UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

UN CONTROLADOR PD AUTOSINTONIZABLE: ESTUDIO E IMPLEMENTACION

T E S I S

QUE PARA OBTENER EL TITULO DE:

INGENIERO MECANICO ELECTRICISTA

P R E S E N T A N :

GABRIEL ESCOBAR ONTIVEROS
GERARDO ESCOBAR VALDERRAMA

Directores de Tesis:
DR. TANG YU M. EN C. GERARDO ESPINOSA

MEXICO, D. F. 1991

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

CONTENIDO

Capítulo 0

Introducción

En la teoria de control se han reportado diversas estructuras para controladores las cuales requieren para su sintonización, es decir, para asignar valores a sus parámetros, de un buen conocimiento del modelo del proceso a controlar.

Un buen conocimiento de la planta o proceso implica conocer tanto la estructura de su modelo como los valores que tomarán sus parámetros. Este modelo podria ser obtenido por métodos experimentales, sin embargo, no hay que perder de vista que muchos de los procesos son variantes con el tiempo, por lo que el modelo obtenido no será válido para todos los casos. También se podria pensar en sintonizar al controlador en forma manual, lo cual se ha venido haciendo por años con los controladores PID, esta forma de sintonizar por supuesto tiene grandes desventajas. En principio, consume una gran cantidad de tiempo y más aun, una buena sintonización del controlador depende en mucho de la experiencia del operador.

Una solución al problema de conocer el modelo del proceso consiste en aplicar algoritmos de identificación, con los cuales es posible conocer la estructura y/o los parámetros del modelo del proceso. De esta forma, a un algoritmo de esta clase lo podemos ver como un bloque al que se alimentan señales provenientes de la planta, y en su salida nos entrega una estimación sobre la estructura y/o parámetros del modelo de dicha planta.

Al añadir un algoritmo de identificación a una estructura de control dada, le estaremos dando a todo nuestro algoritmo de control la caracteristica de autosintonizable. Asi, con base en la información acerca de la planta obtenida del proceso de identificación, se procede a calcular los parámetros del controlador (sintonizarlo) tales que la respuesta del sistema en lazo cerrado cumpla con ciertas caracteristicas.

Esta clase de algoritmos no serían factibles de implementarse a no ser por el rápido y revolucionario progreso de la microelectrónica y las técnicas digitales, las cuales han hecho posible la implementación de reguladores o controladores baratos de tipo digital y que pueden tomar estructuras complejas.

En el progreso de la microelectrónica y básicamente de los microprocesadores cabe destacar en especial una rama que también ha evolucionado vigorosamente: los microcontroladores. Estos dispositivos son aquellos microprocesadores que han sido combinados con RAM, ROM además de varias facilidades de entrada y salida como puertos serie y/o paralelo y algunas veces convertidores A/D y/o D/A sobre un mismo chip.

El enfoque en los microcontroladores no ha sido hacia el manejo de grandes longitudes de palabra ni de grandes espacios de memoria como en el resto de los microprocesadores, su énfasis, más bién, se ha dirigido hacia la integración de las facilidades necesarias dentro de un solo chip para hacer posible el control rápido en tiempo real.

En esta tesis estamos interesados en mostrar la aplicación de una técnica adaptable para la autosintonización de un controlador usando el microcontrolador MC68HC11 fabricado por Motorola. Específicamente, se trata del uso de un algoritmo de identificación para estimar los parámetros ganancia estática y polo del modelo de primer orden de un motor de corriente directa (C.D). y con base en esta información, calcular los valores que deberán tomar los parámetros de un controlador PD tales que el sistema en lazo cerrado responda de alguna forma especificada.

La tesis está organizada de la siguiente manera: En el capítulo I se da una breve descripción del microcontrolador MCU 68HC11. En el capítulo II se describe su tarjeta evaluadora 68HC11EVB también fabricada por Motorola. En el capítulo III se describen y discuten cada una de las partes de que se conforma el algoritmo de control propuesto, para esto se ha dividido al algoritmo en tres etapas fundamentales, a saber: identificación, sintonización y control. El capítulo IV trata la implementación del algoritmo usando el microcontrolador MC68HC11 para llevar a cabo el control de posición del servomecanismo de C.D.. En el capítulo V se exponen las pruebas realizadas y se discuten los resultados obtenidos. Por último, en el capítulo VI, se expresan nuestras conclusiones acerca de las experiencias logradas en este proyecto.

Capítulo 1

Microcontrolador MC68HC11

I.1 INTRODUCCION.

En la actualidad se cuenta con microprocesadores que han sido combinados en un solo chip con memorias RAM, ROM, puertos serie, puertos paralelo, convertidores A/D y/o D/A y a los cuales se les ha dado el nombre de *microcontroladores*. Estos dispositivos nos permiten hacer un control rápido en tiempo real gracias a su estructura compacta, ofreciendo beneficios en costo, tamaño, funcionalidad y facilidad de manejo. Es por ello que se determino elegir un microcontrolador con las ventajas antes mencionadas como lo es el MCU 68HC11.

En el presente capitulo será descrito dicho microcontrolador, presentandose de una manera breve su arquitectura y caracteristicas principales. La información obtenida para este microcontrolador está basada en sus manuales, si se desea profundizar en alguna sección, esta podra ser consultada en la referencia [7].

El microcontrolador HCMOS MC68HC11 de Motorola se destaca por ser un avanzado microcontrolador de 8 bits con capacidades periféricas sofisticadas. Puede alcanzar una velocidad de bus de 2 MHz, su consumo de potencia es bajo y su inmunidad al ruido es alta.

Cuenta con circuiteria de protección contra posibles errores del sistema. Un sistema monitor de reloj re-iniciara al sistema en caso de que el reloj se pierda ó baje su velocidad. Además cuando es detectado un código de operación ilegal se provee una interrupción no mascarable. A continuación son presentadas sus caracteristicas principales.

I.2 CARACTERISTICAS.

A. Hardware.

- 256 bytes de RAM, expandible a 4 K.
- 512 bytes de EEPROM.
- 8 K bytes de ROM.
- Sistema de reloj de 16 bits de carrera libre con: Preescalador programable, 3 funciones de captura de entradas y 5 funciones de salidas de comparación.
- Interfaz de comunicación serie (SCI).
- Interfaz periférica serie (SPI).
- 8 canales de conversión A/D de 8 bits cada uno.
- Circuitería para interrupciones en tiempo real.
- Circuitería para acumulador de pulsos de 8 bits.
- Sistema *WATCHDOG* para fallas de software.

B. Software

- Conjunto de instrucciones mejorado (en relación al M6800/M6801).
- División entera y fraccionaria de 16 x 16.
- Bit de manipulación.
- Modo de espera.
- Modo de paro.

Su correspondiente diagrama a bloques es mostrado en la fig. 1.1 .

I.3 SEÑALES Y MODOS DE OPERACION.

En esta sección se describe la función de las terminales del microcontrolador MC68HC11, así como sus distintos modos de operación.

A. Señales.

El MC68HC11 es disponible en dos presentaciones (de 48 y 52 terminales). En la figura 1.2 se muestra el paquete que fué empleado.

1. Fuentes de poder Vdd y Vss.

La fuente de suministro V_{DD} es una fuente positiva entre -0.34 y +7v, mientras la terminal de la fuente Vss debe encontrarse aterrizada a 0v. (Se recomienda un capacitor de 0.1 µF entre V_{DD} y Vss).

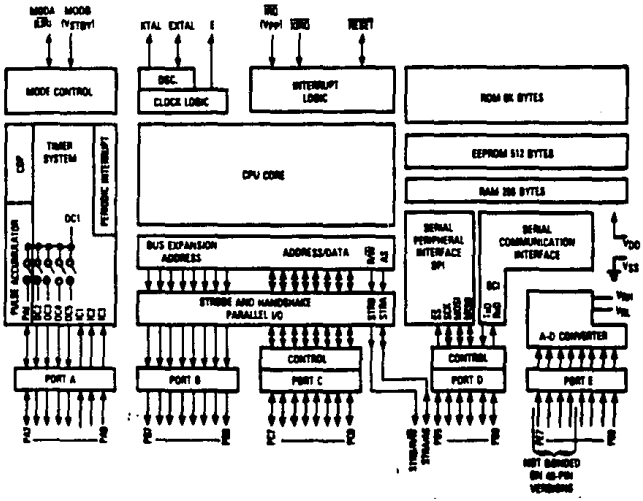


Fig. 1.1 Diagrama a bloques del MCU 68HC11.

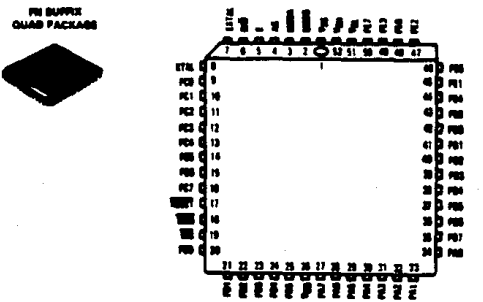


Fig. 1.2 Asignación de terminales en el MCU 68HC11.

2. Reset.

La terminal de reset se activa baja y se emplea para inicializar al microcontrolador (La señal de control es bidireccional).

3. Entradas de reloj externa y manejador de cristal (EXTAL,XTAL).

Estas dos terminales son la interfaz para el cristal u oscilador externo, el cual controla la circuitería interna del generador de reloj en el microcontrolador. Para mayor información acerca de la circuitería recomendada para el cristal u oscilador ver la referencia [7].

4. Salida de Reloj (E).

Esta terminal está conectada al reloj E (generado internamente) el cual puede ser usado como referencia de tiempo. La frecuencia de este reloj es la cuarta parte de la frecuencia del cristal u oscilador.

5. Interrupción pedida (IRQ).

Esta terminal de entrada provee una interrupción asincrona al 68HC11. La interrupción es programada en el registro OPTION (la terminal requiere una resistencia de "pull up" a V_{DD} de 4.7 k ohms).

6. Interrupción no enmascarada (XIRQ).

Esta terminal de entrada provee una interrupción no enmascarada es decir el microcontrolador decide cuando hacer la interrupción (También, requiere una resistencia de "pull up").

B. Modos de operación.

Para seleccionar el modo de operación es necesario colocar niveles lógicos en las terminales MODA y MODB. A continuación se describen los modos de operación que permiten conformar al microcontrolador en una amplia variedad de aplicaciones:

1. Modo simple.

En este modo, el microcontrolador trabaja sin registros externos o bus de datos. Los puertos B y C, así como las líneas STROBE A y B son de propósito general y funcionan como señales de E/S y *HANDSHAKE*.

2. Modo expandido multiplexado.

Este modo fué empleado en el trabajo presente ya que el microcontrolador tiene la capacidad de accesar 64 K de memoria externa, para lo cual se hace una expansión en el bus de direcciones por medio de los puertos B y C junto con las señales de control AS y $R/\overline{W}$.

3. Modo especial "Bootstrap".

Este modo es vesatil ya que no existen limitaciones para un programa en especial que vaya a ser cargado en la RAM interna. El usuario debe cargar 256 bytes del programa de datos a partir de la localidad $0000 hexadecimal de la RAM. Cuando es acabado de cargar se comenzará a ejecutar el código inmediatamente.

4. Modo de prueba.

Este modo involucra varias funciones de prueba, por lo que no es recomendable ya que reduce la seguridad del sistema. Esto se debe a que no existe protección para algunos registros, así que éstos pueden reescribirse continuamente.

I.4 REGISTROS DEL CPU, MODOS DE DIRECCIONAMIENTO Y CONJUNTO DE INSTRUCCIONES.

A. Registros del CPU.

En los siguientes parrafos seran descritos los 7 registros del microcontrolador que están disponibles para ser programados tal como lo muestra la figura.

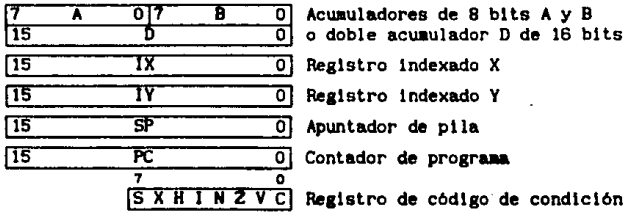


Fig. 1.3 Registros de programación.

1. Acumuladores A y B.

Estos acumuladores son de proposito general y de 8 bits, siendo su función principal el retener resultados de cálculos aritméticos, operandos ó la manipulación de datos. Al ser concatenados ambos acumuladores formaran el registro D.

2. Registros Indexados X(IX) Y(IY).

Estos registros son de 16 bits y son utilizados para el modo de direccionamiento indexado, pudiendo ser empleados como contadores o registros de almacenamiento temporal. El registro IY utiliza un byte y un ciclo extra para su ejecución.

3. Apuntador de Pila (SP).

Este registro es de 16 bits y permite importar datos que son almacenados durante interrupciones ó llamados de subrutinas ya que contiene la dirección de la próxima localidad. La pila es configurada en la sequencia "El último en entrar (Push) es el primero en salir (Pull)".

4. Contador del Programa (PC).

Este registro de 16 bits contiene la dirección de la próxima instrucción a ejecutar.

5. Registro de Código de Condición (CCR).

Este registro de 8 bits tiene en cada uno de sus bits el resultado de la última instrucción ejecutada. A continuación se explican cada uno de sus bits:

- Carry/Borrow (C).- Es 1 si existió acarreo durante la última operación.
- Overflow (V).- Es 1 si existió sobreflujo aritmético en la última operación.
- Zero (Z).-Es 1 si la última operación aritmética, lógica ó manipulación de datos fué cero.
- Negative (N).- Es 1 si la última operación aritmética, lógica ó manipulación de datos fué negativa.
- Máscara de Interrupción (I).- El bit I es fijado por hardware o instrucción de programa desabilitando todas las fuentes de interrupción mascarables.
- Medio Carry (H).- Es 1 cuando existe acarreo entre los bits 3 y 4 del ALU para una instrucción ADD, ABA, ADC.
- Máscara de interrupción (X).-El bit X se fija por hardware ($\overline{\text{reset}}$ o $\overline{\text{XIRQ}}$) y es limpiada por programa (TAP ó RTI).
- Desabilitación de paro (S).-El bit S es fijado cuando la instrucción de paro fué desabilitada.

B. Modos de direccionamiento.

A continuación se describen 6 modos de direccionamiento las cuales son usados para referencia de memoria; estos son:

1. Direccionamiento Inmediato.

El dato actual está contenido en el byte inmediatamente siguiente a la instrucción .

2. Direccionamiento Directo.

Este direccionamiento permite el acceso de memoria de $0000 a $00FF. Estos 256 bytes de área estan disponibles para datos.

3. Direccionamiento Extendido.

Este direccionamiento contiene la dirección absoluta del operando en 16 bits.

4. Direccionamiento Indexado.

Este direccionamiento utiliza los registros indexados (X ó Y) para calcular la dirección efectiva. La cual es variable y depende tanto del contenido de los registros indexados, como de la cantidad de offset contenida en la instrucción.

5. Direccionamiento Inherente.

Toda la información está contenida en el código de operación.

6. Direccionamiento Relativo.

Este direccionamiento es usado para instrucciones enramadas en donde el byte signado seguido del código de operación es sumado al contenido del contador de programa.

C. Conjunto de instrucciones.

El CPU del MC68HC11 cuenta con 91 códigos de operación, un segundo registro indexado IY de 16 bits, instrucciones de espera y paro, dos intrucciones de división de 16 x 16 y con un bit de manipulación de instrucciones.

La tabla 10-1 de la referencia [7] contiene todas las posibles instrucciones en todos los modos de direccionamiento. Mostrando para cada instrucción el operando, número de bytes en código de máquina y el tiempo de ejecución.

Las tablas 10-2 a 10-8 de la referencia [7] proveen la información del bus de direcciones , del bus de datos y de las lineas de lectura y escritura durante cada ciclo ó instrucción. La información está

categorizada en grupos de acuerdo al modo de direccionamiento y número de ciclos por instrucción, indicando las excepciones.

I.5 MEMORIAS.

En esta sección se describen las memorias RAM, ROM y EEPROM; así como su espacio respectivo en el mapa de memoria.

A. Mapa de memoria.

La composición del mapa de memoria depende del modo de operación elegido; para este trabajo se ha empleado el modo de operación expandido por lo que su correspondiente mapa de memoria es mostrado en la fig. 1.4.

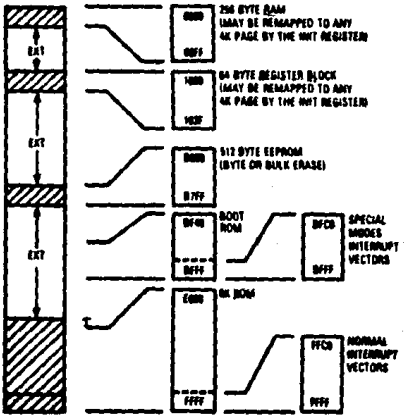


Fig. 1.4 Mapa de memoria del MCU 68HC11.

Las áreas del mapa de memoria designadas como "EXT" se encuentran disponibles para memoria externa y/o algún dispositivo de E/S. Las localidades $1000 a $103F son ocupadas por los 64 registros y bits de control.

B. RAM.

Esta memoria permite almacenar o leer datos que son manipulados a lo largo del programa. Los 256 bytes de RAM, así como los registros de control pueden ser recolocados durante la inicialización al ser configurado el registro INIT, junto con el bit ROMON en cero del registro CONFIG (La memoria ROM es desabilitada y su espacio de memoria es ocupado por la RAM).

C. ROM.

Esta memoria sostiene las instrucciones de un programa de aplicación. Las instrucciones son programadas en el microcontrolador cuando éste es manufacturado, por lo que dichas instrucciones no podran ser modificadas. Los 8 K de ROM pueden desabilitarse limpiando el bit ROMON del registro CONFIG.

D. EEPROM.

Esta memoria puede almacenar la información aún sin suministro de energia; además de que es posible borrar o reprogramar dicha información.

El mecanismo de escritura o programación de la EEPROM es controlado por el registro PPROG. Para el caso donde se desee desabilitar dicha memoria bastara con limpiar el bit EEON del registro CONFIG.

I.8 CONVERSION A/D.

En ésta sección se describen las caracteristicas del convertidor A/D y los distintos modos en que éste puede operar.

El MC68HC11A8 incluye 8 canales de entrada multiplexada y de aproximaciones sucesivas para los cuales son minimizados los errores de conversión causados por las variaciones en la señal de entrada mediante un *MUESTREADOR-RETENEDOR*.

La referencia de voltaje de la conversión es provista por las terminales V_{RL} Y V_{RH} por lo que se recomienda aplicar V_{RL}= 0v y V_{RH}= 5v (la diferencia V_{RH}-V_{RL} no debe ser menor de 2.5 o 3v para que no exista degradación en el dispositivo). Con ésto, si la señal de entrada es V_{RL} la conversión dara $00 y si es V_{RH} dara $FF (escala completa). De esta forma cada conversión será hecha en 32 ciclos de reloj siempre y cuando la

velocidad del reloj sea mayor a 750 KHz.

A. Operación como canal-simple.

Están presentes 2 tipos de operación. Cuando el bit SCAN del registro ADCTL es limpiado, se harán 4 conversiones en *uno* de los canales seleccionados, colocadose el primer resultado en el registro ADR1 y el cuarto resultado en el registro ADR4. Al terminar las 4 conversiones el proceso se detiene. Cuando SCAN= 1 se estarán haciendo conversiones en dicho canal almacenando la quinta conversión en el registro ADR1, la sexta conversión reescrita en ADR2 y así sucesivamente.

B. Operación como canal-multiple.

Están presentes 2 tipos de operación. Cuando el SCAN= 0 se hacen 4 conversiones, *una por cada canal*; el primer resultado del canal 1 es almacenado en el registro ADR1, mientras que el resultado en el canal 4 es almacenado en el registro ADR4. Al terminar las 4 conversiones el proceso se detiene. Cuando SCAN= 1 se están haciendo conversiones en cada uno de los cuatro canales almacenando la quinta conversión en el registro ADR1, la sexta conversión reescrita en ADR2 y así sucesivamente, siendo esta última operación la empleada en el trabajo presente.

C. Registro ADCTL.

Este registro nos indica en su bit 7 si la conversión ha sido terminada, mientras que con sus otros bits configuramos el modo de operación y elegimos los canales de conversión.

D. Habilitación y selección del reloj.

La habilitación del convertidor es hecha con el bit ADPU= 1 del registro OPTION, mientras que la selección del reloj es hecha con el bit CSEL del mismo registro. Si CSEL= 0 el sistema A/D utiliza el reloj E > 750 KHz; cuando CSEL= 1 el sistema A/D utiliza el reloj interno RC el cual corre a 1.5 MHz.

I.7 RELOJ PROGRAMABLE.

A. Reloj programable.

Este reloj está formado por un contador de carrera libre de 16 bits que es controlado por el reloj E y del cual podemos obtener 4 preescalaciones (dividir por 1, 4, 8 ó 16) según la necesidad requerida. La elección del factor de preescalamiento se hace através del registro TMSK2; mientras que el valor actual de dicho contador está contenido en el registro TCNT.

El sistema tiene además 3 registros de entrada de captura y 5 registros de salida comparada (todos los registros de 16 bits).

Los registros de *entrada de captura* graban el valor del contador cuando un flanco es detectado en la linea de entrada. El flanco fija el bit ICxF del registro TFLG1 y puede causar una interrupción si el bit ICxI es fijado en el registro TMSK1. Estos registros son útiles para determinar mediante software el periodo y/o el ancho de pulso de una señal ó bién para establecer una referencia de tiempo.

Los registros de *salida comparada* son comparados con el contador de carrera libre y cuando ambos son iguales es mandada una señal de salida fijandose el bit OCxF de la bandera TFLG1. Esto nos permite producir un pulso de duración especifica ó generar un retardo. El retardo se lleva a cabo sumando el número correspondiente del retardo deseado al valor presente del contador y escribiendo el valor de dicha suma en uno de los registros de salida comparada. Por lo que la interrupción ocurrira cuando el contador llegue al valor de dicha suma. De esta manera fué como se controló la frecuencia de muestreo del presente trabajo.

I.8 E/S PARALELA.

Este sistema cuenta con 4 puertos que realizan las operaciones de E/S paralela. Permitiendo leer y/o escribir información através de los mismos. Las funciones de los puertos son descritas a continuación.

Los puertos C y D son usados como entradas y/o salidas de propósito general bajo el control de su respectivo registro de dirección de datos.

Los puertos A (excepto su terminal 7), B y E son usados como entradas ó salidas y no tienen registro de dirección de datos.

Los puertos B, C junto con las terminales STRA y STRB se emplean para los modos de *STROBE* y/o *HANDSHAKE*.

Por último mencionaremos los 2 tipos de comunicación con que cuenta el microcontrolador MC68HC11; Estos son:

1. Interfaz de Comunicación Serie (SCI).

Este sistema puede ser empleado para conectar una computadora personal ó una terminal al microcontrolador, ó a varios microcontroladores interconectados formando una red de comunicación serie. Esta comunicación está provista con un formato estandar NRZ (un bit de inicio, 8 o 9 bits de datos y un bit de paro) y gran variedad de velocidades de baudaje.

2. Interfaz Periférica Serie (SPI).

Como su nombre lo indica esta comunicación permite al microcontrolador comunicarse con otros dispositivos periféricos. Esta interfaz sincrona es capaz de interprocesar comunicación en un sistema *MAESTRO-MULTIPLE* y los dispositivos periféricos pueden ser tan simples como un registro TTL ó tan complejos como un subsistema completo tal como un display ó un convertidor A/D. La SPI es bastante flexible con los numerosos periféricos estandar y además puede ser configurada ya sea como maestro ó como esclavo.

Capítulo 2

Tarjeta MC68HC11EVB

II.1 INTRODUCCION.

En el presente capitulo será descrita la tarjeta de evaluación MC68HC11EVB a la que nos referiremos como EVB para simplificar términos. Presentandose de una manera breve su arquitectura y sus caracteristicas principales. Podrá ser consultada la referencia [8] si se requiere mayor información acerca de dicha tarjeta.

Esta tarjeta es una herramienta de bajo costo, la cual permite al usuario depurar o evaluar (emular) programas que son manejados por el microcontrolador MC68HC11.

Contiene un programa monitor llamado BUFFALO (Bit User Fast Friendly Aid to Logical Operations) el cual permanece residente en memoria EPROM externa al microcontrolador. Dicho programa es usado para depurar el ensamblado del código de usuario mediante comandos dados por la terminal.

II.2 CARACTERISTICAS.

-Capacidad de cargar una computadora huésped.
-Linea de ensamblado/desensamblado.
-Circuiteria para depurar/evaluar al MCU MC68HC11.
-MC68HC24FN Unidad de Remplazo de Puertos (PRU) con circuiteria de E/S para el microcontrolador.
-MC68B50 Adaptador de Interfaz para Comunicación Asincrona (ACIA) con circuiteria para el puerto de E/S de una terminal.
-RS-232C Compatible con los puertos de E/S de una terminal ó computadora huésped.

II.3 INSTRUCCIONES DE OPERACION.

La tarjeta EVB cuenta con 2 modos de operación: modo depurador y modo evaluador.

El modo *depurador* permite al usuario depurar el ensamblado de su código mediante 2 metodos:

1.-Utilizar el ensamblador de linea en el programa monitor via RAM de usuario EVB.

2.-Ensamblar desde la computadora huésped bajando el código a la RAM del usuario EVB via archivos-S de Motorola.

EL modo *evaluador* permite evaluar el código de usuario en el sistema objetivo utilizando la memoria del microcontrolador. El código deberá ser rehubicado de la RAM de usuario EVB a la ROM 8K del microcontrolador (modificando direcciones).

II.4 DESCRIPCION DEL HARDWARE.

El diagrama a bloques para la EVB es mostrado a continuación:

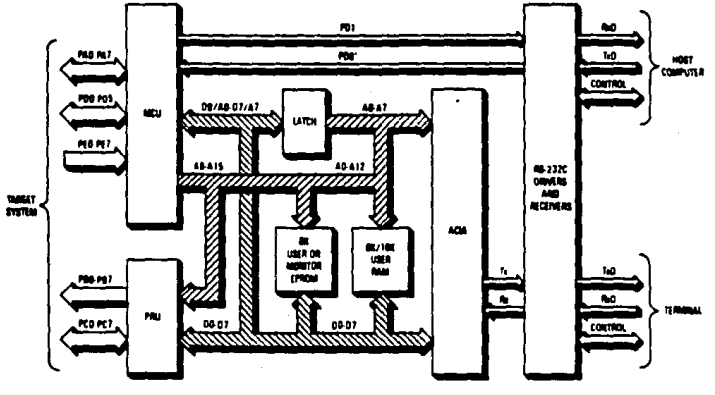


Fig. 2.1 Diagrama a bloques EVB.

1. Memoria.

La EVB tiene el mapa de memoria mostrado en la fig. 2.2.

Son provistos 8K de memoria RAM para evaluar programas normalmente sostenidos en ROM. Si se require adicionar mayor cantidad de memoria RAM deberá ser configurado el conector J3 de la tarjeta.

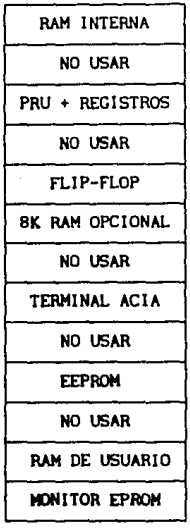


Fig. 2.2 Mapa de memoria EVB.

2. Microcomputadora.

La EVB y el microcontrolador operan en el modo multiplexado expandido. Para la operación de la EVB el registro CONFIG (implementado en la EEPROM del microcontrolador) deberá estar programado tal que su bit ROMON se encuentre limpio. Cuando el bit esté listo la ROM interna del microcontrolador se encontrará desabilitada y aquel espacio de memoria será externamente espacio accesado. Esto permite que en el espacio de memoria $E000-$FFFF se tenga el programa monitor BUFFALO pero en una EPROM externa (el programa monitor también utiliza las localidades $0036-$00FF de la RAM interna del microcontrolador).

La interfaz del sistema objetivo con la tarjeta es provista por el microcontrolador y el PRU.

3. Unidad de Remplazo de Puertos (PRU).

Esta unidad remplaza los puertos de E/S B y C del microcontrolador usados en modo simple y provee mediante el conector P1 las líneas de E/S requeridas en modo simple para la evaluación del sistema objetivo.

4. Circuitos de interfaz del puerto E/S RS-232C.

Estos circuitos proveen la comunicación y transferencia de datos entre la EVB y una terminal externa ó computadora huésped.

La EVB utiliza al circuito MC68B50 como ACIA, para comunicarse con el puerto (E/S) de la terminal (la velocidad de baudaje es seleccionada con el conector J5 entre los 300 y 9600 bauds). Una segunda comunicación se hace con el puerto (E/S) de la computadora huésped fijandose la velocidad de baudaje a 9600 bauds vía SCI y usando un reloj de 2 MHz (el baudaje se modifica mediante el registro ONSCI del programa monitor).

II.5 COMANDOS DEL MONITOR.

A continuación se describen algunos de los comandos más empleados:

ASM <dirección>.- Ensambla a partir de la dirección dada, permitiendo modificar el código de esa dirección (desensamble con [ctrl A]).

BR Despliega, fija o remueve los puntos de paro. Si durante la ejecución del programa es encontrado un punto de paro el programa se detendra.

G <dirección>.- Ejecuta o corre el programa a partir de la dirección indicada.

HELP.- Despliega información acerca de los comandos.

MD.- Despliega el contenido de la memoria a partir de una dirección o un bloque de memoria.

MM <dirección>.- Permite modificar o examinar el contenido de la dirección de memoria indicada.

RM [p,y,x,a,b,c,s].- Modifica el contenido de los registros: contador de programa, registro indexado y,x; acumulador a,b,c; apuntador de pila.

T<n>.- Ejecuta n instrucciones paso a paso mostrando el registro actual de los registros anteriormente mencionados.

II.6 PROCEDIMIENTO PARA BAJAR PROGRAMAS A LA TARJETA EVB.

A. Consideraciones.

La velocidad de baudaje entre la PC y la EVB deben ser identicas , el puerto asincrono de la PC debe estar configurado para modo *terminal* conectando el puerto serie de la PC al conector P2 de la EVB mediante el cable RS-232C. El programa a ejecutar debe tener la extensión *.MX* .

B. Procedimiento.

C>KERMIT	Prompt de la PC. Entrar al programa kermit
IBM-PC kermit-MS VX.XX	
Type ? for help	
kermit-MS>SET PORT2	Fijamos el puerto 2 de la PC
kermit-MS>SET BAUD 9600	Fijamos el baudaje de la PC
kermit-MS>CONNECT	Hacemos la conexión de la PC a la EVB
[connecting to host, type Control-] C to return to PC]	
(RETURN)	
>LOAD T	Comando para bajar el programa vía puerto terminal
(CTRL)]C	
kermit-MS>PUSH	
The IBM Personal Computer DOS	
Version X.XX (C)Copyright IBM Corp 1983	
C>TYPE (NOMBRE.MX)>COM2	Nombre del archivo serie Motorola
C>EXIT	Se baja el programa a la tarjeta
kermit-MS>CONNECT	Regresamos al programa monitor BUFFALO
(CTRL)]C	
kermit-MS>EXIT	salimos del programa kermit

Capítulo 3

Estudio del algoritmo

III.1 INTRODUCCION

El algoritmo aquí presentado es un algoritmo de control autosintonizable con base en un controlador proporcional derivativo (PD) para llevar a cabo el control de posición de un motor de C.D. Específicamente utilizaremos el servomecanismo modular de C.D. MS-150 fabricado por Feedback. Como primer punto requerimos conocer el modelo de la planta con la finalidad de poder calcular un controlador adecuado. El conocer el modelo de la planta implica conocer tanto su estructura como sus parámetros. En nuestro caso, suponemos conocida la estructura del modelo de la planta siendo ésta una suposición de gran importancia, ya que permitirá dirigir nuestra atención solo a la búsqueda de los parámetros de la planta. Estos parámetros se estiman con la ayuda de un algoritmo de identificación o estimación. Una vez estimado el modelo de la planta, se procede a sintonizar al controlador, es decir, a dar valores a los parámetros del él, en este caso a K_p y K_d para el PD. Finalmente, conectamos el controlador PD al motor a fin de controlar su posición. A continuación se describen detalladamente cada una de estas etapas.

III.2 MODELO DEL MOTOR

El primer paso consiste en establecer la estructura del modelo del motor alrededor de la cual se desenvolverá gran parte de nuestro estudio. Esta estructura deberá ser sencilla, y deberá incluir las características más relevantes de la dinámica del motor. Información acerca del modelado del motor se puede encontrar en [1], a continuación se presenta un

extracto de ella.

Los motores de C.D. pueden ser controlados en su velocidad y/o posición alimentando una señal a su inducido "control de inducido" ó a su campo "control de campo". En el primero (fig 3.1), el motor posee un campo fijo que se genera con base en un imán permanente o bien, alimentando con corriente continua a un bobinado en el estator, aparte, a su inducido o armadura se aplica una señal de cuya magnitud dependederá su funcionamiento.

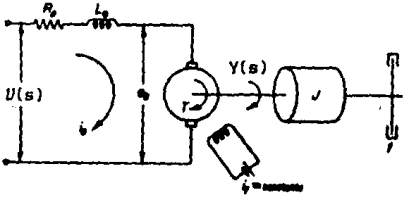


Fig. 3.1 Motor de C.D. controlado por inducido

donde Y(s) es la posición de la flecha del motor, V(s) su velocidad y U(s) el voltaje de alimentación al inducido.

La función de transferencia del motor teniendo como salida su posición angular y como entrada el voltaje alimentado al inducido es:

$$\frac{Y(s)}{U(s)} = \frac{Km}{s(Tm\,s + 1)} \qquad (3.1)$$

donde:

- Km = K / ($Ra \cdot F + K \cdot Kb$) = Ganancia del motor
- Tm = $Ra \cdot J$ / ($Ra \cdot F + K \cdot Kb$) = Constante de tiempo del motor
- K = Constante del par motor
- Kb = Constante de fuerza Electromotriz
- F = Coeficiente de fricción viscosa del motor(lbs-ft/rad/seg)
- J = Coeficiente de inercia del motor y carga (lbs-ft-seg^2)
- Ra = Resistencia del devanado de inducido (ohms)
- y(t) = Posición angular de la flecha del motor
- u(t) = Señal de alimentación

Para mas detalle ver [1].

Una representación equivalente se obtiene reordenando los parámetros

del modelo anterior tal que se muestren en forma explícita a los polos y a la ganancia estática de la siguiente forma:

$$\frac{Y(s)}{U(s)} = \frac{K}{s\,(\,s + a\,)} \qquad (3.2)$$

Donde: $K = K_m/T_m$

$a = 1/T_m$

En la función de transferencia anterior observamos la existencia de un par de polos, uno de ellos localizado en el origen (s=0), y el otro situado a la izquierda (s=-a) del eje imaginario; también se presenta una ganancia (K). Obsérvese que todos estos valores dependen de las características físicas del motor. Debe notarse además que, el comportamiento de este sistema en lazo abierto es inestable debido la existencia del polo en el origen (s=0).

Muchas veces, como en el caso del servomotor MS-150 de Feedback utilizado en la parte experimental de este trabajo, se cuenta con instrumentos capaces de sensar independientemente a las señales de velocidad y posición de un motor, y siendo que cada uno de estos instrumentos posee su propia ganancia, conviene establecer el siguiente diagrama a bloques para el motor:

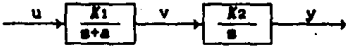


Fig. 3.2 Diagrama a bloques del servomotor

donde K_1 se debe a la ganancia estática del motor y a la ganacia del sensor de velocidad, y K_2 es la ganancia del sensor de posición.

Además, si $K_1 \cdot K_2 = K$, se obtiene la función de transferencia:

$$\frac{Y(s)}{U(s)} = \frac{K_1 \cdot K_2}{s\,(\,s + a\,)} \qquad (3.3)$$

Este será el modelo del motor a usar a lo largo del presente trabajo.

Puede verse también de la fig. 3.2 que la función de transferencia del modelo del motor teniendo como salida la velocidad es:

$$\frac{V(s)}{U(s)} = \frac{K_1}{s + a} \qquad (3.4)$$

III.3 CONTROLADOR

En el análisis y diseño de cualquier sistema de control, uno de los puntos más importantes a considerar, es definir la forma en cómo la planta se deberá controlar.

En un sistema de control de lazo cerrado, el controlador compara el valor efectivo de salida de una planta con el valor deseado, determina la desviación y produce una señal de control que reduce la desviación a cero o a un valor mínimo. La forma en que el controlador produce la señal de control recibe el nombre de *acción de control*.

En *Control clásico* se han planteado una serie de controladores a los que se clasifica de acuerdo a la manera en que generan su señal de control de la siguiente forma:

1.-Control proporcional (P)

2.-Control proporcional e integral (PI)

3.-Control proporcional y derivativo (PD)

4.-Control proporcional, integral y derivativo (PID).

El punto, ahora, es seleccionar alguna estructura para el controlador entre las antes citadas y anexarla a nuestro sistema de control.

Nuestro objetivo es controlar la posición de un motor de C.D., y necesitamos que la respuesta del sistema sea estable, rápida y sin grandes oscilaciones. Así que, tomando en cuenta estas características, se tomará a la estructura de control PD. En este tipo de controlador, la parte proporcional P ayuda a incrementar la velocidad de respuesta, mientras que la parte derivativa D ayuda en los transitorios haciéndo más amortiguado y estable al sistema. El seleccionar esta acción presenta la ventaja adicional de que al añadir el controlador al sistema no se incrementa su orden (como lo veremos en un análisis posterior), manteniéndose como un sistema de segundo orden, pudiendo así, utilizar la teoría ya desarrollada para analizar a éste tipo de sistemas.

A continuación se presenta un breve estudio comparativo entre los controles P y PD aplicados al modelo del motor.

A. Control P.

El control proporcional (fig. 3.3) no es más que un amplificador de ganancia ajustable.

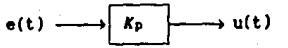


Fig. 3.3 Diagrama a bloques del controlador proporcional

donde:

K_p = Sensibilidad proporcional o ganancia.

$$\frac{U(s)}{E(s)} = K_p \qquad (3.5)$$

Teniendo al motor como la planta a controlar junto con el controlador de ganancia K_p en lazo cerrado, llegamos al siguiente diagrama a bloques:

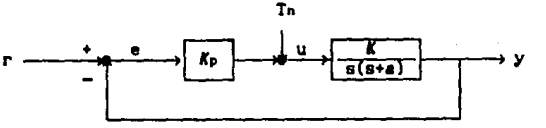


Fig. 3.4 Motor con controlador proporcional

La función de transferencia (F.T.) de lazo cerrado es :

$$Y(s) = \frac{K_pK}{s^2 + as + K_pK} R(s) + \frac{1}{s^2 + as + K_pK} T_n(s) \qquad (3.6)$$

donde R(s) es la señal de referencia y Tn(s) es una señal de perturbación de par.

Aplicando el criterio de Ruth, para que el sistema sea estable, deberá cumplirse que:

$a > 0$

$K_pK > 0$

pero $K > 0 \Rightarrow K_p$ tiene que ser mayor que cero.

Siendo el sistema estable y considerando un escalón de magnitud T_n como par perturbador, es posible encontrar el error de estado estable e_{ss} aplicando el teorema del valor final, de donde se obtiene:

$$e_{ss} = T_n/K_p \qquad (3.7)$$

Esto indica que en un sistema con controlador proporcional existe error de estado estable cuya magnitud es inversamente proporcional al valor de K_p. Sin embargo, si aumentamos este valor, la respuesta del sistema tiende a ser menos amortiguada. Revisando el lugar geométrico de

las raices (L.G.R.) también podemos observar dicho comportamiento.

La F.T. de lazo abierto es :

$$G(s) = \frac{KpK}{s(s+a)} \qquad (3.8)$$

De donde el L.G.R. es :

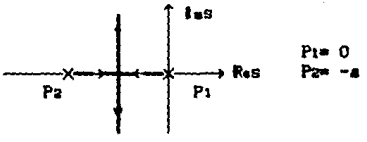


Fig. 3.5 L.G.R. para el sistema con controlador P.

En esta figura se observa que, para valores de K_p grandes, el sistema en malla cerrada tiene raices complejas y conjugadas cuya parte imaginaria tiende a crecer, provocando que los polos del sistema se alejen del eje real, obteniendose un comportamiento subamortiguado cada vez más oscilatorio.

B. Control proporcional-derivativo (PD)

Es posible amortiguar más al sistema si se añade al controlador P una parte derivativa. Existen diferentes formas de lograr la caracteristica derivativa, por ejemplo, se podria derivar directamente a la señal de error y afectarla por una constante (constante derivativa), o bien, si se tiene acceso a la señal de velocidad, se podria realimentar dicha señal afectándola también por una constante, esta estrategia es común encontrarla en aplicaciones industriales [4] y es el caso presentado a continuación.

En la figura 3.6 se muestra el diagrama a bloques del sistema de control con realimentación de velocidad y de posición del motor para lograr el controlador PD.

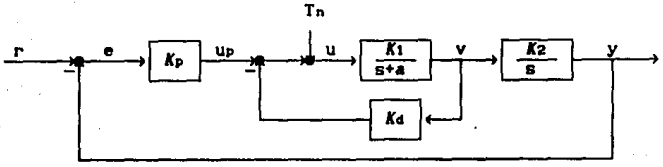


Fig. 3.8 Motor con controlador PD

Donde:

r = Entrada de referencia

e = r - y : error de desviación

K_p = constante proporcional ó ganancia

K_d = constante derivativa

La F.T. en lazo cerrado del sistema es:

$$Y(s) = \frac{K_p \cdot K_1 \cdot K_2}{s^2 + s(a + K_d \cdot K_1) + K_p \cdot K_1 \cdot K_2} R(s) + \frac{K_1 \cdot K_2}{s^2 + s(a + K_d \cdot K_1) + K_p \cdot K_1 \cdot K_2} T(s)$$

(3.9)

Aplicando el criterio de Ruth, para que el sistema sea estable, debe cumplirse que:

$$a + K_d \cdot K_1 > 0$$

$$K_p \cdot K_1 \cdot K_2 > 0$$

donde:

$a>0$, $K_1>0$ y $K_2>0$ $\Rightarrow$ $K_p>0$ y $K_d>-a/K_1$

Al igual que en el controlador P para calcular el error de estado estable se aplica un par perturbador de valor T_n encontrandose que:

$$e_{ss} = T_n/K_p \qquad (3.10)$$

Es decir sigue presente el error de estado estable. Sin embargo, ahora es posible disminuirlo (aumentando el valor de K_p) sin que la respuesta del sistema se vuelva oscilatoria ya que está presente la acción derivativa. Veamos el L.G.R. :

La F.T. de lazo abierto es :

$$G(s) = \frac{K_p K_1 K_2}{s\,(s + a + K_1 K_d)} \qquad (3.11)$$

De donde el L.G.R. respecto de K_p es mostrado en la fig. 3.7 .

Al hacer la comparación con el L.G.R. del control proporcional (fig. 3.5), se observa que ahora con el control PD se ha ampliado la distancia que existe entre los polos de lazo abierto en relación directa a K_d si K_1 es constante, provocando que los polos sean reales y distintos para un rango de valores de K_p más grande, ésto indica que se tiene la posibilidad de incorporar mayores valores de ganancia para la parte proporcional K_p logrando que la respuesta del sistema sea más rápida y más amortiguada a diferencia del caso del control solo proporcional.

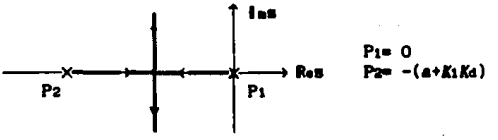


Fig. 3.7 L.G.R. para el sistema con controlador PD.

III.4 SINTONIZACION

Conocido el modelo de la planta y la estructura del controlador, habrá que diseñar algún mecanismo que nos permita generar los valores que tomarán los parámetros del controlador a fin de cumplir con nuestro objetivo de control. Este objetivo establece que la respuesta transitoria del sistema, esto es, la posición del motor, deberá cumplir con ciertas especificaciones.

En el apartado anterior se planteó un sistema de control que involucraba un controlador PD con realimentación de velocidad (fig. 3.6). La función de transferencia de lazo cerrado para ese sistema, sin incluir la perturbación de par es:

$$\frac{Y(S)}{R(S)} = \frac{K_p \cdot K_1 \cdot K_2}{s^2 + s(a + K_d \cdot K_1) + K_p \cdot K_1 \cdot K_2} \qquad (3.12)$$

Como se observa el sistema es de segundo orden, lo cual, como se mencionó anteriormente, proporciona una gran ventaja por ser uno de los sistemas para los cuales se tiene una mayor información y conocimiento.

Con el objeto de calcular los parámetros K_p y K_d realizaremos una igualación término a término de esta ecuación (3.12) con la función de transferencia general para los sistemas de segundo orden.

$$G(S) = \frac{\omega_n^2}{s^2 + s(2\xi\omega_n) + \omega_n^2} \qquad (3.13)$$

donde: ξ = Relación de amortiguamiento.

ω_n = Frecuencia natural no amortiguada del sistema.

De esta manera llegaremos a expresiones que relacionen a estos dos parámetros (K_p y K_d) con los parámetros del motor (K_1, K_2 y a) y con los parámetros ξ y ω_n.

$$K_p = \frac{\omega_n^2}{K_1 \cdot K_2} \qquad (3.14)$$

$$K_d = \frac{2\xi\omega_n - a}{K_1} \qquad (3.15)$$

Los parámetros ξ y ω_n definen la localización de los polos del sistema y por lo tanto su comportamiento y la forma de su respuesta, por lo que, estamos obligando a nuestro sistema a cumplir con ciertas especificaciones establecidas a través de ξ y ω_n.

Analizando las ecuaciones anteriores para K_p y K_d observamos que éstos parámetros han quedado en función de los parámetros del motor K_1, K_2 y a, y de los términos ξ y ω_n, como se había previsto.

Suponiendo conocidos los parámetros del motor, sólo queda asignar valores a los términos ξ y ω_n para así estar en la posibilidad de calcular los valores para K_p y K_d a través de las ecuaciones (3.14) y (3.15). Ver fig. 3.8.

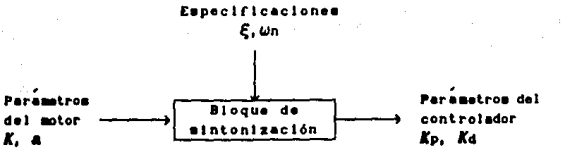


Fig. 3.8 Bloque de sintonización.

Para determinar los valores de los términos ξ y ω_n, nos basaremos en el cálculo de las *especificaciones de diseño*. Estas especificaciones se establecen en términos de la respuesta transitoria del sistema de segundo orden a una entrada escalón unitario (considerando al sistema originalmente en reposo), y son utilizadas para el diseño de sistemas de control, siendo algunas de estas:

1.-Tiempo de retardo, t_d

2.-Tiempo de crecimiento, t_r

3.-Tiempo de pico, t_p

4.-Sobreimpulso máximo, M_p

5.-Tiempo de establecimiento, t_s.

Estas especificaciones aparecen indicadas en la figura 3.9.

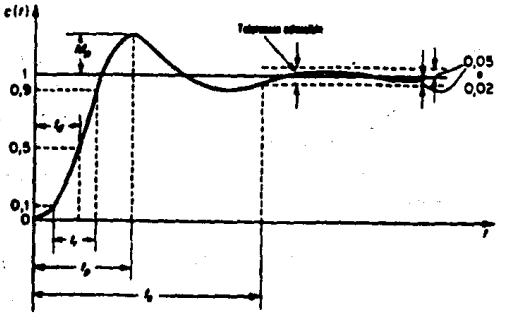


Fig. 3.9 Especificaciones de diseño.

Se hace notar que todos estos términos son válidos para casos subamortiguados, y solo algunos de ellos son aplicables a los casos sobre

ó criticamente amortiguados. Por ejemplo, para un sistema sobreamortiguado no se aplican t_p y M_p.

Así, para una respuesta transitoria deseable de un sistema de segundo orden, se recomienda que la relación de amortiguamiento esté entre 0.4 y 0.8 [1]. Valores inferiores a 0.4 para ξ dan excesivo sobreimpulso en la respuesta transitoria y un sistema con $\xi > 0.8$ responde muy tardiamente.

El valor de ξ generalmente es determinado por un requerimiento de máximo sobreimpulso (M_p) permitido, definiéndose este último como:

$$M_p = e^{-(\xi/\sqrt{1-\xi^2})\pi} \qquad (3.16)$$

de donde:

$$\xi = \frac{1}{\sqrt{\left(\frac{\pi}{\ln M_p}\right)^2 + 1}} \qquad (3.17)$$

Los límites de M_p para obtener una respuesta transitoria deseable (evaluados en la ec. anterior) son :

$M_p = 0.253$ para $\xi = 0.4$

$M_p = 0.015$ para $\xi = 0.8$

Por otro lado, se tiene que el tiempo de establecimiento para una banda de tolerancia del 2% se define como:

$$t_s = \frac{4}{\xi \cdot \omega_n} \quad \text{seg} \qquad (3.18)$$

Note que el tiempo de establecimiento está determinado principalmente por ω_n por lo que:

$$\omega_n = \frac{4}{\xi \cdot t_s} \quad \text{rad/seg} \qquad (3.19)$$

De lo anterior concluimos que, especificando un cierto sobrepaso M_p y un cierto tiempo de establecimiento t_s para la respuesta del sistema, podemos conocer la magnitud de ξ y de ω_n. Cabe aclarar que pudieron haberse tomado otras especificaciones como t_r, que da una medida de la velocidad de respuesta del sistema, con las que se conseguirían similares resultados.

Existen otros métodos de sintonización para controladores PID y sus derivados [4]. Muchos de ellos se basan en la forma de la respuesta en el tiempo o en la frecuencia de la planta a controlar y con base en ella se calculan los parámetros que habrá de tomar el controlador, estos métodos

generalmente no precisan del conocimiento del modelo de la planta, tan solo requieren revisar la respuesta del sistema tras aplicar una determinada señal de excitación que puede ser un escalón o una senoidal de cierta frecuencia entre otras. Estos métodos al igual que el aquí expuesto son aproximaciones pero que han mostrado generar resultados aceptables.

III.5 ALGORITMO DE IDENTIFICACION

Gran parte de las técnicas de diseño de sistemas de control se desarrollan con base en el buen entendimiento de la planta bajo estudio así como de su ambiente. Sin embargo, en muchas ocasiones la planta a controlar es muy compleja, y a veces, entenderla por completo resulta demasiado difícil, e inclusive, en muchas de las ocasiones es imposible. Es por esto que dichas técnicas de diseño de sistemas de controladores requieren incorporar alguna *técnica de identificación*, destinada a obtener un mejor entendimiento de la planta a controlar, ya sea en su estructura y/o en los valores de sus parámetros, estos casos son conocidos en la literatura como identificación estructural e identificación paramétrica respectivamente.

Los sistemas de identificación son en general *recursivos*, esto es, permiten actualizar periódicamente el modelo de la planta con base en los estimados previos más la lectura de algunas ciertas señales provenientes de la planta [4 cap. 3].

En un sistema de identificación se prefiere seleccionar una estructura para el modelo del proceso, a ésta se le considera fija y muy cercana al caso real, restando solo el problema de obtener o estimar los parámetros del motor [3 cap. 13].

El algoritmo o procedimiento que se sigue para este fin, se puede resumir de la siguiente manera: Se excita a la planta con una cierta señal U "rica" en frecuencias [4 cap. 1], se toman lecturas de la señal de entrada U y de salida de la planta V y con base en ellas se evalúan una serie de ecuaciones en forma recursiva tal que, su resultado converge a los valores de los parámetros del modelo de la planta (fig. 3.10).

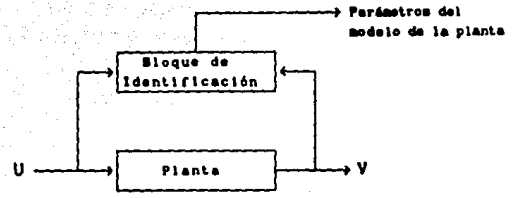


Fig. 3.10

A continuación se describe con mas detalle este proceso.

A. Parametrización

Parametrizar una ecuación consiste en obtener una representación equivalente para dicha ecuación pero en otro espacio. Se hace uso de este artificio matemático con el objeto de facilitar el análisis de dicha ecuación.

La función de transferencia del modelo del motor tomando como salida su velocidad es:

$$\frac{V}{U} = \frac{K_1}{s + a} \qquad (3.19)$$

haciendo

$$\frac{V}{U} = \frac{K_1}{s + a + \lambda - \lambda} \qquad \lambda>0 \qquad (3.20)$$

reordenando la ecuación anterior llegamos a:

$$V = (\lambda - a)\frac{V}{s + \lambda} + K_1 \frac{U}{s + \lambda} \qquad (3.21)$$

si se define

$$\theta_1 = \lambda - a \qquad (3.22a)$$

$$\theta_2 = K_1 \qquad (3.22b)$$

$$\phi_1 = \frac{1}{s + \lambda} V \qquad (3.23a)$$

$$\phi_2 = \frac{1}{s + \lambda} U \qquad (3.23b)$$

Entonces se puede observar a θ_1 y θ_2 como un par de parámetros que han sustituido a los parámetros K_1 y a del modelo original, aunque sus valores continúan relacionados a estos. ϕ_1 y ϕ_2 son señales que pueden ser

obtenidas por filtrado estable de las señales de entrada U y de salida V de la planta respectivamente. En ambos casos el filtro es de primer orden con el polo en $s=-\lambda$.

Resumiendo el resultado anterior y expresandolo en forma matricial queda:

$$V = [\theta_1 \quad \theta_2] \begin{bmatrix} \phi_1 \\ \phi_2 \end{bmatrix} \tag{3.24}$$

$$V = \theta^T \phi \tag{3.25}$$

Nótese que esta forma de presentar al modelo de la planta aporta la ventaja de poder tener concentrados, en un solo vector, a los parámetros del modelo. Esto además nos dice que, si se conocen los parámetros verdaderos del modelo y el valor de las señales $\phi(t)$ se puede evaluar esta ecuación y obtener así el valor de la señal de salida.

Si en lugar de conocer los parámetros reales del modelo tenemos unos valores estimados para los mismos, la ecuación toma la siguente forma:

$$\hat{V} = \hat{\theta}^T \phi \tag{3.26}$$

Esto quiere decir que, si ahora contamos con las señales $\phi(t)$ (lo cual es posible por medición directa) y con los estimados de los parámetros del modelo $\hat{\theta}$ entonces podemos tener también una aproximación de la señal de salida de la planta $\hat{V}$.

B. Ley de adaptación

Calculando el valor estimado de la salida de la planta $\hat{V}$ como se describió en la sección anterior (ec. 3.26) y dado que tenemos acceso a la señal de salida de la planta V, podemos obtener la diferencia que existe entre ambas señales de velocidad de la siguiente manera:

$$e_1 = \hat{v} - v \tag{3.27}$$

Y de la ecuación (3.26)

$$e_1 = \hat{\theta}^T \phi - v \tag{3.28}$$

A esta diferencia le denominamos *error de identificación* [5 cap. 0].

El problema clave ahora es obtener un mecanismo de ajuste tal, que conforme estime los parámetros del modelo y éstos se acerquen al valor real, haga tender hacia cero al error de identificación. En la literatura se han reportado un gran número de estos mecanismos a los que se les refiere como *leyes de adaptación* [5 cap. 2]. En nuestro caso, para la

implementación del algoritmo de identificación, se incorporó la denominada *ley de adaptación tipo gradiente*, la cual fué utilizada en las primeras aplicaciones de esquemas de identificación mostrando buenas propiedades de convergencia y estabilidad. Esta ley se basa en la idea de reducir $e_1^2(\theta)$ ajustando los parámetros θ en una dirección decreciente, es decir:

$$\frac{d\hat{\theta}}{dt} = -g \frac{\partial}{\partial \hat{\theta}} (e_1^2(\hat{\theta})), \qquad g>0 \qquad (3.29)$$

$$= -2g\, e_1(\hat{\theta}) \frac{\partial}{\partial \hat{\theta}} (e_1(\hat{\theta})) = -2g\, e_1(\hat{\theta}) \frac{\partial}{\partial \hat{\theta}} (\hat{v} - v) =$$

como v no depende de $\hat{\theta}$ entonces

$$= -2g\, e_1(\hat{\theta}) \frac{\partial}{\partial \hat{\theta}} (\hat{v}) \qquad (3.30)$$

y de la ecuacion (3.26)

$$\frac{\partial}{\partial \hat{\theta}} (\hat{v}) = \phi \qquad (3.31)$$

sustituyendo tenemos

$$\frac{d\hat{\theta}}{dt} = -2g\, e_1\, \phi \qquad (3.32)$$

haciendo $\gamma = 2g$ nos queda

$$\frac{d\hat{\theta}}{dt} = -\gamma\, e_1\, \phi \qquad (3.33)$$

donde llamamos a:

- γ ganancia de adaptación
- $\phi(t)$ vector de señales filtradas v(t) y u(t)
- $\hat{\theta}(t)$ vector de parámetros estimados
- $e_1(t)$ error de identificación

Sabemos por la condición de *excitación persistente* [4 cap. 1] que, si las señales $\phi_1(t)$ y $\phi_2(t)$ son señales "ricas" en frecuencia entonces el valor estimado de los parámetros $\hat{\theta}(t)$ tenderá a su valor real.

A partir de los parámetros estimados $\hat{\theta}_1$ y $\hat{\theta}_2$ se pueden obtener los estimados de los parámetros originales de la planta. Despejando K_1 y a de las ecuaciones (3.22) y considerando que ambos parámetros son estimados, tenemos:

$$\hat{a} = \lambda - \hat{\theta}_1 \qquad (3.34a)$$

$$\hat{K}_1 = \hat{\theta}_2 \qquad (3.34b)$$

Finalmente el esquema del algoritmo queda de la siguiente manera:

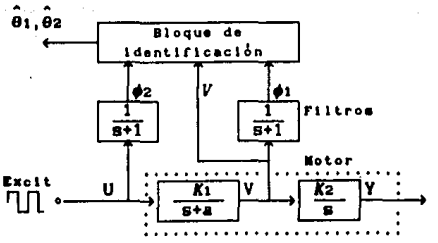


Fig. 3.11 Esquema del algoritmo de identificación

Nótese en este diagrama que las señales u(t) y v(t) se hacen pasar a traves de filtros de primer orden con polos en s=-λ a fin de obtener las señales ϕ_1 y ϕ_2. Estos filtros se implementan fuera del bloque de estimación escogiendo por facilidad λ=1. Nótese también que se está alimentando una señal cuadrada a la planta con lo cual se asegura que las señales ϕ_1 y ϕ_2 sean "ricas" en frecuencia. Estas dos últimas señales junto con la de velocidad v(t) se introducen al bloque de estimación con el fin de evaluar las ecuaciones (3.28) y (3.33) de donde se tendrá como solución a los estimados de los parámetros $\hat{\theta}_1$ y $\hat{\theta}_2$ y de allí usando las ecuaciones (3.34) a los estimados de los parámetros originales de la planta $\hat{K}$ y $\hat{a}$.

Conociendo ya los valores estimados de los parámetros del modelo, procedemos a calcular los parámetros del controlador K_p y K_d, en la forma descrita en la sección anterior (sec. III.4), es decir, podemos sintonizar al controlador. El proceso de estimación de los parámetros será realizado utilizando un microcontrolador, por esta razón es necesario discretizar a las ecuaciones envueltas en dicho proceso.

C. Discretización

En primer lugar se discretizan las ecuaciones (3.28) y (3.33) contenidas en el bloque de estimación.

De la ecuación (3.33)

$$\dot{\hat{\theta}}_1 = -\gamma\, \phi_1\, e_1 \qquad (3.35a)$$

$$\dot{\hat{\theta}}_2 = -\gamma\ \phi_2\ e_1 \qquad (3.35b)$$

La derivada de una cierta función puede interpretarse como el valor de la pendiente de la curva que describe dicha función evaluada en un punto dado. Usando esta interpretación podemos realizar una aproximación a la derivada que además será factible de evaluarse por medios digitales. El proceso que se sigue es el siguiente: se considera al tiempo como la variable independiente y a la variable de la cual se extrae la derivada (en este caso θ_1 ó θ_2) la variable dependiente, y se calcula el cociente entre el incremento de la variable dependiente y el incremento correspondiente en el tiempo. Por facilidad se escoge al incremento en el tiempo del tamaño del periodo de muestreo, por lo que el incremento en la variable dependiente resulta ser la diferencia entre dos muestras contiguas. Este es en realidad un método de discretización equivalente al "método de discretización rectangular hacia atrás", del que se podría obtener una transformación $s \rightarrow z$ y de allí obtener una ecuación en diferencias equivalente. Es práctica común el utilizar estos métodos de discretización [3].

Aplicando el concepto anterior a las ecuaciones (3.35a) y (3.35b) obtenemos:

$$\frac{\hat{\theta}_1(k+1) - \hat{\theta}_1(k)}{h} = \gamma\ \phi_1(k)\ e_1(k) \qquad (3.36a)$$

$$\frac{\hat{\theta}_2(k+1) - \hat{\theta}_2(k)}{h} = \gamma\ \phi_2(k)\ e_1(k) \qquad (3.36b)$$

despejando queda:

$$\hat{\theta}_1(k+1) = \hat{\theta}_1(k) + h\ \gamma\ \phi_1(k)\ e_1(k) \qquad (3.37a)$$

$$\hat{\theta}_2(k+1) = \hat{\theta}_2(k) + h\ \gamma\ \phi_2(k)\ e_1(k) \qquad (3.37b)$$

Note que esta forma de discretizar ofrece la ventaja de incluir pocas operaciones aritméticas lo que permite que los cálculos sean rápidos.

Para el error de identificación, discretizando la ec. (3.28) obtenemos:

$$e_1(k) = \hat{\theta}_1(k)\ \phi_1(k) + \hat{\theta}_2(k)\ \phi_2(k) - v(k) \qquad (3.38)$$

Las ecuaciones (3.37) y (3.38) se evaluan conjuntamente en cada ciclo de muestreo. La ecuación (3.38) genera el valor del error de identificación, el cual será utilizado por las ecuaciones (3.37) para

estimar los parámetros. Las condiciones iniciales para las ecuaciones (3.37) pueden considerarse nulas o bién asumir algún valor, y dependiendo de la cercanía de éste a los valores reales de los parámetros, el algoritmo tardará menos tiempo en converger.

D. Criterio de convergencia

A medida que la ley de actualización hace converger a los estimados de los parámetros del motor a su valor real, también hará tender al error de identificación hacia cero. Surge entonces el problema de saber hasta que momento mantener activada la identificación, es decir, en que momento podemos asegurar que ya se cuenta con estimados aceptables para los parámetros del modelo.

Para reconocer ese instante, se establece un criterio con base en la observación del error de identificación, y lo denominamos *criterio de convergencia*. Se intuye que la forma de la señal del error de identificación será decreciente y oscilatoria con una frecuencia similar a la de la señal cuadrada de excitación alimentada a la planta . El criterio aprovecha esta característica decreciente del error de identificación y lo procesa a fin de hacer más clara dicha característica. El proceso consiste en calcular el valor promedio de los valores absolutos de los diez últimos errores de identificación que se han presentado hasta un cierto instante de muestreo. A este promedio le denominamos *error de identificación medio* y lo denotamos e_{im}.

$$e_{im} = \frac{1}{10} \sum_{i=k-10}^{k} |e_i(i)| \qquad (3.39)$$

Así, en el momento en que el e_{im} alcance un valor inferior a un cierto límite previamente establecido, podemos decir que el algoritmo ha convergido, es decir, ha alcanzado valores para los estimados de los parámetros muy cercanos a sus valores reales. Alcanzado ese momento, se desactiva el bloque de identificación y se procede a sintonizar al controlador que, finalmente, se conecta al sistema.

III.6 SIMULACION

Se llevaron a cabo simulaciones digitales del algoritmo de identificación y del sistema en lazo cerrado con el controlador propuesto, utilizando para ello el paquete de simulación SIMNON [6].

A. Algoritmo de identificación

En esta simulación se incluyen las representaciones del modelo del motor, de los dos filtros y del bloque de identificación junto con el criterio de convergencia (fig. 3.12). El modelo del motor se representa como un modelo de primer orden continuo con sus parámetros K_1, K_2 y a fijos. Los filtros que generan las señales ϕ_1 y ϕ_2 son representados como un par de sistemas continuos de primer orden con su polo en $s=-\lambda$. El bloque de identificación junto con el criterio de convergencia se tienen representados como un sistema discreto (ver Apéndice B).

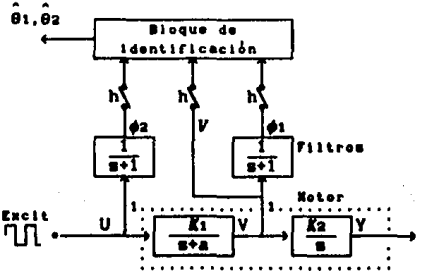
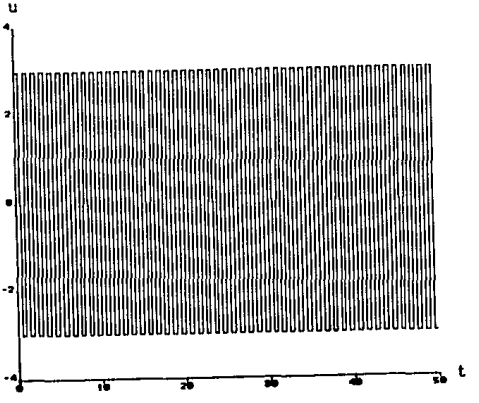


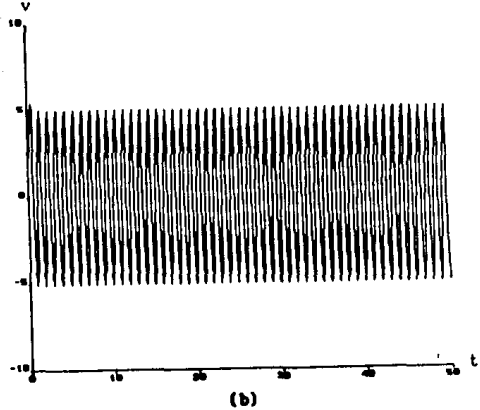
Fig. 3.12 Estructura del algoritmo de identificación.

Se dieron valores a los parámetros nominales del modelo del motor K_1=10, K_2=7.773 y a=5 que son valores muy cercanos a los reales, se propuso λ=1 para los dos filtros así como una ganancia de adaptación γ=3, esta ganancia nos permite tener una rápida convergencia en los parámetros sin salir de estabilidad. Se alimentó a la planta con una señal cuadrada de amplitud ±3 volts a una frecuencia de 2 Hz (fig. 3.13a), con la cual se aseguró excitación persistente al sistema. Finalmente el programa de simulación fué corrido usando una frecuencia de muestreo de 100 Hz (h=0.01

seg.) para la parte discreta.

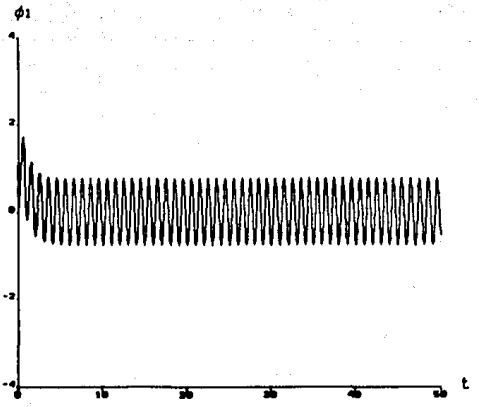


(a)

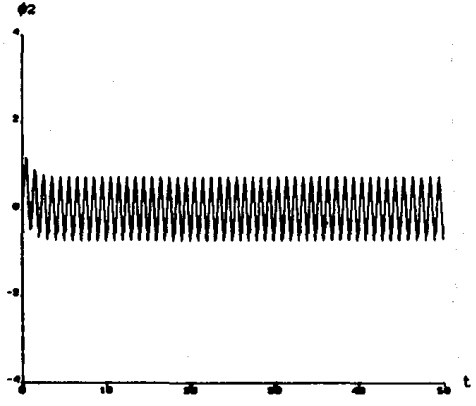


(b)

Fig. 3.13 Señal de (a) excitación, (b) velocidad



(a)

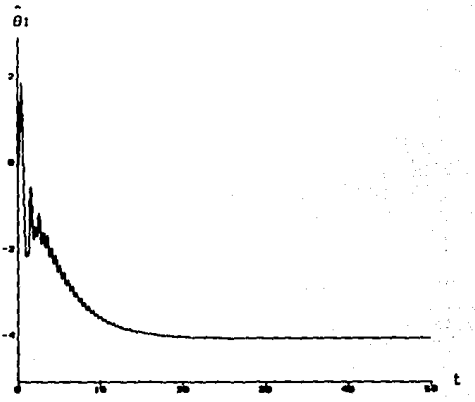


(b)

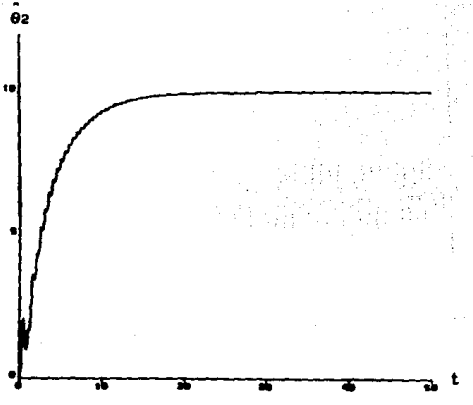
Fig. 3.14 (a) ϕ_1 y (b) ϕ_2

Las señales anteriores (figs. 3.13 y 3.14) son utilizadas dentro del bloque de identificación y con ellas se estiman los parámetros $\hat{\theta}_1$ y $\hat{\theta}_2$ que

siguen las trayectorias mostradas en las siguientes figuras:



(a)

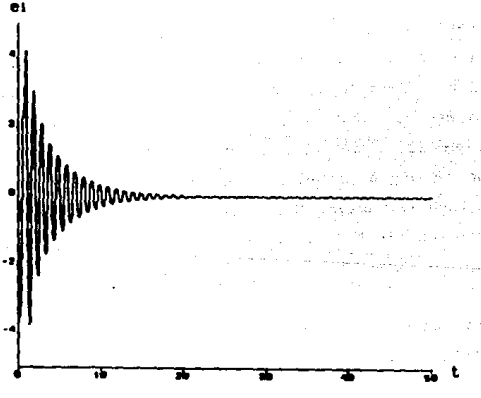


(b)

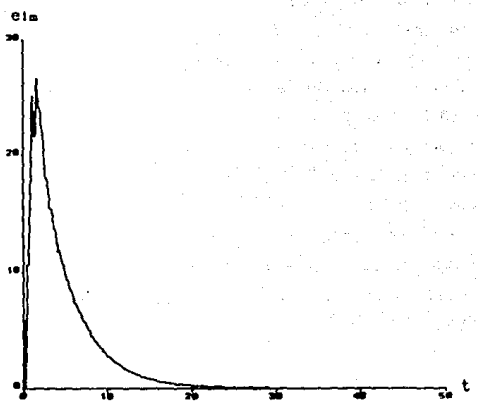
Fig. 3.16 Trayectorias de los parámetros estimados (a) $\hat{\theta}_1$ y (b) $\hat{\theta}_2$

Nótese como ambos parámetros convergen asintóticamente a unos ciertos valores. Así, el parámetro $\hat{\theta}_2$ converge al valor de 10 mientras que el parámetro $\hat{\theta}_1$ converge al valor -4. A partir de estos dos valores podemos calcular los valores de los parámetros originales del modelo del motor con las ecuaciones (3.34a) y (3.34b) de donde resulta $K_1 = 10$ y $a = 5$, que son los valores propuestos originalmente para el modelo del motor, por lo que queda demostrada la efectividad del algoritmo.

Ahora bien, como se mencionó con anterioridad, para desactivar la identificación de los parámetros se requiere aplicar un criterio de convergencia basado en la observación de la característica decreciente del error de identificación. En las siguientes gráficas (fig. 3.16) se muestran las señales del error de identificación e_i así como del error de identificación medio e_{im}. Nótese como efectivamente e_i tiende a decrecer, y como el e_{im} nos ayuda a detectar más claramente esta característica.



(a)



(b)

Fig. 3.16 (a) Error de identificación y
(b) Error de identificación medio

Ahora bastará tan solo con comparar el e_{im} con un cierto límite tal, que cuando el e_{im} sea inferior a este, se detenga la identificación. Este límite deberá ser lo suficientemente pequeño para que se asegure una buena estimación. Adviértase además que la trayectoria del e_{im} empieza con un valor de cero que es ya inferior al del límite establecido, por lo que deberá pensarse en algún mecanismo que impida una desactivación errónea del bloque de identificación. Una solución a este problema consiste en dejar transcurrir un cierto periodo de tiempo antes de empezar a revisar el valor de e_{im}.

B. Etapa de control

Una siguiente simulación consistió en probar el sistema en su etapa de control, es decir, el motor conectado en lazo cerrado con su controlador PD (fig. 3.17). En este caso, nuevamente se representa al motor como un sistema continuo, mientras que al controlador como un sistema discreto (ver apéndice B).

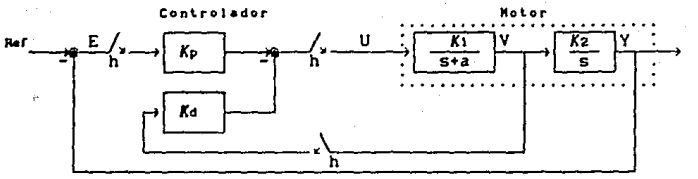


Fig. 3.17 Sistema en lazo cerrado con controlador PD.

Los parámetros que toma el motor K_1, K_2 y a son los mismos que en el caso anterior. Los parámetros del controlador se calculan con base en las ecuaciones (3.14) y (3.15), considerando las especificaciones ξ=0.3578 y ω_n=25.7126 que corresponden a M_p=30% y t_s=0.5 seg. Se corrió el programa con una frecuencia de muestreo para la parte discreta de 100 Hz. En la figura 3.18 se muestra la gráfica resultante para la salida del sistema (posición del motor) para los datos anteriores.

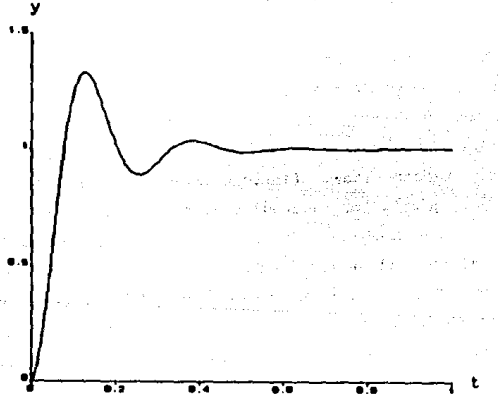


Fig. 3.18 Señal de salida (Y) del motor.

De esta gráfica se obtienen los dos siguientes valores:
M_p=32.62%

t_s=0.5217 seg.

que son valores muy cercanos a los valores propuestos.

Como resultado de las simulaciónes anteriores, llegamos a la conclusión de que el algoritmo descrito funciona en la forma esperada, con lo que se está en la posibilidad de poder llevar a cabo su implementación.

III.7 RESUMEN

El algoritmo de control aquí estudiado, se desarrolla en tres etapas (fig. 3.19). En la primera (interruptores (1) cerrados), se realiza la identificación de la planta, para esto, se excita al motor con una señal cuadrada, esta señal de excitación U y la de velocidad del motor V son filtradas con filtros de primer orden para generar las señales ϕ_1 y ϕ_2 que, junto con V, son alimentadas al bloque de identificación. El algoritmo de identificación evalúa un conjunto de ecuaciones en diferencias de las cuales se generan estimados para los parámetros del motor. Se implementó además un criterio de convergencia con la finalidad de detener al proceso de estimación una vez que se hallan alcanzado estimados aceptables. Con base en estos estimados y en ciertas especificaciones deseadas para la respuesta del sistema pasamos a la etapa de sintonización del controlador (interruptor (2) cerrado) donde el bloque de sintonización se encarga de calcular los parámetros o ganacias a alimentarse al controlador, en este caso, K_p y K_d para el PD. Finalmente pasamos a la tercer etapa (interruptores (3) cerrados), aquí se desconectan de la planta los bloques de identificación y sintonización, y en su lugar se conecta el controlador PD con sus parámetros recien actualizados, el cual lleva a cabo el control de la posición del motor, para esto, toma lecturas del error de la salida del sistema respecto de la referencia y también de la velocidad del motor, y con ellas genera la señal de control que habrá de ser alimentada al motor para cerrar el lazo.

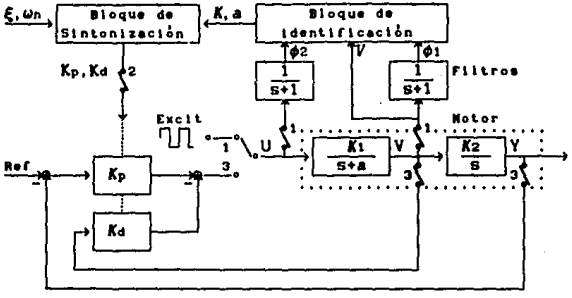


Fig. 3.19 Diagrama a bloques del algoritmo de control.

Capítulo 4

Implementación del algoritmo

IV.1 INTRODUCCION

En este capítulo describiremos la forma en que se implementa el algoritmo de control estudiado usando el microcontrolador MC68HC11 [7]. El algoritmo se probó con la ayuda de la tarjeta de evaluación M68HC11EVB [8] y el kit MS-150 para motor de C.D. de Feedback [2]. El proceso consiste en identificar los parámetros del modelo del motor para sintonizar el controlador PD y posteriormente pasar a la etapa de control, en la que se pretende controlar la posición angular de la flecha del motor de C.D. Los bloques de identificación y sintonización, así como el controlador PD (fig. 3.16) son bloques de tipo digital y su implementación se realizó con base en software dentro del microcontrolador. La señal de excitación también fué generada por el microcontrolador a través de un programa. En lo que respecta a los filtros, debido a su sencillez y para evitar un mayor número de cálculos dentro del microcontrolador, se prefirió alambrarlos analógicamente como se describirá mas adelante.

Primeramente se procedió a diseñar una tarjeta a la que denominamos *tarjeta de interfaz* la cual es capaz de acoplar las señales provenientes del motor al microcontrolador y viceversa (fig. 4.1). Esta tarjeta incluye un convertidor D/A, un par de filtros y algunos amplificadores.

Teniendo lista la circuitería externa necesaria, se desarrolló el software que permitiría al microcontrolador ejecutar cada una de las etapas del algoritmo (identificación, sintonización y control), y además controlar el paso entre cada una de ellas.

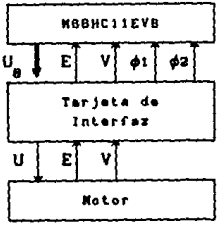


Fig. 4.1 Transmisión de señales entre M68HC11EVB y motor.

En este capitulo se describe en principio el kit del servomotor empleado, la tarjeta de interfaz y el programa desarrollado.

IV.2 SERVOMECANISMO MODULAR DE C.D. MS-150

El servomecanismo modular de C.D. [2] esta formado por los siguientes módulos (fig. 4.2):

Potenciometros de posición. Convierten las posiciones de entrada y salida en señales eléctricas proporcionales.

La unidad operacional (OU). Es un sumador y en particular se usa como nodo comparador de un sistema de control; se tiene una ganancia G=-1 para cada voltaje de entrada.

La unidad atenuadora (AU). Es un potenciómetro no aterrizado empleado como divisor de voltaje.

Preamplificador (PA). El voltaje de salida de este bloque idealmente siempre es positivo y proporciona una ganancia de 50 con un voltaje máximo de 15 volts.

Módulo Servo-Amplificador. Este módulo controla directamente al motor y se puede alambrar para una excitación de campo o de armadura (con la excitación de armadura se obtiene un comportamiento más estable del sistema).

Tacogenerador (TG). Este dispositivo es un pequeño generador que proporciona un voltaje proporcional a la velocidad angular del eje rápido del motor.

Freno magnetico (FM). Este dispositivo magnético permite aplicar diversas cargas a la flecha del motor. Su principio de operación se basa en la aparición de una fuerza contraria a la dirección del movimiento de una superficie conductora que se desplaza a travez de un campo magnético. Este freno está graduado en porcentajes (0-100%) para diferentes posiciones, donde dicho porcentaje aumenta conforme aumenta la superficie de interacción entre el freno y un disco de aluminio atado a la flecha del motor.

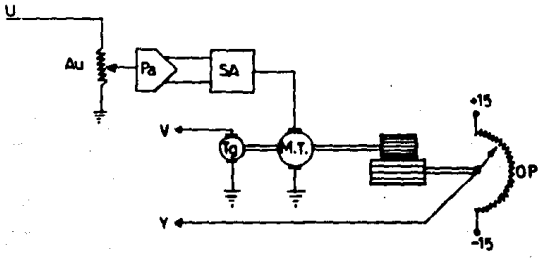


Fig. 4.2 Esquema del servomotor MS-150.

Ya que en el algoritmo se van a estimar dos parámetros, es deseable que ambos tengan valores más o menos cercanos, es decir, que esten dentro de la misma escala, en el caso de la ganancia K_1, se espera que esta sea de un gran valor comparada con el valor del polo a, así que, se optó por añadir un atenuador a la entrada del sistema para compensar esta elevada ganancia debida fundamentalmente al preamplificador.

En la figura 4.3 se muestra el diagrama a bloques del sistema anterior en el que consideramos que se cuenta con sensores para cada una de las variables del sistema, velocidad y posición angular, es decir, tacogenerador y potenciómetro de salida respectivamente.

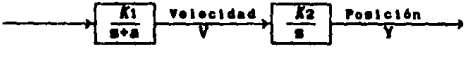


Fig. 4.3 Diagrama a bloques del servomotor

donde a es el polo del sistema, K_1 es la ganancia estática tomada desde el atenuador hasta el tacogenerador y K_2 es un factor que relaciona las ganancias de ambos sensores.

La función de transferencia del sistema considerando como entrada a la señal de control U y como salida a la señal de posición Y es:

$$\frac{Y}{U} = \frac{K_1 K_2}{s(s+a)} \qquad (4.1)$$

Y considerando como salida a la señal de velocidad V queda:

$$\frac{V}{U} = \frac{K_1}{s+a} \qquad (4.2)$$

Adviértase que estas ecuaciones corresponden exactamente a las obtenidas en el capítulo anterior para el modelo del motor.

Los parámetros K_1 y a se van a identificar con la ayuda del algoritmo. El valor de estos dos parámetros puede estar variando dependiendo de diversos factores entre los que destaca la cantidad y tipo de carga que se le imponga al motor. El parámetro K_2, sin embargo, se considerará fijo ya que sólo nos está relacionando las ganancias de los dos sensores, las cuales se mantienen prácticamente fijas, siendo innecesaria su identificación.

El valor de K_2 se obtiene experimentalmente y a través de medios gráficos de la siguiente manera:

Se alimenta una señal constante al motor:

$$u(t) = c_1 \qquad (4.3)$$

Cuando el motor alcanza una velocidad constante, se toma la lectura de la salida del tacogenerador, la cual deberá ser también constante:

$$v(t) = c_2 \qquad (4.4)$$

Usando el graficador, se traza la señal de posición proveniente del potenciómetro de salida, esta señal resulta ser una recta con una cierta pendiente:

$$y(t) = m \cdot t \qquad (4.5)$$

dividiendo la ecuación (4.5) por la ecuación (4.4) obtenemos:

$$\frac{y(t)}{v(t)} = \frac{m}{c_2} t \qquad (4.6)$$

Por otro lado, del modelo del motor sabemos que:

$$Y = \frac{K_2}{s} V \qquad (4.7)$$

y en el dominio del tiempo, considerando y(0)=0, tenemos:

$$y(t) = K_2 \cdot v(t) \cdot t \qquad (4.8)$$

de donde:

$$\frac{y(t)}{v(t)} = K_2 \cdot t \qquad (4.9)$$

Comparando la ecuaciones (4.6) y (4.9) llegamos a que el valor de K_2 esta dado por:

$$K_2 = \frac{a}{c_2} \qquad (4.10)$$

Siguiendo el procedimiento anterior se encontró el siguiente valor para K_2:

$$K_2 = 7.773\ s^{-1}$$

IV.3 TARJETA DE INTERFAZ

La implementación de esta tarjeta cumple con el objetivo de acoplar las señales que van de la tarjeta evaluadora M68HC11EVB hacia el servomecanismo MS-150 y viceversa.

Siguendo el esquema de la figura 4.1, se observa que el motor recibe una señal de control U proveniente del microcontrolador, esta señal deberá ser de tipo analógica pues la planta es continua, sin embargo, el microcontrolador no es capaz de entregarnos una señal con esta caracteristica, así que, es necesario implementar un convertidor D/A, el cual recibirá paquetes de 8 bits provenientes del puerto B de salida del microcontrolador y nos entregará una señal analógica que puede ya ser alimentada al motor. El convertidor D/A se implementa con base en el circuito DAC0800 [10] el cual, tiene 8 bits de resolución, y está alambrado en una configuración tal, que permite un rango de voltaje de salida de -10 a +10 volts (fig. 4.4).

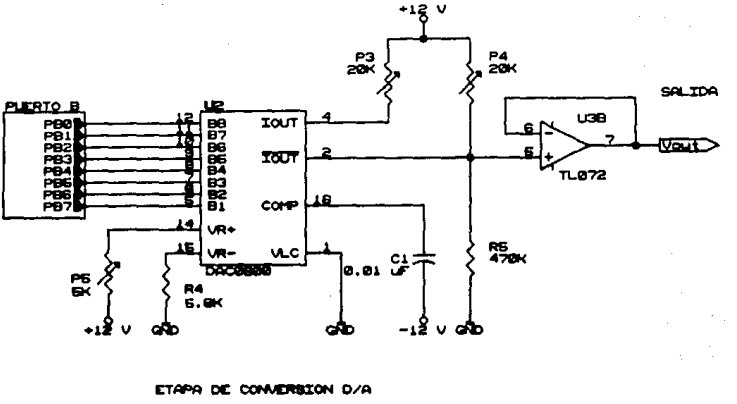


Fig. 4.4 Convertidor D/A.

También de la figura 4.1 se observa que, es necesario tomar lecturas de varias señales, como V, ϕ_1, ϕ_2 y E cuyos valores deberán restringirse al rango ±10 volts. Ahora bien, dado que el microcontrolador sólo acepta señales analógicas en el rango 0-5 volts, es necesario implementar algunos circuitos con base en amplificadores que lleven a cabo el mapeo de un rango hacia otro. Estos circuitos de amplificadores estan descritos por la siguiente ecuación:

$$v_o = 0.25v_i + 2.5 \qquad \text{[volts]} \qquad (4.11)$$

En total se alambraron cuatro amplificadores, uno para cada señal V, ϕ_1, ϕ_2 y E, siguiendo el modelo de la figura 4.5.

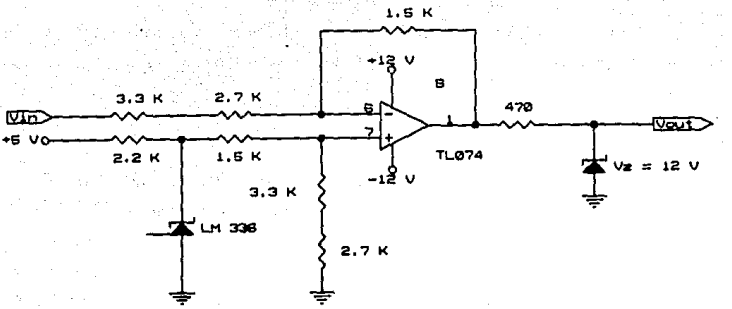
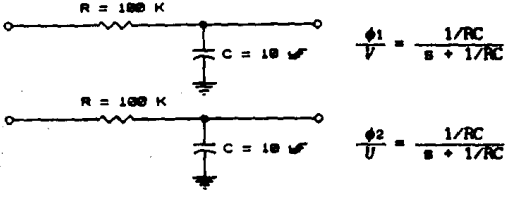


Fig. 4.5 Amplificador para realizar el mapeo entre los rangos (-10,+10) a (0,+5) volts.

Las señales ϕ_1 y ϕ_2 utilizadas por el algoritmo de identificación, se generan como resultado de filtrar a las señales V y U respectivamente (ver ecs. 3.23). Los dos filtros utilizados para las dos señales tienen las mismas caracteristicas, ambas son de primer orden y su polo se ubica en λ=-1 por facilidad. Estos filtros se implementaron en forma analógica de la siguiente manera:



$$\frac{\phi_1}{V} = \frac{1/RC}{s + 1/RC}$$

$$\frac{\phi_2}{U} = \frac{1/RC}{s + 1/RC}$$

Fig. 4.6 Filtros analógicos para obtener las señales ϕ_1 y ϕ_2.

En este caso los polos estan dados por λ=1/RC, y haciendo por facilidad λ=1, deberá tenerse que RC=1. Cumpliendo con lo anterior se proponen los siguientes valores:

R = 100 KΩ

C = 10 μF

Finalmente todos los circuitos descritos anteriormente se integran en una sola tarjeta a la que hemos denominado *tarjeta de interfaz* (Apénd. D).

IV.4 PROGRAMACION DEL MCU 68HC11

El programa implantado en el microcontrolador da la capacidad a este de ejecutar, en orden, cada una de las etapas de que consta el algoritmo bajo estudio. En el Anexo A se presenta el listado del programa fuente que cumple con dicho objetivo.

Como se describió en el capítulo II, la tarjeta evaluadora M68HC11EVB ofrece la ventaja de poder cargar programas desde una computadora hacia dicha tarjeta, pudiéndo escribirse los programas utilizando algún editor externo. El proceso que se sigue para este fin es el siguiente: primero, se lleva a cabo la escritura del programa con nemónicos utilizando un editor capaz de generar código ASCII (Pascal, wordstar, etc.), en seguida, este archivo (terminación *.S) es ensamblado a través del programa ensamblador PASM6C11, el archivo resultante de este programa (terminación *.LST) se liga utilizando el programa PLD y finalmente, el archivo resultante, es pasado por el programa UBUILDS que transcribe dicho archivo y genera una nueva versión de este (terminación *.MX) con un formato tal que es posible comunicar este último vía puerto serie desde la computadora hacia la tarjeta evaluadora M68HC11EVB.

Los programas PASM6C11, PLD, UBUILDS y otros son proporcionados en la paquetería de usuario proporcionada junto con la tarjeta evaluadora. El programa ensamblador para este microcontrolador ofrece ciertas ventajas, como el uso de nombres para las variables y etiquetas en lugar de valores de direcciones, permite hacer saltos hacia nombres de etiquetas con lo que se evita la molestia de calcular estos desplazamientos, permite definir cual será la dirección inicial de nuestro programa a partir de la cual se deberá empezar a escribir éste cuando sea transmitido desde la computadora, y además permite definir el espacio de memoria RAM que será empleado por las diversas variables.

El programa desarrollado, permite ejecutar cada una de las etapas del algoritmo: identificación, sintonización y control, y además tiene la capacidad de pasar en forma automática por cada una de ellas (fig. 4.7).

Previo a ésto, y para lograr este objetivo, se debe pasar por una *etapa de inicialización* donde se prepararán algunos elementos necesarios para una mejor claridad y buen funcionamiento del sistema, ésto incluye la declaración de variables y registros, la configuración del microcontrolador, así como la inicialización de las variables a emplear.

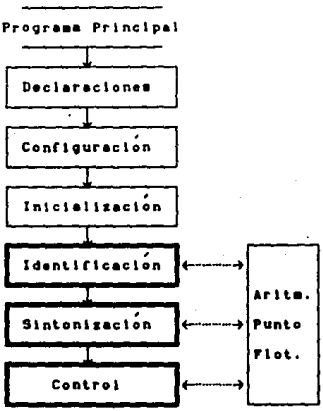


Fig. 4.7 Programa principal.

Una vez realizada la *etapa de inicialización* anterior, se pasa a la *etapa de identificación* donde se estiman los dos parámetros del motor θ_1 y θ_2. En seguida viene la *etapa de sintonización* del controlador, aquí, se calculan los parámetros K_p y K_d del controlador con base en los estimados de los parámetros antes obtenidos. Finalmente, se llega a la *etapa de control* donde se conecta el controlador PD en lazo cerrado con el motor para controlar su posición angular.

Con el objeto de evitar errores por redondeo o saturación se decidió usar un paquete de rutinas aritméticas en formato de punto flotante [11] que incluye las operaciones aritméticas básicas así como algunas otras rutinas para manejo de memoria y conversiones.

El programa se escribió en forma modular. A continuación se describen con más detalle cada una de las secciones antes mencionadas.

A. Declaración de variables y registros

Esta es una sección del programa fuente que nos permite asignar nombres a las variables y registros que se van a usar a lo largo de éste, dándole mayor claridad y mejor entendimiento. Debe tenerse presente que esta sección de declaraciones que se escribe al inicio del programa fuente, desaparecerá al momento de hacer el ensamblado del programa, ya que sólo son indicaciones para el programa ensamblador y no parte del programa.

Como se vió en el capitulo II los puertos, los registros de configuración, los registros del reloj y los registros para comunicaciones del microcontrolador tienen ya asignada ua dirección. El declarar estos registros consiste en darle nombre a esas direcciones para posteriormente usar sólo el nombre del registro y no su dirección, esto por supuesto, da más claridad al programa.

Entre los registros utilizados se encuentran:

a) Registros de configuración:

OPTION($1039), TMSK2($1024), ADCTL($1030).

b) Registros del reloj:

TCNTH($100E), TCNTL($100F), TOC5($101E), TFLG1($1023)

c) Registros de adquisición:

ADR1($1031), ADR2($1032), ADR3($1033), ADR4($1034)

d) Puerto de salida:

PORTB($1004)

Además de dar nombre a los registros del microcontrolador, también se da nombre a las localidades de memoria en RAM a partir de la localidad $0000 las que serán ocupadas por las variables o constantes que requiera el programa. Algunas de estas variables ocupan 1, 2, 3 y hasta 4 bytes. Entre las variables definidas se encuentran los contadores, las banderas y las máscaras que utilizan 1 y 2 bytes. Además se han asignado espacios de memoria para los dos acumuladores utilizados por las operaciones de punto flotante. Se definen también variables de 4 bytes de longitud. La razón de usar 4 bytes en estas variables, se debe a que en formato de punto flotante, éste es el número de bytes necesarios para almacenar un número en la memoria. Entre estas últimas destacan los parámetros K_p y K_d del controlador; los parámetros estimados del motor $\hat{\theta}_1$ y $\hat{\theta}_2$; el parámetro fijo K_2 también del motor; los valores de las muestras tomadas de las señales

v(t), e(t), $\phi_1(t)$ y $\phi_2(t)$; el valor de la señal de control u(t) generada dentro del microcontrolador; así como las constantes $h \cdot \gamma$, ξ y ω_n.

En seguida se presenta una tabla donde aparecen los nombres de algunas variables utilizadas en el programa, su dirección y las respectivas variables a que corresponden en el algoritmo:

VARIABLE	NOMBRE EN EL PROGRAMA	DIRECCION
e	E1	$0038
u	U	$003A
up	UP	$003E
ud	UD	$0042
v	VEL	$002A
ϕ_1	FI1	$002E
ϕ_2	FI2	$0032
$h \cdot \gamma$	HGAMA	$000A
ξ	PSI	$000E
ω_n	WN	$0012
K_2	K2	$0016
K_p	KP	$001A
K_d	KD	$001E
θ_1	TETA1	$0022
θ_2	TETA2	$0026
e_1	EIU	$0078
e_{1a}	EIM	$004A
e_{10} ~ e_{19}	EI0 ~ EI9	$004E ~ $0072

B. Configuración del microcontrolador

Esta es también una sección localizada al inicio del programa, con ella, se persigue indicar al microcontrolador, a través de sus diversos registros, la manera en que serán utilizados algunos de sus recursos. Configurar al microcontrolador consiste en cargar determinadas palabras de control en los registros provistos para tal fin. De esta manera, para indicar al microcontrolador que el puerto PE será utilizado como puerto de conversiones A/D y que el reloj utilizado para estas conversiones será el reloj del sistema E enviamos la palabra $90 hacia el registro OPTION. Para indicar que el factor para el prescalador del reloj utilizado en la rutina

de tiempo será 1 enviamos $03 al registro TMSK2.

C. Inicialización de variables

Esta etapa consiste en asignar valores a las variables antes declaradas. De esta manera, se limpian las variables que se usarán como contadores CONT, CONT1 y CONT3; se inicializan las banderas PRIMERA y YACONV; se asignan valores a las constantes AMPL, MAXERR, DELAY, NUMUNO, NUMDOS, K2, etc. y finalmente se dan condiciones iniciales a los estimados de los parámetros que intervienen en la ley de adaptación, en este caso son condiciones iniciales nulas, esto es:

$$\hat{\theta}_1(0)=0$$

$$\hat{\theta}_2(0)=0$$

D. Paquete de aritmética en punto flotante

Considerando que nuestro principal interés radica en el cálculo de un par de parámetros de los cuales no se tiene conocimiento a ciencia cierta, sobre su valor ni de la forma en que evolucionaran sus estados , y ya que además existen diversas variables con diferentes escalas, resulta conveniente utilizar el formato de punto flotante para representar a estas variables. Este formato nos permite manejar un amplio rango de valores, por lo que en principio se evitan los posibles problemas de saturación y errores por redondeo. Claro que todas estas ventajas implican un mayor consumo de tiempo, pero aún así, es posible alcanzar frecuencias de muestreo considerables, del orden de 1000 Hz ó inferiores dependiendo de la cantidad de operaciones que se efectúen por cada ciclo, en el caso de la etapa de identificación, la frecuencia se ajustó a 100 Hz, mientras que para la etapa de control, ésta se ajustó a 200 Hz.

Se ha incorporado al programa un paquete de rutinas que permiten ejecutar operaciones entre variables representadas en este formato. Este paquete incluye además de las cuatro operaciones básicas (suma, resta, multiplicación y división), una serie de funciones para realizar conversiones entre formatos ASCII a punto flotante y viceversa, punto flotante a entero y viceversa, incluye también funciones trigonométricas (sin, cos ,tan), raiz cuadrada, cambio de signo y funciones para leer o almacenar datos en la memoria, entre otras. El paquete completo ocupa un poco más de 2K bytes de memoria y requiere 10 bytes de RAM además del

STACK para operar. En nuestro programa hemos incluido tan sólo aquellas rutinas y funciones indispensables, e inclusive, se modificaron algunas con el objeto de adecuarlas más a nuestras necesidades.

Podría pensarse que incorporar este tipo de rutinas al programa lo haría más difícil de entender y más complejo, sin embargo, esto no es así, ya que, estas rutinas han sido diseñadas en cierta manera tal que resultan fáciles de usar y además dan claridad al programa. La forma de utilizar a muchas de ellas se traduce a tan sólo cargar los datos a operar en un par de variables a las que denominamos acumuladores y hacer la llamada a la rutina que ejecute la operación requerida. Para mayor detalle ver Apéndice C.

E. Identificación

En esta etapa se lleva a cabo la estimación de los dos parámetros del motor, $\hat{\theta}_1$ y $\hat{\theta}_2$. Esta etapa es una de las más importantes dentro del algoritmo, ya que de la calidad con que sean estimados dichos parámetros dependerá la calidad de la sintonización y por lo tanto el buen desempeño del controlador.

Esta etapa se compone de varias subrutinas que actúan conjuntamente dentro de un ciclo (fig. 4.8) que se repite hasta que se han alcanzado buenos estimados de los parámetros, esto último, es supervisado por una subrutina que prueba el cumplimiento de un cierto criterio de convergencia en cada ciclo. En seguida se decriben cada una de las subrutinas que integran esta etapa:

1.Generación de la señal de excitación

Esta subrutina de nombre SALCUAD, se encarga de generar una señal cuadrada alterna de una determinada frecuencia y amplitud, ambas características pueden variarse en el programa, y para demostración se ha propuesto una frecuencia de 2 Hz y una amplitud de ±3 volts. La señal es alimentada al motor a través del puerto B del microcontrolador y el convertidor D/A. Como se mencionó en el capítulo anterior, esta señal "rica" en frecuencia cumple con la función de asegurar excitación persistente en nuestro sistema.

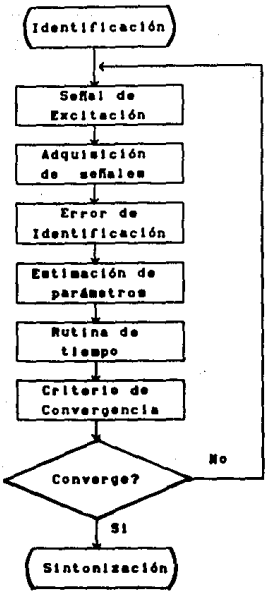


Fig 4.8 Etapa de identificación

2.Adquisición de señales

El propósito de esta subrutina (ENTIDEN) es tomar lecturas de las señales necesarias para el proceso de identificación, es decir, lee los valores de las señales V, ϕ_1 y ϕ_2, a través del puerto E usando sus terminales como canales de conversión A/D. Esta subrutina se auxilia de otras subrutinas, la primera, CONVER, realiza las conversiónes A/D de las muestras de las señales y deposita los valores resultantes en los registros ADR1 a ADR4, y por último la subrutina CARGA escala estos datos, los representarlos en formato de punto flotante y los envia a las variables VEL, FI1 y FI2 en la memoria.

3.Cálculo del error de identificación

Esta subrutina (CALEIDEN) está dedicada exclusivamente a evaluar el error de identificación descrito por la ecuación (3.38) y no es más que,

la traducción a lenguaje ensamblador de dicha ecuación. Hace uso de los valores de las señales antes procesados FI1, FI2 Y VEL, así como de los valores estimados previos de los parámetros TETA1 Y TETA2. Su resultado se deposita en la variable EIU.

4. Estimación de los parámetros

El objetivo de esta subrutina es actualizar los valores de los parámetros estimados TETA1 y TETA2, para ello resuelve recursivamente las ecuaciones (3.37a y b). Esta subrutina utiliza los valores procesados de las señales ϕ_1 y ϕ_2, (FI1 y FI2), el valor del error de identificación actualizado EIU (calculado en la subrutina anterior), el valor de la constante $h\gamma$ (HGAMA), así como los valores previos de los parámetros estimados TETA1 y TETA2. Una vez evaluadas dicha ecuaciones, el resultado de estas, que es el valor más reciente de $\hat{\theta}_1$ y $\hat{\theta}_2$, se deposita en las variables TETA1 y TETA2, para ser utilizadas en el siguiente ciclo.

Conviene resarcar que el valor de la constante HGAMA es proporcionado por el usuario al correr el programa, debe tenerse en cuenta además que, este valor en realidad reune a dos constantes una de ellas, el periodo de muestreo h, y la otra, la ganancia de adaptación γ. Para nuestro caso hemos fijado una frecuencia de muestreo de 100 Hz con lo que h=0.01 y además de las simulaciones se observó un buen comportamiento para valores de γ alrededor de 3, por lo que se propuso el valor 0.03125 para HGAMA, que en formato de punto flotante se representa:

HGAMA = $ 7C 00 00 00

5. Criterio de convergencia

El objetivo de esta subrutina (CRITERIO) es detener el proceso de identificación de acuerdo al criterio descrito en el capitulo anterior, donde se calcula el promedio denominado error de identificación medio con base en la ecuación (3.39). La subrutina obtiene este promedio de la siguiente manera: en los primeros diez ciclos, suma los diez primeros errores de identificación y en los siguientes ciclos sólo añade el último y resta el primero. Se declararon las variables EI0-EI9 para mantener los 10 valores últimos valores del error de identificación, además se realizó una subrutina (CORRETAB) que corre esta tabla de valores en cada ciclo a fin de mantenerla actualizada.

Finalmente, el error de identificación medio (EIM) es comparado contra un cierto límite previamente fijado MAXERR y si resulta que dicho

error EIM es inferior a ese límite, en ese momento se limpia la bandera YACONV, la cual indicará al programa que los parámetros ya han convergido a unos valores adecuados, deteniéndose así la identificación para proseguir con la siguiente etapa.

6. Rutina de tiempo

El propósito fundamental de esta subrutina, TMUEST, es introducir un determinado tiempo de retardo dentro del ciclo con el objeto de ajustar la frecuencia de muestreo del sistema. Esta subrutina utiliza el sistema de reloj provisto en el microcontrolador, del cual ya se ha hablado en el capitulo I, y funciona de la siguiente manera: se carga en el acumulador D el valor del contador de libre carrera contenido en los registros TCNTH y TCNTL, se añade el número de pasos que deseamos que cuente, el que ha sido previamente cargada en la variable DELAY, se almacena el resultado en el registro de comparación de salida del reloj TOC5 con el cual se habrá de estar comparando el contador de libre carrera en cada paso, y en el momento en que la cuenta de éste iguale al valor del registro TOC5 se encenderá una bandera en el registro de banderas TFLG1 lo cual nos indicará que se han contado los pasos requeridos (DELAY).

F. Sintonización

Esta etapa del programa está enfocada al cálculo de los parámetros del controlador K_P y K_d. Las expresiones para calcular dichos parámetros están descritas por las ecuaciones (3.14) y (3.15). Tales expresiones están en función de los valores estimados TETA1 y TETA2, así como de las especificaciones PSI y WN que se suministran, previamente codificadas en formato de punto flotante, antes de correr el programa. Los valores resultantes de los dos parámetros son almacenados en la memoria dentro de las variables KP y KD, los cuales serán utilizados en la etapa de control.

G. Control del motor

Esta es la etapa final del algoritmo, su propósito es realizar el control de la posición del motor de C.D.. Teniendo en cuenta que los parámetros del controlador ya han sido calculados en la etapa anterior, es factible ahora conectar dicho controlador en lazo cerrado con la planta a fin de cumplir con el objetivo de control.

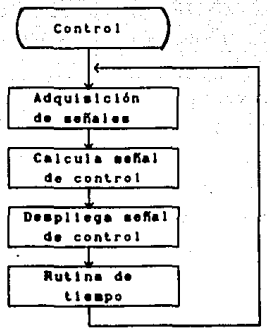


Fig. 4.9 Etapa de control.

Esta etapa al igual que la etapa de identificación está integrada por una serie de subrutinas que se ejecutan dentro de un ciclo (fig. 4.9). En este caso, el ciclo no se detiene a menos que abortemos el programa.

El proceso se concentra básicamente en calcular la señal de control que se alimentará al motor. Para esto, es necesario tomar muestras de las señales velocidad y error de posición del motor y con base en ellas calcular dicha señal de control conforme a la acción de control PD estudiada en la sección III.3 (fig. 3.6). En seguida se describen cada una de las subrutinas que conforman esta etapa del algoritmo.

1. Adquisición de señales

Esta subrutina actúa en forma similar a la subrutina descrita en la etapa de identificación, pero en este caso, sólo se lleva a cabo la lectura de las señales error de posición y velocidad, nuevamente aquí se realiza la conversión de ambas muestras a formato de punto flotante y se almacenan en la memoria dentro de las variables E1 y VEL respectivamente.

2. Calculo de señal de control

En realidad aquí se tienen contempladas tres subrutinas, la primera de ellas se dedica a calcular la parte proporcional de la señal de control, para esto, multiplica la señal de error E1 por la constante KP y el resultado lo almacena en la variable UP, otra rutina está dedicada a calcular la parte derivativa, para esto, multiplica la señal de velocidad

VEL por la constante KD y el resultado lo almacena en la variable UD, finalmente la tercer subrutina suma ambas partes y con ello obtiene la señal de control total, U, que se habrá de entregar al motor.

3. Despliegue de señal de control

Aquí se envia la señal de control hacia el puerto B en forma de señal digital de 8 bits, la cual será transformada a señal analógica a través del convertidor D/A. Dado que la señal de control U recien calculada está en formato punto flotante habrá que realizar la conversión que transfiera valores en este formato a un formato de 8 bits tal que sea posible entregar este resultado al puerto B. Lo anterior se realiza a través de la subrutina FLT2INT que pertenece al paquete de rutinas para manejar aritmética de punto flotante.

4. Subrutina de tiempo

Esta subrutina actúa en forma similar a la descrita en la etapa de identificación, sólo cambia el número de pasos indicados por la variable DELAY que se requiere contar.

Capítulo 5

Experimentos y resultados

V.1 INTRODUCCION

La realización de los experimentos que se plantean en esta sección así como la presentación de los resultados obtenidos comprueban la efectividad del algoritmo de identificación bajo diferentes condiciones de operación, específicamente ante variaciones en la carga. Se comprueba también que los estimados obtenidos como resultado de aplicar este algoritmo son muy cercanos a los valores reales y finalmente se evalua la capacidad del microcontrolador 68HC11 para resolver esta clase de problemas. Para la comprobación del primer punto, se analizaron los resultados obtenidos tras realizar diversos experimentos sobre la etapa de identificación, en el caso del segundo punto, se realizó también un análisis pero en este caso de resultados generados en la etapa de control.

La variación en la carga se realiza aplicando el freno magnético que viene incluido en el kit del servomecanismo descrito en la sección IV.2, donde se establece que la carga aumentará gradualmente conforme se incrementa el porcentaje de superficie de interacción entre el freno magnético y el disco de aluminio acoplado a la flecha del motor.

Al someter a la flecha del motor a diferentes cargas se está en posibilidad de variar los parámetros de su modelo, y se espera que el algoritmo sea capaz de identificar el nuevo valor de éstos.

Para obtener las gráficas aquí presentadas se utilizó la tarjeta de adquisición de señales LabMaster [12] conectada a una PC. En esta tarjeta se dispone de hasta 16 canales de conversión A/D con un rango de entrada ±10 volts. Los datos registrados por la tarjeta son almacenados en archivos que, posteriormente se despliegan en forma de gráficas a través

del paquete de graficación Grapher.

V.2 ETAPA DE IDENTIFICACION

Los resultados aquí presentados muestran la convergencia de los parámetros a ciertos valores, la trayectoria que siguen para llegar a su valor final y la forma en que afectan los cambios de las características de la planta en dichos resultados. Específicamente se muestra como dichos parámetros convergen a diferentes valores para diferentes cargas (posiciones del freno magnético).

En todos los experimentos que se presentan aquí se utiliza el valor 0.03125 para la constante $h \cdot \gamma$. Su codificación en formato de punto flotante es:

HGAMA = $ 7C 00 00 00

1.Convergencia del algoritmo

El objetivo del primer experimento es obtener los valores finales de $\hat{\theta}_1$ y $\hat{\theta}_2$ para las diferentes posiciones del freno (0, 25, 50, 75%). Para tal fin se ejecuta el programa para cada una de estas posiciones en donde el criterio de convergencia se cumple. A partir de los estimados obtenidos se calculan K_p y K_d del controlador. Para la sintonización se proponen las siguientes especificaciones:

$\xi = 0.4$

$\omega_n = 20$ rad/seg

que codificadas en formato de punto flotante dan:

PSI = $ 7F 48 00 00

WN = $ 85 20 00 00

Estos valores, junto con el de HGAMA, se alimentan en las localidades correspondientes antes de correr el programa.

Recuérdese que los valores de $\hat{\theta}_1$, $\hat{\theta}_2$, K_p y K_d están almacenados en localidades de la memoria RAM del microcontrolador, ocupando cada uno de ellos 4 bytes de espacio, por lo que para obtener dichos valores basta con observar su contenido tras haber corrido el programa.

Del experimento anterior se obtuvo la siguiente tabla de valores:

%	$\hat{\theta}_1$	$\hat{\theta}_2$	$\hat{a}$	$\hat{K}_1$	K_p	K_d
0	-4.9556	11.2944	5.9556	11.2944	4.5563	0.8893
25	-4.0690	8.4768	5.0690	8.4768	6.0707	1.2895
50	-3.8672	8.1224	4.8672	8.1224	6.3356	1.3706
75	-3.1075	7.2426	4.1075	7.2426	7.1052	1.6420

Tabla 5.1 Valores finales para $\hat{\theta}_1$, $\hat{\theta}_2$, $\hat{a}$, $\hat{K}_1$, K_p y K_d.

de donde los valores de $\hat{a}$ y $\hat{K}_1$ se calculan usando las ecuaciones (3.34a) y (3.34b) a partir de los valores de $\hat{\theta}_1$ y $\hat{\theta}_2$.

Como se observa, conforme se incrementa la carga, los parámetros $\hat{a}$ y $\hat{K}_1$ se reducen. La explicación a este comportamiento se da a continuación:

De las ecuaciones (3.2) y (3.1) para el modelo del motor sabemos que:

$$K = \frac{K_a}{\tau_a} = \frac{K}{R_a J} \qquad (4.12)$$

$$a = \frac{1}{\tau_a} = \frac{R_a F + K K_b}{R_a J} \qquad (4.13)$$

La acción del freno magnético sobre este modelo ocaciona que el valor de la inercia J se incremente y dado que los parámetros K y a varian en forma inversa a esta inercia, ambos decrecen a medida que se incrementa la carga.

Nótese también de la tabla anterior que a medida que la perturbación de par crece también aumenta el valor de las constantes K_p y K_d del controlador. Revisando el análisis en estado estable del capitulo III, encontramos que el error de estado estable es inversamente proporcional al valor de la constante K_p, de donde esta variable tiende a decrecer para cada experimento.

2.Evolución de los parámetros

El siguiente experimento consistió en registrar la evolución de los parámetros $\hat{\theta}_1$ y $\hat{\theta}_2$ durante un periodo de 50 segundos, tiempo suficiente para permitir que los valores de los estimados se establezcan. Para cumplir con este objetivo, se modificó el programa de tal forma que se pudiera desplegar por el puerto B de salida y a través del convertidor D/A el valor del parámetro $\hat{\theta}_1$ ó $\hat{\theta}_2$ conforme este se iba calculando dentro del

microcontrolador.

La evolución de los parámetros $\hat{\theta}_1$ y $\hat{\theta}_2$ se obtuvo para las mismas posiciones del freno, sus gráficas se presentan a continuación:

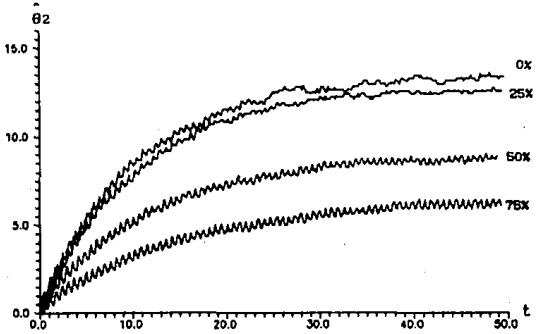


Fig 5.1 $\hat{\theta}_2(t)$ para diversas posiciones del freno.

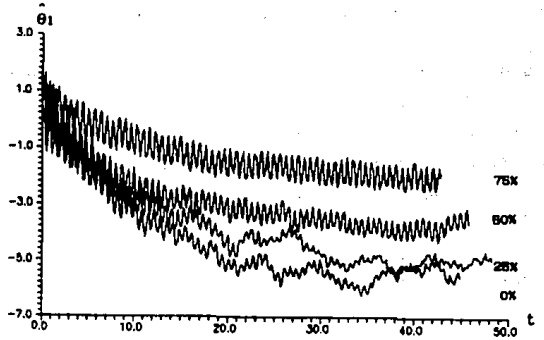


Fig 5.2 $\hat{\theta}_1(t)$ para diversas posiciones del freno.

Nótese de las curvas anteriores que el parámetro $\hat{\theta}_2$ excede algunas veces al valor +10, en estos casos, fué necesario escalar dicho valor antes de desplegarlo a fin de no saturar el convertidor D/A que no puede entregar voltajes superiores a +10 volts, aunque posteriormente se revertió dicha modificación para obtener la gráfica real.

3. Error de identificación

Como se explicó en el capitulo anterior, el algoritmo de identificación procura llevar al error de identificación a un valor mínimo a la vez que lo utiliza para estimar el valor de los parámetros θ_1 y θ_2. El siguiente experimento consistió en desplegar los valores que va tomando el error de identificación, que se calcula dentro del microcontrolador, a través del arreglo puerto B - convertidor D/A. En la figura 5.3 se presenta la gráfica resultante, en ella se puede apreciar la característica decreciente del error de identificación.

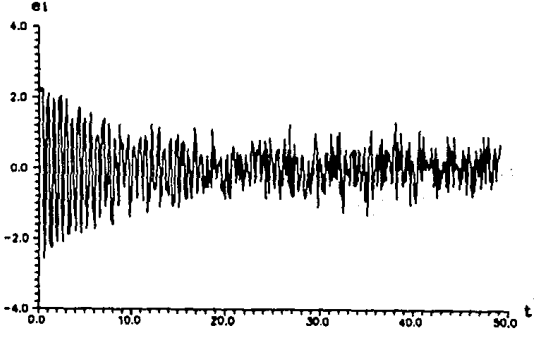


Fig. 5.3 Error de identificación.

4. Respuesta a perturbaciones

Un experimento interesante consistió en observar la forma en que el algoritmo reacciona ante la presencia de perturbaciones en el periodo de tiempo en que realiza la identificación. Estas perturbaciones fueron básicamente cambios bruscos en la magnitud de la carga y por lo tanto en

los valores de los parámetros del motor.

El experimento se desarrollo de la siguiente manera: Se colocó el freno magnético al motor en la posición 75%, se corrió el programa modificado tal que permitiera desplegar a través del arreglo D/A el valor de los parámetros $\hat{\theta}_1$ y $\hat{\theta}_2$ (uno por cada corrida) y en el tiempo t_1 después de haber iniciado la identificación se cambió el freno magnético a la posición 0%. Las gráficas resultantes se presentan a continuación:

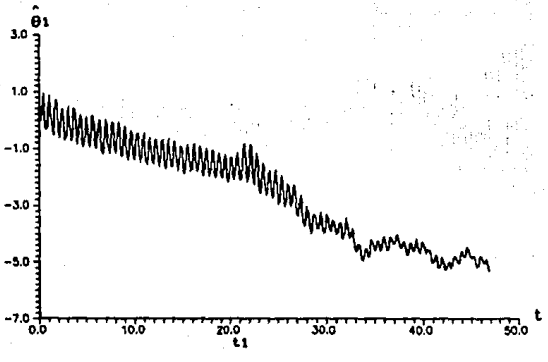


Fig. 5.4 $\hat{\theta}_1(t)$ **con perturbación en** t_1**=21 seg.**

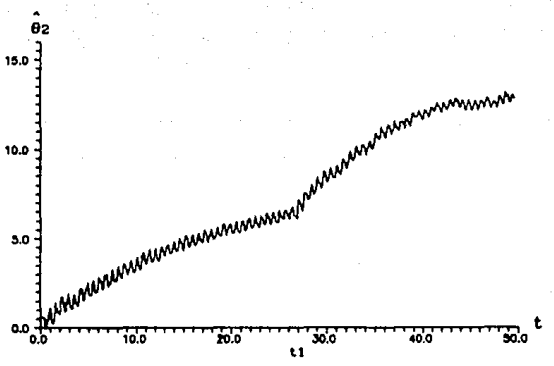


Fig. 5.5 $\hat{\theta}_2(t)$ con perturbación en t_1=27 seg.

De las gráficas anteriores se observa que efectivamente el algoritmo asimila las variaciones en las características de la planta y hace tender a los parámetros a sus nuevos valores. Nótese por ejemplo en la gráfica de la figura 5.5 como el parámetro $\hat{\theta}_2$ ya casi se ha establecido en el tiempo t=20 seg. y que al aplicar la perturbación en el tiempo t= 27 seg. el algoritmo hace que el valor de este parámetro cambie y se acerque a uno nuevo, algo similar sucede con el parámetro $\hat{\theta}_1$. Nótese además que los cambios en los parámetros en estas gráficas están acordes con los resultados de los dos experimentos anteriores, ésto es, si el porcentaje del freno se decrementa, el valor en los parámetros $\hat{\theta}_1$ y $\hat{\theta}_2$ se eleva en valor absoluto.

V.3 ETAPA DE CONTROL

El objetivo final de nuestro algoritmo, al cual está concentrada su última etapa, consiste en realizar el control de la posición angular de la flecha del motor usando el controlador PD, que se sintoniza con base en los parámetros estimados del motor. Se proponen básicamente dos señales de referencia: una señal tren de pulsos y una señal senoidal ambas de baja

frecuencia. Se demuestra que aún para cambios en los parámetros de la planta, es decir, para cada posición del freno magnético, la respuesta del sistema se mantiene con la misma forma, la cual ha sido establecida a través de las especificaciones ξ y ω_n alimentadas al programa.

1.Respuesta a tren de pulsos

Primeramente se propuso como señal de referencia un tren de pulsos a muy baja frecuencia 0.1 Hz y de amplitud 2 volts (fig. 5.6).

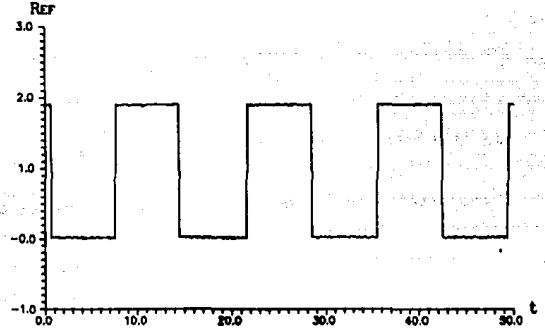


Fig 5.6 Señal de referencia tren de pulsos.

La razón de usar este valor para la amplitud es debido a que la señal de error que se genera es pequeña, así, la señal de control no se sale del rango ±10.0 con lo cual se evita la saturación del convertidor D/A.

Para sintonizar al controlador se proponen como primer caso, las siguientes especificaciones:

M_p = 5%

t_s = 0.25 seg

de donde:

ξ = 0.6901

ω_n = 16.664 rad/seg

que codificados en formato de punto flotante dan:

PSI = $ 80 30 AA C0

WN = $ 85 05 4F DF

Estos dos valores junto con el de HGAMA ($h \cdot \gamma$ = 0.03125) son alimentados a las memorias correspondientes antes de correr el programa.

El resultado de este experimento es que la respuesta del sistema es muy parecida a la señal de referencia anterior y además se mantiene la misma forma aún en presencia de variaciones en los parámetros de la planta. Para este efecto se tomaron lecturas de la señal de posición (Y), de error (E) y de control (U) para diferentes cargas (0, 25, 50 y 75% de freno), que se muestran a continuación:

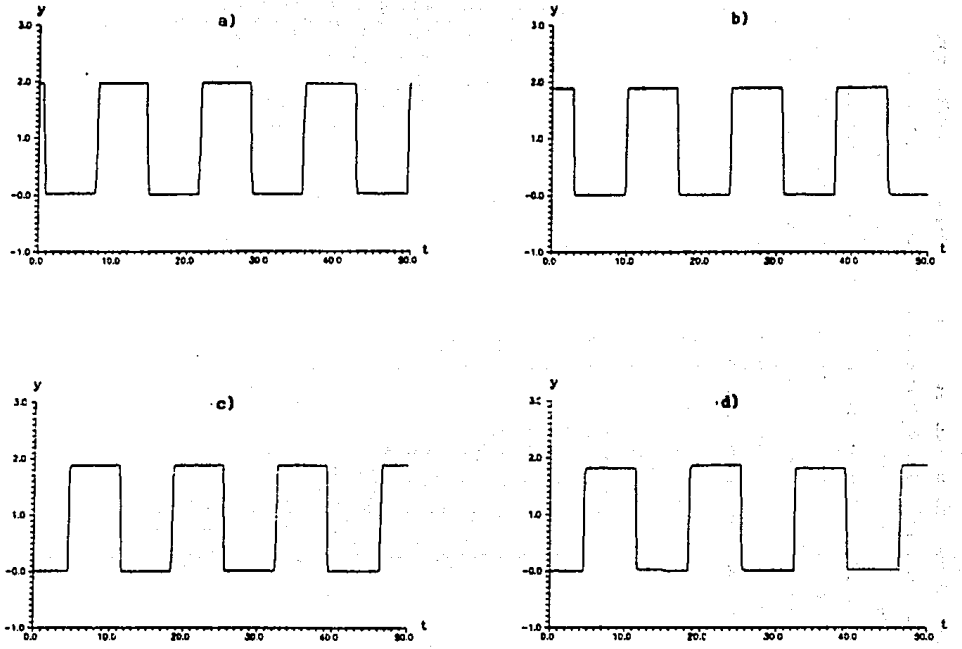


Fig 5.7 Señales de posición (Y) para freno al:
a)0%, b)25%, c)50% y d)75%

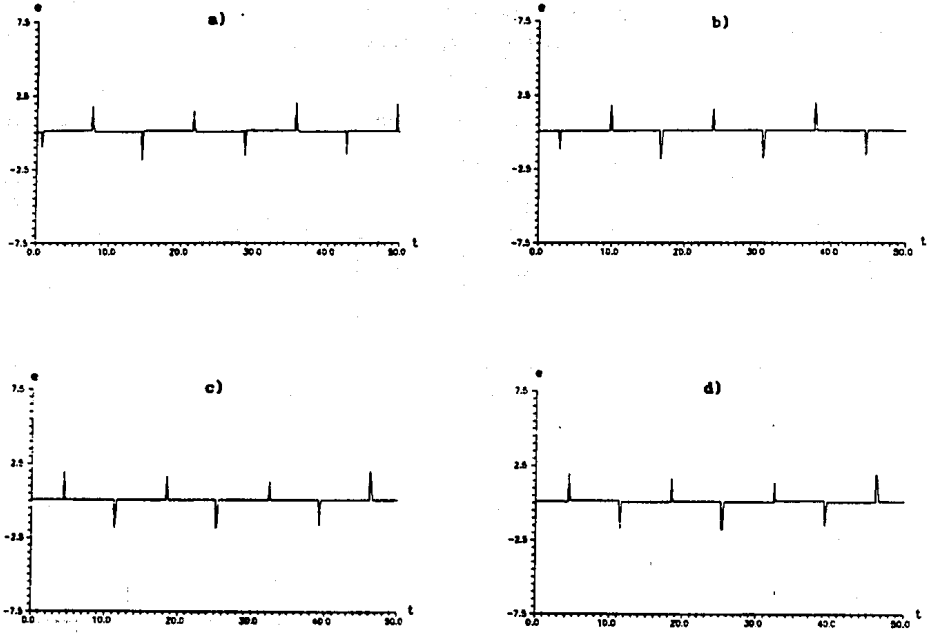


Fig 5.8 Señales de error (E) para freno al:
a)0%, b)25%, c)50% y d)75%

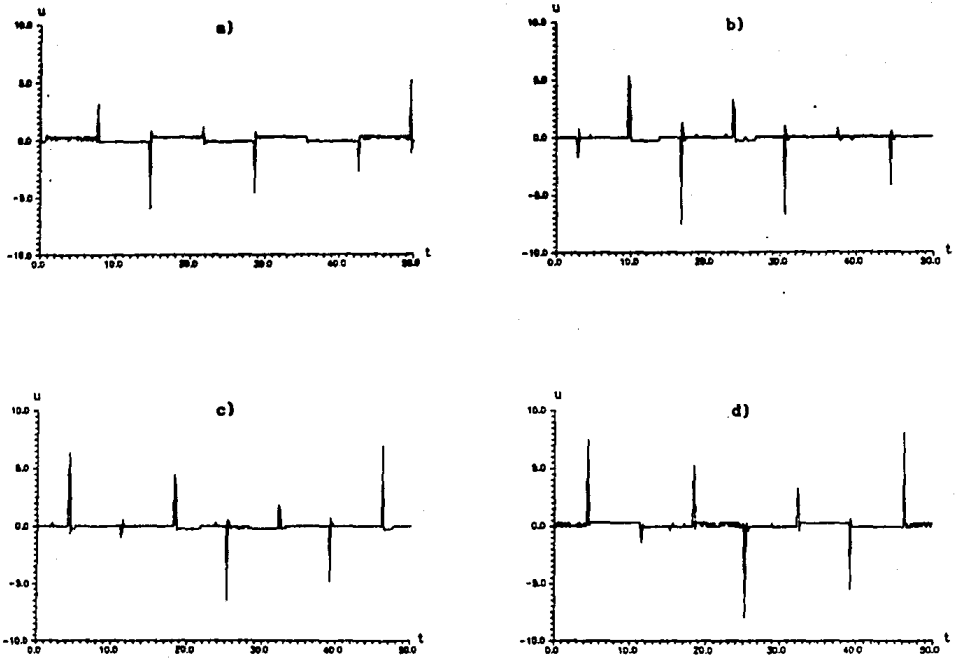
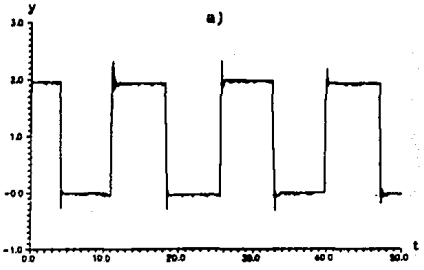
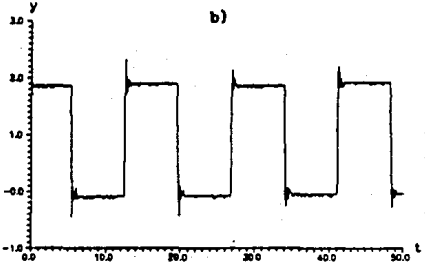


Fig 5.9 Señales de control (U) para freno al:
a)0%, b)25%, c)50% y d)75%





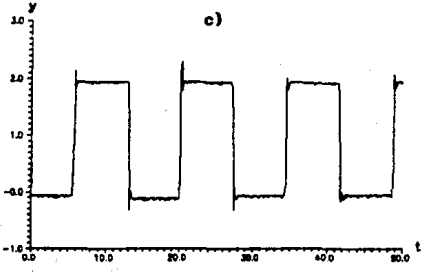


Fig. 5.10 Señales de posición (Y) para freno al:
a)0%, b)25% y c)50%

De las gráficas anteriores podemos notar lo siguiente:

Las señales de respuesta (Y) del sistema son similares entre si y con la señal de referencia, lo que indica que el controlador ayuda a compensar los cambios que se presentan en los parámetros de la planta. Nótese también la similitud en forma y dimensiones de las señales de error (E), esto resulta como consecuencia de la similitud entre las señales de salida (Y).

Se observa que las señales de control (U) si cambian en su magnitud al presentarse cambios en los parámetros de la planta, ésto es de esperarse puesto que un cambio en los parámetros de la planta viene acompañado de un cambio en los parámetros del controlador el cual ayuda a compensar los cambios en la planta. Se observa que, a medida que se incrementa la carga, la magnitud de la señal de control (U) también se incrementa. Incrementar la carga sobre la flecha del motor exige un mayor esfuerzo por parte del sistema y ésto se traduce en elevar la magnitud de la señal que habrá de alimentarse a la planta, es decir, la señal de control U.

Se realizó otra prueba en la que nuevamente se tomaron lecturas para la señal de salida (Y) del sistema bajo diferentes cargas (0, 25 y 50% de freno) y para el mismo tipo de señal de referencia solo que ahora con otros valores para las especificaciones M_p y t_s, y por lo tanto para ξ y ω_n (fig. 5.10). Los valores de las especificaciones propuestas en este caso son:

$\xi = 0.4$

$\omega_n = 20$ rad/seg

que codificados en formato de punto flotante dan:

PSI = $ 7F 48 00 00

WN = $ 85 20 00 00

donde, estas especificaciones corresponden a:

$M_p = 31$ %

$t_s = 0.5$ seg

Note nuevamente que existe una cierta similitud en la forma de estas gráficas en donde en donde el sobrepaso M_p es cercano al especificado.

Por último se efectuaron un par de pruebas en las que se tomaron lecturas para la señal de salida bajo ciertas especificaciones en M_p y t_s y para una determinada carga (freno al 0%). Las figuras 5.11 y 5.12 fueron obtenidas con la ayuda de un graficador analógico y en ellas se pueden apreciar con más detalle dichas características (M_p y t_s) en la respuesta del motor.

Para el primer caso se propusieron las siguientes especificaciones:

M_p = 15 %

t_s = 0.25 s

y experimentalmente se obtubieron los siguientes resultados:

M_p = 16 %

t_s = 0.36 s

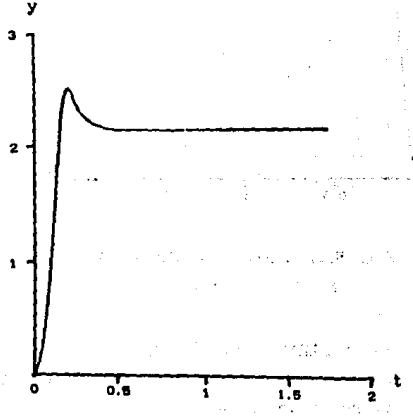


Fig. 5.11 Señal de posición (Y)
M_p= 16% , t_s=0.36 s

Para el segundo caso se propusieron las siguientes especificaciones:

M_p = 30%

t_s = 1 s

obteniéndose experimentalmente:

M_p = 23.42%

t_s = 0.508 s

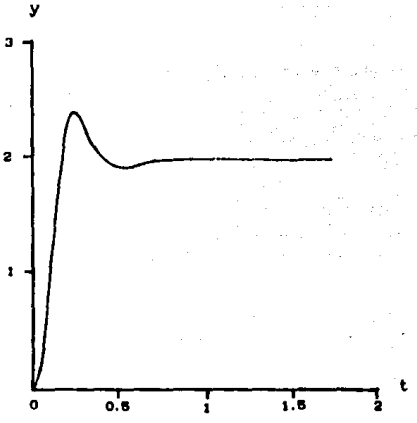


Fig. 5.12 Señal de posición (Y)
M_p=23.42% , t_s=0.508

2.Respuesta a señal senoidal: Seguimiento

El objeto de esta prueba es observar el desempeño del sistema cuando se requiere seguir una señal variante en el tiempo. En este caso, se aplicó la señal de referencia senoidal de la figura 5.13, descrita por la siguiente ecuación:

$$r = 2 \cdot \text{sen}(0.4\pi t) \text{ [volts]}$$

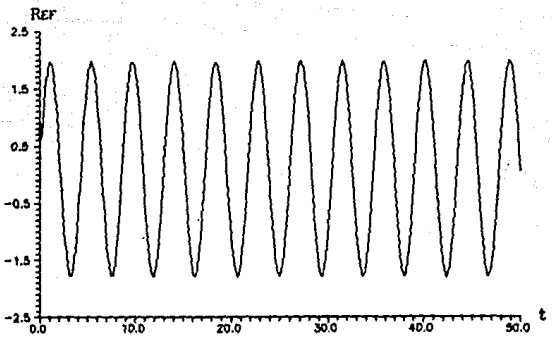


Fig. 5.13 Señal de referencia senoidal.

Se tomaron lecturas para las señales posición de salida (Y), error (E) y control (U). Las gráficas resultantes se presentan a continuación:

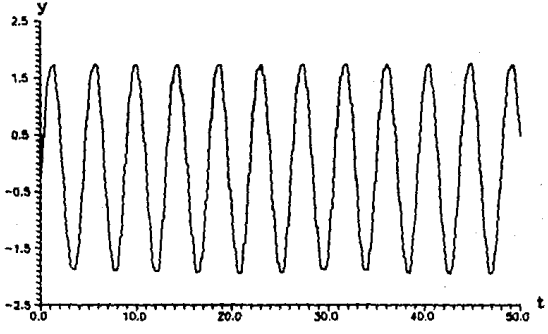


Fig. 5.14 Señal de posición de salida (Y).

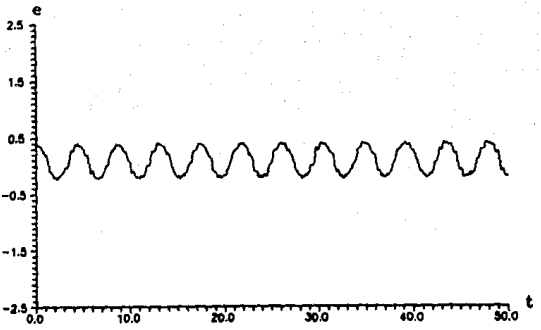


Fig. 5.15 Señal de error (E).

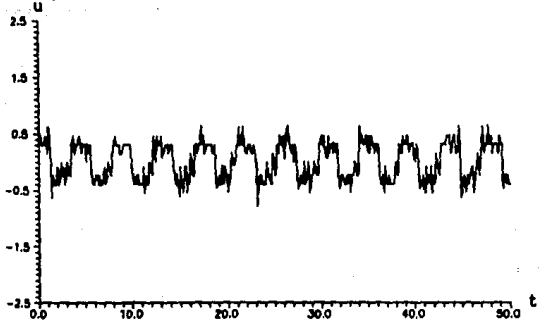


Fig. 5.16 Señal de control (U).

De las gráficas anteriores notamos que la respuesta del sistema es de forma senoidal, de menor magnitud y con un cierto defasamiento, resultado que se espera del análisis del diagrama de bode del sistema [1]. Se

observa además que la señal de error también es de forma senoidal y de una magnitud pequeña pero de tomarse en cuenta. Nótese también que, aunque el error cruza el punto cero, su tendencia es seguir oscilando alrededor de este punto en forma sostenida. De ésto último, podemos concluir que, el sistema bajo estudio presenta problemas cuando se trata de realizar el seguimiento de una señal variante en el tiempo, mismos resultados se presentan en [13], donde a este controlador se le refiere como "controlador proporcional", también se muestran algunas de sus caracteristicas y se recomienda su uso en el control de posición puro, esto es, para posiciones angulares deseadas constantes.

Capítulo 6

Conclusiones

En este trabajo se presenta la implementación de un controlador PD autosintonizable para realizar el control de posición de un motor de C.D. con base en el microcontrolador MC68HC11 fabricado por Motorola.

El algoritmo de control incluye las etapas: identificación, sintonización y control, que son ejecutadas secuencialmente.

Para dar la característica autosintonizable al controlador propuesto se utilizó un algoritmo de identificación el cuál nos proporciona, dada una estructura previamente establecida, la información acerca de los parámetros de la planta.

Este algoritmo no precisa del conocimiento previo de los parámetros de la planta, de hecho las condiciones iniciales para los estimados de éstos siempre se supusieron nulas, sin embargo es claro que el proceso de identificación se aceleraría si se supusieran condiciones iniciales cercanas al valor de convergencia.

El algoritmo de identificación utilizado mostró la capacidad de asimilar los cambios que pudieran presentarse en las características de la planta conduciendo a los estimados de los parámetros de ésta hacia un valor adecuado.

Al realizar la identificación, el algoritmo toma alrededor de 15 seg. para conducir los parámetros a un valor aceptable conforme al criterio de convergencia establecido. Mejores parámetros estimados se podrían alcanzar si se hiciera más estricto dicho criterio, pero ésto implicaría un mayor consumo de tiempo.

El algoritmo estudiado puede ser utilizado en conjunto con otras estructuras de control que al igual que el PD propuesto, requieren para su sintonización del conocimiento de la planta o proceso a controlar.

El hecho de haber elegido el método de sintonización propuesto, se debe a que las fórmulas que éste emplea para el cálculo de los parámetros son sencillas y requieren de un mínimo de operaciones para su evaluación, ésto es de gran importancia sobre todo en casos donde se está restringido a usar unicamente operaciones aritméticas básicas. Note que sólo se pone atención a la forma que deberá tener la respuesta transitoria del sistema, aunque también se podría pensar en sintonizar al controlador considerando especificaciones del error de estado estable.

Se observó, de los resultados, que las respuestas en el tiempo del sistema eran similares para diferentes eventos en los que se planteaban las mismas especificaciones (ξ y ω_n) pero diferentes características en el motor (diferentes cargas), lo cuál reafirma la característica adaptable del algoritmo.

Notamos también de los resultados experimentales y de simulación que, existe una cierta desviación en cuanto a las especificaciones requeridas en la respuesta del sistema (M_p y t_s), esto se explica debido a que la forma propuesta para sintonizar los parámetros es en sí, solo una aproximación. Como se describió anteriormente, los parámetros del controlador PD son calculados con fórmulas derivadas de un análisis en el espacio continuo, más aún si realizamos un análisis del sistema en el espacio discreto observaremos la prescencia de un cero en la función de transferencia del sistema discretizado, el cuál, dependiendo de su posición relativa a los polos del sistema, provocará un cambio en la respuesta de este.

A pesar de la sencillez del controlador PD propuesto, se obtiene una respuesta aceptable del sistema, además de que resulta fácil su implementación en el microcontrolador ya que sólo se requieren 3 operaciones aritméticas para ello. Para el caso en que el motor solo contara con el sensor de posición, la parte derivativa del controlador se podría obtener aproximando el cálculo de la derivada de la señal de error, lo que no complicaría en mucho al controlador.

El sistema con control PD aquí presentado tiene buen comportamiento cuando son manejadas señales de referencia constantes, no sin embargo cuando se trata de seguir señales variantes en el tiempo donde aparece un error de posición considerable, por lo que no sería recomendable su uso en

procesos donde la precisión juega un papel importante, como es el caso de los robots.

A futuro se plantea la posibilidad de reunir a las tres etapas en un solo ciclo de tal manera que se ejecuten a un mismo tiempo. Esto daría como resultado una estructura de *"control adaptable"*, que en principio evitaría el uso de las rutinas que prueban el criterio de convergencia y de las que generan la señal de excitación, además se debe tener presente que ahora la señal de referencia deberá ser rica para que de esta manera se asegure excitación persistente en el sistema.

En el microcontrolador MC68HC11 fué factible la implementación de las tres etapas, y además fué posible alcanzar frecuencias de muestreo adecuadas para nuestro propósito a pesar de haber utilizado formato de punto flotante para representar y operar a las variables, lo cuál implica un mayor consumo de tiempo.

Para los casos de estructuras de control más complejas podría pensarse en usar también formato de punto flotante en donde se recomendaría reducir el tamaño de la mantisa de 3 bytes a 2 bytes en las variables, con los respectivos cambios en las subrutinas que las operan, y en el caso extremo utilizar formato de punto fijo.

La implementación de algoritmos como el aquí presentado e incluso más complejos puede realizarse también con base en otros dispositivos como el DSP 56000, con el que se planea continuar experimentando a fin de seguir avanzando sobre este trabajo.

Referencias

[1]. Ogata, Katsuhiko. *Ingeniería de Control Moderna*. Prentice Hall, 1980 México.

[2]. Feedback Instruments Limited. *Servo Modular Tipo* MS150. FI Ltd, 1983 England.

[3]. Åström, Karl J. & Wittenmark, Björn. *Computer Controlled Systems. Theory & Design*. Prentice Hall, 1984 Englewood Cliffs.

[4]. Åström, K.J. & Wittenmark, Björn. *Adaptive Control*. Adison Wesley, 1989.

[5]. Sastry, S., and M. Bodson. *Adaptive Control: Stability, Convergence and Robustness*. Prentice Hall. Englewood Cliffs, NJ. 1988.

[6]. Elmqvist Hilding, K. Åström & T. Schönthal. *Simnon. User's Guide for MS-DOS Computers Versión* 1.0 . 1986. Sweden.

[7]. Motorola Inc. MC68HC11A8 *Advance Information. HCMOS Single-Chip Microcontroller*. 1988. USA.

[8]. Motorola Inc. M68HC11EVB *Evaluation Board User's Manual*. 1986. USA.

[9]. Franklin Gene F. & J. David Powell. *Digital Control of Dinamic Systems*. 1980.

[10]. Motorola Inc. *Linear/interface integrated circuits*. 1983. USA.

[11]. Motorola Inc. *Semiconductor Application note* MC68HC11 *Floating Point Package*. 1987. USA.

[12]. Scientific Solutions Inc. *Lab-Master. Installation Manual, User's Guide*. 1985. Ohio.

[13]. Kelly Rafael. *Control de Movimiento de Robots Manipuladores*. DEPFI UNAM. 1989.

Apéndice A

Programa para 68HC11

```
*****************************************************
*PROGRAMA PARA MC68HC11 QUE REALIZA:
*     IDENTIFICACION: LEY GRADIENTE
*     SINTONIZACION: PARAMETROS KP Y KD PARA PD
*     CONTROL: POSICION DEL SERVOMOTOR DE CD MS-150
*ARITMETICA CON FORMATO DE PUNTO FLOTANTE
*PUERTO B PARA A/D
*CANALES DE ENTRADA D/A:
*     PE4----VELOCIDAD
*     PE5----POSICION
*     PE6----FI 1 (VELOCIDAD FILTRADA)
*     PE7----FI 2 (U FILTRADA)
*****************************************************
*
*
*****************************************************
*DECLARACION DE REGISTROS
*****************************************************
*
*
*REGISTROS DE CONFIGURACION
*****************************************************
OPTION          EQU             $1039
ADCTL           EQU             $1030
SCCR2           EQU             $102D
TMSK2           EQU             $1024
*
*
*REGISTROS DEL TIMER
*****************************************************
TCNTH           EQU             $100E
TCNTL           EQU             $100F
TOC5            EQU             $101E
TFLG1           EQU             $1023
*
*
*REGISTROS DE ADQUICISION
*****************************************************
ADR1            EQU             $1031
```

```
ADR2            EQU             $1032
ADR3            EQU             $1033
ADR4            EQU             $1034
*
*
*PUERTO DE SALIDA
*****************************************************
PORTB           EQU             $1004
*
*
                ORG             $0000
*****************************************************
*DECLARACION DE VARIABLES
*****************************************************
*
*
*ESPACIO DE MEMORIA PARA LOS ACUMULADORES
*****************************************************
FPACC1EX        RMB             1               EXPONENTE DEL ACUM1
FPACC1MN        RMB             3               MANTISA DEL ACUM1
MANTSGN1        RMB             1               SIGNO DEL ACUM1
FPACC2EX        RMB             1               EXPONENTE DEL ACUM2
FPACC2MN        RMB             3               MANTISA DEL ACUM2
MANTSGN2        RMB             1               SIGNO DEL ACUM2
*
*
*VARIABLES EN RAM DE 32 BITS
*****************************************************
HGAMA           RMB             4
PSI             RMB             4
WN              RMB             4
K2              RMB             4
KP              RMB             4
KD              RMB             4
TETA1           RMB             4
TETA2           RMB             4
VEL             RMB             4
FI1             RMB             4
FI2             RMB             4
E1              RMB             4
U               RMB             4
UP              RMB             4
UD              RMB             4
AUX             RMB             4
EIM             RMB             4
EI0             RMB             4
EI1             RMB             4
EI2             RMB             4
EI3             RMB             4
EI4             RMB             4
EI5             RMB             4
EI6             RMB             4
EI7             RMB             4
EI8             RMB             4
```

```
EI9             RMB             4
EIU             RMB             4
NUMUNO          RMB             4
NUMDOS          RMB             4
*
*
*VARIABLES EN RAM DE 8 BITS
*****************************************************
AMPL            RMB             1
MAXERR          RMB             1
DIVFREC         RMB             1
CONT            RMB             1
CONT1           RMB             1
CONTT           RMB             1
DELAY           RMB             2
PUNTLOC         RMB             1
CONT3           RMB             1
RETAR           RMB             1
PRIMERA         RMB             1
REPMIN          RMB             1
YACONV          RMB             1
*
*
*****************************************************
*CONFIGURACION DEL SISTEMA
*****************************************************
                ORG             $C000
MPROG           LDAA            #$03
                STAA            TMSK2           PRESCALADOR
                LDAA            #$90
                STAA            OPTION          HABILITACION DE A/D
                LDS             #$FF            INICIALIZA STACK
*
*
*****************************************************
*PROGRAMA PRINCIPAL
*****************************************************
*
*
*****************************************************
*INICIALIZACION DE VARIABLES
*****************************************************
                LDAA            #$A6
                STAA            AMPL            AMPL=3
                LDAA            #$81
                STAA            MAXERR          MAXERR IDEN=4
                LDAA            #$19
                STAA            DIVFREC         FREC/25 ~ 4 HZ
                LDD             #$1900
                STD             DELAY
                LDAA            #$86
                STAA            PUNTLOC
                LDAA            #$14
                STAA            RETAR
```

```
                LDAA        #$01
                STAA        REPMIN
                LDAA        #$01
                STAA        YACONV
                CLR         CONT
                CLR         CONT1
                CLR         EIM                 SE LIMPIA TABLA DE EI
                CLR         EI0
                CLR         EI1
                CLR         EI2
                CLR         EI3
                CLR         EI4
                CLR         EI5
                CLR         EI6
                CLR         EI7
                CLR         EI8
                CLR         EI9
                CLR         CONT3
                CLR         PRIMERA
                LDD         #$0000
                STD         NUMUNO+2
                STD         NUMDOS+2
                LDD         #$8100
                STD         NUMUNO
                LDD         #$8200
                STD         NUMDOS
                LDD         #$8378
                STD         K2
                LDD         #$BC8A
                STD         K2+2                K2:=7.773
CI              LDX         #TETA1
                JSR         CLEAR               TETA1(0)=0
                LDX         #TETA2
                JSR         CLEAR               TETA2(0)=0
*
*****************************************************
INICIO          JSR         SALCUAD             GENERA SEÑAL CUADRADA
                JSR         ENTIDEN             LECTURA DE DATOS
                JSR         CALEIDEN            CALCULA ERROR DE IDENT.
                JSR         IDENPAR             ESTIMA PARAMETROS
                JSR         THUEST              AJUSTA FREC. MUESTREO
                JSR         CRITERIO            YACONV=1 :YA CONVERGIO
                LDAA        YACONV
                BNE         INICIO              YACONV=0 :LOOP
                JSR         SINTONI             SINTONIZA CONTROLADOR
                JSR         CONTPD              CONTROLA MOTOR
*
*
*****************************************************
*SUBRUTINA PARA GENERAR SEÑAL CUADRADA
*****************************************************
SALCUAD         LDAA        CONT
                INCA
                STAA        CONT
```

```
                LDAB            AMPL                    B:=AMPLITUD
                EORA            DIVFREC                 FRECUENCIA 4HZ/2 = 2HZ
                BNE             SAL1
                COMB                                    B:=-AMPLITUD
                STAB            AMPL
                CLR             CONT
SAL1            STAB            PORTB                   B=>PUERTO B
                RTS
*
*
*****************************************************
*SUBRUTINA DE ADQUISICION DE SE&ALES (IDENTIF.)
*****************************************************
ENTIDEN         JSR             CONVER
                LDX             #ADR1
                LDY             #VEL
                JSR             CARGA                   VEL:=VELOCIDAD EN P.F.
                LDX             #ADR3
                LDY             #FI1
                JSR             CARGA                   FI1:=FI1 EN P.F.
                LDX             #ADR4
                LDY             #FI2
                JSR             CARGA                   FI2:=FI2 EN P.F.
                RTS
*
*
*****************************************************
*SUBRUTINA DE CONVERSION A/D
*****************************************************
CONVER          LDAA            #$14                    CONVERSION MULTICANAL
                STAA            ADCTL                   4 CONVERSIONES Y STOP
INCOMPL         LDAA            ADCTL                   PE4,PE5,PE6,PE7
                ANDA            #$80                    REVISA BANDERA DE
                BEQ             INCOMPL                 CONVERSION COMPLETA
                RTS
*
*
*****************************************************
CARGA           LDAA            0,X
                JSR             ESCAL                   D:=A*10 (EN FMTO A 2'S)
                LDX             #FPACC1MN
                JSR             CLEAR
                STD             0,X                     (X):=D
                JSR             SINT2FLT                ACC1:=(X) EN P.F.
                PSHY
                PULX                                    X:=Y
                JSR             PUTFPAC1                (Y):=ACC1
                RTS
*
*
*****************************************************
*SUBRUTINA DE ESCALAMIENTO
*****************************************************
ESCAL           LDAB            #$50                    $50=10.0(EN Q3)
```

```
                EORA            #$7F
                BPL             APOS
                COMA
                MUL
                JSR             DOSC
                RTS
APOS            MUL
                RTS                                     D:=A*10.0 (EN FMTO A 2'S)
*
*
*****************************************************
*ERROR DE IDENTIFICACION
*****************************************************
CALEIDEN        LDX             #TETA1
                JSR             GETFPAC1                ACC1:=TETA1
                LDX             #FI1
                JSR             GETFPAC2                ACC2:=FI1
                JSR             FLTMUL                  ACC1:=TETA1*FI1
                LDX             #EIU
                JSR             PUTFPAC1                EIU:=ACC1
                LDX             #TETA2
                JSR             GETFPAC1                ACC1:=TETA2
                LDX             #FI2
                JSR             GETFPAC2                ACC2:=FI2
                JSR             FLTMUL                  ACC1:=TETA2*FI2
                LDX             #EIU
                JSR             GETFPAC2                ACC2:=EIU
                JSR             FLTADD                  ACC1:=TETA1*FI1+TETA2*FI2
                LDX             #VEL
                JSR             GETFPAC2                ACC2:=VEL
                JSR             FLTSUB                  ACC1:=TETA1*FI1+TETA2*FI2
                LDX             #EIU                      -VEL
                JSR             PUTFPAC1                EIU:=ACC1
                RTS
*
*
*****************************************************
*IDENTIFICACION DE PARAMETROS
*****************************************************
*
*IDENTIFICACION DEL PARAMETRO TETA1
IDENPAR         LDX             #EIU
                JSR             GETFPAC1                ACC1:=EIU
                LDX             #HGAMA
                JSR             GETFPAC2                ACC2:=HGAMA
                JSR             FLTMUL                  ACC1:=HGAMA*EIU
                JSR             CHSGN1                  ACC1:=-HGAMA*EIU
                LDX             #AUX
                JSR             PUTFPAC1                AUX:=-HGAMA*EIU
                LDX             #FI1
                JSR             GETFPAC2                ACC2:=FI1
                JSR             FLTMUL                  ACC1:=-HGAMA*FI1*EIU
                LDX             #TETA1
                JSR             GETFPAC2                ACC2:=TETA1
```

```
                JSR             FLTADD          ACC1:=TETA1-HGAMA*FI1*EIU
                LDX             #TETA1
                JSR             PUTFPAC1        TETA1:=TETA1-HGAMA*FI1*EIU
*
*IDENTIFICACION DEL PARAMETRO TETA2
                LDX             #AUX
                JSR             GETFPAC1        ACC1:=-HGAMA*EIU
                LDX             #FI2
                JSR             GETFPAC2        ACC2:=FI2
                JSR             FLTMUL          ACC1:=-HGAMA*FI2*EIU
                LDX             #TETA2
                JSR             GETFPAC2        ACC2:=TETA2
                JSR             FLTADD          ACC1:=TETA2-HGAMA*FI2*EIU
                LDX             #TETA2
                JSR             PUTFPAC1        TETA2:=TETA2-HGAMA*FI2*EIU
                RTS
*
*
**************************************************
*CRITERIO DE CONVERGENCIA
**************************************************
CRITERIO        LDAA            CONT1
                INCA
                STAA            CONT1
                EORA            #$0A
                BNE             NOCONV
                CLR             CONT1
                LDX             #EIM
                JSR             GETFPAC1        ACC1:=EIM
                LDX             #EIO
                JSR             GETFPAC2        ACC2:=EIO
                JSR             FLTSUB          ACC1:=EIM-EIO
                LDX             #EIU
                JSR             GETFPAC2        ACC2:=EIU
                CLR             MANTSGN2
                JSR             PUTFPAC2        EIU:=ABS(EIU)
                JSR             FLTADD          ACC1:=EIM-EIO+EIU
                LDX             #EIM
                JSR             PUTFPAC1        EIM:=EIM-EIO+EIU
                JSR             CORRETAB        CORRE LA TABLA DE EI
                TST             PRIMERA
                BNE             CRIT2
                LDAA            CONT3
                INCA
                STAA            CONT3
                EORA            RETAR
                BNE             NOCONV
                LDAA            #$01
                STAA            PRIMERA
CRIT2           LDAA            MAXERR          A=MAX EIM
                CMPA            FPACC1EX        A - EIM
                BMI             NOCONV          EIM > A : NOCONV
                DEC             REPMIN
                BNE             NOCONV
```

```
                CLR             YACONV              EIM < A : YACONV=0
NOCONV          RTS
*
*
*****************************************************
*SUBRUTINA PARA CORRER LA TABLA DE ERRORES
*****************************************************
CORRETAB        CLR             CONTT
                LDX             #EI1
                LDY             #EIO
MASBYTE         LDD             0,X
                STD             0,Y
                INX
                INX
                INY
                INY
                INC             CONTT
                LDAA            CONTT
                EORA            #$14                PARA 10 ERRORES
                BNE             MASBYTE
                RTS
*
*
*****************************************************
*SINTONIZACION DE LOS PARAMETROS KP TI TD DEL PID
*****************************************************
SINTONI         LDX             #WN
                JSR             GETFPAC1            ACC1:=WN
                JSR             GETFPAC2            ACC2:=WN
                JSR             FLTMUL              ACC1:=WN*WN
                LDX             #TETA2
                JSR             GETFPAC2            ACC2:=TETA2=K
                JSR             FLTDIV              ACC1:=WN*WN/K
                LDX             #K2
                JSR             GETFPAC2            ACC2:=K2
                JSR             FLTDIV              ACC1:=WN*WN/(K*K2)
                LDX             #KP
                JSR             PUTFPAC1            KP:=WN*WN/(K*K2)
                LDX             #PSI
                JSR             GETFPAC1            ACC1:=PSI
                LDX             #NUMDOS
                JSR             GETFPAC2            ACC2:=2
                JSR             FLTMUL              ACC1:=2*PSI
                LDX             #WN
                JSR             GETFPAC2            ACC2:=WN
                JSR             FLTMUL              ACC1:=2*PSI*WN
                LDX             #TETA1
                JSR             GETFPAC2            ACC2:=TETA1
                JSR             FLTADD              ACC1:=TETA1+2*PSI*WN
                JSR             CHSGN1              ACC1:=-TETA1-2*PSI*WN
                LDX             #NUMUNO
                JSR             GETFPAC2            ACC2:=1
                JSR             FLTADD              ACC1:=1-TETA1-2*PSI*WN
                LDX             #TETA2
```

```
                JSR             GETFPAC2            ACC2:=TETA2
                JSR             FLTDIV              ACC1:=(1-TETA1-2*PSI*WN)/
                LDX             #KD                      TETA2
                JSR             PUTFPAC1            KD:=(1-TETA1-2*PSI*WN)/
                RTS                                      TETA2
*
*
*****************************************************
*SUBRUTINA DE TIEMPO
*****************************************************
TMUEST          LDAA            TCNTH
                LDAB            TCNTL
                ADDD            DELAY
                STD             TOC5
                LDAA            #$08
                STAA            TFLG1
LOOP            LDAA            TFLG1
                ANDA            #$08
                BEQ             LOOP
                RTS
*
*
*****************************************************
*SUBRUTINA QUE REALIZA CONTROL PD
*****************************************************
*
*****************************************************
*INICIALIZACION DE VARIABLES
*****************************************************
CONTPD          LDAA            #$88
                STAA            PUNTLOC
                LDD             #$1500
                STD             DELAY
                LDX             #UD
                JSR             CLEAR
*
INICIO2         JSR             ENTRADA             LECTURA DE ERROR Y VELOC
                JSR             PROP                CALCULA PARTE PROPOR. UP
                LDAA            PRIMERA
                BEQ             PRIMVEZ
                JSR             DERIV               CALCULA PARTE DERIV. UD
PRIMVEZ         JSR             PD                  ACC1:=UP+UD
                JSR             SALIDA              ACC1=>PUERTO B
                JSR             TMUEST              AJUSTA FREC MUESTREO
                LDAA            #$01
                STAA            PRIMERA
                JMP             INICIO2             CONTINUA CON EL CICLO
*
*
*****************************************************
*SUBRUTINA DE ADQUISICION
*****************************************************
ENTRADA         JSR             CONVER
                LDX             #FPACC1MN
```

```
                JSR             CLEAR
                LDAA            ADR2                    A:=ADR2 (ERROR)
                EORA            #$7F
                STAA            0,X                     (X):=A
                JSR             SINT2FLT                ACC1:=(X) EN PF
                LDX             #E1
                JSR             PUTFPAC1                E1:=ACC1 (ERROR EN PF)

                LDX             #FPACC1MN
                JSR             CLEAR
                LDAA            ADR1                    A:=ADR1 (VELOCIDAD)
                EORA            #$7F
                STAA            0,X                     (X):=A
                JSR             SINT2FLT                ACC1:=(X) EN PF
                LDX             #VEL
                JSR             PUTFPAC1                VEL:=ACC1 (VELOC EN PF)
                RTS
*
*
*****************************************************
*PROPORCIONAL
*****************************************************
PROP            LDX             #E1
                JSR             GETFPAC2                ACC2:=E1
                LDX             #KP
                JSR             GETFPAC1                ACC1:=KP
                JSR             FLTMUL                  ACC1:=KP*E1
                LDX             #UP
                JSR             PUTFPAC1                UP:=KP*E1
                RTS
*
*
*****************************************************
*DERIVATIVA
*****************************************************
DERIV           LDX             #VEL
                JSR             GETFPAC1                ACC1:=VEL
                LDX             #KD
                JSR             GETFPAC2                ACC2:=KD
                JSR             FLTMUL                  ACC1:=VEL*KD
                LDX             #UD
                JSR             PUTFPAC1                UD:=VEL*KD
                RTS
*
*
*****************************************************
* U= PROPOR + DERIV
*****************************************************
PD              LDX             #UP
                JSR             GETFPAC2                ACC2:=UP
                JSR             FLTADD                  ACC1:=UD+UP
                RTS
*
*
```

```
*****************************************************
*SUBRUTINA PARA SALIDA USANDO PUERTO B
*****************************************************
SALIDA      JSR         FLT2INT          FPACC1MN:=U EN ENTERO
            LDAA        FPACC1MN         A:=U EN ENTERO
            EORA        #$80
*****************************************************
DA          STAA        PORTB            A=>PUERTO B
            RTS
*
*
*****************************************************
*CONJUNTO DE RUTINAS PARA REALIZAR ARITMENTICA DE
*PUNTO FLOTANTE
*****************************************************
*
*
*
*****************************************************
*SUMA CON PUNTO FLOTANTE
*FPACC1 + FPACC2 -> FPACC1
*****************************************************
FLTADD      EQU         *
            LDX         #FPACC2EX
            JSR         CHCK0
            BNE         FLTADD1
FLTADD6     CLC
FLTADD10    NOP
            RTS
FLTADD1     LDX         #FPACC1EX
            JSR         CHCK0
            BNE         FLTADD2
FLTADD4     LDD         FPACC2EX
            STD         FPACC1EX
            LDD         FPACC2MN+1
            STD         FPACC1MN+1
            LDAA        MANTSGN2
            STAA        MANTSGN1
            BRA         FLTADD8
FLTADD2     LDAA        FPACC1EX
            CMPA        FPACC2EX
            BEQ         FLTADD7
            SUBA        FPACC2EX
            BPL         FLTADD3
            NEGA
            CMPA        #23
            BHI         FLTADD4
            TAB
            ADDB        FPACC1EX
            STAB        FPACC1EX
            LDX         #FPACC1MN
            BRA         FLTADD5
FLTADD3     CMPA        #23
            BHI         FLTADD6
```

```
            LDX     #FPACC2MN
FLTADD5     LSR     0,X
            ROR     1,X
            ROR     2,X
            DECA
            BNE     FLTADD5
FLTADD7     LDAA    MANTSGN1
            CMPA    MANTSGN2
            BEQ     FLTADD11
            TST     MANTSGN1
            BPL     FLTADD8
            LDX     FPACC2MN
            PSHX
            LDX     FPACC1MN
            STX     FPACC2MN
            PULX
            STX     FPACC1MN
            LDX     FPACC2MN+2
            PSHX
            LDX     FPACC1MN+2
            STX     FPACC2MN+2
            PULX
            STX     FPACC1MN+2
FLTADD8     LDD     FPACC1MN+1
            SUBD    FPACC2MN+1
            STD     FPACC1MN+1
            LDAA    FPACC1MN
            SBCA    FPACC2MN
            STAA    FPACC1MN
            BCC     FLTADD9
            LDAA    FPACC1MN
            COMA
            PSHA
            LDD     FPACC1MN+1
            COMB
            COMA
            ADDD    #1
            STD     FPACC1MN+1
            PULA
            ADCA    #0
            STAA    FPACC1MN
            LDAA    #$FF
            STAA    MANTSGN1
FLTADD9     JSR     FPNORM
            BCC     FLTADD12
*            LDAA    #UNFERR
            SEC
            JMP     FLTADD10
FLTADD12    JMP     FLTADD6
*
FLTADD11    LDD     FPACC1MN+1
            ADDD    FPACC2MN+1
            STD     FPACC1MN+1
            LDAA    FPACC1MN
```

```
                ADCA            FPACC2MN
                STAA            FPACC1MN
                BCC             FLTADD12
                ROR             FPACC1MN
                ROR             FPACC1MN+1
                ROR             FPACC1MN+2
                INC             FPACC1EX
                BNE             FLTADD12
*               LDAA            #OVFERR
                SEC
                JMP             FLTADD10
*
*
*************************************************
*RESTA CON PUNTO FLOTANTE
*FPACC1 - FPACC2 -> FPACC1
*************************************************
FLTSUB          EQU             *
                BSR             CHSGN2
                JSR             FLTADD
CHSGN2          LDAA            MANTSGN2
                EORA            #$FF
                STAA            MANTSGN2
                RTS
CHSGN1          LDAA            MANTSGN1
                EORA            #$FF
                STAA            MANTSGN1
                RTS
*
*
*************************************************
*MULTIPLICACION CON PUNTO FLOTANTE
*FPACC1 * FPACC2 -> FPACC1
*************************************************
FLTMUL          EQU             *
                LDX             #FPACC1EX
                JSR             CHCK0
                BEQ             FPMULT3
                LDX             #FPACC2EX
                JSR             CHCK0
                BNE             FPMULT4
                CLRA
                CLRB
                STD             FPACC1EX
                STD             FPACC1MN+1
                BRA             FPMULT3
FPMULT4         LDAA            MANTSGN1
                EORA            MANTSGN2
                STAA            MANTSGN1
                LDAA            FPACC1EX
                ADDA            FPACC2EX
                BPL             FPMULT1
                BCC             FPMULT2
*FPMULT5         LDAA            #OVFERR
```

```
FPMULT5     SEC
            BRA         FPMULT6
FPMULT1     BCS         FPMULT2
*            LDAA        #UNFERR
            SEC
            BRA         FPMULT6
FPMULT2     ADDA        #$80
            STAA        FPACC1EX
            JSR         UMULT
FPMULT3     TST         FPACC1EX
            BEQ         FPMULT5
            CLC
FPMULT6     NOP
            RTS
*
*
UMULT       EQU         *
            LDX         #0
            PSHX
            PSHX
            TSX
            LDAA        #24
            STAA        0,X
UMULT1      LDAA        FPACC2MN+2
            LSRA
            BCC         UMULT2
            LDD         FPACC1MN+1
            ADDD        2,X
            STD         2,X
            LDAA        FPACC1MN
            ADCA        1,X
            STAA        1,X
UMULT2      ROR         1,X
            ROR         2,X
            ROR         3,X
            ROR         FPACC2MN
            ROR         FPACC2MN+1
            ROR         FPACC2MN+2
            DEC         0,X
            BNE         UMULT1
            TST         1,X
            BMI         UMULT3
            LSL         FPACC2MN
            ROL         3,X
            ROL         2,X
            ROL         1,X
            DEC         FPACC1EX
UMULT3      TST         FPACC2MN
            BPL         UMULT4
            LDD         2,X
            ADDD        #1
            STD         2,X
            LDAA        1,X
            ADCA        #0
```

```
                STAA            1,X
                BCC             UMULT4
                ROR             1,X
                ROR             2,X
                ROR             3,X
                INC             FPACC1EX
UMULT4          INS
                PULX
                STX             FPACC1MN
                PULA
                STAA            FPACC1MN+2
                RTS
*
*
*************************************************
*ESTA SUBRUTINA HACE LA DIVISION CON PUNTO
*FLOTANTE DE    FPACC1 / FPACC2   GUARDANDO
*EL RESULTADO EN FPACC1
*************************************************
FLTDIV          EQU             *
                LDX             #FPACC2EX
                JSR             CHCK0
                BNE             FD1
*                LDAA            #DIVOERR
*                SEC
                RTS
FD1             LDX             #FPACC1EX
                JSR             CHCK0
                BNE             FD2
                CLC
                RTS
FD2             LDAA            MANTSGN2
                EORA            MANTSGN1
                STAA            MANTSGN1
                LDX             #0
                PSHX
                PSHX
                PSHX
                LDAA            #24
                PSHA
                TSX
                LDD             FPACC1MN
                CPD             FPACC2MN
                BNE             FD3
                LDAA            FPACC1MN+2
                CMPA            FPACC2MN+2
FD3             BHS             FD4
                INC             FPACC2EX
                BNE             FD14
*FD8             LDAA            #OVFERR
FD8             SEC
FD6             PULX
                PULX
                PULX
```

```
        INS
        RTS
FD4     LDD     FPACC1MN+1
        SUBD    FPACC2MN+1
        STD     FPACC1MN+1
        LDAA    FPACC1MN
        SBCA    FPACC2MN
        STAA    FPACC1MN
        DEC     0,X
FD14    LSR     FPACC2MN
        ROR     FPACC2MN+1
        ROR     FPACC2MN+2
        LDAA    FPACC1EX
        LDAB    FPACC2EX
        NEGB
        ABA
        BMI     FD5
        BCS     FD7
*       LDAA    #UNFERR
        BRA     FD6
FD5     BCS     FD8
FD7     ADDA    #$81
        STAA    FPACC1EX
FD9     LDD     FPACC1MN
        STD     4,X
        LDAA    FPACC1MN+2
        STAA    6,X
        LDD     FPACC1MN+1
        SUBD    FPACC2MN+1
        STD     FPACC1MN+1
        LDAA    FPACC1MN
        SBCA    FPACC2MN
        STAA    FPACC1MN
        BPL     FD10
        LDD     4,X
        STD     FPACC1MN
        LDAA    6,X
        STAA    FPACC1MN+2
FD10    ROL     3,X
        ROL     2,X
        ROL     1,X
        LSL     FPACC1MN+2
        ROL     FPACC1MN+1
        ROL     FPACC1MN
        DEC     0,X
        BNE     FD9
        COM     1,X
        COM     2,X
        COM     3,X
        LDD     FPACC1MN+1
        SUBD    FPACC2MN+1
        LDAA    FPACC1MN
        SBCA    FPACC2MN
        LDD     2,X
```

```
                BCC             FD11
                CLC
                BRA             FD13
FD11            ADDD            #1
FD13            STD             FPACC1MN+1
                LDAA            1,X
                ADCA            #0
                STAA            FPACC1MN
                BCC             FD12
                ROR             FPACC1MN
                ROR             FPACC1MN+1
                ROR             FPACC1MN+2
FD12            CLC
                JMP             FD6
*
*
*****************************************************
*SUBRUTINAS PARA CARGAR LOS ACUMULADORES CON DATOS
*DE LA MEMORIA
*****************************************************
GETFPAC1        EQU             *
                LDD             0,X
                BEQ             GETFP12
                CLR             MANTSGN1
                TSTB
                BPL             GETFP11
                COM             MANTSGN1
GETFP11         ORAB            #$80
GETFP12         STD             FPACC1EX
                LDD             2,X
                STD             FPACC1MN+1
                RTS
*
*
GETFPAC2        EQU             *
                LDD             0,X
                BEQ             GETFP22
                CLR             MANTSGN2
                TSTB
                BPL             GETFP21
                COM             MANTSGN2
GETFP21         ORAB            #$80
GETFP22         STD             FPACC2EX
                LDD             2,X
                STD             FPACC2MN+1
                RTS
*
*
*****************************************************
*SUBRUTINA PARA SALVAR LOS ACUMULADORES EN LA
*MEMORIA APUNTADA POR X
*****************************************************
PUTFPAC1        EQU             *
                LDD             FPACC1EX
```

```
                TST             MANTSGN1
                BMI             PUTFP11
                ANDB            #$7F
PUTFP11         STD             0,X
                LDD             FPACC1MN+1
                STD             2,X
                RTS
*
PUTFPAC2        EQU             *
                LDD             FPACC2EX
                TST             MANTSGN2
                BMI             PUTFP21
                ANDB            #$7F
PUTFP21         STD             0,X
                LDD             FPACC2MN+1
                STD             2,X
                RTS
*
*
*****************************************************
*SUBRUTINA QUE CHECA SI EL ACUMULADOR APUNTADO POR
*X ES CERO
*****************************************************
CHCK0           EQU             *
                PSHB
                PSHA
                LDD             0,X
                BNE             CHCK01
                LDD             2,X
CHCK01          PULA
                PULB
                RTS
*
*
*****************************************************
*SUBRUTINA PARA CONVERSION ENTERO CON SIGNO A
*PUNTO FLOTANTE
*****************************************************
SINT2FLT        EQU             *
                LDD             FPACC1MN
                PSHA
                BPL             SINTFLT1
                JSR             DOSC
                STD             FPACC1MN
SINTFLT1        BSR             UINT2FLT
                CLR             MANTSGN1
                PULA
                LDAB            #$FF
                TSTA
                BPL             SINTFLT2
                STAB            MANTSGN1
SINTFLT2        CLC
                RTS
*
```

```
*
UINT2FLT    EQU     *
            LDX     #FPACC1EX
            JSR     CHCK0
            BNE     UINTFLT1
            RTS
UINTFLT1    LDAA    PUNTLOC
            STAA    FPACC1EX
            JSR     FPNORM
            CLC
            RTS
*
*
*************************************************
*SUBRUTINA PARA REALIZAR LA CONVERSION DE PUNTO
*FLOTANTE A ENTERO CON SIGNO
*************************************************
FLT2INT     EQU     *
            LDX     #FPACC1EX
            JSR     CHCK0
            BEQ     FLT2INT3
            LDAB    FPACC1EX
            CMPB    #$81
            BLO     FLT2INT2
            CMPB    #$87            87:SOLO REQUERIMOS 7 BITS
            BHI     FLT2INT4
            SUBB    #$88
FLT2INT5    LSR     FPACC1MN
            INCB
            BNE     FLT2INT5
            CLR     FPACC1EX
            TST     MANTSGN1
            BPL     FLT2INT1
            LDAA    FPACC1MN
            NEGA
            STAA    FPACC1MN
            CLR     MANTSGN1
FLT2INT1    RTS
*FLT2INT4    LDAA    #TOLGSMER
FLT2INT4    LDAA    #$7F
            CLR     FPACC1EX
            TST     MANTSGN1
            BPL     FLT2INT6
            NEGA
FLT2INT6    STAA    FPACC1MN
            CLR     MANTSGN1
*            SEC
            RTS
FLT2INT2    LDD     #0
            STD     FPACC1EX
            STD     FPACC1MN+1
FLT2INT3    RTS
*
*
```

```
*****************************************************
*SUBRUTINA PARA NORMALIZAR
*****************************************************
FPNORM      EQU          *
*            LDX          #FPACC1EX
*            BSR          CHCK0
*            BEQ          FPNORM3
            TST          FPACC1MN
            BMI          FPNORM3
            LDD          FPACC1MN
FPNORM2     DEC          FPACC1EX
            BEQ          FPNORM4
            LSLD
            BPL          FPNORM2
            STD          FPACC1MN
FPNORM3     CLC
            RTS
FPNORM4     SEC
            RTS
*
*
*****************************************************
*SUBRUTINA PARA REALIZAR DOS COMPLEMENTO DEL ACC D
*****************************************************
DOSC        COMA
            COMB
            ADDD         #1
            RTS
*
*
*****************************************************
*SUBRUTINA PARA LIMPIAR UNA VARIABLE DE 32 BITS
*****************************************************
CLEAR       PSHA
            PSHB
            LDD          #$0000
            STD          0,X
            STD          2,X
            PULB
            PULA
            RTS
*
*
            END          $C000
```

Apéndice B

Programas en SIMNON

B.1 Identificación

```
MACRO idnmotz2
   SYST motor idenz filtro crit mtidcnz
   WRITE 'Amplitud de la senal: '
   READ amp num
   PAR r:amp
   WRITE 'Frecuencia de la senal: '
   READ frec num
   PAR w:frec
   WRITE 'Polo de la planta: '
   READ polo num
   PAR a:polo
   WRITE 'Ganancia de la planta: '
   READ k1p num
   PAR k1:k1p
   WRITE 'Ganancia de adaptacion: '
   READ gamadp num
   PAR gamma:gamadp
   STORE teta1 teta2 u[motor] y2[motor] ei eim
   PLOT aesti kesti
   AXES h 0 50 v -10 10
   SIMU 0 50 - mark
END

CONTINUOUS SYSTEM motor
INPUT u
OUTPUT y1 y2
STATE yp1 yp2
DER dyp1 dyp2
TIME t
"Dinamica de la planta
   dyp1=k2*yp2
   dyp2=-a*yp2+k1*u
   y1=yp1
   y2=yp2
"Parametros
   a:5
   k1:10
```

```
   k2:7.773
END

DISCRETE SYSTEM idenz
INPUT y2 f11 f12
OUTPUT ei
STATE teta1 teta2
NEW nteta1 nteta2
TIME t
TSAMP ts
"Error de identificacion
   ei=teta1*f11+teta2*f12-y2
"Ley de actualizacion
   nteta1=teta1-h*gamma*f11*ei
   nteta2=teta2-h*gamma*f12*ei
"Calculo de los parametros del motor
   kest1=teta2
   aest1=lamda-teta1
   ts=t+h
"Parametros
   h:0.01
   gamma:3
   lamda:1
END

CONTINUOUS SYSTEM filtro
INPUT u y2
OUTPUT f11 f12
STATE ff11 ff12
DER dff11 dff12
"Dinamica de los filtros
   dff11=y2-lamda*ff11
   dff12=u-lamda*ff12
   f11=ff11
   f12=ff12
"Parametros
   lamda:1
END

DISCRETE SYSTEM crit
INPUT ei1
STATE ei0 ei1 ei2 ei3 ei4 ei5 ei6 ei7 ei8 ei9
NEW nei0 nei1 nei2 nei3 nei4 nei5 nei6 nei7 nei8 nei9
TIME t
TSAMP ts
"Criterio IEA
   nei0=ei1
   nei1=ei2
   nei2=ei3
   nei3=ei4
   nei4=ei5
   nei5=ei6
   nei6=ei7
   nei7=ei8
```

```
   ne18=e19
   ne19=abs(e11)
   elm=e10+e11+e12+e13+e14+e15+e16+e17+e18+e19
   ts=t+h1
   !!sto=if elm<maxerr then 1 else 0
"Parametros
   h1..1
   maxerr:4
END

CONNECTING SYSTEM mtidcnz
TIME t
   u=if sin(w*t)>0 then r else -r
   u[motor]=u
   u[filtro]=u
   y2[filtro]=y2[motor]
   y2[idenz]=y2[motor]
   f11[idenz]=f11[filtro]
   f12[idenz]=f12[filtro]
   e11[crit]=e[idenz]
"Parametros
   w:0.5
   r:.1
END
```

B.2 Control

```
MACRO servopdz
   SYST motor pdz mtpdconz
   WRITE 'Polo de la planta: '
   READ polo num
   PAR a:polo
   WRITE 'Ganancia de la planta: '
   READ klp num
   PAR k1:klp
   WRITE 'Valor de Kp: '
   READ kpc num
   PAR kp:kpc
   WRITE 'Valor de Kd: '
   READ kdc num
   PAR kd:kdc
   WRITE 'Amplitud del escalon: '
   READ amp num
   PAR r:amp
   STORE yp1 yp2 e u[motor]
   PLOT yp1
   AXES h 0 1 v 0 5
   SIMU 0 1 - mark
END

DISCRETE SYSTEM pdz
INPUT y2 e
OUTPUT u
```

```
TIME t
TSAMP ts
"Control
   u=kp*e-kd*y2
   ts=t+h
"Parametros
   kp:10.91089
   kd:2.30394
   h:.005
END

CONNECTING SYSTEM mtpdconz
TIME t
   u[motor]=u[pdz]
   y2[pdz]=y2[motor]
   e[pdz]=r-y1[motor]
"Parametros
   r:2
END
```

Apéndice C

Rutinas de punto flotante

1. Formato en punto flotante del acumulador

Los 10 bytes de RAM reservados para este paquete se destinan a crear un par de variables las cuales estarán funcionando a la manera de acumuladores a los que se ha denominado FPACC1 y FPACC2. Cada uno de estos acumuladores consta de 5 bytes, los que están repartidos en un exponente de un byte, una mantisa de 3 bytes y un byte más que es usado para indicar el signo de la mantisa.

Fig. C.1 Formato del acumulador en punto flotante.

El byte del exponente indica la posición del punto binario, empezando con el valor 128 ($80) para indicar que el punto se localiza inmediatamente antes de la primer cifra de la mantisa, ésto por supuesto, impide que se presenten casos con exponentes negativos y hace más fáciles las comparaciones en punto flotante. Este exponente de un sólo byte proporciona un rango dinámico del orden de $1x10^{\pm 38}$.

Los tres bytes de la mantisa son utilizados para mantener la parte entera y la fraccionaria del número. Asumimos que la mantisa esta normalizada, lo cual quiere decir que, el bit más significativo (MSB) siempre es un uno.

Se utiliza un byte separado para indicar el signo de la mantisa, una mantisa positiva tendrá asociado un cero ($00) mientras que una negativa un menos uno ($FF).

Ejemplos:

4	83	800000	00
-0.5	80	800000	FF
0.25	7F	800000	00
3.1415927	82	C90FDB	00
-3.1415927	82	C90FDB	FF

En nuestro programa se han apartado las 10 localidades necesarias para ambos acumuladores de la siguiente manera:

```
*
FPACC1EX        RMB     1
FPACC1MN        RMB     3
MANTSGN1        RMB     1
FPACC2EX        RMB     1
FPACC2MN        RMB     3
MANTSGN2        RMB     1
*
```

2. Formato en punto flotante de la memoria

La forma en que los números en este formato son almacenados dentro de la memoria difiere un poco respecto del formato con que son manejados en los acumuladores. El formato con que son almacenados los números en la memoria es llamado "forma normalizada de bit oculto" [11]. La idea es guardar números de 5 bytes en espacios de memoria de 4 bytes con la finalidad de optimizar espacio de memoria.

Fig. C.2 Formato en punto flotante de la memoria.

Así que, el número es almacenado en 4 bytes consecutivos, donde el exponente ocupa el primer byte (dirección más pequeña). La mantisa ocupa los tres 3 bytes restantes. Ahora bien, dado que el bit más significativo de la mantisa es siempre un 1, podemos reemplazarlo y en su lugar colocar

el signo de la mantisa. Así, el bit más significativo de la mantisa, deberá tener un cero para números positivos, y un uno para números negativos.

Ejemplos:

4	83	000000
-0.5	80	800000
0.25	7F	000000
3.1415927	82	490FDB
-3.1415927	82	C90FDB

3.Rutinas y funciones de punto flotante

A continuación se presenta un breve descripción de cada una de las rutinas utilizadas dentro del programa. Esta descripción incluye, el nombre de la subrutina, la operación que realiza, su entrada o entradas, su salida y subrutinas que son llamadas.

Multiplicación

Nombre: FLTMUL
Operación: FPACC1 x FPAAC2 → FPACC1
Llamadas: CHCKO
Entrada: FPACC1 y FPACC2
Salida: FPACC1

Suma

Nombre: FLTADD
Operación: FPACC1 + FPAAC2 → FPACC1
Llamadas: CHCKO
Entrada: FPACC1 y FPACC2
Salida: FPACC1

Substracción

Nombre: FLTSUB
Operación: FPACC1 - FPAAC2 → FPACC1
Llamadas: FLTADD
Entrada: FPACC1 y FPACC2
Salida: FPACC1

División

Nombre: FLTDIV

Operación: FPACC1 ÷ FPAAC2 → FPACC1

Entrada: FPACC1 y FPACC2

Salida: FPACC1

Entero sin signo a punto flotante

Nombre: UINT2FLT

Operación: (Entero no signado de 16 bits) → FPACC1

Llamadas: FPNORM, CHCKO

Entrada: Los 16 bits más significativos de la mantisa de FPACC1 deberán contener un entero no signado de 16 bits.

Salida: FPACC1 contiene la representación en punto flotante del entero no signado de 16 bits.

Entero con signo a punto flotante

Nombre: SINT2FLT

Operación: (Entero signado de 16 bits) → FPACC1

Llamadas: UINT2FLT

Entrada: Los 16 bits más significativos de la mantisa de FPACC1 deberán contener un entero signado de 16 bits.

Salida: FPACC1 contiene la representación en punto flotante del entero signado de 16 bits.

Punto flotante a entero

Nombre: FLT2INT

Operación: FPACC1 → (Entero signado de 16 bits)

Llamadas: CHCKO

Entrada: FPACC1 deberán contener un número en punto flotante dentro del rango -32767≤FPACC1≤65535.

Salida: Los 8 bits menos significativos de la mantisa de FPACC1 contendrán al número de 8 bits signado

4.Rutinas para manejo de memoria

Anteriormente vimos que la forma en que los números son representados en los acumuladores (formato del acumulador en PF) difiere de la forma en que éstos son almacenados en la memoria (formato de la memoria en PF), y

que, esto primordialmente se hizo para ahorrar memoria, sobre todo, cuando se tienen varias variables con formato de punto flotante. El paquete cuenta con cuatro subrutinas que permiten realizar conversiones de un formato a otro al mismo tiempo que permiten mover un número desde la memoria hacia algún acumulador o viceversa, es decir, podemos cargar a los acumuladores con números desde la memoria, o bién, salvar los valores de los acumuladores en la memoria.

Obtiene FPACC(x)

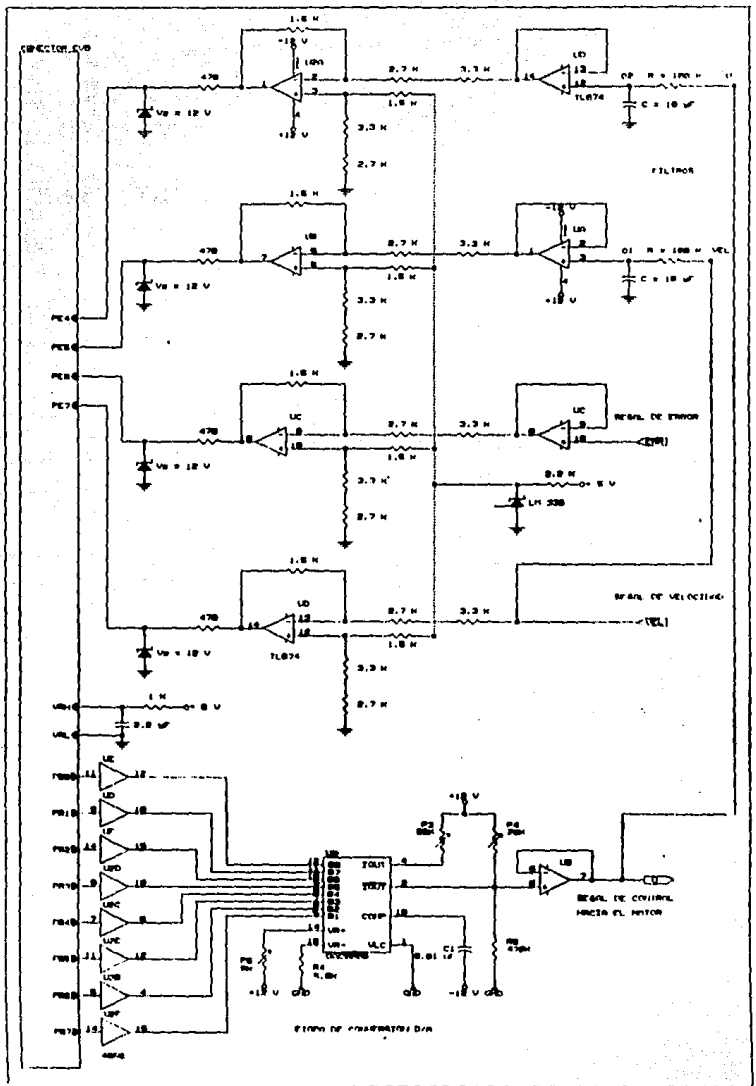
Nombre:	GETFPAC1 ó GETFPAC2
Operación:	(X)→FPACC1 ó (X)→FPACC2
Entrada:	El registro de índice X apunta al número de 4 bytes en la memoria que va a ser movido al acumulador de punto flotante FPACC1 ó FPACC2.
Salida:	El número apuntado por X esta en el acumulador de punto flotante especificado FPACC1 ó FPACC2.

Coloca FPACC(x)

Nombre:	PUTFPAC1 ó PUTFPAC2
Operación:	FPACC1→(X) ó FPACC2→(X)
Entrada:	El registro de índice X apunta a 4 localidades de memoria consecutivas donde el número será almacenado.
Salida:	El acumulador de punto flotante especificado FPACC1 ó FPACC2 es movido dentro de cuatro localidades de memoria consecutivas apuntadas por el registro de índice X.

Apéndice D

Tarjeta de interfaz


FILTROS
TL074
LM 336
TL074
SEÑAL DE ERROR
SEÑAL DE VELOCIDAD
SEÑAL DE CONTROL HACIA EL MOTOR