señales de salida (control) dentro de un intervalo de tiempo determinado, se está realizando en definitiva un control de procesos en tiempo real.

La figura 1.9, muestra un esquema de adquisición de datos, procesamiento y control en tiempo real.

Debe tenerse presente que un sistema de adquisición de datos como el que disponemos, por si solo no constituye una unidad de adquisición de datos y control, puesto que si bien dispone del hardware necesita del computador, y del software adecuado que le "indiquen" las tareas que debe ejecutar y en que momento.

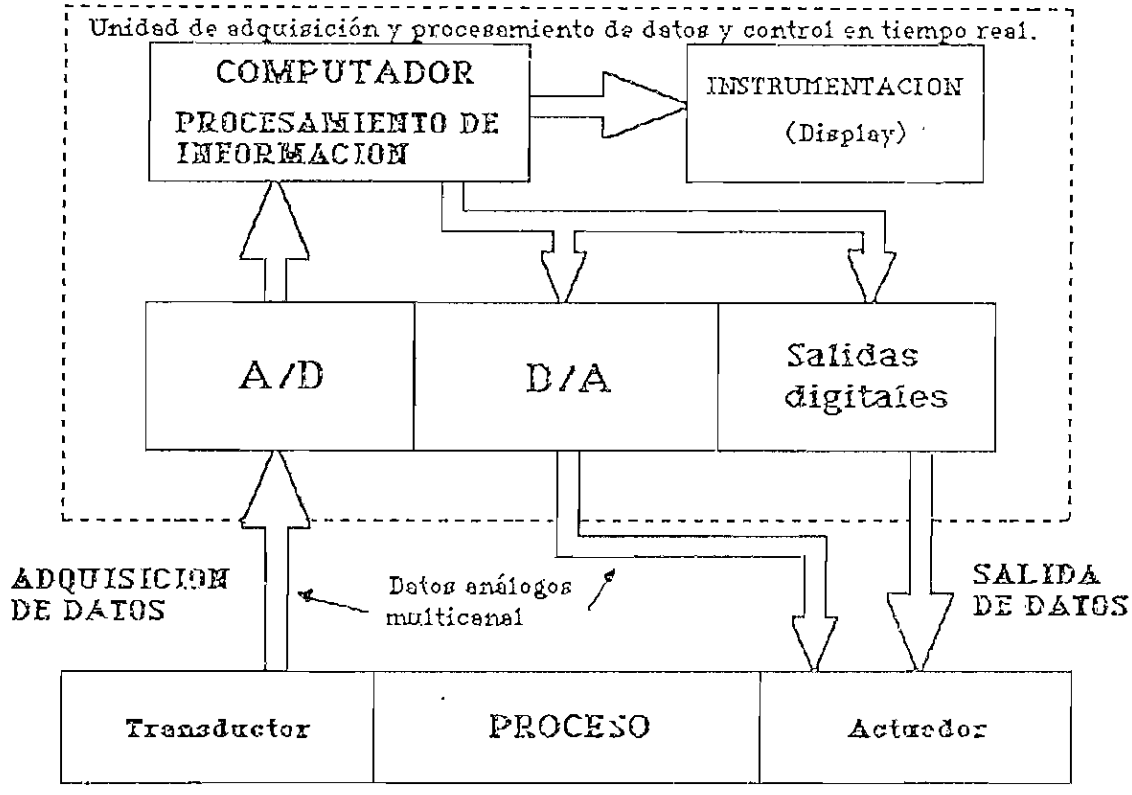


Figura 1.9.- Adquisición y procesamiento de datos y control en tiempo real.

Tampoco un computador común podría realizar este tipo de trabajo de manera independiente, puesto que carece de las interfaces que le van a permitir la adquisición de datos para su respectivo procesamiento y luego con los resultados realizar el control en tiempo real.

Sin embargo su combinación entrega una poderosa unidad de adquisición y procesamiento de datos y control en tiempo real , cuyo esquema básico de trabajo

se muestra en la figura 1.10. La configuración que se muestra es la existente en la facultad de Ingeniería Eléctrica, y que se utiliza para el desarrollo de esta tesis.

En la figura mencionada no se debe perder de vista que tanto el transductor como el actuador, deben incluir acondicionadores de señal, ya que se trabajará con señales normalizadas.

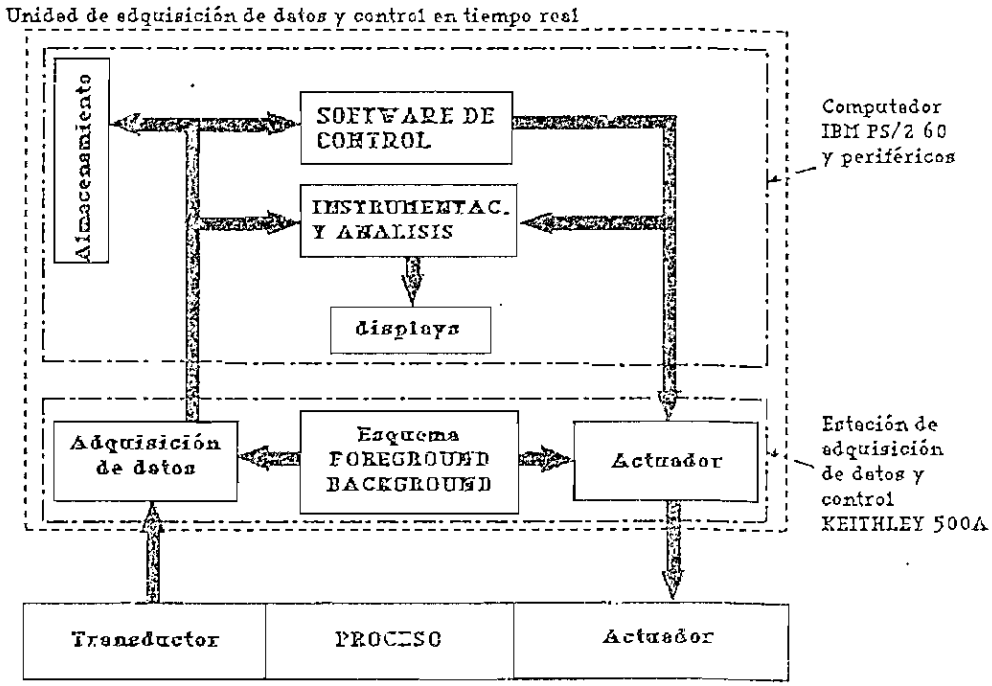


Figura 1.10.- Unidad de adquisición y procesamiento de datos y control en tiempo real.

Una vez que se ha establecido esta unidad, surge la necesidad de incorporarla a un esquema más real de control e instrumentación, así se puede mencionar como ejemplo el análisis del status (instrumentación) y el control de una planta o proceso en que se requiere de mantener ciertas variables de interés, como temperatura, presión, nivel de líquidos, voltaje, corriente, posición, etc., lo más cercano posible a ciertos valores deseados, denominados valores o puntos de referencia, por lo que surge la necesidad de regular dichas variables.

Normalmente se necesitaba, toda una instrumentación basta y compleja (por razones de seguridad, confiabilidad y operación), sin embargo actualmente se han desarrollado técnicas que permiten la utilización del computador en línea (ON-LINE), esto es, como un elemento activo del lazo de control, dejando atrás su antiguo papel de auxiliar remoto, que actua independientemente y fuera del lazo de control (OFF-LINE).

Dada la naturaleza de las aplicaciones posibles en nuestro medio, se pone énfasis en el esquema de control digital directo, el mismo que incluye al computador dentro del lazo de control, por lo que constituye un control e instrumentación en línea. En este esquema mostrado en la figura 1.11 el computador está conectado al lazo de control a través de una interfaz de conversión analoga-digital A/D, y digital-análoga D/A del sistema de adquisición de datos y control, el cual incluye manipulación de períodos de muestreo (en esquema backgrouud). Este sistema se conoce también como sistema de datos muestreados ya que se maneja señales tanto continuas como discretas.

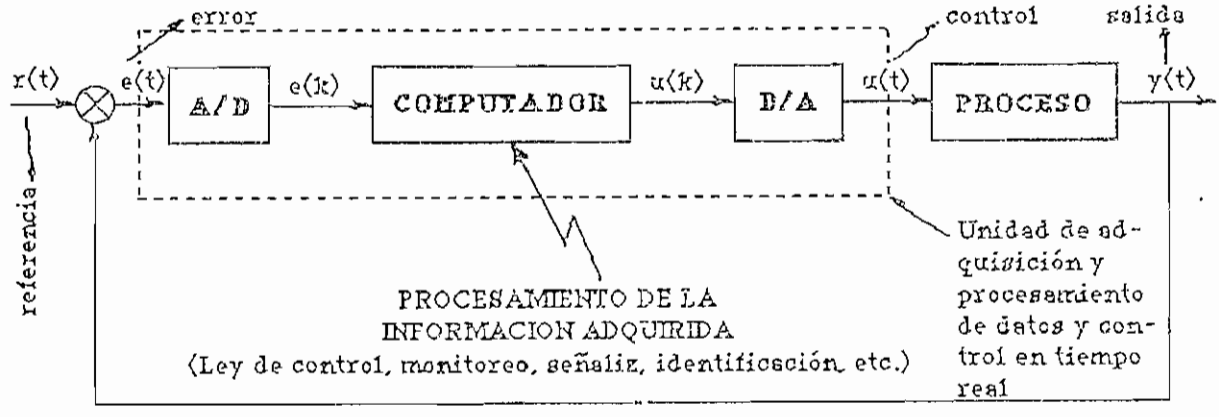


Figura 1.11.- Control digital directo. (diagrama de bloques).

La gran ventaja de tener un esquema en tiempo real como el propuesto es que dada la gran rapidez de comunicación, interconexión, capacidad de memoria y costo (relativamente bajo) de los actuales computadores, unido esto al poderío del software dedicado disponibles en el mercado, se obtiene sistemas de gran precisión, rapidez y flexibilidad; flexibilidad que no se consigue con sistemas de control e instrumentación construidas en base a hardware. El software proporciona esta flexibilidad que permite aplicar el mismo algoritmo de control a diferentes plantas, cambiando únicamente ciertos parámetros, de acuerdo a la planta o proceso controlado, e inclusive el cambio de la ley de control es ahora más sencillo y versátil pues se opera sobre un software y no sobre un hardware difícil y costoso de cambiar. Tanbién en este esquema se puede realizar una instrumentación más eficiente (inteligente) del status de la planta con mayor detalle, pues incluye señalizaciones, monitoreo, etc..

Para terminar este capítulo introductorio, se presenta un diagrama que trata de resumir las diferentes actividades en cuanto a adquisición y procesamiento de datos y control que se desarrollan en los subsiguientes capítulos de esta tesis. El diagrama mencionado se presenta en la figura 1.12.

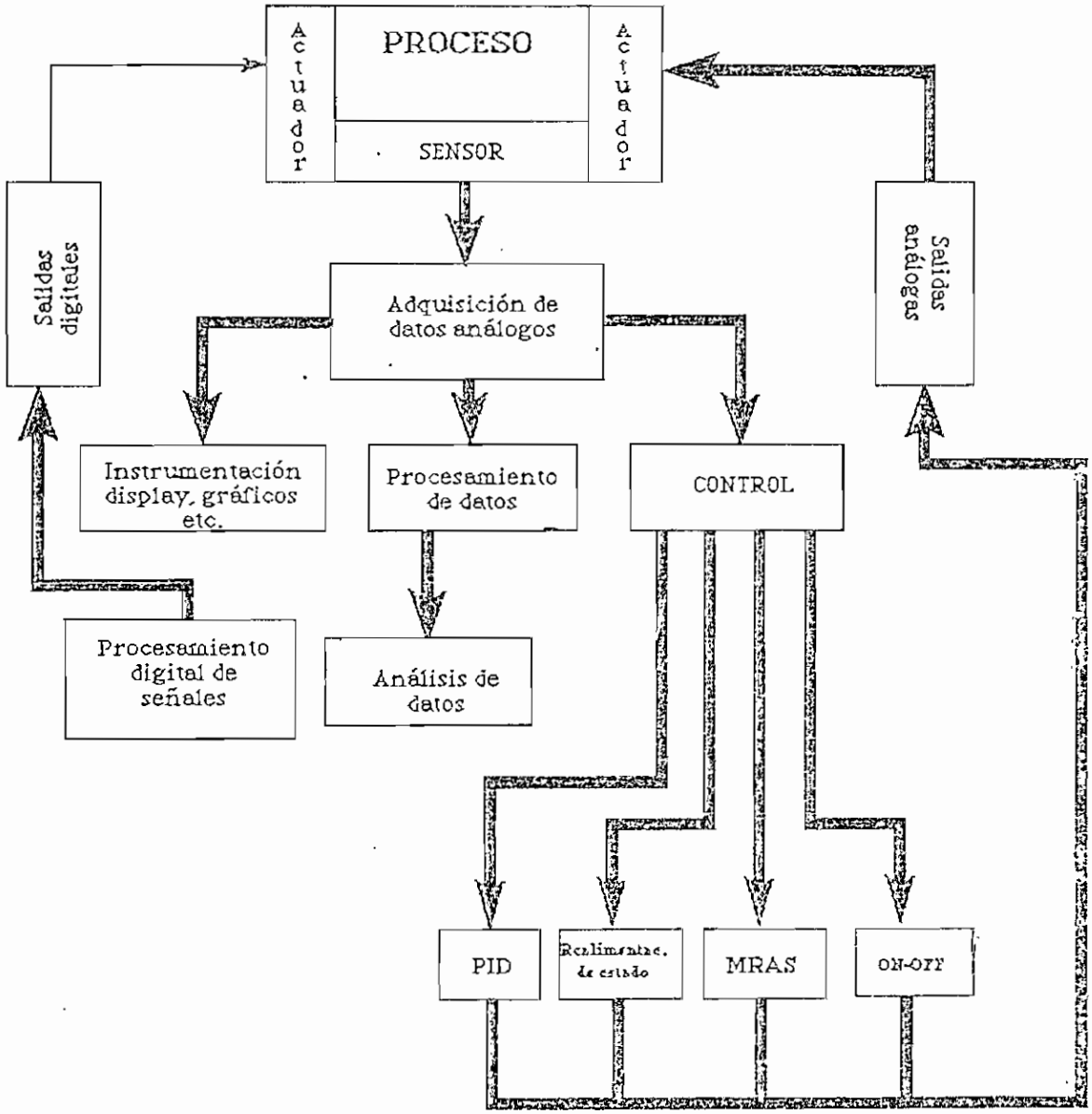


Figura 1.12.- Actividades de la unidad de adquisición y procesamiento de datos y control.

CAPITULO II

2 ADQUISICION Y SALIDA DE DATOS

2.1 ADQUISICION DE DATOS MEDIANTE CANALES ANALOGOS

2.2 SALIDA DE DATOS MEDIANTE CANALES ANALOGOS

2.3 SALIDA DE DATOS MEDIANTE CANALES DIGITALES

2.4 COMANDOS ADICIONALES

2.1.- ADQUISICION DE DATOS MEDIANTE CANALES ANALOGOS

El sistema de adquisición de datos KEITLHEY 500A tiene 8 canales o puertos de entradas análogas ubicados en el módulo AMM1 el cual se describió en el capítulo 1; de estos canales se puede obtener voltajes normalizados entre -10V y +10V, y además están predefinidos como ANLG0, ANLG1, ..., ANLG7, mediante el archivo CONFIG.TBL, realizado con el programa GONFIG.EXE.

La adquisición de datos se puede realizar utilizando dos modos diferentes de trabajo: en FOREGROUND o trabajo dedicado y en BACKGROUND o trabajo de fondo, mediante interrupciones no mascarables, es decir de máxima prioridad. En cada uno de los casos mencionados, se puede hacer adquisición de datos univariable y multivariable; desde estos dos puntos de vista se ha hecho una clasificación que permite enfocar más en detalle el manejo y utilización de la estación KEITHLEY 500A.

De esta manera en cada parte del capítulo dos se tratan cuatro subtemas, en cada uno de los cuales se expone un objetivo o propósito, una explicación de los comandos específicos, y se incluye una rutina tanto en diagrama de flujo como en su listado realizado en QUICK500 en el medio ambiente del QuickBASIC 4.5, al final del capítulo se incluye una explicación de los comandos adicionales que se haya utilizado en el desarrollo de estas rutinas.

Los ejemplos desarrollados para adquisición de datos análogos requieren de una fuente de datos exterior, como puede ser un circuito, una fuente de voltaje, etc., del mismo modo las rutinas de salidas análogas requieren de un osciloscópio donde visualizar los resultados, puesto que no se presentan los resultados en pantalla dada la alta frecuencia con que salen los datos. Por último para los ejemplos de salidas digitales, la visualización requiere de un circuito sencillo construido a base de diodos de luz.

2.1.1.- ADQUISICION UNICANAL DE DATOS ANALOGOS EN FOREGROUND.-

En este numeral se tiene como objetivo realizar adquisición de datos en tiempo real a través de un solo canal análogo, en foreground. Esta rutina ayuda a que el usuario del sistema de adquisición de datos pierda el temor al término tiempo real. El foreground no da la posibilidad de fijar períodos de tiempo para la adquisición o salida de datos, requerimiento fundamental de un sistema de control. Tampoco se puede se realizar análisis de datos puesto que no se tiene un arreglo de memoria Quick500 donde ir almacenando los valores. Se puede decir que este procedimiento en realidad no tiene una aplicación de tipo práctico, a no ser una simple instrumentación, sin embargo servirá como una herramienta de tipo didáctico.

COMANDOS ESPECIFICOS DEL QUICK500.- En esta sección se necesita únicamente un comando de aplicación específica del Quick500 para la adquisición de datos univariable en foreground. Este comando es:

CALL READVAR

PROPOSITO: Rutina de foreground que muestrea entradas digitales, análogas y de frecuencia. El comando READVAR lee un canal simplemente y coloca el valor censado en una variable BASIC.

FORMATO: CALL READVAR (ion$, iotype%, range%, val, euf%, tm$)

PARAMETROS:

ion$ (string-entrada).- Indica el nombre del canal de entrada que será leido, este nombre debe haber sido creado con anterioridad, por comando o por el CONFIG.

iotype% (integer-entrada).- Identifica el tipo de canal a ser leido, existen tres valores válidos: 1.- análogo, 4.- digital, 8.- frecuencia y pulsos.

range% (integer-entrada).- Solo se utiliza para frecuencia, si el canal no es utilizado para frecuencia, este parámetro tiene el valor de -1.

va! (real-salida).- READVAR envía el valor obtenido en el canal leido al parámetro va! (valor), variable BASIC para fácil acceso del usuario.

euf% (integer-entrada).- Permite al usuario especificar en que tipo de unidades necesita los valores (esto fue explicado en el capítulo 1). Si euf% = 0 se obtiene los valores en, voltios.

tm$ (string-entrada).- Modo de disparo; READVAR solo permite dos posibilidades:

"NT"	No trigger	Ejecución inmediata.
"WST"	Wait singleground trig.	Espera disparo de singlegr.

DIAGRAMA DE FLUJO.- El diagrama de flujo de la adquisición de datos análogos unicanal en foreground se presenta en la figura 2.1.

LISTADO DE LA RUTINA BASICA.- Este es el listado mínimo que se necesita para lograr una adquisición de datos univariable en foreground. Se utiliza el canal análogo ANLG0.

```
CLS
LOCATE 10, 18: PRINT "Dato adquirido ="
LOCATE 12, 18: PRINT "Presione SPACE BAR para terminar la adquisición"

CALL SOFTINIT
CALL INIT
'Instrucciones que inicializan en SOFTWARE y HARDWARE al sistema

ion$ = "ANLG0"                  'Nombre del canal de entrada utilizado
va! = 1                         'Valor inicial del parámetro va!
iotype% = 1                     'Canal análogo
range% = -1                     'No se trata de un canal de frecuencia
euf% = 0                        'Resultado de adquisición en voltios
tm$ = "nt"                      'No espera disparo, ejecución inmediata

'Definición de parámetros
```

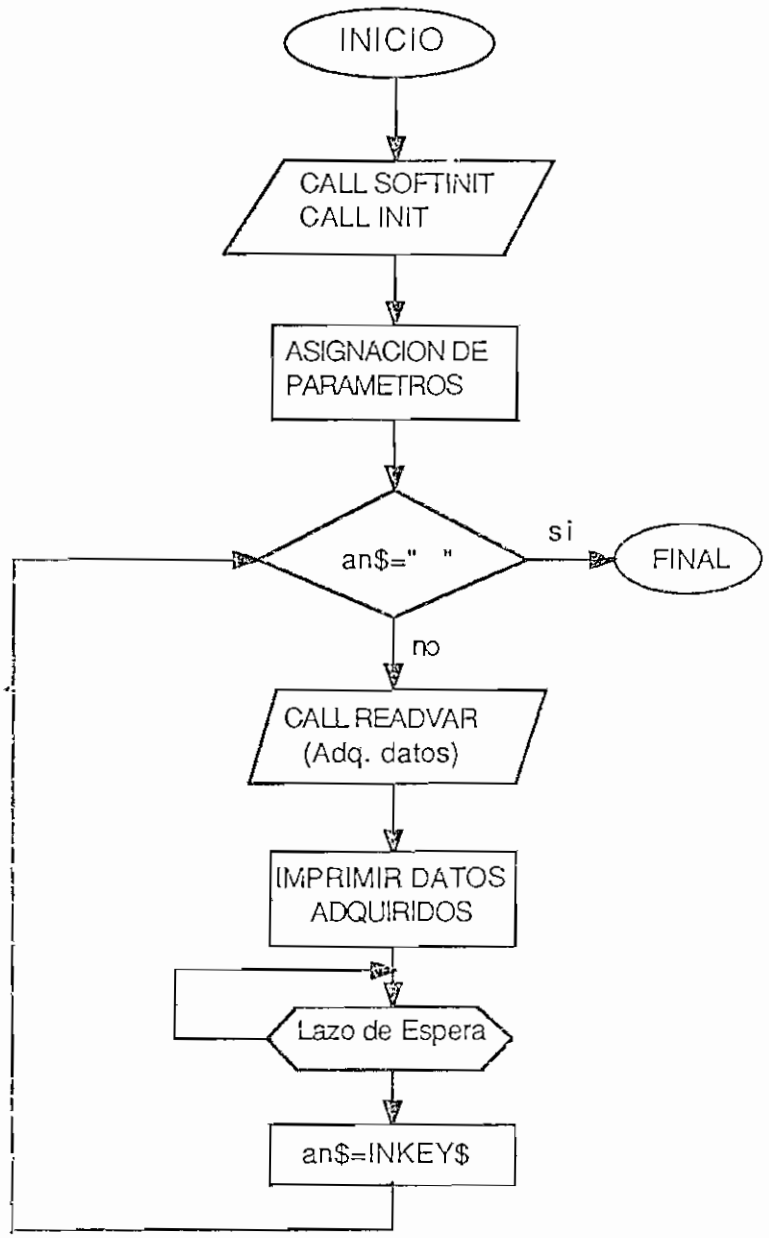


Figura 2.1.- Diagrama de flujo: Adquisición de datos univariable en Foreground

```
COLOR 3
WHILE INKEY$ <> CHR$(32)

      CALL READVAR (ion$, iotype%, range%, va!(), euf%, tm$)
      'Lectura del valor adquirido por el canal análogo 0.

      LOCATE 10, 35: PRINT va!
      FOR i = 1 TO 4000: NEXT
WEND
COLOR
CLS
```

2.1.2.- ADQUISICION MULTICANAL DE DATOS ANALOGOS EN FOREGROUND.-

En esta sección se tiene como objetivo realizar adquisición de datos en tiempo real a través de varios canales análogos de entrada, en foreground. Esta subrutina es súmamente similar a la desarrollada para adquisición de datos univariable.

COMANDOS ESPECIFICOS DEL QUICK500.- Se necesita unicamente un comando de aplicación específica del Quick500 para la adquisición de datos multivariable en foreground. Este comando es:

CALL READARRAY

PROPOSITO: Rutina de foreground que muestrea entradas digitales, análogas y de frecuencia. El comando READARRAY es capaz de leer varios canales de entrada al mismo tiempo, colocando los valores adquiridos en un arreglo BASIC, (vector).

FORMATO: CALL READARRAY (ion$, iotype%, range%, va!, euf%, tm$)

PARAMETROS:

ion1$ (string-entrada).- Indica el nombre de los canales de entrada que serán leidos, este nombre debe haber sido creado con anterioridad, por comando o por el CONFIG.EXE.

iotype% (integer-entrada).- Identifica el tipo de los canales a ser leidos, existen tres valores válidos: 1.- análogo, 4.- digital, 8.- frecuencia y pulsos.

range% (integer-entrada).- Solo se utiliza para frecuencia, si los canales no son utilizado para frecuencia, este parámetro tiene el valor de -1.

val() (real-salida).- READARRAY envía el valor obtenido en los canales leidos al parámetro val() (valor), arreglo BASIC para fácil acceso del usuario.

euf% (integer-entrada).- Permite al usuario especificar en que tipo de unidades necesita los valores. Si euf% = 0 se obtiene los valores en voltios.

tm$ (string-entrada).- Modo de disparo; READARRAY solo permite dos posibilidades:

"NT"	No trigger	Ejecución inmediata.
"WST"	Wait singleground trig.	Espera disparo de singlegr.

DIAGRAMA DE FLUJO.- El diagrama de flujo de la adquisición de datos análogos multicanal en foreground se presenta en la figura 2.2.

LISTADO DE LA RUTINA BASICA.- Este es el listado mínimo que se necesita para lograr una adquisición de datos multivariable en foreground, permite incluso el muestreo de los ocho canales de entrada de la estación.

```
CLS

FOR CAN% = 0 TO 1
      LOCATE 10 + CAN%, 15: PRINT "Canal"; CAN%; "Voltaje ="
NEXT

COLOR 9
LOCATE 20, 6: PRINT "Presione SPACE BAR para terminar la adquisición"
COLOR 7

CALL SOFTINIT
CALL INIT

INPUT "Cuantos canales desea muestrear";c
'Instrucciones que inicializan en SOFTWARE y HARDWARE al sistema
```

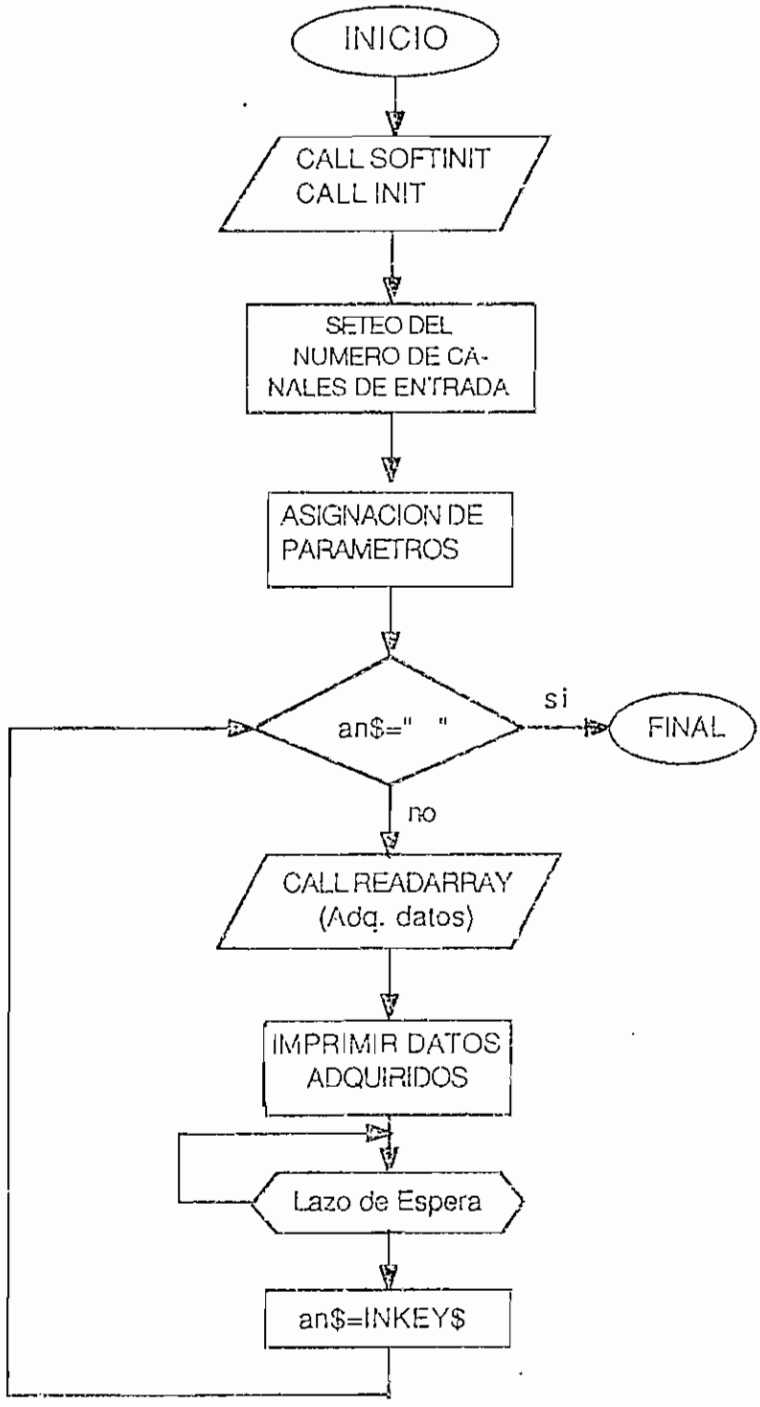


Figura 2.2.- Diagrama de flujo: Adquisición de datos multivariable en Foreground

```
SELECT CASE c
CASE c=1
      ionl$ = "ANLG0"
CASE c=2
      ionl$ = "ANLG0 ANLG1"
CASE c=3
      ionl$ = "ANLG0 ANLG1 ANLG2"
CASE c=4
      ionl$ = "ANLG0 ANLG1 ANLG2 ANLG3"
CASE c=5
      ionl$ = "ANLG0 ANLG1 ANLG2 ANLG3 ANLG4"
CASE c=6
      ionl$ = "ANLG0 ANLG1 ANLG2 ANLG3 ANLG4 ANLG5"
CASE c=7
      ionl$ = "ANLG0 ANLG1 ANLG2 ANLG3 ANLG4 ANLG5 ANLG6"
CASE c=8
      ionl$ = "ANLG0 ANLG1 ANLG2 ANLG3 ANLG4 ANLG5 ANLG6 ANLG7"
END SELECT
DIM va!(2)                  'Dimensionamiento del arreglo BASIC
iotype% = 1                 'Canales análogos
range% = -1                 'No se trata de canales de frecuencia
euf% = 0                          'Resultado de adquisición en voltios
tm$ = "nt"                  'No espera disparo, ejecución inmediata

'Definición de parámetros

COLOR 3
DO

      CALL READARRAY (ionl$, iotype%, range%, va!(), euf%, tm$)
      'Lectura de los datos adquiridos por los respectivos canales

      FOR CAN% = 1 TO 0
            LOCATE 10 + CAN%, 35: PRINT USING "##.## Volts";
            va!(CAN%)
      NEXT
      FOR i = 1 TO 4000: NEXT
LOOP UNTIL INKEY$ = " "
COLOR 7
CLS
```

2.1.3.- ADQUISICION UNICANAL DE DATOS ANALOGOS EN BACKGROUND.-

En esta sección se tiene como objetivo realizar adquisición de datos en tiempo real a través de un solo canal análogo, en background, sistema que utiliza interrupciones no mascarables. Es una rutina sencilla pero de gran utilidad, y será usada en la mayoría de aplicaciones que tengan ingreso de datos univariable.

Debido a que se trata de una rutina que realiza trabajo de fondo mediante interrupciones, permite al tiempo que se realiza la adquisición de datos realizar

cualquier otra actividad en foreground, como son impresiones, gráficos, análisis de datos, y aplicarlos al control en tiempo real.

COMANDOS ESPECIFICOS DEL QUICK500.- En esta sección se hace una descripción de los comandos que se necesita para la adquisición de datos en background, así como de los comandos que habilitan y deshabilitan las interrupciones de background.

CALL ANIN

PROPOSITO: Rutina que realiza el muestreo de hasta 64 canales de entrada análogos. El comando ANIN toma medidas a intervalos constantes de tiempo de un número determinado de canales y crea un área del Quick500, en la que se guardan los valores adquiridos. Puede ejecutarse por si solo o ser disparado por otro comando del Quiuck500.

FORMATO: CALL ANIN (arn$, sn!, ionl$, bintv%, cy%, tn$, bfn$)

PARAMETROS:

arn$ (string-entrada).- Nombre del arreglo creado debido a la ejecución del comando ANIN, estos arreglos son accesibles a través de los comandos de manejo de áreas solamente.

sn! (real-entrada).- indica la profundidad o longitud del arreglo creado, en el caso univariable es un vector, en el caso multivariable es una matriz.

ionl$ (strin-entrada).- Indica los nombres de los canales de entrada que serán muestreados por el comando ANIN. Los nombres de cada canal se separan por espacios y/o por comas. El número de canales utilizado especifica el ancho del arreglo de memoria creado.

bintv% (integer-entrada).- Especifica cada que número de períodos de interrupción se realiza un muestreo de los canales de entrada habilitados. (mayor explicación en el capítulo 3)..

cy% (integer-entrada).- Especifica el número de veces que se debe repetir esta tarea de background, si se asigna cy% = -1, esta tarea se ejecuta de manera indefinida.

tm$ (string-entrada).- Modo de disparo, ANIN soporta los siguientes modos de disparo:

"NT" No trigger	Ejecución inmediata.
"WBT" Wait background trig.	Espera disparo de background
"WGO" Wait for GONOW	Espera disparo de GONOW
"BT" Background trigger	Actúa como disparador de backg

bfn$ (string-entrada).- sirve para nombrar las tareas ejecutadas, si no se utiliza se debe setear un string nulo "".

CALL ARGETVALF/ARGETVALI

PROPOSITO: Permite accesar a un simple valor guardado en un arreglo Quick500, especificando su ubicación en profundidad y anchura, y expresarlo como una variable BASIC. ARGETVALF y ARGETVALI ejecutan funciones similares. La única diferencia es que ARGETVALI enlista solo valores binarios en una variable entera, en tanto que ARGETVALF guarda este valor en una variable real y puede espresarse en el tipo de unidades más conveniente.

FORMATO: CALL ARGETVALF (arn$, dep!, wid%, ion$, va!, euf%)

CALL ARGETVALI (arn$, dep!, wid%, ion$, va%)

PARAMETROS:

arn$ (string-entrada).- Identifica el área desde la que se toma datos, este arreglo es creado anteriormente, mediante los comandos ARMAKE o ANIN.

dep! (real-entrada).- Indice de la profundidad del dato dentro del arreglo. El máximo valor permisible es la longitud del arreglo, esta longitud (en su valor máximo) depende de la cantidad de memoria reservada para el almacenamiento de datos al instalar el programa (64K, para la configuración existente).

wid% (integer-entrada).- Indica la ubicación del dato buscado respecto de la anchura del arreglo, cuando se utiliza ion$ no se utiliza wid%, en cuyo caso se asigna el valor de -1.

ion$ (string-entrada).- Se utiliza con el mismo propósito que el parámetro wid%, indicando el nombre de un canal de entrada. Si se va a utilizar wid%, ion$ debe setearse como un strig nulo "".

va! (real-salida).- Variable BASIC real en la que se guarda el valor tomado del arreglo Quick500, en cualquier tipo de unidades, si se utiliza ARGETVALI, va% es una variable entera, y el valor almacenado debe ser binario.

euf% (integer-entrada).- Especifica el tipo de unidades en que se almacenará el valor en la variable BASIC, el 0 asigna voltios.

CALL INTON

PROPOSITO: Habilita las interrupciones, a la vez que setea la frecuencia de las mismas de acuerdo a un valor especificado por el usuario. Las tareas de backgroud que generalmente son seteadas con anterioridad, no se ejecutan hasta que el comando INTON sea ejecutado.

FORMATO: CALL INTON (ir%, tu$)

PARAMETROS:

ir% (integer-entrada).- Parámetro que especifica el período de interrupción. Tiene ciertos valores válidos de acuerdo a las unidades de tiempo escogidas.

tu$ (string-entrada).- Setea las unidades de tiempo y existen cuatro cadanas de caracteres válidos:

"HMIC"	cientos de microsegundos	ir% = 1-32767
"MIL"	milisegundos	ir% = 1-32767
"SEC"	segundos	ir% = 1-4400
"MIN"	minutos	ir% = 1-74

CALL INTOFF

PROPOSITO: Deshabilita las interrupciones que fueron habilitadas por el comando INTON. Este comando no borra o limpia las tareas de background, es decir que si existe un nuevo INTON se volverán a ejecutar estas tareas, aunque se puede perder el sincronismo, ya que todos los timers (en software y hardware) son deshabilitados.

FORMATO: CALL INTOFF

DIAGRAMA DE FLUJO.- El diagrama de flujo de la adquisición de datos análogos unicanal en background se presenta en la figura 2.3.

LISTADO DE LA RUTINA BASICA.- Esta rutina, con pequeñas adecuaciones es una de las más utilizadas, puesto que para realizar control se debe trabajar

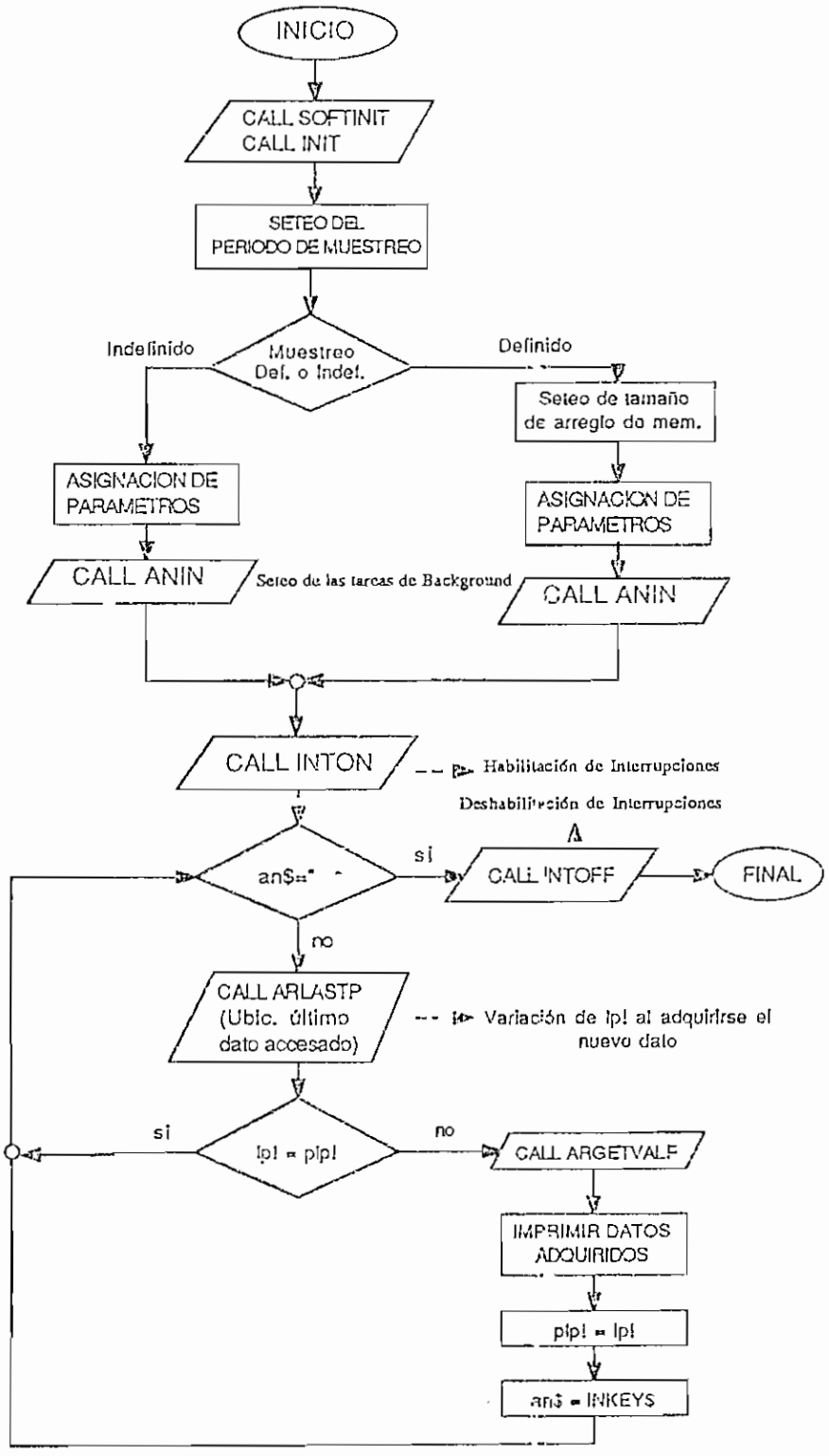


Figura 2.3.a.- Diagrama de flujo: Adquisición de datos univariable en Background

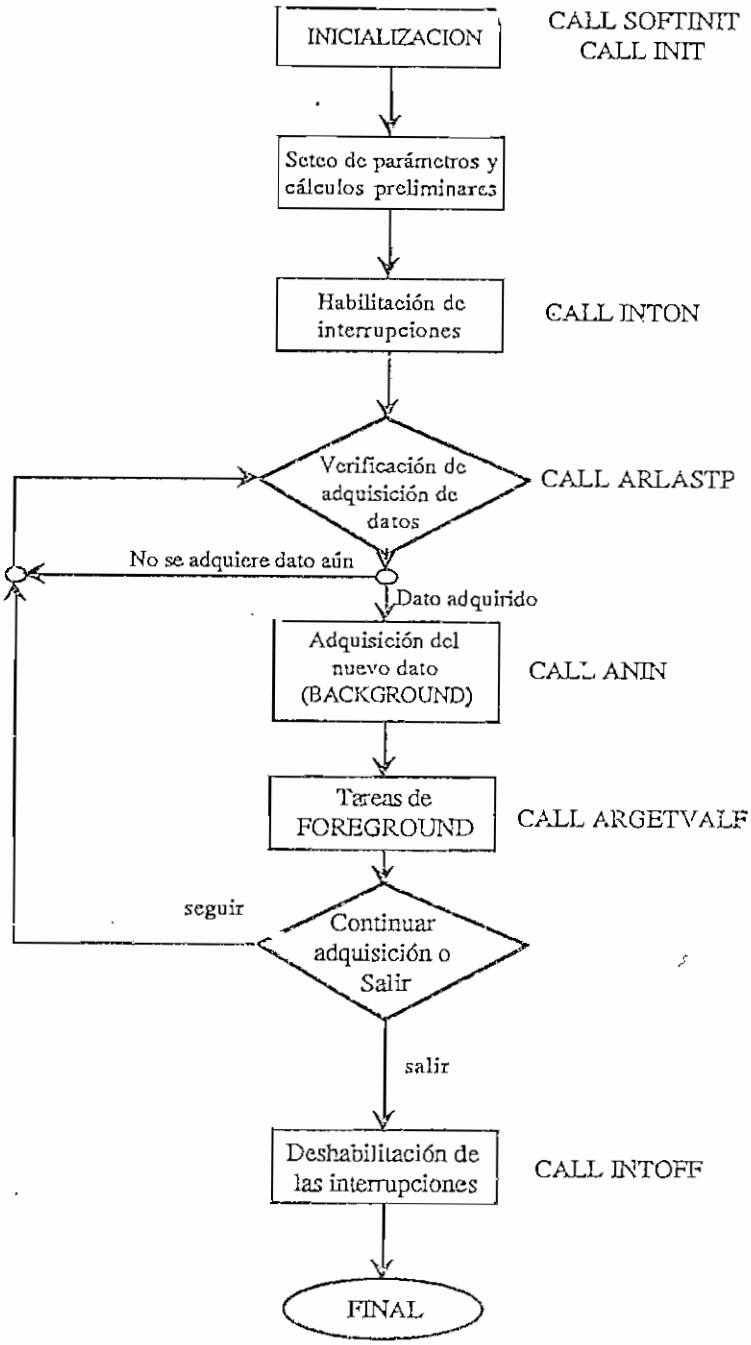


Figura 2.3.b.- Diagrama secuencial de tareas: Adquisición de datos univariable en Background

necesariamente con interrupciones. Para el presente ejemplos se utiliza el canal análogo ANLG0.

```
CLS
COLOR 6
LOCATE 5, 3: PRINT "Las opciones válidas son:"
LOCATE 7, 3: PRINT "Centenas de microsegundos    (hmic)"
LOCATE 8, 3: PRINT "milisegundos                 (mil)"
LOCATE 9, 3: PRINT "segundos                     (sec)"
LOCATE 10, 3: PRINT "minutos                     (min)"
LOCATE 1, 3: INPUT "Período de muestreo (unidades de tiempo)= "; tu$
LOCATE 12, 3: INPUT "Período de muestreo ( # de u. de tiempo)= "; bintv%
LOCATE 3, 3: PRINT "Inicio de la Adquisición de Datos"
CLS
LOCATE 24, 15: PRINT "presione SPACE BAR para terminar la adquisición"

COLOR 9

CALL SOFTINIT
CALL INIT

'Instrucciones que inicializan en SOFTWARE y HARDWARE al sistema

arn$ = "entrada%"
dep! = 20!
ion$ = "ANLG0"
cy% = -1
tm$ = "nt"
bfn$ = "tarea1"

'Seteo de los parámetros de la tarea

CALL ANIN(arn$, dep!, ion$, BINTV%, cy%, tm$, bfn$)

'Comando que setea la adquisición de datos por el canal análogo 0

CALL INTON(1, tu$)

'Habilita las interrupciones

an$ = ""
WHILE an$ = ""
        arn$ = "entrada%"

        CALL arlastp("entrada%", lp!)
        'Asegura que se opera con el último dato adquirido

        IF lp! <> plp! THEN

                CALL argetvalf("entrada%", lp!, -1, "ANLG0", va!, 0)

                'Obtiene un dato del arregló de datos adquiridos

                LOCATE 2 + lp!, 16: PRINT "medida # "; lp!; "     "
                LOCATE 2 + lp!, 40: PRINT va!; "     "
```

```
            plp! = lp!
        END IF
        an$ = INKEY$
WEND

CALL INTOFF

'Deshabilita las interrupciones
COLOR 3
```

2.1.4.- ADQUISICION MULTICANAL DE DATOS ANALOGOS EN BACKGROUND.-

En esta sección se tiene como objetivo realizar adquisición de datos en tiempo real a través de varios canales análogos, en background. Esta rutina tiene las mismas aplicaciones que la anterior, para el caso multivariable.

COMANDOS ESPECIFICOS DEL QUICK500.- Los comandos utilizados en la adquisición de datos de background son prácticamente los mismos para unicanal y multicanal, sin embargo se aprovecha para explicar comandos alternativos.

CALL ARGETF/ARGETI

PROPOSITO: Permite transferir bloques de datos de un arreglo Quick500 a un arreglo BASIC. La única diferencia entre ARGETF y ARGETI, es que el primero solo maneja valores binarios, en tanto que el segundo maneja cualquier unidad permitida por el sistema.

FORMATO: ARGETF (arn$, dep1!, dep2!, wid%, ion$, barn!(), euf%)
ARGETI (arn$, dep1!, dep2!, wid%, ion$, barn%())

PARAMETROS:

arn$ (string-entrada).- Identifica el área desde la que se toma datos, este arreglo es creado anteriormente, mediante los comandos ARMAKE o ANIN.

dep1! dep2!.- (real-entrada).- Delimitan un segmento de un arreglo Quick500. ARGETF/ARGETI recupera los valores de dicho segmento de arreglo a un arreglo BASIC. Dep1! es el límite inferior, y dep2! es el límite superior.

wid% (integer-entrada).- Indica la ubicación del dato buscado respecto de la anchura del arreglo, cuando se utiliza ion$ no se utiliza wid%, en cuyo caso se asigna el valor de -1.

ion$ (string-entrada).- Se utiliza con el mismo propósito que el parámetro wid%, indicando el nombre de un canal de entrada. Si se va a utilizar wid%, ion$ debe setearse como un strig nulo "".

barn!() (real-salida).- Al ejecutar ARGETF/ARGETI un bloque de valores especificado se transfiere de un arreglo Quick500 a un arreglo BASIC.

euf% (integer-entrada).- Especifica el tipo de unidades en que se almacenará el valor en arreglo BASIC, el 0 asigna voltios.

DIAGRAMA DE FLUJO.- El diagrama de flujo de la adquisición de datos análogos multicanal en background se presenta en la figura 2.4.

LISTADO DE LA RUTINA BASICA.- Se ha estructurado un ejemplo para ocho canales de entrada, pero por efectos de utilización de la pantalla solo se muestran dos en la misma.

```
CLS

CALL SOFTINIT
CALL INIT

'Instrucciones que inicializan en SOFTWARE y HARDWARE al sistema

COLOR 2
pregua:
LOCATE 1, 3: PRINT "Adquisición de datos a través de 8 canales análogos"
LOCATE 5, 3: PRINT "indicaciones: cientos de microsegundos:  hmic"
LOCATE 6, 3: PRINT "              milisegundos            : mil"
LOCATE 7, 3: PRINT "              segundos                : sec"
LOCATE 8, 3: PRINT "              minutos                 : min"
LOCATE 3, 3: INPUT "Periodo de muestreo (UNIDADES DE TIEMPO)= "; tu$
LOCATE 10, 3: PRINT "Período de muestreo ("; tu$; : INPUT ") "; bintv%
```

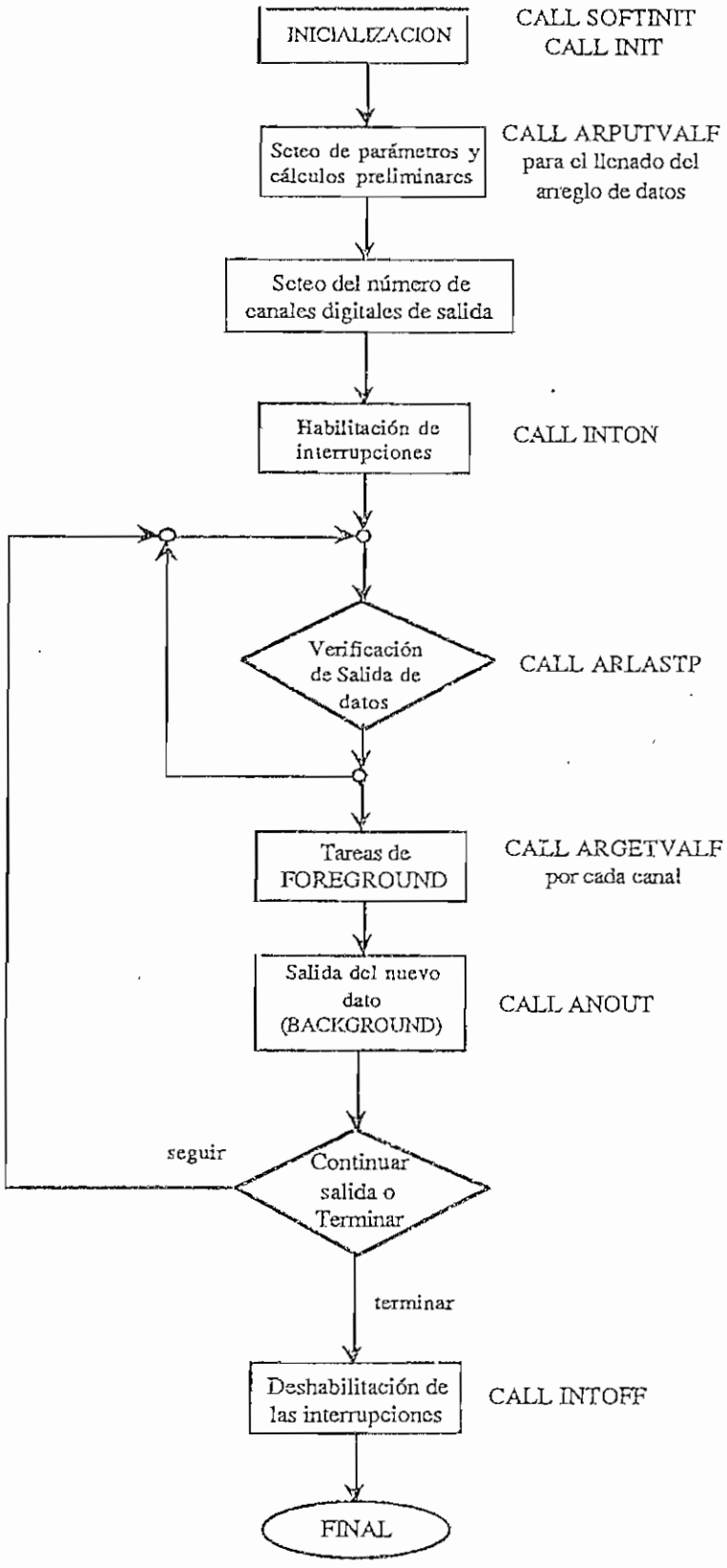


Figura 2.4.a.- Diagrama de flujo: Adquisición de datos multivariable en Background

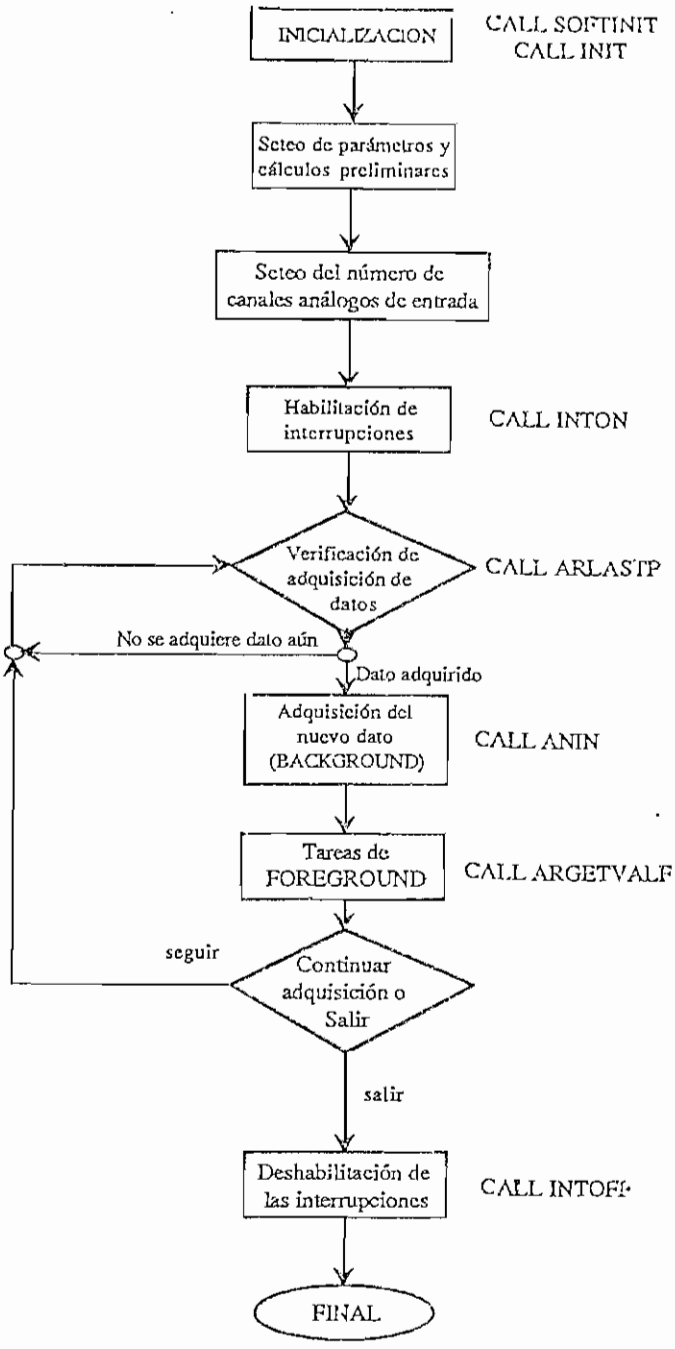


Figura 2.3.b.- Diagrama secuencial de tareas: Adquisición de datos multivariable en Background

```
IF tu$ = "hmic" THEN: GOTO sigaa
ELSEIF tu$ = "mil" THEN: GOTO sigaa
ELSEIF tu$ = "sec" THEN: GOTO sigaa
ELSEIF tu$ = "min" THEN: GOTO sigaa
END IF
GOTO pregua
sigaa:
dep! = 20: l! = -(dep!)
LOCATE 12, 3: INPUT "Cuantos canales desea muestrear (máximo 8)"; n
SELECT CASE n:
     CASE 1: ionl$ = "ANLG0"
     CASE 2: ionl$ = "ANLG0 ANLG1"
     CASE 3: ionl$ = "ANLG0 ANLG1 ANLG2"
     CASE 4: ionl$ = "ANLG0 ANLG1 ANLG2 ANLG3"
     CASE 5: ionl$ = "ANLG0 ANLG1 ANLG2 ANLG3 ANLG4"
     CASE 6: ionl$ = "ANLG0 ANLG1 ANLG2 ANLG3 ANLG4 ANLG5"
     CASE 7: ionl$ = "ANLG0 ANLG1 ANLG2 ANLG3 ANLG4 ANLG5 ANLG6"
     CASE 8: ionl$ = "ANLG0 ANLG1 ANLG2 ANLG3 ANLG4 ANLG5 ANLG6 ANLG7"
END SELECT
LOCATE 15, 3: PRINT "Que canales (2) desea mostrar en pantalla"
LOCATE 17, 3: PRINT "Primer canal mostrado (0-"; n - 1; ")"; : INPUT c1
LOCATE 18, 3: PRINT "Segundo canal mostrado (0-"; n - 1; ")"; : INPUT c2
SELECT CASE c1
     CASE 0: ion$ = "ANLG0": CASE 1: ion$ = "ANLG1"
     CASE 2: ion$ = "ANLG2": CASE 3: ion$ = "ANLG3"
     CASE 4: ion$ = "ANLG4": CASE 5: ion$ = "ANLG5"
     CASE 6: ion$ = "ANLG6": CASE 7: ion$ = "ANLG7"
END SELECT
SELECT CASE c2
     CASE 0: iona$ = "ANLG0": CASE 1: iona$ = "ANLG1"
     CASE 2: iona$ = "ANLG2": CASE 3: iona$ = "ANLG3"
     CASE 4: iona$ = "ANLG4": CASE 5: iona$ = "ANLG5"
     CASE 6: iona$ = "ANLG6": CASE 7: iona$ = "ANLG7"
END SELECT
DIM barn0!(20)
DIM barn1!(20)
CLS
LOCATE 1, 16: PRINT "ENTRADAS:CANAL ANALG"; c1; "CANAL ANALG"; c2
LOCATE 2, 14: PRINT"----------------------------------------"
LOCATE 23, 14: PRINT "presione SPACE BAR para terminar la adquisición de datos"
COLOR 7: lp! = 1

CALL ANIN("entrada%", dep!, ionl$, bintv%, -1, "nt", "tarea1")

'comando que setea entrada de datos a través de canales análogos como una tarea de Background

CALL INTON(1, tu$)        'comando que habilita las interrupciones de BACKGROUND

an$ = ""
WHILE an$ = ""
     arn$ = "entrada%"
      CALL arlastp(arn$, lp!)

      'comando que indica la profundidad del último dato accesado

      IF lp! <> plp! THEN
```

```
            CALL ARGETF("entrada%", 1!, dep!, -1, ion$, barn0!(), 0)
            CALL ARGETF("entrada%", 1!, dep!, -1, iona$, barn1!(), 0)

           'comandos que recuperan el dato del arreglo de memoria

           IF lp! = 1 THEN l! = l! + dep!
            LOCATE 2 + lp!, 16: PRINT "medida#"; lp! + l!; "    "
            LOCATE 2 + lp!, 40: PRINT barn0!(lp!), barn1!(lp!); "     "
           plp! = lp!
      END IF
      an$ = INKEY$
WEND

CALL INTOFF        'comando que deshabilita las interrupciones del BACKGROUND
```

2.2.- SALIDA DE DATOS MEDIANTE CANALES ANALOGOS.-

El sistema de adquisición de datos KEITLHEY 500A consta de 5 canales de salida análoga, los cuales están ubicados en el módulo AOM1 descrito en el capítulo 1, estos canales son capaces de entregar voltajes normalizados de -10V a +10V, estos canales han sido predefinidos como ANOUT0, ANOUT1, ..., ANOUT4, mediante el archivo CONFIG.TBL hecho mediante el programa CONFIG.EXE.

Al igual que las entradas análogas, se puede obtener salidas análogas con los modos de trabajo FOREGROUND y BACKGROUND, tanto unicanal como multicanal. Entonces siguiendo el mismo formato de trabajo desarrollado, se explicará la salida de datos en cada modo de trabajo, tanto en unicanal como en multicanal.

2.2.1.- SALIDA UNICANAL DE DATOS ANALOGOS EN FOREGROUND.-

Tal como sucediera con las rutinas de entradas análogas, esta rutina de salida análoga en foreground, es una rutina que se ha desarrollado más bien con fines didácticos, pues su utilización no da la posibilidad de fijar períodos de tiempo para la adquisición o salida de datos, requerimiento fundamental de un sistema de control,

tampoco se puede se realizar análisis de datos puesto que no se tiene un arreglo de memoria Quick500 donde almacer los valores.

COMANDOS ESPECIFICOS DEL QUICK500.- La salida de datos a través de un canal análogo en foreground solo requiere de un comando específico del Quick500, el mismo que se explica a continuación:

CALL WRITEVAR

PROPOSITO: Comando que escribe un valor (tomado de una variable BASIC de fácil acceso) especificado anteriormente en un canal de salida. Este canal puede ser análogo, digital, de frecuencia o de pulsos. Si se trata de un canal análogo, se puede sacar los valores en binario o en unidades de ingeniería.

FORMATO: WRITEVAR (ion$, iotype%, va!, euf%, tm$)

PARAMETROS:

ion$ (string-entrada).- Indica el nombre del canal de salida, donde se escribirá un valor, este nombre debe haber sido creado con anterioridad, por comando o por el CONFIG.

iotype% (integer-entrada).- Identifica el tipo de canal de salida, existen tres valores válidos: 1.- análogo, 4.- digital, 8.- frecuencia y pulsos.

range% (integer-entrada).- Solo se utiliza para frecuencia, si el canal no es utilizado para frecuencia, este parámetro tiene el valor de -1.

va! (real-entrada).- WRITEVAR envía el valor obtenido por va! (valor), variable BASIC para fácil acceso del usuario, al canal de salida.

euf% (integer-entrada).- Permite al usuario especificar en que tipo de unidades necesita los valores. Si euf% = 0 se obtiene los valores en voltios.

tm$ (string-entrada).- Modo de disparo; WRITEVAR solo permite dos posibilidades:

"NT"	No trigger	Ejecución inmediata.
"WST"	Wait singleground trig.	Espera disparo de singlegr.

DIAGRAMA DE FLUJO.- El diagrama de flujo de la salida de datos análogos unicanal en foreground se presenta en la figura 2.5.

LISTADO DE LA RUTINA BASICA.- Este listado es el mínimo que se tiene para realizar la salida de datos análogos en foreground, no existe ninguna tarea adicional salvo la impresión en pantalla de los valores sacados por el canal escogido. Los resultados obtenidos a través del canal análogo de salida ANOUT0 se deben visualizar en un osciloscopio.

```
CLS
LOCATE 1, 3: PRINT "Valor de Salida ="
LOCATE 22, 16: PRINT "presione SPACE BAR para terminar salida de datos"

CALL SOFTINIT
CALL INIT

'Instrucciones que inicializan en SOFTWARE y HARDWARE al sistema

ion$ = "ANOUT0"
va! = 0
iotype% = 2
euf% = 0

tm$ = "nt"

'Definicion de parametros
COLOR 0
WHILE INKEY$ <> CHR$(32): v = v + 1
      va! = va! + 1
      IF va! > 10 THEN va! = 0
      IF v > 16 THEN v = 1
      LOCATE 1, 22 + va! * 2: PRINT va!; SPACE$(200)

      CALL WRITEVAR(ion$, iotype%, va!, euf%, tm$)

      'salida del dato análogo a través del canal 0

      COLOR v
WEND
```

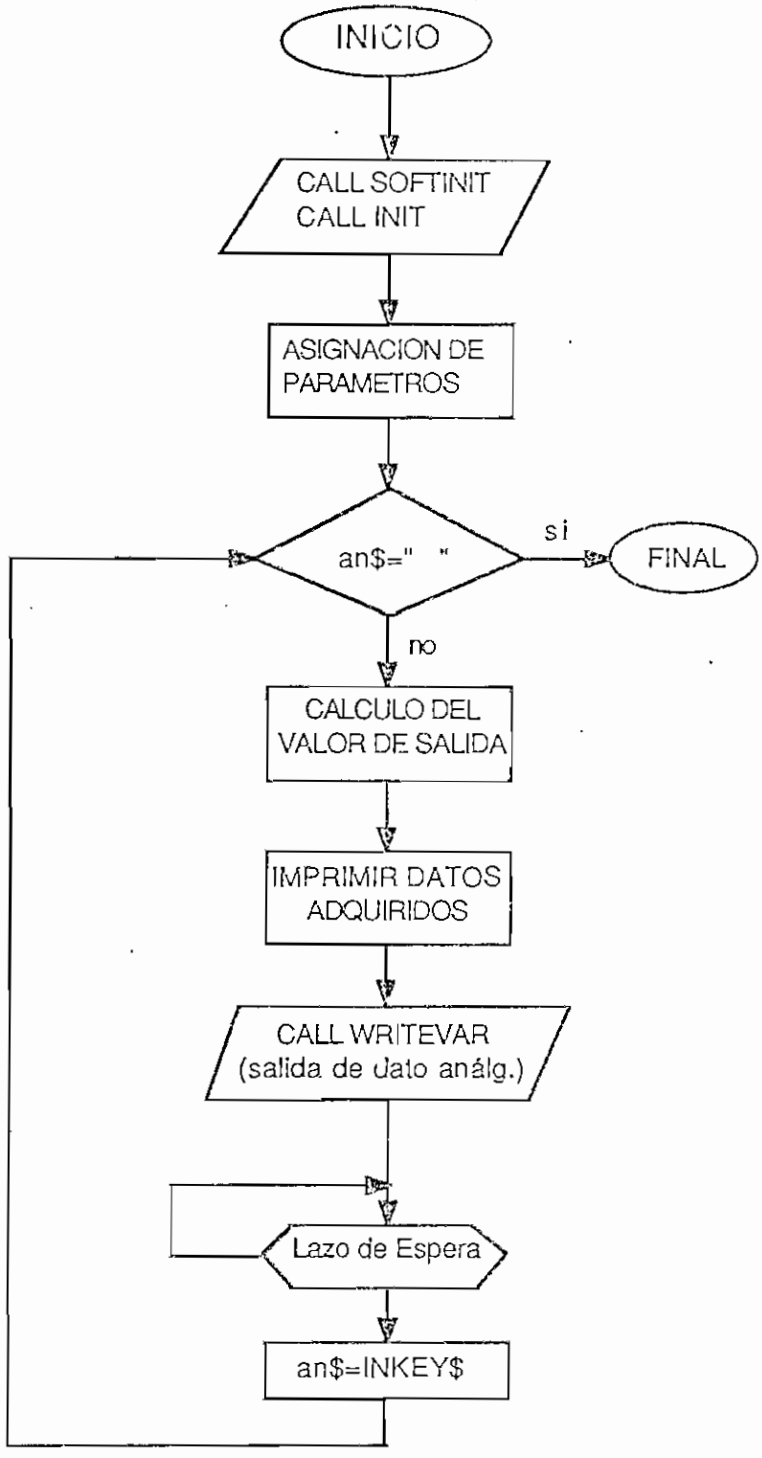


Figura 2.5.- Diagrama de flujo: Salida de datos univariable en Foreground

2.2.2.- SALIDA MULTICANAL DE DATOS ANALOGOS EN FOREGROUND.-

El desarrollo para salida de datos multivariable en foreground es bastante similar a la salida de datos univariable en su estructura, por lo que aquello que se dijo en ese caso es perfectamente aplicable al caso multicanal.

COMANDOS ESPECIFICOS DEL QUICK500.- La salida de datos análogos en el modo de trabajo dedicado requiere de un único comando del Quick500, el mismo que se explica a continnación.

CALL WRITEARRAY

PROPOSITO: Este comando, obtiene múltiples valores de un arreglo BASIC, y los escribe en una lista de canales de salida especificados, estos canales de salida pueden ser análogos, digitales, de frecuencia o de pulsos. Si los canales de salida son análogos, se puede escribir los valores en binario o en unidades de ingeniería.

FORMATO: CALL WRITEARRAY (ion!$, iotype%, va!(), euf%, tm$)

PARAMETROS:

ion!$ (string-entrada).- Indica el nombre de los canales de salida utilizados, este nombre debe haber sido creado con anterioridad, por comando o por el CONFIG.EXE.

iotype% (integer-entrada).- Identifica el tipo de los canales de salida, existen cuatro valores válidos: 1.- análogo, 4.- digital, 8.- frecuencia y pulsos.

range% (integer-entrada).- Solo se utiliza para frecuencia, si los canales no son utilizado para frecuencia, este parámetro tiene el valor de -1.

va!() (real-entrada).- WRITEARRAY envía los valores obtenidos en el parámetro va!() (valor, arreglo BASIC para fácil acceso del usuario) a los canales de salida escogidos.

euf% (integer-entrada).- Permite al usuario especificar en que tipo de unidades necesita los valores (esto fue explicado en el capítulo 1). Si euf% = 0 se obtiene los valores en voltios.

tm$ (string-entrada).- Modo de disparo; WRITEARRAY solo permite dos posibilidades:

"NT"	No trigger	Ejecución inmediata.
"WST"	Wait singleground trig.	Espera disparo de singlegr.

DIAGRAMA DE FLUJO.- El diagrama de flujo de la salida de datos análogos multicanal en foreground se presenta en la figura 2.6.

LISTADO DE LA RUTINA BASICA.- El listado que se escribe a continuación nos permite sacar datos a través de varios canales de salida análoga, e imprimir estos valores en pantalla, este listado mínimo permite la utilización de los cinco canales de salida análoga. Los resultados se deben observar en un osciloscopio.

```
CLS
LOCATE 1, 1: PRINT "Canal de Salida :          1                    2"
LOCATE 2, 1: PRINT "-------------------------------------------------------------"
LOCATE 22, 16: PRINT "presione SPACE BAR para terminar salida de datos"

CALL SOFTINIT
CALL INIT

'Instrucciones que inicializan en SOFTWARE y HARDWARE al sistema

ionl$ = "ANOUT0 ANOUT1"              'Parámetro que especifica los
                                     'canales de salida utilizados.
DIM va!(2)
va!(1) = 0
va!(0) = 0
iotype% = 2
euf% = 0
tm$ = "nt"
'Definicion de parametros

COLOR 0
WHILE INKEY$ <> CHR$(32): v = v + 1
      va!(0) = va!(0) + 1: va!(1) = 11 - va!(0)
```

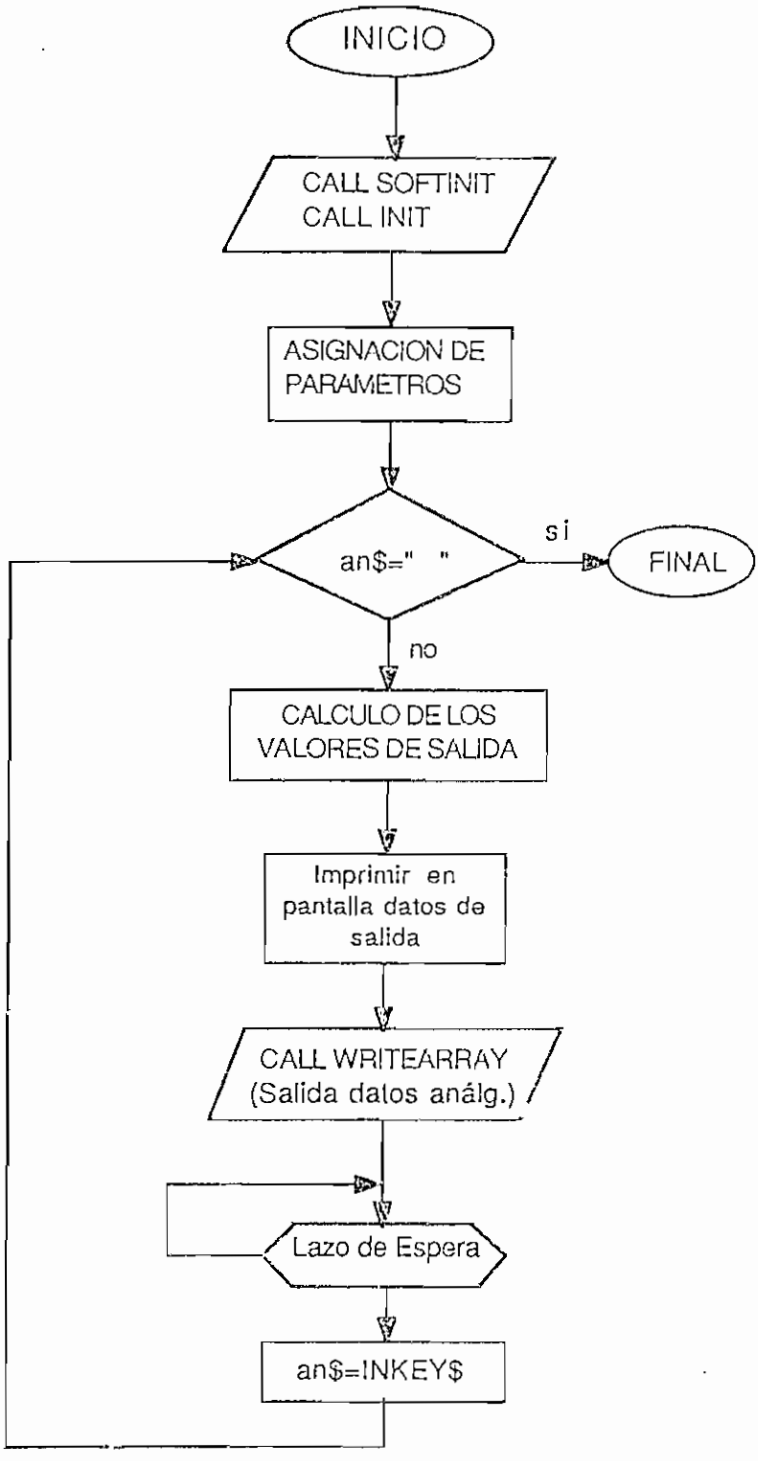


Figura 2.6.- Diagrama de flujo: Salida de datos multivariable en Foreground

```
        IF va!(0) > 10 THEN va!(0) = 0
        IF v > 16 THEN v = 1
        LOCATE 3, 22 + va!(0) * 2: PRINT va!(0); SPACE$(200)
        LOCATE 3, 52 + va!(0) * 2: PRINT va!(1); SPACE$(200)

        CALL WRITEARRAY(ionl$, iotype%, va!(), euf%, tm$)

        'Escritura de datos en los canales de salida respectivos
        COLOR v
WEND
```

2.2.3.- SALIDA UNICANAL DE DATOS ANALOGOS EN BACKGROUND.-

El objetivo de esta sección es sacar datos en tiempo real a través de un solo canal análogo, en background, sistema que utiliza interrupciones no mascarables. Es una rutina sencilla que sin embargo es de gran utilidad, y será incluida en la mayoría de aplicaciones que tengan salida de datos análogos univariable.

Al tratarse de una rutina que realiza trabajo de fondo mediante interrupciones que se presentan a intervalos de tiempo iguales, permite al tiempo que se realiza la salida de datos, realizar cualquier otra actividad en foreground, como son impresiones, gráficos, análisis de datos, algoritmos de control, etc. Todas estas tareas deberán ser terminadas antes de que se presente una nueva interrupción.

COMANDOS ESPECIFICOS DEL QUICK500.- Se hará una descripción de los comandos que se necesita para la salida de datos en background, tanto en el proceso de obtener estos valores de una variable BASIC, como en el proceso de efectivamente sacar estos valores por los canales de salida respectivos.

CALL ARMAKE

PROPOSITO: Crea un arreglo de memoria con el formato Quick500. Dicho arreglo puede ser de cuatro tipos: de bit, byte, palabra o un arreglo real. Todo arreglo tiene

dos dimensiones ancho y profundidad. El ancho se relaciona con el número de canales que pueden ser accesados y la profundidad es el número de valores trabajados por cada canal.

FORMATO: CALL ARMAKE(arn$, dep!, wid%, ionl$)

PARAMETROS:

arn$ (string-entrada).- Asigna un nombre al arreglo creado por ARMAKE, este nombre debe ser de 8 caracteres máximo, más el caracter que indica el tipo de arreglo que se crea. Si no se especifica este caracter, el arreglo creado será real.

dep! (real-entrada).- Define la profundidad o longitud del vector.

wid% (integer-entrada).- Define el ancho del arreglo creado. Cuando se utiliza el parámetro ionl$, el parámetro wid% se setea -1.

ionl$ (string-entrada).- Se utiliza como alternativa del parámetro wid% para definir el ancho del arreglo. Si no se utiliza se setea un string nulo "".

CALL ANOUT

PROPOSITO: Rutina que toma valores de un arreglo previamente creado y lo saca a través de canales de salida análogos. El comando ANOUT saca valores a intervalos constantes de tiempo de un número determinado de canales pero no crea un área del Quick500, en la que se guardan los valores adquiridos, este arreglo debe ser creado de antemano por ARMAKE, ANIN, etc. Puede ejecutarse por si solo o ser disparado por otro comando del Quick500.

FORMATO: CALL ANOUT (arn$, ionl$, bintv%, cy%, tm$, bfn$)

PARAMETROS:

arn$ (string-entrada).- Nombre del arreglo accesado por el comando ANOUT. Nótese que el comando ANOUT no crea el arreglo de memoria, utiliza uno que se debe haber creado anteriormente.

ionl$ (strin-entrada).- Indica los nombres de los canales de salida que serán muestreados por el comando ANOUT. Los nombres de cada canal se separan por espacios y/o por comas. El número de canales utilizado especifica el ancho del arreglo de memoria creado.

bintv% (integer-entrada).- Especifica cada que número de períodos de interrupción se realiza un muestreo de los canales de entrada habilitados. (mayor explicación en el capítulo 3).

cy% (integer-entrada).- Especifica el número de veces que se debe repetir esta tarea de background, si se asigna cy% = -1, esta tarea se ejecuta de manera indefinida.

tm$ (string-entrada).- Modo de disparo, ANOUT soporta los modos de disparo:

"NT" No trigger	Ejecución inmediata.
"WBT" Wait background trig.	Espera disparo de background
"WGO" Wait for GONOW	Espera disparo de GONOW
"BT" Background trigger	Actúa como disparador de backg

bfn$ (string-entrada).- sirve para nombrar las tareas ejecutadas, si no se utiliza se debe setear un string nulo "".

CALL ARPUTVALF/ARPUTVALI

PROPOSITO: Toma valores de una variable BASIC y los guarda en una arreglo Quick500. ARPUTVALF y ARPUTVALI ejecutan funciones similares. La única

diferencia es que ARPUTVALI enlista solo valores binarios en una variable entera, en tanto que ARPUTVALF guarda este valor en una variable real y puede espresarse en el tipo de unidades más conveniente.

FORMATO: CALL ARPUTVALF (arn$, dep!, wid%, ion$, va!, euf%)
CALL ARPUTVALI (arn$, dep!, wid%, ion$, va%)

PARAMETROS:

arn$ (string-entrada).- Identifica el arreglo en el que se almacenan los datos, este arreglo es creado anteriormente, mediante los comandos ARMAKE o ANIN.

dep! (real-entrada).- Indice de la profundidad del dato dentro del arreglo. El máximo valor permisible es la longitud del arreglo.

wid% (integer-entrada).- Indica la ubicación del dato buscado respecto de la anchura del arreglo, cuando se utiliza ion$ no se utiliza wid%, en cuyo caso se asigna el valor de -1.

ion$ (string-entrada).- Se utiliza con el mismo propósito que el parámetro wid%, indicando el nombre de un canal de entrada. Si se va a utilizar wid%, ion$ debe setearse como un strig nulo "".

va! (real-entrada).- Variable BASIC real que contiene el valor que se guarda en el arreglo Quick500, en cualquier tipo de unidades, si se utiliza ARGETVALI, va% es una variable entera, y el valor almacenado debe ser binario.

euf% (integer-entrada).- Especifica el tipo de unidades en que se almacenará el valor tomado de la variable BASIC, el 0 asigna voltios.

DIAGRAMA DE FLUJO.- El diagrama de flujo de la salida de datos análogos unicanal en background se presenta en la figura 2.7.

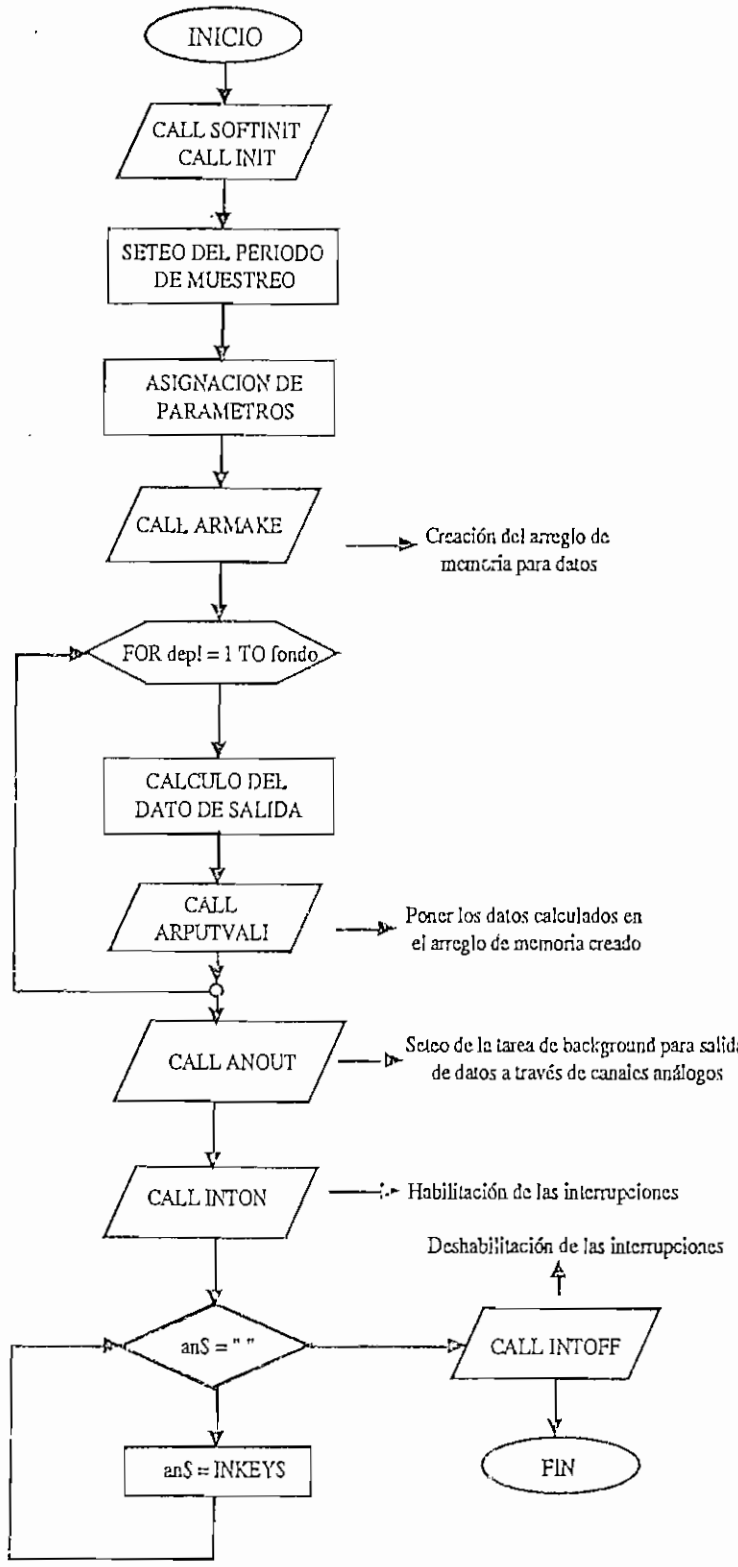


Figura 2.7.a.- Diagrama de flujo: Salida de da os univariable en Background

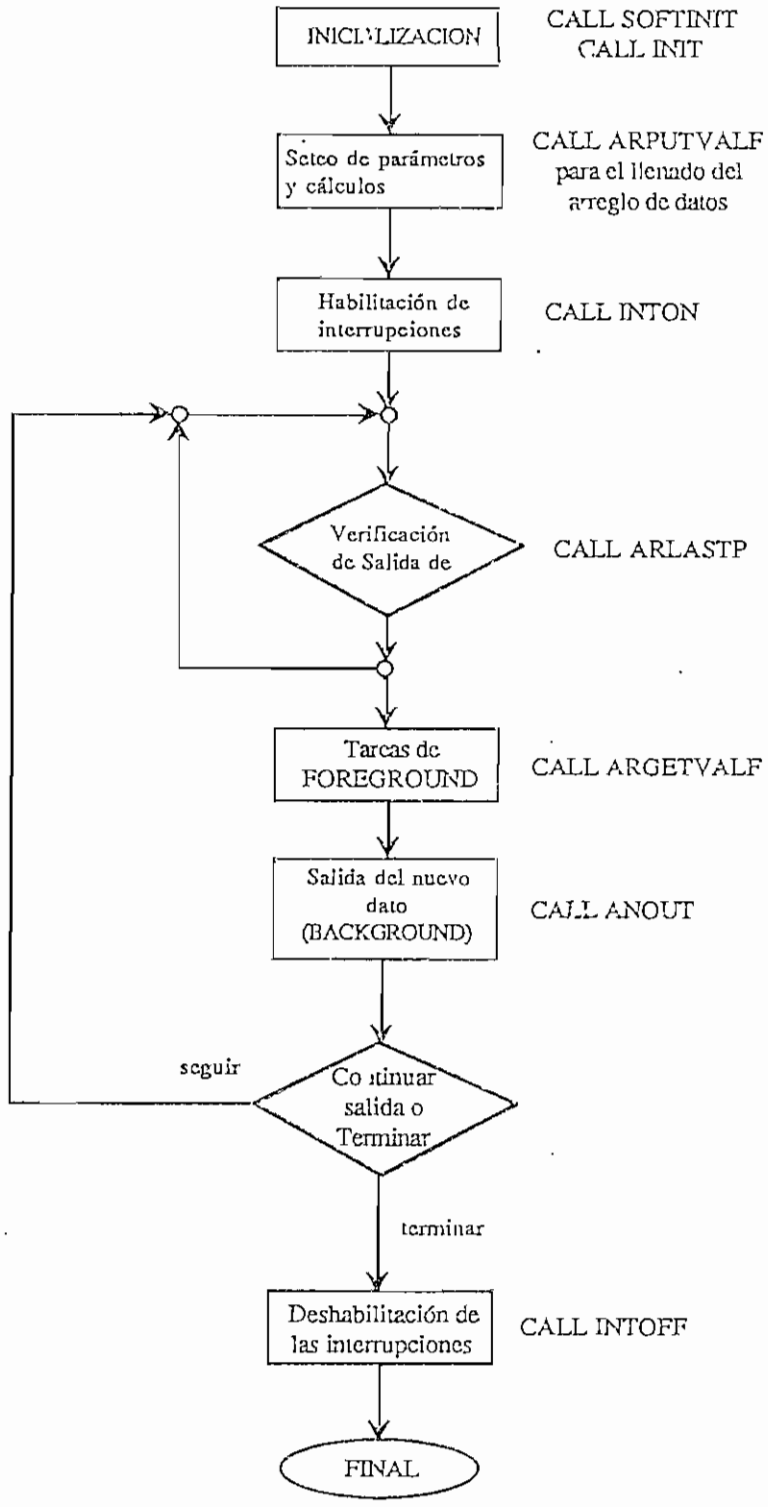


Figura 2.7.a.- Diagrama de flujo: Salida de datos univariable en Background

LISTADO DE LA RUTINA BASICA.- Esta es una de las rutinas básicas de mayor utilidad, puesto que al trabajar basado en interrupciones, permite realizar otras tareas simultáneamente, tales como gráficos, análisis, y sobre todo desarrollar y ejecutar algoritmos de control. Es resultado es una onda senoidal, cuya frecuencia depende del período de muestreo escogido, esta debe ser observada mediante osciloscopio.

```
CLS

CALL SOFTINIT
CALL INIT

'Instrucciones que inicializan en SOFTWARE y HARDWARE al sistema

ion1$ = "ANOUT0"
LOCATE 3, 3: INPUT "Periodo de muestreo (ms)= "; BINTV%
dep! = 100
arn$ = "datosen%"

'Seteo de los parámetros del arreglo a crearse

CALL ARMAKE("datosen%", dep!, -1, ion1$)

'Comando que crea un arreglo de memoria formato Qaick500

FOR dep! = 1 TO 100
      d% = INT(SIN((dep! - 1) * 6.28 / 100) * 2047.5 + 2047.5)
      PRINT d%,

      CALL ARPUTVALI("datosen%", dep!, -1, "ANOUT0", d%)

      'Comando que almacena valores calculados en el arreglo de memoria
NEXT dep!

CALL ANOUT("datosen%", "ANOUT0", BINTV%, -1, "nt", "tarea2")

'Comando que saca los datos una vez que se habiliten las 'interrupciones

LOCATE 22, 22: PRINT "Empieza la salida de datos de una sinusoide de 1Hz."
LOCATE 25, 21: PRINT "Presione la barra espaciadora para terminar la salida"

CALL INTON(1, "mil")

'Comando que habilita las interrupciones del BAKGROUND.

an$ = ""
WHILE an$ = ""
      an$ = INKEY$
WEND
```

```
CALL INTOFF
'Comando que deshabilita las interrupciones
END
```

2.2.4.- SALIDA MULTICANAL DE DATOS ANALOGOS EN BACKGROUND.-

El objetivo de esta sección es sacar datos en tiempo real a través de varios canales análogos, mediante el modo de trabajo de fondo o background, sistema que utiliza interrupciones no mascarables. Esta subrutina de gran utilidad, se incluye en muchas de las aplicaciones posteriores. Todo aquello que se dijo de la salida de datos univariable es perfectamente aplicable a este caso.

COMANDOS ESPECIFICOS DEL QUICK500.- Tanto para el caso unicanal como para el multicanal se utilizan los mismos comandos, pero se aprovecha este numeral para explicar comandos alternativos en la realización de las mismas tareas.

ARPUTF/ARPUTI

PROPOSITO: Permite transferir rápidamente bloques de datos del BASIC a un área de formato Quick500, realizando el trabajo 15 veces más rápido que el BASIC. ARPUTF y ARPUTI realizan funciones similares, con la diferencia de las unidades que pueden manejar; ARPUTF todo tipo de unidades de ingeniería y ARPUTI solo valores binarios.

FORMATO: ARPUTF(arn$, dep1!, dep2!, vid%, ion$, barn!(), euf%)

ARPUTI(arn$, dep1!, dep2!, vid%, ion$, barn%())

PARAMETROS:

arn$ (string-entrada).- Identifica el área de formato Quick500 desde la que se entrega datos, este arreglo es creado anteriormente, mediante los comandos ARMAKE o ANIN.

dep1! dep2!.- (real-entrada).- Delimitan un segmento de un arreglo Quick500. ARPUTF/ARPUTI incorpora los valores a dicho segmento de arreglo desde un arreglo BASIC. Dep1! es el límite inferior, y dep2! es el límite superior.

wid% (integer-entrada).- Indica la ubicación del dato buscado respecto de la anchura del arreglo, cuando se utiliza ion$ no se utiliza wid%, en cuyo caso se asigna el valor de -1.

ion$ (string-entrada).- Se utiliza con el mismo propósito que el parámetro wid%, indicando el nombre de un canal de entrada. Si se va a utilizar wid%, ion$ debe setearse como un string nulo "".

barn!() (área real-entrada).- Al ejecutar ARPUTF/ARPUTI un bloque de valores especificado se transfiere de un arreglo Quick500 a un arreglo BASIC.

euf% (integer-entrada).- Especifica el tipo de unidades en que se almacenará el valor en arreglo BASIC, el 0 asigna voltios.

DIAGRAMA DE FLUJO.- El diagrama de flujo de la salida de datos análogos multicanal en background se presenta en la figura 2.8.

LISTADO DE LA SUBRUTINA BASICA.- Se ha estructurado un ejemplo que utiliza los cinco canales de salida análoga con que se cuenta, se trata de un generador de ondas que en cada canal de salida genera una onda diferente. Los resultados deben ser observados en osciloscopio.

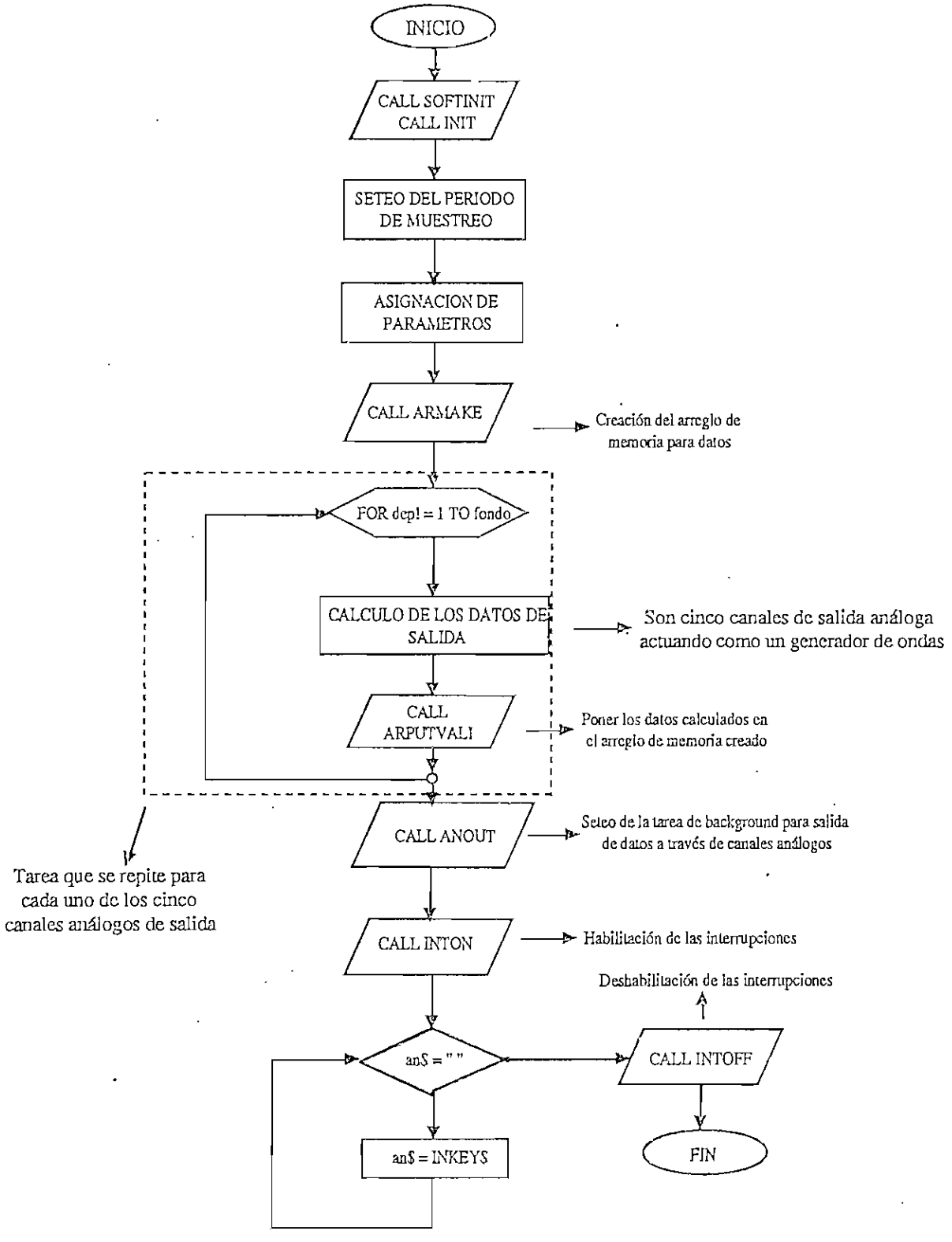


Figura 2.7.a.- Diagrama de flujo: Salida de datos multivariable en Background

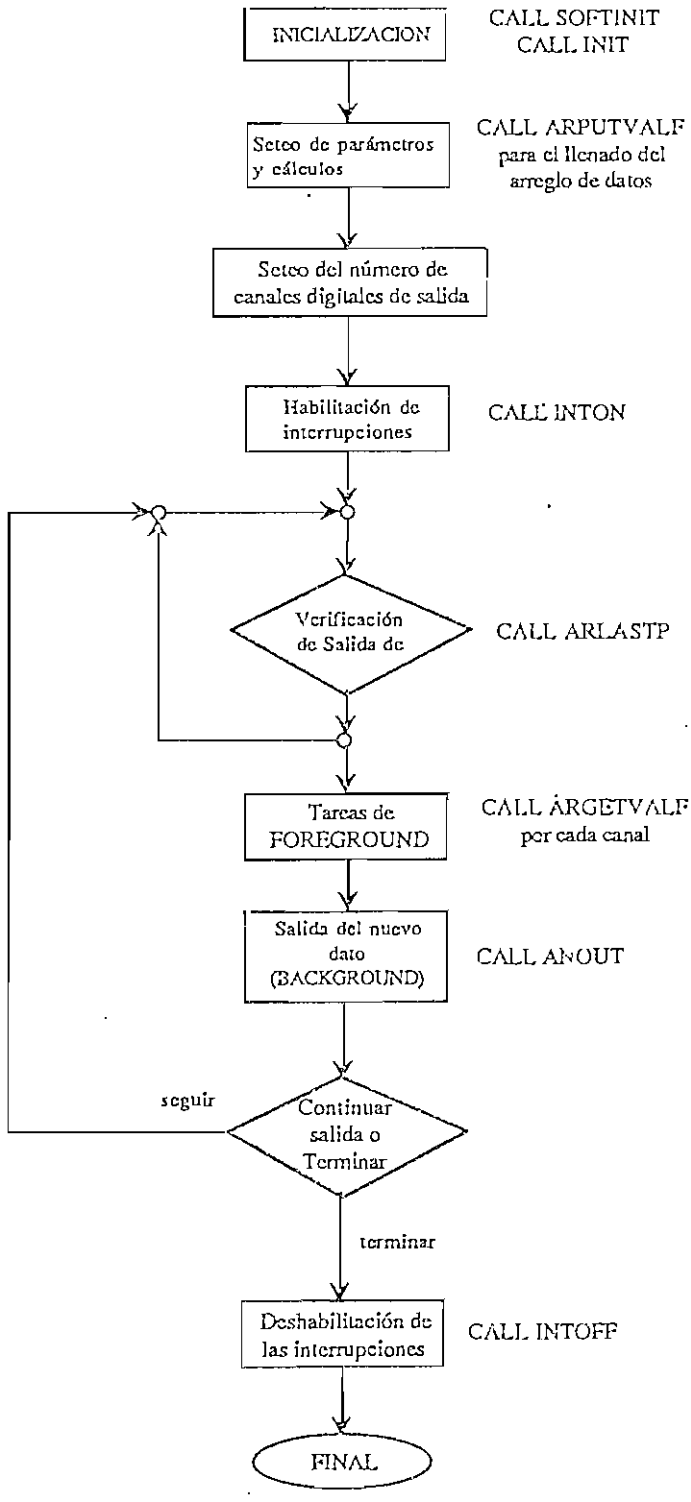


Figura 2.7.a.- Diagrama de flujo: Salida de datos multivariable en Background

```
CLS

CALL SOFTINIT
CALL INIT

'Instrucciones que inicializan en SOFTWARE y HARDWARE al sistema

COLOR 3
LOCATE 6, 6: INPUT "Periodo de muestreo (ms)= "; bintv%

ionl$ = "ANOUT0 ANOUT1 ANOUT2 ANOUT3 ANOUT4"
dep! = 100
arn$ = "datosen%"

'Seteo de los parámetros para creación del arreglo de memoria.

CALL ARMAKE("datosen%", dep!, -1, ionl$)

'Creación del arreglo de memoria para los datos de salida.

FOR dep! = 1 TO 100
        d1% = INT(SIN((dep! - 1) * 6.28 / 100) * 2047.5 + 2047.5)
        d2% = INT(COS((dep! - 1) * 6.28 / 100) * 2047.5 + 2047.5)

        CALL arputvali("datosen%", dep!, 1, ionl$, d1%)
        CALL arputvali("datosen%", dep!, 2, ionl$, d2%)

        'Comandos que colocan los valores calculados en arreglo de memoria correspondiente

NEXT dep!

FOR dep! = 1 TO 100
        va1% = INT(((2047.5 * dep! / 100)) + 2047.5)

      . CALL arputvali("datosen%", dep!, 3, ion'$, va1%)

NEXT
va4% = 4095: va5% = 0
FOR dep! = 1 TO 50
        va2% = INT(((2047.5 * dep! / 50)) + 2047.5)

        CALL arputvali("datosen%", dep!, 4, ionl$, va2%)
        CALL arputvali("datosen%", dep! + 50, -1, "ANOUT4", 0)

        va3% = INT(((2047.5 * (50 - dep!) / 50)) + 2047.5)

       CALL arputvali("datosen%", dep! + 50, 4, ionl$, va3%)
       CALL arputvali("datosen%", dep!, -1, "ANOUT4", 4095)

NEXT

CALL ANOUT("datosen%", ionl$, bintv%, -1, "nt", "tarea2")

'Comando que efectúa la salida de datos una vez que se habilitan las interrupciones.

COLOR 2
```

```
LOCATE 12, 20: PRINT "Salida de datos de ondas de "; 10 / bintv%; "Hz."
LOCATE 13, 11: PRINT "a través de dos canales análogos con amplitudes diferentes"
COLOR 3
LOCATE 25, 15: PRINT "Presione SPACE BAR para terminar la salida de datos"

CALL INTON(1, "mil")

'Comando que habilita las interrupciones del BACKGROUND.

an$ = ""
WHILE an$ = ""
      an$ = INKEY$
WEND

CALL INTOFF

'Comando que deshabilita las interrupciones
```

2.3.- SALIDA DE DATOS MEDIANTE CANALES DIGITALES.-

Dado que una salida digital, puede unicamente tener dos valores posibles, esto es un "uno lógico" o un "cero lógico", se prefirió trabajar a nivel de puertos, donde cada puerto contiene ocho canales (cada canal constituye un bit), teniendo por lo tanto un byte, con lo cual ya se puede tener diversos valores hasta +127, lo cual permite realizar un ejemplo visualizable. Todos los comandos que se utilizan para la salida de datos digitales ya han sido explicados en los apartados anteriores, por lo que no se incluirá en este caso dicho estudio, tal como ha sido el formato de trabajo seguido hasta ahora.

Existe sin embargo un comando que se utiliza debido a que se trabaja con puertos y no con canales y es el IONMDIG, ya que en el archivo CONFIG.TBL, que es la tabla de configuración se definieron los nombres de cada canal, no así los de los puertos, lo que obligó a definir sus nombres sobre la marcha con el comando antes anotado. No está por demás anotar que para los canales análogos, existe también la posibilidad de definir nombres con el comando IONMANA.

2.3.1.- SALIDA UNIVARIABLE DE DATOS DIGITALES EN FOREGROUND.-

Como en los casos anteriores tratados en este capítulo, el trabajo en foreground es abordado más bien desde el punto de vista didáctico, ya que como se ha visto, este modo de trabajo impide setear períodos de muestreo, realizar tareas diferentes y quizás lo más importante, no permite la creación de áreas o arreglos en formato Quick500, para almacenar datos a sacarse, o adquiridos, según sea el caso. Lo importante es que el ejemplo desarrollado permita estudiar los diferentes comandos del Quick500 y observar de manera casi inmediata su utilidad. Dicho ejemplo utiliza un circuito de diodos de luz, en el cual se enciende uno de ellos a la vez, y cada vez uno diferente.

DIAGRAMA DE FLUJO.- El diagrama de flujo de la salida de datos digitales unicanal en foreground se presenta en la figura 2.9.

LISTADO DE LA SUBRUTINA BASICA.-

```
CLS
LOCATE 1, 3: PRINT "Valor de Salida ="
LOCATE 20, 16: PRINT "presione SPACE BAR para terminar salida de datos"

CALL SOFTINIT
CALL INIT

'Instrucciones que inicializan en SOFTWARE y HARDWARE al sistema

CALL IONMDIG("PORT1", 5, -1, 1)

'Define que el puerto 1 del módulo AOM1, ubicado en el slot 5 se 'llama PORT1
ion$ = "PORT1"
iotype% = 4                    'salida de naturaleza digital
va! = 1
euf% = -1
tm$ = "nt"

'Definición de parámetros

WHILE INKEY$ <> CHR$(32)
      va! = (va! * 2)                    'cálculo del valor de salida
      IF va! > 128 THEN va! = 1
      LOCATE 1, 22: PRINT va!; SPACE$(2)

                  CALL WRITEVAR(ion$, iotype%, va!, euf%, tm$)
```

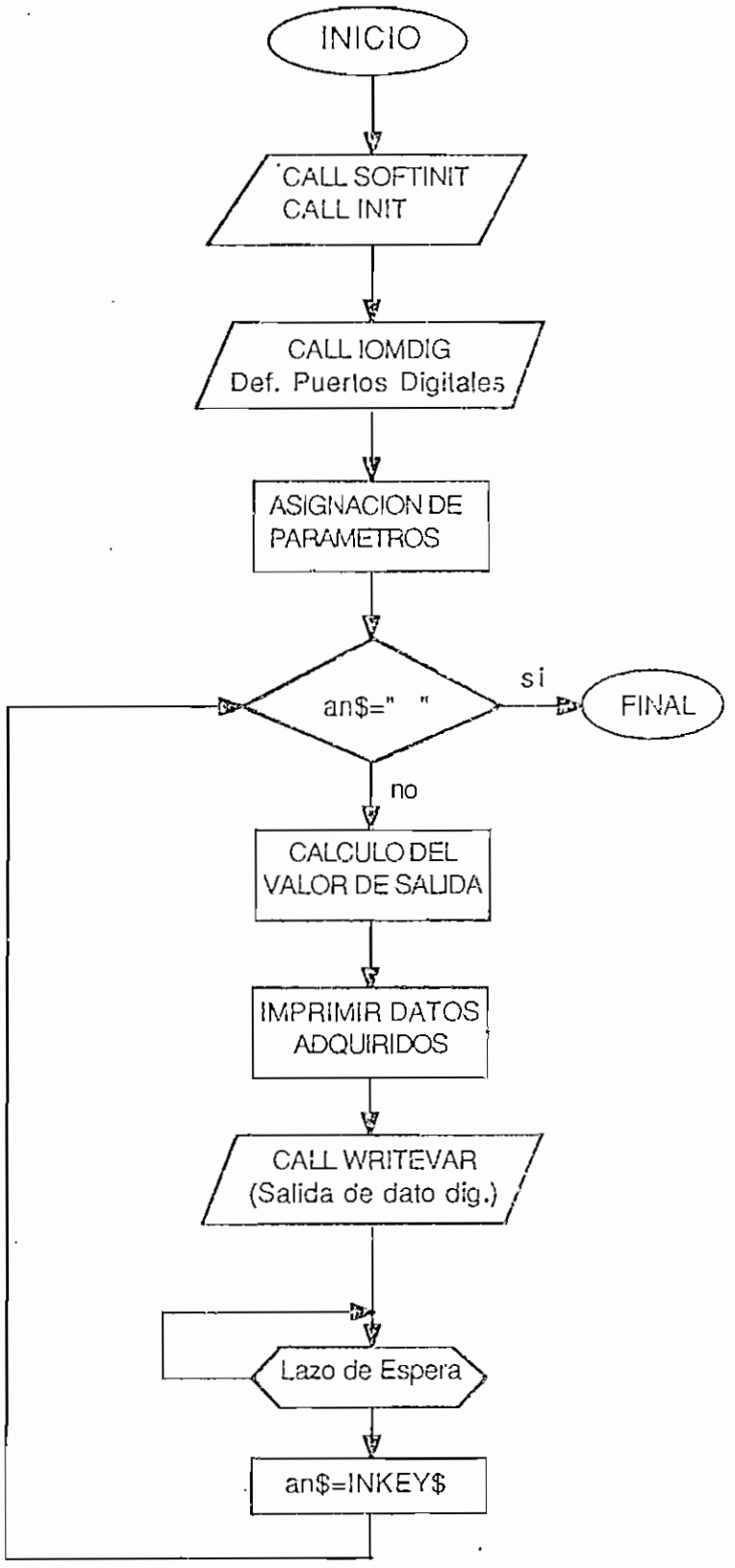


Figura 2.9.- Diagrama de flujo: Salida de datos digitales univariable en Foreground

```
'saca el valor calculado a travésd del puerto 1
FOR i = 1 TO 2000: NEXT i

WEND
```

2.3.2.- SALIDA MULTIVARIABLE DE DATOS DIGITALES EN FOREGROUND.-

Al igual que en el caso univariable, se trata este apartado de una manera más bien didáctica, por lo que se ha buscado un ejemplo que permita visualizar los resultados de una manera fácil, así mismo, como se tiene dos puertos únicamente, el ejemplo que se desarrolla para este caso particular, consiste en lo mismo que el anterior solo que en este caso se tiene la tarea para cada puerto de salida.

DIAGRAMA DE FLUJO.- El diagrama de flujo de la salida de datos digitales multicanal en foreground se presenta en la figura 2.10.

LISTADO DE LA SUBRUTINA BASICA.-

```
CLS

COLOR 14
LOCATE 4, 33: PRINT "Valor de Salida "
LOCATE 20, 16: PRINT "presione SPACE BAR para terminar salida de datos"

CALL SOFTINIT
CALL INIT

'Instrucciones que inicializan en SOFTWARE y HARDWARE al sistema

CALL IONMDIG("PORT0", 5, -1, 1)

'Define que el puerto 1 del módulo AOM1, ubicado en el slot 5 se 'llama PORT0

CALL IONMDIG("PORT1", 5, -1, 2)

'Define que el puerto 2 del módulo AOM1, ubicado en el slot 5 se 'llama PORT1

ion1$ = "PORT0 PORT1"
DIM va!(2)
va!(0) = 1
va!(1) = 128
iotype% = 4                    'salida de naturaleza digital
euf% = -1
tm$ = "nt"
'Definicion de parametros
```

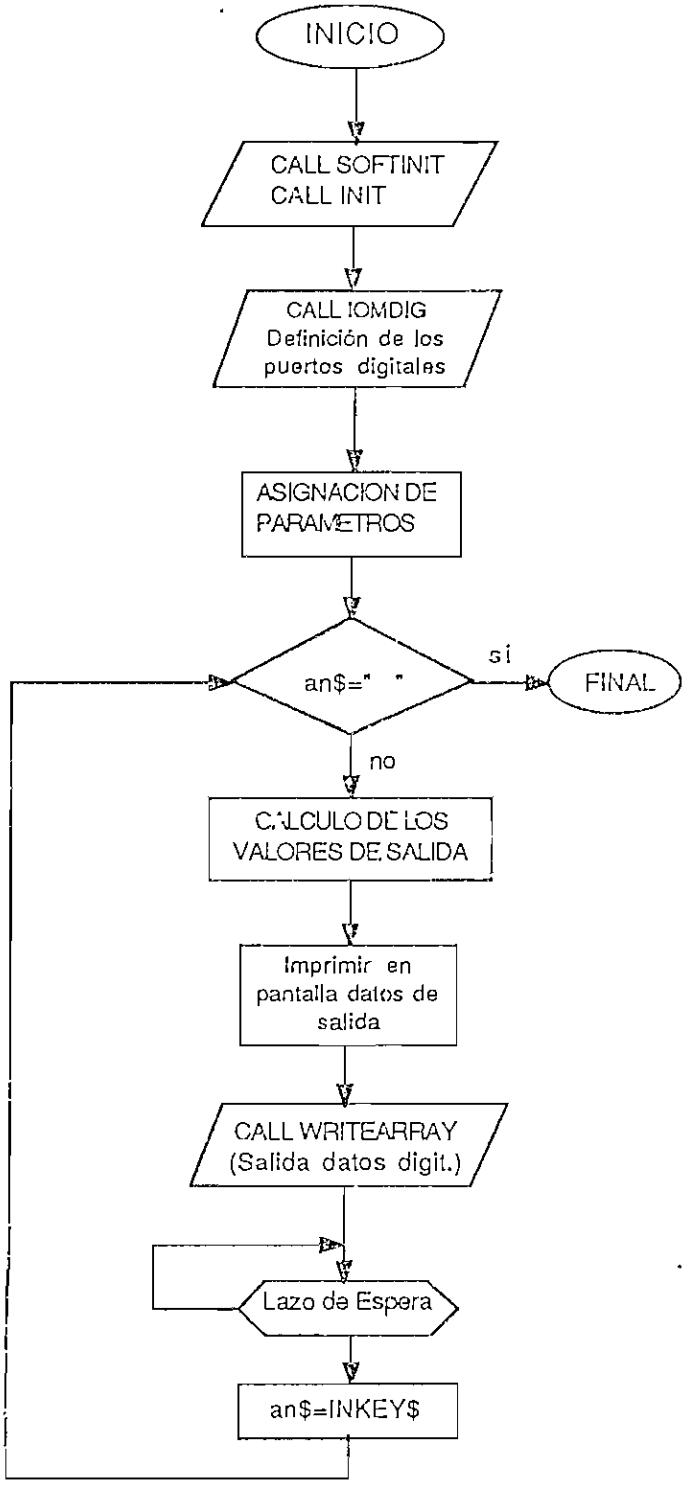


Figura 2.10.- Diagrama de flujo: Salida de datos digitales multivariable en Foreground

```
WHILE INKEY$ <> CHR$(32)
        va!(0) = va!(0) * 2: va!(1) = va!(1) / 2
        IF va!(0) > 128 THEN va!(0) = 1
        IF va!(1) < 1 THEN va!(1) = 128
        IF v > 8 THEN v = 0
        COLOR 4
         LOCATE 6, 43 + 4 * v: PRINT va!(0); SPACE$(2)
        COLOR 3
         LOCATE 6, 37 - 4 * v: PRINT va!(1)
        v = v + 1

          CALL WRITEARRAY(ionl$, iotype%, va!(), -1, tm$)

           'salida de datos por los puertos de salida digital resp.

        FOR I = 1 TO 2000: NEXT I

WEND
```

2.3.3.- SALIDA UNIVARIABLE DE DATOS DIGITALES EN BACKGROUND.-

El objetivo de esta sección es sacar datos en tiempo real a través de un solo puerto digital, mediante el modo de trabajo de fondo o background, sistema que utiliza interrupciones no mascarables.

Al tratarse de una subrutina que realiza trabajo de fondo mediante interrupciones que se presentan a intervalos de tiempo iguales, permite al tiempo que se realiza la salida de datos realizar cualquier otra actividad en foreground, como son impresiones, gráficos, análisis de datos, algoritmos de control, etc..

Al tratarse de datos digitales, cobra especial importancia el tener manejo sobre la frecuencia de salida de los datos, sobre todo cuando el objetivo final es realizar control. El ejemplo desarrollado tiene los mismos resultados que los ejemplos de foreground, con la diferencia que en este caso si se puede setear períodos de muestreo.

DIAGRAMA DE FLUJO.- El diagrama de flujo de la salida de datos digitales unicanal en background se presenta en la figura 2.11.

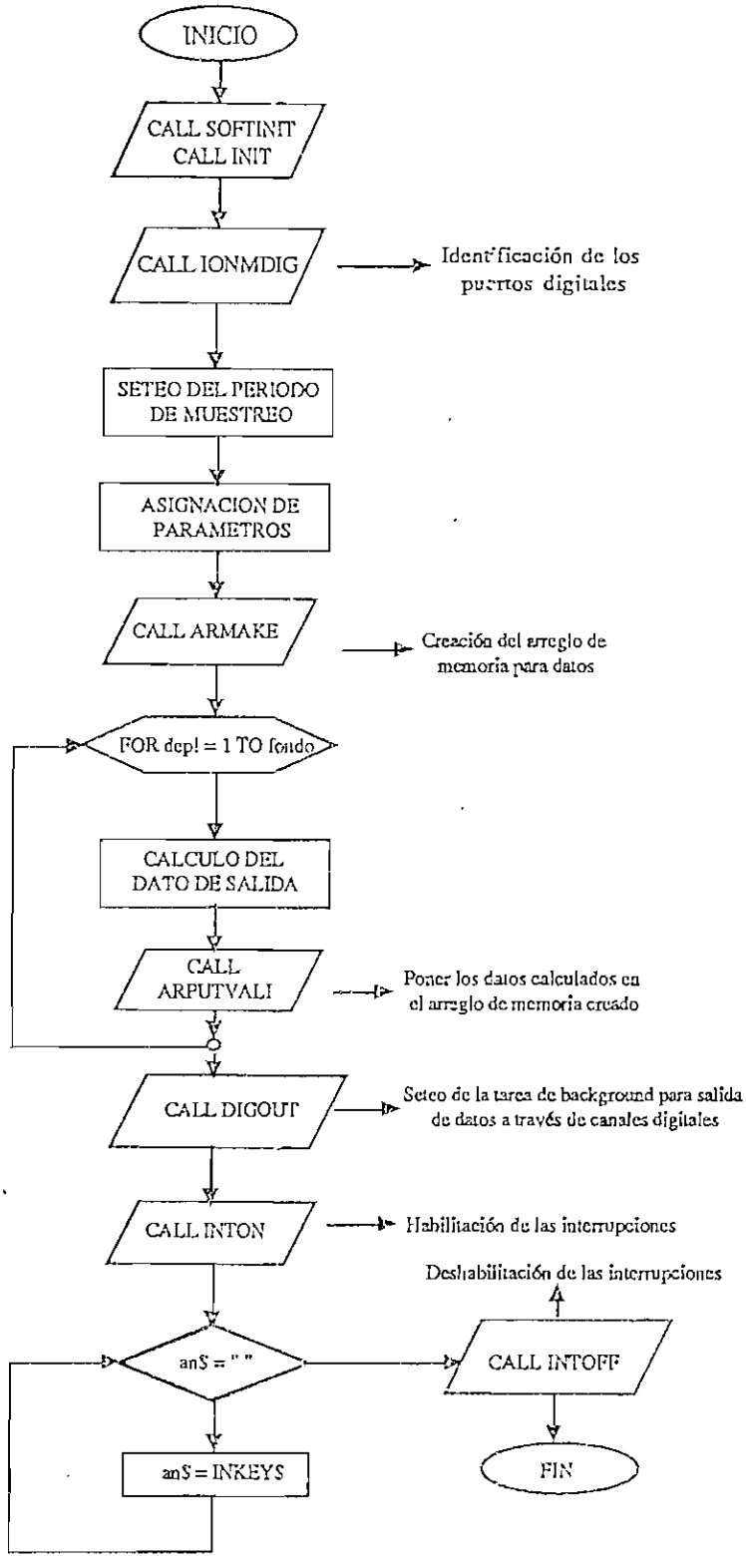


Figura 2.11.- Diagrama de flujo: Salida de datos digitales univariable en Background

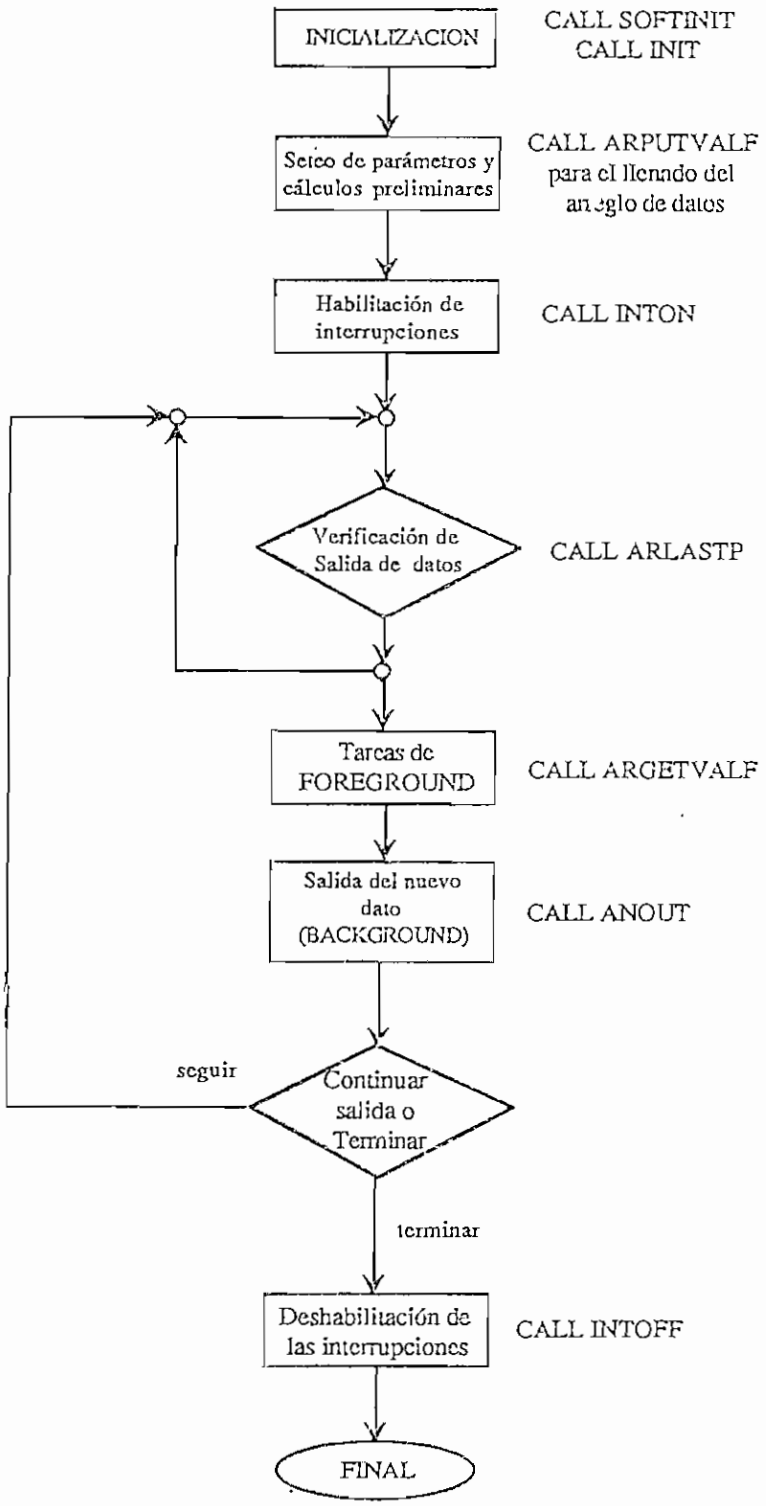


Figura 2.11.b.- Diagrama secuencial de tareas: Salida de datos digitales univariable en Background

LISTADO DE LA RUTINA BASICA.- El trabajo en background de las salidas digitales es muy similar al desarrollado para las salidas análogas, utiliza los mismos comandos del Quick500, y prácticamente sigue el mismo formato.

```
CLS
LOCATE 1, 3: PRINT "Inicio de la Salida de Datos"

CALL SOFTINIT
CALL INIT

'Instrucciones inicializan en SOFTWARE y HARDWARE al sistema

CALL IONMDIG("PORT1", 5, -1, 1)

ionl$ = "PORT1"
LOCATE 3, 3: INPUT "Periodo de muestreo (dms seg)= "; BINTV%
dep! = 15
arn$ = "datoarea%"
tm$ = "nt'

'Seteo de parámetros

CALL ARMAKE(arn$, dep!, -1, ionl$)

'Comando que crea el arreglo receptor de los datos de salida

otro:
FOR dep! = 1 TO 7

      COLOR dep!
      d% = 2 ^ dep!
      LOCATE 6 + dep!, 10 + 3 * dep!: PRINT d%

      CALL ARPUTVALI("datoarea%", dep!, -1, "PORT1", d%)

      'Toma de datos del arreglo de memoria para su salida

NEXT dep!
FOR dep! = 9 TO 15

      COLOR dep!
      d% = 2 ^ (16 - dep!)
      LOCATE 6 + 16 - dep!, 10 + 3 * dep!: PRINT d%

            CALL ARPUTVALI("datoarea%", dep!, -1, "PORT1", d%)

      'Toma de datos del arreglo de memoria para su salida

NEXT dep!

CALL DIGOUT(arn$, ionl$, BINTV%, -1, tm$, "tarea1")

'Comando que setea la tarea de background, de salida de datos 'digitales
```

```
LOCATE 5, 3: PRINT "Salida de datos."
COLOR 4
LOCATE 23, 15: PRINT "Presione SPACE BAR para terminar"

CALL INTON(100, "mil")

'Comando que habilita las interrupciones del BAKGROUND.

an$ = ""
WHILE an$ = ""
      an$ = INKEY$
WEND

CALL INTOFF

'Comando que deshabilita las interrupciones
END
```

2.3.4.- SALIDA MULTIVARIABLE DE DATOS DIGITALES EN BACKGROUND.-

El objetivo de este numeral es sacar datos en tiempo real a través de varios puertos digitales, mediante el modo de trabajo de fondo o background, sistema que utiliza interrupciones no mascarables. Todo aquello que se dijo de la salida de datos univariable es perfectamente aplicable a este caso ya que el modo de trabajo es el mismo. El ejemplo desarrollado tiene los mismos resultados que los ejemplos de foreground, con la diferencia que en este caso si se puede setear períodos de muestreo.

DIAGRAMA DE FLUJO.- El diagrama de flujo de la salida de datos digitales multicanal en background se presenta en la figura 2.12.

LISTADO DE LA RUTINA BASICA.- Esta rutina es desarrollada en base al formato utilizado para el caso univariable.

```
CLS
LOCATE 1, 3: PRINT "Inicio de la Salida de Datos"

CALL SOFTINIT
CALL INIT

'Instrucciones inicializan en SOFTWARE y HARDWARE al sistema
```

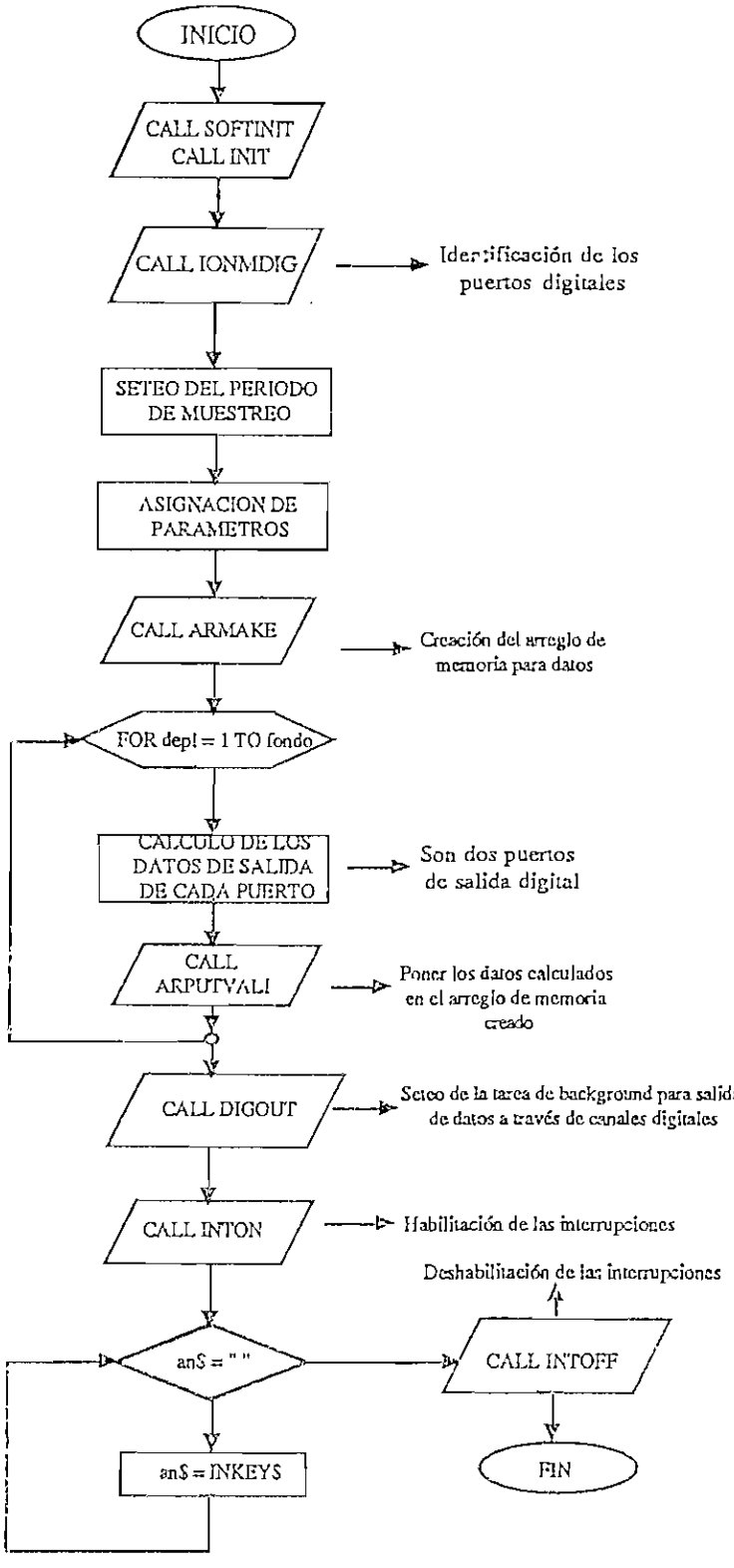


Figura 2.11.- Diagrama de flujo: Salida de datos digitales multivariable en Background

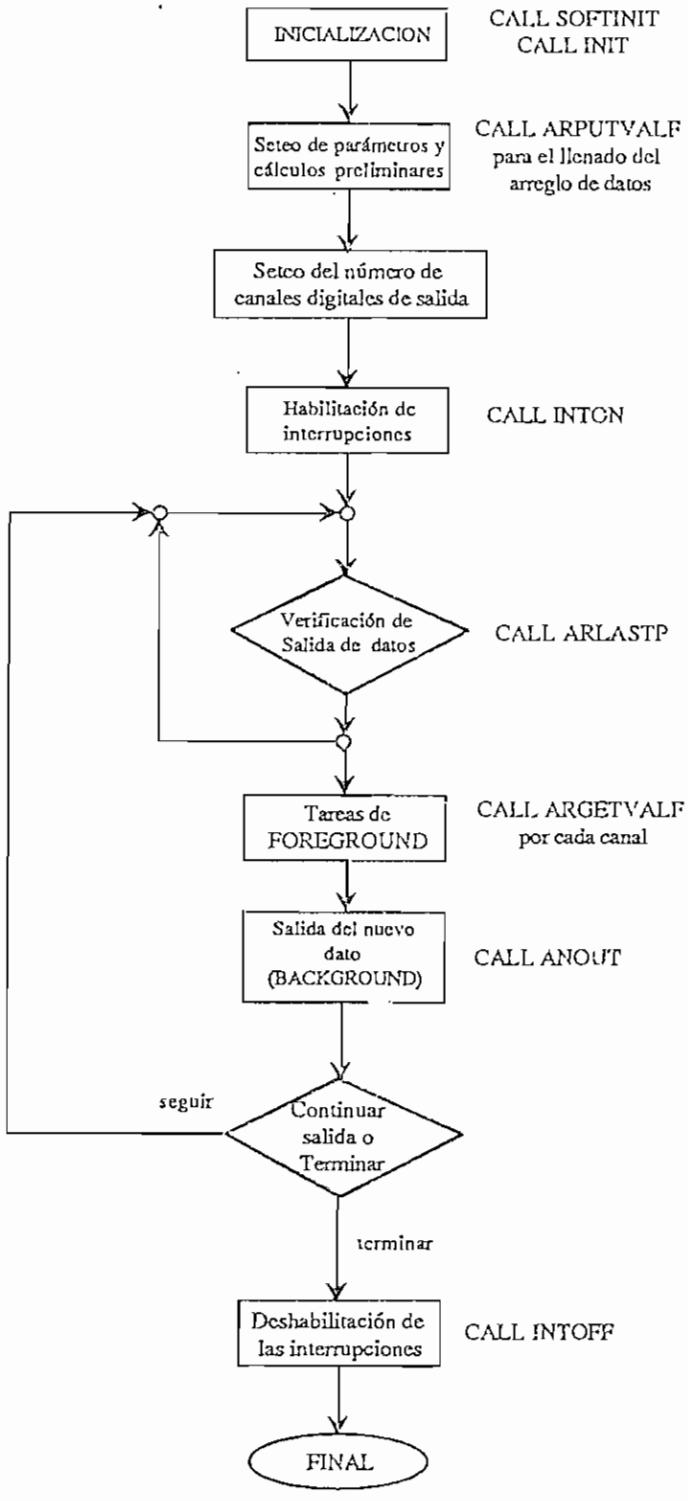


Figura 2.12.b.- Diagrama secuencial de tareas: Salida de datos digitales multivariable en Background

```
CALL IONMDIG("PORT0", 5, -1, 1)
CALL IONMDIG("PORT1", 5, -1, 2)

ionl$ = "PORT0 PORT1"
LOCATE 3, 3: INPUT "Periodo de muestreo (dcs seg)= "; BINTV%
dep! = 15
arn$ = "entrada%"
cy% = -1
tm$ = "nt"
wid% = 2
'Seteo de parámetros

CALL ARMAKE(arn$, dep!, wid%, ionl$)

'Comando que crea el arreglo de memoria donde se almacenan los 'datos de salida

otro:
FOR dep! = 1 TO 8

      d% = 2 ^ dep! - 1
      LOCATE 6 + dep!, 10 + 3 * dep!: PRINT d%

      CALL ARPUTVALI("datoarea%", dep!, 1, "", d%)

      'Comando que almacena datos de salida en el arrg. de mem.

NEXT dep!
FOR dep! = 9 TO 15

      d% = 2 ^ (16 - dep!) - 1
      LOCATE 6 + 16 - dep!, 10 + 3 * dep!: PRINT d%
      CALL ARPUTVALI("datoarea%", dep!, 1, "", d%)

      'Comando que almacena datos de salida en el arrg. de mem.

NEXT dep!

FOR dep! = 1 TO 8

      d1% = 2 ^ (9 - dep!) - 1
      LOCATE 15 + dep!, 10 + 3 * dep!: PRINT d1%
      CALL ARPUTVALI("datoarea%", dep!, 2, "", d1%)

      'Comando que almacena datos de salida en el arrg. de mem.

NEXT dep!
FOR dep! = 9 TO 15

      d1% = 2 ^ (dep! - 7) - 1
      LOCATE 15 + 16 - dep!, 10 + 3 * dep!: PRINT d1%
      CALL ARPUTVALI("datoarea%", dep!, 2, "", d1%)

      'Comando que almacena datos de salida en el arrg. de mem.

NEXT dep!

CALL DIGOUT("datoarea%", ionl$, BINTV%, -1, "nt", "tarea1")

'Comando que setea la tarea de BACKGROUND, salida de datos 'digitales
```

```
LOCATE 5, 3: PRINT "Salida de datos."
COLOR 4
LOCATE 25, 15: PRINT "Presione SPACE BAR para terminar"

CALL INTON(100, "mil")

'Comando que habilita las interrupciones del BAKGROUND.

an$ = ""
WHILE an$ = ""
      an$ = INKEY$
WEND

CALL INTOFF

'Comando que deshabilita las interrupciones
```

2.4.- COMANDOS ADICIONALES.-

Existen comandos que han sido utilizados en este capítulo referentes al seteo de los canales de adquisición y salida de datos, que se han quedado sin explicación al no caber en los temas que se ha estado tratando. Además tampoco cabe su explicación dentro de los temas que se han de tratar en el capítulo tres, por lo que se hace a continuación una ligera descripción de ellos.

La utilización de estos comandos reemplaza la acción del programa CONFIG y de las tablas de configuración que este crea.

CALL IONMANA

PROPOSITO: Comando que asocia una cadena de caracteres con un canal de entrada o salida de la estación KEITHLEY 500A, identificando el slot de ubicación del módulo, canal, conversor análogo digital y su resolución, señal de acondicionamiento análoga.

CALL IONMANA(ion$, slot%, chan%, res%, ga%, strg%)

PARAMETROS:

ion$ (string-entrada).- Nombre con el que se le conocerá al canal en adelante, hasta la finalización del programa en curso.

slot% (integer-entrada).- El valor asignado a este parámetro indica cual de los 10 slots se está usando. El valor dado a este parámetro afectará al multiplexer global del módulo AIM1, el cual selecciona 1 de las 14 entradas para pasar junto con el conversor A/D. Los valores válidos son del 0-15; del 1 al 10 especifican slots. 0, 13, 14, 15 especifican voltajes de referencia tomados desde la fuente de poder del sistema.

chan% (integer-entrada).- El valor asignado a este parámetro identifica el canal utilizado. Este parámetro afecta al multiplexer local del módulo afectado. Los valores válidos son de 0-32, o el número de canales existentes en el módulo. Cuando a slot% se asigna valor para voltaje de referencia, el parámetro chan% debe ser 0.

ga% (integer-entrada).- El número que se asigne a este parámetro indica la ganancia de la señal sensada. tiene cuatro valores posibles 1, 2, 5, 10.

res% (integer-entrada).- Indica la resolución del conversor A/D o D/A. Puede ser de 12 o 14 bits.

CALL IONMDIG

PROPOSITO: Comando que asocia una cadena de caracteres con un canal de entrada o salida de la estación KEITHLEY 500A, identificando el slot de ubicación del módulo, canal o puerto.

CALL IONMDIG (ion$, slot%, chan%, port%)

PARAMETROS:

ion$ (string-entrada).- Nombre con el que se le conocerá al canal en adelante, hasta la finalización del programa en curso.

slot% (integer-entrada).- El valor asignado a este parámetro indica cual de los 10 slots se está usando. El valor dado a este parámetro afectará al multiplexer global del módulo AIM1, el cual selecciona 1 de las 14 entradas para pasar junto con

el conversor A/D. Los valores válidos son del 0-15; del 1 al 10 especifican slots. 0, 13, 14, 15 especifican voltajes de referencia tomados desde la fuente de poder del sistema.

chan% (integer-entrada).- El valor asignado a este parámetro identifica el canal utilizado. Este parámetro afecta al multiplexer local del módulo afectado. Los valores válidos son de 0-32, o el número de canales existentes en el módulo. Cuando se va a utilizar puertos digitales, este parámetro debe ser -1.

port% (integer-entrada).- Cada puerto se compone de 8 canales digitales, entonces máximo se puede tener 4 puertos, por lo que son válidos los valores 0-3. Cuando se utiliza canales, este parámetro debe valer -1.

CAPITULO III

3 OPERACION MULTITAREA

3.0 INTRODUCCION

3.1 INTERRUPCIONES Y PERIODO DE MUESTREO

3.2 DISPAROS

3.3 INTERRUPCIONES Y PERIODO DE MUESTREO

3.4 MANEJO DE ARREGLOS DE MEMORIA PARA DATOS

3.5 PROCESAMIENTO DE DATOS

3.6 TRANSFORMADA RAPIDA DE FOURIER

3.7 LISTADO DEL SOFTWARE MULTITAREA

3.0.- INTRODUCCION.-

Una vez que en el capítulo 2 se ha hecho una descripción del software fundamental del sistema KEITHLEY 500A, como es la adquisición y salida de datos, ahora en el presente capítulo se tratará acerca de opciones adicionales que entrega el Quick500, tales como setear períodos de muestreo, interrupciones, graficación en tiempo real, manejo de arreglos de memoria para datos, tanto para almacenamiento como para exportación y procesamiento de datos. Se ha querido además tener como opción el algoritmo de la transformada rápida de Fourier, la misma que se obtiene sobre un arreglo de datos almacenados en memoria; el afán de esta opción es demostrar que también se puede hacer análisis en frecuencia de los datos adquiridos de cierta planta o proceso.

El trabajo en este capítulo se ha dividido en secciones que permiten describir las diferentes tareas que se pueden realizar en foreground, mientras en bakground se realiza adquisición y/o salida de datos. En cada una de dichas secciones se hace también una explicación breve de todos los comandos del Quick500 que se vayan utilizando por primera vez; se presenta además un ejemplo de utilización del tema tratado, del cual se presenta un diagrama secuencial de tareas. Al final del capítulo se presenta el listado del paquete "software multitarea", el mismo que contiene como subrutinas los ejemplos desarrollados para cada numeral del capítulo 3.

3.1.- INTERRUPCIONES Y PERIODO DE MUESTREO.-

Si bien el período de muestreo se basa y depende de las interrupciones, se debe hacer una diferenciación, pues se trata de dos conceptos distintos del Quick500. Las interrupciones tienen su propio período, el cual es impuesto únicamente por el comando CALL INTON y sus parámetros, mientras que el

período de muestreo, tiene como base el período de las interrupciones, pero se fija mediante un parámetro adicional, el bintv% (intervalo de background) de los comandos de entrada y salida de datos en background como ANIN, ANOUT, DIGOUT.

3.1.1.- INTERRUPCIONES.-

Como se ha anotado antes, el período de las interrupciones es fijado únicamente por el comando CALL INTON. el mismo que al habilitar las interrupciones, deja seteado también el período con que estas se presentan. Dicho comando crea un reloj base, el mismo que va a entregar el mando de manera alternativa a uno de los dos esquemas de trabajo con que se cuenta, el foreground y el background, tal como se muestra en la figura 3.1.

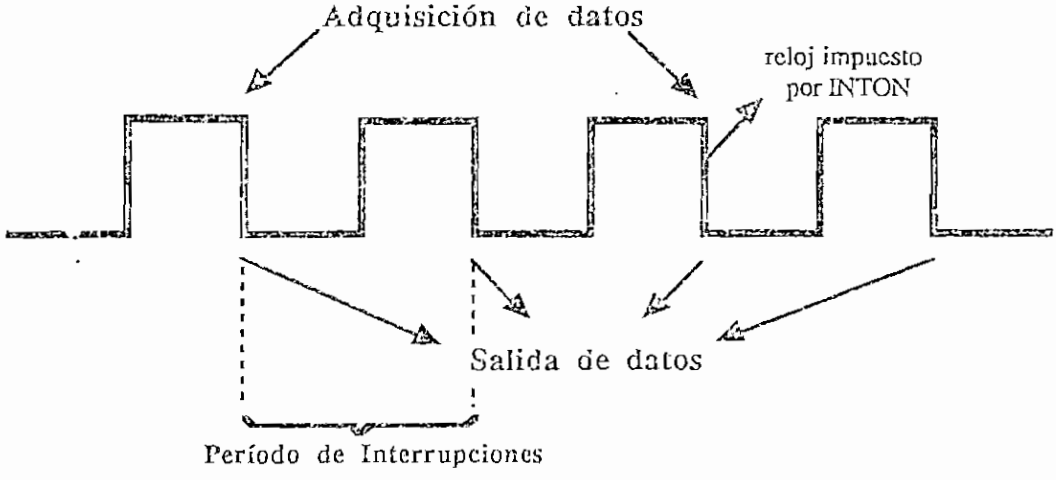


Figura 3.1.- Período de interrupciones.

Una vez que las tareas de background han sido terminadas, se deshabilita las interrupciones mediante el comando CALL INTOFF. Los comandos INTON e INTOFF fueron explicados anteriormente.

3.1.2.- PERIODO DE MUESTREO.-

El muestreo de los diferentes canales sea de adquisición o de salida de datos no se realiza necesariamente cada interrupción, sino más bien cada cierto número de interrupciones, número que es impuesto por el parámetro bintv% presente en cada uno de los comandos de adquisición y salida de datos en background, lo cual deja la opción de determinar un período de muestreo diferente para cada tarea de background, con la base de un único período de interrupciones.

Esto cobra gran importancia en control, puesto que el período de muestreo que se impone, suele ser muy superior al tiempo que se demora entre la adquisición del dato hasta la obtención del resultado que debe ser devuelto a la planta, dato que en teoría debería ser entregado inmediatamente (sin ningún retardo), de allí que mientras menor sea el retardo, menor será el error cometido. Si bien se tiene diferentes períodos de muestreo para cada tarea, es conveniente que en cada nuevo muestreo de entrada (siempre mayor o igual que el de salida), se repita un ciclo, lo cual trae la necesidad de sincronizar las tareas, lo que se consigue teniendo multiplicidad exacta entre los períodos de muestreo de entrada y de salida, tal como se muestra en la figura 3.2.

Se debe hacer un estimado del tiempo mínimo que se requiere para obtener el dato de salida (adquisición + conversión A/D + algoritmo), lo cual a su vez ayuda a saber cual es el tiempo mínimo de muestreo de entrada Tmín. Este tiempo mínimo sirve para determinar los períodos de muestreo de entrada y salida definitivos.

$$V = INT\left(\backslash F(T_{m\text{-}in}; T_{mín})\right) \qquad T_{m\text{-}out} = INT\left(\backslash F(T_{m\text{-}in}; V)\right)$$

$$T_{m\text{-}in\ mod} = T_{m\text{-}out} * V$$

Donde Tm-in es el período de muestreo de entrada deseado, mayor que Tmín, Tm-out período de muestreo de salida mínimo posible pero mayor que Tmín y Tm-in mod período de muestreo de entrada modificado.

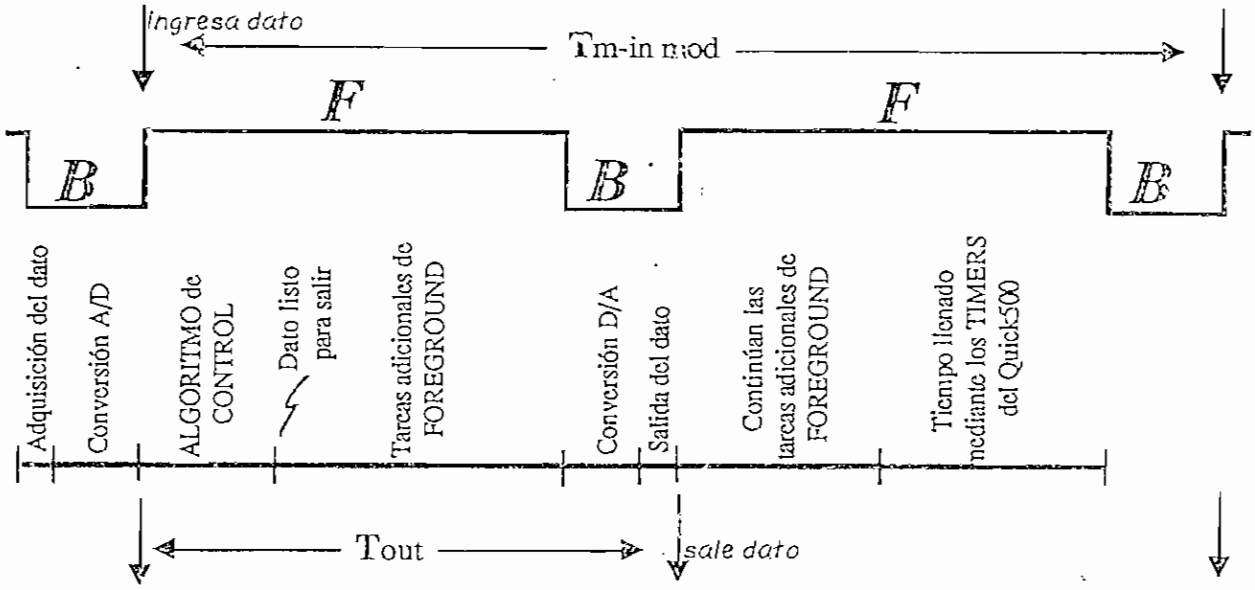


Figura 3.2.- Periódos de muestreo y tareas de backgound y foreground.

Como se observa en la figura 3.2, todo el tiempo que sobra desde el final de las tareas de foreground, hasta la nueva interrupción es llenado con la actividad de los timer (del Quick500), con lo cual se asegura tener utilizado todo el tiempo de máquina, esto a su vez asegura que todas las actividades de foreground se realicen una vez por cada dato adquirido, es decir, que también se sincroniza de esta manera las tareas de foreground y background.

COMANDOS ESPECIFICOS DEL QUICK500.- Los comandos nuevos que se utilizan, son aquellos que manejan los temporizadores del Quick500 y el comando STATUS, que ayuda a conocer el estado actual de los timers.

CALL TIMERSTART

PROPOSITO: Inicializa los 8 timers del Quick500, los timers incrementan su cuenta cada interrupción, por lo que la resolución de los timers es determinada por la frecuencia de las interrupciones. Es una tarea de background.

FORMATO: CALL TIMERSTART(tim%(), tm$, bfn$)

PARAMETROS:

tim%() (integer-entrada).- Es un arreglo unidimensional de 8 elementos, correspondiente a los timers 0-7 del Quick500. Se tiene como valores válidos para este parámetro, los siguientes:

-1	Detiene el timer y congela el valor actual.
0	Deshabilita el timer.
1	Resetea el timer y comienza la cuenta.

tm$ (string-entrada).- Modo de disparo, este comando puede ser disparado por otro comando de background o por el comando CALL GONOW. Las tres opciones válidas son: "wbt", "wgo", "nt".

bfn$ (string-entrada).- Permite asignar un nombre a la tarea de background TIMERSTART, este nombre es necesario si la tarea va a ser accesado por HALT o STATUS, si no se utiliza, se debe poner un string nulo " ".

CALL TIEMRREAD

PROPOSITO: Lee uno de los 8 timers del Quick500, y permite transferir el valor del tiempo transcurrido a una variable BASIC.

FORMATO: CALL TIMERREAD(val#())

PARAMETRCS:

val#() (real doble precisión-entrada).- Arreglo unidimensional de 8 elementos representa los valores asociados con los timers del Quick500. El valor

obtenido en cada elemento de este arreglo representa el tiempo transcurrido en términos del período de interrupción. Por ejemplo si se tiene una interrupción cada 10 ms., y el valor de val#(2) = 1689, entonces el tiempo transcurrido desde que el timer 2 fue activado es de 16.89 segundos.

CALL STATUS

PROPOSITO: Indica el estado actual de una tarea de background especificada. Este comando informa si la tarea está inactiva o no presente, ejecutándose o esperando algún disparo.

FORMATO: CALL STATUS(bfn$, status%)

PARAMETROS:

bfn$ (string-entrada).- Indica cual es la tarea de background de la que se está obteniendo el estado actual. Si el comando STATUS no encuentra la tarea, la reportará como inactiva.

status%(integer-salida).- Indica el estado de la tarea de background, tiene 4 valores válidos:

0	Tarea inactiva o no se encuentra.
1	Tarea en ejecución.
2	Tarea esperando disparo de background ("wbt").
3	Tarea esperando al comando CALL GONOW ("wgo").

En este numeral se ha desarrollado un ejemplo de entrada y salida de datos en background, utilizando las opciones que han sido descritas, dicha rutina acepta cualquier algoritmo sin mayores cambios.

DIAGRAMA SECUENCIAL DE TAREAS.- En la figura 3.3 se presenta el diagrama secuencial de tareas de la rutina de adquisición y salida de datos en background, nótese que los procesos de tiempo real no dependen del algoritmo introducido.

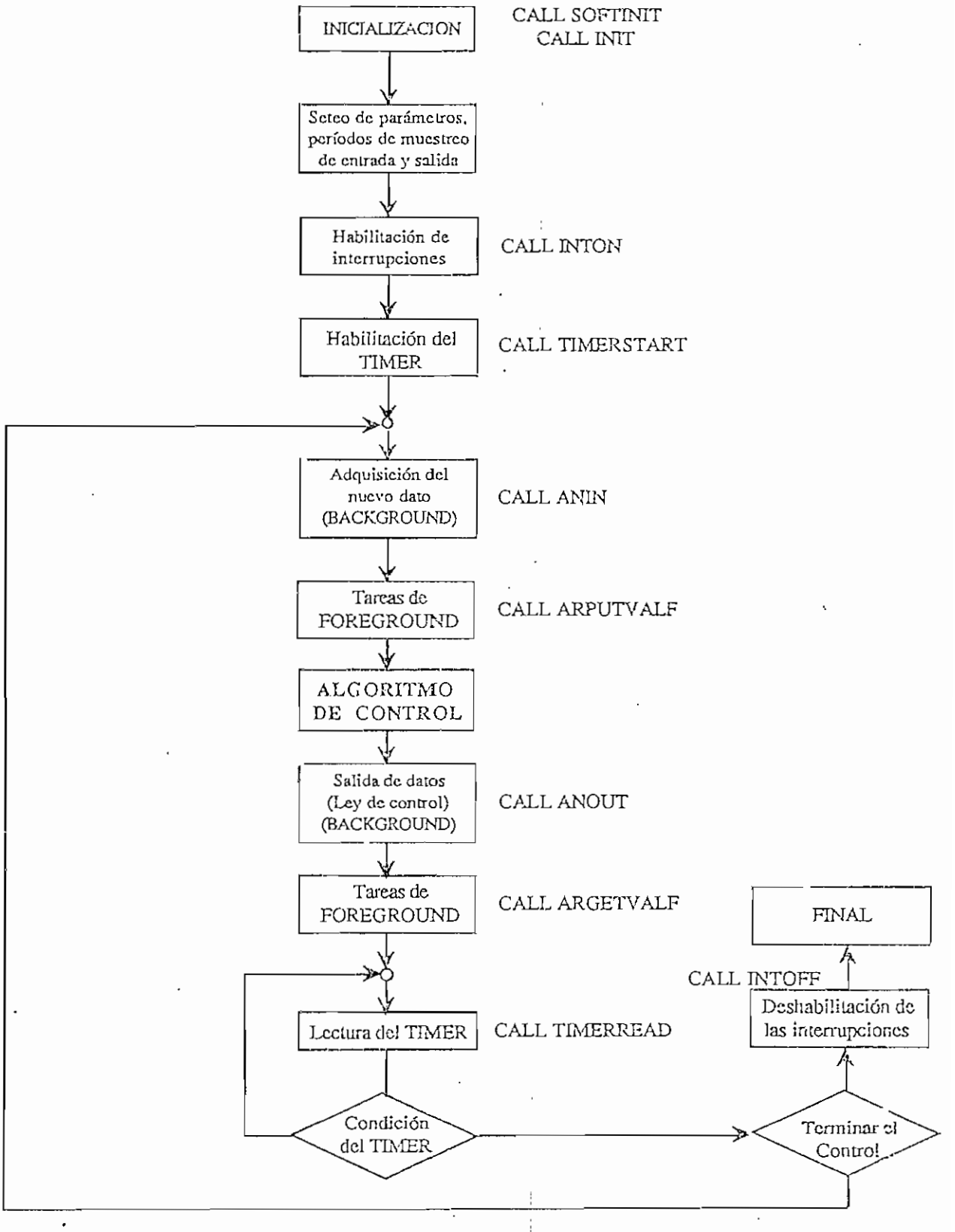


Figura 3.3.- Diagrama secuencial de tareas: Adquisición y salida de datos en background.

LISTADO DE LA RUTINA.- En el listado del paquete de "software multitarea" que se presenta al final del capítulo, El ejemplo de entrada y salida de datos en background es la subrutina llamada "INOUTGR".

3.2.-DISPAROS.-

Los disparos constituyen una herramienta muy utilizada en la adquisición de datos y sistemas de control. Provee medios para, en condiciones externas o internas obtener una respuesta programada. Un ejemplo de disparo externo puede ser el abrir y cerrar de un interruptor que activa un comando de background. Esta comunicación bi-direccional entre el computador y el mundo real es muy utilizada en procesos de control en tiempo real. Un ejemplo de disparo interno puede ser el caso de la terminación de una tarea de adquisición análoga de datos, lo que provoca el inicio de una secuencia digital de salida; este modo de disparo usa dos comandos de background operados en secuencia. Se describen 5 tipos de disparo que se pueden utilizar en el Quick500, cada uno de los cuales tiene ventajas y desventajas, pero utilizados adecuadamente ofrecen un soporte completo para todas las necesidades.

1.- DISPARO DESDE BASIC.- El propósito es suspender la ejecución de ciertos comandos hasta que se cumpla una condición deseada. Utiliza el "WHILE.....WEND" del BASIC. Entre las ventajas que presenta se menciona la facilidad de usar sentencias de BASIC o Quick500 como expresión condicional y para adquirir información; pero la desventaja es que el lazo en el cual se prueba la condición para el disparo es muy lento y la operación de foreground es suspendida hasta comprobarse la condición.

2.- DISPARO DE FOREGROUND.- Suspende la ejecución del BASIC o el Quick500 subsecuente a un comando de foreground, hasta que una condición deseada sea

verdadera. Para su ejecución el comando de disparo del Quick500 es llamado: la condición deseada se incluye en sus parámetros, por ejemplo el comando SCHMITRIG chequeará si una entrada es mayor a 10 voltios

```
CALL SCHMITRIG("ANLG0", 10!, 10!, "above",.....)
```

Una vez que la condición se ha cumplido, se ejecutan las subsecuentes sentencias que pueden ser de una instrumentación de alarma por ejemplo.

```
PRINT "ALARMA"
BEEP
```

Los dos comandos que se pueden utilizar en disparo de foreground son SCHMITRIG y DIGINTRIG (no disponible). Entre sus ventajas se puede mencionar su velocidad, dado que el lazo de prueba está hecho en asembler, el comando o sentencia que se ejecuta al cumplirse la condición puede ser de BASIC o del Quick500.

Las desventajas son sus limitaciones, dada la capacidad de los comandos de disparo de foreground y la paralización de las actividades de este esquema de trabajo hasta que la condición sea cumplida.

3.- DISPARO EN SINGLEGROUND.- Suspende la ejecución de un comando Quick500 que debe reaccionar inmediatamente a una condición, hasta que ésta sea verdadera. Se implementa en dos pasos: 1º Se llama al comando de disparo en singleground "st" (modo de trabajo simple), entregándose toda la información referente al disparo; 2º se ejecuta un comando Quick500, utilizando el parámetro tm$ = "wst". Ambos comandos pueden estar separados entre si por varias líneas de programa que no contengan ningún otro "st".

Existen 9 comandos que tienen la capacidad de esperar un disparo de singleground: ANINQ, HREADVAR, HREADARRAY, HWRITEVAR, HWRITEARRAY,

READVAR, READARRAY, WRITEVAR, WRITEARRAY. Los comandos de disparo en singleground son los mismos de foreground, estos son: SCHMITRIG y DIGINTRIG.

La ventaja de este modo de disparo es que al estar escrito en asembler el lazo de prueba de la condición, la respuesta es sumamente rápida. Sus desventajas son las limitaciones de las condiciones sobre las que se debe tomar la decisión y la suspensión de las actividades de foreground hasta que se cumpla la condición.

4.- DISPARO DE BACKGROUND.- El propósito de un disparo desde background es vigilar la entrega de una condición sin detener el proceso del foreground. Cuando se cumple la condición, la tarea es activada o disparada y la tarea de disparo se desactiva. Para realizar un disparo desde el background sólo se necesita cumplir con dos pasos, primero llamar un comando del Quick500 con el parámetro "modo de diparo" en "bt" (background trigger), éste setea la información acerca del disparo. Un segundo paso es llamar un comando con el parámetro "modo de disparo" en "wbt" (wait a background trigger). Estos dos comandos pueden quedar separados por varias líneas de programa, pero que no contengan un "bt" adicional.

Al correr un programa, el comando "bt" no es activado si se lo encuentra primero, únicamente se setea la tarea sin ser comunicada al background, y el control pasa directamente al siguiente comando. Cuando es encontrado un comando "wbt", las tareas "bt" y "wbt" son comunicadas juntas al background, se activa la tarea "bt" que comienza por chequear la condición de disparo así como la siguiente interrupción. La condición de disparo se chequea cada interrupción, la misma que al cumplirse dispara la tarea "wbt" mientras que la tarea "bt" se auto desactiva.

Los comandos que tienen la capacidad disparar otra tarea desde background ("bt"), son: DIGINTRG, FREQIN, DIGIN, ANIN, SCHMITRIG, PULSEIN, DIGOUT, ANOUT.

Mientras que 12 comandos tienen la capacidad de esperar un disparo desde BACKGROUND ("wbt"): ANIN, FREQIN, GONOW, TIMERSTART, DIGOUT, ANOUT, HALT, PULSECOUNT, DIGIN, CLOCKREAD, PULSEIN, TIMERREAD. Los comandos que han sido resaltados son los que se dispone en la actualidad.

Las interrupciones no necesariamente deben estar activadas cuando comandos con "bt" y "wbt" son llamados, sin embargo si deben activarse para ser ejecutados estos comandos.

Este tipo de disparo obtiene una respuesta sumamente rápida, en el orden de los useg., ya que las tareas que esperan disparo "wbt", son ejecutadas inmediatamente después de las tareas "bt", siguiendo una secuencia de background.

5.- DISPARO POR GONOW.- El comando GONOW provee el medio para disparar hasta 16 tareas de background simultáneamente. GONOW puede ser autodisparado en background o en foreground. Se implementa en dos pasos: 1º se setea 1-16 comandos de background con el modo de disparo "wgo" (wait a gonow). En un segundo paso es llamado el comando GONOW, si es en foreground será ejecutado inmediatamente y si es llamado en background, puede esperar disparo de background "wbt", donde será ejecutado por un comando que contenga "bt".

Las ventajas de este modo de disparo son importantes: el poder disparar hasta 16 tareas simultáneamente y la ventaja del comando GONOW de poder ser considerado como foreground o como background. La única desventaja es que cuando es disparado por otro tipo de disparo, asume sus desventajas.

COMANDOS ESPECIFICOS DEL QUICK500.- Los comandos que han sido utilizados en lo referente a disparos son SCHMITRIG y GONOW, de los cuales se incluye una descripción breve a continuación

CALL SCHMITRIG

PROPOSITO: Comando que lee un canal de entrada análoga y determina si una señal ha cumplido una condición específica, por ejemplo, si ha excedido un nivel de disparo. Este comando seguirá muestreando el canal continuamente hasta que la condición impuesta sea verdadera, cuando esto sucede, SCHMITRIG habilita la ejecución de otros procesos asociados a este comando mediante el parámetro del modo de disparo. SCHMITRIG opera como disparo de background, foreground e incluso singleground.

FORMATO: CALL SCHMITRIG(ion$, thrl!, thrh!, chm$, euf%, tm$, bfn$, cy%)

PARAMETROS:

ion$ (string-entrada).- Parámetro que indica el canal de entrada análoga que va a ser leido por SCHMITRIG.

thrl! (real-entrada).- Parámetro que setea el nivel umbral bajo de la señal que está siendo monitoreada.

thrh! (real-entrada).- Parámetro que setea el nivel umbral alto de la señal que está siendo monitoreada, thrh! debe ser siempre mayor que thrl!.

chm$ (string-entrada).- Permite especificar el estado de la señal monitoreada que causa el disparo a ser implementado. Existen cuatro opciones válidas:

"above"	Señal mayor o igual que thrh!
"below"	Señal menor o igual que thrl!
"betw"	Señal entre thrh! y thrl! pero no igual a ninguna.
"notbetw"	Señal $\geq$ que thrh! o $\leq$ que thrl!.

euf%.- (integer-entrada).- Setea las unidades en que se expresan thrh! y thrl!. El 0 setea voltios y el -1 valores binarios.

CALL GONOW

PROPOSITO: Permite disparar hasta 16 tareas de background, que tienen el modo de disparo seteado como "wgo". Cuando se ejecuta el GONOW dichas tareas son ejecutadas simultáneamente. GONOW puede ser configurada tanto como tarea de foreground como de background, pudiendo en este modo ser disparado ("wbt") por otra tarea de background ("bt").

FORMATO: CALL GONOW(tm$, bfn$)

PARAMETROS:

tm$ (string-entrada).- Determina el modo de disparo usado por el GONOW en background. Existen dos opciones válidas para este parámetro:

"wbt" wait a background trigger

"nt" no trigger. (ejecuta al GONOW como rutina foreground)

bfn$ (string-entrada).- Permite asignar un nombre a la tarea de background, de manera que esta pueda ser accesada. Cuando se utiliza esta tarea como de background, se puede setear como un string nulo.

Se ha desarrollado un ejemplo en el que se utiliza los modos de disparo más utilizados, como son el disparo desde background, disparo por GONOW y disparo desde BASIC. Dicho ejemplo realiza un control ON-OFF de cualquier planta.

DIAGRAMA SECUENCIAL DE TAREAS.- En la figura 3.4 se presenta el diagrama secuencial de tareas de la rutina de un control ON-OFF realizado a base de los disparos del Quick500, dicho proceso es realizado en background, por lo que se utiliza los tipos de disparo que en este caso son pertinentes, esto es disparo en background y disparo por GONOW.

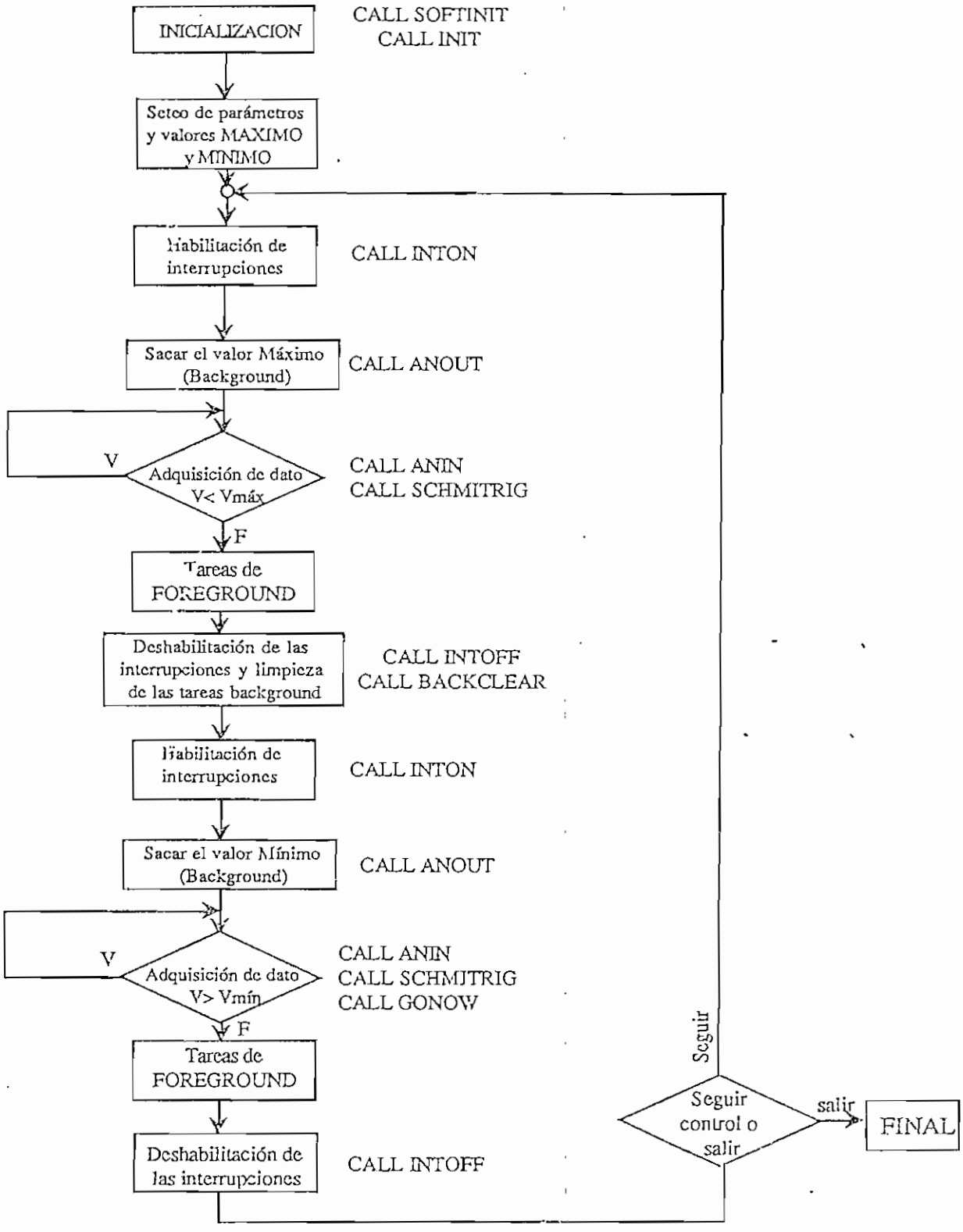


Figura 3.3.- Diagrama secuencial de tareas: Control ON-OFF mediante disparos.

LISTADO DE LA RUTINA.- En el listado del paquete de "software multitarea" que se presenta al final del capítulo, el ejemplo que realiza un control ON-OFF basado en disparos de background es la subrutina "ONOFDIS".

3.3.- GRAFICOS EN TIEMPO REAL CON QUICK500.- Graficar en tiempo real con los comandos del Quick500 representa una gran ventaja ya que permite planificar en un solo comando muchos parámetros en torno al gráfico, tales como canal o canales a graficarse, el arreglo de memoria donde se encuentran los datos, los colores del gráfico, tipo de pantalla, valores máximos y mínimos, escalas, resolución, unidades, etc..

Sin embargo existe una desventaja de peso que no permite utilizar con eficacia los comandos de graficación en tiempo real del Quick500, en condiciones normales toma demasiado tiempo graficar los datos, demorando muchas veces un tiempo mayor al período de muestreo, lo que provoca que se pierdan muchos datos en la graficación. Lo que sucede es que para utilizar adecuadamente las capacidades de graficación en tiempo real del Quick500 se necesita de un RTMDS software (Monitor de tiempo real y sistema directo de almacenamiento) y de una tarjeta acceleradorea de gráficos RTM, la misma que se instala entre el monitor CRT y el adaptador de gráficos existente, ocupando uno de los slots de expansión del computador. La tarjeta RTM tiene soporte para algunos tipos de monitores IBM compatibles.

El equipo con que se cuenta para el desarrollo del presente trabajo de tesis carece de dicha tarjeta aceleradora de gráficos en tiempo real, por lo que a ciertas frecuencias ya resulta impráctico graficar en tiempo real utilizando los comandos del Quick500, por esta razón se tomo la desición de hacer rutinas en BASIC para la graficación en tiempo real toda vez que se pudo conseguir tiempos de graficación menores y mejor resolución en pantalla. De todas maneras se incluye un ejemplo

de la graficación en tiempo real con los mencionados comandos, dicho ejemplo es un generador de ondas elemental.

COMANDOS ESPECIFICOS DEL QUICK500.- En este numeral se describen brevemente los comandos de gráficos en tiempo real con que cuenta el Quick500.

CALL GRAPH

PROPOSITO: La presente información cubre las versiones estándar y RTM del comando GRAPH. Para gozar de las ventajas de la tarjeta RTM, debe haberse instalado esta tarjeta, y además haberse modificado la instalación del Quick500. GRAPH es uua rutina de foreground, que grafica horizontalmente datos almacenados en un arreglo de memoria, sobre una base lineal de tiempo.

FORMATO: CALL GRAPH(arn$, widl$, coll$, displm$, miny!, maxy!, mrm$, res%, dep1!, dep2!, euf%)

PARAMETROS:

arn$ (strig-entrada).- Nombre del arreglo de datos accesado por el comando GRAPH, se pueden especificar subarreglos para la graficación.

widl$ (string-entrada).- Los coeficientes de anchuras incluids en este parámetro especifican que sub-áreas se van a graficar. Cada sub-área representa un canal de adquisición o salida de datos. Es una cadena de valores enteros que indican las anchuras de las sub-áreas que deben ser accesadas, estos valores van separados por comas y/o por espacios

coll$ (string-entrada).- Es una cadena de valores enteros separados por comas y/o espacios que representan colores:

	Paleta 0	Paleta 1
0	background	background
1	verde	cyan
2	rojo	magenta

3	café	blanco

displm$ (string-entrada).- Determina como se presentará el gráfico en pantalla, ya sea scroll, pagec, pageo.

"scroll"	Muestra la pantalla como un rollo de papel.
"pagec"	Cuando se llena la pantalla, es limpiada y sigue el gráfico desde el eje.
"pageo"	Cuando se llena la pantalla, sigue el gráfico desde el eje sin limpiar la pantalla.

miny! (real-entrada).- Determina el mínimo valor de la coordenada Y en el gráfico, puede expresarse en cualquier tipo de unidades.

maxy! (real-entrada).- Determina el máximo valor de la coordenada Y en el gráfico, puede expresarse en cualquier tipo de unidades. Maxy! debe ser mayor que miny! siempre.

dep1!, dep2! (reales-entradas).- Dos parámetros que deben usarse juntos, determinan una sub-área dentro de un arreglo de datos para su graficación. Para graficar todo el arreglo dep1!=1 y dep2! igual al fondo del arreglo. Dep2! debe ser siempre mayor que dep1!.

mrm$ (string-entrada).- Parámetro de magnificación o reducción del número de puntos (datos) graficados por GRAPH. Existen tres strings válidos: "magnify", "reduce" y "normal". El factor de ampliación o reducción está dado por el parámetro res%.

res% (integer-entrada).- Número entero que actúa junto al parámetro mrm$ para reducir o aumentar el número de puntos graficados. Los valores válidos oscilan entre 1-100. Se puede setear -1, en cuyo caso el gráfico será automáticamente ajustado a una ventana simple.

euf% (integer-entrada).- determina las unidades de maxy! y miny!. El 0 asigna voltios, mientras que el -1 asigna valores binarios para los parámetros antes mencionados.

CALL GRAPHRT

PROPOSITO: Rutina que permite graficar en tiempo real, uno o varios canales accesados por las siguientes rutinas de background: ANIN, ANOUT o DIGOUT.

FORMATO: CALL GRAPHRT(arn$ o bfn$, widl$, coll$, displm$, miny!, maxy!, npts!, euf%)

PARAMETROS:

arn$ (string-entrada),- Nombre del arreglo de memoria accesado por el comando GRAPHRT, se pneden abarcar varias sub-áreas (canales).

**bfn$** (string-entrada).- La rutina de background a ser graficada por el comando GRAPHRT, es llamada como tarea, nombre que se otorga a cada rutina de background en el parámetro bfn$, también existente en ellas.

widl$ (string-entrada).- Las anchuras incluidas en este parámetro, especifican, cuales sub-áreas van a ser graficadas. Es una cadena de valores enteros que indican las anchuras de las sub-áreas que deben ser accesadas, estos valores van separados por comas y/o por espacios

coll$ (string-entrada).- Es una cadena de valores enteros separados por comas y/o espacios que representan colores:

	Paleta 0	Paleta 1
0	background	background
1	verde	cyan
2	rojo	magenta
3	café	blanco

displm$ (string-entrada).- Determina como se presentará el gráfico en pantalla, ya sea scroll, pagec, pageo.

"scroll"	Muestra la pantalla como un rollo de papel.
"pagec"	Cuando se llena la pantalla, es limpiada y sigue el gráfico desde el eje.

"pageo"	Cuando se llena la pantalla, sigue el gráfico desde el eje sin limpiar la pantalla.

miny! (real-entrada).- Determina el mínimo valor de la coordenada Y en el gráfico, puede expresarse en cualquier tipo de unidades.

maxy! (real-entrada).- Determina el máximo valor de la coordenada Y en el gráfico, puede expresarse en cualquier tipo de unidades. Maxy! debe ser mayor que miny! siempre.

npts! (real-entrada).- Especifica el número de puntos que se grafican antes de que el comando GRAPHRT sea suspendido, si se asigna -1 graficará indefinidamente hasta ser suspendido.

euf% (integer-entrada).- determina las unidades de maxy! y miny!. El 0 asigna voltios, mientras que el -1 asigna valores binarios para los parámetros antes mencionados.

CALL GRLABEL

PROPOSITO: Comando que imprime texto cuando se tiene pantalla de gráficos de alta resolución, permitiendo tener textos en gráficos creados con el comando HGRAPHRT.

FORMATO: CALL GRLABEL(text$, win%, nwin%, loc1$, loc2$)

PARAMETROS:

text$ (string-entrada).- Puede consistir de un máximo de 64 caracteres en el plano horizontal y de 21 caracteres en el plano vertical para una ventana, 10 para 2 ventanas y 6 para tres ventanas.

win$ (integer-entrada).- Especifica el número de ventana a la que se quiere asignar el texto, existen tres valores válidos: 1.- ventana superior; 2.- ventana central (inferior si se tiene sólo dos ventanas); 3.- ventana inferior.

nwin$ (integer-entrada).- Se le puede asignar los valores 1, 2 o 3, lo cual indica el número de ventanas que se están utilizando. Este número debe ser igual al parámetro wind% del comando HGRAPHRT.

loc1$ (string-entrada).- Determina la ubicación del texto dentro de la ventana de gráfico. Existen tres posibilidades: "left" izquierda de la ventana y alineción vertical, "top" parte superior de la ventana y alineación horizontal y "bottom" parte inferior de la pantalla y alineación horizontal.

loc2$ (string-entrada).- Una vez que se ha ubicado el texto, loc2$ permite justificar el texto, así si se tiene alineación horizontal tenemos justificación a la derecha "right", a la izquierda "left" o centrado "crt". Si se tiene alineación vertical se puede justificar arriba "top", abajo "bottom" o centrado "ctr". En definitiva la combinación de loc1$ y loc2$ determinan la ubicación del texto.

CALL HGRAPHRT

PROPOSITO: La siguiente información abarca las versiones estándar y RTM del comando HGRAPHRT. Este comando es una rutina de foreground que realiza gráficos de alta resolución en tiempo real de uno o varios canales de entrada o salida accesados por las rutinas ANIN, ANOUT o DIGOUT de background. Para utilizar este comando se debe antes ubicar la sentencia SCREEN 2, al final se debe tener SCREEN 0 para abandonar el modo gráfico.

FORMATO: CALL GRAPHRT(arn$ o bfn$, widl$, displm$, minyl!, maxyl!, eufl% npts!, wind%, grid$)

PARAMETROS:

arn$ (string-entrada),- Nombre del arreglo accesado por el comando HGRAPHRT, se pueden especificar varias sub-áreas.

bfn$ (string-entrada).- La rutina de background a ser graficada por el

comando GRAPHRT, es llamada como tarea, nombre que se otorga a cada rutina de background en el parámetro bfn$, también existente en ellas.

widl$ (string-entrada).- Es una cadena de valores enteros que indica las posiciones dentro de la lista de IONAMES de los canales a ser graficados, estos valores van separados por comas y/o por espacios.

displm$ (string-entrada).- Determina como se presentará el gráfico en pantalla, ya sea scroll, pagee, pageo.

"scroll"	Muestra la pantalla como un rollo de papel.
"pagee"	Cuando se llena la pantalla, es limpiada y sigue el gráfico desde el eje.
"pageo"	Cuando se llena la pantalla, sigue el gráfico desde el eje sin limpiar la pantalla.

minyl! (real-entrada).- Determina la lista de valores mínimos de la coordenada Y en el gráfico, puede expresarse en cualquier tipo de unidades.

maxyl! (real-entrada).- Determina la lista de valores máximos de la coordenada Y en el gráfico, puede expresarse en cualquier tipo de unidades. Maxyl! debe ser mayor que minyl! respectivamente.

eufl% (integer-entrada).- determina las unidades de maxyl! y minyl!. El 0 asigna voltios, mientras que el -1 asigna valores binarios para los parámetros antes mencionados.

npts! (real-entrada).- Especifica el número de puntos que se grafican antes de que el comando GRAPHRT sea suspendido, el -1 graficará de manera indefinida.

wind% (integer-entrada).- Un valor entero indica el número de ventanas que se quiere graficar. Los valores válidos son 1, 2 o 3; es decir se puede tener hasta 3 ventanas.

grid$ (string-entrada).- Permite escoger si se quiere o no tener grilla o cuadrícula en el gráfico. Los strings válidos son: "grid" y "nogrid".

DIAGRAMA SECUENCIAL DE TAREAS.- En la figura 3.5 se presenta el diagrama secuencial de tareas de la rutina de un generador de ondas, el mismo que grafica en tiempo real en pantalla, las ondas que son sacadas al exterior por los canales de salida análoga.

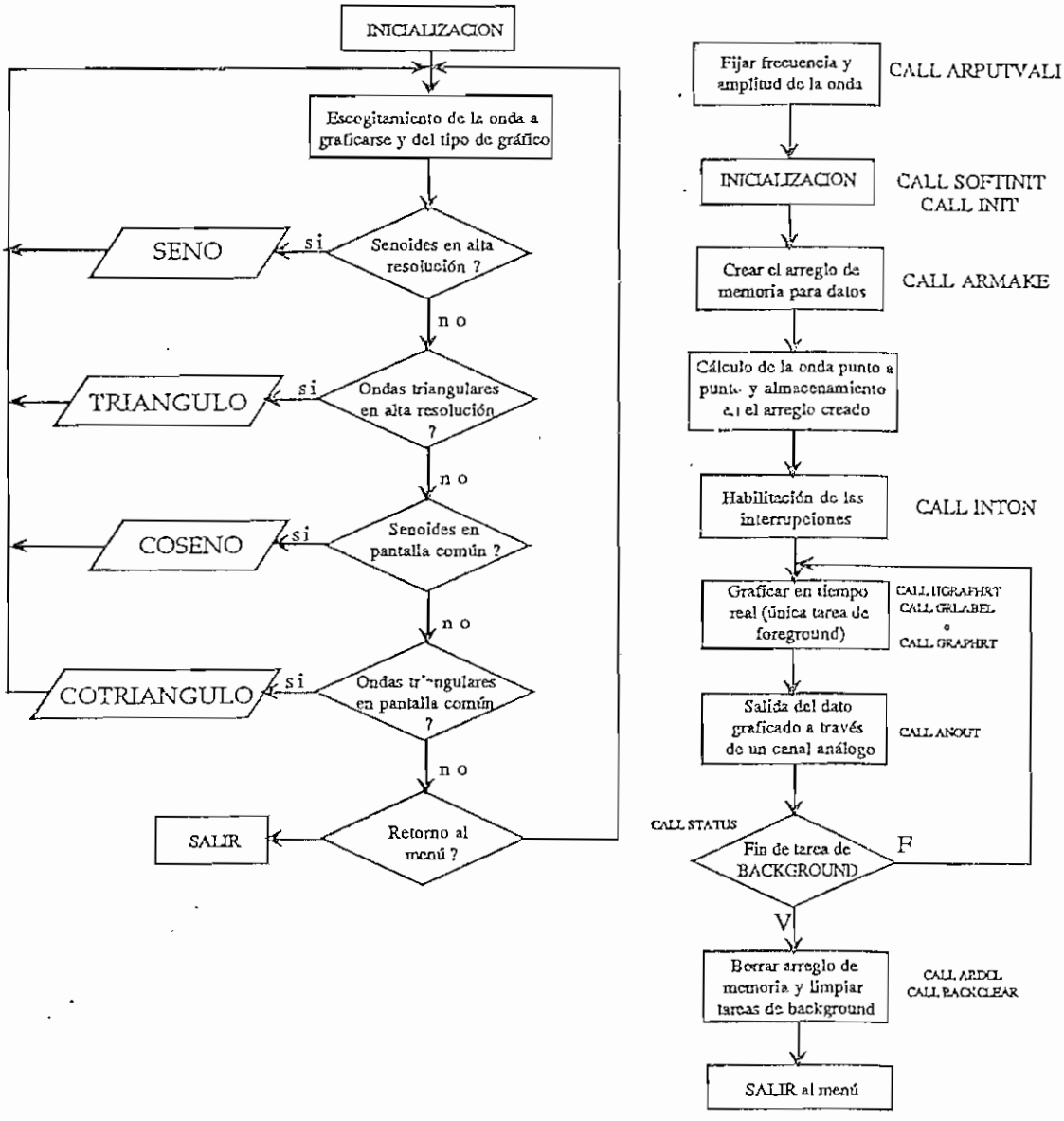


Figura 3.5.- Diagrama secuencial de tareas: Graficación en tiempo real: A la izquierda el menú y a la derecha la estructura de cada rutina de gráfico.

LISTADO DE LA RUTINA.- En el listado del paquete de "software multitarea" que se presenta al final del capítulo, el ejemplo del generador de ondas con graficación en tiempo real utilizando los comandos del Quick500 es la subrutina "GRAF500".

3.4.- MANEJO DE ARREGLOS DE MEMORIA PARA DATOS.-

Los arreglos de memoria para datos son herramientas indispensables para el trabajo con la estación KEITHLEY 500A, cuando se trabaja en background. Estos arreglos que son creados por los comandos ANIN o ARMAKE sirven para almacenar los datos que se han adquirido o que se sacan a través de los canales análogos o digitales de entrada o salida.

Así, cuando se ha adquirido datos, es importante ubicar estos datos en un arreglo de memoria, para lo cual se tiene los comandos ARPUTF, ARPUTI, ARPUTVALF o ARPUTVALI; cuando se tiene salida de datos en tiempo real se debe tener las ondas almacenadas en un arreglo de este tipo y para recuperar estos valores antes de sacarlos a través de los canales de salida se utiliza los comandos ARGETVALF, ARGETVALI, ARGETF o ARGETI, de igual manera se puede obtener información del arreglo como profundidad, anchura, último dato accesado. Esta última información es de gran importancia ya que ayuda sobre manera en los procesos de tiempo real como se ha visto ya en todos los ejemplos de background presentados en el capítulo 2.

Pero además es importante poder guardar estos datos en disco para posteriores análisis o para obtención de resultados gráficos en papel mediante la utilización de hojas electrónicas por ejemplo y por lo tanto también poder recuperar datos guardados en memoria de disco para su utilización en procesos de

tiempo real, así por ejemplo se pueden adquirir en tiempo real ondas especiales, guardarlas para luego utilizarlas en procesos que necesiten de dichas ondas.

Para este numeral se ha desarrollado un ejemplo que adquiere señales en tiempo real, y almacena estos datos en un arreglo de memoria, al terminar la adquisición de datos salva esta información en disco. Luego se pide información completa del arreglo de memoria y se borra de memoria el arreglo de datos creado. Posteriormente se recupera la información guardada en disco con un nombre diferente para el arreglo de memoria, se pide nuevamente información del arreglo para observar las diferencias y similitudes.

Se debe indicar que mediante la utilización del comando ARWRITE, se ha obtenido todos los resultados gráficos que se presentan en el capítulo 4, guardando la información en formato importable a LOTUS 123 (Posteriormente estos datos fueron transferidos a EXCEL para Macintosh para su graficación).

COMANDOS ESPECIFICOS DEL QUICK500.- Algunos de los comandos referentes al manejo de arreglos de memoria ya han sido explicados, por lo que en este numeral se describe brevemente los restantes comandos.

CALL ARLASTP

PROPOSITO: Reporta la ubicación (profundidad) del último dato de un arreglo accesado por una rutina de background. Si ARLASTP se ejecuta sobre un arreglo no accesado por una rutina de background, el valor reportado será cero.

FORMATO: CALL ARLASTP(arn$, lp!)

PARAMETROS:

arn$ (string-entrada).- Parámetro que contiene el nombre del arreglo Quick500, del cual se reporta el último dato accesado. Dicho arreglo debe estar

residente en memoria al ejecutarse el comando ARLASTP.

lp! (real-salida).- ARLASTP ubica en este parámetro el índice de profundidad del último dato accesado por una rutina de background del Quick500. Nótese que en este parámetro no se ubica el dato sino su ubicación.

CALL ARSTATUS

PROPOSITO: Reporta información variada acerca de un arreglo de memoria Quick500 para datos, incluyendo profundidad, anchura, último dato accesado e incluso una descripción del arreglo.

FORMATO: CALL ARSTATUS(arn$, dep!, wid%, lp!, lab$)

PARAMETROS:

arn$ (string-entrada).- Parámetro que contiene el nombre del arreglo Quick500, del cual se reporta la información. Dicho arreglo debe estar residente en memoria al ejecutarse el comando ARSTATUS.

dep! (real-salida).- Reporta el tamaño o profundidad del arreglo de memoria del cual se reporta la información.

wid% (integer-salida).- Reporta el ancho (número de canales) del arreglo de memoria.

lp! (real-salida).- ARLASTP ubica en este parámetro el índice de profundidad del último dato accesado por una rutina de background del Quick500.

lab$ (string-salida).- En este parámetro se reporta un nivel descriptivo dado al arreglo de memoria mediante el comando ARLABEL. Se guarda como una variable alfanumérica BASIC para facilidad de acceso.

CALL ARLABEL

PROPOSITO: Permite hacer una descripción de un arreglo de memoria de datos Quick500. Dicha descripción será salvada o recuperada con el arreglo.

FORMATO: CALL ARLABEL(arn$, lab$)

PARAMETROS:

arn$ (string-entrada).- Parámetro que contiene el nombre del arreglo Quick500, del cual se realiza la descripción. Dicho arreglo debe estar residente en memoria al ejecutarse el comando ARLABEL.

lab$ (string-entrada).- Este parámetro recibe una cadena de hasta 255 caracteres para hacer la descripción de un arreglo de memoria de datos del Quick500.

CALL ARSAVE

PROPOSITO: Permite al usuario guardar la información en diskette o en disco duro, la única limitación es el monto de memoria libre en disco. Es una rutina de singleground y se puede ejecutar incluso cuando las interrupciones están habilitadas.

FORMATO: CALL ARSAVE(arn$, fn$)

PARAMETROS:

arn$ (string-entrada).- Parámetro que contiene el nombre del arreglo Quick500 que será salvado en disco cuando el comando ARSAVE sea ejecutado. Dicho arreglo debe estar residente en memoria al ejecutarse el comando ARSAVE, caso contrario se recibirá un mensaje de error.

fn$ (string-entrada).- Permite al usuario dar un nombre DOS estándar al archivo creado con la ejecución del comando ARSAVE. El nombre asignado se sumará al directorio del disco y será tratado como cualquier archivo DOS.

CALL ARWRITE

PROPOSITO: Salva la información de un arreglo de datos en diskette o disco duro

en diferentes formatos. El tamaño del arreglo salvado depende únicamente del monto de memoria disponible en disco. ARWRITE hace fácil exportar datos a hojas electrónicas, procesadores de palabras, bases de datos y programas de análisis. Los datos se salvan en formato ASCII.

FORMATO: CALL ARWRITE(arn$, fn$, euf%, ftype%, sr%, tunits%)

PARAMETROS:

arn$ (string-entrada).- Parámetro que contiene el nombre del arreglo Quick500 que será salvado en disco cuando el comando ARWRITE sea ejecutado. Dicho arreglo debe estar residente en memoria al ejecutarse el comando ARWRITE, caso contrario se recibirá un mensaje de error.

fn$ (string-entrada).- Permite al usuario dar un nombre DOS estándar al archivo creado con la ejecución del comando ARWRITE. El nombre asignado se sumará al directorio del disco y será tratado como cualquier archivo DOS.

euf% (string-entrada).- Determina las unidades con que serán salvados los datos. El 0 asigna voltios. Si se asigna -1, los datos se salvan como binarios.

ftype% (integer-entrada).- indica el tipo de archivo deseado:

0	Mismo formato que ARSAVE
1	Archivo ASCII
2	Archivo binario con 16 bytes header
3	Binario con DADiSP readable header
4	ASCII importable a LOTUS como archivo .PRN
5	Binario exportable a ASYST/ASYSTANT.

sr%.- Frecuencia a la cual se va a recolectar los datos. Se debe setear este parámetro como sr%= ir%*bintv% (del conamdo INTON y las tareas de background respectivamente). El valor de sr% crea una columna de datos con la base de tiempo.

tunits% (integer-entrada).- Unidades de tiempo utilizadas en la

recolección de datos. Así: 0.- centenas de us, 1.- ms, 2.- segundos y 3.- minutos.

CALL ARLOAD

PROPOSITO: Recupera un arreglo especificado desde disco a memoria, dicho arreglo debió haber sido grabado anteriormente con el comando ARSAVE. Es una rutina de singleground por lo que puede ejecutarse incluso cuando las interrupciones están habilitadas.

FORMATO: CALL ARLOAD(arn$, fn$)

PARAMETROS:

arn$ (string-entrada).- Parámetro que contiene el nombre del arreglo Quick500 que se recupera a memoria cuando el comando ARLOAD es ejecutado. El archivo DOS debe haber sido creado mediante la ejecución del comando ARSAVE.

fn$ (string-entrada).- Nombre DOS estándar asignado cuando se ejecuta el comando ARSAVE. El nombre asignado se sumará al directorio del disco y será tratado como cualquier archivo DOS.

CALL ARDEL

PROPOSITO: Borra los arreglos de memoria de datos del Quick500. ARDEL puede utilizarse para liberar bloques de memoria previamente ocupadas.

FORMATO: CALL ARDEL(arn$)

PARAMETROS:

arn$ (string-entrada).- Parámetro que contiene el nombre del arreglo Quick500 que será eliminado de memoria cuando el comando ARDEL sea ejecutado. El parámetro arn$ debe referirse a un arreglo creado previamente sea por ARMAKE o por ANIN. Si el usuario quiere borrar de memoria un arreglo inexistente, recibirá un mensaje de error.

DIAGRAMA SECUENCIAL DE TAREAS.- En la figura 3.6 se presenta el diagrama secuencial de tareas de la rutina de manejo de arreglos de memoria.

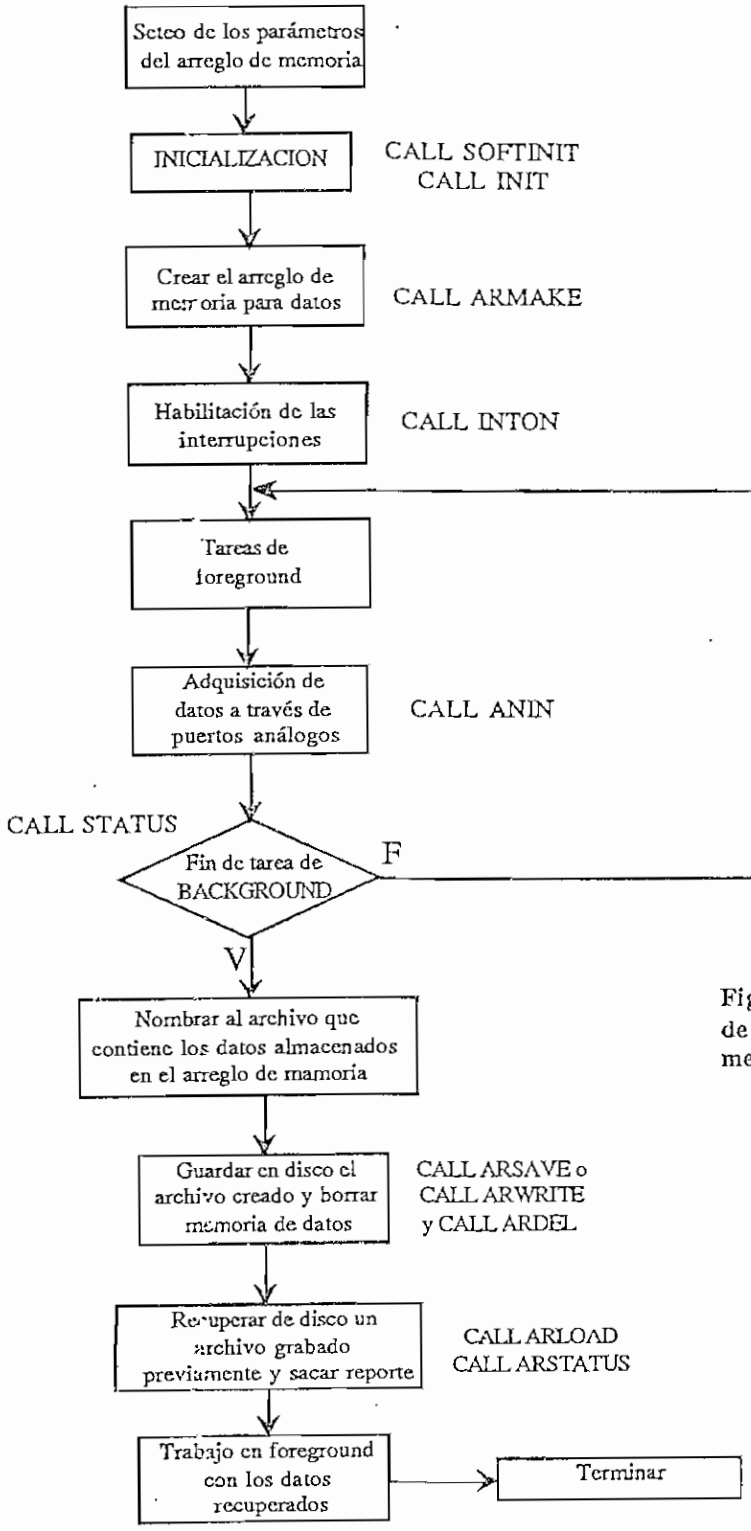


Figura 3.6.- Diagrama secuencial de tareas: Manejo de arreglos de memoria de datos.

LISTADO DE LA RUTINA.- En el listado del paquete de "software multitarea" que se presenta al final del capítulo, el ejemplo de manejo de arreglos de memoria para datos es la subrutina "AREAS."

3.5.- PROCESAMIENTO DE DATOS.-

Cuando se han recopilado datos y estos son almacenados en arreglos de memoria para datos en el formato Quick500, se los puede procesar para obtener informaciones adicionales. Así, se puede obtener la derivada de una señal adquirida en un punto determinado, la integral de un grupo de datos, media aritmética, desviación estándar, ecuación de una recta aproximada mediante regresión lineal, etc..

Estas son las posibilidades que han sido consideradas en el Quick500, sin embargo mediante rutinas que se pueden desarrollar en BASIC se puede obtener mayor información. Se ha desarrollado un ejemplo en el que se adquiere una serie de datos que son almacenados en un arreglo de memoria. Sobre dichos datos se obtiene la derivada en un punto determinado por el usuario, se obtiene la integral de un sector de los datos adquiridos, se obtiene además la media aritmética, desviación estándar y una recta aproximada de los datos adquiridos.

COMANDOS ESPECIFICOS DEL QUICK500.- Todos los comandos que realizan procesamiento de datos son descritos brevemente a continuación.

CALL DERIVATIVE

PROPOSITO: Es una rutina de foreground para análisis de datos, ejecutado sobre un arreglo específico de datos y entrega la derivada (pendiente) de una curva en un punto particular.

FORMATO: CALL DERIVATIVE(arr$, dep!, wid%, deriv!, euf%)

PARAMETROS:

arn$ (string-entrada).- Especifica el arreglo de memoria accesado por el comando DERIVATIVE, y se debe referir a una arreglo creado anteriormente.

dep! (real-entrada).- Indica la profundidad del punto sobre el cual se toma la derivada. Se le puede asignar valores desde 1 hasta la profundidad máxima del arreglo.

wid% (integer-entrada).- Especifica la sub-área en la cual se ejecuta el comando DERIVATIVE. Cada valor de wid% corresponde a un canal de entrada o de salida, el comando se ejecuta sobre un solo canal.

deriv! (real-salida).- Reporta la derivada de la curva de datos en el punto especificado por dep! y wid%.

euf% (integer-entrada).- Utilizado para especificar las unidades del parámetro deriv!.

CALL INTEGRAL

PROPOSITO: Rutina de foreground para análisis de datos, al ser ejecutada sobre un arreglo Quick500 reporta la integral de una curva (el área bajo la curva) entre dos puntos específicos. Siempre se ejecuta sobre un solo canal de entrada o salida. La integral es calculada de acuerdo a la regla de Simpson de integración.

FORMATO: CALL INTEGRAL(arn$, wid%, integ!, dep1!, dep2!, euf%)

PARAMETROS:

arn$ (string-entrada).- Especifica el arreglo de memoria accesado por el comando INTEGRAL, y se debe referir a una arreglo creado anteriormente.

wid% (integer-entrada).- Especifica la sub-área en la cual se ejecuta el comando INTEGRAL. Cada valor de wid% corresponde a un canal de entrada o de salida, el comando se ejecuta sobre un solo canal.

integ! (real-salida).- Entrega el área bajo la curva de datos (con la totalidad o parte de los datos que conforman el arreglo).

dep1!, dep2! (reales-entradas).- Indican los límites inferior y superior respectivamente del sub-área de datos sobre la cual se toma la integral por parte del comando INTEGRAL.

euf% (integer-entrada).- Utilizado para especificar las unidades en que se expresa el parámetro integ!.

CALL MEANDEV

PROPOSITO: Rutina de foreground para análisis de datos, al ser ejecutada sobre un arreglo Quick500 determinado se obtiene la media aritmética y desviación estándar de un grupo especificado de datos.

FORMATO: CALL MEANDEV(arn$, wid%, mean!, stdev!, dep1!, dep2!, euf%)

PARAMETROS:

arn$ (string-entrada).- Especifica el arreglo de memoria accesado por el comando MEANDEV, y se debe referir a una arreglo creado anteriormente.

wid% (integer-entrada).- Especifica la sub-área en la cual se ejecuta el comando MEANDEV. Cada valor de wid% corresponde a un canal de entrada o de salida, el comando se ejecuta sobre un solo canal.

dep1!, dep2! (reales-entradas).- Indican los límites inferior y superior respectivamente del sub-área de datos sobre la cual se realizan los cálculos por parte del comando MEANDEV.

mean! (real-salida).- Entrega la media aritmética de todos los datos encerrados dentro del sub-área delimitada por dep1! y dep2!.

stdev! (real-salida).- Entrega la desviación estándar del grupo de datos escogido, se calcula tomando el cuadrado de las desviaciones respecto de la media

aritmética (para cada punto) y dividiendo esta suma para el número de datos menos uno.

euf% (integer-entrada).- Utilizado para especificar las unidades en que se expresan los parámetros mean! y stdev!.

CALL LINREGR

PROPOSITO: Ejecuta una regresión lineal sobre los datos de dos sub-áreas, o de una sub-área contra el índice de fondo (en otras palabras, versus tiempo). Una regresión lineal produce una ecuación de la forma y = mx + b, que describe la relación lineal entre dos variables.

FORMATO: CALL LINREGR(arn$, wid1%, wid2%, m!, b!, det!, corr!, sterr!, dep1!, dep2!, euf%)

PARAMETROS:

arn$ (string-entrada).- Especifica el arreglo de memoria accesado por el comando MEANDEV, y se debe referir a una arreglo creado anteriormente.

wid1% (integer-entrada).- Especifica una de las dos sub-áreas en la cual se ejecuta el comando LINREGR. Esta primera sub-área considera los datos usados para calcular el valor x en la predicción de la ecuación y = mx + b. En un caso especial se puede setear el valor 0, en cuyo caso se toma el índice de fondo (base de tiempo) en la primera sub-área.

wid2% (integer-entrada).- Especifica la segunda de las dos sub-áreas en la cual se ejecuta el comando LINREGR. Esta segunda sub-área considera los datos usados para calcular el valor y en la predicción de la ecuación y = mx + b. En un caso especial se puede setear el valor 0, en cuyo caso se toma el índice de fondo (base de tiempo, entonces t = mx + b) en la primera sub-área.

m! (real-salida).- En este parámetro se recupera la pendiente de la ecuación y = mx + b.

b! (real-salida).- Este parámetro recupera la intersección de la recta aproximada con el eje y, en la ecuación y = mx + b, corresponde al valor de b.

det! (real-salida).- Corresponde al coeficiente de determinación, este valor oscila entre 0 y 1, es el cuadrado del coeficiente de correlación.

corr! (real-salida).- El coeficiente de correlación de la regresión lineal se reporta en este parámetro. El conocimiento de este valor entrega información de la linealidad de la relación entre x y y. El valor de corr! varía entre -1 y 1. Mientras más se aproxime el valor obtenido a 1, más lineal es la relación. Un coeficiente 1 o -1 indica una linealidad perfecta.

sterr! (real-salida).- El error estándar de la estimación se ubica en este parámetro al ejecutarse el comando LINREGR.

dep1!, dep2! (reales-entradas).- Indican los límites inferior y superior respectivamente del sub-área de datos sobre la cual se realizan los cálculos por parte del comando MEANDEV.

euf% (integer-entrada).- Utilizado para especificar las unidades en que se expresan los parámetros mean! y stdev!. Cuando se asigna un 0 se tiene los resultados en voltios.

DIAGRAMA SECUENCIAL DE TAREAS.- En la figura 3.7 se presenta el diagrama secuencial de tareas de la rutina de procesamiento de datos, en la cual se utiliza todos los comandos que se han descrito en el presente numeral.

LISTADO DE LA RUTINA.- En el listado del paquete de "software multitarea" que se presenta al final del capítulo, el ejemplo de manejo de procesamiento de datos es la subrutina "PROCESAMIENTO".

3.6.- LA TRANSFORMADA RAPIDA DE FOURIER.-

Al empezar este estudio se muestra la ecuación que define la transformada de Fourier continua directa:

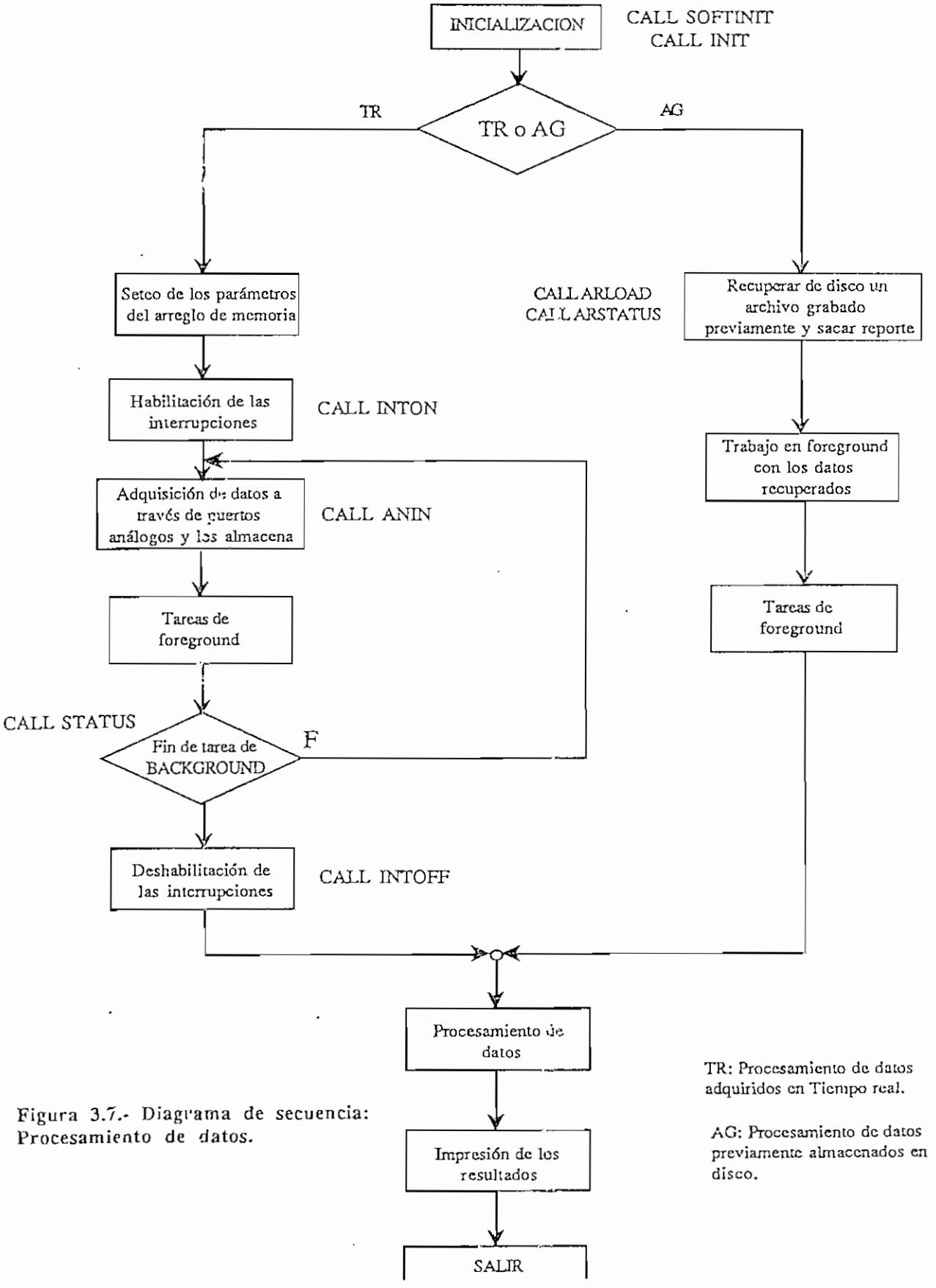


Figura 3.7.- Diagrama de secuencia: Procesamiento de datos.

TR: Procesamiento de datos adquiridos en Tiempo real.

AG: Procesamiento de datos previamente almacenados en disco.

$$f(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} F(\omega) . e^{j\omega t} d\omega \qquad (1)$$

En función de la frecuencia $f=\omega/2\Pi$

$$f(t) = \int_{-\infty}^{\infty} F(f) . e^{j2\pi ft} . df \qquad (2)$$

Para encontrar la transformada discreta de Fourier es necesario considerar una señal f(t) de *ancho de banda* limitado y asumir que la transformada de Fourier de f(t) es F(ω), lo que permite realizar un gráfico hipotético de ambas funciones (figura 3.8).

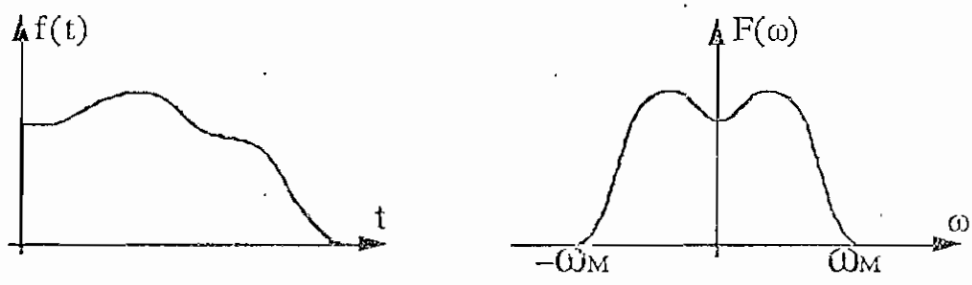


Figura 3.8.- Gráfico de una función en el tiempo y su transformada de Fourier.

El teorema de muestreo dice que la información contenida en f(t) se conserva íntegra en sus muestras tomadas a frecuencias igual o mayores que 2 veces la frecuencia máxima contenida en f(t).

Ya que un computador no puede manejar un número infinito de cantidades, no se puede tabajar con las fórmulas anteriores y se recurre a la transformada discreta de Fourier, en la cual un determinado número de muestras se transforma en un número igual de coeficientes en el dominio de la transformada. Veamos gráficamente lo dicho (figura 3.9):

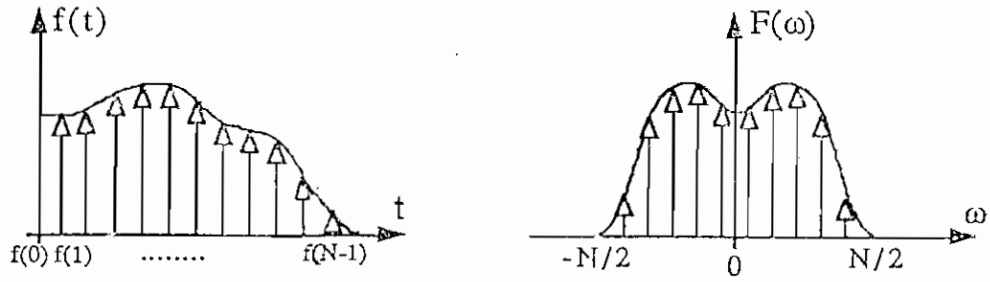


Figura 3.9.- Discretización de la función f(t) y su transformada de Fourier.

Mediante procesos matemáticos se llega a las ecuaciones que definen la transformada discreta de Fourier directa e inversa, las mismas que se escriben en forma matricial (La transformada rápida de Fourier.- Tesis, Marcelo Rodas):

$$[f(k)] = [W^{-nk}].[F(n)] \qquad (3)$$

$$[F(n)] = [W^{nk}].[f(k)] \qquad (4)$$

Donde [F(n)] y [f(k)] representan matrices columna *nx1* , mientras que $[W^{-nk}]$ y $[W^{nk}]$ son matrices cuadradas *nxn* y se llaman matrices de Fourier o de transformación y representan la base para el algoritmo de la transformada rápida de Fourier.

CARACTERISTICAS DE LAS MATRICES DE FOURIER.-

a) Todos los elementos de las matrices son raices n-ésimas de la unidad:

$$\sqrt[N]{1} = \cos\left(2\pi k/N\right) - j\,\mathrm{sen}\left(2\pi k/N\right)$$

como $W = e^{-j2\pi/n}$ entonces $\sqrt[N]{1} = W^{k}$ k=0, 1, 2, 3, ..., N-1.

b) Las raices n-ésimas de la unidad se distribuyen simétricamente en una circunferencia de radio 1 en el plano complejo. Como norma es conveniente

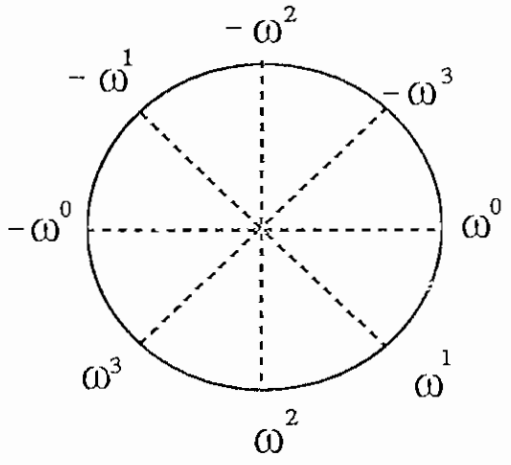
enumerar las raices a partir del eje real y girando en el sentido horario.

c) El orden de las matrices de Fourier es siempre igual a una potencia de 2.

d) Todos los elementos se ordenan en la matriz, de manera que cada columna o fila sea ortogonal respecto a las demás.

e) La matriz de Fourier o de transformación [TF] es N-unitaria, es decir que el producto de [TF] por su conjugada y transpuesta es N[In] , razón por la cual siempre aparece en las fórmulas en factor 1/N.

OBTENCION DE LAS MATRICES DE FOURIER.- Un ejemplo para N=8 para ilustrar su distribución en la circunferencia de radio 1 es:



Estas raices se deben distribuir en la matriz de forma que todo vector sea mutuamente ortogonal, mediante la fórmula:

$$[TF]=[w^{<k>l}] \qquad (5)$$

l = número de orden de la columna .

<k> = número decimal que se obtiene a partir del número de fila respectiva de la siguiente forma: Se escribe el número de orden de la fila en binario, con un número de bits igual al logaritmo en base 2 del número de muestras, luego se invierte este número binario y se obtiene el número decimal equivalente, este valor es <k>, se presenta un ejemplo para 8 muestras:

NUMERO DE FILA	NUMERO BINARIO CON 3 BITS	NUMERO BINARIO INVERTIDO	VALOR DE <K>
0	000	000	0
1	001	100	4
2	010	010	2
3	011	110	6
4	100	001	1
5	101	101	5
6	110	011	3
7	111	111	7

Obtenidos los valores de <k>, se procede a obtener la matriz de Fourier mediante la ecuación (5).

FACTORIZACION DE LAS MATRICES DE FOURIER.- Para transformar [TF] en un producto de submatrices o matrices factor, cuyo número es $n=\log_2 N$. En el ejemplo propuesto si N=8, entonces n=3.

$$[TF(8)] = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i & 1 & -i & -1 & i \\ 1 & i & -1 & -i & 1 & i & -1 & -i \\ 1 & w & w^2 & w^3 & -1 & -w & -w^2 & -w^3 \\ 1 & -w & w^2 & -w^3 & -1 & w & -w^2 & w^3 \\ 1 & w^3 & -w^2 & w & -1 & -w^3 & w^2 & -w \\ 1 & -w^3 & -w^2 & -w & -1 & w^3 & w^2 & w \end{bmatrix} \quad (6)$$

Se recurre a la propiedad de partición de las matrices, entonces [TF(8)] es:

$$[TF(8)] = \begin{bmatrix} A1 & A1 \\ A2 & -A2 \end{bmatrix} \quad \text{donde;}$$

$$A1 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \\ 1 & i & -1 & -i \end{bmatrix} \quad ; A2 = \begin{bmatrix} 1 & w & w^2 & w^3 \\ 1 & -w & w^2 & -w^3 \\ 1 & w^3 & -w^2 & w \\ 1 & -w^3 & -w^2 & -w \end{bmatrix}$$

Entonces (6) se transforma en: $\begin{bmatrix} A1 & 0 \\ 0 & A2 \end{bmatrix} * \begin{bmatrix} I4 & I4 \\ I4 & -I4 \end{bmatrix}$

donde I4 es la matriz identidad de 4x4. Sobre A1 se puede hacer una nueva partición:

$$A1= \begin{bmatrix} A3 & A3 \\ A4 & -A4 \end{bmatrix} = \begin{bmatrix} A3 & 0 \\ 0 & A4 \end{bmatrix} \begin{bmatrix} I2 & I2 \\ I2 & -I2 \end{bmatrix}$$

De igual manera sobre la matriz A2:

$$A2= \begin{bmatrix} A5 & A5W^2 \\ A6 & -A6W^2 \end{bmatrix} = \begin{bmatrix} A5 & 0 \\ 0 & A6 \end{bmatrix} \begin{bmatrix} I2 & I2W^2 \\ I2 & -I2W^2 \end{bmatrix}$$

donde: $A3= \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$ $A4= \begin{bmatrix} 1 & -i \\ 1 & i \end{bmatrix}$ $A5= \begin{bmatrix} 1 & W \\ 1 & -w \end{bmatrix}$ $A6= \begin{bmatrix} 1 & W^3 \\ 1 & -w^3 \end{bmatrix}$

Ahora se tiene la matriz [TF(8)] factorada en tre matrices factor, las mismas que son de gran redundancia, lo que reduce considerablemente el número de multiplicaciones (tiempo de computador). Estas matrices factor nos llevan directamente al algoritmo para la obtención de la transformada rápida de Fourier.

ALGORITMO RAPIDO PARA EL CALCULO DE LA TRANSFORMADA DISCRETA DE FOURIER.- Este algoritmo hace posible el cálculo computacional de la transformada discreta de Fourier, en menor tiempo que cualquier otro algoritmo. Se basa en la obtención directa de las matrices factor, es decir sin necesidad de factorización, ya que a base de estas se realiza un diagrama de flujo.

PROCEDIMIENTO: 1.- La primera matriz factor está constituida por N/2 submatrices de la forma:

$$\begin{bmatrix} 1 & W^{<k>} \\ 1 & -w^{<k>} \end{bmatrix}$$

Las que forman la diagonal principal de la primera matriz factor, siendo el resto de elementos = 0; siguiendo con el ejemplo de 8 muestras, los valores de <k> para N/2 son 0, 2, 1, 3 como se muestra en la ecuación (7).

2.- La segunda matriz factor, se obtiene realizando el producto de Kronecker de las N/4 primeras submatrices de la primera matriz factor por la matriz unitaria I2, obteniéndose N/4 submatrices que van a conformar la diagonal principal, el resto de valores son 0. Para N=8 se realiza el producto de Kronecker de las N/4, osea 2 primeras submatrices de la primera matriz factor por I2.

3.- La tercera matriz factor, se obtiene realizando el producto de Kronecker (o) de las N/8 primeras submatrices de la segunda matriz factor por I2. La cuarta sería las N/16 primeras submatrices de la tercera matriz factor, y así sucesivamente, hasta tener el número de factores necesarios.

Existe un procedimiento alternativo válido desde la tercera matriz factor en adelante: para la tercera matriz factor se realiza el producto de las N/8 primeras submatrices de la primera matriz factor por I4, la cuarta se obtiene del producto de las primeras N/16 submatrices de la primera matriz factor por I8 y así sucesivamente, obteniéndose resultados idénticos.

$$[TF(8)]= \begin{bmatrix} 1 & 1 & & & & & & \\ 1 & -1 & & & & & & \\ & & 1 & w2 & & & & \\ & & 1 & -w2 & & & & \\ & & & & 1 & w & & \\ & & & & 1 & -w & & \\ & & & & & & 1 & w3 \\ & & & & & & 1 & -w3 \end{bmatrix} \begin{bmatrix} 1 & 0 & 1 & 0 & & & & \\ 0 & 1 & 0 & 1 & & & & \\ 1 & 0 & -1 & 0 & & & & \\ 0 & 1 & 0 & -1 & & & & \\ & & & & 1 & 0 & w2 & 0 \\ & & & & 0 & 1 & 0 & w2 \\ & & & & 1 & 0 & -w2 & 0 \\ & & & & 0 & 1 & 0 & -w2 \end{bmatrix} \begin{bmatrix} 1 & 0 & & & 1 & 0 & & \\ 0 & 1 & & & 0 & 1 & & \\ & & 1 & 0 & & & 1 & 0 \\ & & 0 & 1 & & & 0 & 1 \\ 1 & 0 & & & -1 & 0 & & \\ 0 & 1 & & & 0 & -1 & & \\ & & 1 & 0 & & & -1 & 0 \\ & & 0 & 1 & & & 0 & -1 \end{bmatrix}$$

(7)

La transformada rápida de Fourier de un vector [X] de 8 muestras se define :

$$[XT] = 1/8\ [TF(8)].[X] \qquad (8)$$

Finalmente se obtiene la ecuación (9), en la cual se multiplica columnas por filas en lugar de filas por columnas:

$$[X].[3^{o}MF].[2^{o}MF].[1^{o}MF].1/8 = [XT] \qquad (9)$$

Donde [XT] es el vector transformado. A partir de esta última ecuación se puede obtener un diagrama de flujo, a base del cual se realiza una subrutina para calcular la *Transformada rápida de Fourier.*

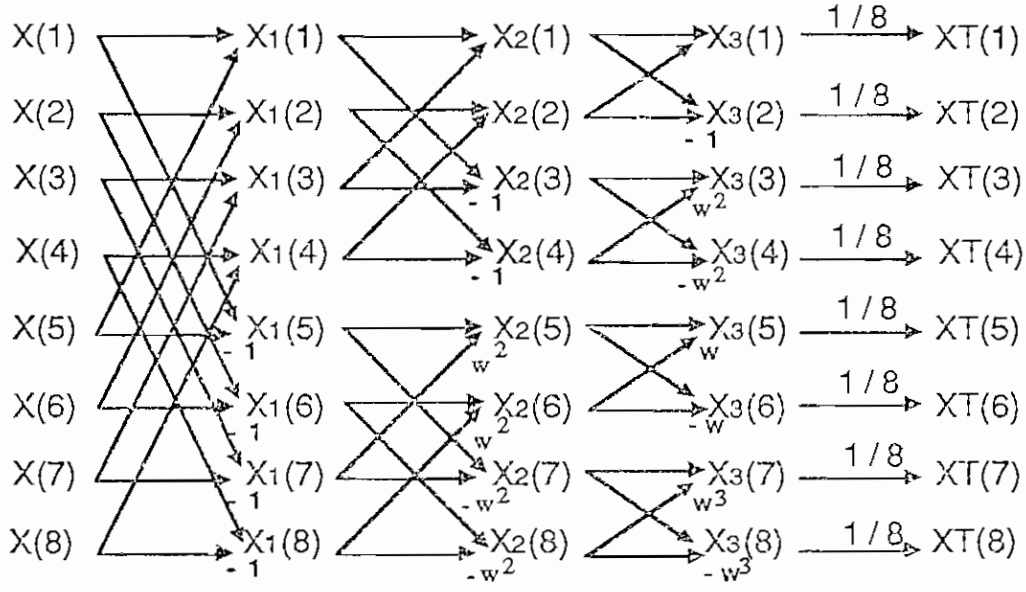


Figura 3.10.- Algoritma para la obtención de la transformada rápida de Fourier.

DIAGRAMA SECUENCIAL DE TAREAS.- En la figura 3.11 se presenta el diagrama secuencial de tareas de la rutina en la que se obtiene la transformada rápida de Fourier de una serie de datos adquiridos en tiempo real.

LISTADO DE LA RUTINA.- En el listado del paquete de "software multitarea" que se presenta al final del capítulo, el ejemplo de utilización de la transformada rápida de Fourier corresponde a la subrutina "TRF_TR"

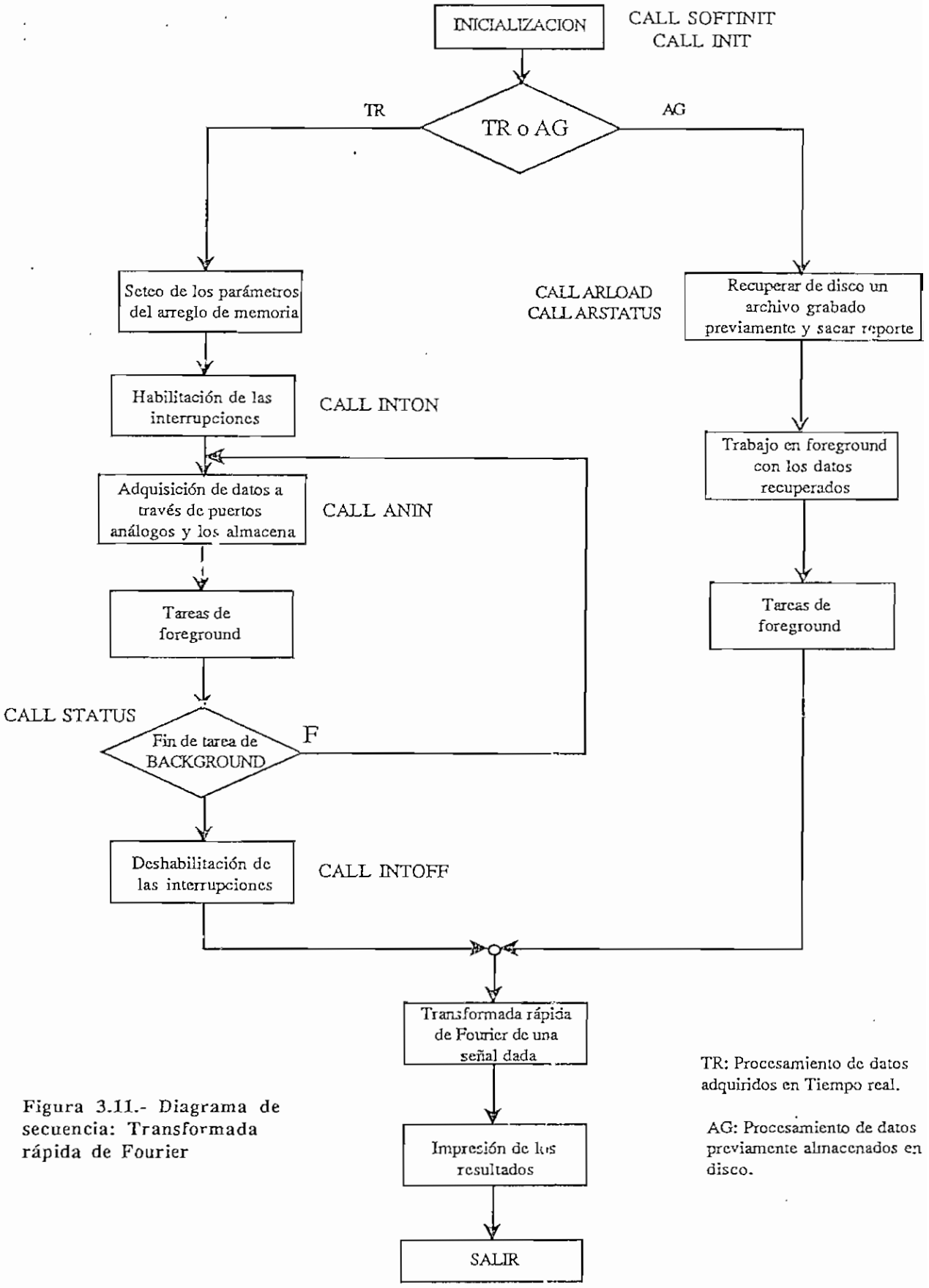


Figura 3.11.- Diagrama de secuencia: Transformada rápida de Fourier

TR: Procesamiento de datos adquiridos en Tiempo real.

AG: Procesamiento de datos previamente almacenados en disco.

3.7.- LISTADO DEL "SOFTWARE MULTITAREA" (CAP30.BAS)

```
DECLARE SUB cotriangulo ()
DECLARE SUB coseno ()
DECLARE SUB INOUTGR ()
DECLARE SUB ONOFDIS ()
DECLARE SUB GRAF500 ()
DECLARE SUB AREAS ()
DECLARE SUB PROCESAM ()
DECLARE SUB TRFTR ()
DECLARE SUB ALGORITMO ()
DECLARE SUB defgraf ()
DECLARE SUB graf (i)
DECLARE SUB seno ()
DECLARE SUB triangulo ()
DECLARE SUB defgraf1 ()
DECLARE SUB graf1 (i%)
DECLARE SUB graf2 (i%)
DECLARE SUB defgraf2 ()

DEFINT I-N
DEFDBL A-H, O-Z
COMMON SHARED valor!, sal!, va!, vmax!, vmin!, i, u, y
menu:
CLS
SCREEN 9
LINE (0, 0)-(600, 335), 10, B
LINE (3, 3)-(597, 332), 10, B
COLOR 2
LOCATE 3, 10: PRINT "   SOFTWARE MULTITAREA DEL SISTEMA DE ADUISICION DE DATOS"
LOCATE 4, 20: PRINT "       Y CONTROL KEITHLEY 500A"
LOCATE 9, 30: PRINT "MENU PRINCIPAL:"
LOCATE 10, 28: PRINT "═══════════════"
COLOR 15
LOCATE 13, 5: PRINT "E/S DE DATOS CON ALGORITMO DE CONTROL(INTERRUPCIONES)          (1)"
LOCATE 14, 5: PRINT "CONTROL ON-OFF (DISPAROS)                                     (2)"
LOCATE 15, 5: PRINT "GENERADOR DE ONDAS ELEMENTAL (GRAFICOS TR)                    (3)"
LOCATE 16, 5: PRINT "MANEJO DE ARREGLOS DE MEMORIA (ARREGLOS DE MEMORIA)           (4)"
LOCATE 17, 5: PRINT "PROCESAMIENTO DE SEÑALES (PROCESAMIENTO DE DATOS)             (5)"
LOCATE 18, 5: PRINT "TRANSFORMADA RAPIDA DE FOURIER                                (6)"
LOCATE 19, 5: PRINT "TERMINAR LA SESION DE TRABAJO                                 (7)"

pregunta:
LOCATE 22, 39: PRINT "      "
LOCATE 22, 5: INPUT "Escoja una opción por favor (1-7):"; op$

IF op$ = "1" THEN CALL INOUTGR: CLEAR : GOTO menu
IF op$ = "2" THEN CALL ONOFDIS: CLEAR : GOTO menu
IF op$ = "3" THEN CALL GRAF500: CLEAR : GOTO menu
IF op$ = "4" THEN CALL AREAS: CLEAR : GOTO menu
IF op$ = "5" THEN CALL PROCESAM: CLEAR : GOTO menu
IF op$ = "6" THEN CALL TRFTR: CLEAR : GOTO menu
IF op$ = "7" THEN END
IF op$ > "7" OR op$ < "1" THEN GOTO pregunta
```

```
DATA 0
DATA 0,1
DATA 0,2,1,3
DATA 0,4,2,6,1,5,3,7
DATA 0,8,4,12,2,10,6,14,1,9,5,13,3,11,7,15
DATA 0,16,8,24,4,20,12,28,2,18,10,26,6,22,14,30,1,17,9,25,5,21,13,29,3,19,11,27,7,23,15,31
DATA 0,32,16,48,8,40,24,56,4,36,20,52,12,44,28,60,2,34,18,50,10,42,26,58,6,38,22,54,14,46,30,62
DATA 1,33,17,49,9,41,25,57,5,37,21,53,13,45,29,61,3,35,19,51,11,43,27,59,7,39,23,55,15,47,31,63
DATA 0,64,32,96,16,80,48,112,8,72,40,104,24,88,56,120 4,68,36,100,20,84,52,116,12,76,44,108,28,92
DATA
60,124,2,66,34,98,18,82,50,114,10,74,42,106,26,90,58,122,6,70,38,102,22,86,54,118,14,78,46,110,30,
94,62,126
DATA 1,65,33,97,17,81,49,113,9,73,41,105,25,89,57,121,5,69,37,101,21,85,53,117,13,77,45,109,29,93
DATA
61,125,3,67,35,99,19,83,51,115,11,75,43,107,27,91,59,123,7,71,39,103,23,87,55,119,15,79,47,111,31,
95,63,127

SUB ALGORITMO
'-----------------------------------------------------------

    sal! = valor! * 2
    IF sal! > 10 THEN
    sal! = 10
    ELSEIF valor! > 10 THEN
    valor! = 10
    ELSEIF sal! < 0 THEN
    sal! = 0
    END IF

'-----------------------------------------------------------
END SUB

SUB AREAS
CLS                          ' Rutina que utiliza todos los comandos de manejo de arreglos de memoria
SCREEN 9
LINE (0, 0)-(639, 350), 2, BF
LINE (4, 4)-(635, 345), 0, BF
COLOR 2, 1
COLOR 15

CALL SOFTINIT
CALL INIT

LOCATE 8, 10: PRINT "Se utilizará un arreglo de memoria que acumula datos -"
LOCATE 9, 10: PRINT "adquiridos mediante los canales análogos ANLG0 y ANLG1"
LOCATE 16, 10: PRINT "Para recuperar la información del disco, se debe in -"
LOCATE 17, 10: PRINT "troducir un nombre estándar DOS de extensión .DAT, es"
LOCATE 18, 10: PRINT "te nombre debe existir en el directorio"
LOCATE 12, 10: PRINT "Para guardar la información en disco, se debe introdu-"
LOCATE 13, 10: PRINT "cir un nombre estándar DOS de extensión .DAT"
COLOR 2
LOCATE 4, 20: PRINT "M A N E J O  D E  A R R E G L O S"
LOCATE 6, 17: PRINT "D E  M E M O R I A  P A R A  D A T O S"

LOCATE 21, 15: PRINT "presione SPACE BAR para continuar"
```

```
DO
    an$ = INKEY$
LOOP WHILE an$ = ""

CLS
SCREEN 9
LINE (0, 0)-(639, 350), 2, BF
LINE (4. 4)-(635, 345), 0, BF
COLOR 2, 1
COLOR 2
LOCATE 4, 20: PRINT "M A N E J O  D E  A R R E G L O S"
LOCATE 6, 17: PRINT "D E  M E M O R I A  P A R A  D A T O S"
COLOR 15
LOCATE 8, 10: PRINT "Desea escoger el período de muestreo (T) en mseg.";
ave1:
COLOR 14: PRINT " (s/n)";
COLOR 2: LOCATE 9, 10: PRINT "De no hacerlo se muestrea con una frecuencia de 50 ms."
COLOR 15: INPUT p$
IF p$ = "s" OR p$ = "S" THEN
LOCATE 8, 70: INPUT "T ="; ir%
ELSEIF p$ = "n" OR p$ = "N" THEN
LOCATE 8, 70: PRINT "T = 50 ms.": ir% = 50
ELSE
GOTO ave1:
END IF
LOCATE 11, 10: PRINT "Desea escoger el número de datos adquiridos";
ave2:
COLOR 14: PRINT " (s/n)";
COLOR 2: LOCATE 12, 10: PRINT "De no hacerlo se adquiere 580 datos"
COLOR 15: INPUT p$
IF p$ = "s" OR p$ = "S" THEN
LOCATE 11, 70: INPUT "N ="; dep!
ELSEIF p$ = "n" OR p$ = "N" THEN
LOCATE 11, 70: PRINT "N = 580 ": dep! = 580
ELSE
GOTO ave2:
END IF
LOCATE 14, 10: PRINT "Desea ingresar un comantario del arreglo (30 caract. máx)";
ave3:
COLOR 14: PRINT " (s/n)";
COLOR 2: LOCATE 15, 10: PRINT "De no hacerlo se asume comentario interno"
COLOR 15: INPUT p$
IF p$ = "s" OR p$ = "S" THEN
LOCATE 17, 10: INPUT "COMENTARIO:"; lab$
ELSEIF p$ = "n" OR p$ = "N" THEN
lab$ = "Arreglo de memoria almacenado": LOCATE 17, 10: PRINT "COMENTARIO ASUMIDO:"; lab$
ELSE
GOTO ave3:
END IF
LOCATE 21, 15: PRINT "presione SPACE BAR para continuar"

DO
    an$ = INKEY$
LOOP WHILE an$ = ""
```

```
'----------------SETEO DE LAS TAREAS DE BACKGROUND----------------

CALL ANIN("entrada%", dep!, "ANLG0", 1, 1, "nt", "tarea")

'-----------------------------------------------------------

SCREEN 9
WIDTH 30, 43
CLS: defgraf1: COLOR 9
LOCATE 1, 5: PRINT "ESTUDIO DEL SISTEMA DE ADQUISICION DE DATOS KEYTLHEY 500A Y APLICACIONES"
COLOR 2
LOCATE 20, 10: PRINT "Tiempo Real"
LOCATE 22, 10: PRINT "Adquisición en Proceso"
LOCATE 41, 10: PRINT "Presione ESP BAR para salir"
COLOR 7
plp! = 10
st% = 1
CALL inton(ir%, "mil")
WHILE st% <> 0
     CALL STATUS("tarea", st%)
     CALL ARLASTP("entrada%", lp!)
      IF lp! <> plp! AND lp! > 0 THEN
           CALL argetvalf("entrada%", lp!, -1, "ANLG0", valor!, 0)

           IF k% > 580 THEN          'reseteo de pantalla gráfica
           defgraf1
           k% = 0
           END IF
           graf1 (k%)
           k% = k% + 1
'-----------------------------------------------
           plp! = lp!
      END IF
WEND

CALL INTOFF
CALL arlabel("entrada%", lab$)
INPUT "nombre del archivo para guardar en disco"; n$

CALL ARSAVE("entrada%", n$)
CALL arwrite("entrada%", "areas.prn", 0, 4, ir%, 1)
CALL ARDEL("entrada%")
CLS
LOCATE 8, 1: INPUT "nombre del archivo a recuperar del disco"; n$
LOCATE 14, 1: PRINT "Presione una tecla para recuperar la información del disco"
WHILE a$ = ""
   a$ = INKEY$
WEND

CALL ARLOAD("entrada%", n$)
CALL ARSTATUS("entrada%", dep!, wid%, lp!, lab$)
CLS
LOCATE 20, 10: PRINT "Estatus de la información recuperada del disco"
```

```
LOCATE 22, 1: PRINT "longitud:"; dep!, "último dato accesado:", "     ancho:"; wid%
LOCATE 23, 1: PRINT "comentarios:"; labS
LOCATE 41, 1: PRINT "presione ESP BAR para salir"

defgraf1
k% = 0
FOR lp! = 1 TO dep!
        CALL argetvalf("entrada%", lp!, -1, "ANLG0", valor!, 0)
LOCATE 22, 50: PRINT lp!

        IF k% > 580 THEN        'reseteo de pantalla gráfica
        defgraf1
        k% = 0
        END IF
        graf1 (k%)
        k% = k% + 1
NEXT
WHILE aw$ = ""
   aw$ = INKEY$
WEND
WIDTH 80, 25
END SUB

DEFSNG A-Z
SUB  coseno             'Rutina que calcula ondas senoidales para graficarlas utilizando los comandos Quick500
CLS                     'GRAPHRT
otroc:
LOCATE 5, 46: PRINT "     "
LOCATE 5, 10: INPUT "Frecuencia de las ondas (0.1-100Hz.)"; f
IF f < .1 OR f > 100 THEN GOTO otroc
otroampc:
LOCATE 7, 42: PRINT "     "
LOCATE 7, 10: INPUT "Amplitud de la onda (0-10 Vpico)"; amp
IF amp < 0 OR amp > 10 THEN GOTO otroampc
SCREEN 2, 0: CLS

CALL SOFTINIT
CALL INIT

CALL ARMAKE("salida%", 100!, 2, "")

FOR dep! = 1 TO 100
    val% = INT((SIN((dep! - 1) * 6.28 / 50) * 2047.5 + 2047.5) * amp / 10 + (10 - amp) * 4095 / 20)
    CALL arputvali("salida%", dep!, 1, "", val%)
    va2% = INT((COS((dep! - 1) * 6.28 / 50) * 2047.5 + 2047.5) * amp / 10 + (10 - amp) * 4095 / 20)
    CALL arputvali("salida%", dep!, 2, "", va2%)
   LOCATE 12, 30: PRINT 100 - dep!
NEXT

B% = INT(100 / f)

'-------------------------------------------------
CALL ANOUT("salida%", "ANOUT0 ANOUT1", B%, -1, "nt", "")
'-------------------------------------------------
```

```
CALL inton(1, "hmic")
        CALL graphrt("salida%", "1 2", "1 2", "pagec", 0!, 4095!, 10000!, -1)
CALL INTOFF

CLS : SCREEN 0
CALL ARDEL("salida%")
CALL backclear

END SUB

SUB  cotriangulo        'Rutina que calcula ondas triangulares para graficarlas utilizando los comandos Quick500
CLS                     'GRAPHRT
otrotc:
LOCATE 5, 46: PRINT "     "
LOCATE 5, 10: INPUT "Frecuencia de las ondas (0.1-200Hz.)"; f
IF f < .1 OR f > 200 THEN GOTO otrotc
otroamptc:
LOCATE 7, 42: PRINT "     "
LOCATE 7, 10: INPUT "Amplitud de la onda (0-10 Vpico)"; amp
IF amp < 0 OR amp > 10 THEN GOTO otroamptc
SCREEN 2, 0: CLS

CALL SOFTINIT
CALL INIT

CALL ARMAKE("salida%", 50!, 2, "")

FOR dep! = 1 TO 50
    va1% = INT(((2047.5 * dep! / 50) * amp / 10) + 2047.5)
    CALL arputvali("salida%", dep!, 1, "", va1%)
NEXT
FOR dep! = 1 TO 25
    va2% = INT(((2047.5 * dep! / 25) * amp / 10) + 2047.5)
    CALL arputvali("salida%", dep!, 2, "", va2%)
    va3% = INT(((2047.5 * (25 - dep!) / 25) * amp / 10) + 2047.5)
    CALL arputvali("salida%", dep! + 25, 2, "", va3%)
NEXT
B% = INT(200 / f)

'-------------------------------------------------
CALL ANOUT("salida%", "ANOUT0 ANOUT1", B%, -1, "nt", "")
'-------------------------------------------------

CALL inton(1, "hmic")
        CALL graphrt("salida%", "1 2", "1 2", "pagec", 2048!, 4095!, 3000!, -1)
CALL INTOFF

CLS : SCREEN 9
CALL ARDEL("salida%")
CALL backclear

END SUB

DEFINT I-N
DEFDBL A-H, O-Z
```

```
SUB defgraf     'Rutina que define la pantalla grafica sobre la que se grafican los datos adquiridos

LINE (0, 12)-(639, 140), 1, BF                  'Para el canal de entrada
LINE (0, 12)-(639, 140), 15, B
'--------
LINE (40, 25)-(629, 25), 5, , &H8888
'--------
LINE (30, 125)-(629, 125), 7
LOCATE 17, 40
PRINT "Tiempo"
LOCATE 5, 2: PRINT "y"
LINE (40, 20)-(40, 130), 7


LINE (0, 185)-(639, 315), 1, BF                  'Para el canal de salida
LINE (0, 185)-(639, 315), 15, B
'--------
LINE (40, 200)-(629, 200), 5, , &H8888
'--------
LINE (30, 300)-(629, 300), 7
LOCATE 39, 40
PRINT "Tiempo"
LOCATE 27, 2: PRINT "u"
LINE (40, 195)-(40, 305), 7

END SUB

SUB defgraf1        'Rutina que define la pantalla grafica sobre la que se grafican los datos adquiridos
                    'un solo canal
DEFINT I-N
DEFDBL A-H, O-Z
LINE (0, 12)-(639, 140), 1, BF
LINE (0, 12)-(639, 140), 15, B
'--------
LINE (40, 125)-(629, 125), 5, , &H8888
LINE (40, 25)-(629, 25), 5, , &H8888
'--------
LINE (30, 75)-(629, 75), 7
LOCATE 17, 40
PRINT "TIEMPO"
LOCATE 5, 2: PRINT "in"
LINE (40, 20)-(40, 130), 7

END SUB

SUB defgraf2                'Rutina que define la pantalla grafica sobre la que se grafican los datos
LINE (200, 12)-(439, 140), 1, BF    'calculados del espectro de densidad de potencia
LINE (200, 12)-(439, 140), 15, B
'--------
LINE (240, 25)-(429, 25), 5, , &H8888
'--------
LINE (230, 125)-(429, 125), 7
LOCATE 5, 19: PRINT "Pyy"
LINE (240, 20)-(240, 130), 7
LOCATE 20, 40
```

```
PRINT "frecuencia"

END SUB

DEFSNG A-Z
SUB graf (i)                'Rutina que grafica los datos adquiridos y de salida en la pantalla gráfica.
IF valor! > 9.99 THEN
    PSET (i + 40, 25), 4
      LOCATE 22, 55: PRINT "canal de entrada saturado"
ELSEIF valor! <.04 THEN
        LOCATE 22, 55: PRINT "canal de entrada saturado"
    PSET (i + 40, 125), 4
ELSE
    y = -10 * valor! + 125
      LOCATE 22, 55: PRINT "                    "
    PSET (i + 40, y), 14
END IF

IF sal! > 9.99 THEN
        LOCATE 41, 55: PRINT "canal de salida saturado"
    PSET (i + 40, 200), 4
ELSEIF sal! < .04 THEN
        LOCATE 41, 55: PRINT "canal de salida saturado"
    PSET (i + 40, 300), 4
ELSE
    u = -10 * sal! + 300
      LOCATE 41, 55: PRINT "                    "
    PSET (i + 40, u), 14
END IF

END SUB

DEFINT I-N
DEFDBL A-H, O-Z
SUB graf1 (i%)              'Rutina que grafica los datos adquiridos en la pantalla gráfica.

DEFINT I-N
DEFDBL A-H, O-Z

IF va! > 9.99 THEN
    PSET (i + 40, 25), 4
ELSEIF va! < -9.99 THEN
    PSET (i + 40, 125), 4
ELSE
    y = -5 * va! + 75
    PSET (i + 40, y), 14
END IF

END SUB

SUB graf2 (i%)              'Rutina que grafica los datos de Pyy calculados en la pantalla gráfica.
IF va! > 9.99 THEN
    PSET (i + 240, 25), 5
ELSEIF va! < -9.99 THEN
    PSET (i + 240, 125), 5
```

```
ELSE
    y = -10 * val + 125
    PSET (i + 240, y), 14
END IF

END SUB

DEFSNG A-Z
SUB GRAF500                    'Riutina que genera el menú de opciones para los gráficos utilizando Quick500
menugraf:
CLS
SCREEN 9
LINE (0, 0)-(600, 335), 14, B
LINE (3, 3)-(597, 332), 2, B
COLOR 14
LOCATE 3, 10: PRINT "  SOFTWARE MULTITAREA DEL SISTEMA DE ADUISICION DE DATOS"
LOCATE 4, 20: PRINT "      Y CONTROL KEITHLEY 500A"
LOCATE 6, 30: PRINT "GENERADOR DE ONDAS"
LOCATE 8, 10: PRINT "Ondas seniodales(HGRAPHRT)      (1)"
LOCATE 9, 10: PRINT "Ondas triangulares(HGRAPHRT)    (2)"
LOCATE 10, 10: PRINT "Ondas senoidales(GRAPHRT)      (3)"
LOCATE 11, 10: PRINT "Ondas triangulares(GRAPHRT)    (4)"
LOCATE 12, 10: PRINT "Salir                          (5)"
LOCATE 20, 10: PRINT "Para terminar presione ESC"
pregunte:
LOCATE 15, 10: INPUT "Escoga el tipo de onda"; a$
IF a$ > "5" THEN GOTO pregunte
IF a$ = "1" THEN
seno
GOTO menugraf
ELSEIF a$ = "2" THEN
triangulo
GOTO menugraf
ELSEIF a$ = "3" THEN
coseno
GOTO menugraf
ELSEIF a$ = "4" THEN
cotriangulo
GOTO menugraf
END IF
END SUB

DEFINT I-N
DEFDBL A-H, O-Z

SUB INOUTGR
CLS                            'Rutina de entrada y salida de daots con algoritmo de control
SCREEN 9
LINE (0, 0)-(639, 350), 2, BF
LINE (4, 4)-(635, 345), 0, BF
COLOR 2, 1: COLOR 15

CALL SOFTINIT
CALL INIT
tmin% = 90
```

```
LOCATE 3, 10: PRINT "ESTUDIO DEL SISTEMA DE ADQUISICION DE DATOS KEITHLRY 500A"
LOCATE 4, 10: PRINT "              Y SUS APLICACIONES"
COLOR 2
LOCATE 8, 10: PRINT "Tiempo de duración del algoritmo(ms) = "; tmin%
LOCATE 9, 10: PRINT "máximo estimado"
LINE (0, 80)-(639, 83), 2, BF
muestra:
LOCATE 11, 10: PRINT "Período de muestreo deseado    (ms) = "; : INPUT bintv%
per% = INT(bintv% / tmin%)
IF per% = 0 THEN
LOCATE 13, 10: PRINT "Período de muestreo menor que el tiempo de duración"
LOCATE 14, 10: PRINT "del algoritmo."
GOTO muestra
END IF
boutv% = bintv% / per%          'optimización del tiempo de salida
bintv% = per% * boutv%          'ajuste del tiempo de muestreo
LOCATE 16, 10: PRINT "Período de muestreo ajustado   (ms) = "; bintv%
LOCATE 18, 10: PRINT "Retardo del dato de salida     (ms) = "; boutv%
bintv% = bintv%
boutv% = boutv%

'----------SETEO DE LAS TAREAS DE BACKGROUND----------
dep! = 1

CALL ANIN("entrada%", dep!, "ANLG0", bintv%, -1, "nt", "tarea1")
CALL ANIN("dataio%", 200!, "ANLG0 ANLG1", bintv%, 1, "nt", "")

CALL ARMAKE("salida%", dep!, -1, "ANOUT0")

CALL ANOUT("salida%", "ANOUT0", boutv%, -1, "nt", "tarea2")

FOR i! = 1 TO dep!
    CALL ARPUTVALF("salida%", i!, -1, "ANOUT0", 0!, 0)
NEXT i!
'-------------------------------------------------------------
'Lazo creado con el afán de tener todo el arreglo referenciado a cero
'--------CALCULOS PRELIMINARES DEL ALGORITMO--------
' Aquí se debe ingresar todos los datos, parámetros, etc., y realizar todos los cálculos que no van dentro del lazo de
'control
'-------------------------------------------------------------
LOCATE 21, 10: PRINT "presione SPACE BAR para iniciar el control"
COLOR 2
DO
    an$ = INKEY$
LOOP WHILE an$ = ""
SCREEN 9
WIDTH 80, 43
CLS
defgraf
COLOR 9
LOCATE 1, 5: PRINT "ESTUDIO DEL SISTEMA DE ADQUISICION DE DATOS KEYTLHEY 500A Y APLICACIONES"
COLOR 2
LOCATE 20, 10: PRINT "Tiempo Real"
LOCATE 22, 10: PRINT "Control en Proceso"
LOCATE 41, 10: PRINT "SPACE BAR para terminar proceso"
```

```
COLOR 7
dt1 = 1
DIM dt(0 TO 7) AS DOUBLE, tim(0 TO 7) AS INTEGER
tim(0) = 1
CALL inton(1, "mil")
'ti1 = TIMER
CALL TIMERSTART(tim(), "nt", "timer0")

'Comando que resetea el timer 0, en este caso y lo habilita para iniciar
'su cuenta, no espera disparo.

valor! = 0
an$ = ""
WHILE an$ = ""
    arn$ = "entrada%"

    CALL ARLASTP("entrada%", lp!)
    CALL ARLASTP("salida%", lp!)

        CALL argetvalf("entrada%", 1!, -1, "ANLG0", valor!, 0)
        LOCATE 20, 36: PRINT USING "Entrada: ##.### v."; valor!

'**********************ALGORITMO DE CONTROL COMO SUBRUTINA**********************

ALGORITMO

'*******************************************************************************

        CALL ARPUTVALF("salida%", 1!, -1, "ANOUT0", sal!, 0)
        LOCATE 22, 36: PRINT USING "Control: ##.### v."; sal!
        IF k% > 580 THEN        'reseteo de pantalla gráfica
        defgraf
        k% = 0
        END IF
        graf (k%)
        k% = k% + 1

    an$ = INKEY$
'ti2 = TIMER

  DO

    CALL TIMERREAD(dt())

  LOOP UNTIL dt(0) / bintv% > dt1
  dt1 = INT(dt(0) / bintv% + 1)
'Sirve para asegurar que se ocupará todo el tiempo restante de la interrupción
'recibir otro dato.
'LOCATE 1, 1: PRINT (ti2 - ti1) * 1000 * 1.3
WEND

CALL INTOFF
CALL arwrite("dataio%", "dataio.prn", 0, 4, bintv%, 1)
CALL ARDEL("entrada%")
CALL ARDEL("salida%")
```

```
WIDTH 80, 25
END SUB

SUB ONOFDIS
preg:
CLS
LINE (0, 0)-(600, 335), 10, B
LINE (3, 3)-(597, 332), 10, B
COLOR 14
LOCATE 5, 5: INPUT "valor en voltios del valor máximo de salida"; vmax!
LOCATE 7, 5: INPUT "valor en voltios del valor mínimo de salida"; vmin!
IF vmin! > vmax! THEN
LOCATE 10, 5: PRINT "El valor mínimo debe ser menor que el valor máximo"
GOTO preg
END IF
COLOR 7
SCREEN 9
WIDTH 80, 43
CLS
LOCATE 1, 1: PRINT "Control ON-OFF: entre"; vmax!; " y "; vmin!; "voltios"
LOCATE 22, 1: PRINT "voltaje de entrada="
LOCATE 41, 1: PRINT "Presione ESP BAR para terminar el control"
DIM STA$(3)
k% = 0
j = 0

CALL SOFTINIT
CALL INIT
defgraf
LINE (40, 25 + (10 - vmax!) * 10)-(629, 25 + (10 - vmax!) * 10), 2, , &H8888
LINE (40, 25 + (10 - vmin!) * 10)-(629, 25 + (10 - vmin!) * 10), 2, , &H8888

an$ = ""
WHILE an$ = ""
    j = j + 1
    CALL ARMAKE("salida%", 1!, -1, "ANOUT0")
       CALL ARPUTVALF("salida%", 1!, -1, "ANOUT0", 9.95, 0)
    CALL ANOUT("salida%", "ANOUT0", 1, 1, "nt", "tareasal")

    CALL inton(1, "mil")
    bfn$ = "tareasal"
    st% = 1
    WHILE st% <> 0
        CALL STATUS(bfn$, st%)
    WEND
    CALL INTOFF
CALL backclear
'end
'---------------
    thrl! = vmax!
    thrh! = vmax! + .1
    chm$ = "betw"
    bfn$ = "bt"
    cy% = 1
```

```
        CALL SCHMITRIG("ANLG0", thrl!, thrh!, chm$, 0, "bt", bfn$, cy%)
        CALL ANIN("ingreso%", 1!, "ANLG1", 1, 1, "wbt", "tarea")
        CALL ANIN("entrada%", 1!, "ANLG0", 1, -1, "nt", "")

        CALL inton(1, "mil")
        bfn$ = "tarea"
        st% = 1
        WHILE st% <> 0

            CALL STATUS(bfn$, st%)
            CALL ARLASTP("entrada%", lp!)
                CALL argetvalf("entrada%", 1!, -1, "ANLG0", valor!, 0)
                CALL argetvalf("ingreso%", 1!, -1, "ANLG1", sal!, 0)
                LOCATE 22, 22: PRINT valor!; "voltios   "
            IF k% > 580 THEN          'reseteo de pantalla gráfica
            defgraf
LINE (40, 25 + (10 - vmax!) * 10)-(629, 25 + (10 - vmax!) * 10), 2, , &H8888
LINE (40, 25 + (10 - vmin!) * 10)-(629, 25 + (10 - vmin!) * 10), 2, , &H8888

            k% = 0
            END IF
            graf (k%)
            k% = k% + 1
        WEND
        CALL INTOFF
'--------------------
          CALL ARPUTVALF("salida%", 1!, -1, "ANOUT0", .08, 0)
        CALL ANOUT("salida%", "ANOUT0", 1, 1, "nt", "tareasal")

        CALL inton(1, "mil")
        bfn$ = "tareasal"
        st% = 1
        WHILE st% <> 0
            CALL STATUS(bfn$, st%)
        WEND
        CALL INTOFF
CALL ARDEL("entrada%")
CALL ARDEL("ingreso%")
'--------------------

        thrl! = vmin! - .1
        thrh! = vmin!
        chm$ = "betw"
        bfn$ = "bt"
        cy% = 1

        CALL SCHMITRIG("ANLG0", thrl!, thrh!, chm$, 0, "bt", bfn$, cy%)
        CALL ANIN("ingreso%", 1!, "ANLG1", 1, 1, "wbt", "tarea")
        CALL ANIN("entrada%", 1!, "ANLG0", 1, -1, "nt", "")

        CALL inton(1, "mil")
        bfn$ = "tarea"
        st% = 1
        WHILE st% <> 0
```

```
        CALL STATUS(bfn$, st%)
        CALL ARLASTP("entrada%", lp!)
            CALL argetvalf("entrada%", 1!, -1, "ANLG0", valor!, 0)
            CALL argetvalf("ingreso%", 1!, -1, "ANLG1", sal!, 0)
            LOCATE 22, 22: PRINT valor!; "voltios   "
        IF k% > 580 THEN          'reseteo de pantalla gráfica
        defgraf
LINE (40, 25 + (10 - vmax!) * 10)-(629, 25 + (10 - vmax!) * 10), 2, , &H8888
LINE (40, 25 + (10 - vmin!) * 10)-(629, 25 + (10 - vmin!) * 10), 2, , &H8888

        k% = 0
        END IF
        graf (k%)
        k% = k% + 1

    WEND
    CALL INTOFF
CALL ARDEL("entrada%")
CALL ARDEL("ingreso%")
CALL ARDEL("salida%")
CALL backclear
an$ = INKEY$
WEND
WIDTH 80, 25
END SUB

SUB PROCESAM

'integ! = 0
max! = 0
min! = 10
CALL SOFTINIT
CALL INIT

esc:
CLS
SCREEN 9
LINE (0, 0)-(639, 350), 2, BF
LINE (4, 4)-(635, 345), 0, BF
COLOR 2, 1
COLOR 15
LOCATE 5, 10: PRINT "Procesamiento sobre arreglos almacenados anteriormente      (1)"
LOCATE 6, 10: PRINT "Procesamiento sobre arreglos adquiridos en tiempo real      (2)"
LOCATE 8, 10: PRINT "Escoja una opción:"; : INPUT p$
IF p$ = "2" THEN
LOCATE 12, 10: INPUT "Período de Muestreo"; ir%
LOCATE 13, 10: INPUT "Número de datos adquiridos"; dep!
'-----SETEO DE LAS TAREAS DE BACKGROUND----------

CALL ANIN("entrada%", dep!, "ANLG0", 1, 1, "nt", "proceso",

'-------------------------------------------------------------------
LOCATE 21, 10: PRINT "presione SPACE BAR para iniciar adquisición"
COLOR 2
DO
```

```
    an$ = INKEY$
LOOP WHILE an$ = ""
SCREEN 9
WIDTH 80, 43
CLS: defgraf1: COLOR 9
LOCATE 1, 5: PRINT "ESTUDIO DEL SISTEMA DE ADQUISICION DE DATOS KEYTLHEY 500A Y APLICACIONES"
COLOR 2
LOCATE 20, 10: PRINT "Tiempo Real"
LOCATE 22, 10: PRINT "Adquisición en Proceso"
COLOR 7
lp! = 1
CALL inton(ir%, "mil")
st% = 1
WHILE st% <> 0

    CALL STATUS("proceso", st%)
    CALL ARLASTP("entrada%", lp!)
        IF lp! <> plp! THEN
         CALL argetvalf("entrada%", lp!, -1, "ANLG0", va!, 0)
         LOCATE 20, 40: PRINT USING "Entrada: ##.### v."; va!
         IF k% > 580 THEN          'reseteo de pantalla gráfica
        defgraf1
        k% = 0
      END IF
         graf1 (k%)
        k% = k% + 1
'-----------------------------------------
        IF va! > max! THEN
            max! = va!
        END IF
        IF va! < min! THEN
            min! = va!
        END IF
LOCATE 32, 10: PRINT USING "valor máximo=       ##.###"; max!
LOCATE 32, 40: PRINT USING "valor mínimo=       ##.###"; min!
'-----------------------------------------
        plp! = lp!
       END IF
WEND

CALL INTOFF
CALL arwrite("entrada%", "proceso.prn", 0, 4, ir%, 1)
ELSEIF p$ = "1" THEN
LOCATE 12, 10: INPUT "nombre del archivo a recuperar del disco"; n$
LOCATE 14, 10: INPUT "Período de muestreo en ms."; ir%
LOCATE 16, 10: PRINT "Presione una tecla para recuperar la información del disco"
WHILE a$ = ""
   a$ = INKEY$
WEND

CALL ARLOAD("entrada%", n$)
CALL ARSTATUS("entrada%", dep!, wid%, lp!, lab$)
SCREEN 9
WIDTH 80, 43
CLS
```

```
LOCATE 1, 3: PRINT "SISTEMA DE ADQUISICION DE DATOS Y CONTROL KEITHLEY 500A"
defgraf1
k% = 0
FOR de! = 1 TO dep!
        CALL argetvalf("entrada%", de!, -1, "ANLG0", va!, 0)
        LOCATE 22, 10: PRINT "ANALISIS ESTADISTICO:"

        IF k% > 580 THEN        'reseteo de pantalla gráfica
        defgraf1
        k% = 0
        END IF
        graf1 (k%)
        k% = k% + 1
'-------------------------------------------------
        IF va! > max! THEN
            max! = va!
        END IF
        IF va! < min! THEN
            min! = va!
        END IF
LOCATE 27, 1: PRINT USING "valor máximo      =##.###"; max!
LOCATE 27, 44: PRINT USING "valor mínimo      =##.###"; min!

'-------------------------------------------------

NEXT
ELSE
GOTO esc:
END IF

'LOCATE 22, 10: PRINT "Calculando valor medio y desviación estándar"
CALL MEANDEV("entrada%", 1, mean!, desv!, 1!, dep!, 0)
LINE (40, 25 + (10 - mean!) * 5)-(40 + dep!, 25 + (10 - mean!) * 5), 2, , &H7777
'integ! = 0
wi% = 1
dep1! = 1
'nu! = INT(dep! / 2)
m! = nu! * 2
IF dep! = m! THEN
m! = dep! - 1
ELSEIF dep! > m! THEN
m! = dep!
END IF
DIM y(m!)
FOR i! = 1 TO m!
    CALL argetvalf("entrada%", i!, -1, "ANLG0", va!, 0)
    y(i!) = va!
NEXT
a = y(1)
FOR j! = 2 TO m! - 1 STEP 2
    a = a + 4 * y(j!) + 2 * y(j! + 1)
NEXT
a = a - y(m!)
a = a * (ir% * .001) / 3
integ! = a
```

```
'FOR dep2! = 2 TO dep!
'   CALL INTEGRAL("entrada%", wi%, integ!, dep1!, dep2!, 0)
'   valor! = 0
'   CALL ARGETVALF("entrada%", dep2!, 1, "ANLG0", valor!, 0)
'   LOCATE 36, 20: PRINT dep2!, valor!, integ!
'NEXT
LOCATE 24, 10: INPUT "Punto de derivación (menor que la profundidad del arreglo)"; de!
CALL derivative("entrada%", de!, 1, deriv!, 1)
LOCATE 24, 10: PRINT "                                                    "

CALL LINREGR("entrada%", 0, 1, m!, B!, det!, corr!, sterr!, 1!, dep!, 0)
LOCATE 25, 1: PRINT USING "valor medio       = ##.###"; mean!
LOCATE 25, 44: PRINT USING "desviación estándar = ##.###"; desv!
LOCATE 29, 1: PRINT USING "integral (área)    = #####.###"; integ!
LOCATE 29, 44: PRINT USING "derivada          = ####.###"; deriv!
LOCATE 31, 1: PRINT USING "recta aproximada  y = ##.####"; m!;
PRINT USING " x + ##.####"; B!
LOCATE 33, 1: PRINT USING "coef. determinación = ##.###"; det!
LOCATE 33, 44: PRINT USING "coef. de correlac  = ##.###"; corr!
LOCATE 35, 1: PRINT USING "error estándar    = #.###"; sterr!
LOCATE 37, 1: INPUT "desea imprimir estos resultados numéricos en papel (s/n)"; p$
IF p$ = "s" OR p$ = "S" THEN
OPEN "LPT1" FOR OUTPUT AS #1
PRINT #1, "              valor medio=          "; mean!
PRINT #1,
PRINT #1, "              desviación estándar=    "; desv!
PRINT #1,
PRINT #1, "              integral (área)=       "; integ!
PRINT #1,
PRINT #1, "              derivada=             "; deriv!
PRINT #1,
PRINT #1, "              recta aproximada=>   y ="; m!; "t + "; B!
PRINT #1,
PRINT #1, "              coef. de determinac=    "; det!
PRINT #1,
PRINT #1, "              coef. de correlac =    "; corr!
PRINT #1,
PRINT #1, "              error estándar estimado= "; sterr!

CLOSE #1
END IF
WHILE aw$ = ""
   aw$ = INKEY$
WEND
WIDTH 80, 25
END SUB

DEFSNG A-Z
SUB  seno                'Rutina que calcula ondas senoidales para graficarlas utilizando los comandos Quick500
CLS
otros:
LOCATE 5, 46: PRINT "      "
LOCATE 5, 10: INPUT "Frecuencia de las ondas (0.1-100Hz.)"; f
IF f < .1 OR f > 100 THEN GOTO otros
otroamps:
```

```
LOCATE 7, 42: PRINT "      "
LOCATE 7, 10: INPUT "Amplitud de la onda (0-10 Vpico)"; amp
IF amp < 0 OR amp > 10 THEN GOTO otroamps
SCREEN 2, 0: CLS

CALL SOFTINIT
CALL INIT

CALL ARMAKE("salida%", 100!, 2, "")

FOR dep! = 1 TO 100
    val% = INT((SIN((dep! - 1) * 6.28 / 50) * 2047.5 + 2047.5) * amp / 10 + (10 - amp) * 4095 / 20)
    CALL arputvali("salida%", dep!, 1, "", val%)
    va2% = INT((COS((dep! - 1) * 6.28 / 50) * 2047.5 + 2047.5) * amp / 10 + (10 - amp) * 4095 / 20)
    CALL arputvali("salida%", dep!, 2, "", va2%)
   LOCATE 12, 30: PRINT 100 - dep!
NEXT

B% = INT(100 / 1)

'-----------------------------------------
CALL ANOUT("salida%", "ANOUT0 ANOUT1", B%, -1, "nt", "")
'-----------------------------------------

CALL inton(1, "hmic")
        labS = "Presione ESC para terminar"
         CALL grlabel(labS, 1, 2, "bottom", "ctr")
        labS = "10      -10"
         CALL grlabel(labS, 1, 2, "left", "bottom")
         CALL grlabel(labS, 2, 2, "left", "bottom")

          CALL hgraphrt("salida%", "1 2", "pagec", "0 0", "4095 4095", "-1", -1!, 2, "grid")

CALL INTOFF

CLS : SCREEN 0
CALL ARDEL("salida%")
CALL backclear

END SUB

DEFINT I-N
DEFDBL A-H, O-Z
SUB TRFTR               'Rutina que calcula la Transformada rápida de Fourier de una onda adquirida o almace-
                        'nada y calcula el espectro de densidad de potencia
CALL SOFTINIT
CALL INIT
esca:
CLS: SCREEN 9
LINE (0, 0)-(639, 350), 2, BF
LINE (4, 4)-(635, 345), 0, BF
COLOR 2, 1: COLOR 15
LOCATE 5, 10: PRINT "Procesamiento sobre arreglos almacenados anteriormente      (1)"
LOCATE 6, 10: PRINT "Procesamiento sobre arreglos adquiridos en tiempo real      (2)"
LOCATE 8, 10: PRINT "Escoja una opción:"; : INPUT pS
```

```
IF p$ = "2" THEN
LOCATE 12, 10: INPUT "Período de Muestreo"; ir%
LOCATE 13, 10: INPUT "Número de datos adquiridos"; dep!
'----------SETEO DE LAS TAREAS DE BACKGROUND----------

CALL ANIN("entrada%", dep!, "ANLG0", 1, 1, "nt", "proceso")

'----------
LOCATE 21, 10: PRINT "presione SPACE BAR para iniciar adquisición"
COLOR 2
DO
    an$ = INKEY$
LOOP WHILE an$ = ""
SCREEN 9
WIDTH 80, 43
CLS: defgraf1: COLOR 9
LOCATE 1, 5: PRINT "ESTUDIO DEL SISTEMA DE ADQUISICION DE DATOS KEYTLHEY 500A Y APLICACIONES"
COLOR 2
LOCATE 20, 10: PRINT "Tiempo Real"
LOCATE 22, 10: PRINT "Adquisición en Proceso"
COLOR 7
lp! = 1
CALL inton(ir%, "nil")
st% = 1
WHILE st% <> 0
    CALL STATUS("proceso", st%)
    CALL ARLASTP("entrada%", lp!)
        IF lp! <> plp! THEN
         CALL argetvalf("entrada%", lp!, -1, "ANLG0", va!, 0)
         LOCATE 20, 40: PRINT USING "Entrada: ##.### v."; va!
         IF k% > 580 THEN          'reseteo de pantalla gráfica
        defgraf1
        k% = 0
        END IF
         graf1 (k%)
        k% = k% + 1
'----------
        IF va! > max! THEN
            max! = va!
        END IF
        IF va! < min! THEN
            min! = va!
        END IF
'----------
         plp! = lp!
        END IF
WEND
CALL INTOFF
CALL arwrite("entrada%", "proceso.prn", 0, 4, ir%, 1)
ELSEIF p$ = "1" THEN
LOCATE 12, 10: INPUT "nombre del archivo a recuperar del disco"; n$
LOCATE 14, 10: PRINT "Presione una tecla para recuperar la información del disco"
LOCATE 16, 10: PRINT "Período de muestreo en (ms)"; : INPUT ; bintv%
WHILE a$ = ""
  a$ = INKEY$
```

```
WEND
CALL ARLOAD("entrada%", n$)
CALL arwrite("entrada%", "proceso.prm", 0, 4, ir%, 1)
CALL ARSTATUS("entrada%", dep!, wid%, lp!, lab$)
SCREEN 9
WIDTH 80, 43
CLS
defgraf1
k% = 0
FOR de! = 1 TO dep!
        CALL argetvalf("entrada%", de!, -1, "ANLG0", va!, 0)
LOCATE 22, 50: PRINT lp!
        IF k% > 580 THEN        'reseteo de pantalla gráfica
        defgraf1
        k% = 0
        END IF
        graf1 (k%)
        k% = k% + 1
'-------------------------------------------------
        IF va! > max! THEN
            max! = va!
        END IF
        IF va! < min! THEN
            min! = va!
        END IF
'-------------------------------------------------
NEXT
ELSE
GOTO esca:
END IF
'LOCATE 25, 10: print "Numero de datos"; 32!
'Empieza el algoritmo de cálculo de la transformada rápida de Fourier para 32 datos (el algoritmo funciona hasta con
'128 datos, pero la falta de memoria del computador obligo a fijar este techo.
dep! = 32!
DIM x(dep!), xt(dep!), xti(dep!), xi(dep!), dato(dep!, dep!)
pi = 3.14159265#
n = dep!
FOR lp! = 1 TO dep!
    CALL argetvalf("entrada%", lp!, -1, "ANLG0", va!, 0)
    x(lp! - 1) = va!
NEXT lp!
m = n: k = 0
WHILE m > 1
    m = m / 2: k = k + 1
WEND
FOR i = 0 TO k
    FOR j = 0 TO ((2 ^ (i)) - 1)
        READ dato(i, j)
    NEXT
NEXT
k = k - 1
p = 0
m = n / 2
d = 0
FOR l = 0 TO k - 1
```

```
        FOR j = 0 TO (2 ^ l - 1)
            FOR i = 2 * p TO m + 2 * p - 1
                xt(i) = x(i) + (x(i + m)) * COS(-dato(l, j) * pi / 2 ^ l) - (xi(i + m)) * SIN(-dato(l, j) * pi / 2 ^ l)
                xti(i) = x(i + m) * SIN(-dato(l, j) * pi / 2 ^ l) + xi(i) + (xi(i + m)) * COS(-dato(l, j) * pi / 2 ^ l)
                xt(i + m) = x(i) - (x(i + m)) * COS(-dato(l, j) * pi / 2 ^ l) + (xi(i + m)) * SIN(-dato(l, j) * pi / 2 ^ l)
                xti(i + m) = -x(i + m) * SIN(-dato(l, j) * pi / 2 ^ l) + xi(i) - (xi(i + m)) * COS(-dato(l, j) * pi / 2 ^ l)
              d = d + 1
            NEXT
            p = p + m: d = 0
        NEXT
        p = 0: m = m / 2
        FOR i = 0 TO n - 1
            x(i) = xt(i)
            xi(i) = xti(i)
        NEXT
    NEXT
    j = 0
    'VIEW PRINT 25 TO 35
    FOR i = 0 TO n - 2 STEP 2
        xt(i) = (1 / n) * (x(i) + (x(i + 1)) * COS(-dato(k, j) * pi / (n / 2)) - (xi(i + 1)) * SIN(-dato(k, j) * pi / (n / 2)))
        xti(i) = (1 / n) * (x(i + 1) * SIN(-dato(k, j) * pi / (n / 2)) + xi(i) + (xi(i + 1)) * COS(-dato(k, j) * pi / (n / 2)))
        xt(i + 1) = (1 / n) * (x(i) - (x(i + 1)) * COS(-dato(k, j) * pi / (n / 2)) + (xi(i + 1)) * SIN(-dato(k, j) * pi / (n / 2)))
        xti(i + 1) = (1 / n) * (-x(i + 1) * SIN(-dato(k, j) * pi / (n / 2)) + xi(i) - (xi(i + 1)) * COS(-dato(k, j) * pi / (n / 2)))

    '   PRINT USING "###.#####"; "            "; xt(i); " "; xti(i); "i" ; : INPUT "     "; s$
    '   PRINT USING "###.#####"; "            "; xt(i + 1); " "; xti(i + 1); "i"; : INPUT "     "; s$
    '   PRINT USING "###.#####"; xt(i); xti(i); : PRINT "i"; : INPUT "     "; s$
    '   PRINT USING "###.#####"; xt(i + 1); xti(i + 1); : PRINT "i"; : INPUT "     "; s$
        j = j + 1
    NEXT
    CLS
    defgraf2
    LOCATE 1, 20: PRINT "Densidad de Potencia (T.R.F.)"
    k% = 0
    FOR i = 0 TO n - 1 - n / 2
        va! = (xt(dato(5, i)) ^ 2 + xti(dato(5, i)) ^ 2) ^ .5
      LOCATE 30, 20: PRINT "frecuencia   Dens. de Potencia"
      LOCATE 32, 20: PRINT USING "   ###.##       Pyy= ###.### "; i * 1000 / (bintv% * 32); va!; : INPUT ; s$
          va! = va! * 5
          IF k% > 580 THEN          'reseteo de pantalla gráfica
          defgraf2
          k% = 0
          END IF
          graf2 (k%)
          graf2 (k%)
          k% = k% + 2
    NEXT
    LOCATE 40, 10: PRINT "Presione una tecla para salir"
    WHILE a$ = ""
      a$ = INKEY$
    WEND
    CLS : WIDTH 80, 25
    END SUB

    DEFSNG A-Z
```

```
SUB triangulo              'Rutina que calcula ondas triangulares para graficarlas utilizando los comandos Quick500
CLS                        'Gráficos de alta resolución.
otrot:
LOCATE 5, 46: PRINT "      "
LOCATE 5, 10: INPUT "Frecuencia de las ondas (0.1-200Hz.)"; f
IF f < .1 OR f > 200 THEN GOTO otrot
otroampt:
LOCATE 7, 42: PRINT "      "
LOCATE 7, 10: INPUT "Amplitud de la onda (0-10 Vpico)"; amp
IF amp < 0 OR amp > 10 THEN GOTO otroampt
SCREEN 2, 0: CLS
CALL SOFTINIT
CALL INIT
CALL ARMAKE("salida%", 50!, 2, "")
FOR dep! = 1 TO 50
    val% = INT(((2047.5 * dep! / 50) * amp / 10) + 2047.5)
    CALL arputvali("salida%", dep!, 1, "", val%)
NEXT
FOR dep! = 1 TO 25
    va2% = INT(((2047.5 * dep! / 25) * amp / 10) + 2047.5)
    CALL arputvali("salida%", dep!, 2, "", va2%)
    va3% = INT(((2047.5 * (25 - dep!) / 25) * amp / 10) + 2047.5)
    CALL arputvali("salida%", dep! + 25, 2, "", va3%)
NEXT
B% = INT(200 / f)

'-----------------------------------------------
CALL ANOUT("salida%", "ANOUT0 ANOUT1", B%, -1, "nt", "")
'-----------------------------------------------

CALL inton(1, "hmic")
        lab$ = "Presione ESC para terminar"
         CALL grlabel(lab$, 1, 2, "bottom", "ctr")
        lab$ = "10       10"
         CALL grlabel(lab$, 1, 2, "left", "bottom")
         CALL grlabel(lab$, 2, 2, "left", "bottom")
          CALL hgraphrt("salida%", "1 2", "scroll", "2048 2048", "4095 4095", "-1", -1!, 2, "grid")
CALL INTOFF
CLS : SCREEN 9
CALL ARDEL("salida%")
CALL backclear

END SUB
```

4.1.-RESULTADOS DE LAS RUTINAS DESARROLLADAS.-

Para demostrar la validez de los programas que se han desarrollado se realizaron varias pruebas, las mismas que arrojan resultados satisfactorios, que se presentan a continuación.

Dichos resultados se presentan mediante curvas obtenidas en un registrador, curvas realizadas en una hoja de cálculo (Microsoft Excel Macintosh) en base a datos que se almacenan y posteriormente se recuperan, o mediante visualización, de acuerdo a la naturaleza de la prueba y sus resultados posibles.

4.1.1.- ADQUISICION DE DATOS ANALOGOS.-

Para la adquisición de datos análogos se realizaron pruebas en background, ya que éste esquema de trabajo permite el almacenamiento de datos para su posterior análisis o graficación.

La primera prueba que se realizó fue la adquisición de una señal de baja frecuencia (0.07 Hz. aprox.) con el objetivo de obtener resultados tanto con el registrador como mediante datos almacenados e importados a una hoja de cálculo y poderlos comparar posteriormente.

El la figura 4.4 se muestra los resultados obtenidos con el registrador, y en la figura 4.1 los resultados obtenidos a partir de datos adquiridos. Es interesante observar que estos datos que fueron adquiridos de un generador de onda presentan una deformación en la onda senoidal que se reproduce en ambas ondas presentadas.

Posteriormente se obtuvieron resultados adquiriendo señales de frecuencias mayores, esto es para 0.7 Hz. (10 veces mayor que la anterior) y para 7 Hz. (100 veces mayor). Estos resultados se presentan en la figura 4.2 y figura

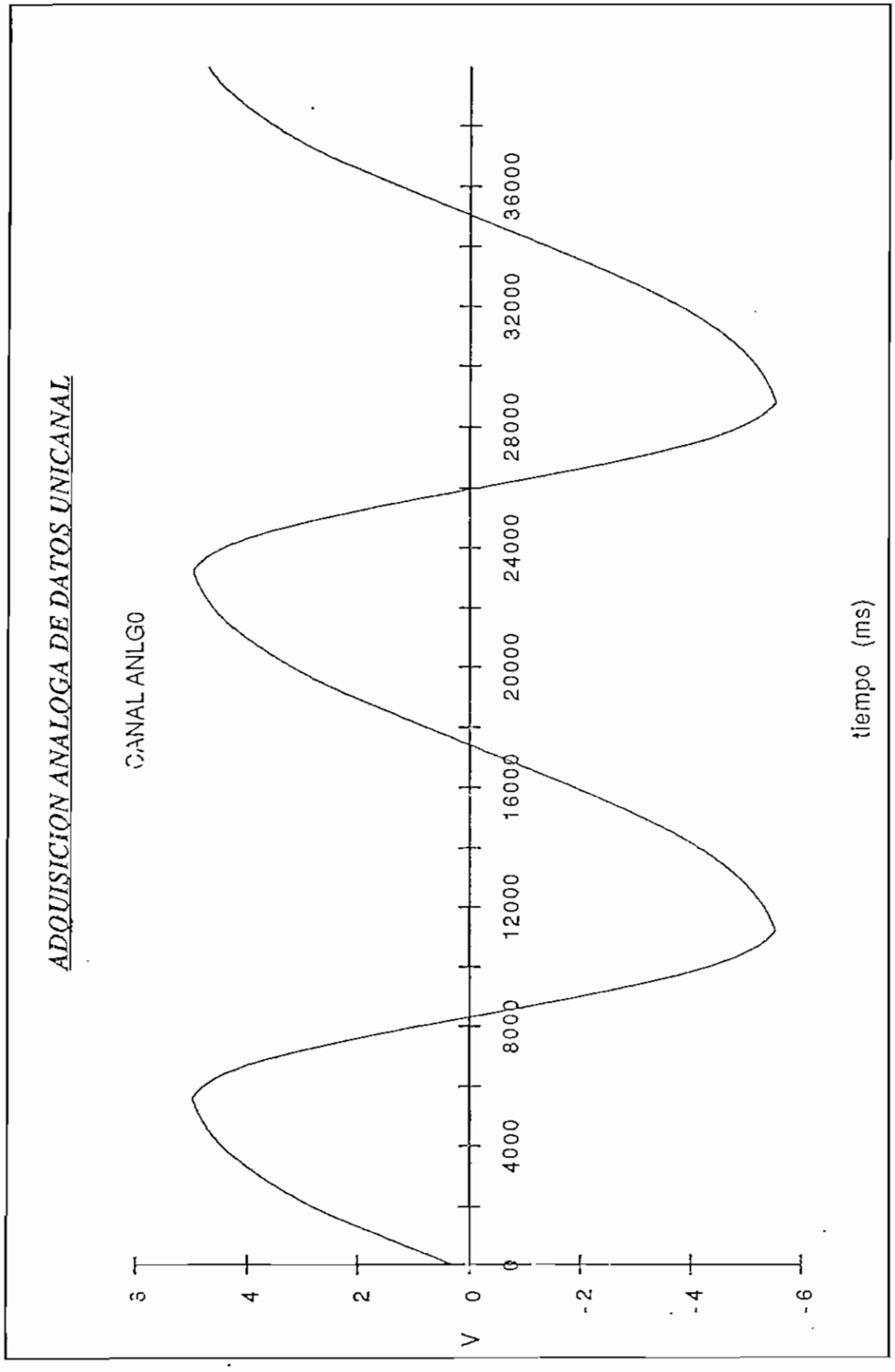


Figura 4.1.- Adquisición análoga de datos unicanal.- (señal de 0.07 Hz.)

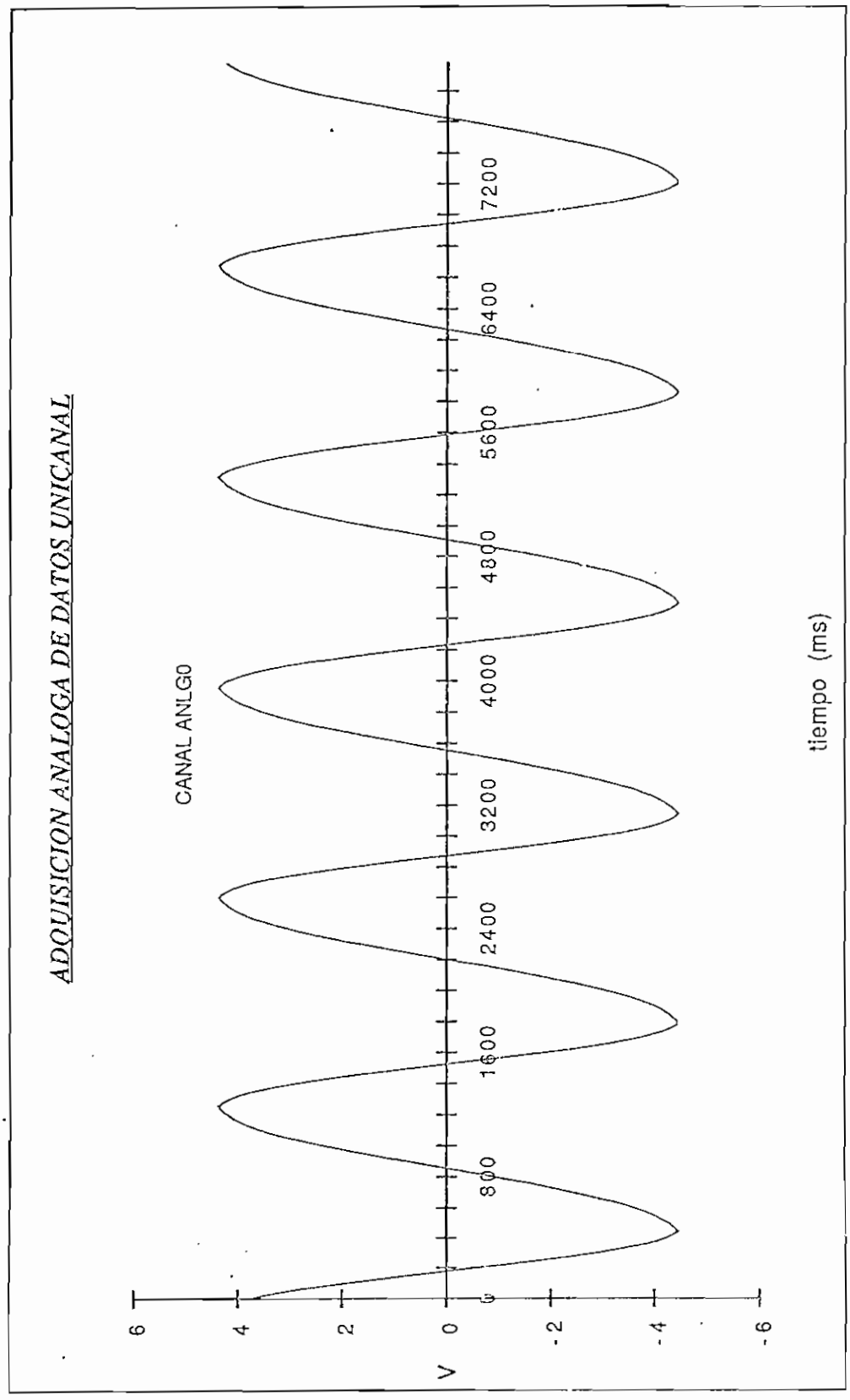


Figura 4.2.- Adquisición análoga de datos unicanal.- (señal de 0.7 Hz.)

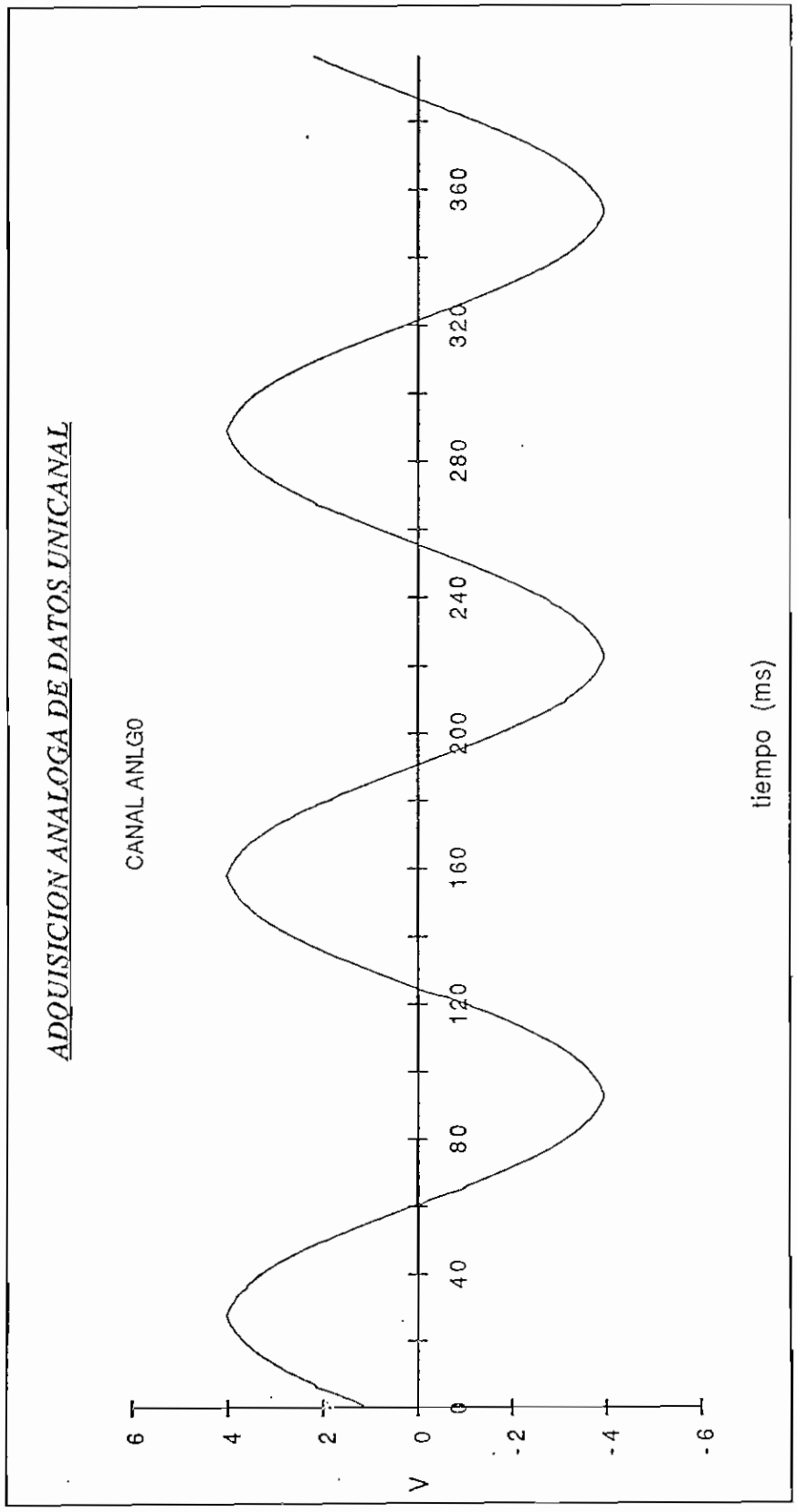


Figura 4.3.- Adquisición análoga de datos unicanal.- (señal de 7 Hz.)

4.3. respectivamente.

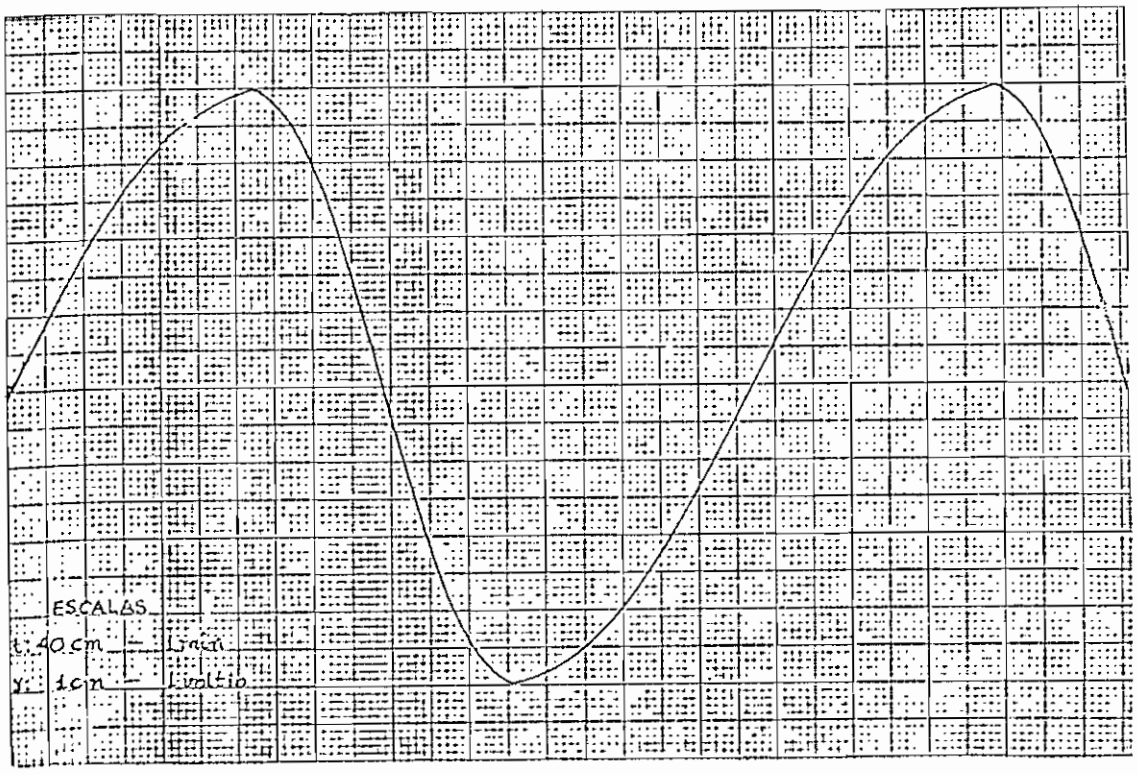


Figura 4.4.- Adquisición de datos análogos. Onda de 0.07 Hz.

Para obtener pruebas del caso multicanal se utilizó un simple circuito R-C, cuyo esquema y valores se muestran en la figura 4.5, del cual se muestrean tanto la entrada que es una señal paso, como la salida. Los resultados obtenidos se muestran en la figura 4.6.

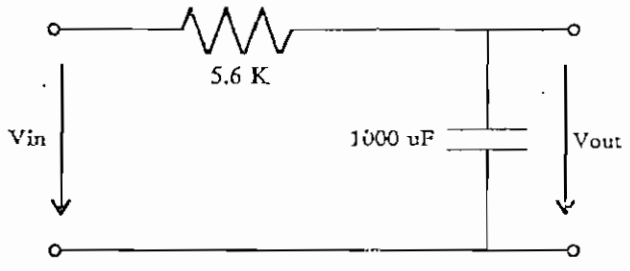


Figura 4.5.- Esquema del circuito R-C utilizado.

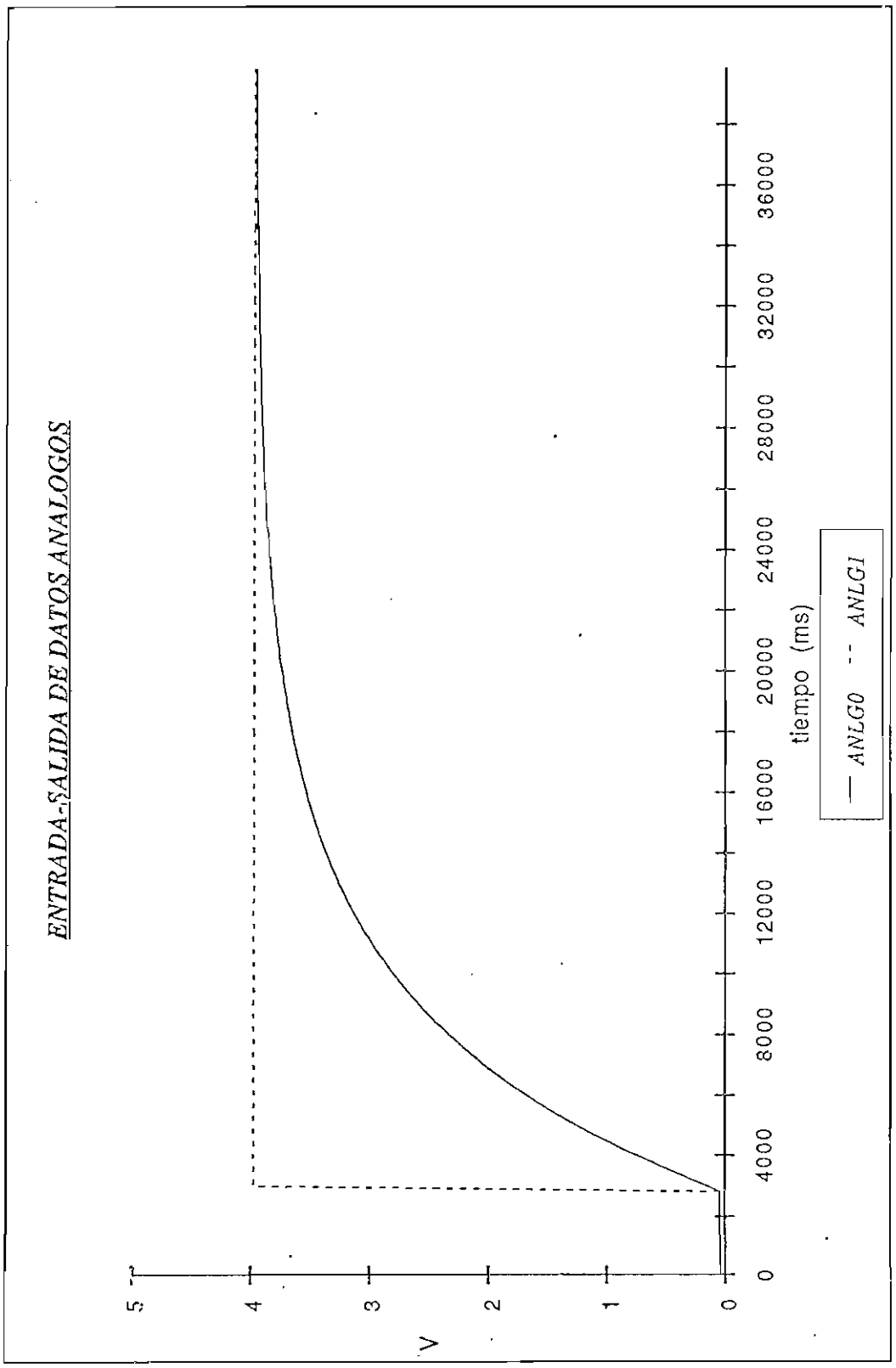


Figura 4.6.- Adquisición análoga de datos multicanal.- (entrada y salida del circuito R-C)

4.1.2.- SALIDA DE DATOS ANALOGOS.-

Para la salida de datos análogos se realizaron las pruebas también en el esquema de background debido a las facilidades que presenta en lo que se refiere a la acumulación de datos para su graficación mediante hoja de cálculo.

Como primera prueba se tiene una sinusoide de 0.1 Hz. de frecuencia la misma que captada por el registrador arroja los resultados que se muestran en la figura 4.7, donde se puede apreciar claramente los pasos de voltaje en cada muestreo del canal de salida. Se muestrea los canales con una período de 100 ms.

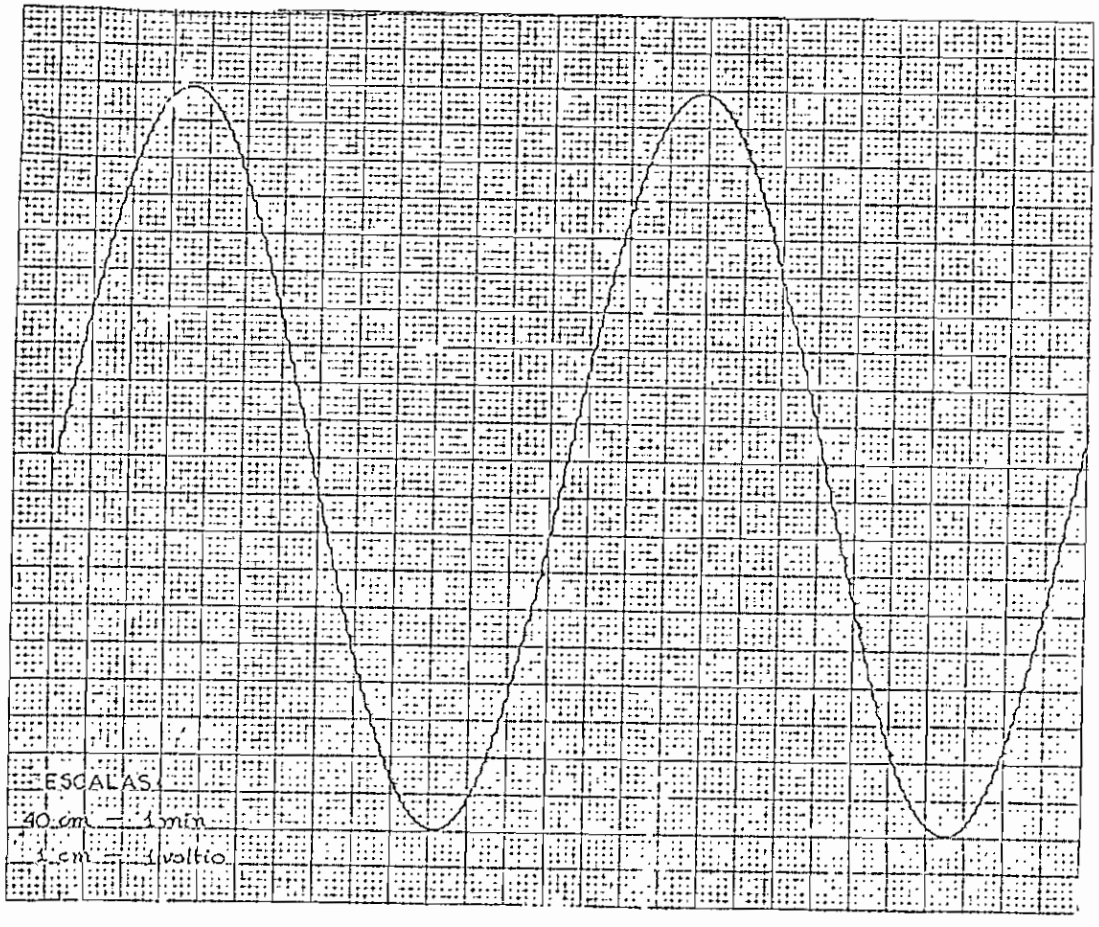


Figura 4.7.- Sinusoide de 0.1 Hz. obtenida de la salida análoga.

Con muestreos mucho más rápidos (cada ms.) se sacan señales compuestas que responden a las siguientes fórmulas:

$$y = \text{sen}(2Hz.) + \text{sen}(5Hz.) + 2*rnd - 1$$

$$y = \text{sen}(2Hz.) + \text{sen}(5Hz.)$$

y cuyos resultados se muestran en la figura 4.8.

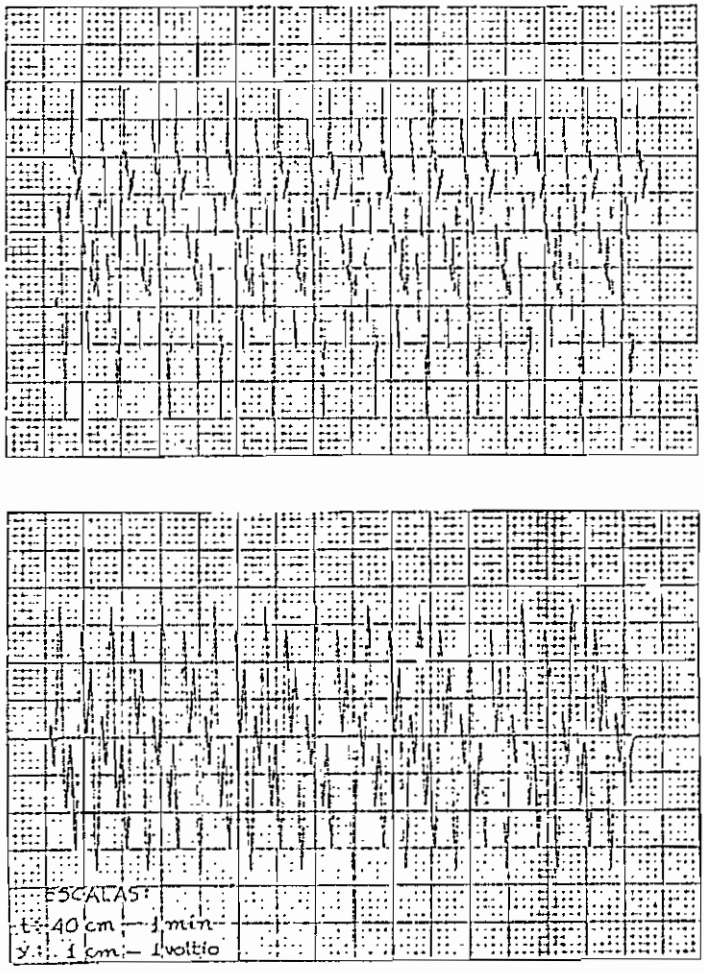


Figura 4.8.- Señales compuestas obtenidas mediante una salida análoga.

Para el caso multicanal se obtuvieron los resultados mediante hoja electrónica, luego de almacenar los datos que salen a través de los diferentes canales análogos, estos resultados se presentan en la figura 4.9. Cada canal análogo de salida aporta una onda diferente.

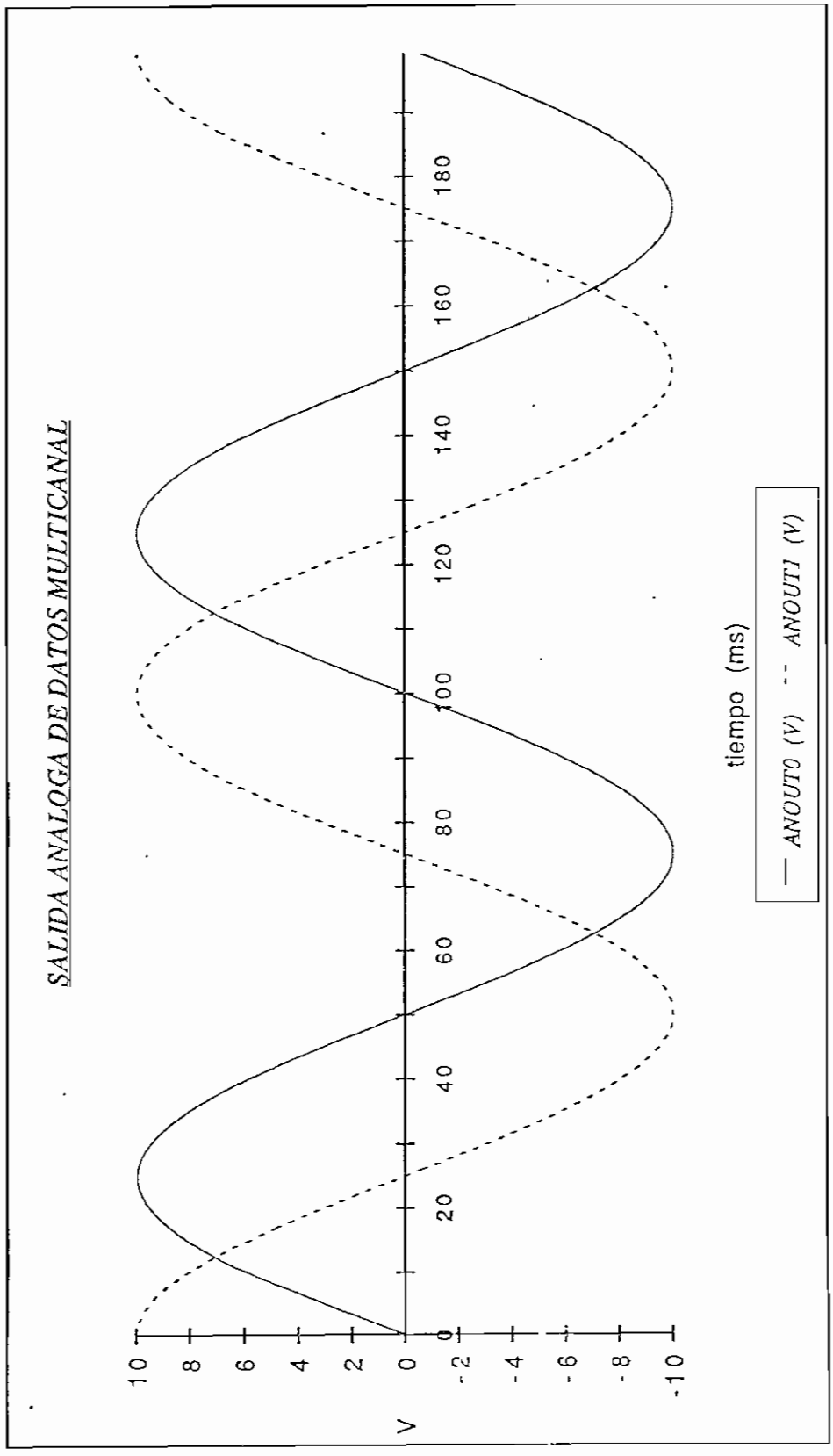


Figura 4.9a.- Salida análoga de datos multicanal.- (canales: ANOUT0 y ANOUT1)

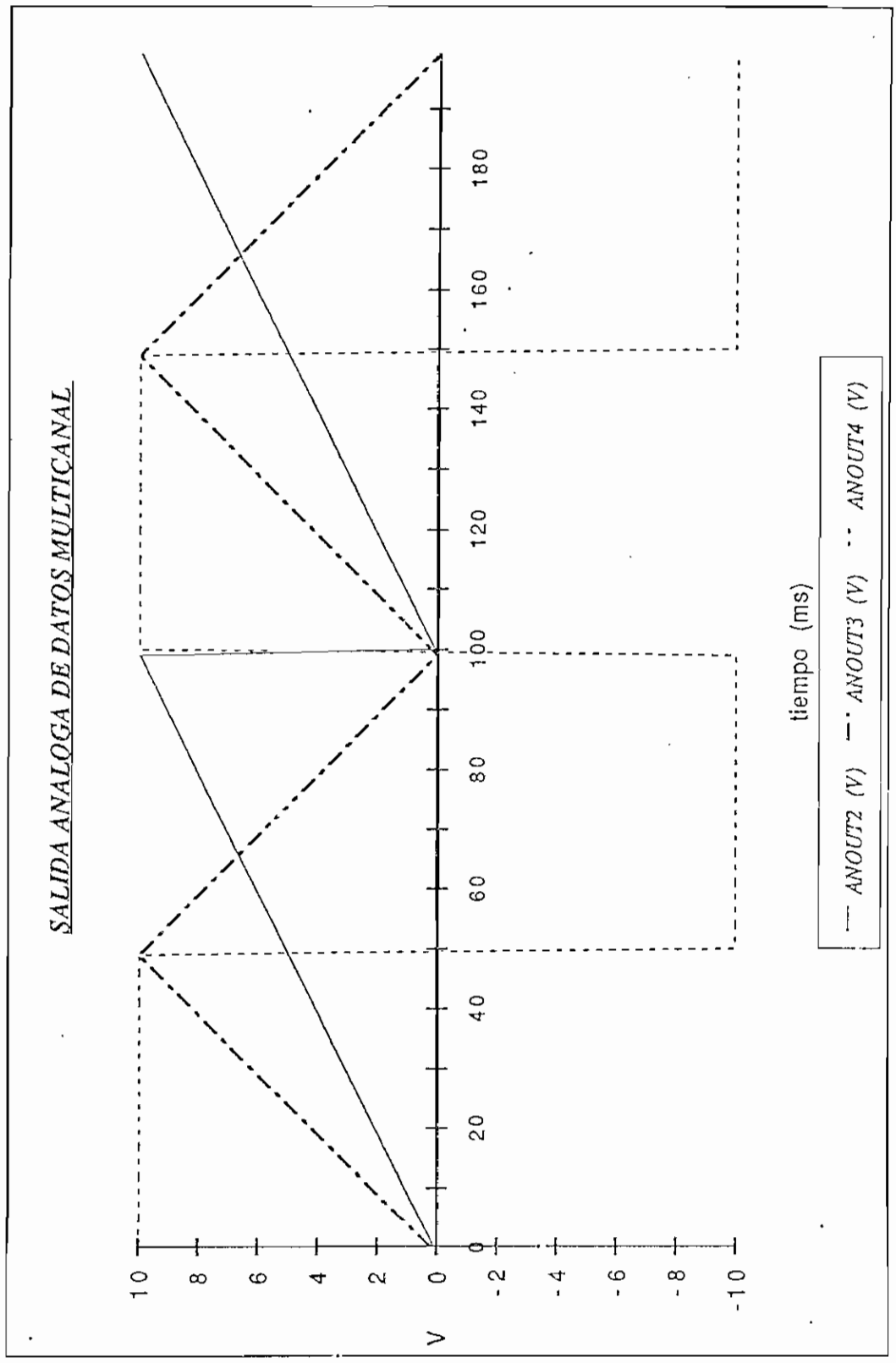


Figura 4.9b.- Salida análoga de datos multicanal.- (canales: ANOUT2, ANOUT3 y ANOUT4)

4.1.3.-SALIDA DE DATOS DIGITALES.-

Para este caso los resultados que se pueden sacar son puramente visuales, tanto en el computador, como mediante un circuito a base de diodos de luz que ayudan precisamente a visualizar los resultados. El esquema de dicho circuito se presenta en la figura 4.10.

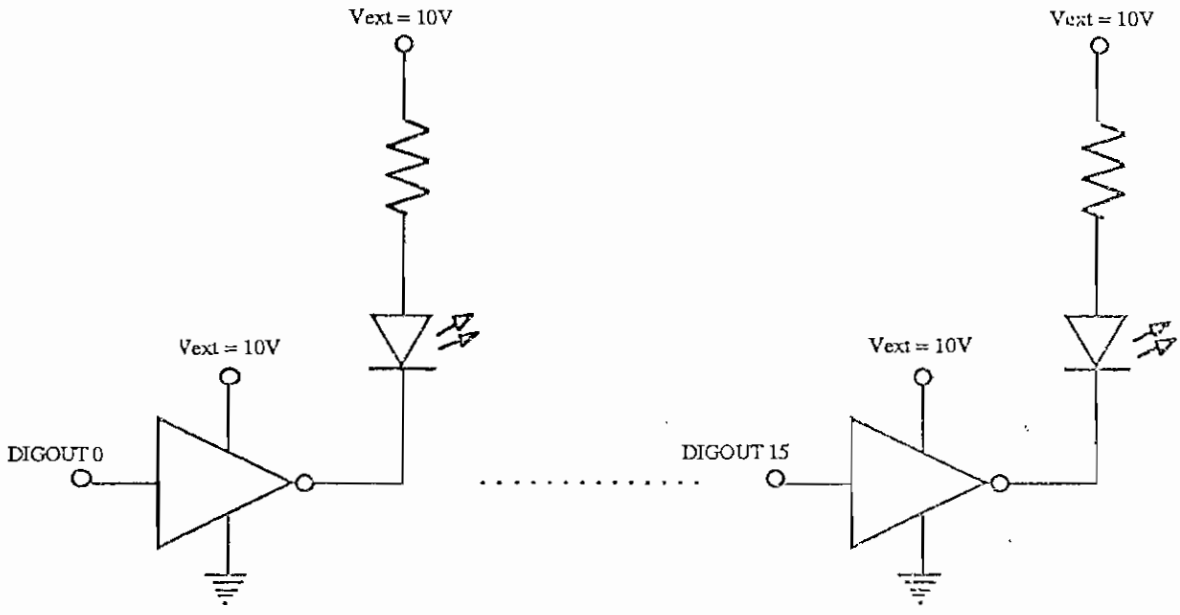


Figura 4.10.- Circuito utilizado para obtención de resultados digitales.

En cada caso se apreciarán rutinas simples que van encendiendo los diferentes leds en base a la fórmula 2^n o en base a $2^n - 1$ ($0 < n < 7$; n entero).

4.1.4.- ENTRADA SALIDA DE DATOS CON ALGORITMO DE CONTROL.

Como se mencionó oportunamente, para la rutina de entrada y salida de datos se implementó un algoritmo que simplemente es una multiplicación por dos, ya que el objetivo es aplicar esta rutina sin tener que realizar cambios de fondo para implementar algoritmos de control. Para probar esta rutina se utilizó el circuitio R-C mostrado en la figura 4.5 alimentado por una señal paso de 4 voltios, los resultados obtenidos se presentan en la figura 4.11.

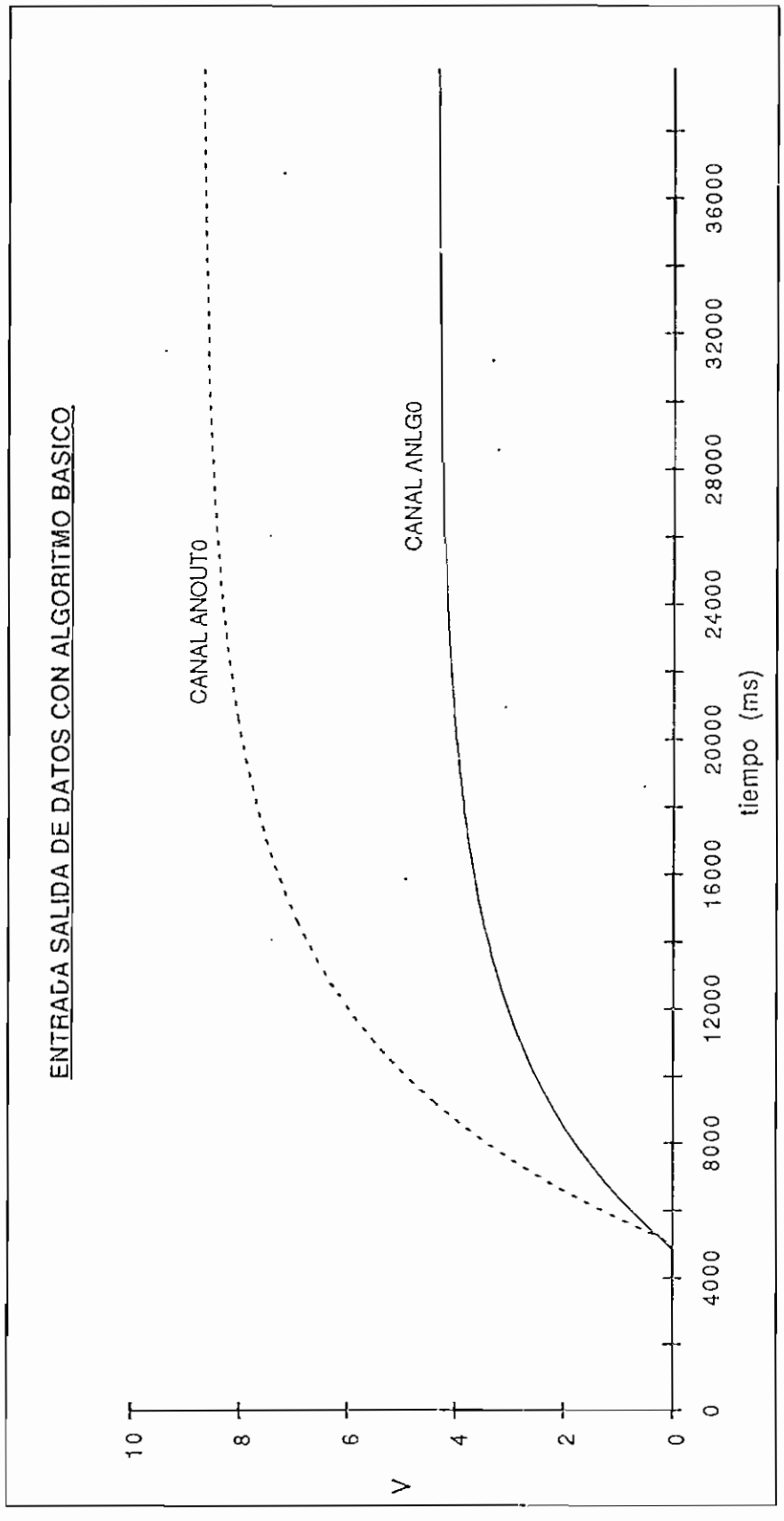


Figura 4.11.- Entrada y salida análoga de datos unicanal.

4.1.5.- CONTROLADOR ON-OFF.-

El objetivo de implementar este controlador fue el de probar la validez de los comandos de disparo del Quick500. Para probar la rutina implementada se aplicó un control ON-OFF sobre el circuito R-C de la figura 4.5, obteniéndose los resultados de la figura 4.12 tanto para la salida del circuito como para el control.

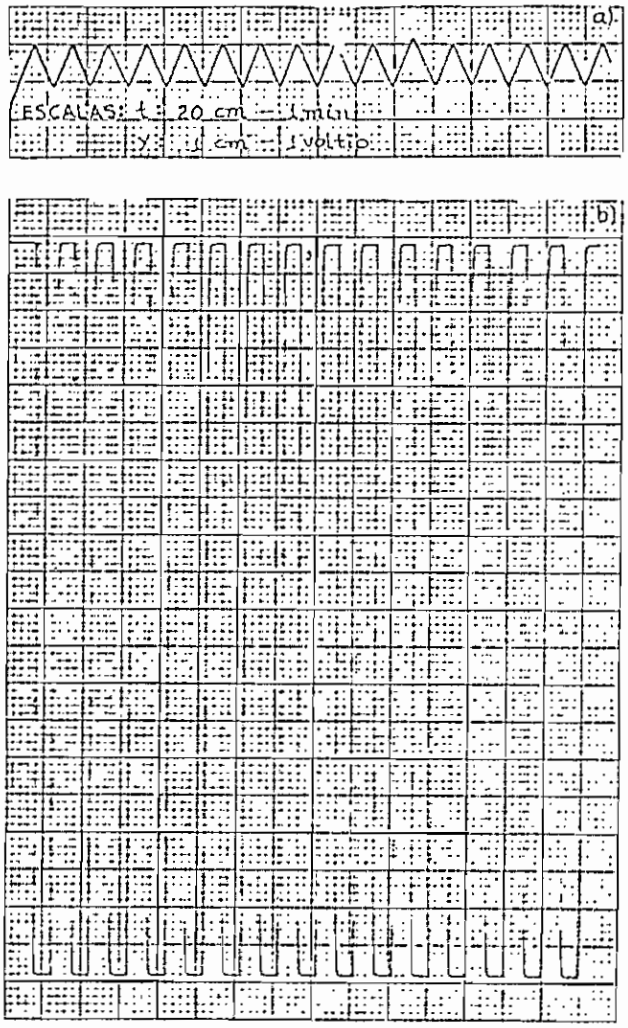


Figura 4.12.- Control ON-OFF (disparos).- a) salida del circuito; b) control.

4.1.6.- GRAFICOS EN TIEMPO REAL.-

Para poder visualizar los resultados que se obtuvieron con la utilización de los comandos que tiene el Quick500 para la graficación en tiempo real se debe necesariamente estar frente al monitor del computador, por lo que no se puede

sino hacer referencia a ellos.

4.1.7.- MANEJO DE ARREGLOS DE MEMORIA PARA DATOS.-

La rutina que se ha desarrollado en el programa presentado, es una recopilación de todas las actividades que se pueden realizar con los arreglos de memoria del Quick500 por lo que presentar resultados de ésta no resulta práctico, sin embargo, como resultados de la validez de estos comandos y su utilización se tiene el adecuado funcionamiento de las rutinas de background, la creación, utilización, borrado de los arreglos de memoria en dichas rutinas y todos los archivos importables a Lotus (posteriormente convertidos a Excel) creados para la presentación de los resultados de las diferentes rutinas, etc..

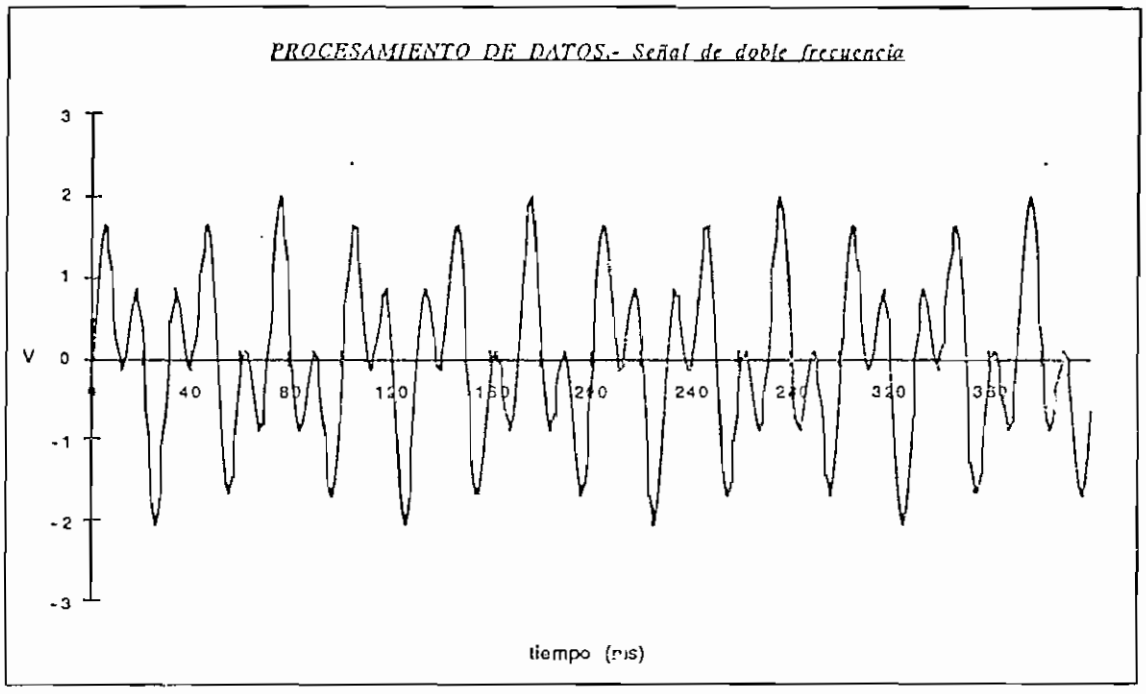
4.1.8.- PROCESAMIENTO DE DATOS.-

Esta rutina realiza procesamiento de tipo estadístico de señales que han sido adquiridas con anterioridad y que se guardaron en disco, o señales que se adquieren en tiempo real en ese mismo instante.

Se obtiene información variada sobre la señal ingresada, todo en base a los comandos del Quick500, obteniéndose resultados satisfactorios en casi todos los casos, con la excepción de la integral, comando que no funcionó, optándose por realizar en BASIC el mismo algoritmo utilizado en el Quick500, esto es la fórmula de Simpson. Se obtuvo resultados sobre algunas ondas, los cuales se muestran en las figuras 4.13, 4.14, 4.15 y 4.16.

4.2 APLICACIONES.-

El presente trabajo de tesis centra su atención en el estudio de las características y potencialidades del sistema de adquisición de datos y control KEITHLEY 500A, presentando para ello los diferentes comandos que conforman

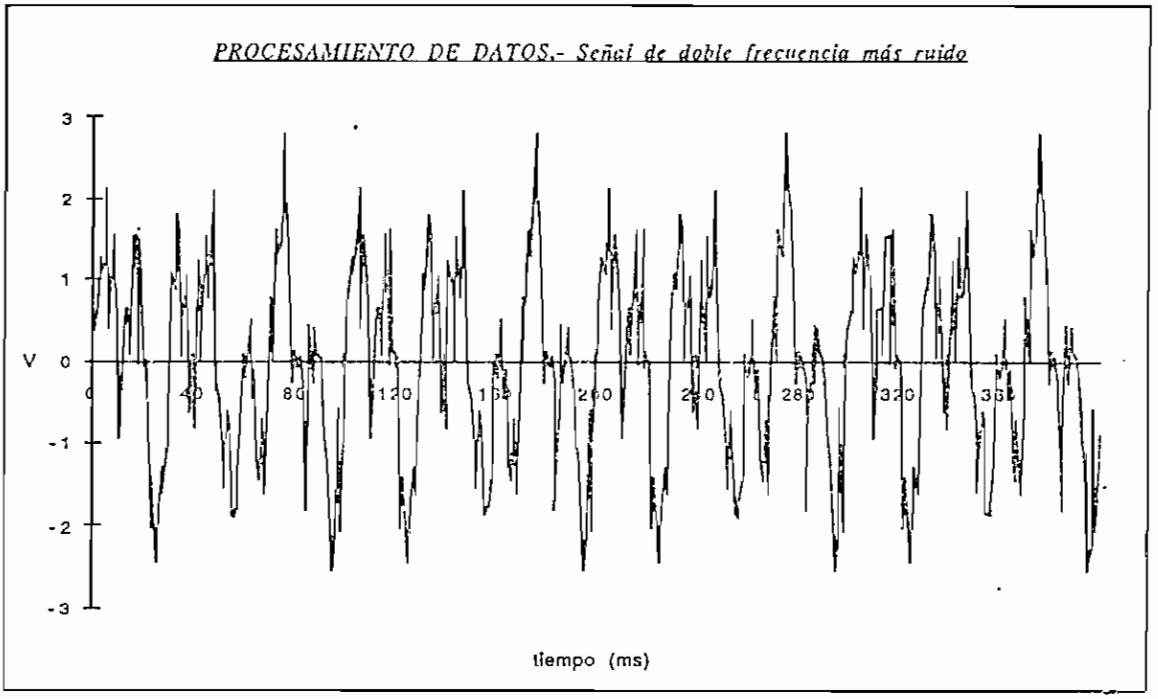


y = Sen (3Hz.) + Sen (7Hz.)

Análisis estadístico:

valor medio	=	-0.0063
desviación estándar	=	1.0006
integral (área)	=	-0.0013
derivada	=	578.7546
recta aproximada	=>	y = -0.001t + 0.106
coef. de determinac	=	0.0042
coef. de correlac	=	0.0652
error estándar	=	0.9997

Figura 4.13.- Procesamiento de datos.- Señal de doble frecuencia.

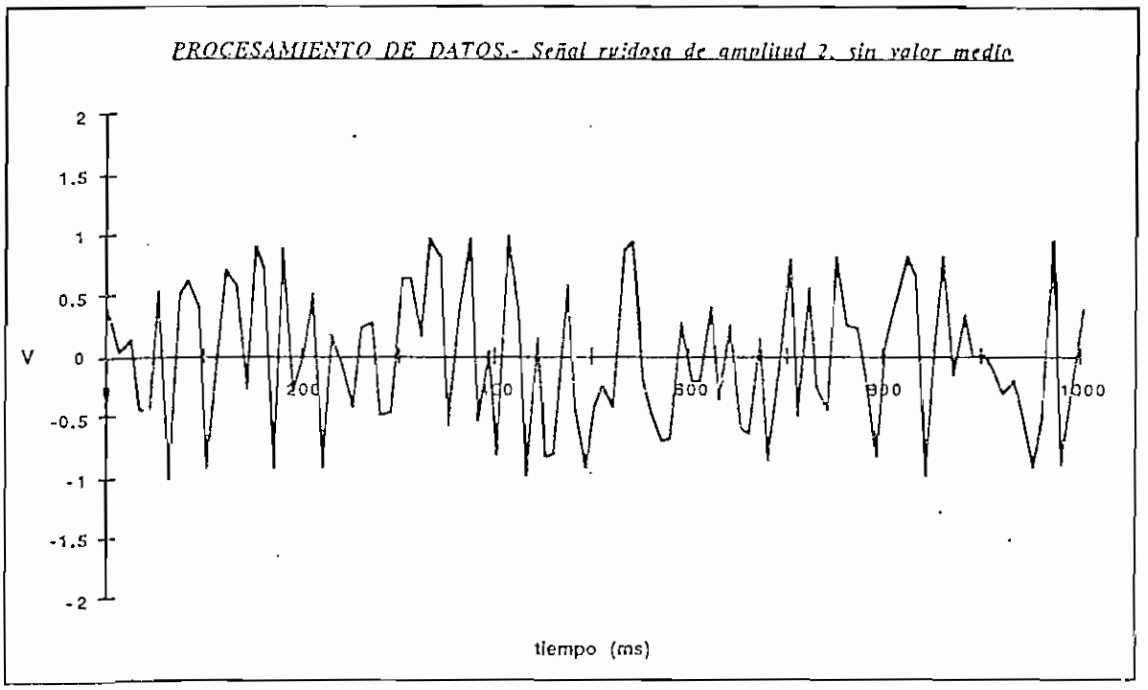


$$y = \text{Sen}\ (3\text{Hz.}) + \text{Sen}\ (7\text{Hz.})\ 2*\text{rnd} - 1$$

Análisis estadístico:

valor medio	=	-0.0078
desviación estándar	=	1.1942
integral (área)	=	0.0033
derivada	=	1221.0012
recta aproximada	=>	y = -0.001t + 0.132
coef. de determinac	=	0.0046
coef. de correlac	=	0.0681
error estándar	=	1.1929

Figura 4.14.- Procesamiento de datos.- Señal de doble frecuencia con ruido

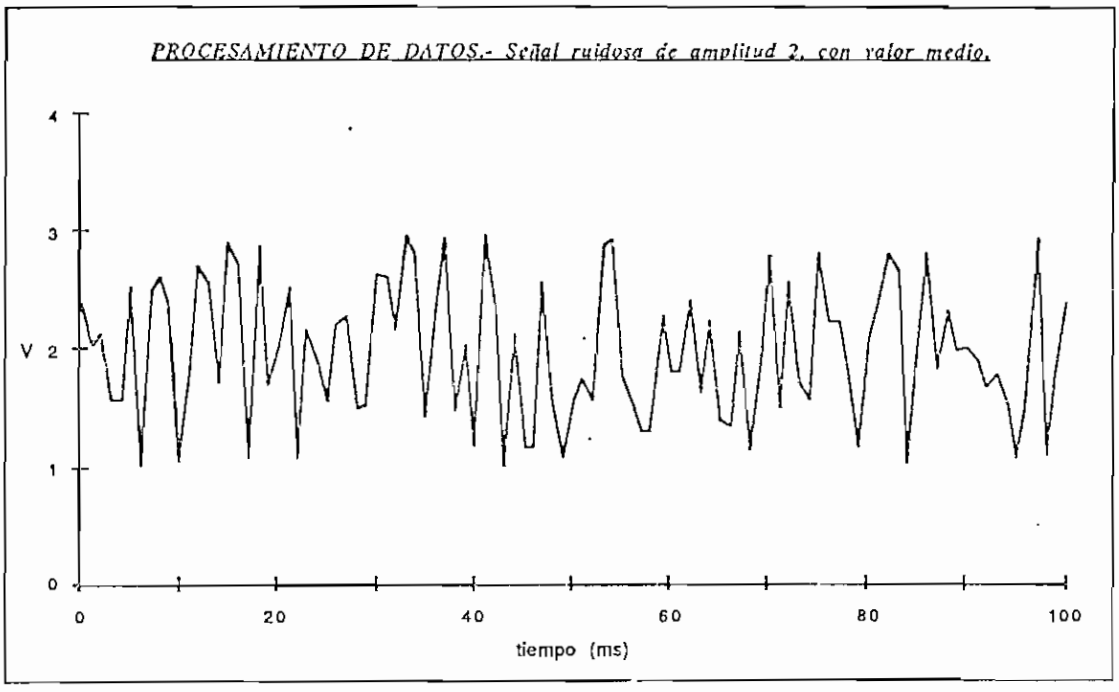


$$y = 2*rnd - 1$$

Análisis estadístico (sobre 400 datos):

valor medio	=	-0.0078
desviación estándar	=	0.5761
integral (área)	=	0.0021
derivada	=	642.2466
recta aproximada	=> y =	-0.000t + 0.020
coef. de determinac	=	0.0008
coef. de correlac	=	0.0282
error estándar	=	0.5766

Figura 4.15.- Procesamiento de datos. Señal ruidosa de valor medio igual a cero.



$$y = 2*rnd + 1$$

Análisis estadístico (sobre 400 datos):

valor medio	=	1.9914
desviación estándar	=	0.5773
integral (área)	=	0.7978
derivada	=	642.2466
recta aproximada	=>	y = -0.000t + 2.019
coef. de determinac	=	0.0008
coef. de correlac	=	0.0281
error estándar	=	0.5778

Figura 4.16.- Procesamiento de datos. Señal ruidosa de valor medio diferente de cero.

su software y desarrollando rutinas que permiten y facilitan el uso del equipo en tareas de instrumentación, procesamiento y control a modo de un manual. Sin embargo para dar una idea más objetiva se incluyen algunas aplicaciones, en las cuales el interés se fija en la utilización de la estación KEITHLEY 500A para aplicar en tiempo real las diferentes técnicas de control. En dichas aplicaciones prácticamente se asume de manera directa el algoritmo, puesto que no es el objetivo que persigue el presente trabajo.

A continuación se presenta una ligera descripción teórica de las diferentes aplicaciones que se incluyen en el presente trabajo. Al finalizar este numeral se presenta el listado completo del "software de aplicación".

4.2.1.- CONTROLADOR P.I.D. DISCRETO.-

Un control PID continuo, es decir en el dominio del tiempo responde a la siguiente ecuación:

$$u = Kp * e + Ki * \int e * dt + Kd * \frac{de}{dt}$$

en la cual se tienen los siguientes parámetros:

u	señal de control
e	error (diferencia entre la referencia y la salida)
Kp	constante proporcional
Kd	constante derivativa
Ki	constante integral

Al sacar la transformada de Laplace de la anterior expresión, el controlador PID responde a la ecuación:

$$u(s) = \left[Kp + \frac{Ki}{s} + Kd * s \right] * e(s)$$

Para la discretización del controlador, se toma la transformada Z de la

expresión anterior de la siguiente manera:

Para el integrador se utiliza el método de aproximación de una integral de la suma de los trapecios donde se tiene que:

$$s=\frac{2}{T}*\frac{z-1}{z+1}$$

Para el derivador se utiliza la discretización mediante la equivalencia del ZOH (zero order hold), obteniéndose:

$$\frac{u(z)}{e(z)}=\frac{Kd}{T}*(1-z^{-1})$$

La expresión de la función de transferencia entre la ley de control y el error es:

$$\frac{u(z)}{e(z)}=Kp+Ki*\frac{T*(z+1)}{2*(z-1)}+\frac{Kd}{T}*(1-z^{-1})$$

En la cual T es el período de muestreo. Si a la expresión anterior se le saca la transformada Z inversa, se obtiene la siguiente ecuación de diferencias:

$$u(k) = u(k-1) + b1*e(k) + b2*e(k-1) + b3*e(k-2)$$

donde:

$$b_1=Kp+Ki\frac{T}{2}+\frac{Kd}{T}$$

$$b_2=-Kp+Ki\frac{T}{2}-2*\frac{Kd}{T}$$

$$b_3=\frac{Kd}{T}$$

u(k) y e(k) son los valores actuales de la ley de control (salida del controlador) y el error (entrada del controlador).

u(k-1) y e(k-1) son los valores anteriores de las mismas señales.

e(k-2) es el error existente dos períodos de muestreo antes del actual.

Un diagrama de bloques que describe el control mediante un PID discreto se presenta en la figura 4.17

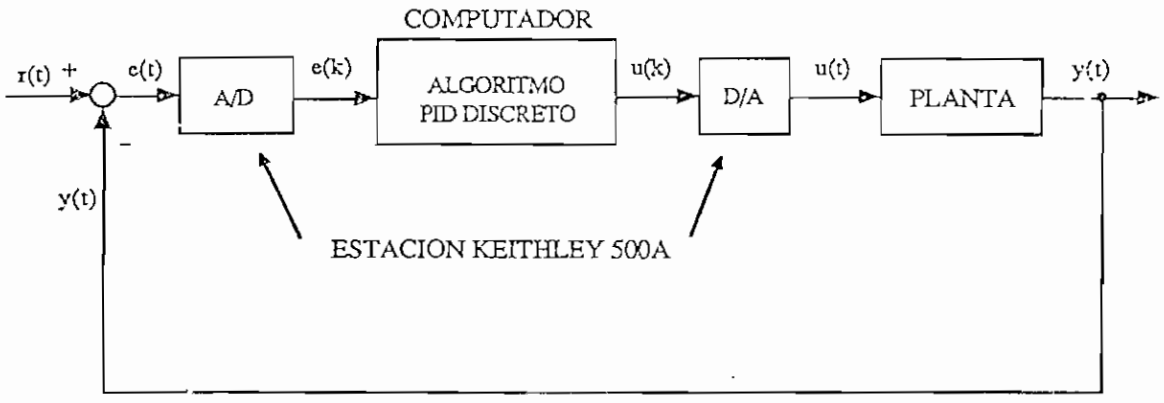


Figura 4.17.- Diagrama de bloques: Controlador PID discreto.

LISTADO DE LA RUTINA.- En el numeral 4.2.7, se presenta el listado completo del "software de aplicación", en el cual, el controlador PID discreto corresponde a la subrutina "PID". Esta rutina fue desarrollada en la base de la rutina "INOUTGR" presentada en el capítulo 3, por lo que el diagrama de flujo para este caso es similar, pues en el bloque del algoritmo de control va el controlador PID.

RESULTADOS.- Para probar este controlador se lo aplicó al circuito R-C de la figura 4.5, utilizando los siguientes parámetros:

Kp = 5
Ki = 2
Kd = 0
T = 500 ms.
r = 6 volt.

Se obtuvieron resultados tanto en el registrador como mediante datos almacenados utilizando la hoja de cálculo. En la figura 4.18 se muestra la respuesta del circuito con dicho controlador mediante el registrador.

La figura 4.19 muestra la ley de control que sale a través del canal ANOUT0 de la estación KEITHLEY 500A e ingresa al circuito.

Estos mismos resultados se obtienen de los datos adquiridos y almacenados y se muestran en las figuras 4.20 y 4.21.

4.2.2.- REALIMENTACION DE ESTADO.-

Los métodos convencionales de control como el del lugar de la raiz y la respuesta de frecuencia son bastante útiles para tratar sistemas de una sola entrada y una sola salida. En cambio cuando se tiene una planta de múltiples entradas y salidas es conveniente utilizar las técnicas del control moderno basada netamente en el dominio del tiempo. Estas técnicas de control moderno requieren de una modelación del sistema mediante variables de estado.

Para el presente trabajo se ha implementado una realimentación de estado según el diagrama de bloques que se presenta en la figura 4.22, en la cual la matriz de realimentación [K] es calculada OFF-LINE (fuera del lazo de control).

LISTADO DE LA RUTINA.- En el numeral 4.2.7, se presenta el listado completo del "software de aplicación", en el cual, la realimentación de estado se encuentra en la rutina "REALEST". Esta rutina fue desarrollada en la base de la rutina "INOUTGR" presentada en el capítulo 3, pero para el caso multivariable, sin embargo, el diagrama de flujo para este caso similar, calzando el algoritmo de control tal como en el caso univariable, pero tomando en cuenta que ahora se utilizan entradas y salidas multicanales.

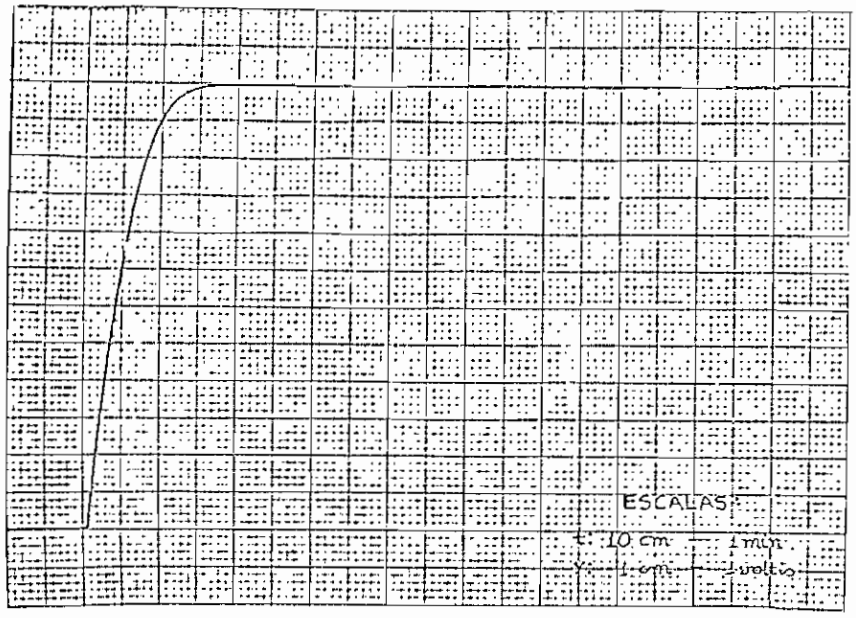


Figura 4.18.- Controlador PID.- Salida de la planta.

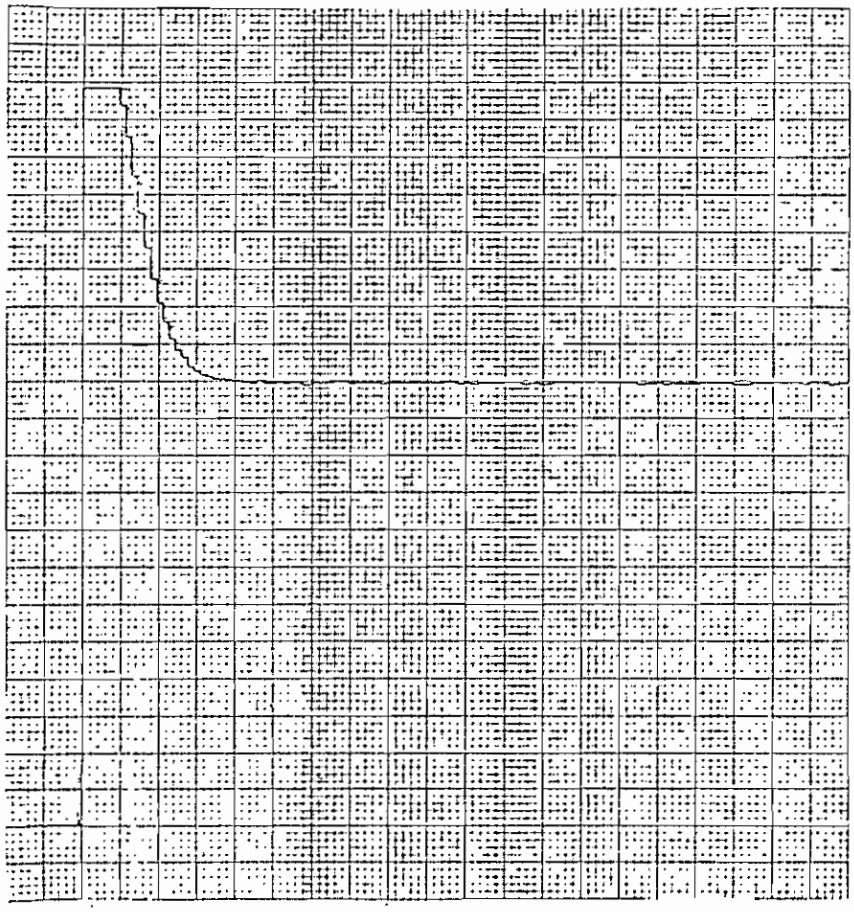


Figura 4.19.- Controlador PID.- Ley de control.

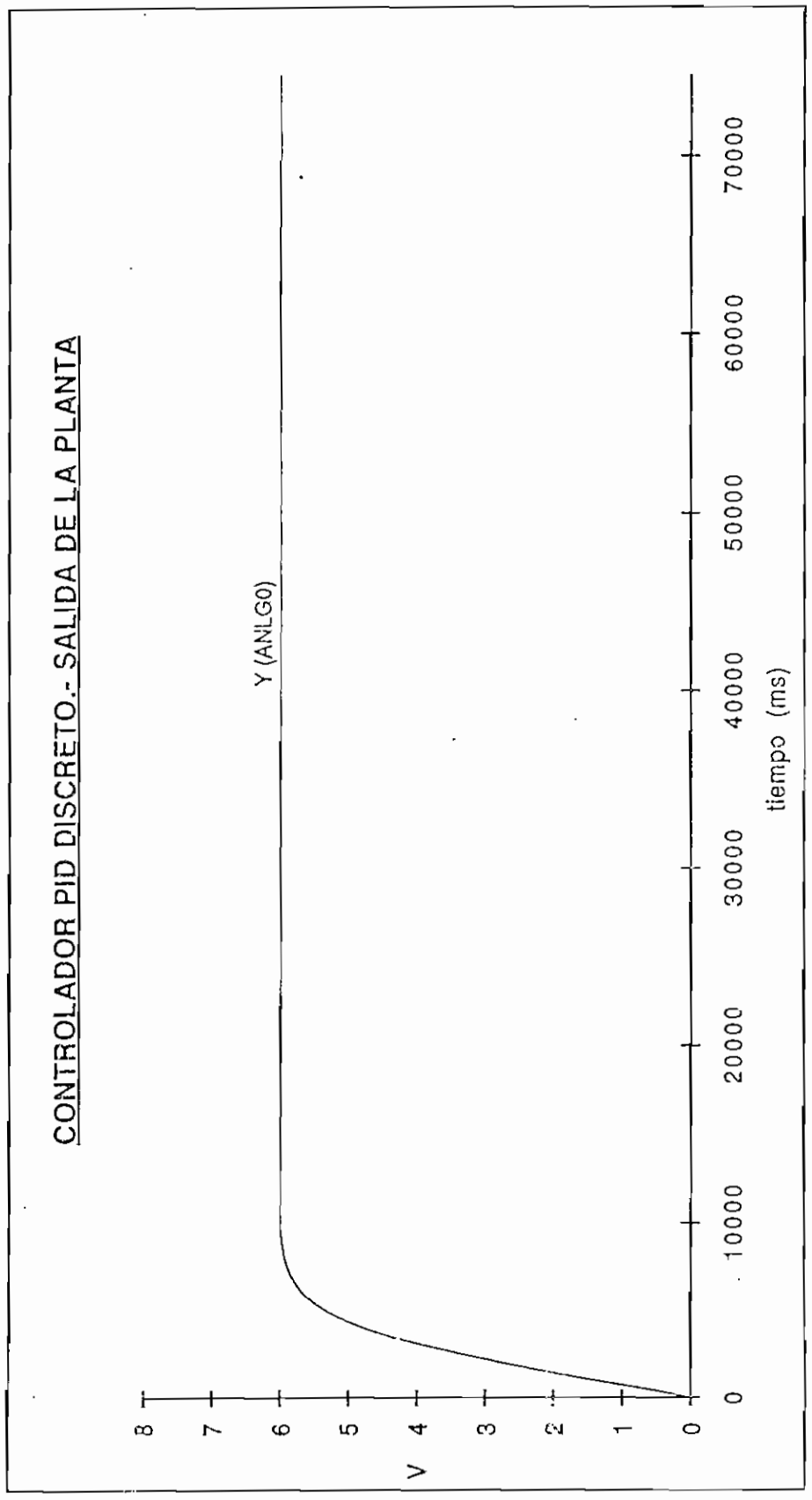


Figura 4.20.- Controlador P.I.D. discreto.- Salida de la planta

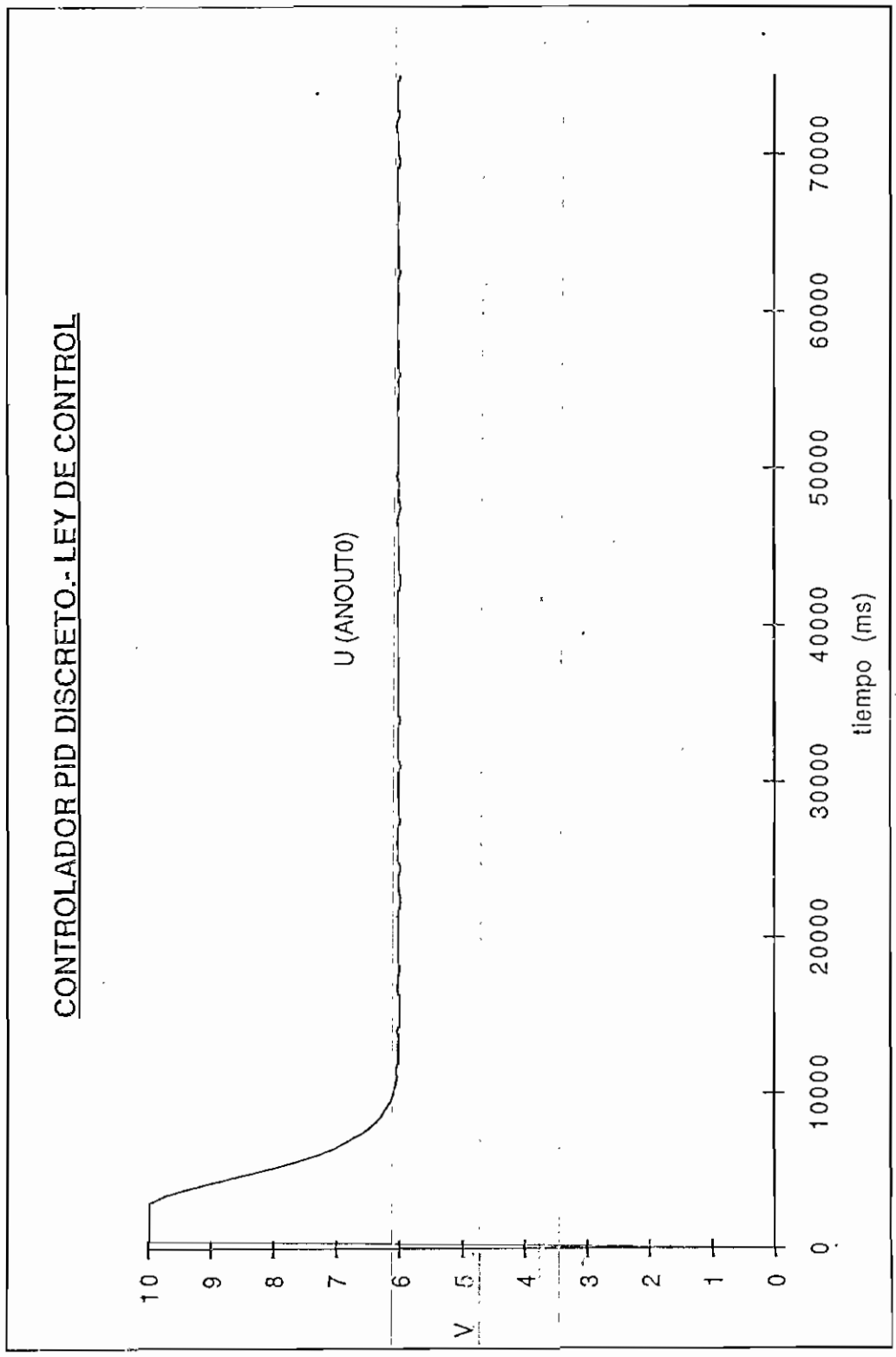


Figura 4.21.- Controlador P.I.D. discreto.- Ley de Control.

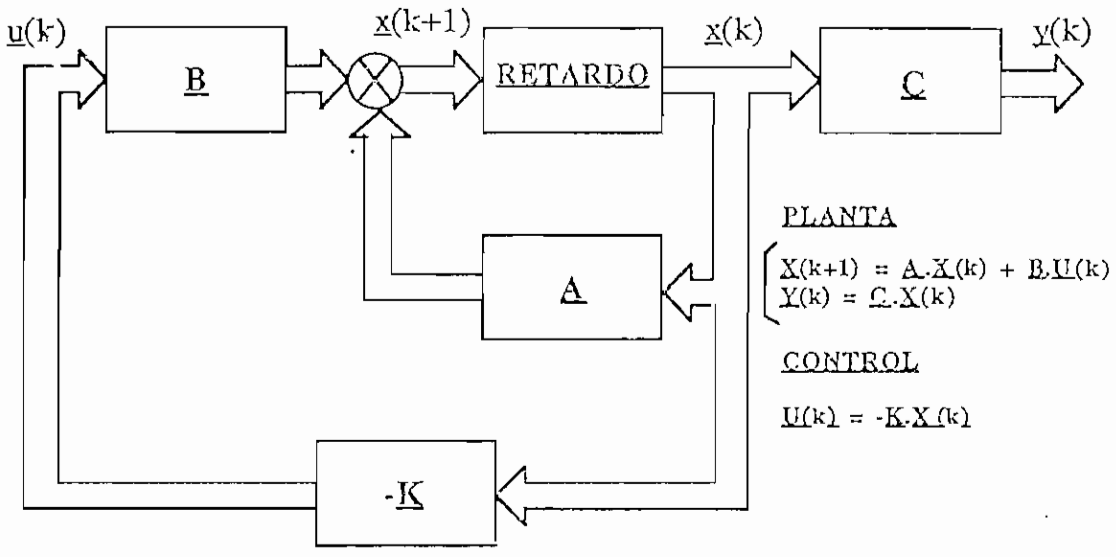


Figura 4.22.- Realimentación de estado discreta.

RESULTADOS.- La rutina de realimentación de estado se aplicó sobre el circuito de segundo orden de la figura 4.25, del cual se obtuvieron los resultados que se muestran en las figuras 4.23 y 4.24, en las que se muestra las salidas del sistema y las leyes de control respectivamente. Dichos resultados se obtuvieron aplicando una matriz de realimentación unitaria, que ilustra el funcionamiento del programa, sin poner énfasis en la asignación de polos para mejorar la respuesta transitoria.

$$K = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Se presentan resultados obtenidos mediante el registrador y mediante datos acumulados e importados a la hoja de cálculo,

4.2.3.- CONTROL ON-OFF EN 3 NIVELES.- El control de tres niveles no es otra cosa que una extensión del control ON-OFF, el mismo que desarrolla su control con dos niveles de voltaje. Un control ON-OFF de dos niveles responde a las

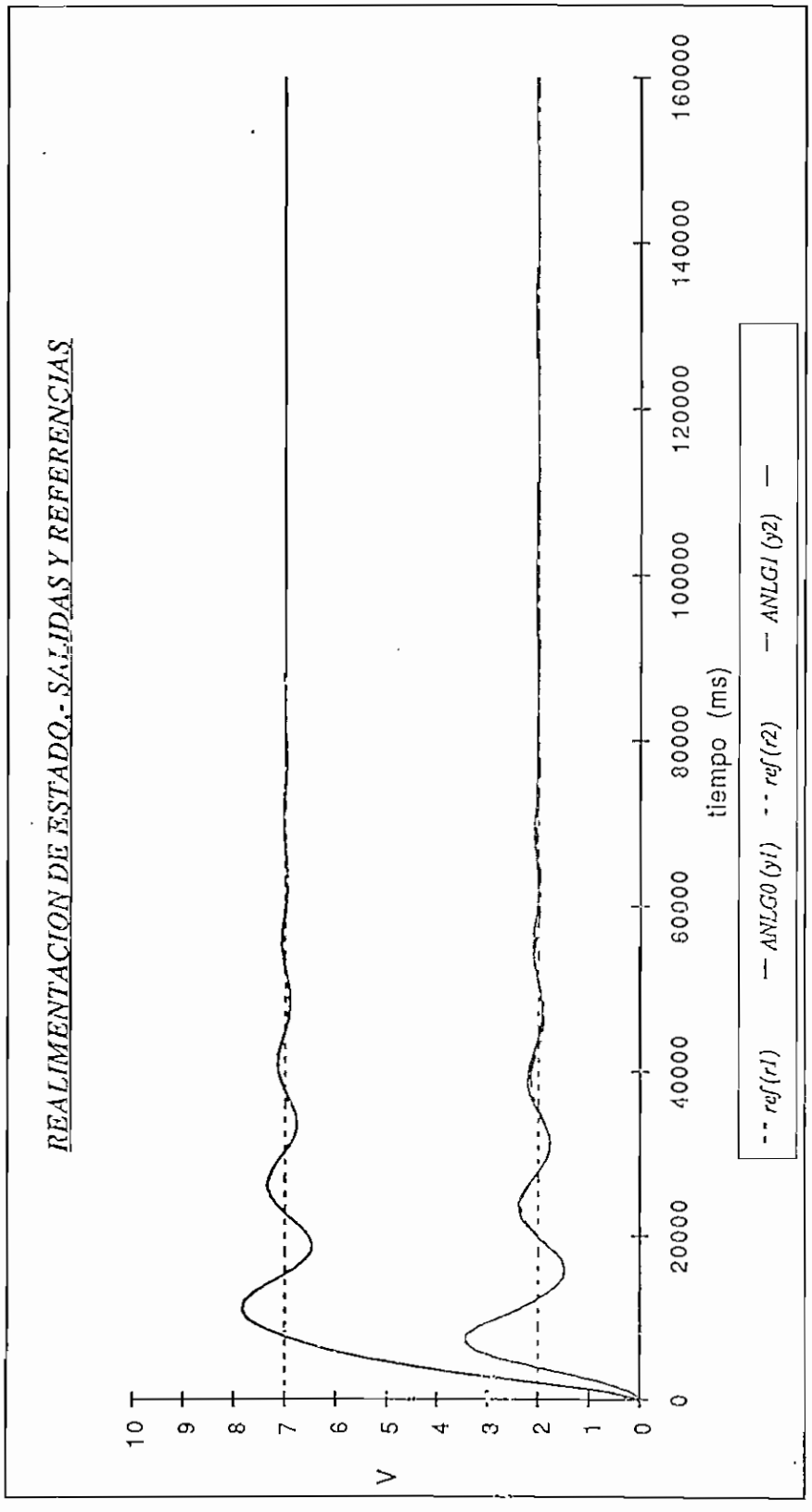


Figura 4.23.- Realimentación de estado.- Salidas de la planta.

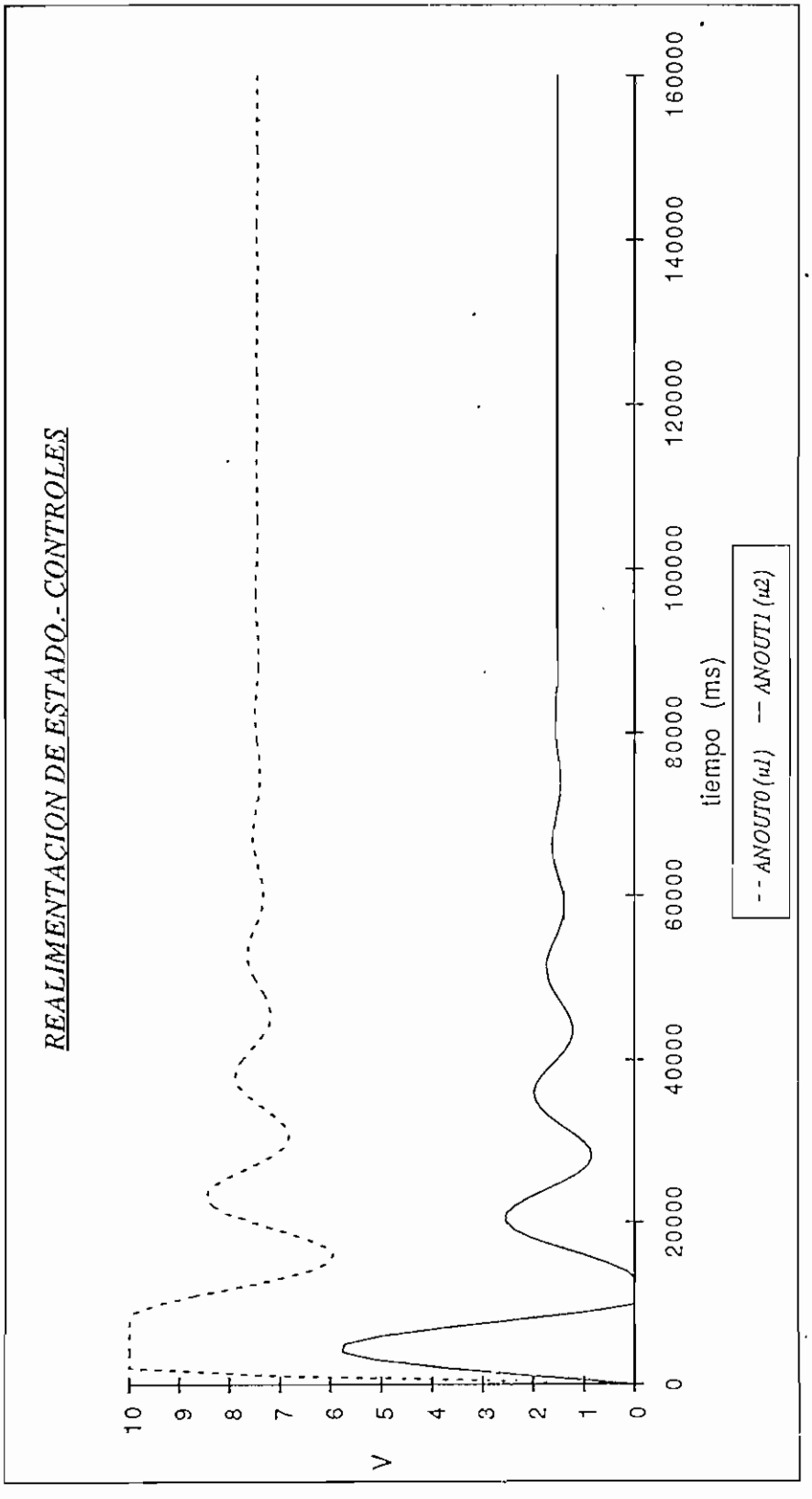


Figura 4.24.- Realimentación de estado.- Leyes de control.

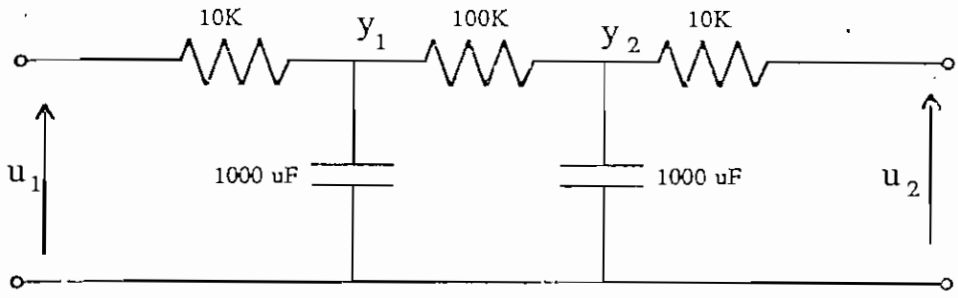


Figura 4.25.- Circuito R-C de segundo orden utilizado.

siguientes ecuanciones:

si y > vmáx. -> u = Vl (valor bajo)

si y < vmín. -> u = Vh (valor alto)

Lo que se representa mediante el gráfico de la figura 4.26.a. Para ampliar esto a un control de tres niveles o más se debe aumentar las condiciones de cambio de voltaje de acuerdo con ciertas condiciones en la señal de salida, las mismas que imponen una nueva entrada a la planta, para este caso a manera de ejemplo se han utilizado las siguientes ecuaciones:

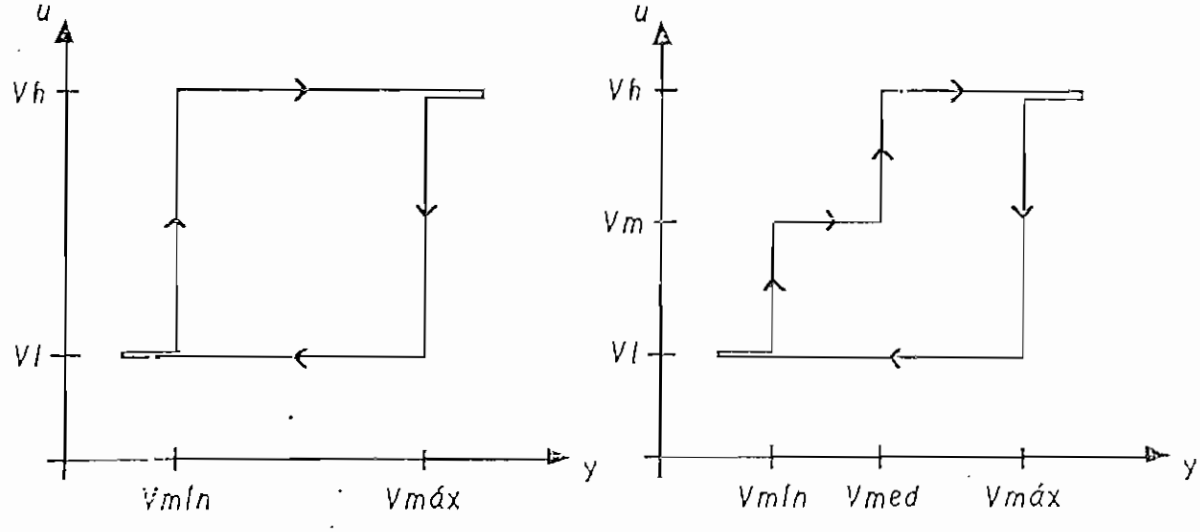


Figura 4.26.- a) Control ON-OFF : b) Control ON-OFF tres niveles

si y > vmed. -> u = Vh (valor alto)

si y > vmáx. -> u = Vl (valor bajo)

si y < vmín. -> u = Vm (valor medio)

LISTADO DE LA RUTINA.- En el numeral 4.2.7, se presenta el listado completo del "software de aplicación", en el cual, los controles ON-OFF y 3 niveles son las subrutinas "ONOFDIS" y "TRESNIV". El diagrama secuencial de tareas del control ON-OFF se presenta en la figura 3.4.

RESULTADOS.- El controlador de 3 niveles de voltaje se aplicó sobre el circuito R-C mostrado en la figura 4.5, obteniéndose los resultados que se muestran en las figuras 4.27 (salida del circuito) y figura 4.28 (ley de control).

4.2.4.- CONTROL ADAPTIVO CON REFERENCIA A MODELO.-

El control adaptivo con referencia a modelo, tiene como objetivos los dos problemas de control, el seguimiento y la regulación. Seguimiento cuando se cambia la referencia del modelo y la regulación cuando el sistema sufre alguna perturbación o cuando se cambian los parámetros de la planta.

El control adaptivo con referencia modelo responde al diagrama de bloques general que se presenta en la figura 4.29, en el cual cada instante se actualiza la ley de control en base a identificación de los parámetros de la planta (en función de los parámetros de entrada y salida) y al modelo de referencia. Para el ejemplo que se desarrolla en este numeral se obvia el cálculo de los parámetros de la planta y directamente se calcula los parámetros del control lo que se conoce como método directo o implícito. (El algoritmo se ha tomado de la tesis del Ing. Hugo Ortiz "Control adaptivo con modelo de referencia para sistemas discretos" y de la tesis del Sr. Jimy Ponce, "Control adaptivo en tiempo real" actualmente en desarrollo, con la autorización de los respectivos autores).

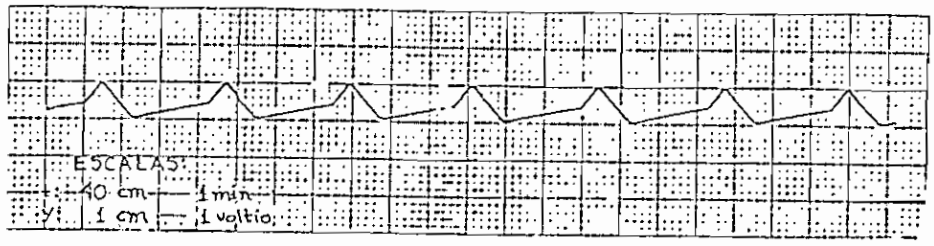


Figura 4.27.- Control ON-OFF tres niveles de voltaje.- Salida de la planta

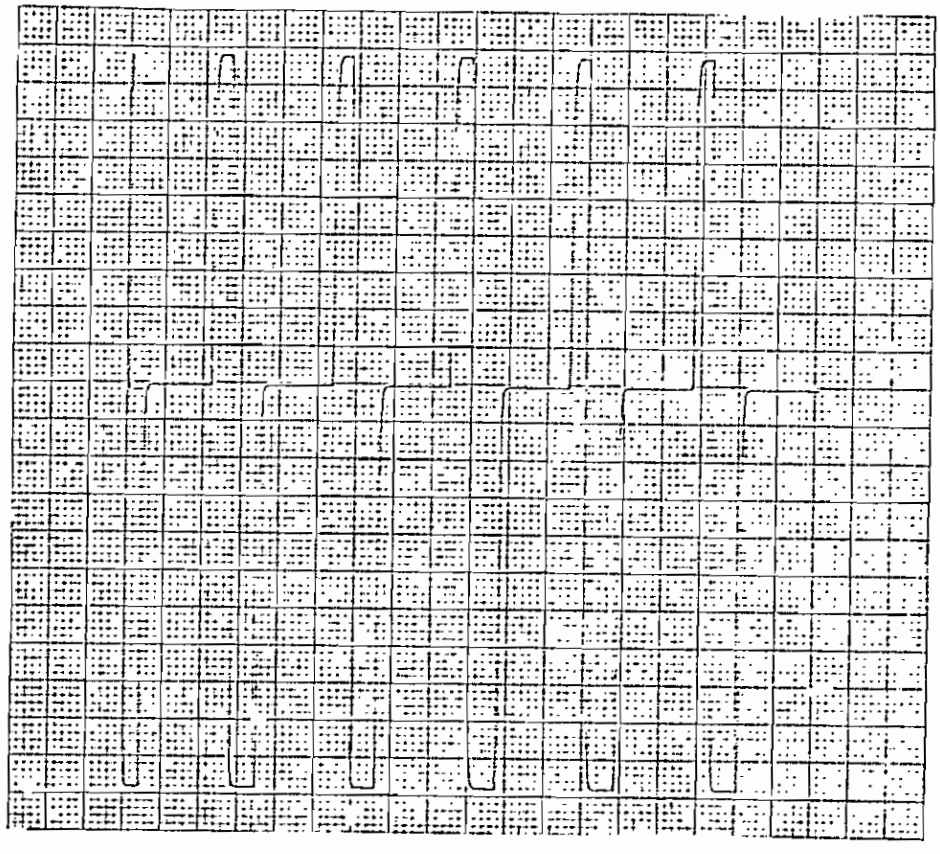


Figura 4.28.- Control ON-OFF tres niveles de voltaje.- Ley de control

LISTADO DE LA RUTINA.- En el numeral 4.2.7, se presenta el listado completo del "software de aplicación", en el cual, el control adaptivo con referencia modelo se ha implementado en la subrutina "IDENT".

RESULTADOS.- Se realizó pruebas de este controlador sobre el circuito R-C que se muestra en la figura 4.30 realizando pruebas de cambio de referencia y además cambio de los parámetros de la planta incluyendo y posteriormente

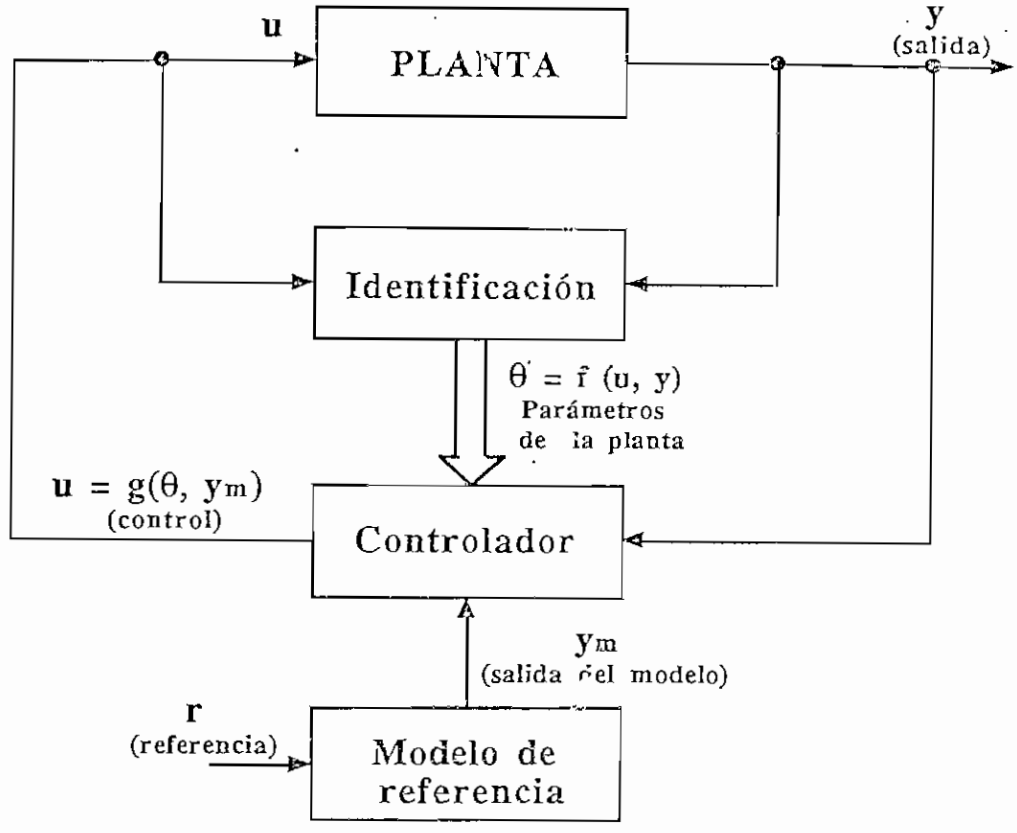


Figura 4.29.- Diagrama de bloques.- Control Adaptivo con referencia a modelo.

retirando una resistencia de 5.6K en paralelo con el condensador de la planta. Los resultados se muestran tanto en el registrador como en hoja de cálculo.

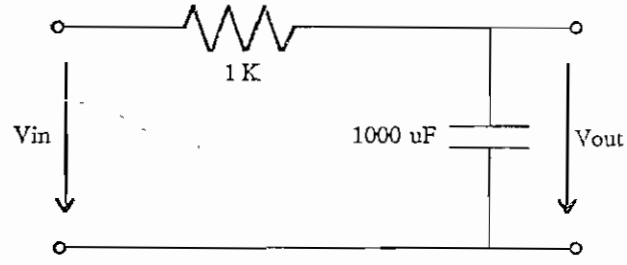


Figura 4.30.- Circuito R-C utilizado para la aplicación del algoritmo MRAS.

En la figura 4.31 se muestra la salida de la planta capturada mediante el

registrador.

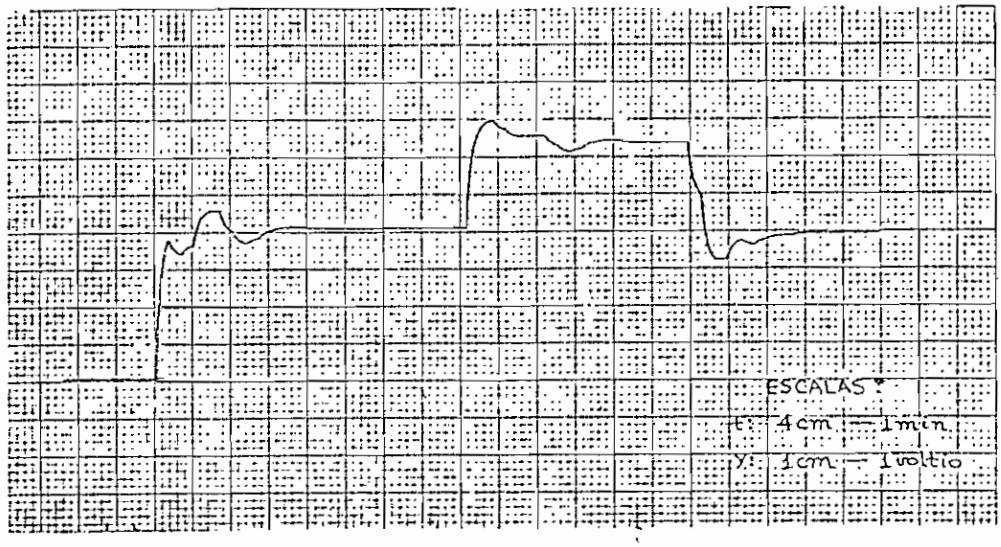


Figura 4.31.- Salida de la planta con la aplicación del algoritmo del MRAS.

La figura 4.32 presenta la salida de la planta, mientras que la ley de control se presenta en la figura 4.33, en la cual se indican los instantes en que se han producido cambios de referencia o cambios de parámetros de la planta.

Los datos con que se corrió éste ejemplo son los siguientes:

Orden del polinomio A (A(z).y = B(z).u)	:	2
Orden del polinomio B	:	2
Orden polinomio de asignación de polos (C)	:	1
C_1 => $C_1 = z-0.2$	:	-.2
Período de muestreo	:	2500 (ms)

4.2.5.- CONTROL DE UN MOTOR A PASOS.-

En muchas situaciones en las que se requiere de un control preciso de la trayectoria a seguir por la mano o la herramienta de un robot manipulador, o en periféricos de un computador como drives e impresoras, es más sencillo y

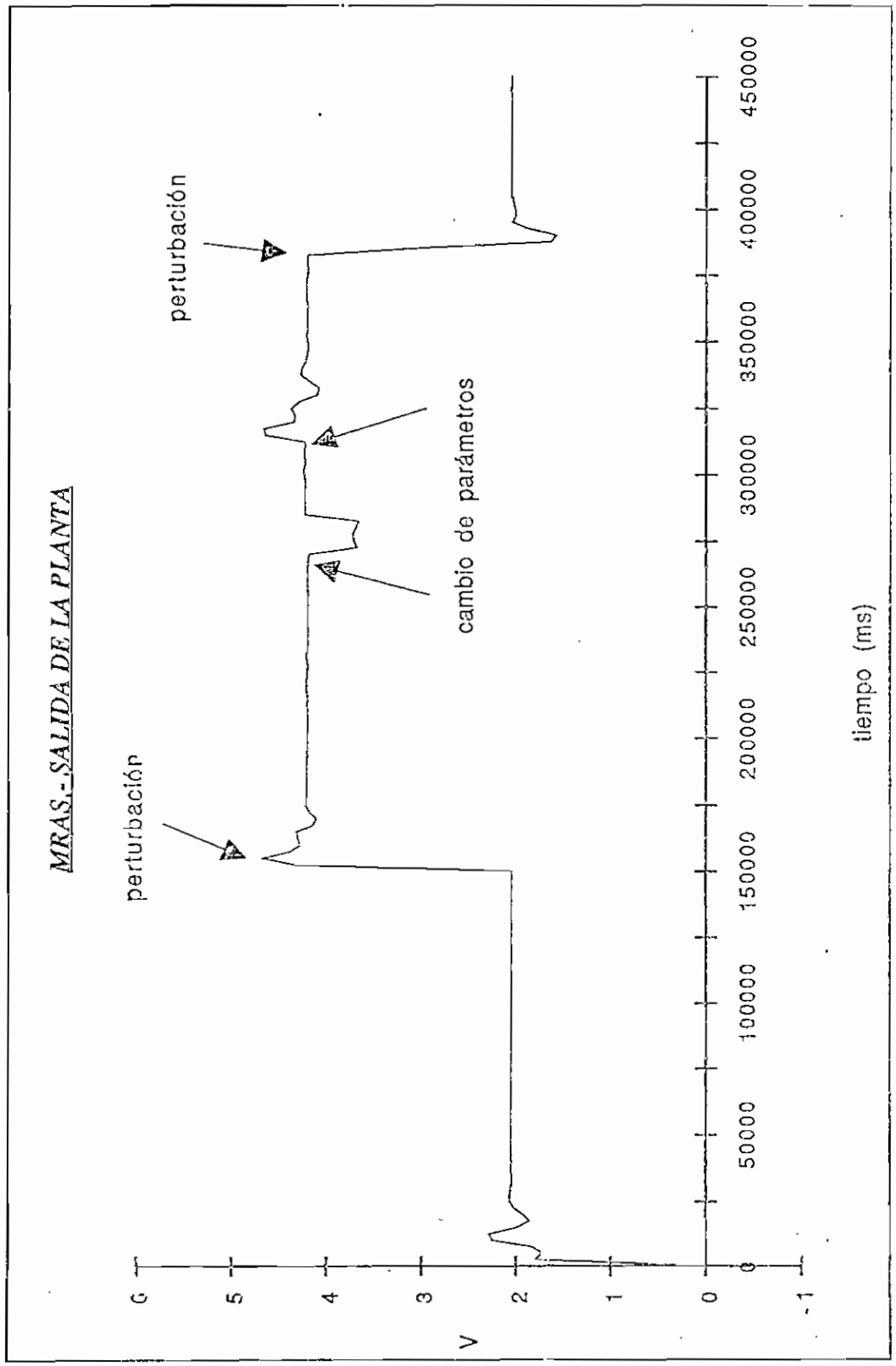


Figura 4.32.- Control adaptivo con referencia modelo.- Salida de la planta

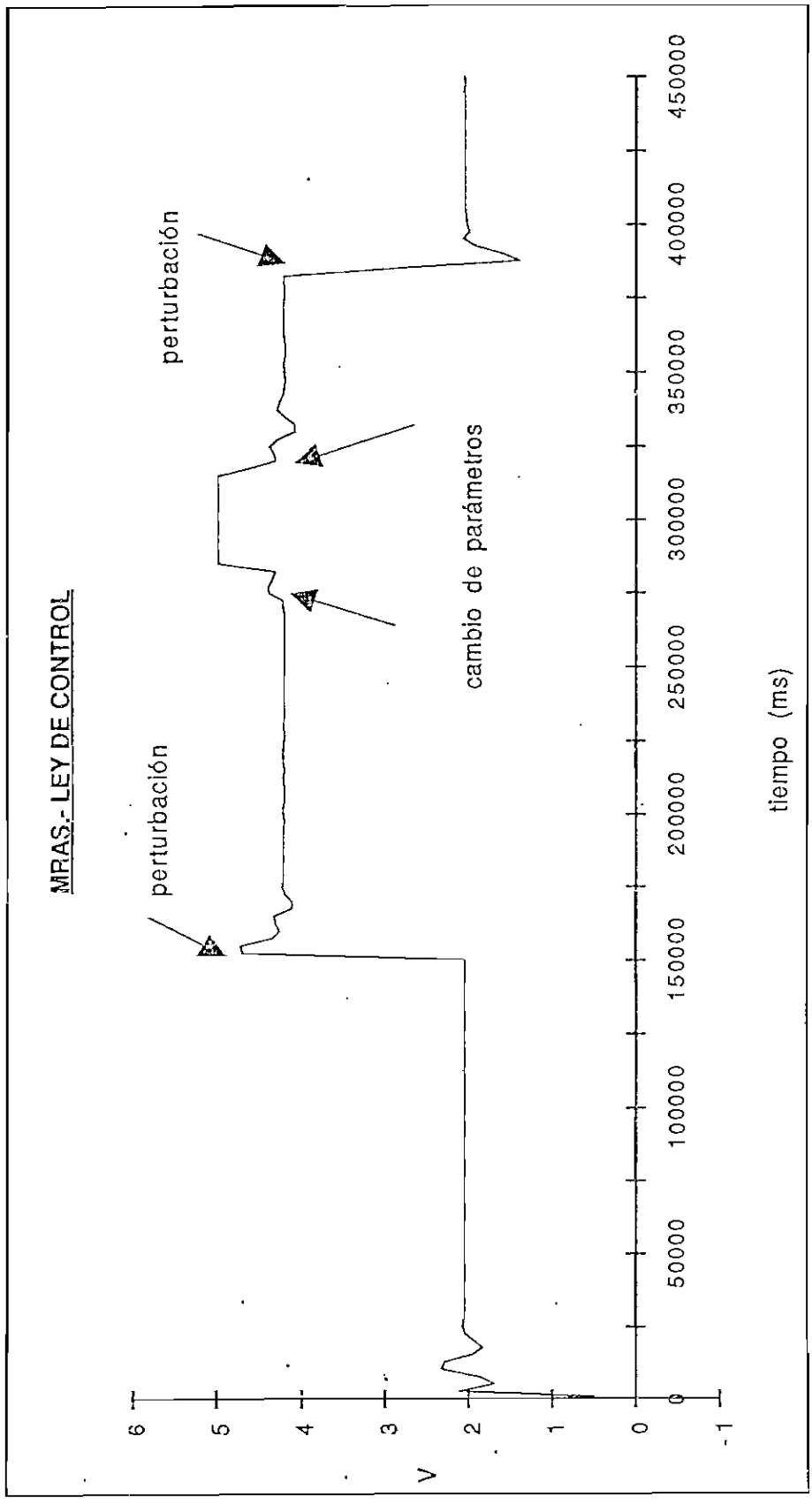


Figura 4.33.- Control adaptivo con referencia modelo.- Ley de control.

económico usar motores de paso que servomotores de corriente continua con realimentación. El único problema de los motores a paso es la limitación de potencia, sin embargo en cierto sentido este problema está siendo superado. El motor de pasos es un elemento capaz de transformar pulsos eléctricos (información digital) en movimientos mecánicos; el eje del motor gira un determinado ángulo por cada impulso de entrada. El resultado de este movimiento, fijo y repetible, es un posicionamiento preciso y fiable. Aunque el motor a pasos es de concepción antigua, solo se ha empleado en la práctica, a partir de la aparición de los semiconductores electrónicos que permiten implementar circuitos adecuados para la actuación y control.

El principio de funcionamiento es sencillo, pues se basa en las fuerzas de atracción y repulsión ejercidas entre polos magnéticos. Si el estator consta de 4 polos de electroimán como se muestra en la figura 4.34, y el rotor es un imán

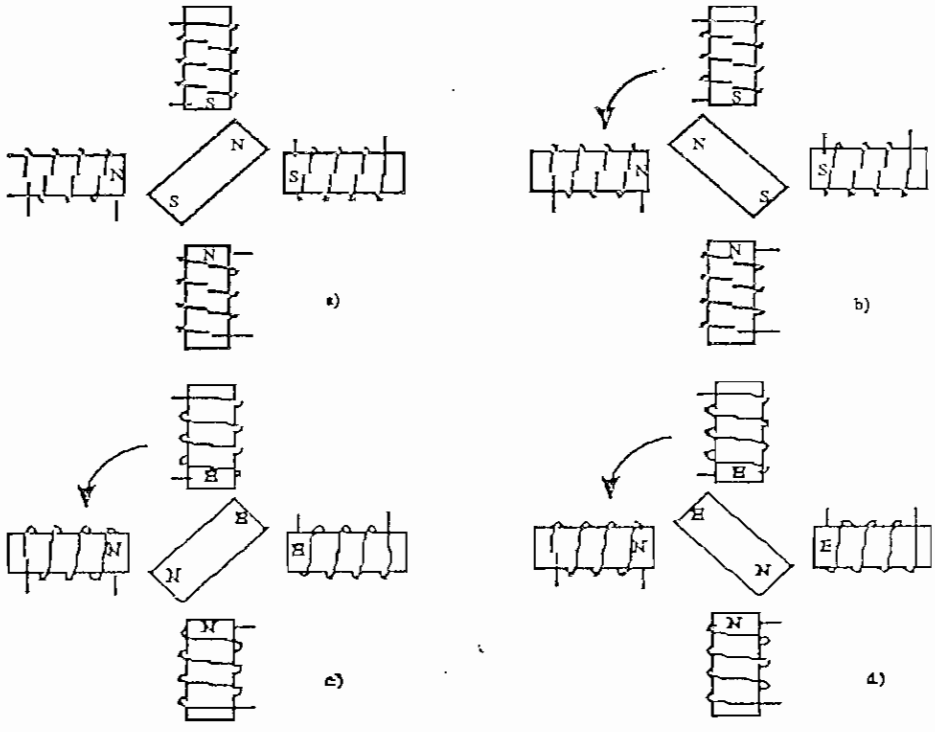


Figura 4.34.- Principio de funcionamiento: motor a pasos de imán permanente de 2 fases.

permanente, al cambiarse la polaridad de los polos del primero, mediante un control externo con la secuencia indicada, el segundo gira en sentido contrario a las agujas del reloj, con incrementos de 90 grados. (Curso de Robótica.- J. M. Angulo pgs. 207-212)

Para el presente trabajo se cuenta con un motor de operación bipolar, es decir que consta de 2 bobinas, al cual se le debe aplicar dos ondas cuadradas desfasadas un cuarto de período entre si (una a cada bobina), como se muestra en la figura 4.35 con lo cual se tiene 4 pasos por cada polo que tenga el estator del motor.

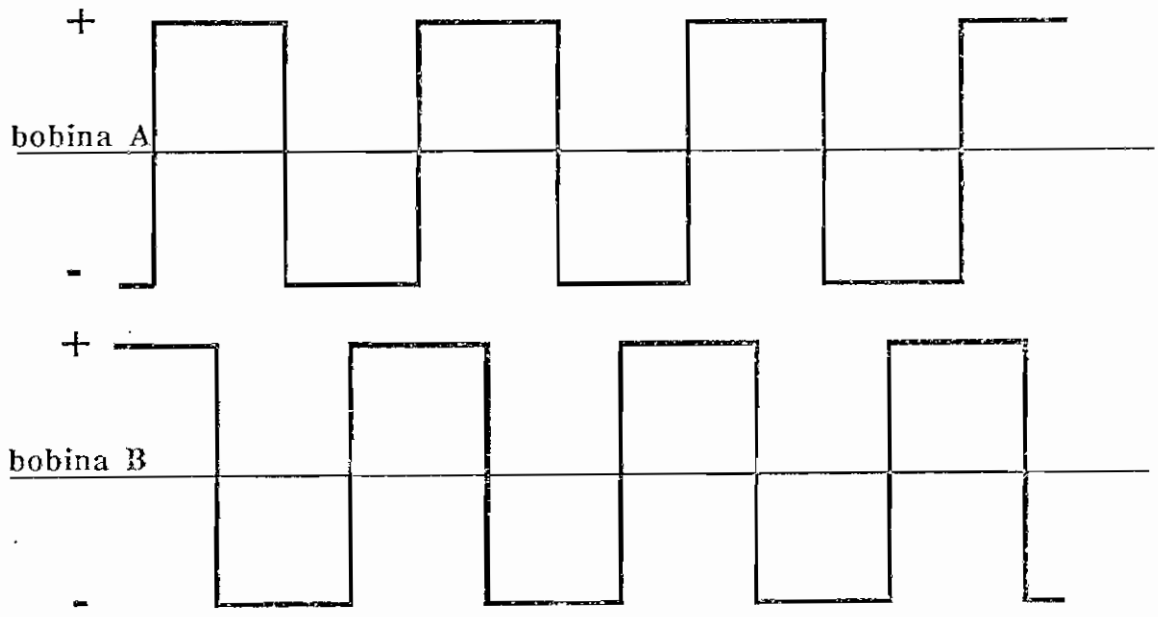
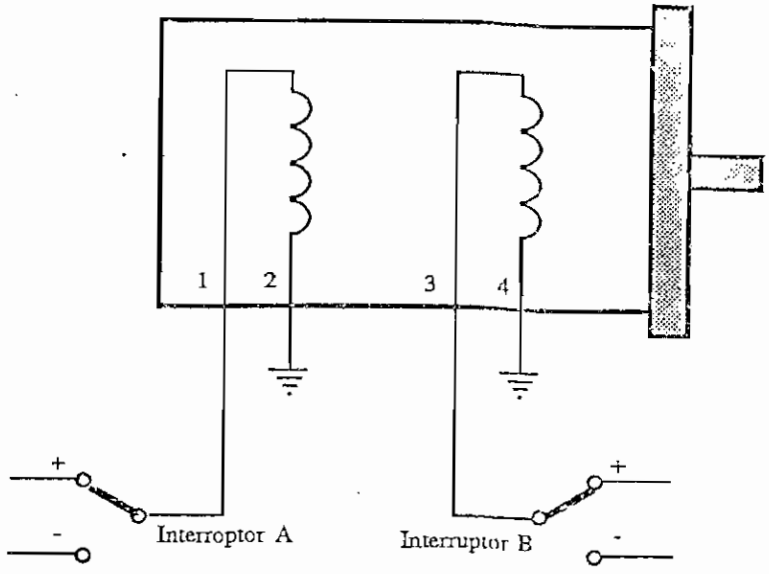


Figura 4.35.- Forma de onda de tensión para los devanados en operación bipolar. Secuencia de cuatro pasos.

Además un motor de pasos de operación bipolar puede girar en ambos sentidos, dependiendo de la secuencia de energización como se muestra en la figura 4.36.



PASO	BOBINA A	BOBINA B
1	+	+
2	+	-
3	-	-
4	-	+
1	+	+

Giro horario

Giro antihorario

Figura 4.36.- a) Operación bipolar; b) secuencias de energización de las bobinas para cuatre pasos.

Se ha desarrollado un ejemplo de aplicación de un control de un motor a pasos, el cual permite escoger diferentes tareas a diferentes velocidades y sentidos de giro. El objetivo es simplemente simular algunos de los comandos del Quick500, mediante la utilización de salidas digitales.

LISTADO DE LA RUTINA.- El listado completo del "software de aplicación" se presenta en el numeral 4.2.7, en el cual la subrutina que maneja al motor a pasos se llama "MOTOR".

RESULTADOS.- Los resultados que se obtienen en este caso también deben ser visualizados con el motor a pasos funcionando. El circuito que se utiliza para controlar al motor a pasos se presenta en la figura 4.37.

4.2.6.- TRANSFORMADA RAPIDA DE FOURIER.-

El objetivo de estudiar la transformada rápida de Fourier en este trabajo, es abrir el campo para el análisis de datos adquiridos mediante la estación KEITHLEY 500A en frecuencia, y como ejemplo de este objetivo se ha implementado el algoritmo que obtiene el espectro de densidad de potencia de una onda adquirida a través de los canales análogos de entrada.

Para probar la validez del algoritmo desarrollado se obtiene la transformada discreta de Fourier de la siguiente señal:

$$y = \text{sen}\ (4Hz.) + \text{sen}\ (12Hz.) + 2*rnd - 1$$

Es decir, contiene dos frecuencias predominantes, y además está contaminada de ruido. Esta señal es generada en un canal análogo de salida (numeral 4.1.2) mediante una rutina desarrollada para generar ondas de este tipo, siendo almacenada en disco para su futuro análisis.

Para la obtención de la transformada se trabaja con 32 datos, por lo que dichas ondas se obtienen con el mismo número de datos por ciclo con el objeto de tener mayor precisión al obtener la transformada discreta de Fourier y el espectro de densidad de potencia. Los resultados obtenidos se muestran en las tablas 4.1 y 4.2 y en la figura 4.38.

Como se aprecia en la figura 4.38, el espectro de densidad de potencia permite determinar las frecuencias predominantes, enmascaradas por el ruido que contamina la señal.

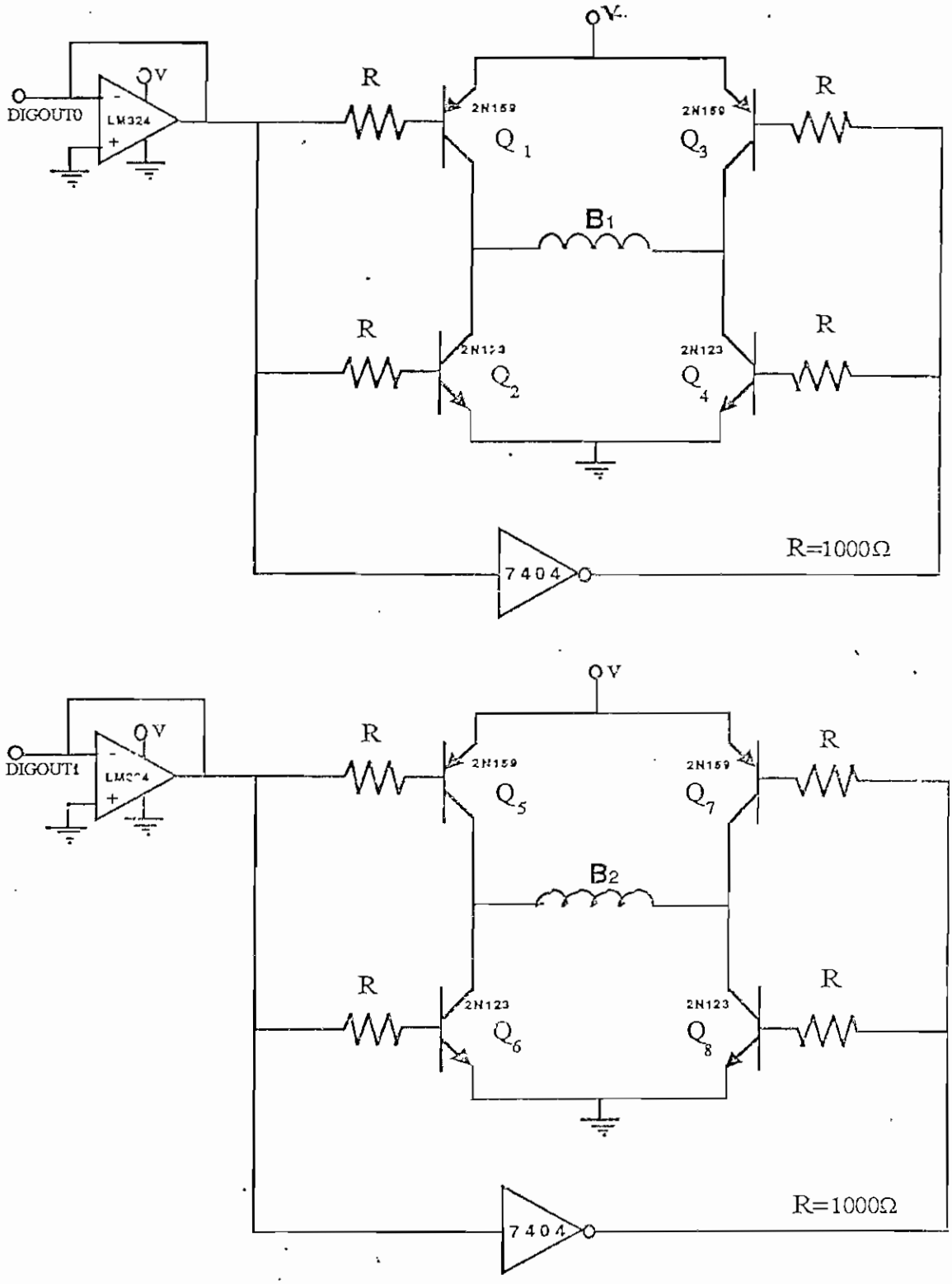


Figura 4.37.- Circuito controlador de un motor a pasos.

ONDA SOBRE LA QUE SE REALIZA EL PROCESAMIENTO:

SEN (4HZ.) + SEN(12HZ.) + 2*RDN - 1

Componente	Transformada de Fourier
1 de 32	0.0788 , 0.0000 i
2 de 32	-0.0485 , 0.0000 i
3 de 32	0.0847 , 0.0414 i
4 de 32	0.0847 , -0.0414 i
5 de 32	0.0726 , -0.4491 i
6 de 32	0.0297 , 0.3337 i
7 de 32	0.0297 , -0.3337 i
8 de 32	0.0726 , 0.4491 i
9 de 32	0.0871 , 0.0475 i
10 de 32	0.1276 , 0.0151 i
11 de 32	-0.1205 , 0.0586 i
12 de 32	-0.0179 , -0.0443 i
13 de 32	-0.0179 , 0.0443 i
14 de 32	-0.1205 , -0.0586 i
15 de 32	0.1276 , -0.0151 i
16 de 32	0.0871 , -0.0475 i
17 de 32	-0.0324 , -0.0288 i
18 de 32	-0.0239 , 0.0639 i
19 de 32	-0.1155 , -0.0926 i
20 de 32	-0.0117 , -0.0938 i
21 de 32	0.0936 , -0.0438 i
22 de 32	0.0174 , 0.0958 i
23 de 32	-0.0206 , -0.0555 i
24 de 32	0.0162 , -0.0235 i
25 de 32	0.0162 , 0.0235 i
26 de 32	-0.0206 , 0.0555 i
27 de 32	0.0174 , -0.0958 i
28 de 32	0.0936 , 0.0438 i
29 de 32	-0.0117 , 0.0938 i
30 de 32	-0.1155 , 0.0926 i
31 de 32	-0.0239 , -0.0639 i
32 de 32	-0.0324 , 0.0288 i

Tabla 4.1.- Resultados de la Transformada rápida de Fourier.

ESPECTRO DE DENSIDAD DE POTENCIA DE LA ONDA:

Y = SEN (4HZ.) + SEN(12HZ.) + 2*RDN - 1

frecuencia(Hz.)	Dens. Potencia
0.00	Pyy= 0.079
1.00	Pyy= 0.043
2.00	Pyy= 0.099
3.00	Pyy= 0.029
4.00	Pyy= 0.455
5.00	Pyy= 0.103
6.00	Pyy= 0.048
7.00	Pyy= 0.095
8.00	Pyy= 0.094
9.00	Pyy= 0.148
10.00	Pyy= 0.134
11.00	Pyy= 0.097
12.00	Pyy= 0.335
13.00	Pyy= 0.059
14.00	Pyy= 0.128
15.00	Pyy= 0.068

Tabla 4.2.- Resultados del espectro de densidad de potencia.

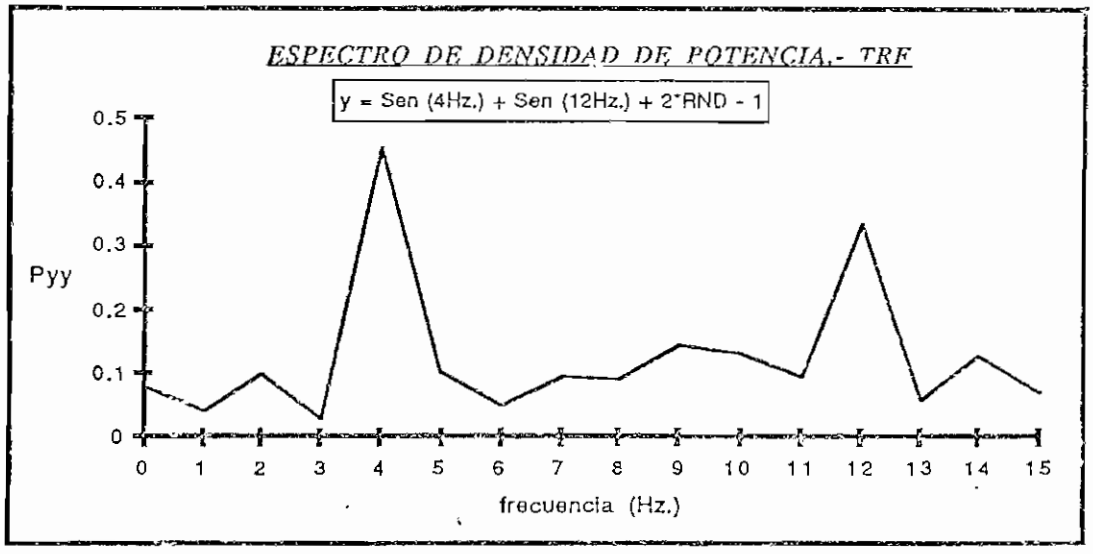


Figura 4.38.- Espectro de densidad de potencia

4.2.7.- LISTADO DEL SOFTWARE DE APLICACION (CAP40.BAS)

```
DECLARE SUB PID ()
DECLARE SUB REALEST ()
DECLARE SUB ONOFDIS ()
DECLARE SUB TRESNIV ()
DECLARE SUB ident ()
DECLARE SUB MOTOR ()
DECLARE SUB algoritmo ()
DECLARE SUB defgraf ()
DECLARE SUB graf (I%)
DECLARE SUB giro ()
DECLARE SUB indefinido ()
DECLARE SUB vent1 ()
DECLARE SUB vent2 ()
DECLARE SUB vent3 ()
DECLARE SUB vent4 ()
DECLARE SUB vent5 ()
DECLARE SUB vent6 ()
DECLARE SUB vent7 ()
DECLARE SUB vent8 ()
DECLARE SUB vent9 ()
DECLARE SUB vent11 ()
DECLARE SUB vent10 ()
DECLARE SUB vent12 ()
DECLARE SUB grafico7 (j%)
DECLARE SUB grafic1 (j%)
DECLARE SUB grafic2 (j%)
DECLARE SUB grafico1 (j%)
DECLARE SUB grafico2 (j%)
DECLARE SUB grafico3 (j%)
DECLARE SUB grafico4 (j%)
DECLARE SUB grafico5 (j%)
DECLARE SUB grafico6 (j%)
DECLARE SUB grafico8 (j%)
DECLARE SUB grafico9 (j%)
DECLARE SUB grafico10 (j%)
DECLARE SUB pantalla ()
DECLARE SUB graf1 ()
DECLARE SUB graf2 ()
DECLARE SUB defgrafic ()
DECLARE SUB algomul ()

DEFINT I-N
DEFDBL A-H, O-Z

COMMON SHARED bar1!(), bar2!(), m, n, I, ii, ko(), k(), L, j
COMMON SHARED bintv%, cy%
COMMON SHARED valor!, sal!, c(), v(), r, vmax!, vmin!, vmed!
menu:
CLS
SCREEN 9
LINE (0, 0)-(600, 335), 10, B
LINE (3, 3)-(597, 332), 10, B
COLOR 2
```

```
LOCATE 3, 10: PRINT "  SOFTWARE DE APLICACION DEL SISTEMA DE ADUISICION DE DATOS"
LOCATE 4, 20: PRINT "       Y CONTROL KEITHLEY 500A"
LOCATE 9, 30: PRINT "MENU PRINCIPAL:"
LOCATE 10, 28: PRINT "=========================="
COLOR 15
LOCATE 13, 5: PRINT "P.I.D. DISCRETO                                    (1)"
LOCATE 14, 5: PRINT "REALIMENTACION DE ESTADO                           (2)"
LOCATE 15, 5: PRINT "CONTROL ON-OFF                                     (3)"
LOCATE 16, 5: PRINT "CONTROL DE TRES NIVELES                            (4)"
LOCATE 17, 5: PRINT "IDENTIFICACION DE SISTEMAS                         (5)"
LOCATE 18, 5: PRINT "CONTROL DE UN MOTOR A PASOS                        (6)"
LOCATE 19, 5: PRINT "TERMINAR LA SESION DE TRABAJO                      (7)"

pregunta:
LOCATE 22, 39: PRINT "       "
LOCATE 22, 5: INPUT "Escoja una opción por favor (1-7):"; op$
IF op$ = "1" THEN CALL PID: CLEAR : GOTO menu
IF op$ = "2" THEN CALL REALEST: CLEAR : GOTO menu
IF op$ = "3" THEN CALL ONOFDIS: CLEAR : GOTO menu
IF op$ = "4" THEN CALL TRESNIV: CLEAR : GOTO menu
IF op$ = "5" THEN CALL ident: CLEAR : GOTO menu
IF op$ = "6" THEN CALL MOTOR: CLEAR : GOTO menu
IF op$ = "7" THEN END
IF op$ > "7" THEN GOTO pregunta

SUB algomul
'-----------------DESARROLLO DEL ALGORITMO-----------------
    FOR L = 0 TO m - 1
        FOR j = 0 TO n - 1
            bar1!(L) = bar1!(L) - k!(L, j) * bar2!(j)
        NEXT
        bar1!(L) = bar1!(L) + ko!(L)
        IF bar1!(L) > 10 THEN
        bar1!(L) = 10
        ELSEIF bar1!(L) < 0 THEN
        bar1!(L) = 0
        END IF
    NEXT
'-----------------REALIMENTACION DE ESTADO-----------------
END SUB

DEFSNG A-Z
SUB algoritmo
'-----------------DESARROLLO DEL ALGORITMO-----------------
        y = valor!
        E = r - y
        v(1) = sal!
        FOR L = 4 TO 3 STEP -1
            v(L) = v(L - 1)
        NEXT

        v(2) = E
        u = 0

        FOR L = 1 TO 4
```

```
            u = u + c(L) * v(L)
        NEXT

        sal! = u
        IF sal! > 10 THEN
            sal! = 10
            LOCATE 41, 55: PRINT "Canal/salida sat."
        ELSEIF sal! < 0 THEN
            sal! = 0
            LOCATE 41, 55: PRINT "Canal/salida sat."
        ELSE
            LOCATE 41, 55: PRINT "              "
        END IF
'----------------------PID DISCRETO-----------------------------

END SUB

DEFINT I-N
DEFDBL A-H, O-Z
SUB defgraf             'Rutina que define la pantalla gráfica para dos señales
LINE (0, 12)-(639, 140), 1, BF
LINE (0, 12)-(639, 140), 15, B
'--------
LINE (40, 25)-(629, 25), 5, , &H8888
'-------
LINE (30, 125)-(629, 125), 7
LOCATE 17, 40
PRINT "TIEMPO"
LOCATE 5, 2: PRINT "y"
LINE (40, 20)-(40, 130), 7

LINE (0, 185)-(639, 315), 1, BF
LINE (0, 185)-(639, 315), 15, B
'---------
LINE (40, 200)-(629, 200), 5, , &H8888
'--------
LINE (30, 300)-(629, 300), 7
LOCATE 39, 40
PRINT "TIEMPO"
LOCATE 27, 2: PRINT "u"
LINE (40, 195)-(40, 305), 7

END SUB

SUB defgrafic           'Rutina que selecciona cuales y que número de ventanas son necesarias, caso
multivariable
SELECT CASE I
  CASE 0
        vent1: vent2
LOCATE 1, 5: PRINT "ESTUDIO DEL SISTEMA DE ADQUISICION DE DATOS KEYTLHEY 500A Y APLICACIONES"
COLOR 2
LOCATE 20, 10: PRINT "Tiempo Real"
LOCATE 22, 10: PRINT "Control en Proceso"
LOCATE 41, 10: PRINT "SPACE BAR para terminar proceso"
COLOR 7
```

```
    CASE 1
        vent3: vent4: vent5
LOCATE 1, 5: PRINT "ESTUDIO DEL SISTEMA DE ADQUISICION DE DATOS KEYTLHEY 500A Y APLICACIONES"
COLOR 2
LOCATE 25, 45: PRINT "Tiempo Real"
LOCATE 28, 45: PRINT "Control en Proceso"
LOCATE 33, 45: PRINT "SPACE BAR para terminar proceso"
COLOR 7
    CASE 2
        vent3: vent4: vent6
LOCATE 1, 5: PRINT "ESTUDIO DEL SISTEMA DE ADQUISICION DE DATOS KEYTLHEY 500A Y APLICACIONES"
COLOR 2
LOCATE 25, 5: PRINT "Tiempo Real"
LOCATE 28, 5: PRINT "Control en Proceso"
LOCATE 33, 5: PRINT "SPACE BAR para terminar proceso"
COLOR 7
    CASE 3
        vent3: vent4: vent5: vent6
LOCATE 1, 5: PRINT "ESTUDIO DEL SISTEMA DE ADQUISICION DE DATOS KEYTLHEY 500A Y APLICACIONES"
COLOR 2
LOCATE 20, 37: PRINT "Tiempo Real"
LOCATE 23, 33: PRINT "Control en Proceso"
LOCATE 41, 27: PRINT "SPACE BAR para terminar proceso"
COLOR 7
    CASE 4
        vent7: vent8:  vent9: vent11
LOCATE 1, 5: PRINT "ESTUDIO DEL SISTEMA DE ADQUISICION DE DATOS KEYTLHEY 500A Y APLICACIONES"
COLOR 2
LOCATE 25, 45: PRINT "Tiempo Real"
LOCATE 28, 45: PRINT "Control en Proceso"
LOCATE 33, 45: PRINT "SPACE BAR para terminar proceso"
COLOR 7
    CASE 5
        vent7: vent8: vent10: vent12
LOCATE 1, 5: PRINT "ESTUDIO DEL SISTEMA DE ADQUISICION DE DATOS KEYTLHEY 500A Y APLICACIONES"
COLOR 2
LOCATE 25, 5: PRINT "Tiempo Real"
LOCATE 28, 5: PRINT "Control en Proceso"
LOCATE 33, 5: PRINT "SPACE BAR para terminar proceso"
COLOR 7
    CASE 6
        vent7: vent8: vent9: vent10: vent11
LOCATE 1, 5: PRINT "ESTUDIO DEL SISTEMA DE ADQUISICION DE DATOS KEYTLHEY 500A Y APLICACIONES"
COLOR 2
LOCATE 31, 45: PRINT "Tiempo Real"
LOCATE 33, 45: PRINT "Control en Proceso"
LOCATE 35, 45: PRINT "SPACE BAR para terminar proceso"
COLOR 7
    CASE 7
        vent7: vent8:  vent9: vent10: vent12
LOCATE 1, 5: PRINT "ESTUDIO DEL SISTEMA DE ADQUISICION DE DATOS KEYTLHEY 500A Y APLICACIONES"
COLOR 2
LOCATE 31, 5: PRINT "Tiempo Real"
LOCATE 33, 5: PRINT "Control en Proceso"
LOCATE 35, 5: PRINT "SPACE BAR para terminar proceso"
```

```
COLOR 7
   CASE 8
        vent7: vent8: vent9: vent10: vent11: vent12
LOCATE 1, 5: PRINT "ESTUDIO DEL SISTEMA DE ADQUISICION DE DATOS KEYTLHEY 500A Y APLICACIONES"
COLOR 2
LOCATE 15, 5: PRINT "Tiempo Real, Control en Proceso"
LOCATE 22, 10: PRINT "Control en Proceso"
LOCATE 15, 45: PRINT "SPACE BAR para terminar proceso"
COLOR 7
END SELECT

END SUB

SUB giro                    'Rutina que selecciona la tarea que va a realizar el motor a paso
CLS
DIM v%(4), u%(4)
v%(1) = 1: v%(2) = 0: v%(3) = 0: v%(4) = 1
u%(1) = 1: u%(2) = 1: u%(3) = 0: u%(4) = 0

CALL ARMAKE("area&", 4!, 1, "DOUT0")
CALL ARMAKE("array&", 4!, 1, "DOUT2")
sentido:
LOCATE 20, 3: INPUT "Escoja sentido de giro Horario (H) o Antihorario (A)"; s$
IF s$ = "h" OR s$ = "H" THEN
    FOR DEP! = 1 TO 4
        CALL ARPUTVALI("area&", DEP!, 1, "", v%(DEP!))
        CALL ARPUTVALI("array&", DEP!, 1, "", u%(DEP!))
    NEXT
ELSEIF s$ = "A" OR s$ = "a" THEN
    FOR DEP! = 1 TO 4
        CALL ARPUTVALI("area&", DEP!, 1, "", u%(DEP!))
        CALL ARPUTVALI("array&", DEP!, 1, "", v%(DEP!))
    NEXT
ELSE
    GOTO sentido
END IF

CALL DIGOUT("area&", "DOUT0", bintv%, cy%, "nt", "tarea")
CALL DIGOUT("array&", "DOUT2", bintv%, cy%, "nt", "")

CLS
SCREEN 9
L = 0
PRINT "motor activado"
CALL inton(1, "tunic")
st% = 1
WHILE st% <> 0
    CALL STATUS("tarea", st%)
WEND

CALL INTOFF

PRINT "Motor Desactivado"
CALL ARDEL("area&")
CALL ARDEL("array&")
```

```
CALL backclear

END SUB

SUB graf (I%)                    'Rutina llama a la rutina de gráfico correspondiente

IF valor! > 9.99 THEN
    PSET (I% + 40, 25), 4
ELSEIF valor! < .04 THEN
    PSET (I% + 40, 125), 4
ELSE
    y = -10 * valor! + 125
    PSET (I% + 40, y), 14
END IF
IF sal! > 9.99 THEN
    PSET (I% + 40, 200), 4
ELSEIF sal! < .04 THEN
    PSET (I% + 40, 300), 4
ELSE
    u = -10 * sal! + 300
    PSET (I% + 40, u), 14
END IF
END SUB

SUB graf1                        'Rutina que grafica un dato adquirido o de salida según el caso

    IF k% > 580 THEN
        defgrafic
        k% = 0
    END IF
    grafic1 (k%)
    k% = k% + 1
    LOCATE 21, 1: PRINT k%

END SUB

SUB graf2                        'Rutina llama a la rutina de gráfico correspondiente

    IF k% > 280 THEN
        defgrafic
        k% = 0
    END IF
    grafic2 (k%)
    k% = k% + 1

END SUB

SUB grafic1 (j)                  'Rutina que grafica un dato adquirido

IF bar1!(0) > 9.99 THEN
    PSET (j + 40, 25), 4
ELSEIF bar1!(0) < .05 THEN
    PSET (j + 40, 125), 4
ELSE
    y = -10 * bar1!(0) + 125
```

```
    PSET (j + 40, y), 14
END IF

IF bar2!(0) > 9.99 THEN
    PSET (j + 40, 200), 4
ELSEIF bar2!(0) < .05 THEN
    PSET (j + 40, 300), 4
ELSE
    u = -10 * bar2!(0) + 300
    PSET (j + 40, u), 14
END IF
LOCATE 21, 40: PRINT USING "y= salida de la planta ##.### voltios"; bar1!(0)
LOCATE 22, 40: PRINT USING "u= ley de control     ##.### voltios"; bar2!(0)

END SUB

SUB grafic2 (k%)                    'Rutina determina las rutinas gráficas que se debe utilizar

SELECT CASE I
    CASE 1
        grafico1 (k%): grafico2 (k%): grafico3 (k%)
    CASE 2
        grafico1 (k%): grafico3 (k%): grafico4 (k%)
    CASE 3
        grafico1 (k%): grafico2 (k%): grafico3 (k%): grafico4 (k%)
    CASE 4
        grafico5 (k%): grafico6 (k%): grafico7 (k%): grafico8 (k%)
    CASE 5
        grafico5 (k%): grafico8 (k%): grafico9 (k%): grafico10 (k%)
    CASE 6
        grafico5 (k%): grafico6 (k%):  grafico7 (k%) :grafico8 (k%): grafico9 (k%)
    CASE 7
        grafico5 (k%): grafico6 (k%): grafico8 (k%): grafico9 (k%): grafico10 (k%)
    CASE 8
        grafico5 (k%): grafico6 (k%):  grafico7 (k%): grafico8 (k%): grafico9 (k%): grafico10 (k%)
END SELECT
END SUB

SUB grafico1 (j)                    'Rutina que grafica un dato adquirido

IF bar1!(0) > 9.99 THEN
    PSET (j + 20, 25), 4
ELSEIF bar1!(0) < .05 THEN
    PSET (j + 20, 125), 4
ELSE
    y = -10 * bar1!(0) + 125
    PSET (j + 20, y), 14
END IF
LOCATE 21, 1: PRINT USING "y= ley de control ##.### voltios"; bar1!(0)

END SUB

SUB grafico10 (j)                   'Rutina que grafica un dato de salida

IF bar2!(2) > 9.99 THEN
```

```
        PSET (j + 340, 240), 4
    ELSEIF bar2!(2) < .05 THEN
        PSET (j + 340, 320), 4
    ELSE
        y = -8 * bar2!(2) + 320
        PSET (j + 340, y), 14
    END IF
    LOCATE 29, 56: PRINT USING "##.### v."; bar2!(2)

    END SUB

    SUB grafico2 (j)                    'Rutina que grafica un dato adquirido

    IF bar1!(1) > 9.99 THEN
        PSET (j + 20, 200), 4
    ELSEIF bar1!(1) < .05 THEN
        PSET (j + 20, 300), 4
    ELSE
        y = -10 * bar1!(1) + 300
        PSET (j + 20, y), 14
    END IF
    LOCATE 22, 1: PRINT USING "y= ley de control ##.### voltios"; bar1!(1)

    END SUB

    SUB grafico3 (j)                    'Rutina que grafica un dato de salida

    IF bar2!(0) > 9.99 THEN
        PSET (j + 340, 25), 4
    ELSEIF bar2!(0) < .05 THEN
        PSET (j + 340, 125), 4
    ELSE
        y = -10 * bar2!(0) + 125
        PSET (j + 340, y), 14
    END IF
    LOCATE 21, 45: PRINT USING "u= salida de la planta ##.### voltios"; bar2!(0)

    END SUB

    SUB grafico4 (j)                    'Rutina que grafica un dato de salida

    IF bar2!(1) > 9.99 THEN
        PSET (j + 340, 200), 4
    ELSEIF bar2!(1) < .05 THEN
        PSET (j + 340, 300), 4
    ELSE
        y = -10 * bar2!(1) + 300
        PSET (j + 340, y), 14
    END IF
    LOCATE 22, 45: PRINT USING "u= salida de la planta ##.### voltios"; bar2!(1)

    END SUB

    SUB grafico5 (j)                    'Rutina que grafica un dato adquirido
```

```
IF bar1!(0) > 9.99 THEN
    PSET (j + 20, 20), 4
ELSEIF bar1!(0) < .05 THEN
    PSET (j + 20, 100), 4
ELSE
    y = -8 * bar1!(0) + 100
    PSET (j + 20, y), 14
END IF
LOCATE 2, 16: PRINT USING "##.### v."; bar1!(0)

END SUB

SUB grafico6 (j)                    'Rutina que grafica un dato adquirido

IF bar1!(1) > 9.99 THEN
    PSET (j + 20, 130), 4
ELSEIF bar1!(0) < .05 THEN
    PSET (j + 20, 210), 4
ELSE
    y = -8 * bar1!(1) + 210
    PSET (j + 20, y), 14
END IF
LOCATE 16, 16: PRINT USING "##.### v."; bar1!(1)

END SUB

SUB grafico7 (j)                    'Rutina que grafica un dato adquirido

IF bar1!(2) > 9.99 THEN
    PSET (j + 20, 240), 4
ELSEIF bar1!(2) < .05 THEN
    PSET (j + 20, 320), 4
ELSE
    y = -8 * bar1!(2) + 320
    PSET (j + 20, y), 14
END IF
LOCATE 29, 16: PRINT USING "##.### v."; bar1!(2)

END SUB

SUB grafico8 (j)                    'Rutina que grafica un dato de salida

IF bar2!(0) > 9.99 THEN
    PSET (j + 340, 20), 4
ELSEIF bar2!(0) < .05 THEN
    PSET (j + 340, 100), 4
ELSE
    y = -8 * bar2!(0) + 100
    PSET (j + 340, y), 14
END IF
LOCATE 2, 56: PRINT USING "##.### v."; bar2!(0)

END SUB

SUB grafico9 (j)                    'Rutina que grafica un dato de salida
```

```
IF bar2!(1) > 9.99 THEN
    PSET (j + 340, 130), 4
ELSEIF bar2!(1) < .05 THEN
    PSET (j + 340, 210), 4
ELSE
    y = -8 * bar2!(1) + 210
    PSET (j + 340, y), 14
END IF
LOCATE 16, 56: PRINT USING "##.### v."; bar2!(1)

END SUB

SUB Ident                'Control adaptivo en tiempo real
CLS
SCREEN 9
LINE (0, 0)-(600, 335), 14, B
LINE (3, 3)-(597, 332), 10, B
COLOR 14
LOCATE 4, 21: PRINT "INGRESE EL GRADO DE A(q) "; : INPUT n
LOCATE 6, 21
PRINT "INGRESE EL GRADO DE B(q) "; : INPUT m
DIM a(n), B(m), c1(n), y(8), ym(8), u(8), d(10)
'
OVER$ = " "
'
'Ingreso del polinomio de control Cr(q)
'Los coeficientes de Cr(q) están en C(1)
'
DIM MED(20)
DIM E(10) AS DOUBLE
DIM F(10, 10) AS DOUBLE, G(10, 10) AS DOUBLE, CM(10, 10) AS DOUBLE
'************************************************************ ***************
'INICIALIZACION DEL SISTEMA DE ADQUISICION DE DATOS
'*************************************************************************

CALL SOFTINIT
CALL INIT
'*************************************************************************
LOCATE 8, 21
COLOR 2
PRINT "INGRESO DEL POLINOMIO DE CONTROL Cr(q) "
LOCATE 10, 21
PRINT "EL GRADO DEL POLINOMIO Cr(q) ES (máx."; n; ")...:";
INPUT GC
IF GC > n THEN
BEEP
GC = n
ELSE
END IF
'
SELECT CASE GC
'
CASE 0
LOCATE 15, 35
```

```
PRINT "C(q)=1"
c1(1) = 0: c1(2) = 0: c1(3) = 0
'
CASE 1
LOCATE 15, 40
PRINT "C1="; : INPUT c1(1)
c1(2) = 0: c1(3) = 0
'
CASE 2
LOCATE 15, 40
PRINT "C1="; : INPUT c1(1)
LOCATE 16, 40
PRINT "C2="; : INPUT c1(2)
c1(3) = 0
'
CASE 3
LOCATE 15, 40
PRINT "C1="; : INPUT c1(1)
LOCATE 16, 40
PRINT "C2="; : INPUT c1(2)
LOCATE 17, 40
PRINT "C3="; : INPUT c1(3)
'
CASE ELSE
BEEP
'-------------------------------------------------------------
ident
'-------------------------------------------------------------

END SELECT
'*************************************************************************
'-------------------------------------------------------------
'PERIODOS DE MUESTREO DEL SISTEMA DE ADQUISICION DE DATOS
'-------------------------------------------------------------
CLS
COLOR 3
LOCATE 2, 10: INPUT "Tiempo de duración del Algoritmo"; TMIN%
REPITE:
LOCATE 5, 10: INPUT "Período de muestreo em [ms] "; bintv%
PER% = INT(bintv% / TMIN%)
IF PER% = 0 THEN
LOCATE 14, 10: PRINT "ERROR PERIODO DE MUESTREO MENOR QUE EL TIEMPO DE "
LOCATE 15, 10: PRINT "DURACION DEL ALGORITMO."
GOTO REPITE
END IF
boutv% = bintv% / PER%  ' para que BINTV% sea múltiplo de BOUTV%
bintv% = PER% * boutv% 'Tiempo de muestreo sea múltiplo de TMIN

CLS
LOCATE 1, 15: PRINT "PERIODO DE MUESTREO AJUSTADO [ms]"; bintv%
LOCATE 3, 15: PRINT "RETARDO DEL DATO DE SALIDA  [ms]"; boutv%
bintv% = bintv%
boutv% = boutv%
'*************************************************************************
'Cálculos del subprograma
```

```
'
LOCATE 20, 3
PRINT STRING$(74, "-")
LOCATE 21, 5
PRINT "INGRESE EL VALOR DE LA DIAGONAL DE F[I,J] :";
INPUT DIAG
'
G = n + m + 1
'Protección Iterativa
FOR I = 1 TO G
FOR j = 1 TO G
CM(I, j) = 0
NEXT j
NEXT I
'
FOR I = 1 TO G
CM(I, I) = DIAG
NEXT I
'
TRK = 0
FOR I = 1 TO G
TRK = TRK + CM(I, I)
NEXT I
'
E(1) = 1: FOR I = 2 TO G: E(I) = 0: NEXT I
'
FLAG$ = " "
CLS
'-------------------------------------------------------------
'SETEO DE TAREAS DE BACKGROUND
'-------------------- ----------------------------------------
DEP! = 1
CALL ANIN("ENTRADA%", DEP!, "ANLG0", bintv%, -1, "NT", "TAREA1")
CALL ANIN("REFER%", DEP!, "ANLG1", bintv%, -1, "NT", "TAREAR")
CALL ANIN("identif%", 200!, "ANLG0 ANLG1 ANLG2", bintv%, 1, "NT", "")

CALL ARMAKE("SALIDA%", DEP!, -1, "ANOUT0")
CALL ANOUT("SALIDA%", "ANOUT0", boutv%, -1, "NT", "TAREA2")

FOR I! = 1 TO DEP!
   CALL ARPUTVALF("SALIDA%", I!, -1, "ANOUT0", 0!, 0)
NEXT I!
'-------------------------------------------------------------
'Este lazo encera la salida
'*************************************************************************
LOCATE 18, 3: PRINT "PRESIONE CUALQUIER TECLA PARA INICIAR CONTROL"
COLOR 4
DO
      an$ = INKEY$
LOOP WHILE an$ = ""
SCREEN 9
WIDTH 80, 43
CLS
defgraf
COLOR 9
```

```
LOCATE 1, 5: PRINT "ESTUDIO DEL SISTEMA DE ADQUISICION DE DATOS KEYTLHEY 500A Y APLICACIONES"
COLOR 2
LOCATE 20, 10: PRINT "Tiempo Real"
LOCATE 22, 10: PRINT "Control en Proceso"
LOCATE 41, 10: PRINT "SPACE BAR para terminar proceso"
COLOR 7
DT1 = 1    'Dato para igualar el timer (seguidor del timer)
DIM DT(0 TO 7) AS DOUBLE, TIM(0 TO 7) AS INTEGER
TIM(0) = 1 'Timer 0 reseteado y habilitado
CALL inton(1, "MIL")
CALL TIMERSTART(TIM(), "NT", "TIMER0")  'Reseteo y habilitación de timer
valor! = 0
an$ = ""
   WHILE an$ = ""
      CALL ARLASTP("ENTRADA%", lp!)
      CALL ARLASTP("SALIDA%", lp!)
      CALL ARLASTP("REFER%", lp!)
         CALL ARGETVALF("ENTRADA%", 1!, -1, "ANLG0", valor!, 0)
         CALL ARGETVALF("REFER%", 1!, -1, "ANLG1", REF!, 0)
         LOCATE 20, 50: PRINT "REFERENCIA";
          LOCATE 21, 50: PRINT USING "##.######  [voltios]"; REF!
'*************************************************************************
'Lazo Principal
'
   ym(1) = REF!
   y(1) = valor!
   'Cálculo de la señal de control
   FOR k = 5 TO 2 STEP -1
   u(k) = u(k - 1)        'Los valores actuales pasan a ser
   y(k) = y(k - 1)        'valores anteriores
   ym(k) = ym(k - 1)
   NEXT k
   X1 = 0
   L1 = 2
   FOR k = 2 TO (m + 1)
   X1 = X1 + E(k) * u(L1)
   L1 = L1 + 1
   NEXT k
   X2 = 0
   L2 = 2
   FOR k = (m + 2) TO (m + n + 2)
   X2 = X2 + E(k) * y(L2)
   L2 = L2 + 1
   NEXT k
    X3 = ym(2) + c1(1) * ym(3) + c1(2) * ym(4) + c1(3) * ym(5)
   u(1) = (1 / E(1)) * (X3 - X2 - X1)
    'Final del cálculo de la señal de control
   '
   SX1 = 1
   LOCATE 20, 1
   PRINT "SEÑAL DE CONTROL"; SPC(SX1); "=";
    PRINT USING "####.######"; u(1);
   LOCATE 21, 1
   PRINT "SALIDA DEL PROCESO"; SPC(SX1); "=";
    PRINT USING "####.######"; y(1)
```

```
'
'Asignación del vector MED(), se almacenan los valores medidos
FOR k = 1 TO m + 1
MED(k) = u(k + 1)
NEXT k
L = 1
FOR k = (m + 2) TO (m + n + 2)
MED(k) = y(L + 2)
L = L + 1
NEXT k
'
'————INICIO DE IDENTIFICACION————————————
S4 = 0
S5 = 0
FOR I = 1 TO G
FOR j = 1 TO G
S4 = S4 + MED(j) * CM(j, I)
NEXT j
S5 = S5 + MED(I) * S4
S4 = 0
NEXT I
S3 = 1 + S5
S6 = 0
FOR I = 1 TO G
FOR j = 1 TO G
S6 = S6 + CM(I, j) * MED(j)
NEXT j
d(I) = S6
S6 = 0
NEXT I
FOR I = 1 TO C
FOR j = 1 TO G
F(I, j) = d(I) * MED(j)
NEXT j
NEXT I
FOR I = 1 TO G
FOR j = 1 TO G
G(I, j) = 0
NEXT j
NEXT I
FOR I = 1 TO G
FOR j = 1 TO G
FOR k = 1 TO G
G(I, j) = G(I, j) + F(I, k) * CM(k, j)
NEXT k
NEXT j
NEXT I
FOR I = 1 TO G
FOR j = 1 TO G
CM(I, j) = CM(I, j) - G(I, j) / S3
NEXT j
NEXT I
'
SUMATRAZA = 0
FOR I = 1 TO G
```

```
SUMATRAZA = SUMATRAZA + CM(I, I)
NEXT I
'Factor de olvido FO
FO = SUMATRAZA / TRK
FOR I = 1 TO G
FOR j = 1 TO G
CM(I, j) = CM(I, j) / FO 'Actual CM(I,J)
NEXT j
NEXT I
'
S7 = 0
FOR I = 1 TO G
S7 = S7 + E(I) * MED(I)
NEXT I
S1 = y(2) + c1(1) * y(3) + c1(2) * y(4) + c1(3) * y(5) - S7
S2 = 0
FOR I = 1 TO G
FOR j = 1 TO G
S2 = S2 + CM(I, j) * MED(j)
NEXT j
'
E(I) = E(I) + FO * S1 * S2

IF E(1) < .01 THEN E(1) = 1
'
S2 = 0
NEXT I
'FIN DE IDENTIFICACION
'*************************************************************************
sal! = u(1)
IF sal! > 10 THEN
   sal! = 10
   LOCATE 14, 25
ELSEIF sal! < 0 THEN
   LOCATE 14, 25
END IF

'*************************************************************************
'-------------------------------------------------------
' SETEO DE SALIDA DE DATOS
'-------------------------------------------------------
      FOR DE! = 1 TO DEP!
      CALL ARPUTVALF("SALIDA%", DEP!, -1, "ANOUT0", sal!, 0)
      NEXT DE!

      ' LOCATE 11, 25: PRINT "LEY DE CONTROL";
      ' LOCATE 11, 40: PRINT USING "##.###### [voltios]"; SAL!
       IF ka% > 580 THEN          'reseteo de pantalla gráfica
       defgraf
       ka% = 0
       END IF
       graf (ka%)
       ka% = ka% + 1
       LOCATE 41, 70: PRINT ka%
anS = INKEYS
```

```
    DO
      CALL TIMERREAD(DT())
      LOOP UNTIL (DT(0) / bintv%) > DT1
      DT1 = INT(DT(0) / bintv% + 1)
    'Congela el Foreground hasta adquirir otro dato
    LOCATE 12, 25

WEND
CALL INTOFF
CALL ARDEL("ENTRADA%")
CALL ARDEL("SALIDA%")
CALL arwrite("identif%", "mras.prn", 0, 4, bintv%, 1)
WIDTH 80, 25
END SUB

SUB  indefinido                    'Rutina que hace girar al motor indefinidamente hasta contraorden externa

CLS
DIM v%(4), u%(4)
v%(1) = 1: v%(2) = 0: v%(3) = 0: v%(4) = 1
u%(1) = 1: u%(2) = 1: u%(3) = 0: u%(4) = 0
cy% = -1
CALL ARMAKE("area&", 4!, 1, "DOUT0")
CALL ARMAKE("array&", 4!, 1, "DOUT2")
sentidog:
LOCATE 20, 3: INPUT "Escoja sentido de giro Horario (H) o Antihorario (A)"; sS
IF sS = "H" OR sS = "h" THEN
    FOR DEP! = 1 TO 4
        CALL ARPUTVALI("area&", DEP!, 1, "", v%(DEP!))
        CALL ARPUTVALI("array&", DEP!, 1, "", u%(DEP!))
    NEXT
ELSEIF sS = "A" OR sS = "a" THEN
    FOR DEP! = 1 TO 4
        CALL ARPUTVALI("area&", DEP!, 1, "", u%(DEP!))
        CALL ARPUTVALI("array&", DEP!, 1, "", v%(DEP!))
    NEXT
ELSE
    GOTO sentidog
END IF
CALL DIGOUT("area&", "DOUT0", bintv%, cy%, "nt", "")
CALL DIGOUT("array&", "DOUT2", bintv%, cy%, "nt", "")
CLS
LOCATE 23, 10: PRINT "Presione SPACE BAR para terminar giro del motor"
SCREEN 9

CALL inton(1, "hmic")

    PRINT "Motor Activado"
    anS = ""
    WHILE anS = ""
        anS = INKEY$
    WEND

CALL INTOFF
```

```
 PRINT "Motor Desactivado"
CALL ARDEL("area&")
CALL ARDEL("array&")
CALL backclear

END SUB

SUB MOTOR                    'rutina que ofrece menú de opciones del motor a pasos

CALL SOFTINIT
CALL INIT
pregmot:
CLS
SCREEN 9
LINE (0, 0)-(600, 335), 10, B
LINE (3, 3)-(597, 332), 10, B
COLOR 10

LOCATE 3, 65: PRINT "     "
LOCATE 3, 7: INPUT "Escoja velocidad alta (a), media (m) ,baja (b) o salir(s)"; ppS

IF ppS = "a" THEN
p = 720
ELSEIF ppS = "m" THEN
p = 150
ELSEIF ppS = "b" THEN
p = 47
ELSEIF ppS = "s" THEN
GOTO elfin
ELSE
GOTO pregmot
END IF
bintv% = INT(10000 / p)

LOCATE 5, 7: PRINT "Control de un Motor a pasos:"
LOCATE 8, 21: PRINT "Cambio de posición (en grados)          (1)"
LOCATE 9, 21: PRINT "Una vuelta completa                     (2)"
LOCATE 10, 21: PRINT "Número definido de vueltas             (3)"
LOCATE 11, 21: PRINT "Giro indefinido                        (4)"
LOCATE 12, 21: PRINT "Salir del programa                     (5)"
escoja:
LOCATE 15, 40: PRINT "     "
LOCATE 15, 3: INPUT "Escoja una de las opciones enunciadas"; o$
IF o$ = "1" THEN
    LOCATE 17, 23: PRINT "     Grados"
    LOCATE 17, 3: INPUT "Variar la posición"; G
    cy% = INT(G / 7.2)
    giro
    CLS
    GOTO pregmot
ELSEIF o$ = "2" THEN
    cy% = 50
    giro
    CLS
    GOTO pregmot
```

```
ELSEIF o$ = "3" THEN
    LOCATE 17, 3: INPUT "Número deseado de giros"; n
    num% = INT(n)
    cy% = 50 * num%
    giro
    CLS
    GOTO pregmot
ELSEIF o$ = "4" THEN
    indefinido
    CLS
    GOTO pregmot
ELSEIF o$ = "5" THEN
    CLS
    GOTO elfin
ELSE
    GOTO escoja
END IF
elfin:
END SUB

SUB ONOFDIS                    'Rutina que realiza control ON-OFF a base de disparos

preg:
CLS
LINE (0, 0)-(600, 335), 10, B
LINE (3, 3)-(597, 332), 10, B
COLOR 14
LOCATE 5, 5: INPUT "valor en voltios del valor máximo de salida"; vmax!
LOCATE 7, 5: INPUT "valor en voltios del valor mínimo de salida"; vmin!
IF vmin! > vmax! THEN
LOCATE 10, 5: PRINT "El valor mínimo debe ser menor que el valor máximo"
GOTO preg
END IF
COLOR 7
SCREEN 9
WIDTH 80, 43
CLS

LOCATE 1, 1: PRINT "Control ON-OFF: entre"; vmax!; " y "; vmin!; "voltios"
LOCATE 22, 1: PRINT "voltaje de entrada="
LOCATE 41, 1: PRINT "Presione ESP BAR para terminar el control"
DIM STA$(3)
k% = 0
j = 0
CALL SOFTINIT
CALL INIT
defgraf
LINE (40, 25 + (10 - vmax!) * 10)-(629, 25 + (10 - vmax!) * 10), 2, , &H8888
LINE (40, 25 + (10 - vmin!) * 10)-(629, 25 + (10 - vmin!) * 10), 2, , &H8888

an$ = ""
WHILE an$ = ""
    j = j + 1
    CALL ARMAKE("salida%", 1!, -1, "ANOUT0")
        CALL ARPUTVALF("salida%", 1!, -1, "ANOUT0", 9.95, 0)
```

```
    CALL ANOUT("salida%", "ANOUT0", 1, 1, "nt", "tareasal")

    CALL inton(1, "mil")
    bfn$ = "tareasal"
    st% = 1
    WHILE st% <> 0

        CALL STATUS(bfn$, st%)
    WEND
    CALL INTOFF
CALL backclear
'-----------------
    thrl! = vmax!
    thrh! = vmax! + .1
    chm$ = "betw"
    bfn$ = "bt"
    cy% = 1

    CALL SCHMITRIG("ANLG0", thrl!, thrh!, chm$, 0, "bt", bfn$, cy%)
    CALL ANIN("ingreso%", 1!, "ANLG1", 1, 1, "wbt", "tarea")
    CALL ANIN("entrada%", 1!, "ANLG0", 1, -1, "nt", "")

    CALL inton(1, "mil")
    bfn$ = "tarea"
    st% = 1
    WHILE st% <> 0

        CALL STATUS(bfn$, st%)
        CALL ARLASTP("entrada%", lp!)
            CALL ARGETVALF("entrada%", 1!, -1, "ANLG0", valor!, 0)
            CALL ARGETVALF("ingreso%", 1!, -1, "ANLG1", sal!, 0)
            LOCATE 22, 22: PRINT valor!; "voltios   "
        IF k% > 580 THEN          reseteo de pantalla gráfica
        defgraf
LINE (40, 25 + (10 - vmax!) * 10)-(629, 25 + (10 - vmax!) * 10), 2, , &H8888
LINE (40, 25 + (10 - vmin!) * 10)-(629, 25 + (10 - vmin!) * 10), 2, , &H8888

        k% = 0
        END IF
        graf (k%)
        k% = k% + 1

    WEND
    CALL INTOFF

'--------------------
      CALL ARPUTVALF("salida%", 1!, -1, "ANOUT0", .08, 0)
    CALL ANOUT("salida%", "ANOUT0", 1, 1, "nt", "tareasal")

    CALL inton(1, "mil")
    bfn$ = "tareasal"
    st% = 1
    WHILE st% <> 0
        CALL STATUS(bfn$, st%)
    WEND
```

```
    CALL INTOFF
CALL ARDEL("entrada%")
CALL / RDEL("ingreso%")
'----------------

    thrl! = vmin! - .1
    thrh! = vmin!
    chm$ = "betw"
    bfn$ = "bt"
    cy% = 1

    CALL SCHMITRIG("ANLG0", thrl!, thrh!, chm$, 0, "bt", bfn$, cy%)
    CALL ANIN("ingreso%", 1!, "ANLG1", 1, 1, "wbt", "tarea")
    CALL ANIN("entrada%", 1!, "ANLG0", 1, -1, "nt", "")

    CALL inton(1, "mil")
    bfn$ = "tarea"
    st% = 1
    WHILE st% <> 0

        CALL STATUS(bfn$, st%)
        CALL ARLASTP("entrada%", lp!)
            CALL ARGETVALF("entrada%", 1!, -1, "ANLG0", valor!, 0)
            CALL ARGETVALF("ingreso%", 1!, -1, "ANLG1", sal!, 0)
            LOCATE 22, 22: PRINT valor!; "voltios   "
        IF k% > 580 THEN        'reseteo de pantalla gráfica
        defgraf
LINE (40, 25 + (10 - vmax!) * 10)-(629, 25 + (10 - vmax!) * 10), 2, , &H8888
LINE (40, 25 + (10 - vmin!) * 10)-(629, 25 + (10 - vmin!) * 10), 2, , &H8888

        k% = 0
        END IF
        graf (k%)
        k% = k% + 1
    WEND
    CALL INTOFF
CALL ARDEL("entrada%")
CALL ARDEL("ingreso%")
CALL ARDEL("salida%")
CALL backclear
an$ = INKEYS
WEND
WIDTH 80, 25
END SUB

SUB pantalla

IF m + n = 2 THEN
    I = 0
ELSEIF m + n = 3 AND m > n THEN
    I = 1
ELSEIF m + n = 3 AND m < n THEN
    I = 2
ELSEIF m + n = 4 AND m = n THEN
    I = 3
```

```
ELSEIF m + n = 4 AND m > n THEN
    I = 4
ELSEIF m + n = 4 AND m < n THEN
    I = 5
ELSEIF m + n = 5 AND m > n THEN
    I = 6
ELSEIF m + n = 5 AND m < n THEN
    I = 7
ELSEIF m + n = 5 AND m = n THEN
    I = 8
END IF

END SUB

SUB PID                          'Rutina que calcula el control PID de una planta
DIM c(4), v(4)
repita:
CLS
SCREEN 9
LINE (0, 0)-(639, 350), 2, BF
LINE (4, 4)-(635, 345), 0, BF
COLOR 2, 1
COLOR 15

CALL SOFTINIT
CALL INIT
TMIN% = 90
LOCATE 3, 10: PRINT "ESTUDIO DEL SISTEMA DE ADQUISICION DE DATOS KEITHLEY 500A"
LOCATE 4, 10: PRINT "              Y SUS APLICACIONES"
LOCATE 6, 10: PRINT "    CONTROLADOR P.I.D. UNIVARIABLE EN TIEMPO REAL"
COLOR 2
LOCATE 9, 10: PRINT "Tiempo de duración del algoritmo(ms) = "; TMIN%
LOCATE 10, 10: PRINT "máximo estimado"
LINE (0, 90)-(639, 93), 2, BF
muestra:
LOCATE 12, 10: PRINT "Período de muestreo deseado     (ms) = "; : INPUT bintv%

PER% = INT(bintv% / TMIN%)
IF PER% = 0 THEN
LOCATE 14, 10: PRINT "Período de muestreo menor que el tiempo de duración"
LOCATE 15, 10: PRINT "del algoritmo."
GOTO muestra
END IF
boutv% = bintv% / PER%          'optimización del tiempo de salida
bintv% = PER% * boutv%          'ajuste del tiempo de muestreo

LOCATE 17, 10: PRINT "Período de muestreo ajustado    (ms) = "; bintv%
LOCATE 19, 10: PRINT "Retardo del dato de salida      (ms) = "; boutv%
bintv% = bintv%
boutv% = boutv%
k% = 0
FOR L = 1 TO 4000: FOR ll = 1 TO 20: NEXT: NEXT
'----------INGRESO DE DATOS NECESARIOS PARA ALGORITMO----------

CLS
```

```
LINE (0, 0)-(639, 350), 2, BF
LINE (4, 4)-(635, 345), 0, BF
LINE (0, 90)-(639, 93), 2, BF
COLOR 2, 1:COLOR 15

LOCATE 3, 10: PRINT "ESTUDIO DEL SISTEMA DE ADQUISICION DE DATOS KEITHLEY 500A"
LOCATE 4, 10: PRINT "              Y SUS APLICACIONES"
LOCATE 6, 10: PRINT "    CONTROLADOR P.I.D. UNIVARIABLE EN TIEMPO REAL"
LOCATE 12, 10: PRINT "Ingreso de parámetros del controlador P.I.D. univariable"
LOCATE 15, 10: INPUT "Kp      ="; tkp
LOCATE 16, 10: INPUT "Ki      ="; tki
LOCATE 17, 10: INPUT "Kd      ="; tkd
LOCATE 18, 10: INPUT "Referencia="; r

'-------------------------------------------------------------

'------CALCULOS PREVIOS DEL ALGORITMO (FUERA DEL LAZO DE CONTROL)------

t = bintv% / 1000
c(1) = 1
c(2) = tkp + tki * t / 2 + tkd / t
c(3) = tki * t / 2 - tkp - 2 * tkd / t
c(4) = tkd / t
FOR L = 1 TO 4
    v(L) = 0
NEXT
'-------------------------------------------------------------

'----------SETEO DE LAS TAREAS DE BACKGROUND----------

DEP! = 1
CALL ANIN("entrada%", DEP!, "ANLG0", bintv%, -1, "nt", "tarea1")
CALL ANIN("acumule%", 200!, "ANLG0 ANLG1", bintv%, 1, "nt", "")
CALL ARMAKE("salida%", DEP!, -1, "ANOUT0")

CALL ANOUT("salida%", "ANOUT0", boutv%, -1, "nt", "tarea2")

FOR I! = 1 TO DEP!
    CALL ARPUTVALF("salida%", I!, -1, "ANOUT0", 0!, 0)
NEXT I!

'-------------------------------------------------------------
'Lazo creado con el afán de tener todo el arreglo referenciado a cero

LOCATE 21, 10: PRINT "presione SPACE BAR para iniciar el control"
COLOR 2
DO
    an$ = INKEY$
LOOP WHILE an$ = ""
SCREEN 9
WIDTH 80, 43
CLS
defgraf
COLOR 9
LOCATE 1, 5: PRINT "ESTUDIO DEL SISTEMA DE ADQUISICION DE DATOS KEYTLHEY 500A Y APLICACIONES"
```

```
COLOR 2
LOCATE 20, 10: PRINT "Tiempo Real"
LOCATE 22, 10: PRINT "Control en Proceso"
LOCATE 41, 10: PRINT "SPACE BAR para terminar proceso"
LINE (40, 125 - 10 * r)-(40, 125 - 10 * r), 7, , &H8888
COLOR 7
DT1 = 1
REDIM DT(0 TO 7) AS DOUBLE, TIM(0 TO 7)  AS INTEGER
TIM(0) = 1

CALL inton(1, "mil")
'ti1 = TIMER
CALL TIMERSTART(TIM(), "nt", "timer0")
'Comando que resetea el timer 0, en este caso y lo habilita para iniciar
'su cuenta, no espera disparo.

valor! = 0
an$ = ""
WHILE an$ = ""
     arn$ = "entrada%"
     CALL ARLASTP("entrada%", lp!)
     CALL ARLASTP("salida%", lp!)
          CALL ARGETVALF("entrada%", 1!, -1, "ANLG0", valor!, 0)
          LOCATE 20, 40: PRINT USING "Salida de la planta: ##.### v."; valor!

'************************ALGORITMO DE CONTROL COMO SUBRUTINA*********************

algoritmo

'*******************************************************************************

          CALL ARPUTVALF("salida%", 1!, -1, "ANOUT0", sal!, 0)
          LOCATE 22, 40: PRINT USING "Ley de Control    : ##.### v."; sal!

          IF k% > 580 THEN         'reseteo de pantalla gráfica
          defgraf
          k% = 0
          END IF
          graf (k%)
          k% = k% + 1
    an$ = INKEY$
'ti2 = TIMER
  DO
    CALL TIMERREAD(DT())
  LOOP UNTIL DT(0) / bintv% > DT1
  DT1 = INT(DT(0) / bintv% + 1)

'Sirve para asegurar que se ocupará todo el tiempo restante de la interrupción
'recibir otro dato.
'LOCATE 1, 1: PRINT (ti2 - ti1) * 1000 * 1.3
WEND

CALL INTOFF
CALL arwrite("acumule%", "pid.prn", 0, 4, bintv%, 1)
CALL ARDEL("entrada%")
```

```
CALL ARDEL("salida%")
pregp:
LOCATE 41, 10: INPUT "Desea realizar un nuevo control"; p$
IF p$ = "s" THEN
    SCREEN 0
    WIDTH 80, 25
    GOTO repita
ELSEIF p$ = "n" THEN
    GOTO endsubpid
ELSE
    GOTO pregp
END IF
endsubpid:
WIDTH 80, 25
END SUB

SUB REALEST                    'Rutina que calcula la realimentación de estado de un sistema multivariable
                               'de hasta tres entradas y tres salidas
CLS
COLOR 4
CALL SOFTINIT
CALL INIT

LOCATE 2, 10: INPUT "Tiempo de duración del algoritmo"; TMIN!%
muestrar:
LOCATE 5, 10: INPUT "Período de muestreo  (ms) "; bintv%
LOCATE 6, 10: INPUT "Número de entradas a la planta(máx. 3)"; m
LOCATE 7, 10: INPUT "Número de salidas de la planta(máx. 3)"; n
DIM bar1!(m), bar2!(n), ko!(m), k!(m, n), bar3!(m)
'-------------------------------------------------------------***
FOR I = 0 TO n - 1
pregunte:
    LOCATE 9 + I, 10: PRINT "Ingrese referencia (volts) de la salida "; I + 1;

    INPUT "="; ko!(I)
    IF ko!(I) > 10 THEN
    LOCATE 20, 10: PRINT "La referencia debe ser menor que 10V."
    GOTO pregunte
    ELSEIF ko!(I) < 0 THEN
    LOCATE 20, 10: PRINT "La referencia debe ser mayor que 0V.  "
    GOTO pregunte
    END IF
NEXT
LOCATE 13, 10: PRINT "Ingrese los coeficientes de la Matriz K de realimentación"
FOR L = 0 TO n - 1
    FOR j = 0 TO m - 1
        LOCATE 15 + j, 10 + 15 * L: PRINT "k("; L + 1; ","; j + 1; ")="; : INPUT k!(j, L)
    NEXT
NEXT

'-------------------------------------------------------------***
PER% = INT(bintv% / TMIN%)
IF PER% = 0 THEN
LOCATE 14, 10: PRINT "Período de muestreo menor que el tiempo de duración"
LOCATE 15, 10: PRINT "del algoritmo."
```

```
GOTO muestrar
END IF

boutv% = bintv% / PER%          'optimización del tiempo de salida
bintv% = PER% * boutv%          'ajuste del tiempo de muestreo

CLS
LOCATE 1, 15: PRINT "Período de muestreo ajustado (ms)"; bintv%
LOCATE 3, 15: PRINT "Retardo del dato de salida  (ms)"; boutv%
bintv% = bintv%
boutv% = boutv%

CALL pantalla

'-----------SETEO DE LAS TAREAS DE BACKGROUND----------

DEP! = 1

CALL ANIN("entrada%", DEP!, "ANLG0 ANLG1 ANLG2", bintv%, -1, "nt", "tarea1")
CALL ANIN("realim%", 200!, "ANLG0 ANLG1 ANLG2 ANLG3 ANLG4 ANLG5", bintv%, 1, "nt", "")
CALL ARMAKE("salida%", DEP!, -1, "ANOUT0 ANOUT1 ANOUT2")
CALL ANOUT("salida%", "ANOUT0 ANOUT1 ANOUT2", boutv%, -1, "nt", "tarea2")

FOR I! = 1 TO DEP!
    CALL ARPUTVALF("salida%", I!, -1, "ANOUT0", 0!, 0)
    CALL ARPUTVALF("salida%", I!, -1, "ANOUT1", 0!, 0)
    CALL ARPUTVALF("salida%", I!, -1, "ANOUT2", 0!, 0)
NEXT I!

'-------------------------------------------------------
'Lazo creado con el afán de tener todo el arreglo referenciado a cero

LOCATE 21, 3: PRINT "presione SPACE BAR para iniciar el control"
COLOR 2
DO
  an$ = INKEY$
LOOP WHILE an$ = ""
  SCREEN 9
  WIDTH 80, 43
  CLS
defgrafic
COLOR 7

DT1 = 1
DIM DT(0 TO 7) AS DOUBLE, TIM(0 TO 7) AS INTEGER
TIM(0) = 1
CALL inton(1, "mil")

CALL TIMERSTART(TIM(), "nt", "timer0")
'Comando que resetea el timer 0, en este caso y lo habilita para iniciar
'su cuenta, no espera disparo.
valor! = 0
an$ = ""
WHILE an$ = ""
    arn$ = "entrada%"
```

```
      CALL ARLASTP("entrada%", lp!)
      CALL ARLASTP("salida%", lp!)
FOR ii = 0 TO m - 1
  SELECT CASE ii
    CASE 0
      inl$ = "ANLG0"
    CASE 1
      inl$ = "ANLG1"
    CASE 2
      inl$ = "ANLG2"
   END SELECT
    CALL ARGETVALF("entrada%", 1!, -1, inl$, bar2!(ii), 0)
NEXT

'**********************ALGORITMO DE CONTROL COMO SUBRUTINA**********************

algomul

'**********************REALIMENTACION DE ESTADO-**********************

FOR ii = 0 TO n - 1
  SELECT CASE ii
    CASE 0
      oul$ = "ANOUT0"
    CASE 1
      oul$ = "ANOUT1"
    CASE 2
      oul$ = "ANOUT2"
   END SELECT

    CALL ARPUTVALF("salida%", 1!, -1, oul$, bar1!(ii), 0)

'-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
    IF I = 0 THEN
     IF k% > 580 THEN
        defgrafic
        k% = 0
     END IF
     grafic1 (k%)
     k% = k% + 1
     LOCATE 21, 1: PRINT k%
    ELSEIF I > 0 THEN
     IF k% > 280 THEN
        defgrafic
        k% = 0
     END IF
     grafic2 (k%)
     k% = k% + 1
    END IF
'-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
NEXT
    an$ = INKEY$
  DO
    CALL TIMERREAD(DT())
'LOCATE 5, 40: PRINT dt1, dt(0)
```

```
    LOOP UNTIL DT(0) / bintv% > DT1
    DT1 = INT(DT(0) / bintv% + 1)
'Sirve para asegurar que se ocupará todo el tiempo restante de la interrupción
'recibir otro dato.
WEND

CALL INTOFF
CALL arwrite("realim%", "realest.prn", 0, 4, bintv%, 1)
CALL ARDEL("entrada%")
CALL ARDEL("salida%")
'Comandos que borran los arreglos de memoria que se utilizaron en el programa.
WIDTH 80, 25
END SUB

SUB TRESNIV                    'Rutina que calcula el control ON-OFF tres niveles de una planta
CLS
LINE (0, 0)-(600, 335), 10, B
LINE (3, 3)-(597, 332), 10, B
COLOR 14
LOCATE 3, 10: PRINT "control TRES NIVELES a base de comandos de disparo"
LOCATE 6, 10: INPUT "Voltaje medio de control (0 < VMED < 10)"; volt!
LOCATE 13, 1: PRINT "INDICACIONES: Sólo aparecerán si existe un error en los datos"
COLOR 15
pregt:
LOCATE 8, 10: INPUT "valor en voltios del valor máximo de salida"; vmax!
IF vmax! > 10 THEN
LOCATE 15, 10: PRINT "valor máximo no debe exceder los 10V          "
GOTO pregt
END IF
preg2:
LOCATE 10, 10: PRINT "valor en voltios del valor medio ( <"; volt!; "V )"; : INPUT vmed!
IF vmed! > vmax! THEN
LOCATE 15, 10: PRINT "valor medio menor que valor máximo          "
GOTO preg2
ELSEIF vmed! > volt! THEN
LOCATE 15, 10: PRINT "valor medio menor que "; volt!; "v                    "
GOTO preg2
END IF
preg3:
LOCATE 12, 10: INPUT "valor en voltios del valor mínimo de salida"; vmin!
IF vmin! > vmed! THEN
LOCATE 15, 10: PRINT "El valor mínimo debe ser menor que el valor medio"
GOTO preg3
END IF

SCREEN 9
WIDTH 80, 43
CLS

LOCATE 1, 1: PRINT "Control 3 NIVELES: entre"; vmax!; " y "; vmin!; "voltios"
'LOCATE 3, 1: PRINT CHR$(34); "tarea"; CHR$(34); "está ahora"
LOCATE 22, 1: PRINT "voltaje de entrada="
DIM STA$(3)
k% = 0
```

```
j = 0
CALL SOFTINIT
CALL INIT
defgraf
LINE (40, 25 + (10 - vmax!) * 10)-(629, 25 + (10 - vmax!) * 10), 2, , &H8888
LINE (40, 25 + (10 - vmin!) * 10)-(629, 25 + (10 - vmin!) * 10), 2, , &H8888
LINE (40, 25 + (10 - vmed!) * 10)-(629, 25 + (10 - vmed!) * 10), 2, , &H8888
an$ = ""
WHILE an$ = ""
    j = j + 1
    CALL ARMAKE("salida%", 1!, -1, "ANOUT0")
       CALL ARPUTVALF("salida%", 1!, -1, "ANOUT0", volt!, 0)
      END
    CALL ANOUT("salida%", "ANOUT0", 1, 1, "nt", "tareasal")
    CALL inton(1, "mil")
    bfn$ = "tareasal"
    st% = 1
    WHILE st% <> 0
        CALL STATUS(bfn$, st%)
    WEND
    CALL INTOFF
    thrl! = vmed! - .1
    thrh! = vmed!
    chm$ = "above"
    bfn$ = "bt"
    cy% = 1
    CALL SCHMITRIG("ANLG0", thrl!, thrh!, chm$, 0, "bt", bfn$, cy%)
    CALL ANIN("ingreso%", 1!, "ANLG1", 1, 1, "wbt", "tarea")
    CALL ANIN("entrada%", 1!, "ANLG0", 1, -1, "nt", "")
    CALL inton(1, "mil")
    bfn$ = "tarea"
    st% = 1
    WHILE st% <> 0
        CALL STATUS(bfn$, st%)
        CALL ARLASTP("entrada%", lp!)
            CALL ARGETVALF("entrada%", 1!, -1, "ANLG0", valor!, 0)
            CALL ARGETVALF("ingreso%", 1!, -1, "ANLG1", sal!, 0)
            LOCATE 22, 22: PRINT valor!; "voltios   "
        IF k% > 580 THEN        'reseteo de pantalla gráfica
        defgraf
LINE (40, 25 + (10 - vmax!) * 10)-(629, 25 + (10 - vmax!) * 10), 2, , &H8888
LINE (40, 25 + (10 - vmin!) * 10)-(629, 25 + (10 - vmin!) * 10), 2, , &H8888
LINE (40, 25 + (10 - vmed!) * 10)-(629, 25 + (10 - vmed!) * 10), 2, , &H8888
        k% = 0
        END IF
        graf (k%)
        k% = k% + 1
    WEND
    CALL INTOFF
'-------------------------------------------------------
    CALL backclear
    CALL ARDEL("entrada%")
    CALL ARDEL("ingreso%")
       CALL ARPUTVALF("salida%", 1!, -1, "ANOUT0", 9.95, 0)
    CALL ANOUT("salida%", "ANOUT0", 1, 1, "nt", "tareasal")
```

```
        CALL inton(1, "mil")
        bfn$ = "tareasal"
        st% = 1
        WHILE st% <> 0
            CALL STATUS(bfn$, st%)
        WEND
        CALL INTOFF
'-----------------
        thrl! = vmax! - .1
        thrh! = vmax!
        chm$ = "above"
        bfn$ = "bt"
        cy% = 1
        CALL SCHMITRIG("ANLG0", thrl!, thrh!, chm$, 0, "bt", bfn$, cy%)
        CALL ANIN("ingreso%", 1!, "ANLG1", 1, 1, "wbt", "tarea")
        CALL ANIN("entrada%", 1!, "ANLG0", 1, -1, "nt", "")

        CALL inton(1, "mil")
        bfn$ = "tarea"
        st% = 1
        WHILE st% <> 0
            CALL STATUS(bfn$, st%)
            CALL ARLASTP("entrada%", lp!)
                CALL ARGETVALF("entrada%", 1!, -1, "ANLG0", valor!, 0)
                CALL ARGETVALF("ingreso%", 1!, -1, "ANLG1", sal!, 0)
                LOCATE 22, 22: PRINT valor!; "voltios   "
            IF k% > 580 THEN          'reseteo de pantalla gráfica
            defgraf
LINE (40, 25 + (10 - vmax!) * 10)-(629, 25 + (10 - vmax!) * 10), 2, , &H8888
LINE (40, 25 + (10 - vmin!) * 10)-(629, 25 + (10 - vmin!) * 10), 2, , &H8888
LINE (40, 25 + (10 - vmed!) * 10)-(629, 25 + (10 - vmed!) * 10), 2, , &H8888
            k% = 0
            END IF
            graf (k%)
            k% = k% + 1
        WEND
        CALL INTOFF

'--------------------
        CALL ARPUTVALF("salida%", 1!, -1, "ANOUT0", .08, 0)
        CALL ANOUT("salida%", "ANOUT0", 1, 1, "nt", "tareasal")
        CALL inton(1, "mil")
        bfn$ = "tareasal"
        st% = 1
        WHILE st% <> 0
            CALL STATUS(bfn$, st%)
        WEND
        CALL INTOFF
CALL ARDEL("entrada%")
CALL ARDEL("ingreso%")
'-----------------

        thrl! = vmin!
        thrh! = vmin! + .1
```

```
        chm$ = "below"
        bfn$ = "bt"
        cy% = 0

        CALL ANIN("entrada%", 1!, "ANLG0", 1, -1, "nt", "")
        CALL ANIN("ingreso%", 1!, "ANLG1", 1, 1, "wgo", "tarea")
        CALL SCHMITRIG("ANLG0", thr!, thrh!, chm$, 0, "bt", "", cy%)
        CALL gonov("wbt", "tareago")

        CALL inton(1, "mil")
        bfn$ = "tarea"
        st% = 1
        WHILE st% <> 0 OR gst% <> 0
            CALL STATUS(bfn$, st%)
            CALL STATUS("tareago", gst%)
            CALL ARLASTP("entrada%", lp!)
                CALL ARGETVALF("entrada%", 1!, -1, "ANLG0", valor!, 0)
                CALL ARGETVALF("ingreso%", 1!, -1, "ANLG1", sal!, 0)
                LOCATE 22, 22: PRINT valor!; "voltios   "
            IF k% > 580 THEN          'reseteo de pantalla gráfica
            defgraf
LINE (40, 25 + (10 - vmax!) * 10)-(629, 25 + (10 - vmax!) * 10), 2, , &H8888
LINE (40, 25 + (10 - vmin!) * 10)-(629, 25 + (10 - vmin!) * 10), 2, , &H8888
LINE (40, 25 + (10 - vmed!) * 10)-(629, 25 + (10 - vmed!) * 10), 2, , &H8888
            k% = 0
            END IF
            graf (k%)
            k% = k% + 1
        WEND
        CALL INTOFF
CALL ARDEL("entrada%")
CALL ARDEL("ingreso%")
CALL ARDEL("salida%")
an$ = INKEY$
WEND
WIDTH 80, 25
'CALL arwrite("tres$", "tres.prn", 0, 4, 30, 1)

END SUB

SUB  vent1                          'Rutina que define una ventana gráfica.

LINE (0, 12)-(639, 140), 1, BF
LINE (0, 12)-(639, 140), 15, B
'-------
LINE (40, 25)-(629, 25), 5, , &H8888
'-------
LINE (30, 125)-(629, 125), 7
LOCATE 17, 40
COLOR 15
PRINT "TIEMPO"
LOCATE 5, 2: PRINT "in"
LINE (40, 20)-(40, 130), 7

END SUB
```

```
SUB  vent10                    'Rutina que define una ventana gráfica.

LINE (324, 120)-(639, 220), 1, BF
LINE (324, 120)-(639, 220), 15, B
'--------
LINE (340, 125)-(340, 215), 7
LINE (334, 210)-(620, 210), 7
'--------
LINE (340, 130)-(620, 130), 5, , &H8888

END SUB

SUB  vent11                    'Rutina que define una ventana gráfica.

LINE (0, 230)-(314, 330), 1, BF
LINE (0, 230)-(314, 330), 15, B
'--------
LINE (20, 235)-(20, 325), 7
LINE (12, 320)-(300, 320), 7
'--------
LINE (20, 240)-(300, 240), 5, , &H8888

END SUB

SUB  vent12                    'Rutina que define una ventana gráfica.

LINE (324, 230)-(639, 330), 1, BF
LINE (324, 230)-(639, 330), 15, B
'--------
LINE (340, 235)-(340, 325), 7
LINE (334, 320)-(620, 320), 7
'--------
LINE (340, 240)-(620, 240), 5, , &H8888

END SUB

SUB  vent2                     'Rutina que define una ventana gráfica.

LINE (0, 185)-(639, 315), 1, BF
LINE (0, 185)-(639, 315), 15, B
'--------
LINE (40, 200)-(629, 200), 5, , &H8888
'--------
LINE (30, 300)-(629, 300), 7
LOCATE 39, 40
COLOR 15
PRINT "TIEMPO"
LOCATE 27, 2: PRINT "out"
LINE (40, 195)-(40, 305), 7

END SUB

SUB  vent3                     'Rutina que define una ventana gráfica.
```

```
LINE (0, 12)-(314, 140), 1, BF
LINE (0, 12)-(314, 140), 15, B
'--------
LINE (20, 20)-(20, 130), 7
LINE (12, 125)-(300, 125), 7
'--------
LINE (20, 25)-(300, 25), 5, , &H8888

END SUB

SUB  vent4                    'Rutina que define una ventana gráfica.

LINE (324, 12)-(639, 140), 1, BF
LINE (324, 12)-(639, 140), 15, B
'--------
LINE (340, 20)-(340, 130), 7
LINE (334, 125)-(620, 125), 7
'--------
LINE (340, 25)-(620, 25), 5, , &H8888

END SUB

SUB  vent5                    'Rutina que define una ventana gráfica.

LINE (0, 185)-(314, 315), 1, BF
LINE (0, 185)-(314, 315), 15, B
'--------
LINE (20, 195)-(20, 305), 7
LINE (12, 300)-(300, 300), 7
'--------
LINE (20, 200)-(300, 200), 5, , &H8888

END SUB

SUB  vent6                    'Rutina que define una ventana gráfica.

LINE (324, 185)-(639, 315), 1, BF
LINE (324, 185)-(639, 315), 15, B
'--------
LINE (340, 195)-(340, 305), 7
LINE (334, 300)-(620, 300), 7
'--------
LINE (340, 200)-(620, 200), 5, , &H8888

END SUB

SUB  vent7                    'Rutina que define una ventana gráfica.

LINE (0, 12)-(314, 110), 1, BF
LINE (0, 12)-(314, 110), 15, B
'--------
LINE (20, 15)-(20, 105), 7
LINE (12, 100)-(300, 100), 7
'--------
LINE (20, 20)-(300, 20), 5, , &H8888
```

```
END SUB

SUB  vent8                    'Rutina que define una ventana gráfica.

LINE (324, 12)-(639, 110), 1, BF
LINE (324, 12)-(639, 110), 15, B
'-------
LINE (340, 15)-(340, 105), 7
LINE (334, 100)-(620, 100), 7
'--------
LINE (340, 20)-(620, 20), 5, , &H8888

END SUB

SUB  vent9                    'Rutina que define una ventana gráfica.

LINE (0, 120)-(314, 220), 1, BF
LINE (0, 120)-(314, 220), 15, B
'--------
LINE (20, 125)-(20, 215), 7
LINE (12, 210)-(300, 210), 7
'--------
LINE (20, 130)-(300, 130), 5, , &H8888

END SUB
```

4.3.- CONCLUSIONES.-

- Al terminar el presente trabajo de tesis se puede afirmar que se ha cumplido con los objetivos trazados para éste, como es el de elaborar una biblioteca de rutinas para las diferentes tareas que se puede realizar con la estación KEITHLEY 500A, escribir una explicación de los comandos del Quick500 y aplicar este estudio tanto en control como en instrumentación.

- Se ha cumplido entonces con uno de los objetivos fundamentales al iniciar este trabajo que es el de crear un manual explicativo de la utilización del sistema de adquisición de datos y control KEITHLEY 500A, tanto en software como en hardware, detallando los requerimientos, necesidades, alcance, utilidad, versatilidad, limitaciones, etc. de la estación antes mencionada.

- El usuario potencial del sistema de adquisición de datos y control KEITHLEY 500A cuenta ahora con una serie de rutinas modelo para realizar control en tiempo real, con el consiguiente ahorro de tiempo en lo que se refiere a familiarización con el equipo, escritura y prueba de las rutinas básicas de adquisición y salida de datos, procesamiento, instrumentación, etc..

- Los resultados que arrojan los programas desarrollados son satisfactorios, toda vez que cumplen con lo que de ellos se esperaba, es decir, realizar instrumentación y control en tiempo real con la utilización de la estación KEITHLEY 500A.

- Una vez que se ha desarrollado una variedad de rutinas para la utilización de la estación KEITHLEY 500A, se ha comprobado que ésta se constituye en una herramienta valiosa para la realización de instrumentación inteligente, procesamiento de información, y aplicar todo ésto al control

digital.

- Se puede notar claramente que se ha puesto especial énfasis en el esquema de trabajo del background, ya que es este modo de trabajo el que permite la realización de control en tiempo real al tener la opción de fijar períodos de muestreo, permite realizar análisis de ondas adquiridas o calculadas ya que cuenta con la facilidad de crear arreglos de memoria para datos, permite realizar instrumentación, etc., pudiendo realizarse todas estas tareas de manera simultánea.

- El esquema de trabajo de foreground no tiene las opciones mencionadas para el background, por lo que únicamente puede ser utilizado para realizar instrumentación. Si el objetivo de una aplicación fuera solamente realizar instrumentación, se recomienda utilizar este esquema de trabajo, ya que permite muestrear las señales de entrada o de salida con una mayor frecuencia máxima (32 KHz. máximo) que el esquema de background (10KHz. máximo).

- El esquema de background trabaja con interrupciones no mascarables, es decir de máxima prioridad, por lo que las tareas de background nunca pueden dejar de realizarse. Si se realizan tareas de foreground en base a datos adquiridos o de salida que se muestrean a frecuencias altas es posible que no se alcance a realizar dichas tareas para todos los datos.

- Esto no tiene mucha importancia cuando se trata de una simple instrumentación, sin embargo, si se trata de un algoritmo de control o de instrumentación inteligente (con alarmas por ejemplo) si es importante. Debido a esto se recomienda fijar períodos mínimos de muestreo de acuerdo a la extensión del programa.

- Se ha creido conveniente desarrollar una pequeña rutina que le permita al usuario generar todo tipo de ondas y conservarlas tanto en archivos propios del Quick500, como en archivos importables a Lotus, para posteriores usos.

- Una de las virtudes del Quick500, es la posibilidad de exportar datos a hojas electrónicas, programas de análisis, bases de datos, etc. para los diferentes usos que se puede dar a los datos adquiridos en tiempo real. Para la obtención de los resultados del presente trabajo de tesis se ha utilizado esta opción para graficar los datos adquiridos o de salida en papel. Resulta de gran importancia el contar con esta opción ya que permite obtener reportes que se presentan a manera de informe, ya que la observación en pantalla sirve más bien para efectos de instrumentación, o para fines didácticos.

- El procesamiento de información que se puede realizar con el Quick500 es bastante elemental, ya que es un paquete de tipo general. Existen paquetes diseñados exclusivamente para el procesamiento de datos, como es el caso del "ASYSTANT", programa que permite entre otras cosas obtener la transformada rápida de Fourier, espectro de potencia, inversión de matrices, obtener determinantes, convoluciones, resolución de ecuaciones de diferencias, operaciones aritméticas básicas, resolución de polinomios, generar ondas, obtener información estadística, graficación, etc..

- Por esta razón, se ha desarrollado el algoritmo de cálculo de la transformada rápida de Fourier y el espectro de densidad de potencia, para tratar de suplir la falta de paquetes poderosos de software como es el "ASYSTANT", cuyo costo es sumamente elevado.

- Se ha desarrollado una serie de rutinas básicas de control como PID discreto,

realimentación de estado, controles ON-OFF, a manera de ejemplo o modelo, con la intención de dejar una puerta abierta para que en un futuro mediato, se desarrolle otros algoritmos de control y estos puedan ser implementados en tiempo real utlizando el equipo con que se cuenta.

- Para dar una aplicación a las salidas digitales con que cuenta el sistema, se ha querido implementar un control para un motor a pasos simulando algunas de las tareas que se podría realizar con el Quick500, si se contase con el módulo STEP1 o STEP2, construidos para el manejo de este tipo de motores.

- El hardware con el que actualmente se cuenta es muy limitado pues, únicamente se cuenta con tres módulos: entrada de datos análogos, salida de datos análogos y salida de datos digitales. Sin embargo, el equipo tiene la capacidad de contar con muchos módulos más para entradas digitales, para instrumentación (es decir para termocuplas, strain gages, LVDT's/RVDT's), control de potencia, control de motores a pasos, para salidas de corriente, entradas de frecuencia, tarjeta aceleradora de gráficos, etc.. Entonces se recomienda hacer lo posible por ampliar el campo de acción de la estación KEITHLEY 500A adquiriendo un mayor número de módulos.

- Igualmente en lo referente al software, el campo de acción es sumamente limitado, pues se cuenta únicamente con el Quick500, que es para la utilización básica de la estación KEITHLEY 500A, existiendo a más del paquete mencionado, otros paquetes de utilización más puntual como el ASYST (software para adquisición y análisis científico de datos), ASYSTANT (software para la adquisición y análisis científico de datos listo para su utilización), ASYSTANT GPIB, DADiSP II (adquisición de datos y procesamiento digital de señales), etc., los que también se recomienda, dentro de

lo posible sean adquiridos, para ampliar el campo de acción en lo referente al control en tiempo real dentro de la Facultad de Ingeniería Eléctrica.

- Este es quizá el primer trabajo de este tipo que se realiza en la Facultad de Ingeniería Eléctrica, por lo que se espera que se constituya en el arranque para futuras aplicaciones del control en tiempo real.

- Se ha hablado de la insuficiencia del equipo con el que se cuenta actualmente, dicha insuficiencia se evidencia en la utilización de los gráficos del Quick500, los mismos que han sido dejados de lado debidi a la falta de la tarjeta aceleradora de gráficos, sin la cual resulta impráctico utilizar los comandos de gráficos del paquete, por lo que finalmente se decidió generar una rutina de gráficos propia desarrollada en BASIC, la cual logra ahorro de tiempo y además mejora la resolución.

- No está por demás indicar que el objetivo del presente trabajo de tesis no es el optimizar el control de sistemas sino más bien, utilizar el sistema de adquisición de datos y control KEITHLEY 500A en aplicaciones básicas de control que puedan servir de punto de partida para aplicaciones futuras.

BIBLIOGRAFIA:

1.- KEITHLEY, "Quick500 Data Acquisition and Control Software", 1988.

2.- OGATA KATSUHIKO, "Ingeniería de Control Moderna", Prentice-Hall International, 1973.

3.- ANGULO JOSE Ma., "Curso de Robótica", Paraninfo, 1985.

4.- BURR-BROWN CORPORATION, Data acquisition and control, 1986.

5.- MICROSOFT, "Microsoft Quick Basic", Microsoft Corporation, 1988.

6.- MICROSOFT, "Microsoft Quick Basic 4.0", Microsoft Corporation, 1987.

7.- IBM, "Disk operating system version 3.3", IBM Corporation, 1987.

8.- KUO BENJAMIN, "Digital Control Systems", Holt-Saunders International Editions, 1981.

9.- ASTROM KARL, "Adaptive Control", Addison Wesley, 1989

10.- MOLER CLEVE, "PC Matlab", The Mathworks, Inc., 1984

11.- FRANKLIN G. & POWELL D., "Digital control of dynamic systems", Addison Wesley, 1981.

12.- SERIES 500.- "Measurements and control systems", 1988

13.- ORTIZ, HUGO.- " Control adaptivo con modelo de referencia para sistemas discretos",1987

MANUAL DE USO

El software desarrollado en el presente trabajo de tesis para el aprovechamiento de la estación KEITHLEY 500A, ha sido agrupado en tres paquetes ejecutables, CAP20.EXE, CAP30.EXE y CAP40.EXE que abarcan respectivamente los siguientes temas: Software de entrada y salida de datos, software multitarea y software de aplicación.

Se ha desarrollado también el archivo TR.BAT para la ejecución del software donde se determina el path correspondiente y además se antepone el *QRUN.BAT* indispensable para ejecutar programas compilados en Quick500, entonces es suficiente tipear "TR" para la ejecución de los programas.

Al ejecutarse el programa la primera actividad que se cumple es verificar si la estación KEITHLEY 500A está presente o no, de no estarlo se genera una alarma y se aborta el programa. Si todo está correcto se presenta un menú maestro cuyo esquema de presentación es el siguiente:

ENTRADA Y SALIDA DE DATOS	(1)
SOFTWARE MULTITAREA	(2)
SOFTWARE DE APLICACION	(3)
TERMINAR	(4)

Seleccionando la última opción (4), se abandona el programa, retornando al sistema operativo. Las tres primeras opciones permiten trabajar en los programas respectivos, así:

Al escoger la opción (1), se ingresa al programa CAP20.EXE, en el cual se

ha desarrollado software de entrada y salida de datos con el objeto de lograr la familiarización del usuario con el equipo. Al correr el programa se tiene el siguiente menú de opciones:

ADQUISICION DE DATOS ANALOGOS	(1)
SALIDA DE DATOS ANALOGOS	(2)
SALIDA DE DATOS DIGITALES	(3)
TERMINAR SESION DE TRABAJO	(4)

De igual manera la última opción aborta el programa y retorna al menú maestro. Cualquiera de las demás opciones permite utilizar la estación de adquisición de datos y control KEITHLEY 500A, así, al escoger la opción (1) se accesa a las rutinas de adquisición de datos mediante canales análogos; la opción (2) permite utilizar las rutinas de salida de datos mediante canales análogos y; la opción (3) accesa a las rutinas de salida de datos mediante canales digitales. En cada una de estas opciones se obtiene en pantalla cuadros similares con las siguientes posibilidades:

UN SOLO CANAL EN MODO DE FOREGROUND	(1)
MULTICANAL EN MODO DE FOREGROUND	(2)
UN SOLO CANAL EN MODO DE BACKGROUND	(3)
MULTICANAL EN MODO DE BACKGROUND	(4)
SALIR AL MENU PRINCIPAL	(5)

En todos los casos en que se trabaja con un solo canal se lo hace con el

canal o puerto cero (ANLG0, ANOUT0, PORT0). Mientras que en los casos multivariables se instruye al usuario en que canales utilizar.Para los programas de salida de datos análogos se debe utilizar osciloscopio para visualizar los resultados.

Retomando el menú inicial, si se escoge la opción (2), se ingresa al programa CAP30.EXE que abarca los ejemplo de operación multitarea en donde se obtiene el siguiente cuadro de opciones:

E/S DE DATOS CON ALGORITMO DE CONTROL (INTERRUPCIONES)	(1)
CONTROL ON-OFF (DISPAROS)	(2)
GENERADOR DE ONDAS ELEMENTAL (GRAFICOS TR)	(3)
MANEJO DE ARREGLOS DE MEMORIA (ARREGLOS DE MEMORIA)	(4)
PROCESAMIENTO DE DATOS (PROCESAMIENTO DE DATOS)	(5)
TRANSFORMADA RAPIDA DE FOURIER (TRF)	(6)
TERMINAR LA SESION DE TRABAJO	(7)

Al igual que siempre al escogerse la última opción (7), se aborta el programa y se regresa al menú maestro. Si se escoge la opción (1), se accesa a la rutina de entrada y salida de datos análogos, el cual utiliza los canales análogo de entrada ANLG0, y de salida ANOUT0. Este programa es el un modelo que se puede aplicar para introducir cualquier algoritmo de control en tiempo real. Se encuentra implementado como algoritmo una simple multiplicación que duplica en la salida una señal adquirida, por lo que se recomienda no utilizar esta rutina en un sistema físico ya que se constituiría una realimentación positiva. Incluye avisos en pantalla cuando los niveles de voltaje permitidos han sido superados y lecturas visuales de los niveles de voltaje completándose así una

instrumentación adecuada paralela a la ejecución del algoritmo de control.

Al escogerse la opción (2), se accesa a un control ON-OFF que se ha incluido para presentar ejemplos de la utilización de los comandos de disparos. Utiliza el canal ANLG0 para la salida de la planta, el canal ANLG1 para adquirir los datos de la entrada de la planta (ley de control) que es enviada por el canal análogo de salida ANOUT0. Incluye una instrumentación similar a la de la opción (1).

La opción (3) tiene por objetivo utilizar los comandos de graficación en tiempo real del Quick500. Se ha implementado un generador de ondas elemental. Los resultados no son óptimos por lo explicado oportunamente, pero el objetivo es visualizar virtudes y defectos de la graficación en tiempo real utilizando dichos comandos.

Al escogerse la opción (4), se obtiene el programa que actúa sobre un arreglo de memoria utilizando prácticamente todos los comandos para el manejo de dichos arreglos de memoria para datos con que se cuenta. Se trata de una rutina de uso totalmente didáctico cuyo objetivo no es otro que familiarizar al usuario con los arreglos de memoria, para esto se adquiere datos análogos a través del canal ANLG0 , estos datos adquiridos acumulados en un arreglo de memoria son posteriormente almacenados en disco, luego se tiene la posibilidad de recuperar la información almacenada en disco. Tanto para el almacenamiento como para la recuperación se recomienda utilizar un nombre de extensión .DAT para el archivo DOS que se crea con los datos almacenados. Para la recuperación se debe asegurar que el archivo pedido esté presente en el directorio desde donde se corre el programa CAP30.EXE (se recomienda correr el programa desde el directorio C:\QB45).

Eligiendo la opción (5) se accesa al procesamiento de datos, donde se obtiene información básica de tipo estadístico y matemático acerca de una señal que puede adquirirse al momento o de una señal almacenada anteriormente.

En el caso de adquirirse la señal en tiempo real al momento de utilizar esta opción, se utilizará el canal análogo ANLG0. Si el análisis se va a realizar sobre una señal adquirida con anterioridad, ésta se puede recuperar de disco, pero se debe asegurar que esté presente en el directorio donde está el programa CAP30.EXE (de no estarlo se regresa al menú maestro).

La última opción posible es la (6), con la cual se puede obtener la transformada rápida de Fourier de una señal que se adquiere en ese momento o que ha sido almacenada en disco con anterioridad. Además se obtiene el espectro de potencia de la señal. El esquema de trabajo es igual al de procesamiento de datos.

Retomando el menú maestro, si se escoge la opción (3), se ingresa al programa CAP40.EXE que contiene los ejemplo de aplicación, el menú que se tiene es el siguiente:

CONTROLADOR P.I.D DISCRETO	(1)
REALIMENTACION DE ESTADO	(2)
CONTROLADOR ON-OFF	(3)
CONTROLADOR DE TRES NIVELES	(4)
CONTROL ADAPTIVO CON REFERENCIA A MODELO	(5)
CONTROL DE UN MOTOR A PASOS	(6)
TERMINAR LA SESION DE TRABAJO	(7)

Al igual que siempre al escogerse la última opción (7), se aborta el

programa y se regresa al menú maestro. Si se escoge la opción (1) se tiene un controlador PID discreto, el mismo que se ha implementado en el esquema de la rutina de entrada y salida de datos presentada en CAP3.EXE. Para su utilización, el canal ANLG0 muestrea la salida del sistema, mientras que el canal ANOUT0 entrega la ley de control a la planta. Adicionalmente para posteriores análisis se puede guardar un cierto número de datos en disco, para lo cual se debe conectar el canal ANLG1 a la entrada de la planta.

La segunda opción (2) es una realimentación de estado, que utiliza un esquema de entrada y salida de datos similar al del PID, pero para el caso multivariable.

Utilizando todo el diseño gráfico del programa es posible tener sistemas de hasta tres entradas y tres salidas, en otro caso la planta puede tener tantas entradas como canales de salida, y tantas salidas como canales de entrada tiene la estación KEITHLEY 500A. Se utilizan los canales ANLG0, ANLG1 y ANLG2, para las salidas 1, 2 y 3 de la planta y se utiliza los canales ANOUT0, ANOUT1 y ANOUT2 para las entradas 1, 2 y 3 de la planta o proceso.

Para el control ON-OFF que es la opción (3), se debe utilizar el canal ANLG0 para muestrear la salida del sistema y el canal ANLG1 para la entrada del sistema. Mediante el canal ANOUT0 se entrega la ley de control al sistema. La opción (4) es un control tres niveles de voltaje para la ley de control y se ha implementado con el ánimo de mostrar que es sencillo aumentar tareas mediante disparos, de forma que se puede llegar a elaborados algoritmos de control . Se utilizan los mismos canales que en el control ON-OFF.

La opción (5) es un control adaptivo con referencia a modelo, el mismo que utiliza el canal ANLG0 para muestrear la salida de la planta, el canal ANLG1

para la referencia, mientras que la ley de control se entrega mediante el canal ANOUT0, adicionalmente se conecta el canal ANLG2 a la entrada de la planta, para efectos de adquisición de datos.

Al escogerse la opción (6) se realiza el control de un motor a pasos mediante canales de salida digital, utilizándose los canales DOUT0 y DOUT1 para las dos bobinas del motor. Es posible escoger velocidad del motor, sentido de giro y cuatro tareas diferentes: posicionamiento, girar una vuelta completa, un número entero de vueltas y giro indefinido del motor.

En todos los casos los parámetros de los controladores se deben calcular OFF-LINE (fuera de línea).

Se ha considerado además que el usuario tiene un conocimiento adecuado de la teoría que se debe manejar para el uso correcto de las distintas técnicas de control que se incluyen en las aplicaciones respectivas, de manera que el ingreso de datos sea consistente y los resultados satisfagan los requerimientos. De cualquier forma los ejemplos que se muestran en los resultados son una buena guía para el uso adecuado de los programas, también resulta conveniente referirse a los diagramas de flujo de las diferentes rutinas.

LISTADO COMPLETO DEL "SOFTWARE DE ENTRADA Y SALIDA DE DATOS" (CAP20.BAS).

```
DECLARE SUB DIOUFORE ()
DECLARE SUB ANOUFORE ()
DECLARE SUB ANOUFOMU ()
DECLARE SUB ANOUBACK ()
DECLARE SUB ANOUBAMU ()
DECLARE SUB DIOUFOMU ()
DECLARE SUB DIOUBACK ()
DECLARE SUB DIOUBAMU ()
DECLARE SUB ANLGIN ()
DECLARE SUB ANALGOUT ()
DECLARE SUB DIGITOUT ()
DECLARE SUB ANINFORE ()
DECLARE SUB ANINFOMU ()
DECLARE SUB ANINBACK ()
DECLARE SUB ANINBAMU ()
'*********************************
'Esta parte del programa sensa si el equipo de adquisición de datos está
'presente, es decir si está encendido, si no lo está alarma al usuario de
'manera auditiva y de manera visual.
'*********************************
CLS: COLOR 10
LOCATE 3, 10: PRINT "       SISTEMA DE ADQUISICION DE DATOS Y CONTROL"
LOCATE 4, 20: PRINT "    KEITHLEY 500A Y APLICACIONES"
DEF SEG = &HCFF9
IF PEEK(&HB) = 255 THEN
      COLOR 0, 10
      BEEP
     LOCATE 11, 13: PRINT "  ERROR: Sistema de adquisición de datos apagado  "
     LOCATE 12, 13: PRINT "por favor enciéndalo y vuelva a correr el programa"
      BEEP: BEEP
      COLOR 10, 0
      DEF SEG
      END
END IF
'*********************************
opcion:
CLS: SCREEN 9
LINE (0, 0)-(600, 335), 11, B
LINE (3, 3)-(597, 332), 11, B
COLOR 11
LOCATE 3, 10: PRINT "       SISTEMA DE ADQUISICION DE DATOS Y CONTROL"
LOCATE 4, 20: PRINT "    KEITHLEY 500A Y APLICACIONES"
LOCATE 9, 20: PRINT "PAQUETES DE SOFTWARE PARA TIEMPO REAL:"
LOCATE 10, 20: PRINT "═══════════════════════════════"
COLOR 15
LOCATE 13, 5: PRINT "SOFTWARE DE ENTRADA Y SALIDA DE DATOS                (1)"
LOCATE 14, 5: PRINT "SOFTWARE MULTITAREA                                  (2)"
LOCATE 15, 5: PRINT "APLICACIONES                                         (3)"
LOCATE 16, 5: PRINT "SALIR                                                (4)"
preguntando:
LOCATE 21, 5: INPUT "Escoja una opción por favor (1-4):"; OP$
```

```
IF OP$ = "1" THEN
GOTO MENU
ELSEIF OP$ = "2" THEN
SHELL "CAP32"
ELSEIF OP$ = "3" THEN
SHELL "CAP4PRU"
ELSEIF OP$ = "4" THEN
END
ELSEIF OP$ > "4" OR OP$ < "1" THEN
GOTO preguntando
END IF
IF OP$ = "2" OR OP$ = "3" THEN GOTO opcion

MENU:
CLS
SCREEN 9
LINE (0, 0)-(600, 335), 10, B
LINE (3, 3)-(597, 332), 10, B
COLOR 2
LOCATE 3, 10: PRINT "       SOFTWARE DE ADQUISICION Y SALIDA DE DATOS"
LOCATE 4, 20: PRINT "     DEL SISTEMA KEITHLEY 500A"
LOCATE 9, 30: PRINT "MENU PRINCIPAL:"
LOCATE 10, 28: PRINT "═══════════════"
COLOR 15
LOCATE 13, 5: PRINT "ADQUISICION DE DATOS ANALOGOS                    (1)"
LOCATE 14, 5: PRINT "SALIDA DE DATOS ANALOGOS                         (2)"
LOCATE 15, 5: PRINT "SALIDA DE DATOS DIGITALES                        (3)"
LOCATE 16, 5: PRINT "TERMINAR LA SESION DE TRABAJO                    (4)"
pregunta1:
LOCATE 20, 5: INPUT "Escoja una opción por favor (1-4):"; OP$
IF OP$ = "1" THEN CALL ANLGIN: GOTO MENU
IF OP$ = "2" THEN CALL ANALGOUT: GOTO MENU
IF OP$ = "3" THEN CALL DIGITOUT: GOTO MENU
IF OP$ = "4" THEN GOTO opcion
IF OP$ > "4" THEN GOTO pregunta1

SUB ANALGOUT
MENU3:
CLS
SCREEN 9
LINE (0, 0)-(600, 335), 3, B
LINE (3, 3)-(597, 332), 3, B
COLOR 3
LOCATE 3, 10: PRINT "       SOFTWARE DE ADQUISICION Y SALIDA DE DATOS"
LOCATE 4, 20: PRINT "     DEL SISTEMA KEITHLEY 500A"
LOCATE 7, 20: PRINT "      SALIDA DE DATOS ANALOGOS:"
LOCATE 8, 18: PRINT "   ═══════════════════════"
COLOR 15
LOCATE 13, 5: PRINT "UN SOLO CANAL MODO DE FOREGROUND                 (1)"
LOCATE 14, 5: PRINT "MULTICANAL MODO DE FOREGROUND                    (2)"
LOCATE 15, 5: PRINT "UN SOLO CANAL MODO DE BACKGROUND                 (3)"
LOCATE 16, 5: PRINT "MILTICANAL MODO DE BACKGROUND                    (4)"
LOCATE 17, 5: PRINT "SALIR AL MENU PRINCIPAL                          (5)"
pregunta3:
```

```
LOCATE 22, 5: INPUT "Escoja una opción por favor (1-5):"; OP$

IF OP$ = "1" THEN CALL ANOUFORE: GOTO MENU3
IF OP$ = "2" THEN CALL ANOUFOMU: GOTO MENU3
IF OP$ = "3" THEN CALL ANOUBACK: GOTO MENU3
IF OP$ = "4" THEN CALL ANOUBAMU: GOTO MENU3
IF OP$ = "5" THEN GOTO FIN3
IF OP$ > "5" OR OP$ = "" THEN GOTO pregunta3
FIN3:
END SUB

SUB ANINBACK
'*************************************************************
'* Esta es una Subrutina que me permite realizar adqui- *
'* sición de datos desde un canal análogo, en el modo - *
'* de trabajo de BACKGROUND.                          *
'*************************************************************

CLS
COLOR 2
pregu:
LOCATE 5, 3: PRINT "indicaciones: cientos de microsegundos:  hmic"
LOCATE 6, 3: PRINT "          milisegundos          :  mil"
LOCATE 7, 3: PRINT "          segundos              :  sec"
LOCATE 8, 3: PRINT "          minutos               :  min"
LOCATE 3, 3: INPUT "Periodo de muestreo (UNIDADES DE TIEMPO)= "; tu$
IF tu$ = "hmic" THEN
GOTO siga
ELSEIF tu$ = "mil" THEN
GOTO siga
ELSEIF tu$ = "sec" THEN
GOTO siga
ELSEIF tu$ = "min" THEN
GOTO siga
END IF
GOTO pregu
siga:
LOCATE 10, 3: PRINT "Período de muestreo ("; tu$; : INPUT ") "; bintv%
pregu1:
LOCATE 14, 3: INPUT "Adquisición Definida (d) o Indefinida (i)"; a$
IF a$ = "d" THEN
LOCATE 16, 3: INPUT "Cuantos datos desea adquirir"; d!
GOTO siga1
ELSEIF a$ = "i" THEN
GOTO siga1
END IF
GOTO pregu1
siga1:
LOCATE 3, 3: PRINT "Inicio de la Adquisición de Datos"
CLS
4 LOCATE 24, 15: PRINT "presione SPACE BAR para terminar la adquisición"
LOCATE 2, 15: PRINT "Medidas realizadas por el canal análogo 0 (ANLG0)"
COLOR 1
```

```
CALL SOFTINIT
CALL INIT

'Estas dos instrucciones inicializan el SOFTWARE y el HARDWARE del sistema
'QUICK500 respectivamente.

ion$ = "ANLG0"
dep! = 20
l! = -dep!
'En las instrucciones anteriores se setea nombre del canal, periodo de mues-
'treo y profundidad del arreglo receptor de los datos
IF a$ = "i" THEN
CALL ANIN("entrada%", dep!, "ANLG0", bintv%, -1, "nt", "tarea1")
d! = 100000
ELSEIF a$ = "a" THEN
CALL ANIN("entrada%", dep!, "ANLG0", bintv%, -1, "nt", "tarea1")
END IF
CALL ANIN("aninba%", 400!, "ANLG0", bintv%, 1, "nt", "tarea1")

'Comando que setea la tarea de BACKGROUND, realiza el muestreo del canal de
'entrada análogo.
CALL INTON(1, tu$)

'Habilita las interrupciones
an$ = ""
WHILE an$ = ""
  arn$ = "entrada%"
  CALL arlastp("entrada%", lp!)

  'Comando que reporta cual fue el último elemento del arreglo al que se accesó
  IF lp! <> plp! THEN
    CALL argetvalf("entrada%", lp!, -1, "ANLG0", va!, 0)

    'saca un dato del arreglo donde se guardan los datos adquiridos
    IF lp! = 1 AND lip = 1 THEN
       l! = l! + dep!: COLOR 14: lip = 0
    ELSEIF lp! = 1 AND lip = 0 THEN
       l! = l! + dep!: COLOR 15: lip = 1
    END IF
    IF lp! + l! > d! THEN GOTO finau
    LOCATE 2 + lp!, 22: PRINT " medida # "; lp! + l!; "    ";
    LOCATE 2 + lp!, 45: PRINT USING "##.###      "; va!
    plp! = lp!
  END IF
finau:
  an$ = INKEY$
WEND

CALL INTOFF
CALL arwrite("aninba%", "aninback.prn", 0, 4, bintv%, 1)
'Deshabilita las interrupciones

END SUB
```

```
SUB  ANINBAMU
'*************************************************************
'* Esta es una Subrutina que me permite realizar entra-      *
'* da de datos con  varios canales analogos, en el mo-       *
'* do de trabajo de BACKGROUND.                              *
'*************************************************************

CLS

CALL SOFTINIT
CALL INIT

'Estas dos instrucciones inicializan el SOFTWARE y el HARDWARE del sistema
'QUICK500 respectivamente.
COLOR 2
pregua:
LOCATE 1, 3: PRINT "Adquisición de datos a través de 8 canales análogos"
LOCATE 5, 3: PRINT "indicaciones: cientos de microsegundos:  hmic"
LOCATE 6, 3: PRINT "              milisegundos          :  mil"
LOCATE 7, 3: PRINT "              segundos              :  sec"
LOCATE 8, 3: PRINT "              minutos               :  min"
LOCATE 3, 3: INPUT "Periodo de muestreo (UNIDADES DE TIEMPO)= "; tu$
LOCATE 10, 3: PRINT "Período de muestreo ("; tu$; : INPUT ") "; bintv%
IF tu$ = "hmic" THEN
GOTO sigaa
ELSEIF tu$ = "mil" THEN
GOTO sigaa
ELSEIF tu$ = "sec" THEN
GOTO sigaa
ELSEIF tu$ = "min" THEN
GOTO sigaa
END IF
GOTO pregua
sigaa:
dep! = 20
l! = -(dep!)
LOCATE 12, 3: INPUT "Cuantos canales desea muestrear (máximo 8)"; n
SELECT CASE n
    CASE 1
   ionl$ = "ANLG0"
    CASE 2
   ionl$ = "ANLG0 ANLG1"
    CASE 3
   ionl$ = "ANLG0 ANLG1 ANLG2"
    CASE 4
   ionl$ = "ANLG0 ANLG1 ANLG2 ANLG3"
    CASE 5
   ionl$ = "ANLG0 ANLG1 ANLG2 ANLG3 ANLG4"
    CASE 6
   ionl$ = "ANLG0 ANLG1 ANLG2 ANLG3 ANLG4 ANLG5"
    CASE 7
   ionl$ = "ANLG0 ANLG1 ANLG2 ANLG3 ANLG4 ANLG5 ANLG6"
    CASE 8
```

```
    ionl$ = "ANLG0 ANLG1 ANLG2 ANLG3 ANLG4 ANLG5 ANLG6 ANLG7"
END SELECT

LOCATE 15, 3: PRINT "Que canales (2) desea mostrar en pantalla"
LOCATE 17, 3: PRINT "Primer canal mostrado (0-"; n - 1; ")"; : INPUT c1
LOCATE 18, 3: PRINT "Segundo canal mostrado (0-"; n - 1; ")"; : INPUT c2
SELECT CASE c1
    CASE 0
        ion$ = "ANLG0"
    CASE 1
        ion$ = "ANLG1"
    CASE 2
        on$ = "ANLG2"
    CASE 3
        ion$ = "ANLG3"
    CASE 4
        ion$ = "ANLG4"
    CASE 5
        ion$ = "ANLG5"
    CASE 6
        ion$ = "ANLG6"
    CASE 7
        ion$ = "ANLG7"
END SELECT
SELECT CASE c2
    CASE 0
        iona$ = "ANLG0"
    CASE 1
        iona$ = "ANLG1"
    CASE 2
        iona$ = "ANLG2"
    CASE 3
        iona$ = "ANLG3"
    CASE 4
        iona$ = "ANLG4"
    CASE 5
        iona$ = "ANLG5"
    CASE 6
        iona$ = "ANLG6"
    CASE 7
        iona$ = "ANLG7"
END SELECT
DIM barn0!(20)
DIM barn1!(20)
CLS
LOCATE 1, 16: PRINT "ENTRADAS:          CANAL ANALG"; c1; "    CANAL ANALG"; c2
LOCATE 2, 14: PRINT "-----------------------------------------"
LOCATE 23, 14: PRINT "presione SPACE BAR para terminar la adquisición de datos"
COLOR 7
lp! = 1

CALL ANIN("entrada%", dep!, ionl$, bintv%, -1, "nt", "tarea1")
CALL ANIN("aninmul%", 200!, ionl$, bintv%, 1, "nt", "")
```

```
'comando que setea la entrada de datos a través de canales análogos como
'una tarea de BACKGROUND

CALL INTON(1, tu$)
'comando que habilita las interrupciones de BACKGROUND
an$ = ""
WHILE an$ = ""
      arn$ = "entrada%"
       CALL arlastr(arn$, lp!)

      'comando que indica la profundidad del último dato accesado
       IF lp! <> plp! THEN

            CALL ARGETF("entrada%", 1!, dep!, -1, ion$, barn0!(), 0)
            CALL ARGETF("entrada%", 1!, dep!, -1, iona$, barn1!(), 0)

           'comandos que recuperan el dato del arreglo de memoria me-
            'diante el parámetro barn!()
    IF lp! = 1 AND lip = 1 THEN
       l! = l! + dep!: COLOR 14: lip = 0
    ELSEIF lp! = 1 AND lip = 0 THEN
       l! = l! + dep!: COLOR 15: lip = 1
    END IF
            LOCATE 2 + lp!, 16: PRINT "medida#"; lp! + l!; "    "
            LOCATE 2 + lp!, 40: PRINT USING "##.###          ##.###     "; barn0!(lp!); barn1!(lp!)
            plp! = lp!
       END IF
      an$ = INKEY$
WEND

CALL INTOFF
CALL arwrite("aninmul%", "aninbamu.prn", 0, 4, binv%, 1)

'comando que deshabilita las interrupciones del BACKGROUND

END SUB

SUB  ANINFOMU
'*************************************************************
'*  Esta es una Subrutina que me permite realizar adqui-  *
'*  sición de datos desde varios canales análogos, en el  *
'*  modo de trabajo de FOREGROUND.                        *
'*************************************************************

CLS
SCREEN 9
LINE (0, 0)-(600, 335), 11, B
LINE (3, 3)-(597, 332), 3, BF
COLOR 9
LOCATE 4, 14: PRINT "  Adquisición de datos en modo de trabajo FOREGROUND  "
LOCATE 5, 14: PRINT "                  Multicanal                  "
LOCATE 20, 16: PRINT "  Presione SPACE BAR para terminar la adquisición  "
LOCATE 8, 20: INPUT "  Número deseado de canales de entrada"; n
LOCATE 8, 20: PRINT "  Número deseado de canales de entrada ="; n
```

```
FOR chn% = 0 TO n - 1
      LOCATE 10 + chn%, 23: PRINT "  canal "; chn%; " voltaje = "
NEXT chn%
COLOR 7

CALL SOFTINIT
CALL INIT

'Estas dos instrucciones inicializan el SOFTWARE y el HARDWARE del sistema
'QUICK500 respectivamente.
SELECT CASE n
    CASE 1
    ionl$ = "ANLG0"
    CASE 2
    ionl$ = "ANLG0 ANLG1"
    CASE 3
    ionl$ = "ANLG0 ANLG1 ANLG2"
    CASE 4
    ionl$ = "ANLG0 ANLG1 ANLG2 ANLG3"
    CASE 5
    ionl$ = "ANLG0 ANLG1 ANLG2 ANLG3 ANLG4"
    CASE 6
    ionl$ = "ANLG0 ANLG1 ANLG2 ANLG3 ANLG4 ANLG5"
    CASE 7
    ionl$ = "ANLG0 ANLG1 ANLG2 ANLG3 ANLG4 ANLG5 ANLG6"
    CASE 8
    ionl$ = "ANLG0 ANLG1 ANLG2 ANLG3 ANLG4 ANLG5 ANLG6 ANLG7"
END SELECT
DIM va!(7)
'Definicion de parametros
COLOR 3
DO
      CALL READARRAY(ionl$, 1, -1, va!(), 0, "nt")
     'Lectura del dato adquirido por los canales análogos habilitados

      FOR chn% = 0 TO n - 1
            LOCATE 10 + chn%, 45: PRINT USING "##.## voltios  "; va!(chn%)
      NEXT chn%
      FOR i = 1 TO 4000
      NEXT i
LOOP UNTIL INKEY$ = " "
END SUB

SUB  ANINFORE
'*************************************************
'*  Esta es una Subrutina que me permite realizar adqui-  *
'*  sición de datos desde un canal análogo, en el modo -  *
'*  de trabajo de FOREGROUND.                             *
'*************************************************

CLS
SCREEN 9
LINE (0, 0)-(600, 335), 1, B
LINE (3, 3)-(597, 332), 1, BF
```

```
COLOR 2
LOCATE 10, 18: PRINT " Dato adquirido ="
LOCATE 11, 18: PRINT "                                    "
LOCATE 12, 18: PRINT " presione SPACE BAR para terminar la adquisición"
LOCATE 4, 18: PRINT "   Adquisición dc datos en modo de FOREGROUND   "
LOCATE 5, 18: PRINT "              un solo canal                "

CALL SOFTINIT
CALL INIT

'Estas dos instrucciones inicializan el SOFTWARE y el HARDWARE del sistema
'QUICK500 respectivamente.
ion$ = "ANLG0"
va! = 1
'Definicion de parametros
COLOR 9
WHILE INKEY$ <> CHR$(32)
      CALL READVAR(ion$, 1, -1, va!, 0, "nt")
     'Lectura del dato adquirido por el canal análogo habilitado
      LOCATE 10, 35: PRINT USING "##.#### voltios              "; va!
      FOR i = 1 TO 4000
      NEXT i
WEND
COLOR 7

END SUB

SUB ANLGIN
MENU2:
CLS
SCREEN 9
LINE (0, 0)-(600, 335), 14, B
LINE (3, 3)-(597, 332), 14, B
COLOR 14
LOCATE 3, 10: PRINT "       SOFTWARE DE ADQUISICION Y SALIDA DE DATOS"
LOCATE 4, 20: PRINT "    DEL SISTEMA KEITHLEY 500A"
LOCATE 8, 21: PRINT "ADQUISICION DE DATOS ANALOGOS:"
LOCATE 9, 19: PRINT "═══════════════════════════"
COLOR 15
LOCATE 13, 5: PRINT "UN SOLO CANAL MODO DE FOREGROUND                (1)"
LOCATE 14, 5: PRINT "MULTICANAL MODO DE FOREGROUND                   (2)"
LOCATE 15, 5: PRINT "UN SOLO CANAL MODO DE BACKGROUND                (3)"
LOCATE 16, 5: PRINT "MILTICANAL MODO DE BACKGROUND                   (4)"
LOCATE 17, 5: PRINT "SALIR AL MENU PRINCIPAL                         (5)"
pregunta2:
LOCATE 20, 5: INPUT "Escoja una opción por favor (1-5):"; OP$
IF OP$ = "1" THEN CALL ANINFORE: GOTO MENU2
IF OP$ = "2" THEN CALL ANINFOMU: GOTO MENU2
IF OP$ = "3" THEN CALL ANINBACK: GOTO MENU2
IF OP$ = "4" THEN CALL ANINBAMU: GOTO MENU2
IF OP$ = "5" THEN GOTO FIN2
IF OP$ > "5" OR OP$ = "" THEN GOTO pregunta2
FIN2:
END SUB
```

```
SUB ANOUBACK
'*************************************************************
'* Esta es una Subrutina que me permite realizar sali - *
'* da de datos desde un canal análogo, en el modo de - *
'* trabajo de BACKGROUND. *
'*************************************************************

CLS
CALL SOFTINIT
CALL INIT

'Estas dos instrucciones inicializan el SOFTWARE y el HARDWARE del sistema
'QUICK500 respectivamente.
ionl$ = "ANOUT0"
COLOR 2
LOCATE 5, 5: INPUT "Periodo de muestreo (ms)= "; bintv%
dep! = 100
am$ = "data%"
'En las instrucciones anteriores se setea nombre del canal, periodo de mues-
'treo y profundidad del arreglo receptor de los datos

CALL ARMAKE("data%", dep!, -1, ionl$)
'Este comando crea el arreglo en el cual se han de ubicar los datos previa
'su salida por el canal análogo escogido previamente
FOR dep! = 1 TO 100
    d! = 10 * (SIN(1 * (dep! - 1) * 6.28 / 100))
    CALL arputvalf("data%", dep!, -1, "ANOUT0", d!, 0)

    'Comando que va colocando los valores calculados en el lugar del
    'área correspondiente
NEXT dep!

CALL ANOUT("data%", "ANOUT0", bintv%, -1, "nt", "tarea2")

'Comando que setea la terea de BACKGROUND de salida de datos por un canal
'análogo, una vez que se hayan habilitado las interrupciones
COLOR 14
LOCATE 20, 16: PRINT "Salida de datos de una sinusoide de "; 1 / bintv%; "Hz."
LOCATE 22, 11: PRINT "Presione la barra espaciadora para terminar la salida"
COLOR 2
CALL INTON(1, "mil")

'Comando que habilita las interrupciones del BACKGROUND.
an$ = ""
WHILE an$ = ""
    an$ = INKEY$
WEND

CALL INTOFF

'Comando que deshabilita las interrupciones
CALL arwrite("data%", "onda13.prn", 0, 4, bintv%, 1)
CALL arsave("data%", "onda13.dat")
END SUB
```

```
SUB ANOUBAMU
'*************************************************************
'* Esta es una Subrutina que me permite realizar sali -  *
'* da de datos desde varios canales análogos, en el mo-  *
'* do de trabajo de BACKGROUND.                          *
'*************************************************************
CLS
CALL SOFTINIT
CALL INIT

'Estas dos instrucciones inicializan el SOFTWARE y el HARDWARE del sistema
'QUICK500 respectivamente.
ionl$ = "ANOUT0 ANOUT1 ANOUT2 ANOUT3 ANOUT4"
COLOR 3
LOCATE 6, 6: INPUT "Periodo de muestreo (ms)= "; bintv%
dep! = 100
arn$ = "datosen%"

'En las instrucciones anteriores se setea nombre de los canales, período de
'muestreo y profundidad del arreglo receptor de los datos
CALL ARMAKE("datosen%", dep!, -1, ionl$)

'Este comando crea el arreglo en el cual se han de ubicar los datos previa
'su salida por los canales análogos escogidos previamente
FOR dep! = 1 TO 100
    d1% = INT(.5 * (SIN((dep! - 1) * 6.28 / 100) * 2047.5 + 4095))
    d2% = INT(.5 * (COS((dep! - 1) * 6.28 / 100) * 2047.5 + 4095))
    CALL arputvali("datosen%", dep!, 1, ionl$, d1%)
    CALL arputvali("datosen%", dep!, 2, ionl$, d2%)

    'Comandos que van colocando los valores calculados en el lugar del
    'area correspondiente
NEXT dep!
FOR dep! = 1 TO 100
    va1% = INT(.5 * ((2047.5 * dep! / 100) + 4095))
    CALL arputvali("datosen%", dep!, 3, ionl$, va1%)
NEXT
va4% = 4095: va5% = 0
FOR dep! = 1 TO 50
    va2% = INT(.5 * ((2047.5 * dep! / 50) + 4095))
    CALL arputvali("datosen%", dep!, 4, ionl$, va2%)
    CALL arputvali("datosen%", dep! + 50, -1, "ANOUT4", 0)
    va3% = INT(.5 * ((2047.5 * (50 - dep!) / 50) + 4095))
    CALL arputvali("datosen%", dep! + 50, 4, ionl$, va3%)
    CALL arputvali("datosen%", dep!, -1, "ANOUT4", 4095)
NEXT

CALL ANOUT("datosen%", ionl$, bintv%, -1, "nt", "tarea2")
'Comando que se encarga de sacar los valores guardados en el area, una vez
'que se hayan habilitado las interrupciones
COLOR 2
LOCATE 12, 20: PRINT "Salida de datos de ondas de "; 10 / bintv%; "Hz."
LOCATE 13, 11: PRINT "a través de dos canales análogos con amplitudes diferentes"
COLOR 3
```

```
LOCATE 25, 15: PRINT "Presione SPACE BAR para terminar la salida de datos"
CALL INTON(1, "mil")

'Comando que habilita las interrupciones del BACKGROUND.
an$ = ""
WHILE an$ = ""
      an$ = INKEY$
WEND
CALL INTOFF
'Comando que deshabilita las interrupciones

CALL arwrite("datosen%", "salida.prn", 0, 4, bintv%, 1)
END SUB

SUB ANOUFOMU
'*************************************************************
'*  Esta es una Subrutina que me permite realizar sali -  *
'*  da de datos a través de varios canales análogos, en-  *
'*  el modo de trabajo de FOREGROUND.                     *
'*************************************************************

CLS
SCREEN 9
LINE (0, 0)-(600, 235), 11, B
LINE (3, 3)-(597, 232), 11, B
LINE (0, 255)-(600, 335), 11, B
LINE (3, 258)-(597, 332), 11, B
LOCATE 8, 3: PRINT "Canal de Salida :          1                    2"
LOCATE 9, 3: PRINT "-------------------------------------------------------------"
LOCATE 22, 16: PRINT "presione SPACE BAR para terminar salida de datos"

CALL SOFTINIT
CALL INIT

'Estas dos instrucciones inicializan el SOFTWARE y el HARDWARE del sistema
'QUICK500 respectivamente.
ionl$ = "ANOUT0 ANOUT1"
DIM va!(2)
va!(1) = 0
va!(0) = 0
'Definicion de parametros
COLOR 0
WHILE INKEY$ <> CHR$(32): v = v + 1
      va!(0) = va!(0) + 1: va!(1) = 11 - va!(0)
      IF va!(0) > 10 THEN va!(0) = 0
      IF v > 15 THEN v = 1
      LOCATE 10, 23 + va!(0) * 2: PRINT va!(0);
      LOCATE 10, 52 + va!(0) * 2: PRINT va!(1);

      CALL WRITEARRAY(ionl$, 2, va!(), 0, "nt")
      FOR i = 1 TO 100: NEXT

     'Escritura del dato calculado en los canales análogos de salida habilitados
      COLOR v
```

```
WEND

END SUB

SUB ANOUFORE
'*************************************************************
'* Esta es una Subrutina que me permite realizar sali - *
'* da de datos a través de un canal análogo, en el mo - *
'* do de trabajo de FOREGROUND.                          *
'*************************************************************
CLS
SCREEN 9
LINE (0, 0)-(600, 235), 11, B
LINE (3, 3)-(597, 232), 11, B
LINE (0, 255)-(600, 335), 11, B
LINE (3, 258)-(597, 332), 11, B
COLOR 11
LOCATE 8, 16: PRINT "Valor de Salida ="
LOCATE 22, 16: PRINT "presione SPACE BAR para terminar salida de datos"

CALL SOFTINIT
CALL INIT

'Estas dos instrucciones inicializan el SOFTWARE y el HARDWARE del sistema
'QUICK500 respectivamente.
ion$ = "ANOUT0"
va! = 0
'Definicion de parametros
COLOR 0
WHILE INKEY$ <> CHR$(32): v = v + 1
      va! = va! + 1
      IF va! > 10 THEN va! = 0
      IF v > 15 THEN v = 1
      LOCATE 8, 35 + va! * 2: PRINT va!;
      CALL WRITEVAR(ion$, 2, va!, 0, "nt")

     'Escritura del dato calculado en el canal análogo de salida habilitado
      COLOR v
WEND
END SUB

SUB DIGITOUT
MENU4:
CLS
SCREEN 9
LINE (0, 0)-(600, 335), 11, B
LINE (3, 3)-(597, 332), 11, B
COLOR 11
LOCATE 3, 10: PRINT "      SOFTWARE DE ADQUISICION Y SALIDA DE DATOS"
LOCATE 4, 20: PRINT "    DEL SISTEMA KEITHLEY 500A"
LOCATE 8, 25: PRINT "SALIDA DE DATOS DIGITALES:"
LOCATE 8, 23: PRINT "═══════════════════════"
COLOR 15
LOCATE 13, 5: PRINT "UN SOLO PUERTO MODO DE FOREGROUND                (1)"
```

```
LOCATE 14, 5: PRINT "MULTIPUERTO MODO DE FOREGROUND                (2)"
LOCATE 15, 5: PRINT "UN SOLO PUERTO MODO DE BACKGROUND             (3)"
LOCATE 16, 5: PRINT "MULTIPUERTO MODO DE BACKGROUND                (4)"
LOCATE 17, 5: PRINT "SALIR AL MENU PRINCIPAL                       (5)"
pregunta4:
LOCATE 22, 5: INPUT "Escoja una opción por favor:"; OP$

IF OP$ = "1" THEN CALL DIOUFORE: GOTO MENU4
IF OP$ = "2" THEN CALL DIOUFOMU: GOTO MENU4
IF OP$ = "3" THEN CALL DIOUBACK: GOTO MENU4
IF OP$ = "4" THEN CALL DIOUBAMU: GOTO MENU4
IF OP$ = "5" THEN GOTO FIN4
IF OP$ > "5" OR OP$ = "" THEN GOTO pregunta4
FIN4:
END SUB

SUB DIOUBACK
'*************************************************************
'* Esta es una Subrutina que me permite realizar sali - *
'* da de datos desde un puerto digital, en el modo de - *
'* trabajo de BACKGROUND.                              *
'*************************************************************
CLS
LOCATE 1, 3: PRINT "Inicio de la Salida de Datos"

CALL SOFTINIT
CALL INIT

'Estas dos instrucciones inicializan el SOFTWARE y el HARDWARE del sistema
'QUICK500 respectivamente.
CALL IONMDIG("PORT1", 5, -1, 1)

ionl$ = "PORT1"
LOCATE 3, 3: INPUT "Período de muestreo (dms seg)= "; bintv%
dep! = 15
'En las instrucciones anteriores se setea nombre de los puertos, período de
'muestreo y profundidad del arreglo receptor de los datos
arn$ = "datoarea%"

CALL ARMAKE("datoarea%", dep!, -1, ionl$)

'Este comando crea el arreglo en el cual se han de ubicar los datos previa
'su salida por los puertos digitales escogidos previamente
otro:
FOR dep! = 1 TO 7
    COLOR dep!
    d% = 2 ^ dep!
    LOCATE 6 + dep!, 10 + 3 * dep!: PRINT d%
    CALL arputvali("datoarea%", dep!, -1, "PORT1", d%)

    'Comando que va colocando los valores calculados en el lugar del
    'área correspondiente
NEXT dep!
FOR dep! = 9 TO 15
```

```
        COLOR dep!
        d% = 2 ^ (16 - dep!)
        LOCATE 6 + 16 - dep!, 10 + 3 * dep!: PRINT d%
        CALL arputvali("datoarea%", dep!, -1, "PORT1", d%)

        'Comando que va colocando los valores calculados en el lugar del
        'area correspondiente
NEXT dep!

CALL DIGOUT("datoarea%", ionl$, bintv%, -1, "nt", "tarea1")

'Comando que setea la tarea de BACKGROUND, realiza el muestreo de los puertos
'de salida digital
LOCATE 5, 3: PRINT "Salida de datos."
COLOR 4
LOCATE 23, 15: PRINT "Presione SPACE BAR para terminar"

CALL INTON(100, "mil")

'Comando que habilita las interrupciones del BACKGROUND.
an$ = ""
WHILE an$ = ""
        an$ = INKEY$
WEND

CALL INTOFF
'Comando que deshabilita las interrupciones

END SUB

SUB DIOUBAMU
'*************************************************************
'* Esta es una Subrutina que me permite realizar sali -  *
'* da de datos desde varios puertos digitales, en modo-  *
'* de trabajo de BACKGROUND.                             *
'*************************************************************
CLS
LOCATE 1, 3: PRINT "Inicio de la Salida de Datos"

CALL SOFTINIT
CALL INIT
'Estas dos instrucciones inicializan el SOFTWARE y el HARDWARE del sistema
'QUICK500 respectivamente.

CALL IONMDIG("PORT0", 5, -1, 1)
CALL IONMDIG("PORT1", 5, -1, 2)

ionl$ = "PORT0 PORT1"
LOCATE 3, 3: INPUT "Periodo de muestreo (dcs seg)= "; bintv%
dep! = 15
'En las instrucciones anteriores se setea nombre de los puertos, periodo de
'muestreo y profundidad del arreglo receptor de los datos
arn$ = "datoarea%"
```

```
'En las instrucciones anteriores se setea nombre de los puertoS, periodo de
'muestreo y profundidad del arreglo receptor de los datos
CALL ARMAKE("datoarea%", dep!, 2, ion!$)

'Este comando crea el arreglo en el cual se han de ubicar los datos previa
'su salida por los puertos digitales escogidos previamente
otros:
FOR dep! = 1 TO 8
    d% = 2 ^ dep! - 1
    LOCATE 6 + dep!, 10 + 3 * dep!: PRINT d%
    CALL arputvali("datoarea%", dep!, 1, "", d%)

    'Comando que va colocando los valores calculados en el lugar del
    'area correspondiente
NEXT dep!
FOR dep! = 9 TO 15
    d% = 2 ^ (16 - dep!) - 1
    LOCATE 6 + 16 - dep!, 10 + 3 * dep!: PRINT d%
    CALL arputvali("datoarea%", dep!, 1, "", d%)

    'Comando que va colocando los valores calculados en el lugar del
    'area correspondiente
NEXT dep!

FOR dep! = 1 TO 8
    d1% = 2 ^ (9 - dep!) - 1
    LOCATE 15 + dep!, 10 + 3 * dep!: PRINT d1%
    CALL arputvali("datoarea%", dep!, 2, "", d1%)
'Comando que va colocando los valores calculados en el lugar del
'area correspondiente

NEXT dep!
FOR dep! = 9 TO 15
    d1% = 2 ^ (dep! - 7) - 1
    LOCATE 15 + 16 - dep!, 10 + 3 * dep!: PRINT d1%
    CALL arputvali("datoarea%", dep!, 2, "", d1%)

    'Comando que va colocando los valores calculados en el lugar del
    'area correspondiente
NEXT dep!

CALL DIGOUT("datoarea%", ion!$, bintv%, -1, "nt", "tarea1")
'Comando que setea la tarea de BACKGROUND, realiza el muestreo de los puertos
'de salida digital
LOCATE 5, 3: PRINT "Salida de datos."
COLOR 4
LOCATE 25, 15: PRINT "Presione SPACE BAR para terminar"

CALL INTON(100, "mil")

'Comando que habilita las interrupciones del BAKGROUND.
an$ = ""
WHILE an$ = ""
    an$ = INKEY$
```

```
WEND

CALL INTOFF
'Comando que deshabilita las interrupciones

END SUB

SUB DIOUFOMU
'*************************************************************
'* Esta es una Subrutina que me permite realizar sali - *
'* da de datos a través de varios puertos digitales, en *
'* el modo de trabajo de FOREGROUND. *
'*************************************************************
CLS
COLOR 14
LOCATE 2, 15: PRINT "Salida de datos digitales multipuerto en foreground"
LOCATE 7, 10: PRINT "Puerto 0 Valor de Salida Puerto 1 "
LOCATE 20, 16: PRINT "presione SPACE BAR para terminar salida de datos"

CALL SOFTINIT
CALL INIT

'Estas dos instrucciones inicializan el SOFTWARE y el HARDWARE del sistema
'QUICK500 respectivamente.
CALL IONMDIG("PORT0", 5, -1, 1)
CALL IONMDIG("PORT1", 5, -1, 2)

ion!$ = "PORT0 PORT1"
DIM va!(2)
va!(0) = 1
va!(1) = 128

'Definición de parámetros
WHILE INKEY$ <> CHR$(32)
    va!(0) = va!(0) * 2: va!(1) = va!(1) / 2
    IF va!(0) > 128 THEN va!(0) = 1
    IF va!(1) < 1 THEN va!(1) = 128
    IF v > 8 THEN v = 0
    COLOR 4
    LOCATE 10, 43 + 4 * v: PRINT va!(0); SPACE$(2)
    COLOR 3
    LOCATE 10, 37 - 4 * v: PRINT va!(1)
    v = v + 1
    CALL WRITEARRAY(ion!$, 4, va!(), -1, "nt")
    'Escritura del dato calculado por el puerto digital habilitado

    FOR i = 1 TO 2000: NEXT i
WEND

END SUB

SUB DIOUFOKE
'*************************************************************
'* Esta es una Subrutina que me permite realizar sali - *
```

```
'*  da de datos a través de un puerto digital, en el mo-  *
'*  do de trabajo de FOREGROUND.                          *
'*********************************************************
CLS
COLOR 14
LOCATE 3, 12: PRINT "Salida digital de datos unipuerto en foreground"
COLOR 3
LOCATE 6, 10: PRINT "Valor de Salida ="
LOCATE 20, 12: PRINT "presione SPACE BAR para terminar salida de datos"

CALL SOFTINIT
CALL INIT

'Estas dos instrucciones inicializan el SOFTWARE y el HARDWARE del sistema
'QUICK500 respectivamente.

CALL IONMDIG("PORT1", 5, -1, 1)
ion$ = "PORT1"
va! = 1
'Definicion de parámetros

WHILE INKEY$ <> CHR$(32)
      va! = va! * 2
      IF va! > 128 THEN va! = 1
      IF v > 8 THEN v = 0
       COLOR 4
       COLOR 3
      LOCATE 6, 28 + 4 * v: PRINT va!; SPACE$(2)
      v = v + 1
        CALL WRITEVAR(ion$, 4, va!, -1, "nt")

     'Escritura del dato calculado por el puerto digital habilitado
      FOR i = 1 TO 2000: NEXT i
WEND

END SUB
```

PROGRAMA PARA GENERAR ONDAS PARA ANALISIS POSTERIOR

```
CLS
CALL SOFTINIT
CALL INIT
inicio:
CLS
'Estas dos instrucciones inicializan el SOFTWARE y el HARDWARE del sistema
'QUICK500 respectivamente.
COLOR 15
LOCATE 2, 10: PRINT "SISTEMA DE ADQUISICION DE DATOS Y CONTROL KEITHLEY 500A"
LOCATE 4, 10: PRINT "          ONDAS SENOIDALES COMPUESTAS"
ion1$ = "ANOUT0"
COLOR 2
LOCATE 5, 5: INPUT "Periodo de muestreo (ms)= "; bintv%
LOCATE 7, 5: INPUT "Número de muestras por período"; dep!
```

```
LOCATE 9, 5: PRINT "La base de frecuencia (f# . base)="; 1000 / bintv% / dep!; "(Hz.)"

arn$ = "data%"
'En las instrucciones anteriores se setea nombre del canal, periodo de mues-
'treo y profundidad del arreglo receptor de los datos

CALL ARMAKE("data%", dep!, -1, ionl$)

'Este comando crea el arreglo en el cual se han de ubicar los datos previa
'su salida por el canal análogo escogido previamente
LOCATE 11, 5: PRINT "En la onda del tipo: a*sen(f1.base.t) + b*sen(f2.base.t) + c*rnd + vm"
LOCATE 12, 5: PRINT "Introduzca los parámetros:"
LOCATE 14, 32: PRINT "Amplitid"; : COLOR 14: PRINT " a"; : COLOR 2: INPUT " de la primera componente      "; a
LOCATE 15, 32: PRINT "Amplitid"; : COLOR 14: PRINT " b"; : COLOR 2: INPUT " de la segunda componente      "; b
LOCATE 16, 32: PRINT "Frecuencia"; : COLOR 14: PRINT " f1"; : COLOR 2: INPUT " de la primera componente   "; f1
LOCATE 17, 32: PRINT "Frecuencia"; : COLOR 14: PRINT " f2"; : COLOR 2: INPUT " de la segunda componente   "; f2
LOCATE 18, 32: PRINT "Amplitid"; : COLOR 14: PRINT " c"; : COLOR 2: INPUT " de la componente de ruido     "; c
LOCATE 19, 32: PRINT "Valor madio"; : COLOR 14: PRINT " vm"; : COLOR 2: INPUT " de la onda "; vm

FOR depor! = 1 TO dep!
    d! = (a * SIN(f1 * (depor! - 1) * 6.28 / dep!) + b * SIN(f2 * (depor! - 1) * 6.28 / dep!)) + c * RND(depor!) - c / 2 + vm
    CALL arputvalf("data%", depor!, -1, "ANOUT0", d!, 0)

    'Comando que va colocando los valores calculados en el lugar del
    'área correspondiente
NEXT depor!

CALL ANOUT("data%", "ANOUT0", bintv%, -1, "nt", "tarea2")
CALL anin("entrada%", 400!, "ANLG0", bintv%, 1, "nt", "")
'Comando que setea la terea de BACKGROUND de salida de datos por un canal
'análogo, una vez que se hayan habilitado las interrupciones
COLOR 14
LOCATE 23, 11: PRINT "Presione la barra espaciadora para terminar la salida"
COLOR 2
CALL INTON(1, "mil")

'Comando que habilita las interrupciones del BACKGROUND.
an$ = ""
WHILE an$ = ""
    an$ = INKEY$
WEND

CALL INTOFF

'Comando que deshabilita las interrupciones
LOCATE 21, 11: INPUT "Desea almacenar la onda en disco (s/n)"; s$
IF s$ = "s" OR s$ = "S" THEN
```

```
CALL arwrite("entrada%", nombre$, 0, 4, bintv%, 1)
CALL arsave("entrada%", nom$)
END IF
desea:
LOCATE 23, 11: INPUT "Desea formar otra onda (s/n)                    "; s$
IF s$ = "s" OR s$ = "S" THEN
CALL backclear
CALL ardel("entrada%")
CALL ardel("data%")
GOTO inicio
ELSEIF s$ = "n" OR s$ = "N" THEN
END
ELSE
GOTO desea
END IF
```