Universidad Nacional Autónoma de México

FACULTAD DE INGENIERIA

DISEÑO Y CONSTRUCCION DE UN CONTROLADOR DIGITAL PID PARA UN MOTOR DE CORRIENTE DIRECTA UTILIZANDO UN SISTEMA MULTIPROCESADOR.

T E S I S

Que para obtener el Título de:
INGENIERO EN COMPUTACION

p r e s e n t a :

GRACIELA NAJERA DEL RIO

Director: Ing. José Antonio Arredondo Garza

México, D. F. 1995

Dedicatorias

A **mi papá:**

Antonio Nájera Tapia, con mucho cariño y admiración porque todos sus hechos, sus enseñanzas, su amor y su recuerdo me impulsaron siempre para lograr éste que es uno de mis más grandes anhelos.

A **mi mamá:**

G. Yolanda del Río de Nájera, agradeciéndole infinitamente su cariño, paciencia, comprensión y ayuda incondicional.

A **mis hermanos:**

Yolanda, Elizabeth, Antonio y Gabriela.

A **mis maestros:**

Con admiración, cariño y agradecimiento por su ayuda y todas sus enseñanzas.

A **mis amigos.**

GRACIELA

A ***mi papá:***

Alfredo Cuevas Godínez, con mucho cariño, porque siempre ha sido mi amigo, me ha impulsado a conseguir mis metas y en todo momento he contado con su apoyo.

A ***mi mamá:***

María del Pilar Angélica Garibay de Cuevas con mucho cariño, porque en cada etapa y en cada éxito o fracaso en mi vida ha estado a mi lado.

A ***mis hermanos:***

Luis Alfredo y Norma Angélica por el cariño que me han brindado.

A ***mis maestros.***

A ***mis familiares y amigos.***

JORGE

Agradecimientos

A la Universidad Nacional Autónoma de México.

A la Facultad de Ingeniería.

Al Ing. José Antonio Arredondo Garza, por todas sus enseñanzas, su apoyo y su valiosa asesoría.

A quienes con sus consejos y actitudes nos impulsaron a lo largo del desarrollo de este trabajo.

INDICE

I. INTRODUCCION

1.1 ASPECTOS HISTORICOS

Los sistemas de control automático han ido invadiendo cada vez más todas las áreas de la ingeniería y de la ciencia, tales sistemas constituyen una especie de indicador de progreso y desarrollo. Además de su extrema importancia en vehículos espaciales, en guiado de proyectiles y en sistemas de pilotaje de aviones, entre otros, el control automático se ha convertido en parte importante e integral de los procesos industriales modernos. El uso de control automático resulta esencial en operaciones industriales como el control de velocidad, presión, temperatura, humedad, viscosidad y flujo en las industrias de procesos; maquinado, manejo y armado de piezas mecánicas en las industrias de fabricación, etc.

Sin embargo, los avances actuales en la teoría y práctica de control automático, que proporcionan medios para lograr el funcionamiento óptimo de sistemas dinámicos, mejorar la calidad y disminuir los costos de producción, liberar de la complejidad de muchas rutinas, de las tareas manuales repetitivas, expandir el ritmo de producción, etc. no surgieron repentinamente, a continuación se presenta un breve bosquejo histórico de los avances que poco a poco dieron origen a lo que hoy conocemos y aplicamos como control automático.

En 1970, Mayr escribe una interesante historia de los primeros trabajos sobre control. Investiga el control de mecanismos desde la antigüedad y describe algunos de los primeros ejemplos, el control de la tasa de flujo para regular un reloj de agua y el control del nivel del líquido en una lámpara de aceite y en un recipiente de vino, que se mantiene lleno a pesar de las muchas tazas que se sacan. Un caso más moderno de un sistema de control descrito por Mayr es el control de temperatura de un horno para calentar una incubadora, sistema diseñado por Drebbel hacia 1620. Este horno consta de una caja que contiene el fuego, con un tubo en la parte superior provisto de un regulador. Dentro de la cámara de combustión está la incubadora de paredes dobles y el hueco que queda entre las paredes se llena con agua. El sensor de temperatura es un recipiente de vidrio lleno de alcohol y mercurio, colocado en la cámara de agua entre las paredes en torno a la incubadora. A medida que el fuego calienta la caja y el agua, el alcohol se dilata y el vástago con

flotador se desplaza hacia arriba, bajando el regulador sobre la boca del tubo. Si la caja está demasiado fría el alcohol se contrae, el regulador se abre y el fuego arde más fuertemente. La temperatura deseada está determinada por la longitud del vástago con flotador, que determina la apertura del regulador para una dilatación determinada del alcohol.

La búsqueda de un medio para controlar la velocidad de rotación de un eje, fue un problema famoso en las crónicas del control automático. De los varios métodos que se intentaron, el más prometedor resultó ser el que usaba un péndulo cónico o regulador de bola flotante y que fue adaptado a la máquina de Watt, la cual constituye uno de los trabajos más significativos en control automático y fue empleado para el control de la velocidad de una máquina de vapor en el siglo XVIII.

Otros inventores introdujeron mecanismos que integraron el error de velocidad y así proporcionaron una reposición automática.

Airy, Minorsky, Hazen y Nyquist, entre muchos otros desarrollaron estudios teóricos muy importantes. En 1922 Minorsky trabajó en controles automáticos de dirección en barcos y mostró cómo se podría determinar la estabilidad, a partir de las ecuaciones diferenciales que describen el sistema. En 1932, Nyquist desarrolló un procedimiento relativamente simple para determinar la estabilidad de los sistemas de malla cerrada, sobre la base de la respuesta en malla abierta. En 1934, Hazen, quien introdujo el término "servomecanismos" para los sistemas de control de posición, estudió el diseño de servomecanismos repetidores capaces de seguir una entrada cambiante.

En el campo del control realimentado de procesos industriales, caracterizado por procesos que además de complejos, son no lineales y están sujetos a retrasos de tiempo relativamente largos entre actuador y sensor, se desarrolló la práctica del control proporcional más integral, más derivativo (PID), descrito por Callender, Hartree y Porter (1936). Esta tecnología, basada en un amplio trabajo experimental y aproximaciones linealizadas simples al sistema dinámico, llevó a experimentos estándar apropiados para la aplicación en el campo y finalmente a una satisfactoria "sintonía" de los coeficientes del controlador PID.

Durante la década de los 40's, los métodos de respuesta en frecuencia permitieron a los ingenieros el diseño de sistemas de control realimentado lineal. Desde el fin de esa década hasta los primeros años de la siguiente, se desarrolló completamente el método del lugar geométrico de las raíces en el diseño de sistemas de control.

Los métodos de respuesta en frecuencia y del lugar geométrico de las raíces, que constituyen la parte fundamental de la teoría de control clásica, llevan a sistemas que son estables y que satisfacen un conjunto de requerimientos de funcionamiento arbitrarios. Pero estos sistemas no eran significativamente óptimos.

Posteriormente, como las plantas modernas con muchas entradas y salidas se han hecho más y más complejas, la descripción de un sistema de control moderno requiere de una gran cantidad de ecuaciones, y la teoría de control clásica que trata de sistemas de entrada y salida única, se vuelve absolutamente impotente ante sistemas de múltiples entradas y salidas. Por ello, desde 1960 se ha desarrollado la teoría de control moderna para afrontar la complejidad creciente de las plantas y las necesidades rigurosas de exactitud y costo en aplicaciones militares, espaciales e industriales.

Dada la fácil disponibilidad de computadoras, actualmente, el uso de las mismas en sistemas de control se ha convertido en una práctica habitual. La gran versatilidad de la computadora ha permitido desarrollar una amplia variedad de tareas, como adquirir datos del proceso para efectuar balances de materia y energía, calcular eficiencias y rendimientos, elaborar reportes con la información procesada, simular modelos de diversos procesos asumiendo condiciones que no pueden ser aplicadas físicamente, etc.

La idea de utilizar computadoras digitales como componentes de control, surgió alrededor de 1950 con Aström y Wittenmark, en aplicaciones de control de misiles y dispositivos aeroespaciales, sin embargo, como las computadoras de esa época eran muy grandes y consumían mucha potencia, la idea se abandonó y se optó por desarrollar computadoras de propósito específico

llamadas DDA (Analizadores Diferenciales Digitales), las cuales se enfocaban únicamente a resolver problemas de la navegación espacial.

En 1956, la compañía TEXACO solicitó a la compañía Aeroespacial Thomson Ramo Woolridge (TRW) un estudio de factibilidad para instalar una unidad de polimerización controlada por computadora en la refinería de Port Arthur, Texas. Este proyecto entró en operación en 1959, la arquitectura del diseño se basó en la computadora RW-300, la cual controlaba 26 flujos, 72 temperaturas, 3 presiones y 3 composiciones. Con este proyecto se inició la primera etapa en la historia del control por computadora. En esta etapa, la computadora actuaba solamente como un supervisor del comportamiento de la planta. Es característico de esta etapa la implementación de dos modos supervisorios de operación: "guía del operador" y "control de referencia". En el modo "guía del operador" la computadora imprimía mensajes al operador indicándole las acciones a tomar, y, en el modo "control de referencia", la computadora ajustaba los puntos de operación de los reguladores analógicos. En ambos casos se empleaban instrumentos analógicos para el control.

A partir de la fecha en que se concluyó exitosamente el proyecto Texaco, los fabricantes de computadoras, las instituciones de investigación y la industria en general, dieron un fuerte impulso al desarrollo de estos sistemas, se iniciaron varios estudios de factibilidad y para 1962 el número de computadoras aplicadas al control de procesos había aumentado enormemente.

En 1962, la compañía Imperial Chemical Industries cambió todos sus instrumentos de control analógicos por una computadora digital para efectuar las funciones de la instrumentación reemplazada: medir 224 variables y controlar 129 válvulas. Lo más importante de este proyecto fue el hecho de que la medición y el control se hacían directamente con la computadora, la cual, pasó a formar parte del lazo de control. Este cambio marcó el inicio de la segunda etapa en el desarrollo del control por computadora, la del "control digital directo" (DDC), la figura 1.1 ilustra este esquema.

Las ventajas más importantes que introdujo la sustitución de la tecnología analógica por la

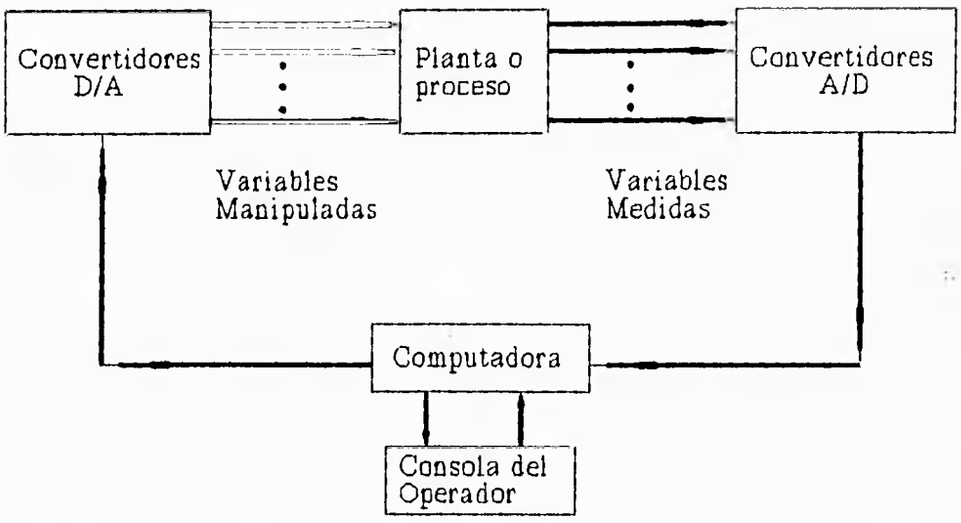


Figura 1.1. Esquema de Control Digital Directo

digital fueron en relación con el costo y la flexibilidad. En la tecnología analógica el costo depende del número de lazos de control, mientras que con los sistemas DDC el costo por lazos de control adicionales es mínimo; a pesar de que normalmente la inversión inicial es más fuerte, finalmente resultan de menor costo. Además, los sistemas DDC son más flexibles, ya que mientras los cambios en los lazos de control analógico se hacían realambrando, en los sistemas digitales los cambios se efectúan programando.

Posteriormente, dos acontecimientos relativos al desarrollo de la tecnología digital influyeron determinantemente en el avance del control digital. El primero de éstos ocurrió a mediados de los años 60's con la aparición de las minicomputadoras, las cuales, por su potencia y reducidas dimensiones eran adecuadas para dar solución a problemas de control de mediana magnitud, y por su menor costo eran accesibles aún para proyectos de bajo presupuesto. La microcomputadora fue el segundo de los acontecimientos mencionados, ya que su aparición significó otro gran impulso en

esta disciplina, porque si bien las minicomputadoras eran pequeñas, no lo eran suficientemente para la mayoría de los pequeños problemas de control, los cuales demandaban soluciones de menor costo y menor tamaño en el equipo empleado. Con el nacimiento de las microcomputadoras un gran número de estos problemas tuvieron solución.

La importancia de estos últimos sucesos en la historia del control es mayúscula, si consideramos que en 1965 a 3 años del advenimiento de los sistemas DDC, el número de computadoras empleadas en el control de procesos era alrededor de 1000 y para 1975, el número aumentó a 100,000.

En la actualidad el perfeccionamiento de la técnica de integración de circuitos a muy grande escala (VLSI) ha permitido la fabricación de microprocesadores muy baratos y poderosos, con lo cual se tienen dispositivos digitales programables al alcance de cualquier proyecto de control, siendo posible además realizar algoritmos más elaborados. Es común la sustitución de equipo analógico por sistemas basados en microprocesador. También se ha realizado el control de plantas por medio de "redes de control distribuido", las cuales, emplean una minicomputadora, por ejemplo, para coordinar un conjunto de microcomputadoras que efectúan el control directo de la planta y que para dicho efecto se encuentran distribuidas a lo largo de ella.

Es importante mencionar algunas de las ventajas que presenta el control digital, sin olvidar que éste surgió y tiene como base sólida toda la teoría del control analógico.

- La tecnología digital es de bajo costo.

- El consumo de potencia es bajo.

- Las señales digitales codificadas que se emplean pueden ser almacenadas por un tiempo indefinido además de que pueden ser transmitidas con mayor confiabilidad mediante el uso de códigos de protección.

· En telemetría es muy útil, cuando se requiere un solo canal de comunicación para varios sistemas de control, multiplexando señales.

· Muchos de los cambios que en un sistema analógico se efectúan alambrando, en un sistema digital se hacen programando.

1.2 OBJETIVO

Diseñar y construir un controlador digital PID para un motor de corriente directa (CD), utilizando un sistema multiprocesador, que resulte versátil y de fácil manejo para cumplir de manera precisa con las especificaciones de control que una determinada aplicación requiera.

1.3 PLANTEAMIENTO DEL PROBLEMA

El concepto de control es muy común, puede referirse a una interacción específica hombre-máquina, como guiar una automóvil, o bien, puede referirse solamente a máquinas, como el control de velocidad de un motor o el control de temperatura de una sala. Una clase especial de sistemas de control está compuesta por aquellos que usan retroalimentación, esta clase se caracteriza por el hecho de que la variable controlada, ya sea velocidad, temperatura, etc., se mide por un sensor y la información se retroalimenta para influir en la variable controlada.

La teoría de control considera varias acciones básicas de control, la acción on-off, la acción proporcional, la acción integral y la acción derivativa, las cuales se pueden aplicar en forma combinada dando lugar a diversos controladores de uso común, de todas estas combinaciones, el controlador PID (Proporcional-Integral-Derivativo) utiliza acciones de control que son complementarias entre sí, por lo que presenta varias ventajas sobre los demás algoritmos, mejorando la precisión y la estabilidad del sistema al que se le aplica. Por esta razón, en el presente proyecto, se empleará

este algoritmo de control.

El perfeccionamiento de la técnica de integración de circuitos a muy grande escala que se ha logrado en la actualidad, ha permitido el uso masivo de sistemas de control digital, empleando microprocesadores, lo cual representa ventajas importantes, pues la tecnología digital tiene un bajo costo, el consumo de potencia es bajo, etc.

Por otra parte, los motores de corriente directa han sido por mucho tiempo un componente fundamental en cualquier planta industrial, debido a la facilidad con la que su velocidad puede controlarse o bien por el gran par que estos pueden desarrollar. Sin embargo, los métodos convencionales para controlar la velocidad, suelen provocar que el motor opere con baja eficiencia, debido al tiempo y esfuerzo que se emplea para establecer la velocidad precisa con el ajuste de los reostatos que usualmente se emplean.

La electrónica de potencia encuentra en esta rama un campo de aplicación muy amplio. El control de velocidad por medios electrónicos trae consigo una mayor precisión y menores gastos de operación para la industria por tratarse de un control automático.

Tomando en cuenta todo lo anterior, el problema consiste en diseñar y construir un controlador digital PID para un motor de corriente directa, que en la industria, puede ser de gran utilidad en aquellos procesos donde la velocidad de operación sea una variable a controlar no importando si el motor opera bajo condiciones de carga o sin esta, e incluso en condiciones de carga variable o con la presencia de perturbaciones tanto internas, como externas. Para ello utilizaremos un sistema multiprocesador de propósito general, basado en dos microcontroladores MC68HC11F1, que ejecutarán el algoritmo de control PID, una interfaz de entrada y una interfaz de salida, de forma que el sistema en su conjunto sea capaz de mantener la velocidad del motor de CD en el valor deseado.

Es importante resaltar que el sistema que se va a construir contará con un medio de

interacción con el usuario completamente amigable, de forma que el operador podrá monitorear y controlar la operación del motor desde una computadora personal.

Este proyecto constituye únicamente un prototipo pero es la base para construir un sistema de control lo suficientemente versátil para poderse user en cualquier planta industrial. Además presenta una característica importante, el uso de un "sistema multiprocesador" para desarrollar el algoritmo de control digital, lo cual se traduce en un aumento en la velocidad de procesamiento y en el empleo de una nueva técnica, producto de las necesidades cada vez mas exigentes de la época actual.

A lo largo del desarrollo del proyecto, seguramente nos enfrentaremos con diversos problemas tanto de hardware como de software, referentes a la estabilidad del sistema, a la construcción de la etapas de interfaz de entrada/salida, y al planteamiento e implementación tanto del algoritmo de control como del sistema de interfaz con el usuario, los cuales se tratarán más a detalle en el momento adecuado.

En el análisis y diseño del controlador digital se seguirá a grandes rasgos, el siguiente procedimiento:

- Elección de la estructura de un controlador analógico PID.

- Obtención mediante aproximaciones de un controlador discreto.

- Una vez obtenida la estructura del controlador discreto, determinación de la ecuación en diferencias, que será la parte central del programa para el "sistema multiprocesador".

1.4 CONCEPTOS GENERALES

1.4.1 Sistemas continuos

En los sistemas continuos las variables que intervienen en sus modelos matemáticos son funciones del tiempo continuo, es decir, la variable tiempo puede tomar todos los valores del conjunto de los números reales, y se representan mediante ecuaciones diferenciales ordinarias lineales.

1.4.2 Sistemas Discretos

En los sistemas discretos las variables que intervienen en sus modelos matemáticos son funciones del tiempo discreto, es decir, la variable tiempo puede tomar todos los valores del conjunto de los números naturales, y se representan mediante ecuaciones en diferencias lineales.

1.4.3 Concepto de sistema en malla abierta

Los sistemas de control en malla abierta son aquellos en los que la salida no tiene efecto sobre la acción de control. Es decir, en un sistema de control en malla abierta, la salida ni se mide ni se realimenta para comparación con la entrada. La figura 1.2 muestra la relación entrada-salida de este tipo de sistemas.

En un sistema de control de malla abierta cualquiera, no se compara la salida con la entrada de referencia, por ello, para cada entrada de referencia corresponde una condición de operación determinada. De forma que la exactitud del sistema depende de la calibración, por ello estos sistemas deben ser cuidadosamente calibrados y para que realmente resulten útiles es necesario

mantener esa calibración. Cuando se presentan perturbaciones, un sistema de control de malla abierta no cumple su función asignada.

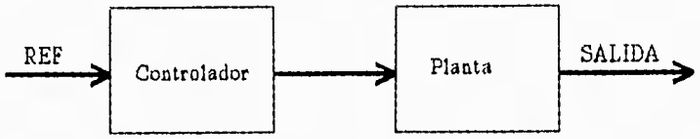


Figura 1.2 Sistema de control en malla abierta

Entonces, en la práctica, solamente se puede usar el control en malla abierta si la relación entre la entrada y la salida es conocida y si no hay perturbaciones ni internas ni externas.

1.4.4 Concepto de sistema en malla cerrada

Un sistema de control en malla cerrada es aquel en el que la señal de salida tiene efecto directo sobre la acción de control. Es decir, los sistemas de control en malla cerrada son sistemas de control realimentado. En ellos, la señal de error actuante, que es la diferencia entre la señal de entrada y la realimentación entra al controlador con el fin de reducir el error y llevar la salida del sistema al nivel deseado. El término "malla cerrada" implica el uso de una acción de realimentación para reducir el error del sistema, para estabilizar sistemas, acelerar la respuesta transitoria, mejorar las características del estado estable, facilitar el rechazo a perturbaciones y disminuir la sensibilidad a las variaciones de los parámetros físicos de los componentes. El diagrama de bloques de la figura 1.3 muestra la relación entrada-salida de un sistema de control de malla cerrada.

Hay numerosos sistemas de control en malla cerrada en la industria y en el hogar. Por ejemplo, los sistemas de control de velocidad de motores, los refrigeradores domiciliarios, los

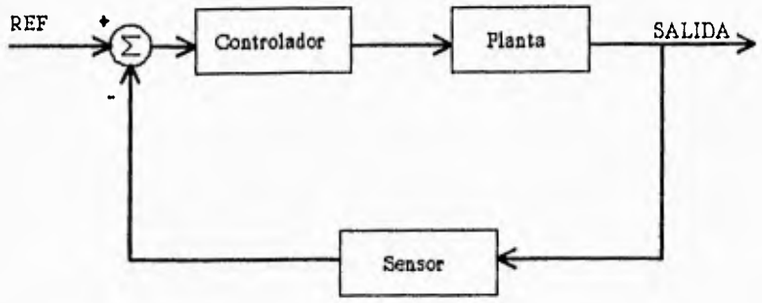


Figura 1.3 Sistema de control en malla cerrada.

calentadores de agua automáticos, los sistemas de calefacción con control termostático, etc.

Sin embargo, en los sistemas de control en malla cerrada, la estabilidad constituye un problema de importancia por la tendencia a sobrecorregir errores, lo cual puede producir oscilaciones de amplitud constante o variable.

Es importante notar que los sistemas de control en malla cerrada solamente tienen ventajas si se presentan perturbaciones no previsibles y/o variaciones imprevisibles de componentes del sistema.

1.4.5 Sistemas analógicos

En un sistema analógico las cantidades físicas son principalmente analógicas en naturaleza. Algunos ejemplos de sistemas analógicos son las computadoras analógicas, sistemas de radiodifusión y grabación de cintas de audio. La señal o las señales de salida de estos sistemas, se caracterizan por tener un número infinito de valores de amplitud.

1.4.6 Sistemas digitales

Un sistema digital es una combinación de dispositivos (eléctricos, mecánicos, fotoeléctricos, etc.) ensamblados a fin de desempeñar ciertas funciones en las cuales las cantidades se representan en forma digital. Algunos de los sistemas digitales más comunes son las computadoras y las calculadoras digitales, los voltímetros digitales y la maquinaria controlada en forma numérica. La señal o las señales de salida de estos sistemas, se caracterizan por tener un número finito de valores de amplitud.

En general, los sistemas digitales ofrecen las ventajas de programabilidad, exactitud y capacidad de almacenamiento. Además, los sistemas digitales son por lo general más versátiles en una gama más amplia de aplicaciones.

En el mundo real, muchas cantidades son analógicas en naturaleza y son estas cantidades las que a menudo se miden, se monitorean o se controlan. De forma que si se desean emplear las ventajas de las técnicas digitales, es obvio que deben existir muchos sistemas híbridos. Entre ellos, los más comunes son los sistemas de control de procesos industriales en los cuales se miden y controlan cantidades analógicas como la temperatura, presión, velocidad, etc. Para la construcción de este tipo de sistemas es necesario emplear convertidores Analógico/Digital y Digital/Analógico.

1.4.7 Teorema de muestreo

Debido a que el muestreo es una propiedad básica de los sistemas de control digital, es necesario tener una buena comprensión del proceso de muestreo.

Muestreo significa el reemplazo de una señal continua en el tiempo por una secuencia de números, la cual representa los valores de la señal en ciertos instantes de tiempo.

Reconstrucción de señales es el proceso de convertir una secuencia de números en una señal continua en el tiempo.

El teorema de muestreo indica que para poder recuperar una señal a partir de sus muestras se debe muestrear al menos al doble de la frecuencia más alta (W) en la señal continua, es decir:

$$f_0 = \frac{1}{T} \geq 2W$$

1.4.8 Especificaciones de respuesta transitoria

En todo sistema de control, el diseñador debe cumplir con ciertas especificaciones para la respuesta transitoria ante una entrada escalón. Esto tiene por objetivo maximizar la eficiencia del proceso que se desea controlar, e incluso en algunos casos, evitar posibles daños. Es habitual especificar lo siguiente:

- Tiempo de retardo (t_d): Es el tiempo que tarda la respuesta en alcanzar por primera vez el 50% de su valor final.

- Tiempo de levantamiento (t_r): Es el tiempo requerido para que la respuesta crezca del 10% al 90% de su valor final.

- Tiempo de pico (t_p): Es el tiempo requerido por la respuesta para alcanzar el primer pico del sobrepaso.

- Sobrepaso (M_p): Es el porcentaje de la desviación máxima de la salida por encima de su valor en estado estable.

- **Tiempo de asentamiento (t_s): Es el tiempo requerido por la curva de respuesta para alcanzar y mantenerse dentro de un determinado rango alrededor del valor final, habitualmente 5% ó 2%.**

En la figura 1.4 se observan las especificaciones en una curva de respuesta dinámica de un sistema de segundo orden ante una entrada escalón unitario.

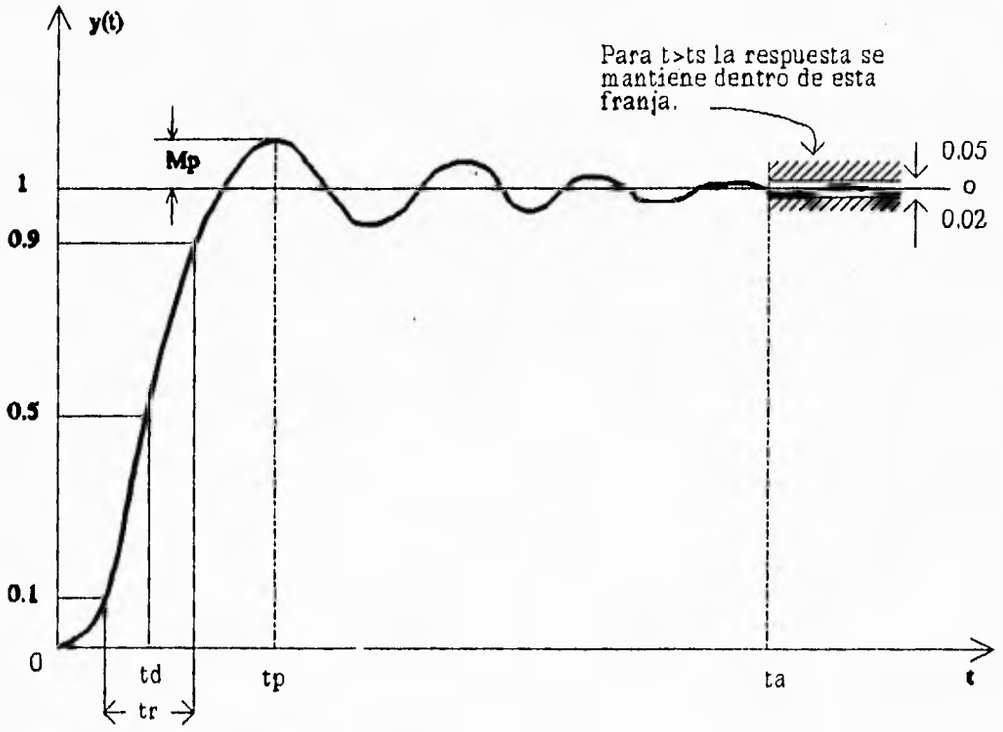


Figura 1.4 Curva de respuesta transitoria.

1.4.9 Procesamiento en paralelo

El procesamiento en paralelo puede definirse como la ejecución simultánea de instrucciones. Esto puede realizarse en muy diversas formas, entre las que destacan:

- Multiprogramación: un solo microprocesador ejecutando varios programas.

- Multiprocesamiento: cuando se ejecutan varias instrucciones en un mismo instante de tiempo.

- "Pipeline" : cuando una determinada función es dividida en pequeñas subfunciones que son procesadas en distintas unidades de hardware al mismo tiempo y posteriormente, estas unidades se unen de tal forma que juntas permitan obtener el resultado final requerido con la ventaja de que las distintas unidades de hardware se mantienen ocupadas la mayor parte del tiempo, lo cual se refleja en un incremento notable en la velocidad de procesamiento. Este tipo de procesamiento se asemeja a la operación de una línea de ensamblaje, en la cual, se tienen productos en diferentes etapas. Estos productos, están siendo manufacturados al mismo tiempo, pero en diferente etapa. De esta forma, a un producto semiterminado se le van agregando, a lo largo de la línea de producción, diferentes partes previamente manufacturadas. El proceso pipeline se asemeja a una producción en serie, en la que no es necesario terminar un producto para comenzar otro.

Sin embargo, es común encontrar sistemas en los que se utiliza una combinación de las formas antes mencionadas.

Sistemas multiprocesador.

Un sistema multiprocesador es aquel en el cual se tienen varios microprocesadores cooperando entre sí para obtener el resultado de un solo proceso. Los microprocesadores pueden comunicarse entre si a través de líneas de datos, por medio de memoria compartida, etc.

Las ventajas más importantes de un sistema multiprocesador son su alta confiabilidad y el incremento en su capacidad y velocidad de procesamiento.

II. ACCIONES BASICAS DE CONTROL

2.1 CLASIFICACION DE CONTROLADORES SEGUN SU ACCION

La teoría de control considera ciertas acciones básicas de control: la acción on-off, la acción proporcional, la acción integral y la acción derivativa. Estas acciones pueden aplicarse de manera combinada dando lugar a los controladores más populares: el controlador encendido-apagado (ON-OFF), el proporcional (P), el proporcional integral (PI) y el proporcional integral derivativo (PID). Su importancia se debe a que dan solución satisfactoria a innumerables aplicaciones en la industria.

2.2 ACCION ON-OFF

En un sistema de control on-off, el elemento accionador tiene solamente dos posiciones fijas, que en muchos casos son simplemente conectado y desconectado. Este tipo de controladores son relativamente simples y económicos, razón por la cual son ampliamente utilizados en diversos sistemas, tanto industriales como domésticos.

Sea la señal de salida del control u(t) y la señal de error actuante e(t). En un control on-off, la señal u(t) permanece en un valor máximo o mínimo, dependiendo de que la señal de error actuante sea positiva o negativa, de modo que:

$$u(t) = \begin{cases} U_1 & para \quad e(t) \geq 0 \\ U_2 & para \quad e(t) < 0 \end{cases} \qquad (2.1)$$

donde U_1 y U_2 son constantes. Generalmente el valor mínimo U_2 es o bien cero o $-U_1$.

Los controles de dos posiciones, son generalmente dispositivos eléctricos, donde habitualmente hay una válvula accionada por un solenoide eléctrico, también son comúnes los controles on-off de tipo neumático.

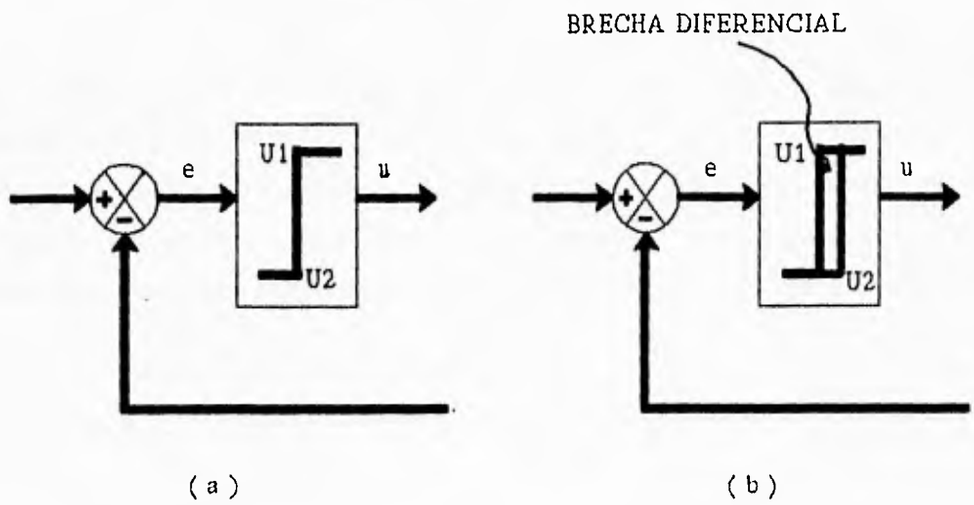


Figura 2.1 Acción de control on-off.

En la figura 2.1 a y b se presentan los diagramas de bloques de controles on-off. El rango en el que se debe desplazar la señal de error actuante antes de que se produzca la conmutación se llama brecha diferencial. En la figura 2.1b se indica una brecha diferencial. Esta brecha diferencial hace que la salida del control u(t) mantenga su valor hasta que la señal de error actuante haya pasado levemente del valor cero. La brecha diferencial se determina dependiendo de la exactitud deseada y la duración de los componentes, ya que al reducir la brecha diferencial se aumenta el número de conmutaciones por minuto y se reduce con ello la vida útil de los componentes.

2.3 ACCION PROPORCIONAL

El control proporcional, es el más simple de los algoritmos de control, tanto por la obtención

de la ecuación en diferencias que lo define, como por su implementación. La función de transferencia de un controlador proporcional continuo está dada por la ecuación:

$$G_C(s) = \frac{U(s)}{E(s)} = K_p \qquad (2.2)$$

donde:

s es la variable de Laplace.
U(s) es la transformada de Laplace de la señal de control.
E(s) es la transformada de Laplace de la señal de error.
K_p es la ganancia del controlador .

Al no existir términos en "s", la discretización es muy simple:

$$G_C(z) = \frac{U(z)}{E(z)} = K_p \qquad (2.3)$$

Despejando la variable U(z) de la función de transferencia anterior, y antitransformendo se obtiene la ecuación en diferencias de la acción de control proporcional, según se muestra a continuación:

$$u(k) = K_p\, e(k) \qquad (2.4)$$

La ecuación de este algoritmo es muy simle debido a que el contrtolador proporcional es un amplificador de la señal e(k).

En general, la aplicación de la acción proporcional aislada, no es suficiente para solucionar la mayoría de los problemas de control, entre muchas razones porque no es posible obtener un error de estado estable igual a cero y porque el tiempo de asentamiento de la respuesta del sistema es muy grande, por ello, es necesaria la aplicación de algoritmos más elaborados.

2.4 ACCION PROPORCIONAL INTEGRAL

La acción proporcional integral requiere un algoritmo ligeramente más complejo que el requerido por la acción proporcional, pero ofrece mayores beneficios. La introducción de un controlador PI en un lazo de control, tiene como ventaja principal, la eliminación del error de estado estable, para cambios en la referencia y eventuales perturbaciones aditivas constantes. Sin embargo, la introducción del término integral produce disminución en los márgenes de estabilidad.

El controlador PI analógico en el dominio de la variable de Laplace, se define comúnmente por medio de la función de transferencia que se enuncia a continuación:

$$G_c(s) = \frac{U(s)}{E(s)} = K_p\left(1 + \frac{1}{T_i s}\right) \qquad (2.5)$$

donde, además de los términos enunciados respecto a la ecuación 2.2, se tiene el parámetro T_i, el cual se define como constante de tiempo de integración.

En la función de transferencia anterior es posible identificar claramente a las componentes que corresponden a la acción proporcional (K_p) y a la acción integral ($1/T_i s$).

La aproximación discreta de estos términos se efectúa por medio del método de "diferenciación hacia adelante", de donde resultan las expresiones discretas 2.6 y 2.7. La suma de ambos términos conducen a la función de transferencia discreta de la acción PI (ecuación 2.8).

acción proporcional: $$K_p \qquad (2.6)$$

acción integral: $$\frac{K_p T}{T_i}\,\frac{z^{-1}}{(1 - z^{-1})} \qquad (2.7)$$

$$G_c(z) = \frac{U(z)}{E(z)} = \frac{K_p T_i (1-z^{-1}) + K_p T z^{-1}}{T_i (1-z^{-1})} \qquad (2.8)$$

Despejando U(z) de la ecuación anterior y antitransformando se obtiene la ecuación en diferencias de la acción proporcional integral, misma que se muestra a continuación:

$$u(k) = u(k-1) + K_p\, e(k) + K_p \left(\frac{T}{T_i} - 1 \right) e(k-1) \qquad (2.9)$$

2.5 ACCION PROPORCIONAL INTEGRAL DERIVATIVA

La combinación de los efectos de acción proporcional, acción de control integral y acción de control derivativa, se llama acción de control PID (proporcional integral derivativa). Esta acción, conjunta las ventajas de cada una de las tres acciones de control individuales. Además, presenta ventajas sobre otros algoritmos, mejorando la precisión y la estabilidad del sistema en malla cerrada. "La retroalimentación proporcional reduce los errores, pero la alta ganancia puede desestabilizar el sistema. El control integral mejora el error de estado estable y proporciona robustez con respecto a la variación de los componentes del sistema, pero también reduce la estabilidad. El control diferencial aumenta la amortiguación y mejora la estabilidad. La combinación de los tres, que como ya se dijo, da lugar al controlador PID, es omnipresente en el control de procesos industriales y es el ingrediente básico en virtualmente todos los sistemas de control."[1]

El controlador PID puede tener alguna de las estructuras siguientes:

a) Ideal

[1] FRANKLIN, POWELL, Control de Sistemas Dinámicos con Retroalimentación, Pág. 115.

b) Clásica

c) Parámetros independientes

d) Industrial

a) Estructura ideal.

La expresión 2.10 enunciada a continuación describe al controlador PID ideal en su forma continua, en términos de "s". Consiste en la superposición de los operadores elementales cuyos efectos en conjunto aportan las ventajas particulares de cada uno de ellos.

$$G_c(s) = \frac{U(s)}{E(s)} = K_p \left(1 + \frac{1}{T_i\, s} + T_d\, s\right) \qquad (2.10)$$

donde:

K_p : Es la constante del modo proporcional.

T_i : Es la constante de tiempo del modo integral.

T_d : Es la constante de tiempo derivativa.

En esta función de transferencia es posible identificar los términos que corresponden a la acción proporcional (K_p), a la acción integral ($1/T_i s$) y a la acción derivativa ($T_d s$).

b) Estructura clásica

La estructura clásica está caracterizada por la siguiente ecuación:

$$G_c(s) = K_p \left(1 + \frac{1/T_i}{s}\right)\left(\frac{1 + T_d s}{1 + T_a s}\right) E(s) \qquad (2.11)$$

c) Estructura de parámetros independientes

Esta estructura queda definida mediante la ecuación:

$$U(s) = K_p \left(1 + \frac{1/T_i}{s}\right) E(s) - \left(\frac{T_d s}{T_a s + 1}\right) Y(s) \qquad (2.12)$$

d) Estructura industrial

La estructura industrial queda definida a través de la siguiente ecuación:

$$U(s) = K_p \left(1 + \frac{1/T_i}{s}\right) \left(R(s) - \left(\frac{T_d s + 1}{T_a s + 1}\right) Y(s) \right) \qquad (2.13)$$

En los tres casos anteriores:

K_p : Es la constante del modo proporcional

T_i : Es la constante de tiempo del modo integral

T_d : Es la constante de tiempo derivativa

T_a : Es la constante de tiempo del filtro que se observa en la ecuación y está definida como: $T_a = T_d/N$, donde $3 \leq N \leq 20$.

Una forma de obtener la aproximación discreta de la acción PID para la estructura ideal puede ser empleando el método de "diferenciación hacia adelante" para la acción integrativa y la "diferenciación hacia atrás" para la acción derivativa, resultando las equivalencias siguientes:

	Forma continua	Forma discreta
acción proporcional:	K	K
acción integral:	$\frac{K}{T_i s}$	$\frac{\alpha z^{-1}}{(1 - z^{-1})}$
acción derivativa:	$KT_d s$	$\frac{1}{\beta}(1 - z^{-1})$

Los parámentros, α y β se definen por las expresiones:

$$\alpha = \frac{KT}{T_i} \qquad \frac{1}{\beta} = \frac{KT_d}{T} \tag{2.14}$$

La ecuación en diferencias total que define al algoritmo PID se obtiene adicionando las componentes enunciadas y antitransformando, lo cual se desarrollará con detalle en el capítulo correspondiente al modelado matemático del sistema.

Para el desarrollo del presente proyecto se empleará la estructura ideal del controlador PID por considerarse ésta adecuada para cumplir con la función de control que se requiere, adicionandole un filtro paso bajas en la parte derivativa, empleado comúnmente en diseños industriales, básicamente, con el fin de hacer físicamente ralizable la acción de control derivativa dentro de la ecuación que define el algoritmo PID. A continuación se muestra la ecuación resultante (2.15).

$$G_c(s) = \frac{U(s)}{E(s)} = K_p\left(1 + \frac{1}{T_i\, s} + \frac{T_d\, s}{T_a s + 1}\right) \qquad (2.15)$$

donde:

K_p : Es la constante del modo proporcional

T_i : Es la constante de tiempo del modo integral

T_d : Es la constante de tiempo derivativa

T_a : Es la constante de tiempo del filtro y está definida como: $T_a = T_d/N$, donde $3 \leq N \leq 20$.

III. MOTOR DE CORRIENTE DIRECTA

3.1 CONCEPTOS BASICOS

Un motor de corriente directa es un dispositivo que transforma energía eléctrica en energía mecánica. Hay básicamente cuatro tipos de motores de corriente directa:

- Motor con excitación independiente.

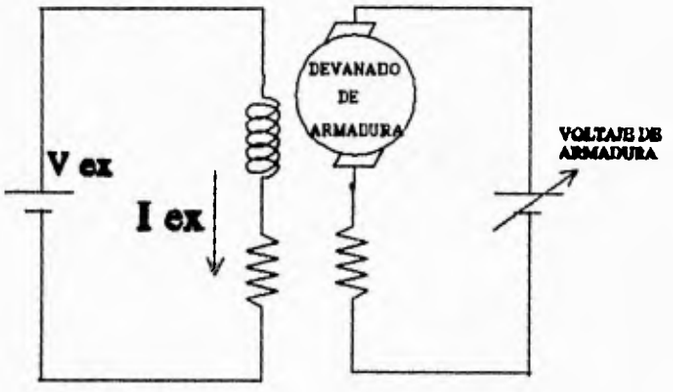


Figura 3.1 Motor con excitación independiente.

- **Motor con excitación en derivación**

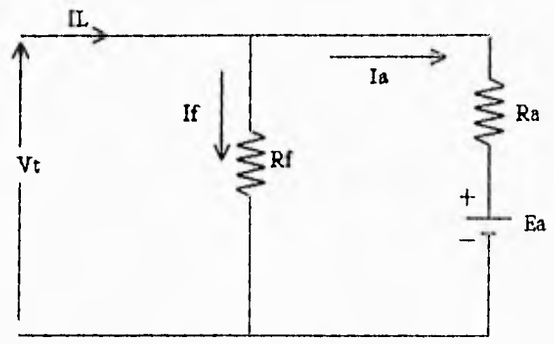


Figura 3.2 Motor con excitación en derivación.

- **Motor compuesto acumulativo**

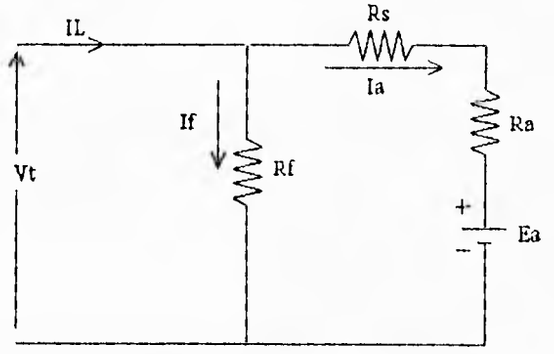


Figura 3.3 Motor compuesto acumulativo.

Motor con excitación en serie

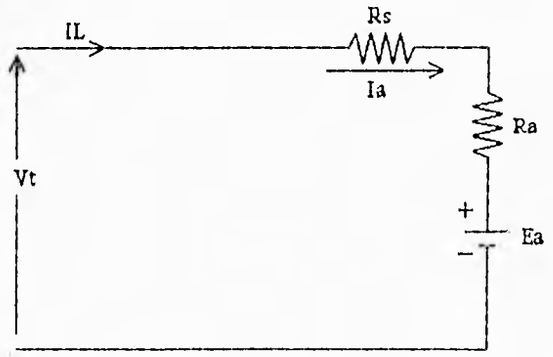


Figura 3.4 Motor con excitación serie.

El motor compuesto se califica con la palabra acumulativo con el fin de destacar que las conexiones al campo en serie son tales que garantizan que el flujo del campo en serie ayude al flujo del campo en derivación. El motor en serie encuentra un campo de aplicaciones muy amplio, en particular en cargas de tracción.

El funcionamiento del motor de corriente directa que opere en forma serie, compuesto acumulativo o en derivación, puede describirse en términos de un circuito equivalente y un conjunto de ecuaciones de funcionamiento que a continuación se presentan. Imponiendo algunas restricciones en el circuito de la figura 3.3, y en las ecuaciones principales para el motor de CD, se obtiene el circuito equivalente y las ecuaciones correctas para el modo de operación deseado. Por lo que para un motor en serie, el circuito equivalente apropiado resulta de eliminar R_f del circuito de la figura 3.3 y por consiguiente de las ecuaciones que describen al motor. De igual forma para un motor en derivación, basta con eliminar Rs. Y finalmente para un motor compuesto acumulativo no hay que hacer ningún ajuste. Sin embargo, para un motor con excitación independiente es necesario profundizar más en el análisis de sus ecuaciones ya que su diagrama de conexiones es distinto al

esquema general que se ha propuesto, por lo que este tipo de motor se analizará posteriormente con más detalle.

3.1.1 Ecuaciones principales de un motor de CD

$$E_a = K_E \Phi n \tag{3.1}$$

$$T = K_T \Phi I_a \tag{3.2}$$

$$V_t = E_a + I_a (R_a + R_s) \tag{3.3}$$

$$I_L = I_f + I_a \tag{3.4}$$

Donde:

E_a Es el valor promedio de la fem inducida en el devanado de armadura o fuerza contraelectromotriz

I_a Corriente de armadura. (Corriente total que entra o sale por las escobillas)

ϕ Flujo en el entrehierro (incluyendo el efecto del devanado de campo en derivación y el devanado de campo en serie)

$$\phi = \phi_a + \phi_{sh} \tag{3.5}$$

n Velocidad del motor en RPM

T Par electromagnético desarrollado por el motor.

V_t Voltaje aplicado o voltaje fuente.

I_L Corriente proveniente de la fuente de voltaje.

I_f Corriente que fluye por el devanado en derivación.

R_a Resistencia del circuito de armadura mas la resistencia de la unión con las escobillas de carbón.

R_s Resistencia del circuito de campo en serie.

R_f Resistencia del campo en derivación.

K_E es una constante del devanado definida como:

$$K_E = \frac{p\,Z}{60\,a} \tag{3.6}$$

K_T es la constante del par definida como:

$$K_T = \frac{p\,Z}{2\,\pi\,a} \tag{3.7}$$

Para las ecuaciones 3.6 y 3.7:

p = No. de polos.

Z = No. de conductores

a = No. de trayectorias en paralelo.

Nótese que para las configuraciones en serie, en derivación o compuesto acumulativo, al variar el voltaje V_t en las terminales, el flujo ϕ también es variable, lo que implica que en las ecuaciones de la 3.1 a la 3.3 se presenten no linealidades.

3.1.2 Motor con excitación independiente

Este motor es el mismo que se fabrica para usarse en derivación, solamente que en esta modalidad se usa una fuente de pequeña capacidad y voltaje constante como fuente de excitación constante, y una fuente de la capacidad de la armadura y de voltaje controlado para alimentar a

esta otra parte, como se aprecia en la figura 3.1.

Algunos fabricantes de motores, sustituyen el circuito eléctrico de exitación constante, por un imán permanente, con el fin de ahorrar energía.

En la figura 3.1, se tiene el circuito equivalente de un motor con excitación independiente, alimentándose con un voltaje de CD tal que éste tendrá una corriente de excitación constante, la cual a su vez producirá un flujo de excitación constante. De acuerdo con esta figura y tomando en cuenta las ecuaciones básicas del motor expuestas anteriormente, se pueden hacer algunas reducciones que traen como consecuencia que las relaciones entre las variables sean lineales. De forma que las ecuaciones de la 3.1 a la 3.4 se transforman en:

$$E_a = K_{E'}\, n \tag{3.8}$$

$$T = K_{T'}\, I_a \tag{3.9}$$

$$V_t = E_a + I_a\ (\ R_a\ ) \tag{3.10}$$

$$I_L = I_a \tag{3.11}$$

donde:

$$K_{E'} = K_E\, \phi$$

$$K_{T'} = K_T\, \phi$$

3.2 RELACION PAR-VELOCIDAD DEL MOTOR DE CORRIENTE DIRECTA

La relación existente entre el par y la velocidad en un motor de corriente directa es muy estrecha, y esta relación puede ser observada fácilmente mediante el análisis de las ecuaciones expuestas anteriormente. Estas ecuaciones pueden ser analizadas para cualquier tipo de motor de CD, a continuación se realizará un análisis para un motor en derivación, sin embargo, un proceso de razonamiento parecido puede ser aplicado a los otros.

$$T = K_T \phi I_a \qquad (3.12)$$

$$V_t = E_a + I_a R_a \qquad (3.13)$$

$$E_a = K_E \phi n \qquad (3.14)$$

sustituyendo 3.14 en 3.13 y despejando I_a:

$$I_a = \frac{Vt - K_E \phi n}{R_a} \qquad (3.15)$$

Cuando no hay carga aplicada a la flecha, el único par necesario es el que repone las pérdidas por rotación. Suponiendo que el voltaje de alimentación Vt sea constante, el motor en derivación opera a flujo constante por lo que la ecuación 3.12 indica que solo se requiere una pequeña corriente de armadura comparada con su valor nominal para subsanar dichas pérdidas. La ecuación 3.15 revela la manera según la cual la corriente de armadura debe ajustarse al valor correcto para satisfacer la ecuación 3.12. En esta expresión V_t, R_a, K_E y ϕ tienen valores fijos, por lo tanto la velocidad es la variable crítica. Suponiendo que para un determinado tiempo el par aplicado a la flecha (fricción mecánica, cargas aplicadas o el viento) aumente, la ecuación 3.12 indica que para mantener el equilibrio entre el par eléctrico y el par aplicado a la flecha, la corriente de armadura (Ia) debe aumentar. Esto implica una disminución en la velocidad, según lo muestra la

ecuación 3.15. En caso contrario, si el par aplicado a la flecha disminuye, la corriente de armadura tendería a reducir su valor para mantener el equilibrio en la ecuación 3.12 y consecuentemente la velocidad en la flecha del motor aumentaría, de acuerdo con la ecuación 3.15.

Otra forma de ver esta relación tan estrecha es considerando el caso de que un motor se encuentre trabajando a una cierta velocidad y súbitamente se le aplique una carga a su flecha, que le demande el par nominal. Está claro que, dado que el par eléctrico desarrollado en ese instante es tan solo suficiente para reponer las pérdidas por fricción mecánica y del aire, no lo es para satisfacer el par de carga, entonces la primera reacción del motor consiste en disminuir su velocidad. Posteriormente, como lo expresa la ecuación 3.15, la corriente de armadura se incrementa de modo que el par eléctrico pueda también incrementarse como lo indica la ecuación 3.12. De hecho, el par de la carga aplicada hace que el motor adopte una determinada velocidad, lo cual exige una corriente suficiente para producir un par eléctrico desarrollado que permita un equilibrio con el par aplicado a la flecha y el par de fricción. Por lo tanto se alcanza así el balance de potencia, donde la potencia eléctrica $E_a I_a$ es igual a la potencia mecánica desarrollada $T\omega_m$.

Basándose en el análisis anterior, resulta evidente que la curva par-velocidad de los motores de CD es una característica importante. En la figura 3.8 se observa un bosquejo de las curvas generales que muestran las características par-velocidad que corresponden a los motores en derivación, compuestos acumulativos y en serie para el caso en el cual se realice un ajuste del par y la velocidad del motor por medio del reóstato mostrado en las figuras 3.5, 3.6 y 3.7. Para facilitar la comparación, las curvas se han trazado sobre un punto común de par y velocidad nominales. Se puede demostrar por medio de un análisis mas detallado, la razón por la cual, las curvas adoptan las formas y posiciones relativas representadas en la figura 3.8.

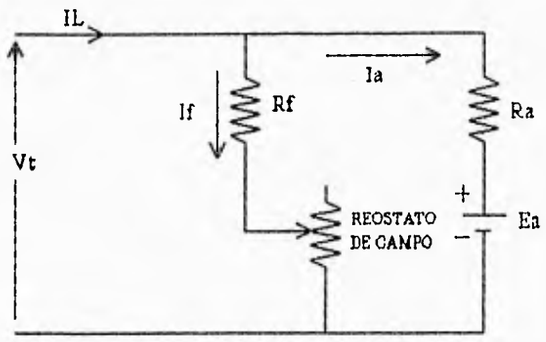


Figura 3.5 Motor en derivación, haciendo uso de un reóstato de campo para controlar su velocidad.

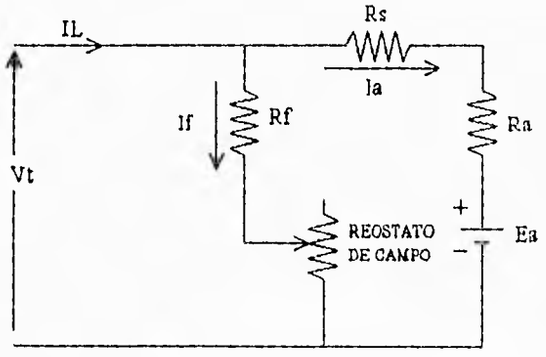


Figura 3.6 Motor compuesto acumulativo haciendo uso de un reóstato para controlar su velocidad.

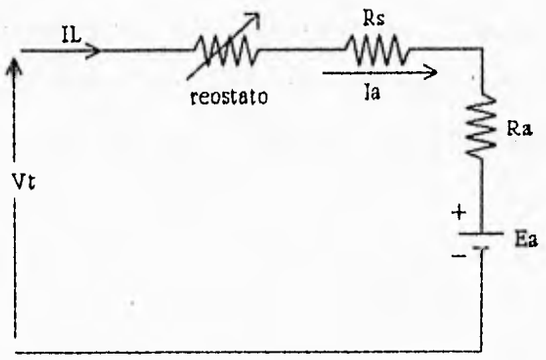


Figura 3.7 Motor serie haciendo uso de un reóstato para controlar su velocidad.

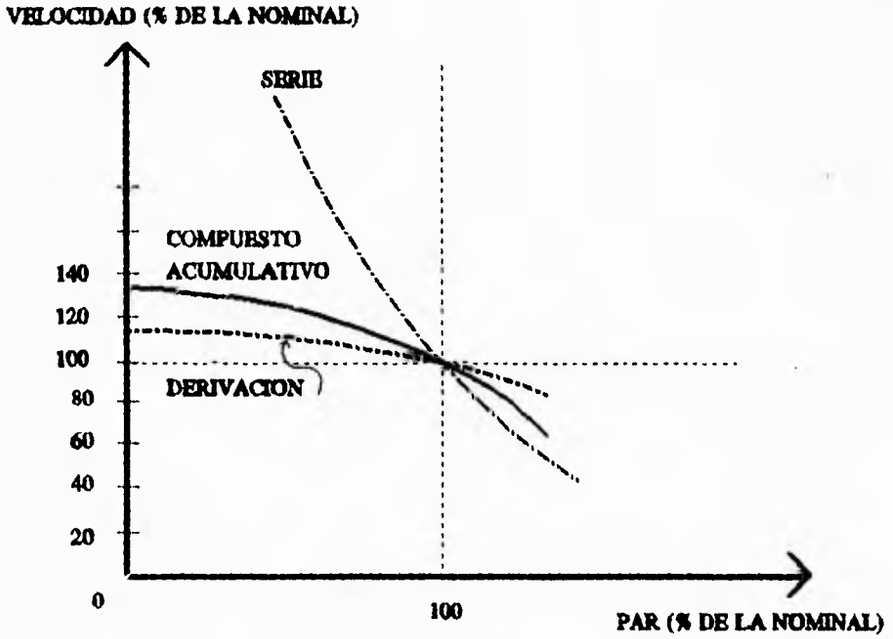


Figura 3.8 Curvas características PAR-VELOCIDAD.

Tratándose del motor con excitación independiente, el análisis de su curva par-velocidad se hará con un mayor detenimiento, por ello y para una mejor comprensión, es conveniente agregar al grupo de ecuaciones básicas planteadas para este tipo de motor, una mas:

$$T = T_m + T_r \qquad (3.16)$$

El par eléctrico es igual a la suma del par mecánico T_m más el par debido a las pérdidas rotacionales T_r.

Sustituyendo la ecuación 3.8 en la ecuación 3.10 se tiene:

$$V_t = K_{E'}\, n + I_a\ (\ R_a\ ) \qquad (3.17)$$

y despejando n:

$$n = \frac{V_t - R_a\ I_a}{K_{E'}} \qquad (3.18)$$

despejando la de 3.9:

$$I_a = \frac{T}{K_{T'}} \qquad (3.19)$$

sustituyendo la ecuación 3.16 en la ecuación 3.19:

$$I_a = \frac{T_m + T_r}{K_{T'}} \qquad (3.20)$$

sustituyendo la ecuación 3.20 en la ecuación 3.18:

$$n = \frac{V_t - R_a\ \dfrac{(\ T_m + T_r\ )}{K_{T'}}}{K_{E'}} \qquad (3.21)$$

reordenando se obtiene:

$$n = - \frac{R_a}{K_T \, K_E} T_m + \frac{K_T \, V_t - R_a \, T_r}{K_E \, K_T} \qquad (3.22)$$

Tomando en cuenta que para este análisis V_t se considera constante, al igual que Tr, se puede ver que la ecuación 3.22 es de la forma y= mx + b, graficando se obtiene la curva par-velocidad para un motor con excitación independiente (ver figura 3.9).

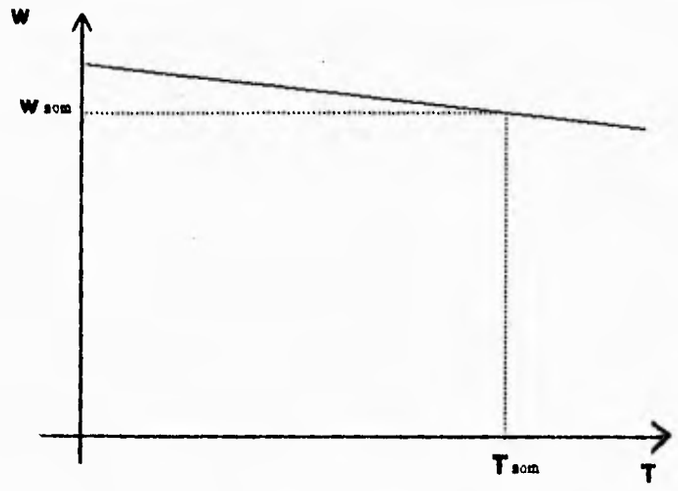


Figura 3.9 Curva PAR-VELOCIDAD para un motor con excitación independiente.

3.3 ECUACIONES DE VELOCIDAD

Al analizar la ecuación de velocidad para cada uno de los motores, resulta:

Para el motor en derivación la ecuación de velocidad se puede escribir como:

$$n = \frac{E_a}{K_E \phi_{sh}} = \frac{V_t - I_a R_a}{K_E \phi_{sh}} \qquad (3.23)$$

Las únicas variables implícitas son la velocidad n y la corriente de armadura I_a. Al par nominal de salida, la corriente de armadura, así como la velocidad están a su valor nominal. Al quitar el par de carga, la corriente de armadura se hace en correspondencia mas pequeña, haciendo mayor el numerador de la ecuación 3.23. Esto da por resultado velocidades mas elevadas.

La ecuación de la velocidad que se aplica al motor compuesto acumulativo es:

$$n = \frac{E_a}{K_E \phi} = \frac{V_t - I_a (R_a + R_s)}{K_E (\phi_{sh} + \phi_s)} \qquad (3.24)$$

La comparación con la expresión análoga del motor en derivación arroja dos diferencias. Una, que el término del numerador, también incluya la caída de voltaje en el devanado de campo serie, aunada a la de la armadura. Dos, el término del denominador se incrementa para tomar en cuenta el efecto del flujo ϕ_s del campo en serie.

La ecuación de la velocidad del motor en serie es:

$$n = \frac{E_a}{K_E \phi_s} = \frac{V_t - I_a (R_a + R_s)}{K_{EE} (I_a)} \qquad (3.25)$$

donde K_{EE} es un nuevo factor de proporcionalidad que permite reemplazar a ϕ_s por la corriente de armadura I_a.

Para obtener par nominal, se requiere de una corriente nominal, lo que se traduce en un flujo abundante. Sin embargo, al quitar el par de carga, fluye menos corriente de armadura. Tomando

en cuenta que I_a aparece en el denominador de la ecuación de velocidad, es fácil observar que la velocidad tendrá un gran crecimiento. De hecho si se desconectara la carga de la flecha del motor, resultarían velocidades peligrosamente elevadas debido a lo pequeño de la corriente de armadura. Las fuerzas centrífugas a estas velocidades pronto dañarían el devanado de armadura. Por esta razón, un motor en serie nunca debe desconectarse de su carga. También es importante advertir que cuando el motor en serie reacciona para desarrollar pares elevados, su velocidad disminuye. Estas características son las que se aprovechan cuando se usan motores en serie con cargas de tracción.

Para el caso del motor con excitación independiente, la ecuación de velocidad es:

$$n = \frac{1}{K_{E'}} V_t - \frac{R_a I_a}{K_{E'}} \qquad (3.26)$$

donde: $K_{E'} = K_E\phi$, dado que ϕ permanece constante.

La ecuación 3.26 es de la forma y = mx + b, y por lo tanto representa una dependencia lineal de n con respecto a Vt. Esta linealidad es lo que hace que el motor con excitación independiente sea el óptimo para cuando se requiere tener un control preciso de la velocidad en un rango amplio.

3.4 CONTROL DE LA VELOCIDAD EN UN MOTOR DE CD

Una de las ventajas que el motor de corriente directa ofrece, es la facilidad con la que su velocidad puede controlarse. De las ecuaciones de velocidad obtenidas anteriormente para los motores serie, en derivación y compuesto acumulativo, puede deducirse la siguiente ecuación, haciendo una modificación de la ecuación 3.24:

$$n = \frac{V_t - I_a \left(R_a + R_e \right)}{K_E \phi} \qquad (3.27)$$

La modificación consiste en la sustitución de R_a por una resistencia externa R_e en el circuito de armadura. Al observar la ecuación anterior puede verse que la velocidad puede controlarse ajustando cualquiera de los tres factores que aparecen en el segundo miembro de la ecuación: Vt, Re o ϕ.

Lo mas simple es ajustar ϕ mediante un reostato de campo como el que se ve en las figuras 3.5 y 3.6. Si se incrementa la resistencia del reóstato de campo, disminuye el flujo en el entrehierro, elevando la velocidad de operación de acuerdo con la ecuación 3.27. Los motores en derivación para propósito general se diseñan para dar un 200% de incremento sobre la velocidad nominal con este método de control de la velocidad. Sin embargo, debido al debilitamiento del flujo, el par permisible que puede entregarse a mayor velocidad se reduce en forma correspondiente, con el fin de evitar una corriente excesiva de armadura.

Un segundo método de ajuste de la velocidad involucra el uso de un resistor externo Re conectado en el circuito de armadura, como se ilustra en la figura 3.10. El tamaño y el costo de este resistor son bastante mayores a los del reóstato de campo porque Re debe ser capaz de manejar la plena corriente de armadura. La ecuación 3.27 expresa que mientras mayor se haga Re, mayor será el cambio en la velocidad. Es frecuente seleccionar al resistor externo para proporcionar cuando mucho un 50% de la caída de velocidad con respecto a la nominal. La desventaja principal de este método de control consiste en la baja eficiencia de operación. Por ejemplo, cuando se obtiene una caída de 50% en la velocidad, se tiene aproximadamente la mitad del voltaje Vt aplicado a Re . En consecuencia, casi el 50% de la potencia de la línea de entrada se disipa en forma de calor en el resistor Re. No obstante, el circuito de control por resistencia en el circuito de la armadura se emplea a menudo en particular para los motores en serie.

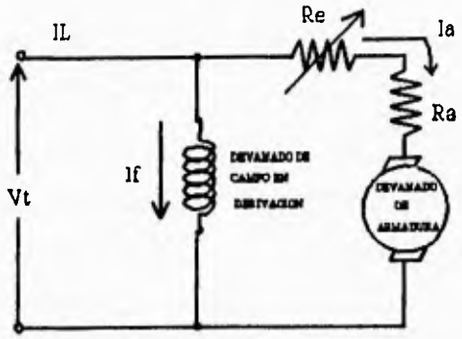


Figura 3.10 Ajuste de la velocidad por medio de un resistor externo.

Un tercer método de control de la velocidad involucra el ajuste del voltaje aplicado a las terminales. Este esquema es el mas deseable desde el punto de vista de la flexibilidad y la alta eficiencia de la operación. Pero también es uno de los más costosos porque requiere su propio suministro de CD. Esto implica la compra de un motor-generador con una capacidad por lo menos igual a la del motor que debe ser controlado. Por lo general esto se justifica únicamente cuando es indispensable tener una operación de muy alta calidad.

En el sistema Ward-Leonard, se emplea este método, controlando la velocidad por medio de la variacion de la corriente de excitacion del generador de CD que alimenta al motor de CD.

Control de velocidad por medios electrónicos.

El desarrollo de los tiristores y transistores con elevados valores de su corriente y voltaje de operación, ha permitido que el control de la velocidad de los motores de corriente directa, mediante el ajuste del voltaje en las terminales sea mas factible. Es conveniente mencionar que el control de velocidad por los medios electrónicos que serán mencionados fue pensado para aplicarse

a motores con excitación independiente y con excitación serie.

Un método frecuentemente utilizado para controlar la velocidad de un motor, es mediante el uso de SCR's, los cuales por sus características de funcionamiento, permiten un ahorro de energía. Las altas corrientes que requieren las armaduras se obtienen de la red de corriente alterna, a través de estos rectificadores controlados de silicio, los cuales se pueden emplear para hacer una rectificación de media onda o de onda completa del voltaje de corriente alterna. Estos dispositivos reciben el nombre de controlados porque aún estando en condiciones de conducir corriente, no lo hacen sino hasta el momento de recibir una señal en su compuerta. Una vez recibida la señal, el rectificador continúa conduciendo hasta que la corriente alterna llega a su valor nulo, y de ahí en adelante impide la conducción en sentido negativo. La señal se puede suministrar en el momento que se inicia la semi-onda positiva, en cuyo caso la conduce íntegramente, o se puede retrasar para que conduzca solamente una fracción de la onda. En el control de velocidad para motores, el control se reduce a utilizar el SCR para suministrar voltaje a las terminales del motor, así, basta con variar el instante en que se le aplica la señal a la compuerta variando consecuentemente el voltaje promedio que recibe el motor. Tomando en cuenta que la rectificación de onda completa, requiere de una circuitería de conmutación muy compleja, normalmente se utiliza la rectificación de media onda, en la cual el rango de control de la velocidad está limitado a una variación de 0 a menos de 1/2 de la velocidad nominal.

El uso de los transistores de efecto de campo de potencia (TMOS) así como de los interruptores apagados por compuerta (GTO) se ha hecho muy común dentro del campo de control debido a sus amplias facilidades de encendido y apagado.

El TMOS, se utiliza frecuentemente en motores cuya corriente nominal sea menor que 10A. Este dispositivo es empleado en circuitos *choppers* del voltaje de alimentación al motor, para controlar la velocidad del mismo mediante un control del ciclo de trabajo de la señal aplicada a la compuerta del TMOS. Basta con que se aplique un nivel de voltaje mayor de un umbral en la compuerta, para poner al TMOS en conducción, y por otra parte, llevando el nivel de voltaje en la

compuerta a cero, el TMOS entra en estado de corte. Así, la velocidad de un motor es controlada con una lógica mas simple, que en el caso del *chopper* con SCR, pues se tiene un control directo sobre el apagado del TMOS.

Para el caso en que se tenga que controlar un motor cuya corriente nominal sea mayor de 10A, existen tiristores llamados interruptores controlados por compuerta (GTO), con los cuales es relativamente sencillo realizar un *chopper* con el que sea posible controlar la velocidad del motor por medio del control del ciclo de trabajo de la señal en la compuerta. Para poner al GTO en conducción, es necesario aplicarle un pulso positivo de voltaje en la compuerta, y para apagarlo, un pulso negativo. La señal en la compuerta, puede ser producida por un circuito de electrónica de relativamente bajo nivel de potencia.

Adicionalmente a los métodos anteriores, no debemos olvidar que una de las formas más precisas y fáciles de implementar el control de la velocidad, consiste en disponer de un voltaje directamente proporcional a la velocidad del motor, una vez que se tiene esta señal analógica, es posible manipularla en forma lineal y finalmente darle la corriente necesaria para que ésta se convierta en el voltaje de entrada a las terminales de armadura del motor. Como sabemos, una forma de darle corriente a una señal analógica de voltaje, en forma lineal, es empleando transistores con las configuraciones requeridas según las características del motor a controlar.

En este proyecto, se pretende construir un controlador PID para un motor de corriente directa con excitación independiente, en el cual, como se expuso anteriormente, se tiene una dependencia lineal de la velocidad con respecto al voltaje aplicado en las terminales.

Los datos de placa del motor son los siguientes:

MICRO SWITCH MOTOR

DC CONTROL MOTOR

No. en catalogo:	22VM52-020-4
Resistencia en las terminales:	1.2 OHMS
Voltaje nominal:	10V
Corriente nominal:	2.1 A

Debido a que nuestro objetivo con este motor es representar a un motor "industrial", decidimos añadirle un disco de acero en el eje con la finalidad de aumentar la inercia de nuestro prototipo, para apegarnos lo más posible a la realidad.

Para la implementación de este proyecto, en el cual se utilizará control digital, se pensó en hacer uso de tiristores para controlar el voltaje promedio en las terminales, seccionando mediante un *chopper* el voltaje proveniente de una fuente de CD, pero después de algunas pruebas experimentales con circuitos *chopper* que nosotros mismos diseñamos y de los cuales se hablará con más detalle en el capítulo de hardware, llegamos a la conclusión de que este procedimiento no es el óptimo. La justificación de este argumento es que toda la teoría de control digital se basa en sistemas lineales, y en el caso en que un circuito *chopper* alimente a un motor que por sus características produzca una fuerza contra-electromotriz grande (como es el caso de los motores que operan a altas velocidades), la fuerza contra-electromotriz, que siempre está presente en las terminales del motor y que a grandes velocidades es muy significativa, provoca que el circuito *chopper* en conjunto con el motor no presente una respuesta lineal.

3.5 USOS DE LOS MOTORES DE CD

Los motores de corriente directa son esenciales en la actualidad, se emplean en bombas centrífugas, ventiladores, engranes, bandas transportadoras, grúas, elevadores, escaleras eléctricas, en procesos químicos para centrifugar ciertas sustancias, en el proceso de fabricación de fibra óptica, en las industrias manufactureras, en empacadoras, en las industrias papeleras, etc.

Como se ve, en la industria se le dan usos muy diversos al motor de corriente directa, y éstos se ven incrementados si se toma en cuenta su facilidad de control, ya que no es fácil encontrar esta característica en otros dispositivos de conversión electromecánica. El motor de corriente directa ofrece una escala muy vasta de control de la velocidad.

De esta forma los controladores de velocidad hacen posible que cada día, los motores de corriente directa tengan más actividad en la industria. Antiguamente, por la falta de controles precisos de velocidad, se utilizaban procesos que compensaban los defectos en el control de velocidad de los motores. Un ejemplo de esto es el control sobre el bombeo de fluidos, en donde debido a la falta de control sobre la velocidad de las bombas centrífugas, se emplea un proceso llamado "recirculación de flujo" en el que un porcentaje del fluido es desviado y regresado al punto de bombeo. Actualmente, muchos de estos procesos continúan en operación, pero sin duda alguna el control de velocidad ofrece otras alternativas.

Un ejemplo del uso del control de velocidad, es en bandas transportadoras, en donde debido a los avances en la automatización de procesos, es imprescindible el contar con una velocidad previamente establecida, esto permite que la banda transportadora se pueda sincronizar con otros procesos.

El control de velocidad, encuentra un amplio campo de aplicación en procesos en donde independientemente de la variación en la carga del motor, se pretenda conservar la velocidad dentro de un rango de tolerancia, el cual podría ser muy pequeño, dependiendo del tipo de controlador que

se utilice. Tal es el caso en elevadores, escaleras eléctricas, y en algunos procesos de centrifugación como en la fabricación de fibra óptica, en el cual el control de la velocidad de centrifugado es un factor determinante en el tiempo y la calidad de la fabricación. En el caso de la fabricación de fibra óptica, así como en otros procesos industriales delicados, el control realimentado de velocidad encuentra su principal área de aplicación.

En algunas industrias embotelladoras y empacadoras se utiliza un control de velocidad en las bandas, lo que permite un ajuste de la velocidad dependiendo de la demanda del producto en el mercado.

En los párrafos anteriores se mencionaron solamente algunos de los múltiples usos que a los motores de corriente directa se les han dado a lo largo de la historia por su facilidad de control de su velocidad y es de esperarse que al existir controladores mas precisos, pueden confiarse a este tipo de motores otras actividades, tantas como la imaginación del hombre lo permita.

IV. EL MICROCONTROLADOR MC68HC11F1

4.1 INTRODUCCION

El microcontrolador MC68HC11F1 es un miembro de la familia de unidades microcontroladoras MC68HC11 de Motorola. Para la construcción del circuito se emplea la tecnología CMOS de alta densidad (HCMOS).

Este dispositivo fue desarrollado en un principio para trabajar en modo expandido no multiplexado (aunque puede trabajar en otros modos) y con ello cubrir la gran demanda que se tenía para muy diversas aplicaciones.

Es un circuito integrado de gran capacidad, en el cual se combina un tamaño pequeño, velocidades de operación elevadas, una demanda baja de potencia, gran inmunidad al ruido, etc. Además, su set de instrucciones combinado con sus 6 modos de direccionamiento, las operaciones aritméticas de 16 bits que puede realizar así como su capacidad de almacenamiento y sus circuitos adicionales, lo hacen ser un dispositivo muy versátil y de amplio uso por su bajo costo, y al mismo tiempo, estas características lo hacen formar parte básica de muchas aplicaciones.

4.2 CARACTERISTICAS

- 1024 Bytes de RAM
- 512 bytes de EEPROM
- Capacidad de direccionamiento de hasta 64 Kbytes
- Cuatro chip-selects programables
- Mecanismo de protección de bloques para EEPROM y CONFIG (registro BPROT)
- 7 Puertos paralelos de entrada/salida: PA, PB, PC, PD, PE, PF y PG
- 8 canales para conversión Analógico/Digital de 8 bits
- Puerto Serie Asíncrono

- Puerto Serie Síncrono de alta velocidad
- Sistema temporizador de 16 bits
- Preescalador programable de 4 etapas
- Tres entradas para captura/cinco salidas de comparación, o bien,
- Cuatro entradas para captura/cuatro salidas de comparación
- Circuito Acumulador de Pulsos de 8 bits
- Circuito de interrupciones en tiempo real
- Sistema "perro guardián" para garantizar una operación correcta del microcontrolador
- Encapsulado PLCC de 68 terminales

En la figura 4.1 se muestra el diagrama a bloques del microcontrolador 68HC11F1, en ella se pueden ver en forma esquemática cada una de las características antes mencionadas.

4.3 ANALISIS DE LA CPU

La CPU (Unidad Central de Procesos) del microcontrolador MC68HC11 es responsable de ejecutar todas las instrucciones de software en la secuencia programada. Esta versión del microcontrolador es capaz de ejecutar todas las instrucciones del M6800 y del M6801 y permite la ejecución de más de 90 nuevos códigos de operación.

4.3.1 Modelo de Programación

La figura 4.2 muestra una representación gráfica de los registros internos de la CPU del MC68HC11, lo cual constituye su modelo de programación. A continuación se describen estos registros.

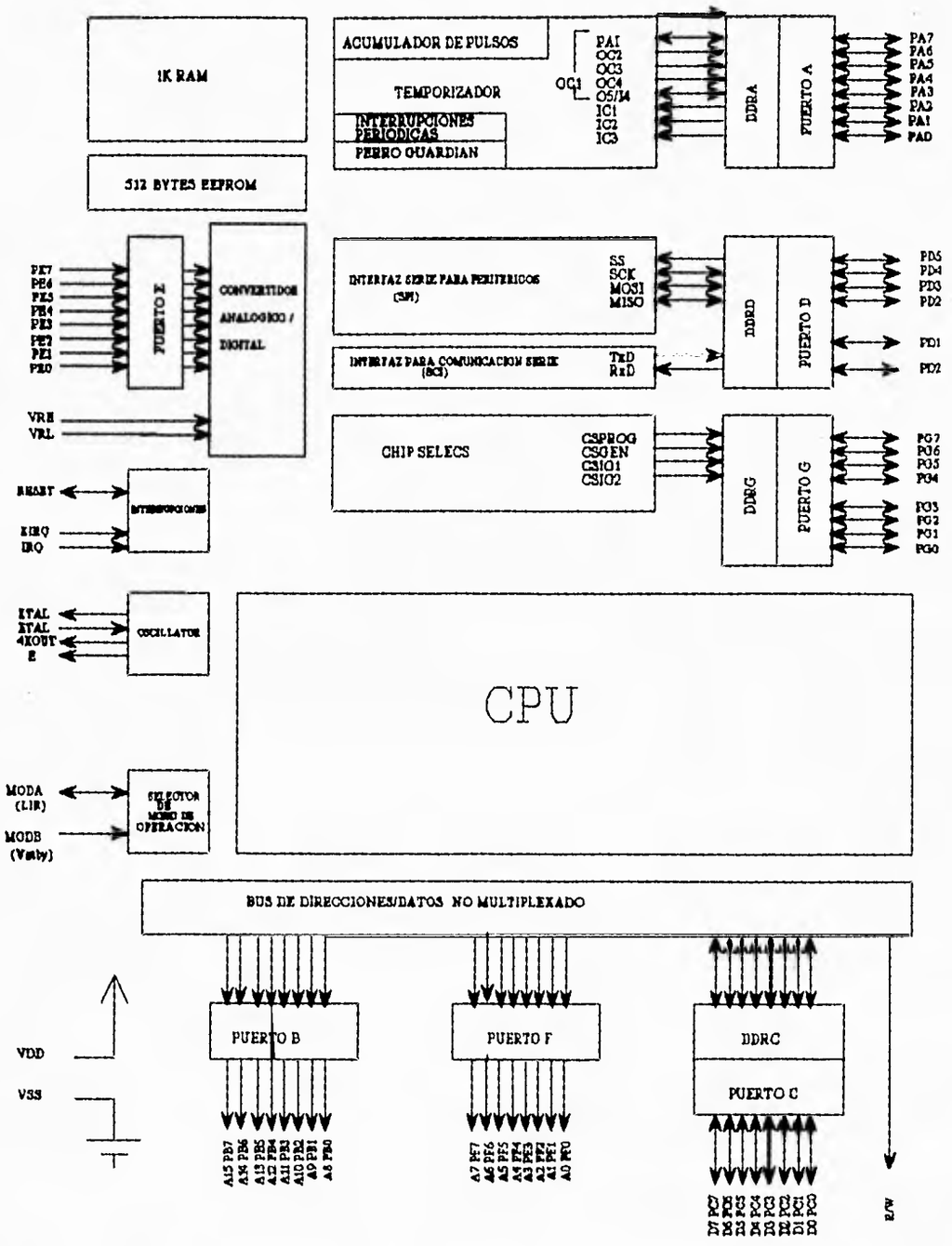


Figura 4.1 Diagrama a bloques del microcontrolador MC68HC11F1.

Figura 4.2 Modelo de programación.

Acumuladores A, B y D

Los acumuladores A y B son registros de propósito general, de 8 bits, usados para almacenar operandos y resultados de operaciones aritméticas o para manipulación de datos.

El acumulador D es un registro de 16 bits y se forma con la concatenación de los acumuladores A y B.

Registros de índice IX e IY

Estos son registros de 16 bits con usos muy variados, sirven como contadores o como

registros de almacenamiento temporal, además son empleados en el modo de direccionamiento indexado en diversas instrucciones, proporcionando un valor índice, al cual se le suma la cantidad especificada dentro de la misma instrucción para obtener una dirección, que es la que finalmente será empleada por la instrucción. Es importante notar que muchas de las instrucciones que usan el registro IY, requieren un byte adicional de código de máquina y un ciclo extra de ejecución.

Apuntador a la pila SP (Stack Pointer)

Una pila es una estructura de datos organizada de forma que al introducir y retirar datos en ella, el primer dato que entra es el último en salir (FIFO). El MC68HC11 tiene una pila relocalizable, por lo que, normalmente el SP debe inicializarse al inicio de un programa.

El registro SP contiene la dirección de la siguiente localidad libre sobre el tope de la pila, es decir, la localidad en la que se podrá introducir un nuevo dato que se pretenda forme parte de ella. Cuando se introduce un dato a la pila, el SP decrementa su valor en forma posterior a la introducción del dato y cuando se remueve un dato de la pila, el SP incrementa su valor después de que el dato ha sido removido.

El uso más importante que se le da a este registro es el almacenamiento temporal de direcciones en llamadas a interrupciones y subrutinas.

Contador de programa PC (Program Counter)

Este registro de 16 bits, contiene la dirección de la siguiente instrucción a ser ejecutada.

Registro de banderas CCR (Condition Code Register)

Este es un registro de 8 bits que contiene cinco indicadores de condición (C,V,Z,N y H), dos bits de máscaras de interrupción (IRQ y XIRQ) y un bit deshabilitador de "estado stop". Se emplea,

entre otras cosas para verificar el tipo de resultado de la instrucción inmediata anterior, para tomar decisiones dentro del algoritmo, para permitir o impedir el uso de interrupciones, saltos condicionados, etc.

4.3.2 Arquitectura Interna

La arquitectura de la CPU del MC68HC11 permite que todas las localidades asignadas a los periféricos, dispositivos de entrada/salida y memoria sean tratados en forma idéntica dentro del mapa de memoria de 64Kbytes. Por esta razón, no existen instrucciones especiales para los dispositivos de entrada/salida y para accesos a memoria. Esta técnica es llamada "Acceso a memoria de E/S". En consecuencia cuando se realiza un acceso a una localidad de memoria externa, el tiempo de acceso es el mismo que al realizar un acceso a una localidad de memoria interna del microcontrolador.

La CPU del MC68HC11 ofrece nuevas características en comparación con las CPUs del M6800 y el M6801. El cambio más significativo es la adición del registro de índice de 16 bits (IY). Se han incluido nuevas instrucciones para manipulación de bits permitiendo el manejo de cualquier bit o combinación de bits en la localidad de memoria que se desee dentro del espacio de direccionamiento de 64Kbytes. Se tienen además, dos nuevas instrucciones para división de operandos de 16 bits por 16 bits. Instrucciones que permiten el intercambio del contenido de cualquier registro de índice con el contenido del acumulador de 16-bits (D). Además, muchas instrucciones han sido mejoradas para realizar operaciones aritméticas de 16 bits de forma que resulte más fácil su uso.

Como en la mayoría de los diseños de las computadoras actuales, la arquitectura de la CPU de este microcontrolador está basada en los siguientes conceptos:

- Se maneja el concepto de programa almacenado en memoria.

- El contenido de esta memoria es direccionable por localidades, sin importar el tipo de datos contenidos en tales localidades.

- La ejecución del programa se hace en forma secuencial (a menos que se especifique lo contrario) de una instrucción a la siguiente.

4.3.3 Lenguaje Ensamblador

Un programa en lenguaje ensamblador controla al microprocesador en su propio lenguaje, sin la ayuda de comprobaciones de un compilador. Una de las dos ventajas más importantes del lenguaje ensamblador es la gran velocidad a la que ejecuta el código que consiste en nemónicos (representaciones simbólicas abreviadas de una instrucción máquina binaria real). La segunda ventaja del lenguaje ensamblador es que le da al programador el control directo de las operaciones de entrada/salida, manejo de dispositivos externos, acceso directo a registros, memoria, etc.

Modos de Direccionamiento

En la CPU del MC68HC11 se pueden emplear 6 modos de direccionamiento:

a) Direccionamiento Inmediato: En este caso el argumento está contenido en el byte o los bytes que siguen inmediatamente a la instrucción.

b) Direccionamiento Extendido: Para este direccionamiento, los dos bytes después del código de operación contienen la dirección absoluta del operando.

c) Direccionamiento Directo: Este tipo de direccionamiento permite al usuario tener acceso a las localidades de memoria $0000 a la $00FF, de forma que en una instrucción determinada que

emplee este tipo de direccionamiento, la CPU asume que el byte más significativo del operando es un $00 por lo que basta escribir el byte menos significativo después del código de operación. Una ventaja de este modo de direccionamiento es que ahorra tiempo de ejecución y espacio en memoria porque en todas las instrucciones que lo empleen no es necesario considerar el byte más significativo del operando.

d) Direccionamiento Indexado: Uno de los registros de índice (IX o IY) es usado en el cálculo de la dirección efectiva, sumando su contenido con una cantidad sin signo de 8 bits incluida en la instrucción; cualquier localidad puede ser direccionada de esta manera en el espacio de direccionamiento de 64Kbytes.

e) Direccionamiento Inherente: En este caso, toda la información está contenida en el código de operación. Los operandos (si existen), son registros.

f) Direccionamiento Relativo: Este direccionamiento es usado solamente por instrucciones de saltos. Si la condición de salto es verdadera, el contenido del byte con signo que está a continuación del código de operación se suma al contenido del contador de programa para formar la dirección efectiva de salto, en caso de que la condición de salto sea falsa, prosigue con la siguiente instrucción.

Tipos de Instrucciones

a) Instrucciones de Acumuladores y Memoria:

a.1) Cargas, almacenamientos y transferencias.
a.2) Operaciones aritméticas.
a.3) Multiplicación y división.
a.4) Operaciones lógicas.

a.5) Prueba de datos y manipulaciones de bits.

a.6) Corrimientos y rotaciones.

b) Instrucciones de Pila y de Registros de Indice.

c) Instrucciones del Registro de Banderas (CCR).

d) Instrucciones de Control de Programa:

d.1) Saltos relativos condicionados.

d.2) Saltos absolutos.

d.3) Llamadas a subrutinas y retornos de subrutinas.

d.4) Manejo de interrupciones.

d.5) Misceláneas

4.4 MODOS DE OPERACION

El microcontrolador MC68HC11F1 puede trabajar en 4 modos de operación, dependiendo de la posición de las terminales MODA y MODB, de acuerdo con la siguiente tabla:

MODA	MODB	MODO SELECCIONADO	TIPO
0	0	Modo Bootstrap	Especial
0	1	Modo Single-Chip	Normal
1	0	Modo Prueba (Test)	Especial
1	1	Modo Expandido	Normal

Tabla 4.1 Modos de Operación del MC68HC11F1.

4.4.1 Modo de operación bootstrap

Este es uno de los modos especiales de operación del microcontrolador, el cual, permite introducir en su RAM interna, programas de propósito especial. Cuando está seleccionado este modo de operación, al reinicializar el microcontrolador, se hace presente un área de ROM en el mapa de memoria, la cual contiene un pequeño programa llamado "bootloader" que inicializa el SCI y permite al usuario cargar su programa en la memoria del chip. Los vectores de RESET e interrupción, se localizan en la ROM en las direcciones $BFC0-$BFFF.

Después de reinicializar el microcontrolador, bajo este modo de operación el programa "bootloader" realiza lo siguiente:

1. Inicializa el puerto serie del microcontrolador a una frecuencia de E/16 por lo que si el reloj E funciona a una frecuencia de 2MHz la comunicación se realizará a 7812 bauds.

2. Espera la recepción de un caracter.

3. La computadora anfitriona debe enviar un caracter de arranque, el cual va a ser un FF. Esta computadora debe estar inicializada a 7812 bauds o a 1200 bauds.

4. Tomando en cuenta el caracter de arranque recibido, el microcontrolador reconoce el baudaje de la computadora anfitriona, en caso de que el caracter de arranque sea distinto de $FF ($C0 o $E0) se concluye que el baudaje de la computadora es de 1200 y se hace el cambio pertinente en la programación del puerto serie del microcontrolador, ya que éste asume que el baudaje será de 7812, es decir, espera recibir un $FF como caracter de arranque.

5. El microcontrolador procede a recibir una cadena de bytes, colocándolos progresivamente a partir de la dirección cero de la RAM.

6. Al terminar la recepción (cuando la computadora anfitriona ya no transmite) el microcontrolador efectúa un salto absoluto a la dirección $0000 para comenzar a ejecutar el programa.

Es importante hacer notar que cada caracter recibido por el microcontrolador en el modo bootstrap es retransmitido como testigo.

En este modo de operación las direcciones de los vectores de interrupción se encuentran dentro del área de ROM y apuntan a direcciones que se ubican en la página cero de la RAM, estas direcciones de RAM deben ser llenadas adecuadamente con saltos a las rutinas de interrupción para usar las interrupciones.

Debido a lo anterior, es necesario tener cuidado de que el programa del usuario no invada las localidades de los pseudovectores de interrupción que esté utilizando y también es necesario considerar que debido al uso de los pseudovectores en el modo bootstrap aumenta el tiempo de latencia de interrupción (tiempo que transcurre desde que se invoca a la interrupción hasta que se empieza a ejecutar).

Dependiendo del caracter de arranque, el "bootloader" efectúa las siguientes funciones alternativas, además de las ya mencionadas:

- Si el caracter de arranque es $00, el "bootloader" genera un salto al origen de la EEPROM.

- Si se conectan entre si las terminales TxD y RxD, al dar RESET en modo bootstrap, el transmisor envía un caracter especial al receptor, el cual lo interpreta como un $00 y por lo tanto salta al origen de la EEPROM.

4.4.2 Modo de operación single-chip

En este modo de operación todo el software requerido para el manejo del microcontrolador debe estar contenido en los recursos internos del mismo (EEPROM y ROM). Tanto la ROM como la EEPROM estarán habilitadas una vez que se haya reiniciado el microcontrolador, siempre y cuando se hayan programado correctamente los vectores de RESET e interrupción en el área dentro del mapa de memoria que les corresponde. Otra característica importante de este modo es que todos los puertos paralelos de entrada/salida están disponibles para propósito general.

4.4.3 Modo de operación prueba

El modo prueba es uno de los modos especiales, y es una variación del modo de operación expandido. Se usa básicamente para pruebas de producción dentro de Motorola, para hacer simulaciones y depuración durante el desarrollo del chip; además está accesible para programación del registro CONFIG, entre otras cosas.

4.4.4 Modo de operación expandido no multiplexado

En el modo de operación expandido - no multiplexado, el microcontrolador puede tener acceso a un espacio de direcciones de hasta 64 Kbytes. Este espacio incluye tanto las direcciones propias del chip, usadas por el modo single-chip como direcciones externas usadas por periféricos y dispositivos de memoria externos.

4.4.5 Mapas de memoria

El modo de operación del microcontrolador determina el mapa de memoria y cuando es

posible tener acceso direcciones externas. Las localidades de memoria para el modo expandido no multiplexado incluyen a las localidades en modo single-chip. Los bits de control del registro CONFIG permiten habilitar o deshabilitar la ROM y la EEPROM dentro del mapa de memoria. La RAM interna se ubica de la dirección $0000 a la $03FF después de reinicializar el microcontrolador. Esta puede ser ubicada en una página delimitada de 4 Kbytes dentro del mapa de memoria, teniendo cuidado de no invadir otras áreas del mapa que se vayan a emplear, para esto se requiere escribir el valor requerido en el registro INIT. De igual forma, el bloque de 96 Bytes de registros que se localiza a partir de la dirección $1000 al reinicializar el microcontrolador, puede ser reubicado en cualquiera de las páginas de 4 Kbytes del mapa de memoria, escribiendo el valor adecuado en el registro INIT. En caso de que la RAM y el bloque de registros sean ubicados en la misma página, no será posible tener acceso a los primeros 96 Bytes de RAM.

La figura 4.3 muestra los mapas de memoria (después de la reinicialización) correspondientes a cada uno de los modos de operación del MC68HC11F1.

4.5 PUERTOS PARALELOS (ENTRADA/SALIDA)

El MC68HC11F1 tiene 6 puertos paralelos de entrada/salida de 8 bits (A,B,C,E,F Y G), y uno adicional de 6 bits (D).

Las funciones de entrada/salida de algunos puertos (B,C,F y G) son afectadas por el modo de operación seleccionado. En los modos single-chip y bootstrap éstos son configurados como puertos de entrada/salida paralele de datos. En los modos expandido y prueba, tanto los puertos B,C,F y G como la terminal R/W son configurados como líneas de expansión de memoria, utilizando los puertos B y F como bus de direcciones, el puerto C como bus de datos, la señal R/W para controlar la dirección del bus de datos, y los 4 bits más significativos del puerto G como habilitadores opcionales de periféricos externos (external chip-selects). Los puertos restantes no son afectados con los cambios del modo de operación.

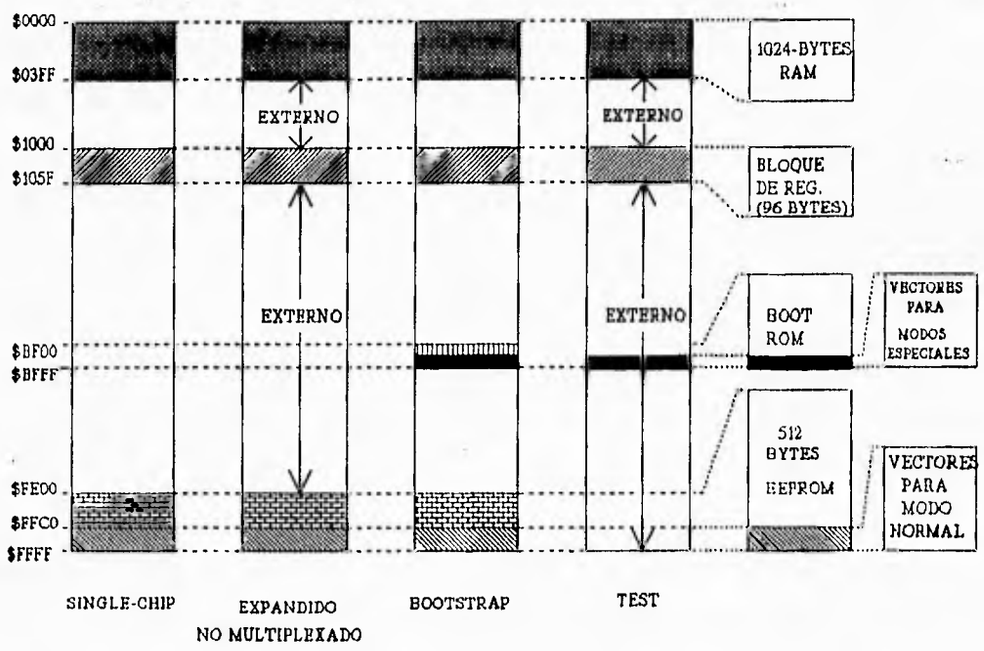


Figura 4.3 Mapas de memoria del MC68HC11F1.

Los puertos A,D y G pueden ser usados para propósito general de entrada/salida, y además cada uno tiene una función alternativa. Las terminales del puerto A controlan las funciones del temporizador, el puerto D maneja las funciones de comunicación por SPI (Interfaz Serie para Periféricos) y SCI (Interfaz de Comunicación Serie), y el puerto G controla las funciones de chip-select. El puerto E puede ser usado para entradas de propósito general y/o para las entradas de los canales del convertidor A/D.

Puerto A

Este es un puerto de propósito general, es bidireccional y cuenta con dos registros; un registro de datos (PORTA) y un registro de control de dirección de datos (DDRA). Las terminales del puerto A pueden ser usadas por el temporizador, el acumulador de pulsos y para funciones de entrada/salida de propósito general. Cuatro terminales de este puerto pueden ser usadas como salida

para las funciones de comparación en el temporizador (OC), tres como entrada para funciones de captura (IC) y una más que puede servir como cuarta terminal para captura o como quinta terminal para comparación en el temporizador. La terminal PA7 es usada por el acumulador de pulsos.

Puerto B

Como ya se mencionó anteriormente, el puerto B cambia sus funciones dependiendo del modo de operación; para single-chip y bootstrap éste es un puerto de salida de propósito general y en el modo expandido y el modo prueba forma parte del bus de direcciones, representando a las líneas más significativas (A15-A8). Este puerto, solo cuenta con un registro, el de datos (PORTB).

Puerto C

El puerto C es bidireccional, cuenta con un registro de datos (PORTC) y con un registro de control de dirección de datos (DDRC). En modo single-chip, este puerto es de propósito general de entrada/salida y opcionalmente puede ser configurado para trabajar como OR alambrada (wired-OR) en este mismo modo de operación, mediante la habilitación del bit CWOM localizado en el registro OPT2. En modo expandido, el puerto C es el bus de datos (D7-D0).

Puerto D

Este es un puerto de 6 bits, de propósito general, bidireccional, cuenta con un registro de datos (PORTD) y con un registro de control de dirección de datos (DDRD). En todos los modos de operación, los 6 bits del puerto pueden ser usadas como terminales de entrada/salida de propósito general o para los subsistemas de comunicación SCI y SPI. El puerto D, también puede ser configurado para realizar la operación OR alambrada.

Puerto E

Este es un puerto de 8 bits de entrada, cuyas terminales pueden ser usadas para propósito general o como canales de entrada del convertidor A/D. Este puerto solo cuenta con un registro, el de datos (PORTE).

Puerto F

En el modo de operación single-chip, el puerto F es un puerto de salida de propósito general y en el modo de operación expandido, este puerto forma parte del bus de direcciones, representando a las líneas menos significativas (A7-A0). Este puerto solo cuenta con un registro, el de datos (PORTF).

Puerto G

Este es un puerto de propósito general, de 8 bits y cuenta tanto con un registro de datos (PORTG) como con un registro de control de dirección de datos (DDRG). En el modo de operación expandido, los cuatro bits más significativos pueden ser utilizados para las funciones de los chip-selects con los que cuenta este microcontrolador.

4.6 PERIFERICOS INTERNOS

4.6.1 Interfaz Asíncrona de Comunicación Serie (SCI)

El SCI es una interfaz de recepción y transmisión asíncrona universal (UART: Universal Asynchronous Receiver Transmitter) para comunicación serie, es uno de los subsistemas de entrada/salida de MC68HC11F1. Este sistema SCI puede ser usado para conectar una computadora personal al microcontrolador, o bien, varios microcontroladores distribuidos pueden usar sus sistemas SCI para formar una red de comunicación serie.

El SCI es un sistema de comunicación duplex integral que usa el formato NRZ (nonreturn-to-zero), un bit de inicio, 8 o 9 bits de datos y un bit de paro. Un generador de baudaje en la unidad microcontroladora se encarga de producir diferentes baudajes. El transmisor y el receptor son funcionalmente independientes pero usan el mismo formato para los datos y el mismo baudaje. Se requiere de un circuito externo para trasladar los niveles RS-232 (típicamente ±12 V) a los 0 y +5 V usados por el microcontrolador y viceversa.

El receptor del SCI del microcontrolador incluye un gran número de características avanzadas, para asegurar una confiable recepción y para participar en el desarrollo de redes de comunicación eficientes. El MC68HC11 resincroniza el reloj del receptor en todas las transiciones uno-a-cero, no solo al inicio de la recepción, por ello, las diferencias de baudaje entre el dispositivo transmisor y el microcontrolador, muy difícilmente ocasionan errores en la recepción. Adicionalmente, el receptor puede configurarse para entrar en un estado de espera llamado "receiver wake up", con el fin de ignorar mensajes dirigidos a otros receptores, y la lógica que se tiene automáticamente habilita al receptor a tiempo para ver el primer caracter del siguiente mensaje. Esto es muy útil cuando se tienen redes punto-multipunto.

En una red, el bit de control WAKE del registro SCCR1 es usado por el receptor de cada

microcontrolador para saber el método que será empleado para despertarlos. Existen 2 métodos, cuando WAKE está a 0 es necesario enviar un caracter "idle" ($FF) y cuando está a 1 es necesario enviar el bit más significativo del primer caracter del mensaje como un 1, esto sirve en ambos casos para poner alerta a todos los microcontroladores, en el segundo caso, llamado Marca de Dirección, al detectar el caracter con el bit más significativo igual a 1, cada microcontrolador debe revisar si el dato que acompaña a este bit es igual a su marca de dirección, en caso afirmativo, el microcontrolador continúa la recepción del mensaje, de lo contrario, el microcontrolador ignorará el mensaje.

El bit RWU del registro SCCR2 permite habilitar o deshabilitar la función WAKE-UP. Cuando RWU=0, se realiza una recepción normal por SCI, es decir, sin detectar ninguno de los métodos antes mencionados, y en caso de que RWU=1, se detecta el método y el microcontrolador continúa la recepción del mensaje.

El transmisor SCI contiene, además de la bandera de estado TDRE (Transmit Data Register Empty), una indicación de transmisión completa (TC). La parte central del transmisor es el registro de corrimiento de transmisión serie, usualmente este registro obtiene los datos del buffer transmisor de solo escritura. Los datos se almacenan en el buffer de transmisión después de que por software, se escribe en el registro SCDR del SCI. Una vez que los datos se transfieren al registro de corrimiento desde el buffer de transmisión, se añade un cero como bit menos significativo que actúa como bit de inicio y un uno lógico se añade como bit más significativo para actuar como un bit de paro.

El bit T8 en el registro de control 1 (SCCR1) se usa como bit extra (el noveno bit) en el registro de transmisión. Este noveno bit se utiliza solamente si el bit M en SCCR1 es puesto a uno para seleccionar el formato de caracter de 9 bits.

Las banderas de estado TDRE y TC del registro SCSR son activadas automáticamente por la lógica del transmisor. Estos dos bits pueden ser leídos en cualquier instante mediante software.

Para generar requerimientos de interrupción por SCI es necesario que además de que se active la bandera de estado TDRE ó TC, su respectivo bit de permiso de interrupción: TIE (Transmit Interrupt Enable) o TCIE (Transmit Complete Interrupt Enable) esté habilitado.

El elemento principal del receptor es el registro de corrimiento serie de recepción. Este registro es habilitado por el bit RE (receive enable) del registro SCCR2. El bit M del registro SCCR1 determina si el corrimiento del caracter de llegada va a ser de 10 o de 11 bits. Después de detectar el bit de paro de un caracter, el dato recibido es transferido desde el registro de corrimiento a SCDR y se activa la bandera RDRF, que indica que el registro de dato recibido está lleno. Cuando un caracter está listo para ser transferido al buffer de recepción pero el caracter previo no ha sido leído, ocurre un error de sobreescritura de datos. En esta condición de sobreescritura, el dato no es transferido y la bandera OR (OverRun) se activa para indicar un error.

Para programar el sistema SCI se utilizan básicamente los registros BAUD (Registro de Control de Baudaje) y SCCR2 (Registro de Control 2 del SCI), mientras que para operarlo se manejan principalmente los registro SCSR (Registro de Estado del SCI) y SCDR (Registro de Datos del SCI).

4.6.2 Interfaz Serie para Periféricos (SPI)

El SPI es uno de los dos subsistemas independientes de comunicación serie incluidos en el MC68HC11F1. Como su nombre lo indica, el SPI da al microcontrolador la capacidad para comunicarse en forma síncrona con dispositivos periféricos y otros microprocesadores, además de que permite la comunicación en un sistema en el que participan múltiples maestros y esclavos. Los dispositivos periféricos pueden ser tan simples como un registro de corrimiento de tecnología TTL, o tan complejos como un subsistema completo, por ejemplo el controlador de un display de cristal líquido ó un convertidor Analógico/Digital. El SPI es lo suficientemente flexible como para tener interfaz directa con numerosos periféricos de diversos fabricantes.

El SPI del microcontrolador puede configurarse, de forma que el microcontrolador opere como maestro o como esclavo.

La lógica de control de reloj permite seleccionar la polaridad de la señal de reloj del SPI, entre dos opciones, dando la posibilidad de elegir la más adecuada según el dispositivo periférico de que se trate. Cuando el microcontrolador es configurado como maestro, se puede seleccionar por medio de software, una de 4 diferentes frecuencias para el reloj del SPI.

Para comunicación entre procesadores, el SPI tiene incluida una lógica para detección de errores, de forma que cuando se intenta escribir en el registro de corrimiento serie antes de que la transmisión haya terminado, se presentará un error por colisión de datos. Además, cuando se trabaja con múltiples microprocesadores configurados como maestros y se pretende emplear más de uno de ellos como maestro a la vez, se cuenta con una lógica de protección tal que deshabilita el sistema SPI de los microcontroladores que así lo requieran para evitar conflictos y errores en la comunicación.

Cuando ocurre una transferencia por SPI, se realiza el corrimiento del caracter de 8 bits a enviar dentro de un registro y al mismo tiempo estos bits están siendo trasladados a la terminal de salida. Por otro lado, conforme se hace el corrimiento del caracter a enviar, se está recibiendo un caracter de 8 bits en forma serie en el mismo registro, el cual proviene de la terminal de entrada del sistema SPI. Otra forma de ver esta transferencia es considerar que existe un registro de corrimiento de 8 bits en el maestro y otro registro de corrimiento de 8 bits en el esclavo y que se concatenan para formar un registro circular de 16 bits; cuando ocurre una transferencia, este registro de 16 bits sufre un corrimiento de 8 bits, de forma que los caracteres de 8 bits del maestro y del esclavo son intercambiados.

El elemento central del sistema SPI es el bloque que contiene el registro de corrimiento y el buffer de lectura de datos. Cuando se realiza una transmisión, el dato se escribe directamente en el registro de corrimiento. Y cuando se trata de una recepción, el dato es recibido en el registro

de corrimiento, y es transferido, en forma simultánea, al buffer de lecture de datos. Siempre y cuendo este caracter sea leído del buffer de lectura de datos, antes de que el siguiente ceracter esté listo pera ser trensferido, no ocurrirán errores de seturación. Pera la programación del microcontroledor se emplea un mismo registro pera leer un caracter del buffer de lectura de detos y pare escribir un ceracter en el registro de corrimiento.

Terminales externas del sistema SPI

Existen 4 terminales de entrada/salida esociedas con el SPI:

- SCK
- MISO
- MOSI
- SS

Cuando está deshabilitado el sistema SPI, estas 4 termineles son tratadas como de entrada/salida de propósito general.

La terminal SCK es de salida cuando el microcontrolador está configurado como maestro y es de entrade, cuando el microcontrolador se configura como esclavo. En el primer ceso, cuendo el microcontrolador está configuredo como meestro, la señal SCK se obtiene directamente del reloj interno del microcontrolador. Cuendo el maestro inicia una transferencie, se generan automáticamente los ciclos de reloj en le terminal SCK. Cuando el SPI está configurado como esclavo, la señal de reloj que envía el maestro permite la sincronización para la transferencia del dato entre los dispositivos maestro y esclavo. A menos que la terminal SS del microcontrolador esclavo se encuentre en estado bajo, le señal SCK es ignoreda.

Las líneas MISO y MOSI son usadas para trensmitir y recibir datos en forma serie. Cuando

el SPI está configurado como maestro, la terminal MISO es la línea de entrada y la terminal MOSI es la línea de salida. Cuando el SPI se configura como esclavo, se intercambian las funciones de estas terminales.

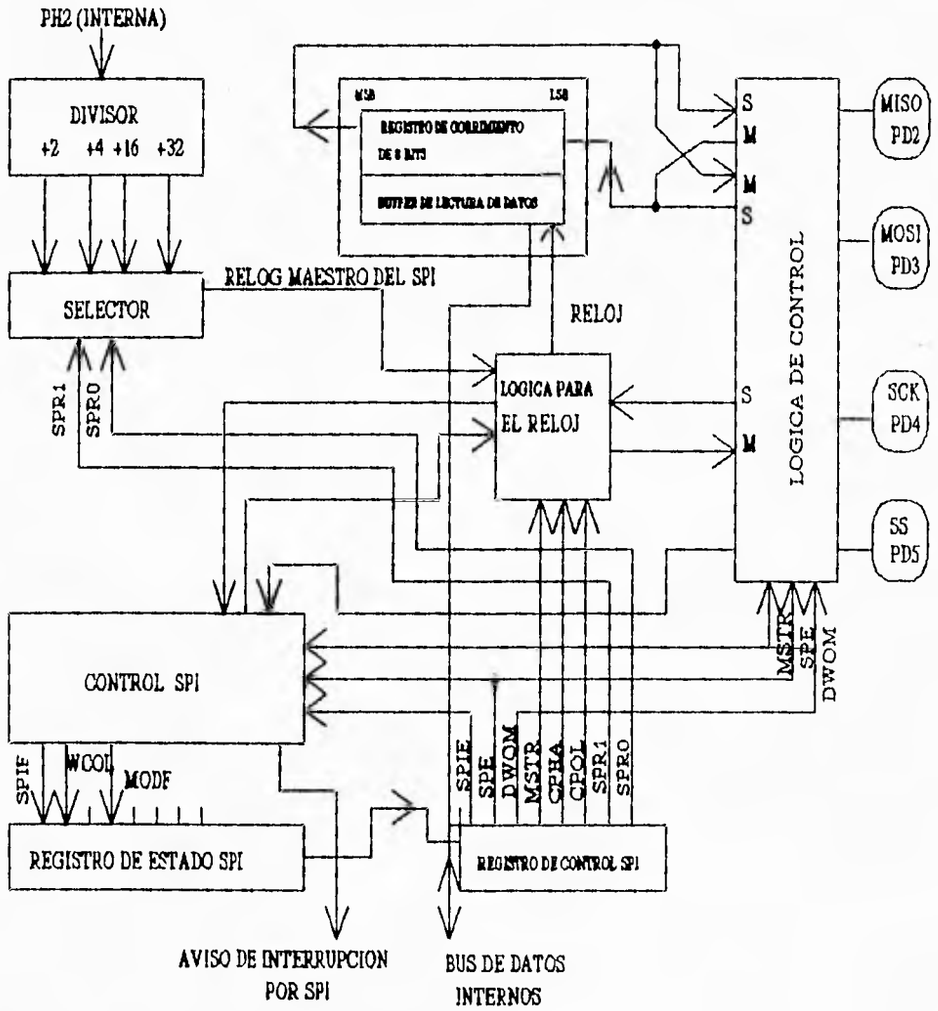


Figura 4.4 Diagrama a bloques del sistema SPI.

La terminal SS se comporta en forma diferente para un dispositivo maestro y para un dispositivo esclavo. Cuando el microcontrolador es configurado como esclavo, esta terminal se emplea para permitir o impedir transferencias por SPI. Si la terminal SS del esclavo está inactivo

(en estado alto), el dispositivo ignora la señal SCK y mantiene la terminal de salida MISO en estado de alta impedancia. En un dispositivo configurado como maestro la terminal SS puede emplearse como entrada de detección de errores o como una salida de propósito general dependiendo del estado lógico en que se encuentre el bit correspondiente (DDRD5) dentro del registro DDRD. Cuando el bit DDRD5 es 1 y el SPI está configurado como maestro la terminal PD5/SS se comporta como una salida de propósito general y es independiente de las actividades del SPI. Cuando el bit DDRD5 es un cero y el SPI está configurado como maestro la terminal SS actúa como una entrada de detección de errores que debe permanecer en estado alto, ya que si cambia a estado bajo indicará que algún otro dispositivo pretende ser el maestro.

4.6.3 Convertidor Analógico a Digital (ADC)

El microprocesador MC68HC11F1 incluye un convertidor analógico a digital (A/D) de aproximaciones sucesivas que utiliza la técnica de redistribución de carga en un arreglo totalmente capacitivo, incluye una entrada multicanalizada de ocho canales analógicos exteriores y no requiere de circuitos externos de muestreo y retención debido al tipo de técnica utilizada. Tiene además dos líneas para establecer el voltaje de referencia: VRL (Voltaje de referencia mínimo) y VRH (Voltaje de referencia máximo). Estas líneas permiten fijar externamente el rango (VRH - VRL) de las entradas analógicas. El funcionamiento de este circuito esta probado y especificado por Motorola para VRL=0 V y VRH=5 V. El error total de este convertidor A/D es de ±1 bit menos significativo. Cada conversión tarda 32 ciclos de reloj siempre y cuando la frecuencia de reloj E sea mayor de 750 KHz. En el proceso de conversión, un voltaje de entrada igual a VRL da por resultado el valor $00 y un voltaje de entrada igual a VRH es convertido a $FF.

Como ya se mencionó anteriormente, el convertidor A/D funciona con base en la técnica de redistribución de carga, la cual consta de 3 etapas:

- Muestreo

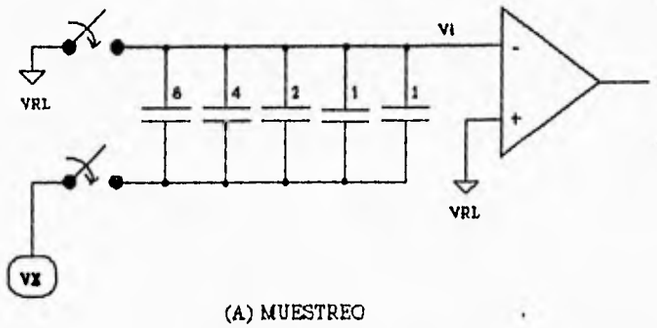
- Retención y
- Aproximaciones Sucesivas

Para explicar el funcionamiento en forma sencilla, se hace referencia a la figura 4.5, en la cual se muestran las 3 etapas de un convertidor A/D de 4 bits, sin olvidar que el convertidor del MC68HC11 es de 8 bits.

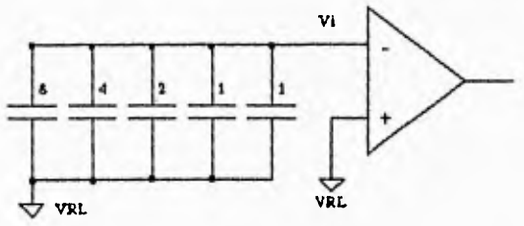
Durante el muestreo, el arreglo capacitivo se carga al voltaje analógico Vx (voltaje a convertir), posteriormente, durante la retención se cambia la referencia dando lugar a que el voltaje de entrada del comparador Vi sea igual a -Vx. En el periodo de aproximaciones sucesivas, se hace uso de un circuito que intercambia la alimentación de cada capacitor (de VRL a VRH). Primeramente, aplica el intercambio al capacitor mas grande, de tal forma que en el nodo Vi, se tiene un voltaje que puede ser mayor o menor que el voltaje de comparación (VRL). En caso de que $Vi > VRL$, la salida del comparador será un 0 lógico y el capacitor será conectado de nuevo a VRL , en caso contrario ($Vi < VRL$), la salida del comparador será un 1 lógico y el capacitor permanecerá conectado a VRH. De la misma manera, este método de conversión por aproximaciones suscesivas continúa probando con el resto de los capacitores, del mayor al menor. Cada vez que el capacitor en turno es conectado hacia VRH, el comparador envía al SAR (registro de aproximaciones suscesivas) un 1 o un 0 lógico. De manera que cada salida del comparador representa un bit del SAR. El proceso continúa hasta llegar al capacitor más pequeño y con el resultado del comparador se establece el bit menos significativo del SAR.

La figura 4.6 muestra detalladamente la secuencia de un conjunto de 4 conversiones. Esta secuencia comienza un ciclo de reloj E después de una escritura al registro de control/estado del convertidor A/D (ADCTL).

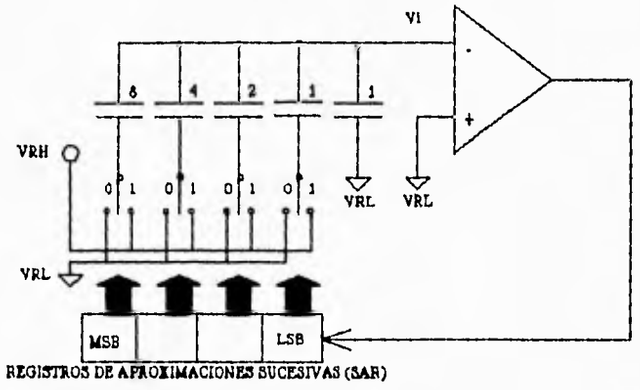
El resultado durante el proceso de conversión se almacena como ya se describió en el registro SAR y posteriormente se transfiere al registro de resultado ADRx, donde $0 \leq x \leq 4$, dependiendo del canal de conversión de que se trate.



(A) MUESTREO



(B) RETENCION



(C) APROXIMACIONES SUCESIVAS

Figura 4.5 Diagrama a bloques del convertidor A/D.

Al final de la cuarta conversión se activa la bandera de conversión completa (CCF) indicando que los resultados de las conversiones ya se encuentran en los registros ADRx.

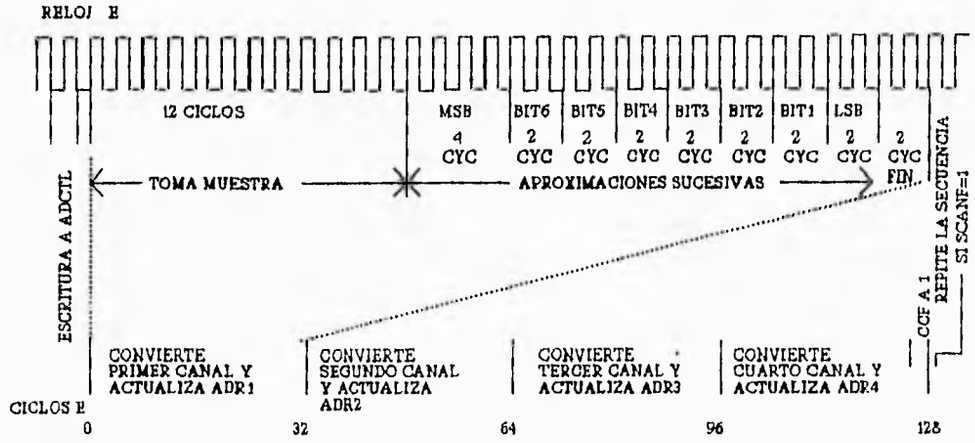


Figura 4.6 Secuencia de un conjunto de 4 conversiones.

Existen 2 variantes de conversión para el circuito A/D del MC68HC11F1, la operación de un solo canal y la de múltiples canales, en la primera, se realizan 4 conversiones sucesivas de la señal analógica presente en un solo canal y en la segunda se realiza una conversión de cada una de las señales presentes en un grupo de 4 canales, (ya sea en el grupo 1: PE0,PE1,PE2 y PE3 o bien el grupo 2 formado por PE4,PE5,PE6 y PE7). El modo de operación de canal único o de múltiples canales se selecciona por medio del bit MULT del registro ADCTL.

Por otra parte, el bit SCAN del registro ADCTL, permite seleccionar entre otras dos variantes para la conversión, en la primera se realiza una única conversión y una vez terminada ésta, se detienen todas las actividades de conversión hasta que se envíe otro comando de conversión al A/D. Y en el segundo caso, el proceso de conversión es continuo, es decir, al terminar la conversión que se haya seleccionado (de canal único o de múltiples canales) se prosigue con la siguiente en forma continua.

El canal o grupo de canales de las señales a convertir se seleccionan por medio de los bits CD,CC,CB y CA del registro ADCTL.

4.6.4 Temporizador Principal

El sistema temporizador está basado en un contador de 16 bits de conteo libre que utiliza como frecuencia de entrada la señal proveniente de un preescalador de 4 estados, el cual divide la frecuencia de la señal E entre 1, 4, 8 ó 16. Las funciones básicas del temporizador son dos: captura y comparación. Cuenta con 3 entradas de captura, usadas para grabar automáticamente el tiempo exacto en que se detecta una transición determinada en la terminal de captura seleccionada. Tiene además 4 salidas de comparación, que permiten generar señales de salida con una frecuencia programada, generar interrupciones cada cierto tiempo, etc. Existe una terminal adicional que puede ser usada ya sea como una cuarta entrada de captura o como una quinta salida de comparación. Las terminales del temporizador son: PA0-PA2 que corresponden a las entradas de captura, PA4-PA7 a las salidas de comparación y PA3 puede programarse como entrada de captura o salida de comparación. La figura 4.7 muestra el diagrama a bloques del sistema temporizador del MC68HC11F1.

El elemento central del sistema temporizador es el contador de 16 bits que se ha mencionado anteriormente. Este contador comienza en $0000 justo después de la reinicialización y cuenta continuamente en forma ascendente. Cuando se alcanza el máximo conteo ($FFFF), el contador vuelve a comenzar con $0000, activa una bandera de sobreflujo (TOF), continúa contando ascendentemente y desactiva nuevamente la bandera. Siempre que el microcontrolador opere en modo normal (single-chip ó expandido no multiplexado), no existe manera de modificar o inicializar el conteo de este contador.

Como ya se dijo, el MC68HC11F1 cuenta con un preescalador que permite seleccionar una de 4 diferentes frecuencias de reloj, esta señal sirve como entrada al contador de 16 bits del temporizador. Esta opción permite que el programador tome una decisión dependiendo de la resolución que requiera del temporizador y la periodicidad de los sobreflujos de conteo. Después de la reinicialización, los registros de control, predefinen que la frecuencia de la señal de entrada al contador sea igual a la frecuencia de la señal E, lo cual proporciona una resolución de 0.5 μs y un

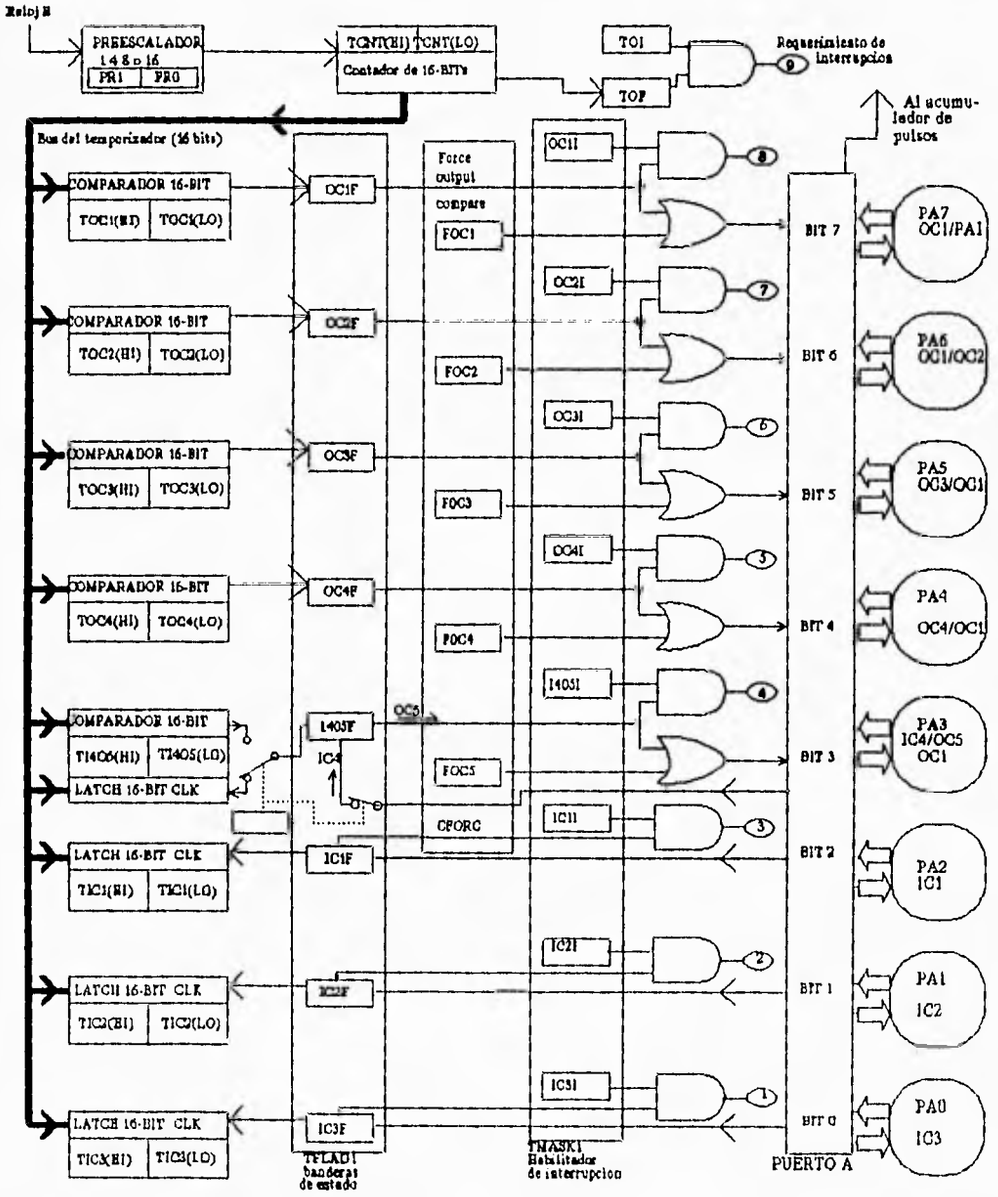


Figura 4.7 Diagrama a bloques del temporizador.

tiempo entre sobreflujos de 32.77 ms (siempre y cuando E=2MHz). Sin embargo el programador puede seleccionar el factor que más le convenga, por ejemplo, si selecciona el factor 16 (E dividido por 16), se tendrá una resolución de 8 μs y un retardo entre sobreflujos de 524.3 ms.

Para el microcontrolador, el tiempo se mide en base a su contador de 16 bits. Las entradas de captura son usadas para almacenar el instante en el cual ocurre un evento externo, logrando lo anterior almacenando el contenido del contador cuando se detecta la transición que se programó en la terminal de entrada seleccionada. El instante en el cual ocurrió el evento se almacena en el registro de captura de entrada (latch de 16 bits) correspondiente a la terminal de captura que se haya seleccionado. Tomando en cuenta que es posible almacenar los instantes de tiempo en que se presentan las transiciones de una señal, la principal aplicación de las entradas de captura es determinar el periodo y/o el ancho de pulso de una señal de entrada.

Las salidas de comparación se utilizan para que en un cierto lapso de tiempo se realice una acción. Esto se logra gracias a que el microcontrolador tiene, para cada una de las 5 posibles salidas de comparación, un comparador de 16 bits y un registro de comparación en el que el programador puede escribir el valor a compararse con el número generado por el contador de 16 bits del temporizador. Así, en cada ciclo de la señal E, el valor en el registro de comparación es comparado con el valor del contador del temporizador. Cuando la salida del contador iguala al valor en el registro de comparación, se genera una señal que activa la bandera de la salida de comparación correspondiente (OCxF) realizando posteriormente la acción programada: se complementa la salida OCx, se pone a cero la salida OCx, se pone a 1 la salida OCx y/o se genera una interrupción. Entre los usos de esta función está el de producir pulsos a una frecuencia específica, otra aplicación es la que le hemos dado en este proyecto, al generar una interrupción cada cierto período, determinado en este caso por la frecuencia de muestreo empleada.

4.6.5 Acumulador de Pulsos

El acumulador de pulsos está basado en un contador de 8 bits y puede funcionar como un simple contador de eventos o para acumulación de tiempo en forma controlada. A diferencia del contador del temporizador principal, el contador del acumulador de pulsos sí puede ser leído o alterado en cualquier momento. Los bits de control permiten al usuario configurar y controlar el subsistema acumulador de pulsos. Además, este sistema tiene dos interrupciones enmascarables asociadas, cada una con sus propios controles y su propio vector de interrupción. El bit PA7 del puerto A está asociado con el acumulador de pulsos.

En muchas aplicaciones se requiere contar eventos, por ejemplo: piezas en una línea de producción y ensamble, ciclos de una señal, unidades de tiempo, etc. Para emplear el acumulador de pulsos como un contador de eventos, es necesario emplear el transductor necesario para que dichos eventos sean traducidos a pulsos de voltaje y lleguen a la terminal PA7. Como transductor puede emplearse un par emisor/detector óptico, que únicamente al detectar la presencia de una pieza genere un pulso, de esta forma, la señal producida puede conectarse directamente a la terminal PA7 y programar el acumulador de pulsos para que funcione como un contador de eventos.

Cuando el acumulador de pulsos opera en el modo de acumulación de tiempo en forma controlada, cambia su función de contador a temporizador actuando en forma similar a las entradas de captura del temporizador principal. En este modo, cuando existe una señal activa en la terminal PAI (PA7), un contador de 8 bits (PACNT) es incrementado cada 64 ciclos de reloj E. El bit de control PEDGE indica que nivel activa el conteo. El uso más común para este modo de operación es medir la duración o el ancho de un pulso. Debido a que el contador no empieza a contar hasta que la señal de entrada cambia al nivel activo programado, la medición del ancho de pulso se hace en forma un poco distinta que como se haría con la función de captura del temporizador principal. Con el acumulador de pulsos, el contador se inicializa a cero antes que el pulso comience, de forma que el tiempo de duración del pulso se puede leer directamente cuando el pulso termina, mientras que en el temporizador principal es necesario capturar el tiempo de inicio y el tiempo de fin del pulso

y posteriormente restar dichos tiempos para obtener la duración del pulso.

4.6.6 Chip-selects programables

Los chip-selects eliminan la necesidad de colocar componentes adicionales externos para realizar una interfaz adecuada con los periféricos del modo expandido no multiplexado. Los registros chip-select controlan la polaridad, el tamaño del bloque de direcciones y la anchura del pulso de reloj para cualquiera de las señales chip-select.

Existen 4 chip-selects programables en el MC68HC11F1, dos para entrada/salida externa (CSIO1 y CSIO2), uno para indicar el espacio de programa externo (CSPRG), y el último es de propósito general (CSGEN).

CSPRG

El chip-select CSPRG es activo bajo y se habilita cuando existe una dirección válida, es decir, que se requiera un acceso a la EEPROM externa. El funcionamiento de esta terminal, es permitido únicamente cuando el bit PCSEN del registro de control de chip-selects (CSCTL) es puesto a 1. En RESET, para el modo expandido este bit es automáticamente puesto a 1 y para el modo single-chip es puesto a 0, con el fin de habilitar y deshabilitar el CSPRG respectivamente. El tamaño del bloque de direcciones se selecciona con los bits PSIZA y PSIZB del registro CSCTL.

CSIO1 y CSIO2

Los chip-selects de entrada/salida (CSIO1 y CSIO2) se emplean para dispositivos externos. Las direcciones a las que apuntan estos chip-selects se encuentran en el bloque de 4 Kbytes del mapa de memoria que contiene los registros de estado y control. El área de memoria habilitada por la terminal CSIO1 se localiza desde la dirección $1060 a la $17FF del mapa de memoria, y el área

de memoria habilitada por la terminal CSIO2 se localiza desde la dirección $1800 hasta la $1FFF. La polaridad y la selección de habilitación/deshabilitación de estos chip-selects es controlada por los bits IO1EN, IO1PL, IO2EN e IO2PL del registro CSCTL.

CSGEN

El chip-select de propósito general es el más flexible de los 4. La polaridad, el tiempo de direccionamiento válido ya sea cuando la señal E está en nivel alto o en bajo, y el tamaño del bloque de direcciones son determinados por los bits GNPOL, GAVLD, GSIZA, GSIZB y GSIZC del registro CSGSIZ. La dirección de inicio es seleccionada con el registro CSGADR.

Cada uno de los cuatro chip-selects está asociado con dos bits en el registro CSSTRH (Clock Stretching). Estos bits permiten el ajuste del ancho de pulso de reloj, desde 0 hasta 3 ciclos (periodos completos de reloj E) con el fin de evitar problemas de sincronía con dispositivos lentos. Cualquiera de los chip-selects puede ser programado para que el ajuste del ancho del pulso se haga únicamente durante el acceso a direcciones que caen dentro del rango de direcciones particular de ese chip select.

La asignación de niveles de prioridad para los cuatro chip-selects evita el conflicto que pudiera haber entre ellos mismos y el direccionamiento a memoria y/o registros. Existen dos conjuntos de prioridades controladas por el bit GCSPR del registro CSCTL que se indican en la siguiente tabla:

GCSPR = 0	GCSPR = 1
Registros internos del chip	Registros internos del chip
RAM interna del chip	RAM interna del chip
Bootloader ROM	Bootloader ROM

EEPROM interna del chip	EEPROM interna del chip
Chip-selects de entrada/salida	Chip-selects de entrada/salida
Chip-select del programa	Chip-select de propósito general
Chip-select de propósito general	Chip-select del programa

Tabla 4.2 Conjuntos de prioridades controladas por el bit GCSPR.

4.6.7 Reinicio (RESET) e Interrupciones

El MC68HC11F1 tiene 4 tipos de reinicio y 18 vectores de interrupción.

Los 4 tipos de reinicio son los siguientes:

- Reinicio externo por terminal RESET
- Reinicio por encendido
- Reinicio por fallas detectadas por el monitor de reloj
- Reinicio por fallas detectadas por el sistema "perro guardián"

Cuando la señal en la terminal RESET cambia a cero lógico y se mantiene en dicho nivel durante 4 ciclos de reloj E, si al ser muestreada 2 ciclos de reloj después continúa en nivel bajo, significa que ha ocurrido un reinicio externo. Si por el contrario al ser muestreada se encuentra en un nivel alto implica que el reinicio fue generado internamente, ya sea por el sistema "perro guardián" o por el monitor de reloj.

El reinicio por encendido ocurre cuando se detecta una transición positiva en el voltaje de alimentación V_{DD}. La lógica de encendido da por resultado un retardo de 4064 ciclos de reloj E. Tomando en cuenta que la frecuencia del reloj E es igual a 2MHz, el reinicio de encendido dura

aproximadamente 2ms, a cuyo término, si la terminal RESET está en nivel bajo, el microcontrolador permanecerá en la condición de reinicio hasta que ésta cambie a nivel alto.

Por otra parte, el reinicio por monitor de reloj se ejecuta cuando se detecta una falla en el sistema de reloj E del microcontrolador, enviando una señal a través de la terminal bidireccional RESET para indicar esta anomalía.

El sistema "perro guardián" vigila que la ejecución de una secuencia de instrucciones se realice dentro de un período límite, en caso de que este tiempo sea mayor de lo previsto, el microcontrolador interpretará esto como un error de procesamiento y por ello ejecutará una reinicialización por fallas detectadas por el sistema "perro guardián". Este sistema se habilita por medio del bit NOCOP dentro del registro CONFIG y los bits de control CR1 y CR0 del registro OPTION seleccionan el período de reinicio.

Existen 18 vectores de interrupción para las 22 fuentes de interrupción del microcontrolador (3 no enmascarables y 19 enmascarables). Las 3 interrupciones no enmascarables son las siguientes:

- Interrupción por código ilegal
- Interrupción por software
- Interrupción externa por la terminal XIRQ

Los sistemas periféricos internos del microcontrolador generan interrupciones enmascarables que son reconocidas solamente si la máscara de interrupción global (bit I del registro CCR) es puesta a cero y si su máscara de interrupción local está habilitada. La prioridad de las interrupciones enmascarables esta dada de acuerdo con la tabla 4.3, esta prioridad puede aumentarse, sin embargo, ninguna puede ser elevada a la más alta prioridad dentro de la tabla. Los cambios de prioridad pueden lograrse mediante la alteración por software del registro de control HPRIO. Este registro puede ser escrito en cualquier momento siempre y cuando el bit I del registro CCR esté puesto a uno.

De esta forma, 19 fuentes de interrupción son enmascarables por la máscara de interrupción global I. Además del bit I, todas estas fuentes, excepto la interrupción externa IRQ, son controladas por diversos permisos particulares de interrupción localizados en sus respectivos registros de control. A la mayor parte de las fuentes de interrupción les corresponden distintos vectores de interrupción y por lo general no es necesaria una revisión continua de los registros de control para determinar de que interrupción se trata, sino que el microcontrolador es capaz de reconocerla y ejecutar el código correspondiente a la rutina de interrupción.

Dirección del vector	Fuente de interrupción	Prioridad	Máscara de interrupción global	Máscara local
FFC0-FFD5	RESERVADOS		.....	.....
FFD6,D7	SCI (Interfaz para comunicación serie)	18	Bit I	
	SCI Transmisión Completa			TCIE
	SCI Registro de Datos de Transmisíon Vacío.			TIE
	SCI Detección de Línea en Estado de Alerta			ILIE
	SCI Receptor Saturado			RIE
	SCI Registro de Datos de Recepción Lleno			RIE
FFD8,D9	SPI Transferencia Serie Completa	"	Bit I	SPIE

FFDA,DB	Flanco de Detección del Acumulador de Pulsos	17	Bit I	PAII
FFDC,DD	Sobreflujo del Acumulador de Pulsos	16	Bit I	PAOVI
FFDE,DF	Sobreflujo del Temporizador	15	Bit I	TOI
FFE0,E1	Entrada de Captura 4 / Salida de Comparación 5 del Temporizador.	14	Bit I	I405I
FFE2,E3	Salida de Comparación 4 del Temporizador	13	Bit I	OC4I
FFE4,E5	Salida de Comparación 3 del Temporizador	12	Bit I	OC3I
FFE6,E7	Salida de Comparación 2 del Temporizador	11	Bit I	OC2I
FFE8,E9	Salida de Comparación 1 del Temporizador	10	Bit I	OC1I
FFEA,EB	Entrada de Captura 3 del Temporizador	9	Bit I	IC3
FFEC,ED	Entrada de Captura 2 del Temporizador	8	Bit I	IC2I
FFEE,EF	Entrada de Captura 1 del Temporizador	7	Bit I	IC1I
FFF0,F1	Interrupción en Tiempo Real	6	Bit I	RTII

FFF2,F3	Terminal IRQ	5	Bit I	Ninguna
FFF4,F5	Terminal XIRQ	4	Bit X	Ninguna
FFF6,F7	Interrupción por Software	*	Ninguna	Ninguna
FFF8,F9	Código Ilegal	*	Ninguna	Ninguna
FFFA,FB	Falla Detectada por el sistema "Perro Guardian"	3	Ninguna	NOCOP
FFFC,FD	Falla Detectada por Monitor de Reloj	2	Ninguna	CME
FFFE,FFFF	Reinicio (RESET)	1	Ninguna	Ninguna

* Mismo nivel que una instrucción.

Máxima prioridad=1, mínima prioridad=18.

Tabla 4.3 Interrupciones enmascarables del MC68HC11F1.

V. MODELADO MATEMATICO DEL SISTEMA

5.1 MODELADO MATEMATICO DEL MOTOR DE CORRIENTE DIRECTA

5.1.1 Modelo matemático

En la figura 5.1, se puede observar el diagrama esquemático de un motor de corriente directa con excitación independiente controlado por el voltaje en la armadura. El análisis de este diagrama nos permitió realizar el planteamiento del modelo matemático del motor de corriente directa que se va a controlar. Es conveniente mencionar que el motor a controlar, es un motor con excitación independiente de imán permanente, lo que significa que en el circuito de la figura 5.1, en vez de tener un circuito eléctrico para producir el flujo en el entrehierro, éste es producido por un imán permanente. Con el fin de hacer este análisis más general, se planteó el circuito eléctrico de excitación presentado en la figura 5.1, no obstante, no se comete ningún error en los elementos del modelo matemático, debido a que si en la figura 5.1 se dibujara el flujo producido por un imán permanente, este flujo también sería constante y el análisis partiría de la ecuación 5.5.

En la figura:

V_f es el voltaje de campo y es constante.
I_f es la corriente de campo.
L_f es la inductancia del devanado de campo.
R_f es la resistencia del devanado de campo.
V_t es el voltaje en las terminales de la armadura.
I_a es la corriente de armadura.
L_a es la inductancia del devanado de armadura.
R_a es la resistencia del devanado de armadura.
E_a es la fuerza contraelectromotriz.
ω es la velocidad angular del eje del motor.

T es el par desarrollado por el motor.

J es el momento de inercia equivalente del motor.

f es el coeficiente de fricción viscosa.

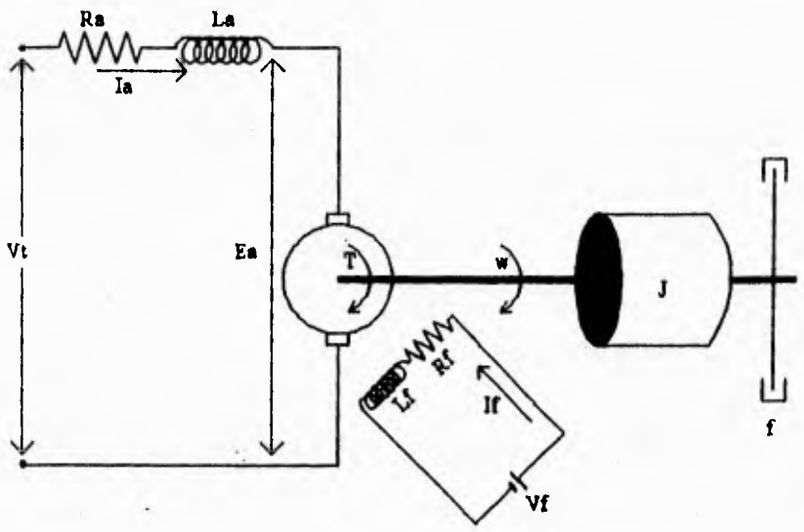


Figura 5.1 Diagrama de un motor de CD con excitación independiente.

De acuerdo con la figura 5.1 se sabe que el flujo ϕ en el entrehierro puede obtenerse de la siguiente forma:

$$\phi = K_f \, I_f \qquad (5.1)$$

donde K_f es una constante.

Además, se sabe que el par desarrollado por el motor (T) es proporcional al producto de la corriente de armadura (I_a) por el flujo ϕ, de forma que puede escribirse:

$$T \, \alpha \, I_a \, \phi \qquad (5.2)$$

y para encontrar una igualdad, en la ecuación anterior debe introducirse una constante (K_1):

$$T = K_1 \, i_a \, \phi \qquad (5.3)$$

Ahora, sustituyendo 5.1 en 5.3 se tiene:

$$T = K_1 \, I_a \, K_f \, I_f \qquad (5.4)$$

En este tipo de motor, el voltaje de campo y por lo tanto la corriente de campo se mantienen constantes, lo que implica que el flujo en el entrehierro sea también constante y entonces la ecuación del par resulta:

$$T = K \, I_a \qquad (5.5)$$

donde $K = K_1 K_f I_f$ es una constante del par motor.

Cuando la armadura está en rotación, se induce un voltaje proporcional al producto del flujo por la velocidad angular. Para un flujo constante, el voltaje inducido E_a es directamente proporcional a la velocidad angular, de forma que:

$$E_a = K_{fcem} \, \omega \qquad (5.6)$$

donde K_{fcem} es una constante de fuerza contraelectromotriz.

Por lo que, la ecuación diferencial del circuito resulta:

$$L_a \frac{dI_a}{dt} + R_a I_a + E_a = V_t \qquad (5.7)$$

La corriente de armadura produce el par que se aplica a la inercia, a la fricción y al par de la carga (T_L), por ello:

$$J \frac{d\omega}{dt} + f\omega + T_L = T \qquad (5.8)$$

Por lo tanto, igualando la ecuación 5.5 con la ecuación 5.8:

$$J\frac{d\omega}{dt} + f\omega + T_L = KI_a \qquad (5.9)$$

Entonces, las ecuaciones que definen al sistema son:

$$E_a = K_{fcem}\omega \qquad (5.10)$$

$$L_a\frac{dI_a}{dt} + R_aI_a + E_a = V_t \qquad (5.11)$$

$$J\frac{d\omega}{dt} + f\omega + T_L = KI_a \qquad (5.12)$$

Considerando condiciones iniciales nulas y obteniendo las transformadas de Laplace de las ecuaciones anteriores resulta:

$$E_a(s) = K_{fcem}\,W(s) \qquad (5.13)$$

$$(L_as + R_a)\,I_a(s) + E_a(s) = V_t(s) \qquad (5.14)$$

$$(Js + f)\,W(s) + T_L(s) = KI_a(s) \qquad (5.15)$$

Considerando a $V_t(s)$ como entrada, W(s) como salida y $T_L(s)$ como perturbación en el eje del motor, es posible construir un diagrama de bloques para representar a las ecuaciones 5.13, 5.14 y 5.15 y obtener a partir de este diagrama (figura 5.2) y aplicando el Teorema de Mason, una función de transferencia para el motor.

$$G_p(s) = \frac{W(s)}{T_L(s)} + \frac{W(s)}{V_t(s)} \qquad (5.16)$$

donde:

$$\frac{W(s)}{T_L(s)} = \frac{\dfrac{-1}{Js + f}}{1 + K\,K_{fcem}\left(\dfrac{1}{Js + f}\right)\left(\dfrac{1}{L_a s + R_a}\right)} \qquad (5.17)$$

$$\frac{W(s)}{V_t(s)} = \frac{K\left(\dfrac{1}{L_a s + R_a}\right)\left(\dfrac{1}{Js + f}\right)}{1 + KK_{fcem}\left(\dfrac{1}{Js + f}\right)\left(\dfrac{1}{L_a s + R_a}\right)} \qquad (5.18)$$

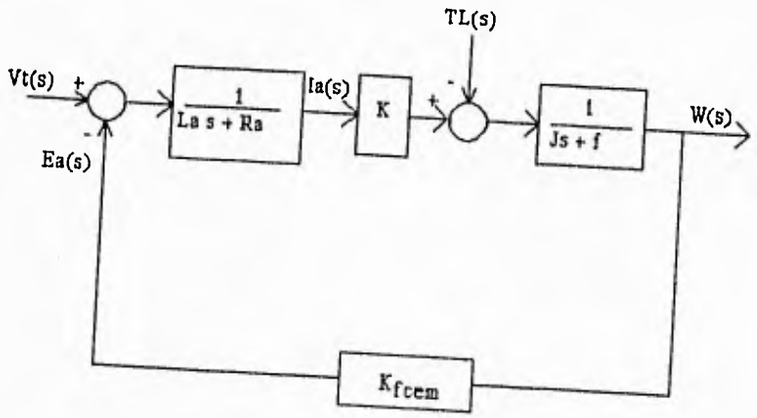


Figura 5.2 Diagrama a bloques de un motor de CD.

Por lo tanto:

$$G_p\ (s) = \frac{K\left(\frac{1}{L_a s + R_a}\right)\left(\frac{1}{Js + f}\right) - \frac{1}{Js + f}}{1 + KK_{fcem}\left(\frac{1}{Js + f}\right)\left(\frac{1}{L_a s + R_a}\right)} \qquad (5.19)$$

$$G_p(s) = \frac{K - L_a s - R_a}{(Js + f)\ (L_a s + R_a) + KK_{fcem}} \qquad (5.20)$$

$$G_p(s) = \frac{K - L_a s - R_a}{JL_a s^2 + (JR_a + fL_a)\, s + R_a f + K\, K_{fcem}} \qquad (5.21)$$

Debido a que el valor de La es muy pequeño (0.1mH), se considera despreciable, por lo que:

$$G_p(s) = \frac{K - R_a}{JR_a s + R_a f + K\, K_{fcem}} \qquad (5.22)$$

Finalmente:

$$G_p(s) = \frac{K_m}{\tau_m s + 1} \qquad (5.23)$$

donde:

$$K_m = \frac{(K - R_a)}{(R_a f + K\, K_{fcem})} \qquad (5.24)$$

K_m es la constante de ganancia del motor y dada la construcción del circuito que se describirá

posteriormente, esta ganancia se considerará igual a 1.

τ_m es la constante de tiempo del motor y resulta:

$$\tau_m = \frac{(JR_a)}{(R_a f + K K_{fcem})} \qquad (5.25)$$

La representación en forma discreta de esta planta, aplicando el método de aproximación ROC, es la siguiente:

$$G_p(z) = K_m \left(\frac{1 - e^{-\frac{1}{\tau_m} T}}{z - e^{-\frac{1}{\tau_m} T}} \right) \qquad (5.26)$$

5.1.2 Comprobación experimental del modelo matemático

Con el propósito de comprobar que el modelo matemático del motor, obtenido anteriormente, se apega lo suficiente a la realidad, se realizaron pruebas en el laboratorio. Estas pruebas consistieron básicamente en aplicar a las terminales de armadura del motor una entrada escalón, y posteriormente observar y analizar la respuesta de velocidad del motor en un osciloscopio y graficar dicha respuesta con un graficador. El transductor de velocidad a voltaje empleado fue un tacogenerador, mismo que se empleará en la implementación del control realimentado.

La gráfica en el osciloscopio demuestra que las simplificaciones que se hicieron en el modelo matemático, fueron adecuadas. La respuesta del motor ante una entrada escalón tiene la forma típica de la respuesta de un sistema de primer orden, lo cual se puede apreciar en la figura 5.3 que muestra la gráfica realizada por el graficador, por ello procedimos a medir la constante de tiempo en el osciloscopio por ser éste un instrumento de medición más preciso, τ_m, que resultó:

$$\tau_m = 1.16 \ [s] \qquad (5.27)$$

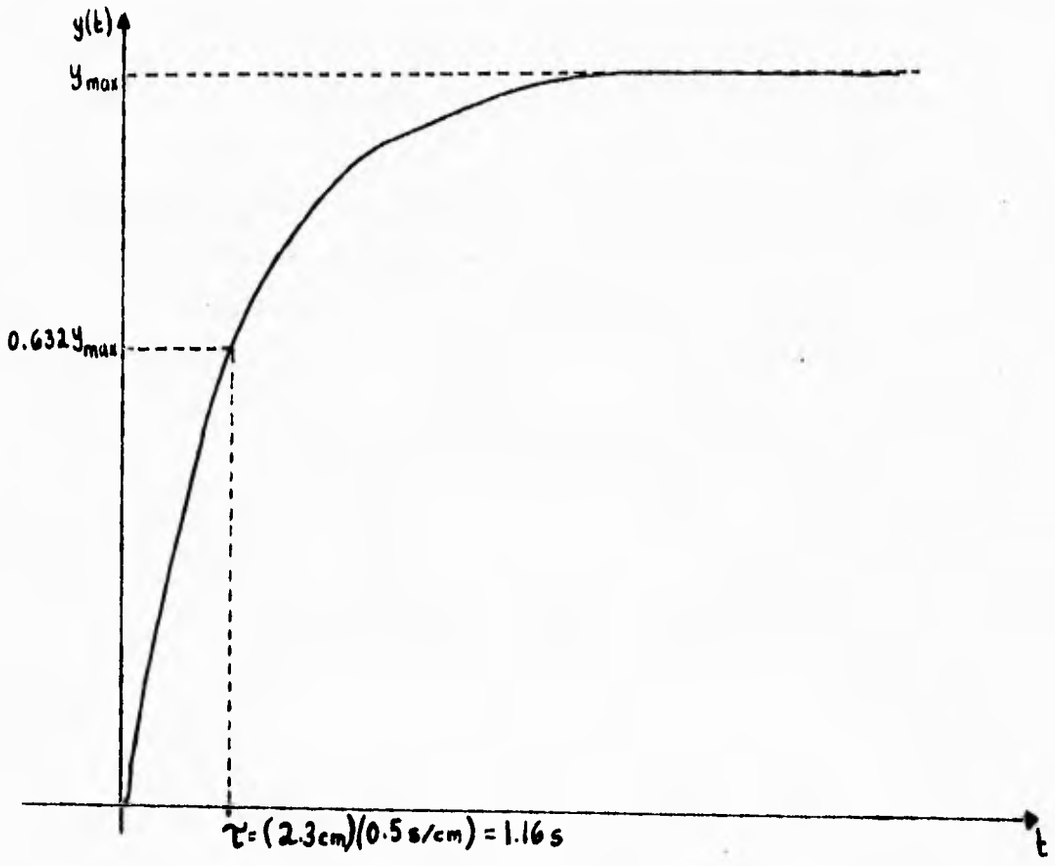


Figura 5.3 Respuesta del motor ante una entrada escalón.

5.2 MODELADO MATEMATICO DEL CONTROLADOR DIGITAL PID

5.2.1 Modelo matemático

El esquema de control que se emplea en este sistema corresponde al tipo Proporcional, Integral y Derivativo (PID), pues se desea conseguir un error de estado estacionario igual a cero y además contar con las ventajas que representa la combinación de las tres acciones de control de que dispone este controlador.

La ecuación matemática que describe al controlador es:

$$G_c(s) = \frac{U(s)}{E(s)} = K_p\left(1 + \frac{T_d s}{T_a s + 1} + \frac{1}{T_i s}\right) \qquad (5.28)$$

donde:

$G_c(s)$ es la función de transferencia del controlador.

U(s) es la señal de control.

E(s) es el error.

K_p es la constante de la acción proporcional.

T_d es la constante de la acción derivativa.

T_i es la constante de la acción integral.

y $\left(\frac{1}{T_a s + 1}\right)$ es un filtro, empleado básicamente para hacer posible la implementación física de la acción de control derivativa.

T_a es la constante del filtro y se define como:

$$T_a = \frac{T_d}{N} \qquad (5.29)$$

donde: $3 \leq N \leq 20$

La realización física del controlador se hará empleando, como ya se ha mencionado, un sistema multiprocesador con dos microcontroladores MC68HC11F1, y el algoritmo a programar en este sistema se desarrolla a continuación:

Aplicando las técnicas de control digital, se procede a discretizar el modelo matemático en forma continua que se describió anteriormente, entonces:

Para la acción proporcional:

forma continua: K_p

forma discreta: K_p

Para la acción de control integral, empleando el método de diferenciación hacia adelante:

forma continua: $\frac{K_p}{T_i\ s}$

forma discreta: $\frac{K_p T}{T_i}\ \frac{z^{-1}}{1 - z^{-1}}$

Para la acción derivativa con filtro y empleando el método de diferenciación en atraso:

forma continua: $\dfrac{K_p T_d s}{T_a s + 1}$

forma discreta: $K_p T_d \dfrac{(1 - z^{-1})}{T_a + T - T_a z^{-1}}$

Por lo tanto la ecuación de transferencia discreta del controlador es:

$$G_c(z) = K_p \left(1 + \frac{T}{T_i} \frac{z^{-1}}{1 - z^{-1}} + \frac{T_d (1 - z^{-1})}{T_a + T - T_a z^{-1}} \right) \quad (5.30)$$

desarrollando se tiene:

$$\frac{U(z)}{E(z)} = \frac{K_p [T_a (T_i - T) + T_d T_i]}{T_i T_a z^{-2} + [T_i (-2T_a - T)] z^{-1} + T_i (T_a + T)} z^{-2} +$$

$$+ \frac{K_p [T_i (-2T_a - T) + T (T_a + T) - 2T_d T_i]}{T_i T_a z^{-2} + [T_i (-2T_a - T)] z^{-1} + T_i (T_a + T)} z^{-1} +$$

$$+ \frac{K_p T_i (T_a + T + T_d)}{T_i T_a z^{-2} + [T_i (-2T_a - T)] z^{-1} + T_i (T_a + T)}$$

(5.31)

ordenando:

$$T_i T_a z^{-2} U(z) - [T_i(2T_a + T)] z^{-1} U(z) + T_i(T_a + T) U(z) =$$
$$K_p [T_a(T_i - T) + T_d T_i] z^{-2} E(z) +$$
$$+ K_p [-T_i(2T_a + T) + T(T_a + T) - 2T_d T_i] z^{-1} E(z) +$$
$$+ K_p [T_i(T_a + T + T_d)] E(z)$$

(5.32)

antitransformando:

$$T_i T_a u(k-2) - [T_i(2T_a + T)] u(k-1) + T_i(T_a + T) u(k) =$$
$$K_p [T_a(T_i - T) + T_d T_i] e(k-2) +$$
$$+ K_p [-T_i(2T_a + T) + T(T_a + T) - 2T_d T_i] e(k-1) +$$
$$+ K_p [T_i(T_a + T + T_d)] e(k)$$

(5.33)

finalmente la ecuación en diferencias a programar en el sistema multiprocesador es:

$$u(k) = \frac{1}{T_i(T_a + T)} [\ K_p [T_a(T_i - T) + T_d T_i] e(k-2) +$$
$$+ K_p [-T_i(2T_a + T) + T(T_a + T) - 2T_d T_i] e(k-1) +$$
$$+ K_p [T_i(T_a + T + T_d)] e(k) - [T_i T_a] u(k-2) +$$
$$+ [T_i(2T_a + T)]\ u(k-1)\ ]$$

(5.34)

5.2.2 Sintonización

El proceso de sintonización de controladores, consiste en ajustar los parámetros del controlador para producir la respuesta de malla cerrada deseada. La sintonización de un controlador proporcional integral y derivativo presenta algunas dificultades puesto que son tres los parámetros que se deben ajustar, la ganancia K_p, el tiempo integral T_i y el tiempo derivativo T_d.

Uno de los métodos de sintonización más comúnmente empleado es el de la curva de reacción de Ziegler-Nichols, aunque existen muchos otros. Sin embargo, todos estos procedimientos generan parámetros iniciales y a partir de éstos es todavía necesario hacer un reajuste en caso de no obtener la respuesta deseada.

Ziegler y Nichols reconocieron que la respuesta a escalón de muchos sistemas de control de procesos tiene la curva en forma de S mostrada en la figura 5.4, denominada curva de reacción del proceso y se puede generar experimentalmente excitando al proceso en malla abierta con una función escalón y obteniendo su respuesta.

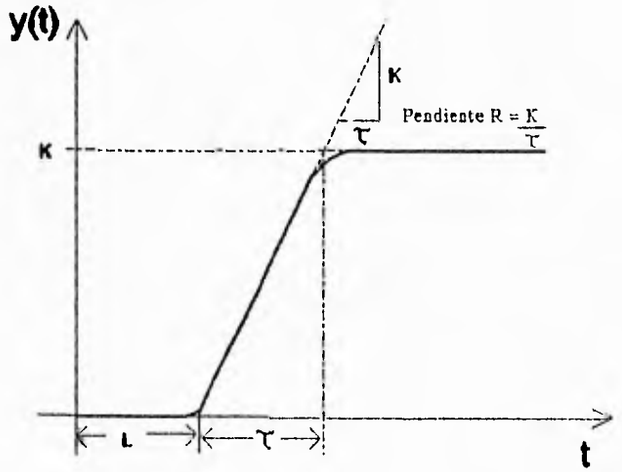


Figura 5.4 Curva de reacción del proceso.

Para esta curva se determina un punto o puntos de inflexión y sobre el que tiene mayor pendiente se traza una recta tangente a la curva. Esta recta intersecta al eje del tiempo, y el tiempo muerto (L) que caracteriza al sistema, se mide desde el origen hasta este punto de intersección, como se muestra en la figura 5.4. El otro dato que se requiere para el cálculo de los parámetros del controlador es R, que corresponde a la pendiente de la recta tangente a la respuesta escalón, que pasa por el punto de inflexión (R).

Una vez teniendo L y R, las fórmulas de sintonización para el controlador PID son:

$$K_p = \frac{1.2}{RL}; \quad T_i = \frac{L}{0.5}; \quad T_d = 0.5L \qquad (5.35)$$

Siguiendo el método anterior, se generó la curva de reacción del motor y a partir de esta curva fue posible obtener los datos R y L, según se puede observar en la figura 5.5. De acuerdo con esta figura, se puede ver que en la curva de reacción del motor, el punto de inflexión, se encuentra muy cercano al origen, por lo que el tiempo muerto L es muy pequeño.

De la figura 5.5 se obtiene:

$$R = 8.02\ [V/s]; \qquad L = 0.1\ [s] \qquad (5.36)$$

Tomando en cuenta los datos R y L y las fórmulas para obtener los parámetros del controlador, se tiene que:

$$K_p = 1.496\ ; \qquad T_d = 0.05; \qquad T_i = 0.2 \qquad (5.37)$$

Con estos valores se realizó una simulación en la computadora con el paquete CC y se observó que era necesario ajustarlos. Este ajuste de parámetros se hizo gradualmente, tomando en cuenta la función que cada uno de estos desempeña en el controlador PID. En cada ajuste se realizó una simulación, hasta que se obtuvieron los parámetros que dieron las características dinámicas que requeríamos; pequeño sobrepaso y tiempo de asentamiento menor que tres veces la constante de tiempo del motor.

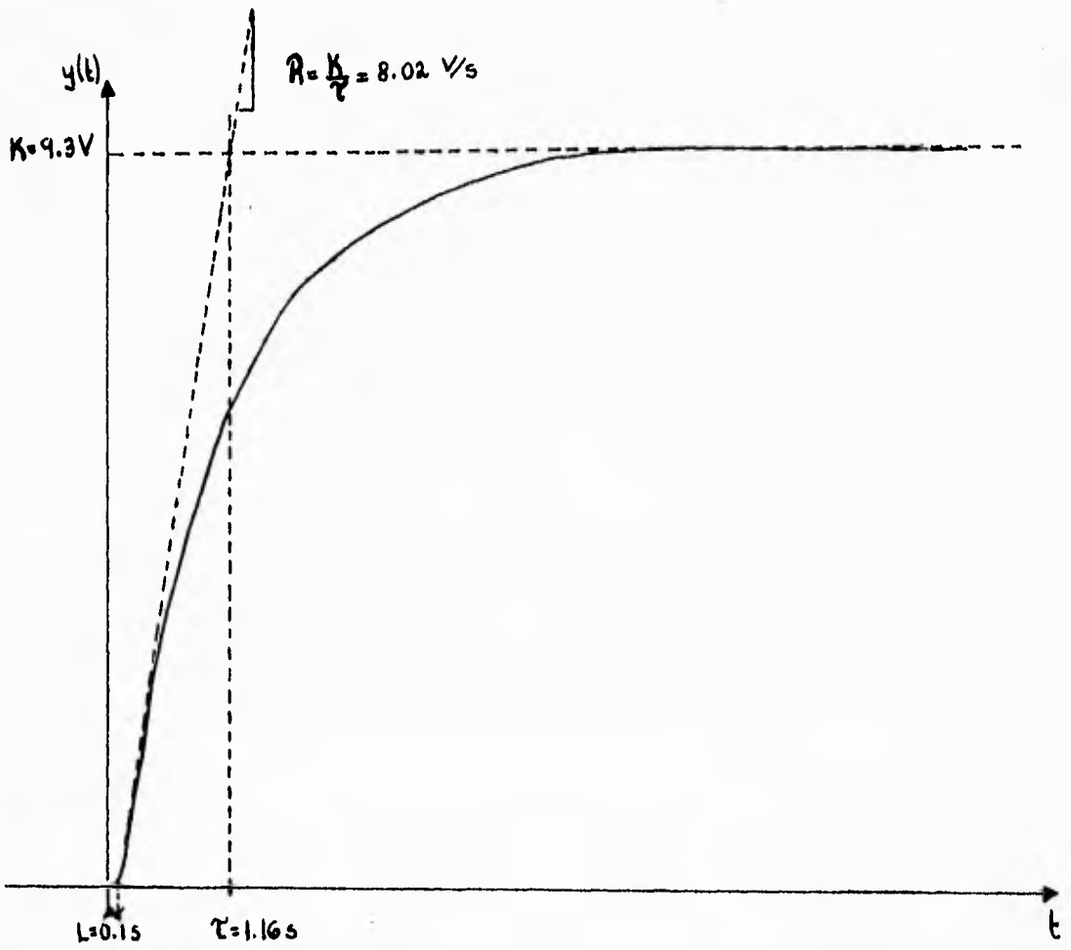


Figura 5.5 Curva de reacción del motor.

Los parámetros resultantes son:

$$K_p = 1.5 \; ; \quad T_d = 0.1 \; ; \quad T_i = 0.7 \qquad (5.38)$$

$$N = 10 \quad \therefore \quad Ta = 0.025$$

En el apéndice A se muestran dos de las simulaciones que realizamos con el paquete CC. La primera, con los parámetros iniciales obtenidos por el método de Ziegler y Nichols, y la segunda con los parámetros con los que será programado el controlador en el sistema multiprocesador. Estas simulaciones incluyen la respuesta transitoria del sistema en malla cerrada y el lugar geométrico de las raíces.

VI. IMPLEMENTACION DEL HARDWARE DEL SISTEMA

6.1 INTRODUCCION

Para hacer posible la implementación del sistema de control, fue necesario el diseño y la construcción de los circuitos que se mencionan a continuación:

- Una tarjeta principal a la que hemos llamado "Sistema Multiprocesador" que incluye un circuito de interfaz con una computadora personal.
- Un circuito de interfaz de entrada al sistema multiprocesador.
- Un circuito de interfaz de salida del sistema multiprocesador.

La figura 6.1 ilustra la ubicación y la función de cada uno de estos circuitos dentro del sistema de control.

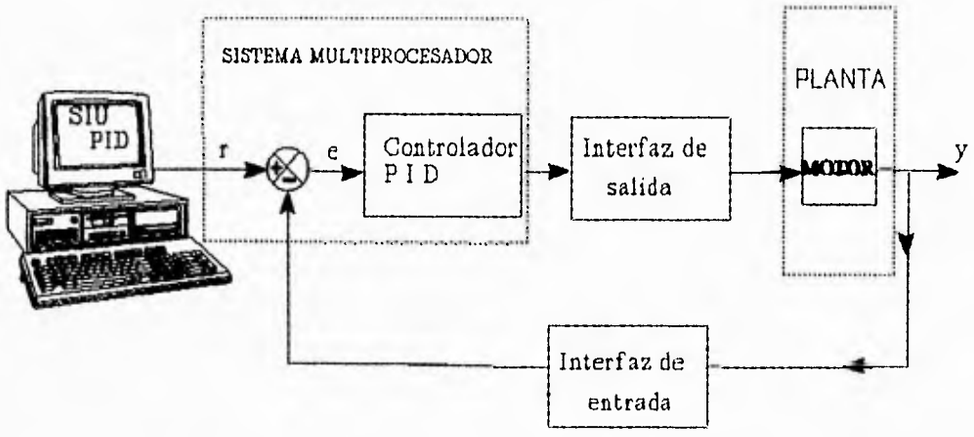


Figura 6.1 Diagrama a bloques del sistema de control PID para un motor de CD.

6.2 TARJETA PRINCIPAL: SISTEMA MULTIPROCESADOR

Esta tarjeta principal se encargará, en forma general, de procesar el algoritmo de control, de establecer la comunicación con la computadora a través de la interfaz RS-232 y con los circuitos de interfaz de entrada y salida. El diagrama esquemático del sistema multiprocesador se muestra en la figura 6.3.

Esta tarjeta, llamada, sistema multiprocesador, está basada en dos microcontroladores MC68HC11F1, configurados en modo expandido no multiplexado e interconectados entre si por medio de su respectivo puerto de comunicación síncrona (SPI), con el propósito de que puedan procesar datos en forma simultánea y cuando así se requiera compartan datos o resultados. Debido a su configuración en modo expandido, cada uno de los microcontroladores hace uso de una memoria UV-PROM de 32 Kbytes y tiene la capacidad de utilizar una memoria RAM de 8Kbytes.

De esta forma, en el sistema multiprocesador (figura 6.2), ambos procesadores se comunican y cooperan entre sí para dar solución a un problema.

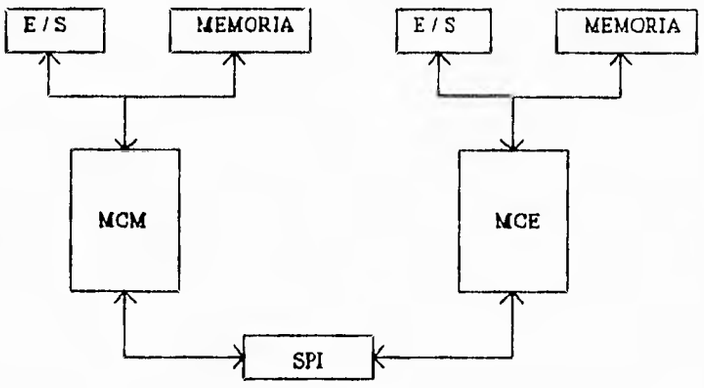


Figura 6.2 Diagrama a bloques de un sistema multiprocesador.

Es necesario hacer notar que a pesar de que se trata de una tarjeta con 2 microcontroladores con dispositivos externos y conexiones idénticas dentro de la tarjeta, existen dispositivos que son comunes para ambos microcontroladores. Uno se estos es el interruptor utilizado en la terminal de RESET, que como se explica en el capítulo IV, esta es una terminal bidireccional por la que puede viajar una señal de reinicialización de un microcontrolador hacia otro o bien una señal externa de reinicio al sistema. Otros dispositivo comunes en la tarjeta son tanto el LED indicador de encendido como el cristal de cuarzo de 8MHz. En cambio hay otros elementos en los que se ha decidido que sean de uso único para cada microcontrolador y es por ello que cada uno tiene su propio circuito MAX232, sus propios interruptores para requerimientos de interrupción por hardware (IRQ, XIRQ) y sus propios interruptores para seleccionar el modo de operación de cada microcontrolador. Por tratarse de una tarjeta de propósito general se utilizaron conectores planos de 26 terminales para extraer de la tarjeta el bus de datos, las 5 terminales correspondientes a los bits menos significativos del bus direcciones, las 4 terminales menos significativas del puerto E, los chip-select CSIO1 y CSIO2, las terminales R/W y E, y los niveles de +5V y GND.

Por estar trabajando en modo expandido, en ambos microcontroladores, los puertos B y F forman el bus de direcciones externas y el puerto C funciona como bus externo de datos. Las terminales del puerto D, PD0-PD5, son empleadas por el sistema de comunicación síncrona (SPI) y las terminales del puerto G, PG4-PG7, funcionan como chip-selects programables, teniendo disponibles como terminales de entrada/salida de propósito general el puerto A y las terminales PG0-PG3 y como terminales de solo entrada las correspondientes a PE0-PE3.

Cada uno de los microcontroladores tiene asignada una función específica para la realización del algoritmo de control. Uno de ellos está configurado como maestro y el otro como esclavo, el microcontrolador maestro (MCM) es el encargado de recibir los datos provenientes de la interfaz de entrada (muestras de la velocidad del motor), calcular el valor del error, dependiendo de la muestra tomada y la referencia o velocidad deseada, y tomando en cuenta estos datos y los cálculos de errores anteriores, obtener un resultado parcial de la salida u(k). Por su parte, el microcontrolador esclavo (MCE) se encargará de hacer un segundo cálculo parcial, tomando en cuenta los resultados

de salidas anteriores (u(k-2), u(k-1)) y posteriormente obtendrá el resultado total de u(k) haciendo la suma de su propio resultado con el que por medio del SPI le entregó el MCM. Esta salida será enviada por el MCE al circuito de interfaz de salida para que el motor reaccione en forma correcta ante la petición de velocidad que se le haga. El controlador PID debe realizar este proceso en un tiempo máximo de 100 ms, por ser éste el periodo de muestreo elegido.

El circuito de interfaz de entrada se conecta al MCM por medio del puerto A y las terminales PG0 y PG1 del puerto G. El circuito de interfaz de salida se conecta al MCE por medio del puerto A y las terminales PG0-PG3.

Esta tarjeta se diseñó asumiendo que la alimentación de voltaje se obtendrá de una fuente regulada externa. Por ello se tiene un conector de dos terminales para recibir la alimentación: +5V y GND.

6.3 INTERFAZ A LA COMPUTADORA PERSONAL

Para enlazar la tarjeta principal con la computadora personal se utilizó la interfaz RS-232, la cual es una de las mas difundidas para enlazar equipos en transmisiones de datos. Los niveles digitales de la señal a transmitir son 0 y 5V a la salida del microcontrolador; con el fin de incrementar la relación señal a ruido y lograr una mejor transmisión, la interfaz RS-232 maneja los niveles de +12 y -12 V. Para realizar la traducción de niveles de 0V a +12V y 5V a -12V se utiliza el circuito MAX232. Estos niveles viajan a través de un cable que puede tener una longitud máxima de 15 metros para asegurar que no haya pérdidas ni degradación de los datos. Se utilizan conectores DB-9 en ambos extremos del cable (ver figura 6.4).

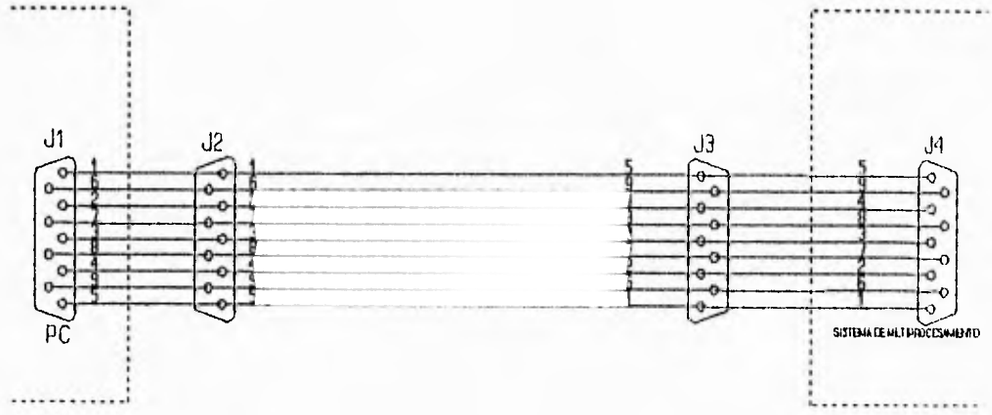


Figura 6.4 Cable para la interfaz RS-232

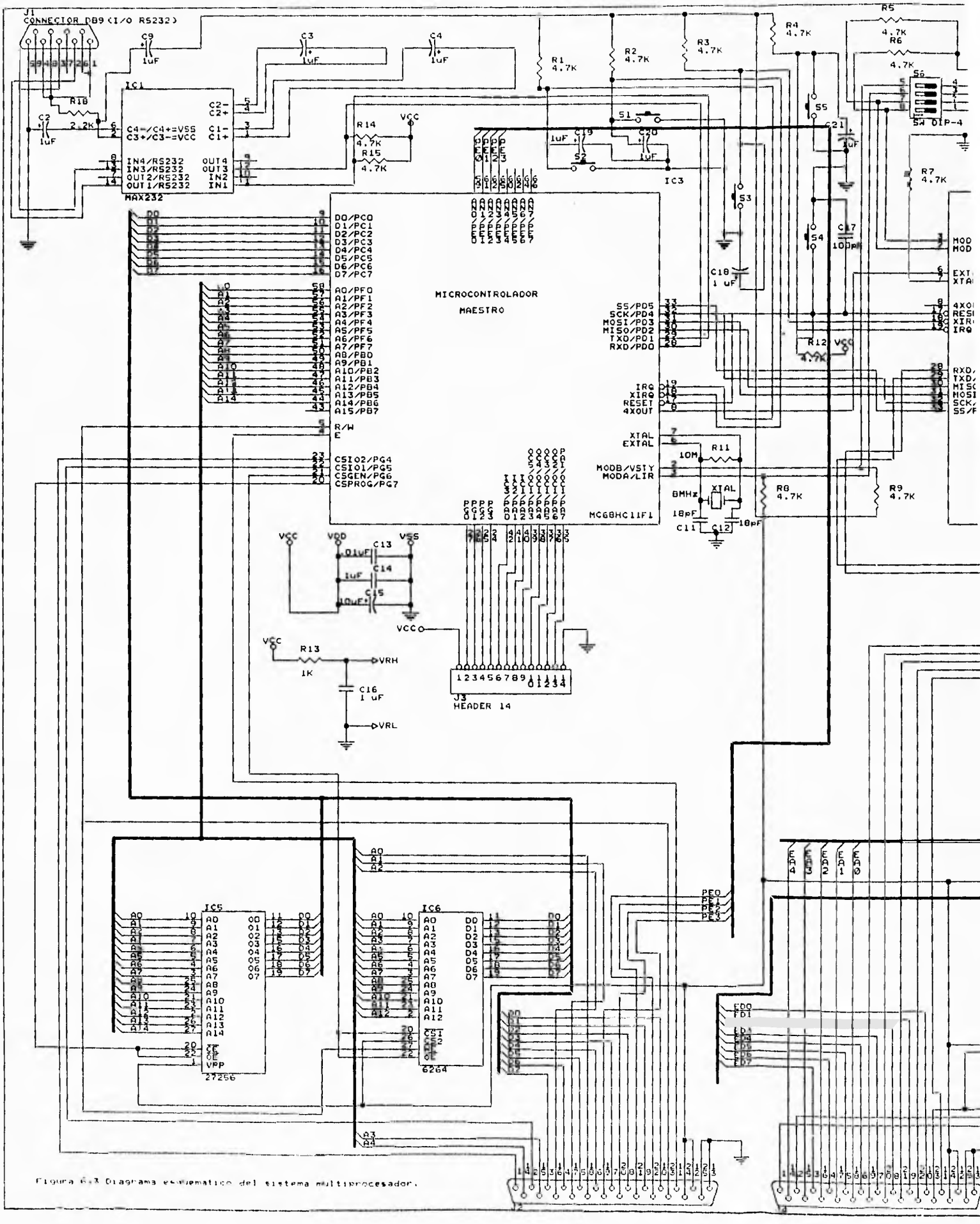
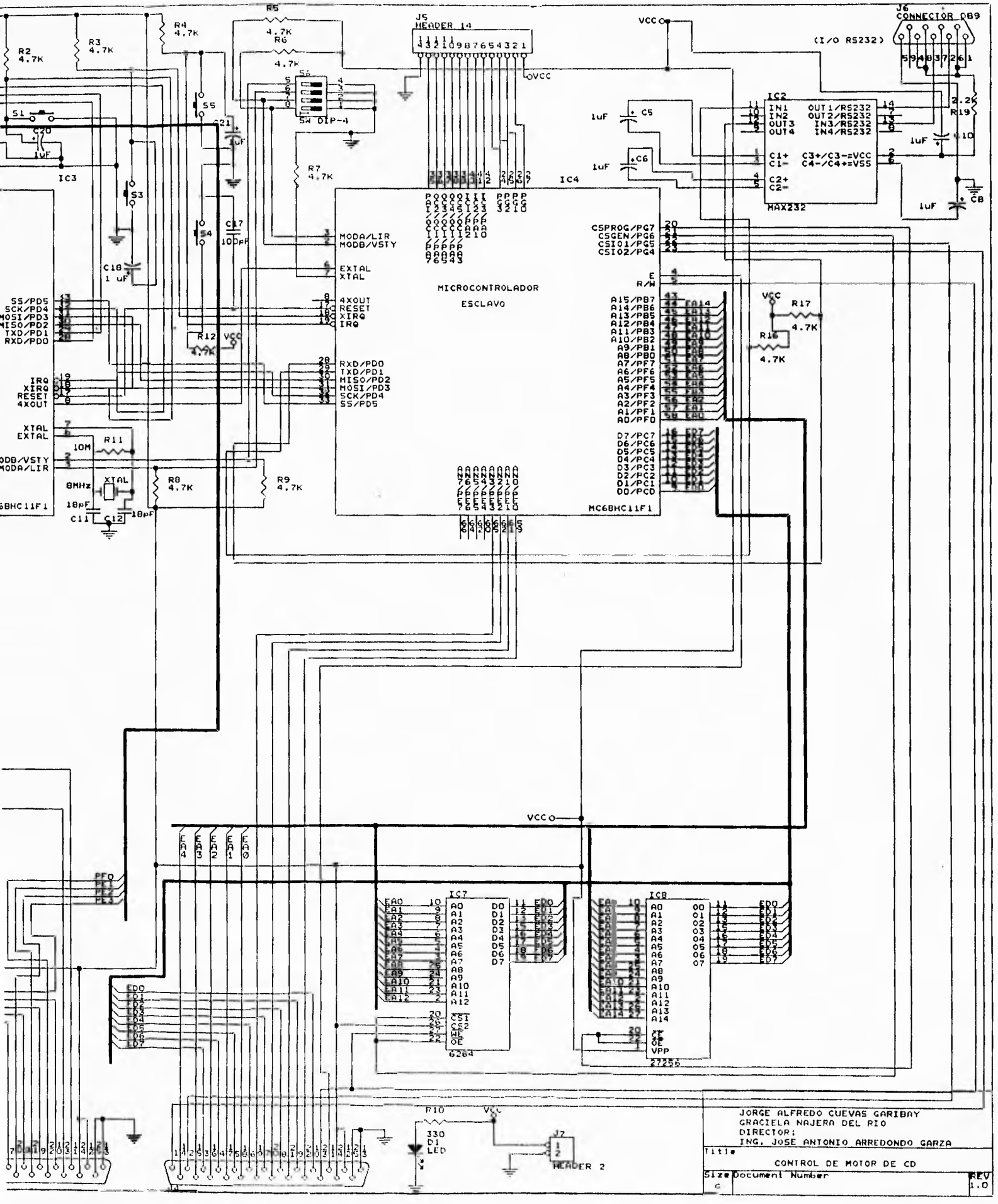


Figura 6.3 Diagrama esquematico del sistema multiprocesador.


MICROCONTROLADOR
ESCLAVO
MC68HC11F1
MAX232
IC2
IC3
IC4
IC7
IC8
6264
27256
J5
HEADER 14
J6
CONNECTOR DB9
(I/O RS232)
J7
HEADER 2
5W DIP-4
8MHz
XTAL
330
D1
LED
JORGE ALFREDO CUEVAS GARIBAY
GRACIELA NAJERA DEL RIO
DIRECTOR:
ING. JOSE ANTONIO ARREDONDO GARZA
Title
CONTROL DE MOTOR DE CD
Size
Document Number
REV
1.0

6.4 CIRCUITO DE INTERFAZ DE ENTRADA AL SISTEMA MULTIPROCESADOR

Este circuito incluye los siguientes bloques:

- Transductor de Velocidad.
- Circuito de adecuación de la señal proveniente del tacogenerador al convertidor analógico a digital.
- Convertidor Analógico a Digital (Convertidor A/D) de 12 bits.

A continuación se presenta una descripción detallada de cada uno de los bloques anteriores.

6.4.1 Transductor de velocidad

La función del transductor de velocidad, es la de convertir el valor de la velocidad de la flecha del motor, a un valor de voltaje proporcional a dicha velocidad.

El transductor empleado en este proyecto es un tacogenerador. Este dispositivo es un puro y simple generador de CD que consiste en una bobina móvil conectada al eje del motor y suspendida en el campo magnético de un imán permanente. Se genera un voltaje por el movimiento de la bobina en el campo, de acuerdo con la ley de Faraday:

$$V = N \frac{d\phi}{dt} \qquad (6.1)$$

donde: V es el voltaje inducido, N es el número de vueltas del embobinado y ϕ es el flujo a través de cada espira. Este flujo ϕ dependerá de la posición del embobinado, se puede ubicar esta posición por medio de un ángulo θ, definido como el ángulo formado por la normal al plano del embobinado (A) y la dirección del campo magnético (B). De esta manera, el flujo magnético a través de cada espira, para una posición cualquiera θ del embobinado, será determinado por la siguiente expresión:

$$\phi = \iint_A \vec{B} \cdot d\vec{A} \qquad (6.2)$$

tomando en cuenta que el campo magnético (B) es constante, la ecuación anterior resulta:

$$\phi = B\,A\cos\theta \qquad (6.3)$$

sustituyendo la última expresión en la ley de Faraday, se obtiene:

$$V = N\,B\,A\,\frac{d(\cos\theta)}{dt} \qquad (6.4)$$

resolviendo la derivada:

$$V = -\,N\,B\,A\,sen\,\theta\,\frac{d\theta}{dt} \qquad (6.5)$$

y la $d\theta/dt$ es la velocidad angular (w) del embobinado; por lo que:

$$V = -\,N\,B\,A\,\omega\,sen\theta \qquad (6.6)$$

En la figura 6.5 se tiene una representación esquemática del tacogenerador que será empleado. Por la forma en que están conectadas las escobillas, y tomando en cuenta que el tacogenerador contiene una gran cantidad de embobinados orientados de forma que el voltaje inducido sea continuo, el voltaje generado proporcional a la velocidad angular del eje del motor, según los datos de placa del tacogenerador, mostrados en la tabla 1, es:

$$V_G = 2.5V/Krpm \qquad (6.7)$$

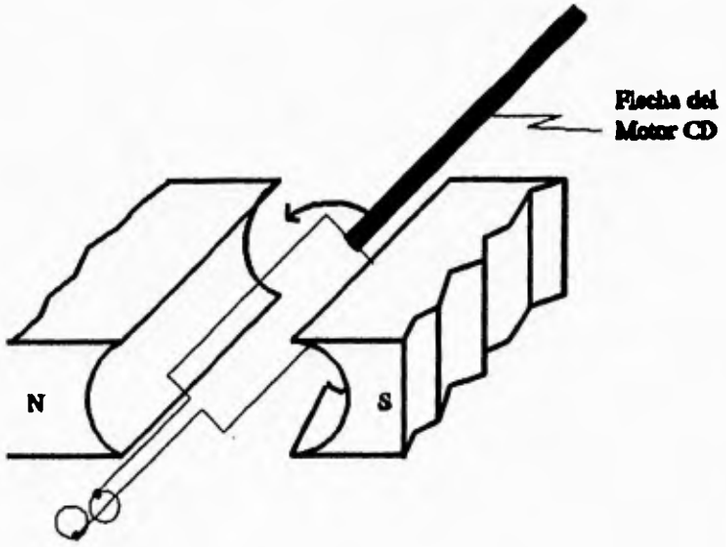


Figura 6.5 Representación esquemática del tacogenerador.

DC ANALOG TACHOMETER
Constante de voltaje: 2.5 V/KRPM Rizo de Voltaje: 1.5 % pp max Resistencia de carga 10KΩ min

Tabla 1. Datos de placa del tacogenerador.

En base a pruebas de laboratorio que se le aplicaron al tacogenerador empleado, se obtuvo su gráfica de respuesta ante diferentes velocidades de entrada, misma que se muestra en la figura 6.6. En esta gráfica se aprecia cómo este transductor responde en forma lineal en el rango

comprendido entre 800 y 3720 rpm, de 0 a 800 rpm, el comportamiento es no lineal. Por ello, empleando este transductor no es posible tener una lectura precisa a bajas velocidades.

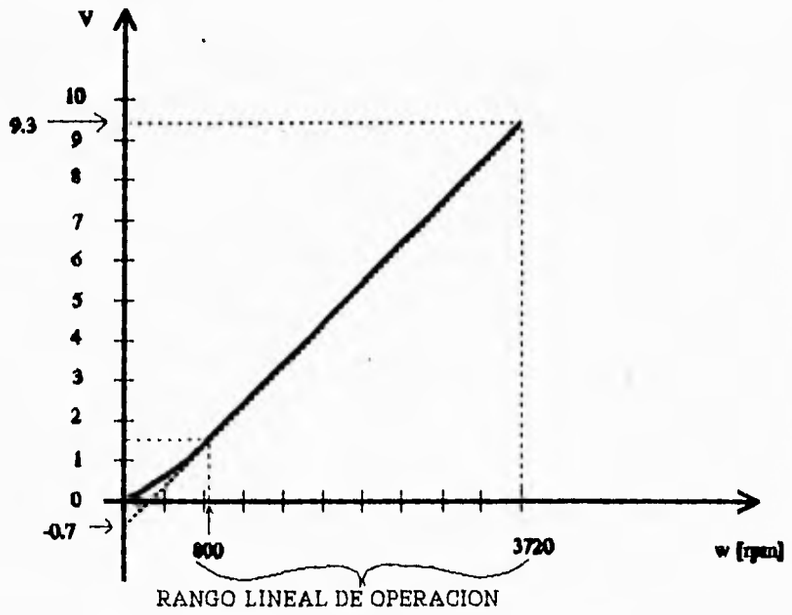


Figura 6.6 Curva de respuesta del Tacogenerador empleado.

Otra Opción

Sabiendo que en muchos casos, no será posible disponer de un tacogenerador como transductor de velocidad, a continuación presentamos otra solución factible y comprobada para construir un sistema transductor de velocidad.

El requerimiento básico de este sistema transductor de velocidad es mantener la siguiente relación lineal:

Voltaje máximo de salida del sensor, 5V, para una velocidad angular de 3720 rpm, donde 3720 rpm corresponde a la velocidad máxima de la flecha del motor, y un voltaje mínimo de salida

del sensor, 0V, para una velocidad angular de 0 rpm.

Este sensor estaría constituido básicamente por:

- Un disco ranurado en la flecha del motor
- Un optointerruptor (sensor óptico)
- Un comparador de nivel
- Un convertidor de Frecuencia a Voltaje

La salida del transductor (voltaje proporcional a la velocidad) sería enviada a un convertidor A/D de 12 bits.

Disco ranurado.

Tomando en cuenta que el rango de operación lineal del convertidor de frecuencia a voltaje es de 1KHz a 10 KHz y que el valor de la velocidad máxima de nuestro motor es de 3720 rpm, o bien de 62 [rev/seg], es necesario multiplicar esta frecuencia por un valor tal que sea posible operar el convertidor de frecuencia a voltaje en el rango lineal. Es por ello que se requiere manufacturar un disco con 160 ranuras, de forma que: (62)(160) es aproximadamente 10KHz.

Optointerruptor.

El optointerruptor a utilizar es el circuito integrado H21A1 de Motorola, el cual consiste en un diodo emisor de luz infrarroja y un fototransistor de silicio npn, ambos en un encapsulado de plástico. Este encapsulado tiene una ranura que separa al diodo emisor de luz infrarroja y al fototransistor. Polarizando el fototransistor con una resistencia de 7.5 KΩ, del colector hacia 15V, llevando el emisor hacia tierra y tomando la salida por el colector, el nivel alto será de 15V y el

nivel bajo de 0.2V. Así al introducir en esta ranura cualquier material, ya sea plástico o metal, que obstruya el paso de luz del diodo emisor de luz hacia el fototransistor, este fototransistor entra en la región de corte y por lo tanto la corriente de colector a emisor es igual a cero, de tal manera que el circuito presentará a su salida un nivel alto de voltaje. En forma opuesta, cuando no exista material alguno que se interponga al paso de la luz infrarroja, el fototransistor se encontrará en saturación y por lo tanto en la salida del circuito habrá un nivel bajo, aproximadamente de 0.2V que corresponde al voltaje V_{CE} de saturación. En la figura 6.7 se muestra el diagrama esquemático del H21A1.

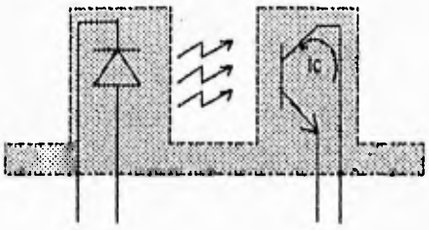


Figura 6.7 Diagrama esquemático del circuito integrado H21A1.

En la figura 6.8 mostramos la forma en la que se está utilizando el disco ranurado junto con el optointerruptor para hacer que cuando gire el disco, obstruya periodicamente el paso de luz; cuando el motor esté operando a su máxima velocidad, esta frecuencia debe ser de 10KHz. Debido a que esta frecuencia requerida de conmutación, es igual a la frecuencia máxima de conmutación especificada por Motorola para el H21A1, es necesario conectar la salida de este circuito, a un comparador de nivel con histéresis, con el fin de obtener esta misma frecuencia en niveles digitales sin deformaciones, ni ruido. La señal resultante del comparador será la entrada al convertidor de frecuencia a voltaje.

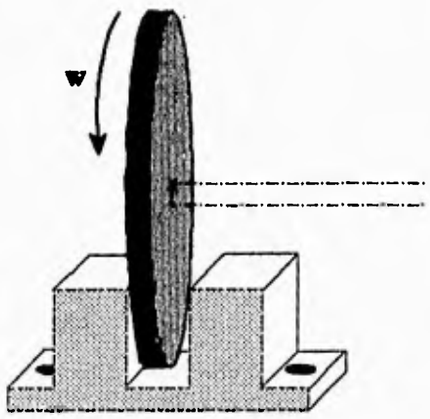


Figure 6.8 H21A1 y disco renurado.

Comperedor de nivel.

Este comparedor de nivel, se implemente básicamante con dos objetivos, los cueles se discuten a continueción. Uno de estos fines es eliminer el ruido que pudiere entrar a la señal en el trayecto, desde el optointerruptor colocado como lo ilustra la figure 6.8 (cerce de la fleche del motor), heste el comperedor. Otro de los fines del comparedor es limitar e 2 el número de niveles de la señel provenienta del optointerruptor, ye que debido a irregularidades en las ranuras del disco, los niveles de volteje resultan muy variebles, por ello, se tomó un criterio para esteblecer un nivel da volteje con el cuel se pudiera hecer le compareción. Se consideró adecuedo tomar un voltaje de compereción de 5V y una histéresis de ±0.05 V pere eliminer el ruido del que se habló anteriormente.

El circuito integredo que nos permitió cumplir ambos objetivos fue el LM311. A la salida de este circuito se tendrá une señal con la misme frecuencia que la señal de entrade pero únicamente tendré dos niveles de voltaje eproximedamente 0.4V y 15V (figure 6.9).

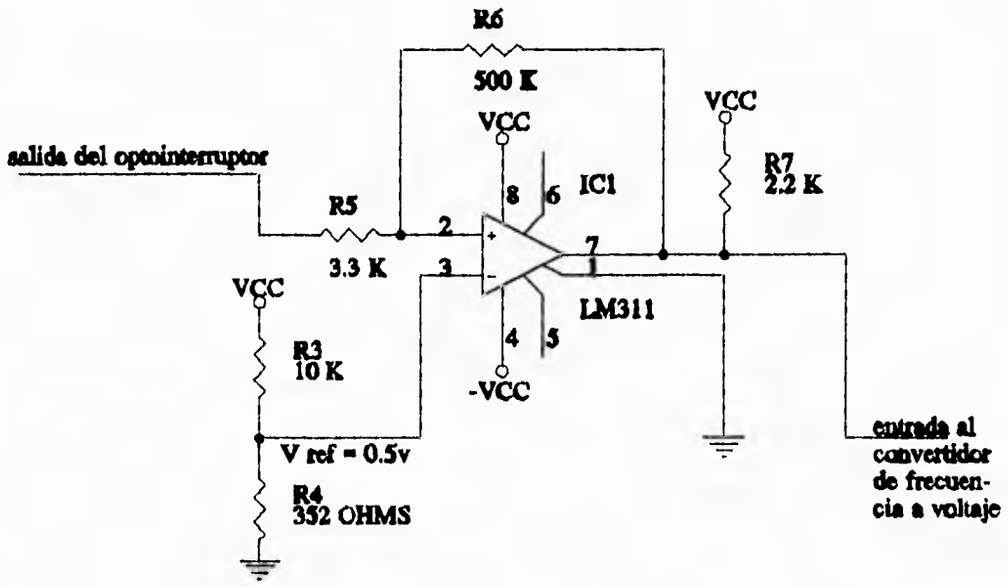


Figura 6.9 Configuración del LM311 en su función de comparador con histéresis.

Convertidor de Frecuencia a Voltaje.

El convertidor de frecuencia a voltaje a utilizar puede ser el LM2907. El voltaje de salida de este circuito está dado por la siguiente expresión:

$$V_0 = V_{cc} \times F_{in} \times C_1 \times R_1 \times K \qquad (6.8)$$

donde:

K es la constante de ganancia y es típicamente igual a 1

C_1 se recomienda que sea mayor que 500 pf

La conexión final de este circuito para obtener un voltaje proporcional en función de la

velocidad del motor se muestra en la figura 6.10:

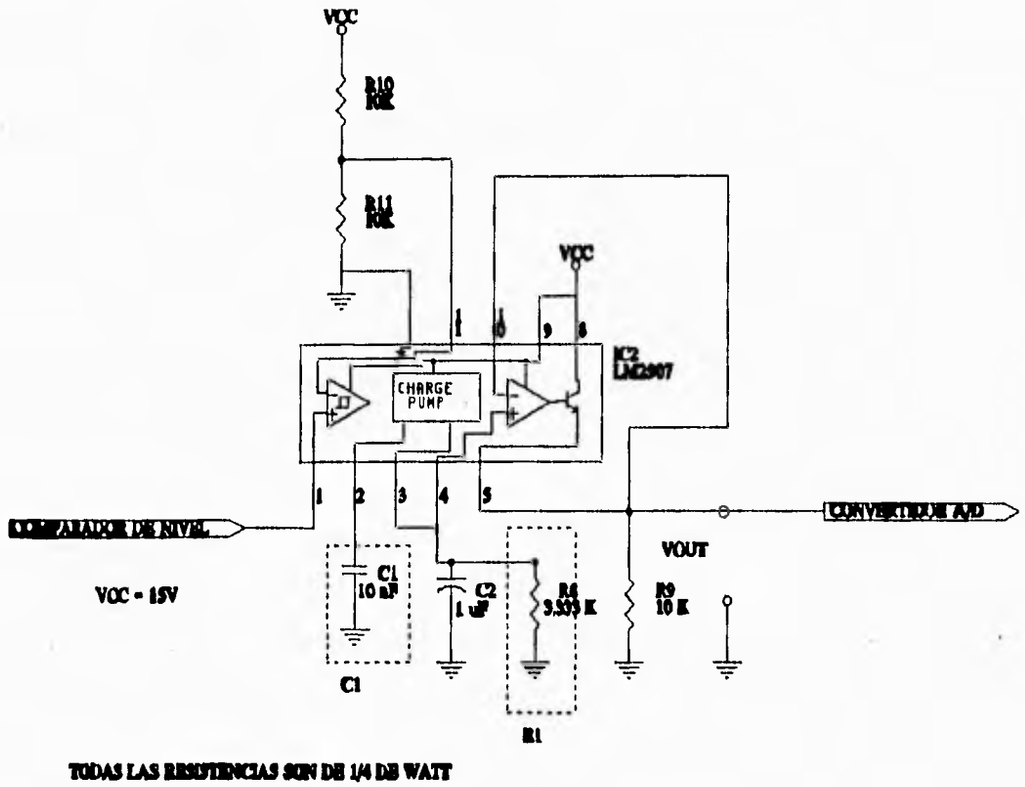


Figura 6.10 Configuración final del LM2907.

Con este circuito se concluye el diseño de un transductor que puede ser empleado en cualquier motor en caso de no disponer de un tacogenerador como el que nosotros empleamos en este proyecto.

6.4.2 Circuito de adecuación de la señal de entrada proveniente del tacogenerador, al convertidor analógico a digital

Cuando el motor se alimenta con 10 V, opera a máxima velocidad (3720 rpm) y en las terminales de salida del tacogenerador se obtiene un voltaje de 9.3V, por su parte, el convertidor A/D opera con un voltaje de entrada máximo de 5V. Por lo que se hace necesario emplear un circuito de adecuación de la señal para que el convertidor A/D realice su función.

Dentro del rango de operación lineal, independientemente del voltaje en las terminales del tacogenerador, el cual puede variar entre 1.45V y 9.3V, al sumar 0.7V y posteriormente dividir este resultado entre 2, se obtendrá el voltaje proporcional a la velocidad del motor, que se ajusta a las características de funcionamiento del convertidor A/D.

De esta forma, a máxima velocidad, deben llegar 5V al convertidor A/D y así tener un resultado de conversión $FFF, al 50% de la velocidad máxima del motor, se deben presentar 2.5 V a la entrada del convertidor A/D con el fin de tener una conversión digital $7FF, y así sucesivamente para cualquier velocidad en el rango de operación lineal del tacogenerador. Por ello el circuito de adecuación consta de un amplificador operacional en configuración sumador inversor de ganancia unitaria y de un amplificador operacional inversor con ganancia de 0.5. El circuito integrado empleado es el TL082 (IC3 en la figura 6.12). El circuito sumador inversor (IC3A) agrega 0.7V a la señal proveniente del tacogenerador (el potenciómetro POT1 permite ajustar el voltaje de 0.7V), posteriormente el amplificador operacional inversor (IC3B) con ganancia de 0.5 divide la señal de salida de IC3A entre 2. Logrando de esta forma la adecuación de la señal de entrada al convertidor A/D ADC1205.

Con el fin de calibrar de forma exacta el voltaje de entrada al convertidor A/D se utilizó el resistor variable POT2.

El relevador DIA050000 de estado sólido (IC4) que se observa en la figura 6.12 tiene la

función de proteger al convertidor A/D el cual, por seguridad, no debe recibir ningún voltaje si no está polarizado.

6.4.3 Convertidor Analógico a Digital (Convertidor A/D)

Para desempeñar esta función se utilizó el circuito integrado de National Semiconductor ADC1205, el cual es un convertidor analógico a digital de 12 bits y 1 bit extra para signo. El encapsulado es del tipo DIP con 24 pines. El resultado de las conversiones las proporciona en 2 bytes, uno a la vez, a través de su puerto de salida de información de 8 terminales.

De acuerdo con las hojas de especificaciones de este circuito, el tiempo en que se realiza la conversión del dato, de su forma analógica a su forma digital es de 109 μs para condiciones de funcionamiento similares a las que serán empleadas en nuestro circuito, es decir para un voltaje de polarización DVcc=AVcc=5.0V (terminales 6 y 24 del circuito integrado), f_{CLK}=1MHz y temperatura ambiente $T_A \approx 25°C$, el esquema de conexiones de este circuito se muestra en la figura 6.12.

Para tomar la lectura de los 12 bits de salida del convertidor por un solo puerto de 8 bits, es necesario seguir un protocolo, y tomar primero el byte menos significativo, y en seguida, el byte más significativo, del cual solamente se requieren los 4 bits menos significativos. El protocolo mencionado se ilustra en la figura 6.11. Para el funcionamiento de este protocolo se requiere conectar las terminales CS y STATUS del ADC1205 a 1 lógico, y la terminal CD a 0 lógico.

De acuerdo con lo anterior, es claro que el cable que interconecta al convertidor A/D con el sistema multiprocesador, deberá tener 8 líneas para datos y 2 líneas para cumplir con los requerimientos del protocolo. Por conveniencia en este cable se incluyen 2 líneas de alimentación, una para 5V y otra para GND.

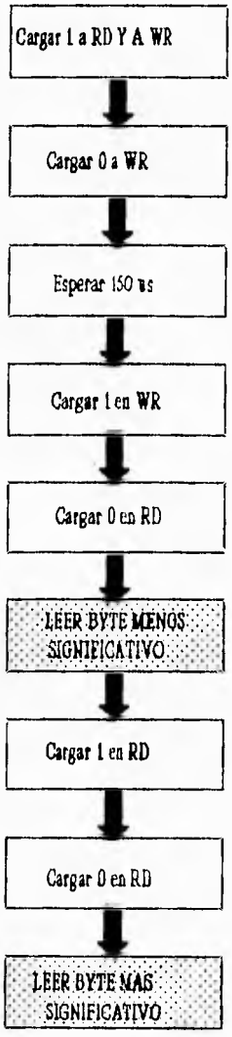


Figura 6.11 Protocolo para lectura de la conversión del ADC1205.

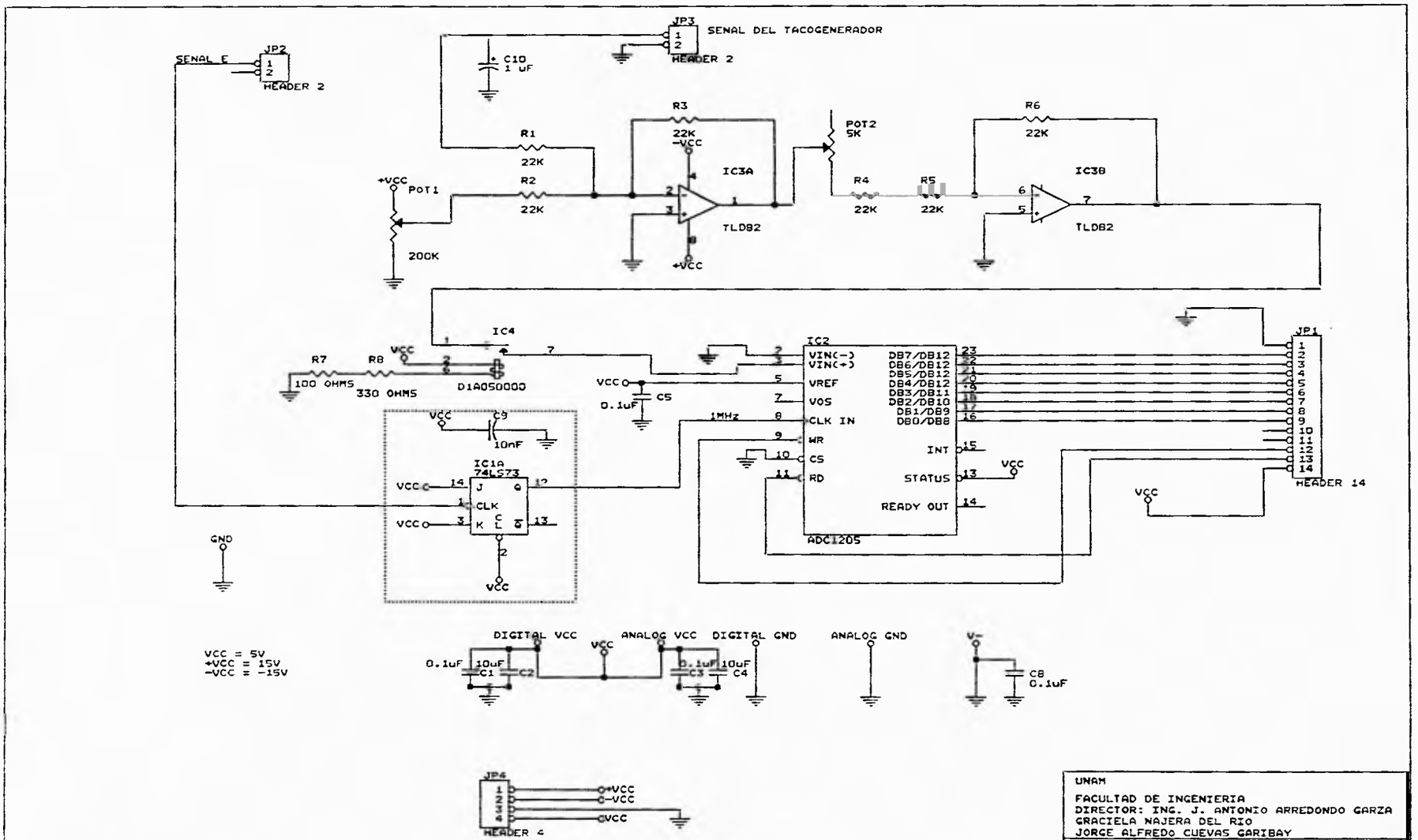


Figura 6.12 Diagrama esquematico del circuito: Interfaz de entrada.

6.5 CIRCUITO DE INTERFAZ DE SALIDA DEL SISTEMA MULTIPROCESADOR

Este circuito incluye los siguientes bloques:

- Etapa 1

  Convertidor Digital a Analógico (Convertidor D/A) de 12 bits
  Circuito de adecuación
  Oscilador controlado por voltaje

- Etapa 2

  Optoacoplador
  Convertidor de Frecuencia a Voltaje (Convertidor F/V)
  Circuito de Potencia

A continuación se presenta una descripción detallada de cada uno de los bloques anteriores. Y en la figura 6.13 se muestra un diagrama a bloques del circuito de interfaz de salida.

El diseño de la interfaz de salida involucra un optoacoplador, debido a que es necesario aislar la etapa de baja potencia de la etapa de alta potencia. La forma que elegimos para hacer este acoplamiento óptico, es generando una señal cuadrada, cuya frecuencia sea proporcional al voltaje de corriente continua, que deseamos obtener en la etapa 2, como voltaje en las terminales del motor y que es el mismo que entrega el convertidor D/A a su salida. Así, con el optoacoplador adecuado, podemos hacer una transmisión óptica del voltaje de salida del convertidor D/A modulado en frecuencia, y en la etapa 2, demodular la señal recibida e interpretarla como un voltaje de corriente continua que debe ser el que se aplique a las terminales del motor.

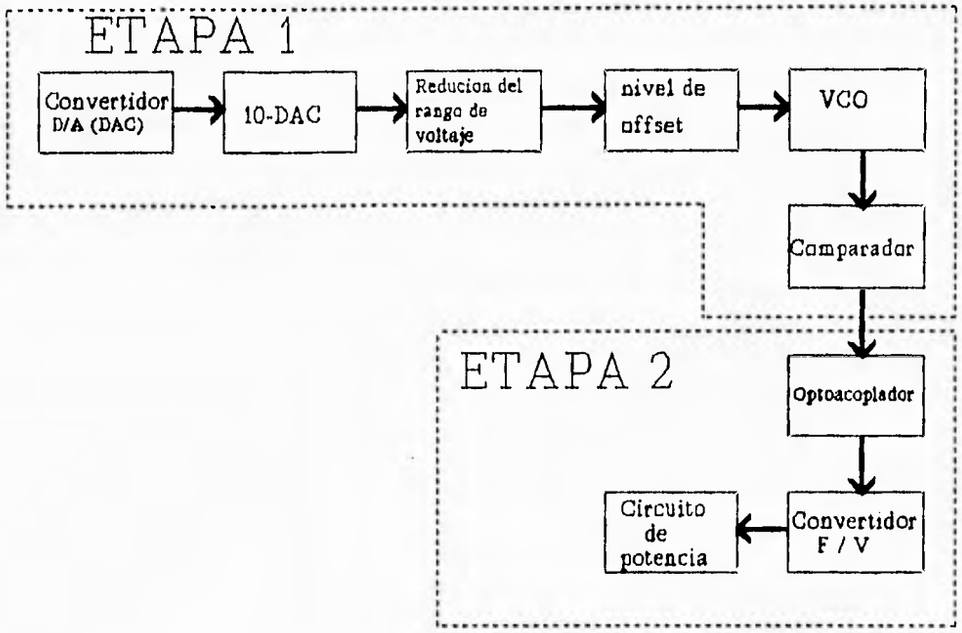


Figura 6.13 Diagrama a bloques del circuito de interfaz de salida.

6.5.1 Etapa 1

Convertidor Digital a Analógico (Convertidor D/A)

Se utilizó el circuito integrado de National Semiconductor DAC1230, el cual es un convertidor digital a analógico de 12 bits. Como se muestra en la figura 6.14, este circuito integrado consta básicamente de un arreglo de resistencias denominado red en escalera R-2R, con un excelente coeficiente de variación a la temperatura (menor a 0.0002%/°C). Esta red en escalera acepta entradas de valores binarios y proporciona un voltaje de salida analógico proporcional al valor digital de entrada.

Por las características de construcción de este circuito integrado, es necesario agregarle un amplificador operacional, que actuando en configuración inversora nos proporcione el voltaje

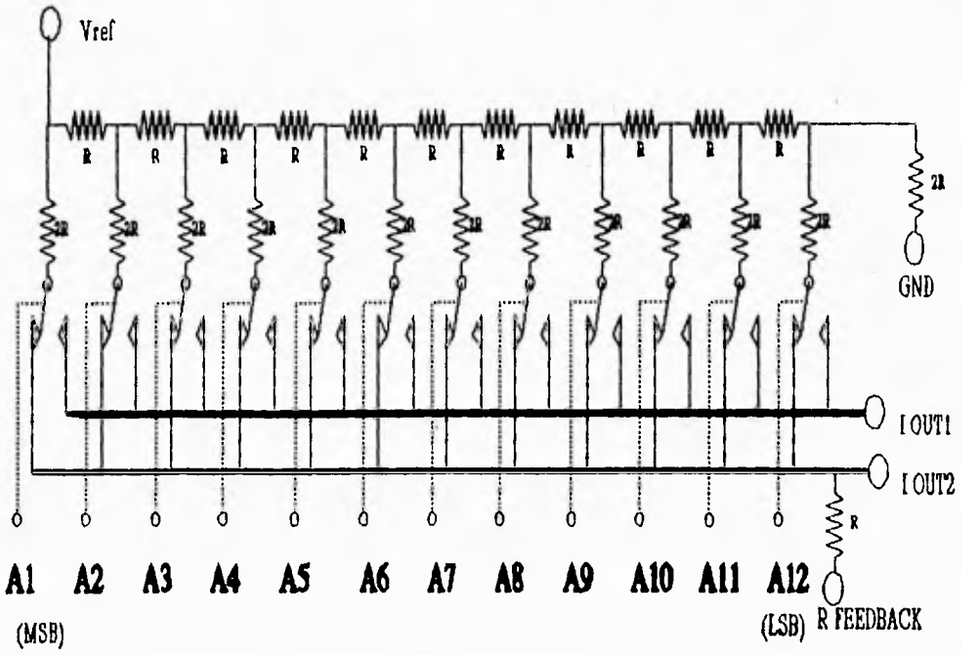


Figura 6.14 Red en escalera

analógico correspondiente al valor digital de entrada, y además permita ajustar el voltaje de offset; el amplificador operacional elegido es el circuito integrado de LM741.

De acuerdo con la figura 6.14, los interruptores de corriente conectados a cada entrada binaria energizan las terminales seleccionadas de la escalera y la salida es una suma ponderada de la corriente de referencia. Mediante un desarrollo matemático que involucre los arreglos R-2R y la resistencia de realimentación mostrada en la figura anterior, se puede demostrar que el voltaje de salida es:

$$V_{out} = - V_{ref} \left(\frac{A1}{2} + \frac{A2}{4} + \frac{A3}{8} + \ldots + \frac{A12}{4096} \right) \qquad (6.9)$$

En la figura 6.15, se muestra un diagrama funcional del circuito integrado DAC1230. En este diagrama se observan los registros internos de almacenamiento temporal de que dispone este

circuito integrado, además de la lógica de control para tales registros. Tomando en cuenta que el DAC1230 dispone únicamente de 8 entradas digitales, es necesario manejar un protocolo para la entrada de los 12 bits y la obtención de la salida analógica, de acuerdo con los diagramas de tiempo que se presentan en las hojas de aplicación. Dicho protocolo se muestra en la figura 6.16. El hecho de disponer de registros internos, hace que este circuito integrado sea muy versátil ya que evita la necesidad de tener una circuitería externa, además, su entrada digital de 8 bits, lo hace compatible con una gran cantidad de microprocesadores y microcontroladores de 8 bits, tales como el MC68HC11F1.

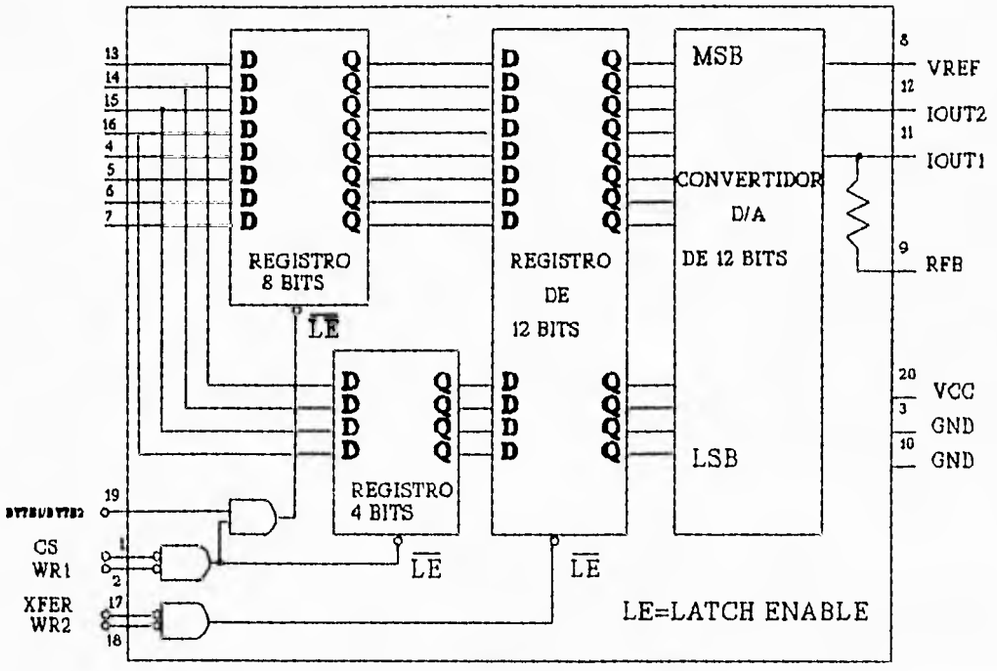


Figura 6.15 Diagrama funcional DAC1230.

El voltaje de referencia elegido para el convertidor D/A fue de -10V, con la finalidad de que el voltaje máximo a la salida de este circuito fuera de 10V. Para establecer este voltaje de referencia se empleó el circuito integrado LM7910 (IC2 en la figura 6.18). El regulador LM7810 (IC1) se usa para establecer un voltaje de alimentación (Vcc) de 10V.

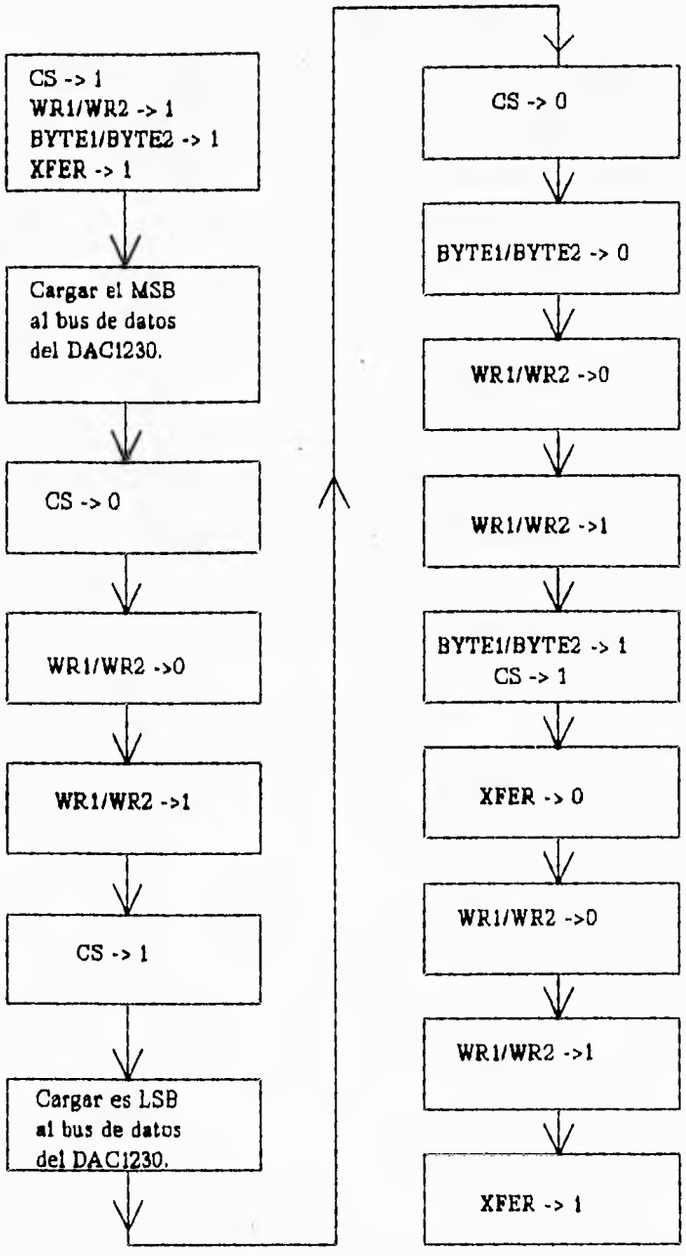


Figura 6.16 Protocolo empleado para cargar datos en el DAC1230.

El diagrama de conexiones correspondiente al convertidor D/A se muestra dentro del esquemático de la figura 6.18, en donde se pueden apreciar los dispositivos electrónicos mencionados y la interconexión de estos. El conector JP2 consta de 14 líneas, 8 de las cuales forman el bus de datos, 4 son líneas de control, y 2 son los niveles de voltaje 5V y GND. Estas señales provienen del sistema multiprocesador, conector J5, y son enviadas por el MCE a través del puerto A y 4 líneas del puerto G (figura 6.3).

Circuito de adecuación

Tomando en cuenta las características de funcionamiento del Oscilador Controlado por Voltaje, así como del Convertidor de Frecuencia a Voltaje que se describirán posteriormente, resultó necesario hacer una adecuación del voltaje analógico proveniente del convertidor D/A, con el fin de hacer compatibles las características de operación de cada uno de los circuitos empleados, mismos que permiten hacer una acoplación óptica de la salida del convertidor D/A hacia la etapa 2 (alta potencia), y con esto evitar cualquier daño al sistema de control. Como se aprecia en la figura 6.13, este circuito de adecuación consta de tres bloques los cuales se implementaron haciendo uso de amplificadores operacionales.

Los dos primeros bloques del diagrama de este circuito de adecuación, ejecutan las siguientes operaciones, el primero ejecuta una resta:

$$V_{compl} = 10 \cdot V_{DAC}$$

donde V_{compl} es el complemento a 10V (máxima salida) del voltaje del convertidor D/A, la obtención de este complemento se requiere para eliminar la incompatibilidad entre circuitos que se plantea aquí: en el oscilador controlado por voltaje LM566, a mayor voltaje de entrada, menor frecuencia de salida, es decir, la pendiente de la frecuencia de salida de este circuito respecto a el voltaje V_{IN} es negativa; a diferencia del convertidor de frecuencia a voltaje, en donde a mayor frecuencia de entrada, mayor voltaje de salida. Suponiendo que no existe ningún rango para el voltaje V_{IN}, si el

voltaje V_{DAC} se aplicara directamente a la entrada V_{IN} del LM566, la salida de este circuito no sería compatible con la entrada requerida por el convertidor de frecuencia a voltaje. Por lo tanto, es necesario obtener el complemento a 10V del voltaje V_{DAC}, de esta forma un voltaje elevado en V_{DAC} daría por resultado un voltaje V_{compl} pequeño, aplicando este voltaje V_{compl} a la entrada del LM566, la salida sería una frecuencia elevada, correspondiente al voltaje elevado del convertidor D/A. Se podría decir entonces que la función de este bloque es invertir la pendiente de la gráfica de la frecuencia de salida del LM566 en función del V_{DAC}.

El segundo bloque tiene por objetivo reducir el rango de variación del voltaje V_{compl}, el cual en vez de variar de 0 a 10V solo debe variar entre 0 y 3V, de acuerdo con las características del oscilador controlado por voltaje.

Estos dos bloques se implementaron haciendo uso de un amplificador operacional en configuración sumador inversor que se observa en la figura 6.18 (IC5A). El voltaje de salida en la terminal 1 de IC5A es:

$$V = V_{compl} * 3/10$$

El tercer bloque en el circuito de adecuación, es el que tiene por finalidad proporcionar un voltaje de offset al voltaje de salida del circuito anteriormente descrito (IC5A), ello, debido a lo requerido por el voltaje de entrada del circuito LM566. Este bloque se implementó haciendo uso de un amplificador operacional en configuración sumador inversor, y se muestra en IC5B en la figura 6.18. Así, en la terminal 7 de IC5B el voltaje es:

$$V_{IN'} = -(9+V_{compl}*3/10)$$

El circuito IC5C de la figura 6.18 tiene la finalidad de invertir el voltaje $V_{IN'}$. Así, en la terminal 8 de IC5C el voltaje es:

$$V_{IN} = 9+V_{compl}*3/10$$

o bien:

$$V_{IN} = 9 + (10-V_{DAC}) * 3/10$$

Los amplificadores operacionales que se utilizaron en la implementación de este circuito de adecuación (IC5A,IC5B,IC5C,IC5D) se tomaron del circuito integrado con cuatro amplificadoras operacionales TL074.

Oscilador Controlado por Voltaje (VCO)

Un oscilador controlado por voltaje (VCO) es un circuito que proporciona una salida oscilatoria (generalmente una onda cuadrada o triangular) cuya frecuencia puede ajustarse sobre un rango controlado por voltaje de corriente continua. El circuito integrado empleado como VCO es el LM566, el cual contiene los circuitos para generar tanto ondas cuadradas, como triangulares, cuya frecuencia se fija por un capacitor (C_1) y una resistencia (R_1) externos y se puede variar por un voltaje aplicado de corriente continua. En este caso se empleará la salida que genera ondas cuadradas.

En la figura 6.17 se puede observar que el LM566 contiene fuentes de corriente para cargar y descargar el capacitor C_1 a una tasa impuesta por la resistencia R_1 y el voltaje de entrada. Un circuito disparador Schmitt se utiliza para conmutar la fuente de corriente entre la carga y la descarga del capacitor, de esta forma, el voltaje triangular, desarrollado a través del capacitor, y la onda cuadrada del disparador Schmitt, se proporcionan como salidas a través de los amplificadores separados.

La frecuencia central de operación del LM566 puede calcularse a partir de la siguiente ecuación:

$$f_0 = \frac{2}{R_1 C_1}\left[\frac{V_A - V_{IN}}{V_A}\right] \qquad (6.10)$$

donde:

R_1 debe estar dentro del rango $2K\Omega \leq R_1 \leq 20K\Omega$.

V_{IN} debe ser mayor o igual que 3/4 de V_A y menor o igual que V_A.

f_0 debe ser menor que 1MHz.

V_A debe estar dentro del rango $10V \leq V_A \leq 24V$.

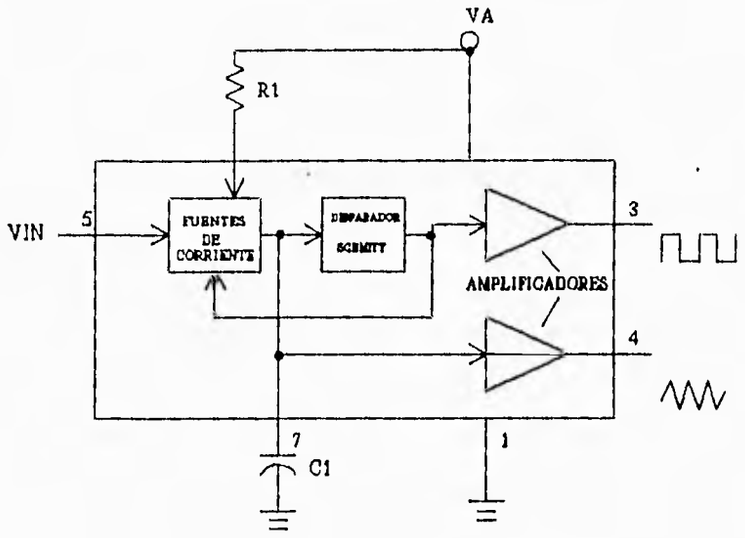


Figura 6.17 Diagrama esquemático del LM566.

Tomando en cuenta el rango al que debe pertenecer V_{IN}, y con el fin de que no hubiese problemas ocasionados por la saturación del amplificador operacional, del que proviene este voltaje de entrada al VCO, se optó por manejar un voltaje V_A de 12V, para lo cual se empleó el circuito regulador LM7812, de manera que:

$$V_A = 12V \quad 9 \leq V_{IN} \leq 12$$

Si se desea que f_0 = 5KHz cuando VDAC = 5V, de la ecuación definida anteriormente para V_{IN}, $V_{IN} = 9+(10-5)*3/10 = 10.5V$, y si $C_1 = 10nF$, de la ecuación 6.10 resulta R_1 = 5KΩ.

Una característica importante del LM566, de acuerdo con la ecuación 6.10, es que tiene

un comportamiento lineal pero con pendiente negativa, lo cual indica que a mayor V_{IN}, la frecuencia de salida disminuye, y viceversa, a menor V_{IN}, la frecuencia de salida aumenta, lo cual se contrapone con el funcionamiento del convertidor de frecuencia a voltaje que se describirá posteriormente, esta es una de las razones por las cuales se requiere el circuito de adecuación anteriormente descrito, sin olvidar su utilidad para proporcionar un voltaje V_{IN} dentro del rango $9 \leq V_{IN} \leq 12$. Entonces, el circuito de adecuación, tiene la función de entregar el valor proporcional del voltaje proveniente del convertidor D/A dentro del rango establecido para la correcta operación del LM566 así como invertir la pendiente de la respuesta de este circuito VCO para su posterior optoacoplamiento y compatibilidad con la etapa 2 del circuito de interfaz de salida. En el diagrama esquemático de la figura 6.18 se puede observar la conexión del VCO que ha sido explicada.

Una vez teniendo una señal cuya frecuencia es directamente proporcional al voltaje del convertidor D/A, el siguiente bloque en el diagrama esquemático de la figura 6.13, es un circuito comparador de nivel. Este tiene por objetivo, reducir a dos, los múltiples niveles de voltaje de la señal de salida, del circuito VCO LM566, ya que a la salida de éste, el nivel bajo varía entre 0.5 y 3V y el nivel alto entre 8 y 10V dependiendo de la frecuencia de salida. Con el comparador de nivel que se plantea en este bloque, se pretende que el nivel bajo esté en 0V y el alto en 15V. El circuito comparador empleado es el LM311, el cual tiene un tiempo de respuesta de 200 ns. A la salida de este circuito se obtuvo una señal con la misma frecuencia que la señal de entrada pero únicamente tendrá dos niveles de voltaje, aproximadamente 0.4V y 15V.

La señal de salida del comparador LM311 es transmitida hacia la tarjeta del circuito al que hemos denominado etapa 2, haciendo uso de un par trenzado de cobre como medio de transmisión, el cual se conecta, por una parte al conector JP6 de la etapa 1 y por otra, al conector JP7 de la etapa 2, del circuito de interfaz de salida.

En la figura 6.18, el conector JP1 se emplea para recibir la alimentación de la fuente de ±15V y 5V, mientras que el conector JP8 se usa para llevar estos voltajes de alimentación al circuito de interfaz de entrada.

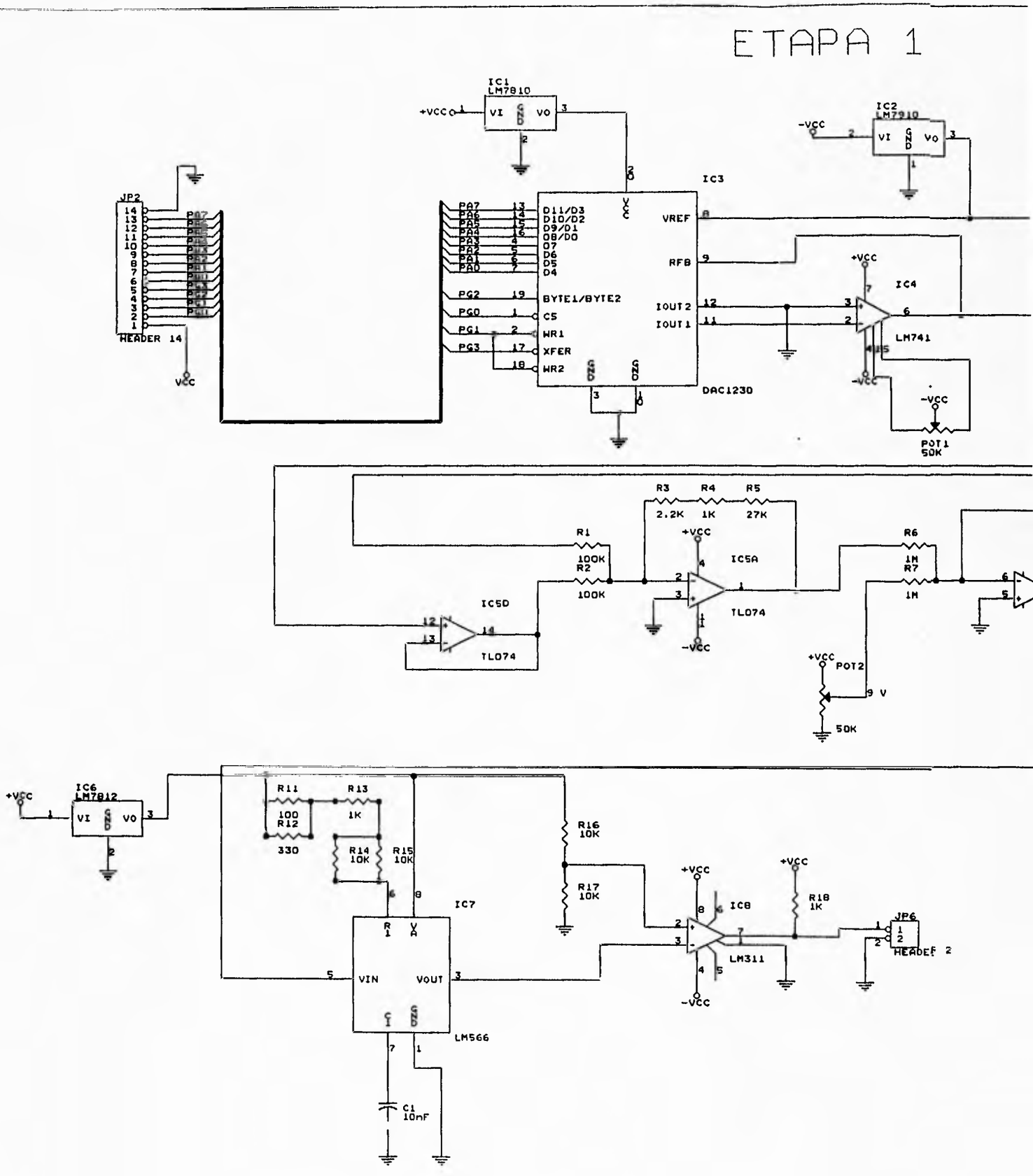
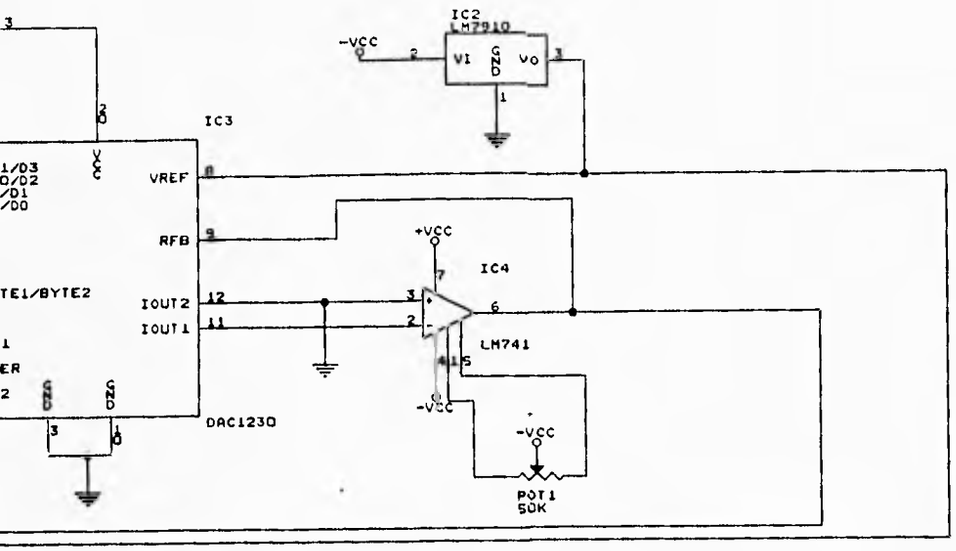
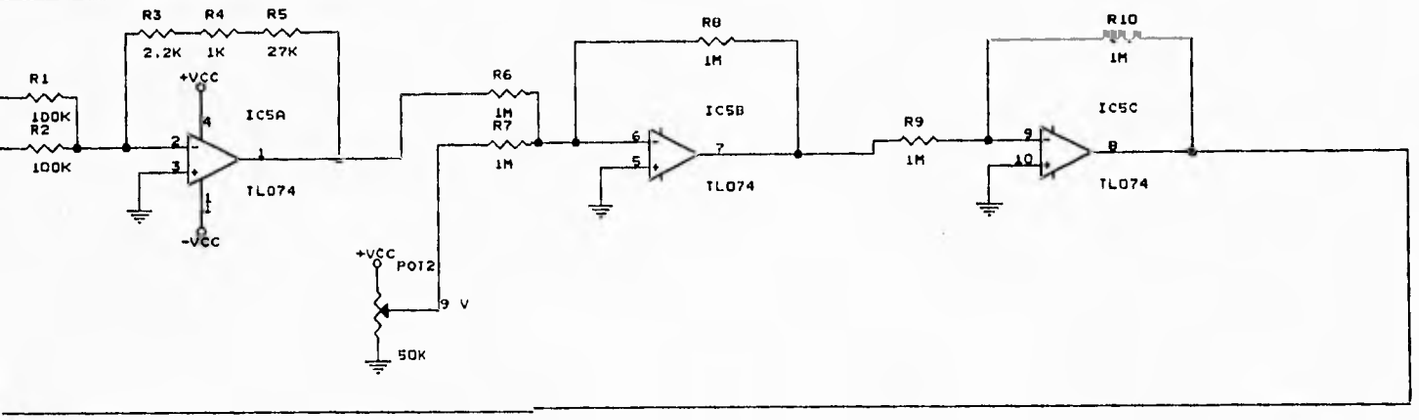
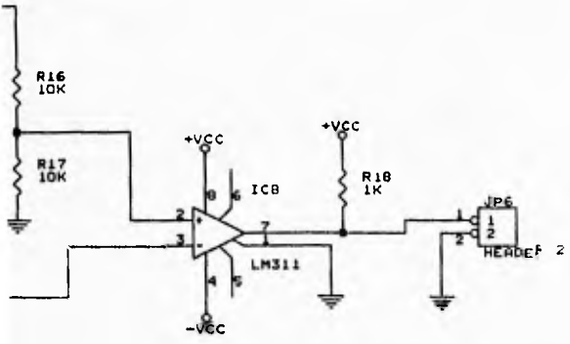


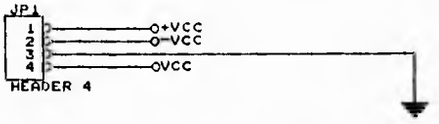
Figura 6.18 Diagrama esquematico del circuito de interfaz de salida (etapa

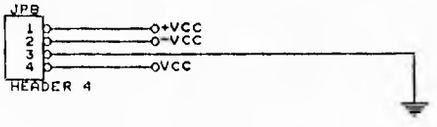
ETAPA 1











ito de interfaz de salida (etapa 1)

UNAM	
FACULTAD DE INGENIERIA	
DIRECTOR: ING. J. ANTONIO ARREDONDO GARZA	
JORGE ALFREDO CUEVAS GARIBAY	
GRACIELA NAJERA DEL RIO	
Title	CONTROL PID PARA MOTOR CD
Size: C	Document Number: INTERFAZ DE SALIDA (ETAPA 1)
	REV

6.5.2 Etapa 2

En el diagrama a bloques del circuito de interfaz de salida (figura 6.13), la etapa 2 está conformada por los últimos tres bloques, los cuales se describen a continuación.

Optoacoplador

La señal modulada en frecuencia proveniente del conector JP7, lleva consigo la información del voltaje de salida del convertidor D/A. Por lo anteriormente expuesto, esta señal será optoacoplada, para lo cual se empleará el circuito integrado de Motorola 4N25, dado su bajo costo y su eficiente operación. Este dispositivo está formado por un diodo emisor de luz infrarroja y un fototransistor detector de pulsos. El diodo emisor de luz transmite al fototransistor los pulsos que recibe en sus terminales, provenientes del comparador LM311. Este fototransistor se encuentra polarizado de forma tal que en su emisor se tiene la señal modulada en frecuencia proveniente de la etapa 1, pero con la ventaja de que el dispositivo 4N25 permitió aislar la etapa de baja potencia (etapa 1) de la de alta potencia (etapa 2).

En la figura 6.20 se puede observar la forma en la que se conectó el optoacoplador 4N25 (IC11), así como la interconexión de éste con el resto del circuito de la interfaz de salida (etapa 2).

Convertidor de Frecuencia a Voltaje (Convertidor F/V)

El convertidor de frecuencia a voltaje utilizado fue el LM2907 de Motorola, cuyo diagrama esquemático se muestra en la figura 6.19. En este diagrama se puede apreciar que el LM2907 consta esencialmente de dos partes, la primera de ellas esta formada por un comparador y un circuito de "charge pump", al cual se le conecta en forma externa un capacitor C_1, y una resistencia R_1 en paralelo con un capacitor C_2. La segunda parte consta de un amplificador operacional, que

para la presente apliceción, se conectará en configuración seguidora, y la salida de éste se encuentra conectada a un transistor seguidor de voltaje. En el emisor de este transistor se conecta una resistencia externa de 10KΩ, y es de este punto de donde se obtiene el voltaje proporcionel a la frecuencia de la señal de entrada.

El voltaje de salida de este circuito está dado por la siguiente expresión:

$$V_0 = V_{cc} \times F_{in} \times C_1 \times R_1 \times K \qquad (6.11)$$

donde:

K es la constante de ganancia y es típicamente iguel a 1

C_1 se recomienda que sea mayor que 500 pf para que el velor R_1 no sea muy alto, puesto que esto ocasionaría problemes de impedancia en el terminal 3 y traería como consecuencia degradamiento en la linealidad del circuito.

En la figura 6.20 se observa la conexión del LM2907 (IC9) con el resto de la etapa 2. El emisor del fototransistor del optoacoplador 4N25, está conectado directamente con la terminal de entrada al convertidor F/V, y dada la operación del mismo, aunado al conjunto de resistencias y capacitores que están conectados a él, a la salida de este convertidor tendremos un voltaje de corriente continua, igual al voltaje de salida del convertidor D/A. Se consideró conveniente colocar una resistencia variable de 50K (POT3) en serie con la resistencia de 100K (R22), con la finalidad de dar opciones de calibración del sistema en caso necesario.

Este dispositivo debe operar de la siguiente forma: ante una señal de entrada cuya frecuencia sea de 10KHz, el voltaje de salida del convertidor F/V debe ser de 10V, y ante una frecuencia de 1KHz, la señal de salida debe ser de 1V. A menor frecuencia de la señal de entrada la linealidad del convertidor F/V no está garantizada. Esta es otra de las razones por las cuales no se garantiza un control preciso a velocidades pequeñas.

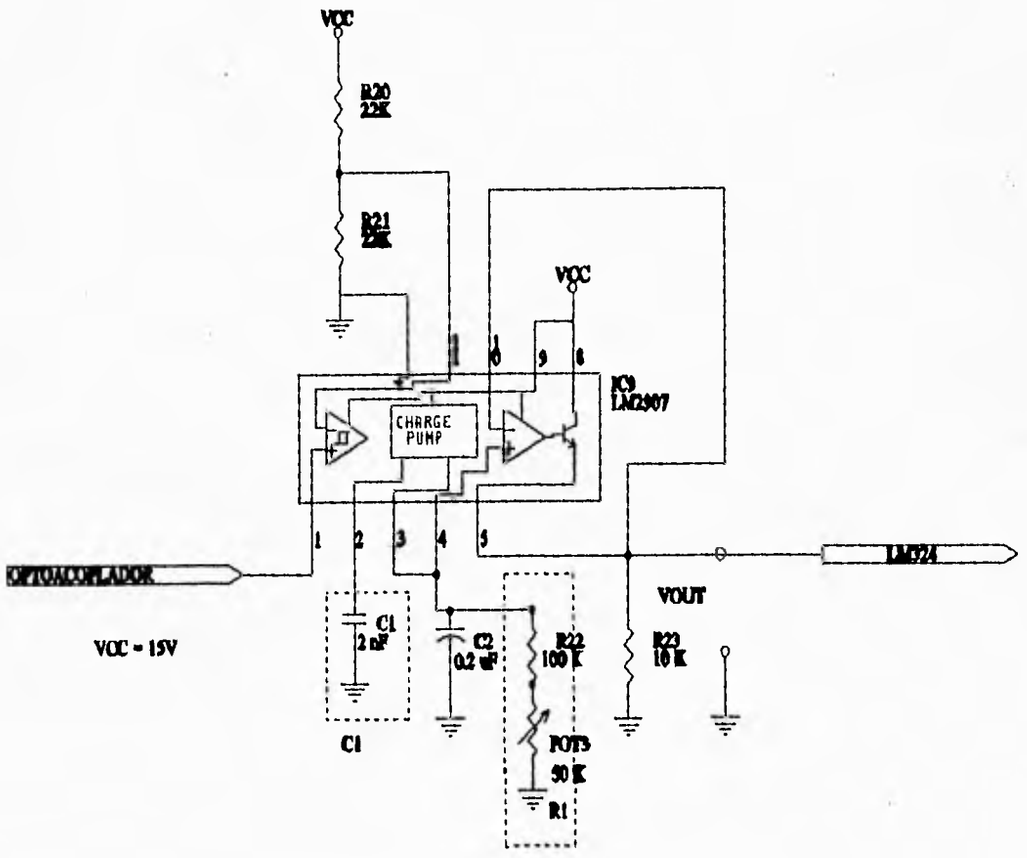


Figura 6.19 Convertidor F/V y conexiones.

Circuito de potencia

En la terminal 5 del convertidor F/V se tiene ya el nivel de voltaje que va a ser aplicado a las terminales del motor, y dadas las características del motor (datos de placa), solo es necesario amplificar la corriente a ser conducida hacia el motor. Para lograr lo anterior, se decidió emplear un amplificador operacional en configuración seguidora, cuya terminal de salida se conectó a la base de un arreglo de transistores, conocido como Darlington, la conexión se muestra en la figura 6.20. El arreglo de transistores Darlington que adquirimos viene integrado en un mismo empaquetado. El

colector de este "transistor Darlington", lo conectamos al voltaje de alimentación de 18V, el emisor hacia el motor, y la base hacia la salida del amplificador operacional antes mencionado. La retroalimentación de la típica configuración seguidora, conecta la entrada inversora del amplificador operacional, con el emisor del transistor Darlington. Esta configuración permite tener una alta impedancia de entrada vista desde la salida del convertidor F/V, lo que es conveniente para la operación apropiada de este convertidor, además esta configuración es necesaria, para garantizar que el voltaje en el emisor del transistor Darlington sea el mismo que el que se tiene en la terminal de salida del convertidor F/V. El amplificador operacional empleado en esta etapa es el LM324. Se escogió este amplificador operacional debido a que es posible polarizarlo con una sola fuente, en este caso de 18V. Como ya se mencionó anteriormente, el emisor del transistor Darlington se conectó al cable de polarización positiva del Motor de CD. Como se aprecia en la figura 6.20, el cable de polarización negativa del motor se conectó a tierra.

El transistor Darlington que se utilizó es el TIP142, el cual está integrado en un encapsulado TO-3JP.

Para calcular el disipador necesario para el TIP142 se ocuparon los siguientes datos tomados de las hojas de aplicación de este circuito:

Resistencia térmica entre la unión(j) y el encapsulado(c): $\theta_{jc} = 0.65$ °C/W
Resistencia térmica entre el encapsulado(c) y el disipador(s): $\theta_{cs} = 0.95$ °C/W
Temperatura máxima en la unión: $T_j = 150$ °C

Por otro lado, se sabe que la potencia que disipa este circuito es P = VI; donde V es el voltaje VCE máximo que se presenta en el transistor Darlington y es igual a 18V, mientras que I = 3A, de forma que P = 54W.

Con estas consideraciones, resolviendo el circuito térmico para una temperatura de unión máxima de 150°C, la resistencia térmica entre el disipador(s) y el aire (A) es:

$$\theta_{SA}=(150-25-54(0.95+0.65))/54)$$

$$\theta_{SA}= 0.7148\ ^{\circ}C/W$$

Según la curva θ_{SA} Vs. longitud del disipador, que proporcionan los fabricantes de disipadores EG&G, basta con 12 cm del perfil del disipador modelo 1542.

Ahora, tomando en cuenta las características básicas de funcionamiento de todo inductor, se sabe que:

$$V_L = L \frac{di}{dt} \qquad (6.12)$$

de manera que al momento de aplicar cambios de corriente bruscos, es necesario proteger al transistor. La ecuación anterior muestra como una disminución brusca de corriente que se le aplique a un inductor, trae consigo un aumento negativo considerable del voltaje en sus terminales. La cantidad de voltaje negativo en las terminales del inductor depende tanto de la inductancia como de la pendiente con la que se quitó corriente al inductor.

En el funcionamiento del circuito del diagrama 6.20, es posible que de forma repentina se corte la corriente al motor, y las consecuencias pueden ser desastrosas. El diodo D1 sirve para proteger tanto al transistor como al motor. En condiciones de altas di/dt, este diodo no permite que el voltaje del emisor se eleve mas de 0.7V con respecto a tierra.

Con este circuito de potencia, se da por terminada la explicación del circuito de interfaz de salida, el cual como dijimos en un principio, tiene por objeto llevar a las terminales del motor, un voltaje proporcional al valor digital de salida del controlador PID (sistema multiprocesador). Sin embargo no debemos olvidar que aunque este diseño es óptimo y funcional, existen, como en todos las áreas, y especialmente en la Ingeniería, muchas otras opciones para cumplir con un objetivo determinado, por ello, a continuación explicaremos otra opción posible, y también analizada, para construir un circuito de interfaz de salida. Además se mencionarán los motivos por los cuales se prefirió construir el circuito descrito anteriormente.

Otras opciones

Un *chopper* se puede utilizar para controlar el voltaje de armadura de un motor. Una de las formas mediante las cuales se logra esto, es haciendo que un tiristor o un TMOS, trabaje en modo de modulación por ancho de pulso. Con el tiristor trabajando en modo modulación por ancho de pulso, se debe implementar un circuito que genere una señal con un periodo constante T, de manera que, dependiendo de un voltaje de control (Vc), el cual puede provenir del convertidor D/A, se haga variar el ciclo de trabajo de la señal de periodo constante T. Esta señal modulada por ancho de pulso, es la que se encarga de activar al interruptor de corriente escogido (TMOS, tiristor, etc.). De esta forma, se controla el voltaje promedio que llaga al motor.

En la figura 6.21 se aprecia como se puede controlar el voltaje promedio aplicado al motor, al variar el tiempo alto T_A de la señal que activa y desactiva al TMOS o tiristor elegido. El voltaje promedio esta dado por:

$$V_o = \frac{T_A}{T} V_{max} \qquad (6.13)$$

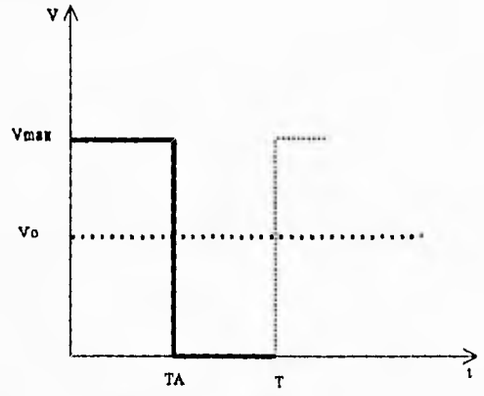


Figura 6.21 Voltaje aplicado a las terminales del motor utilizando un particionador.

Recordando, la ecuación de velocidad para un motor con excitación independiente, expuesta en el capítulo II es:

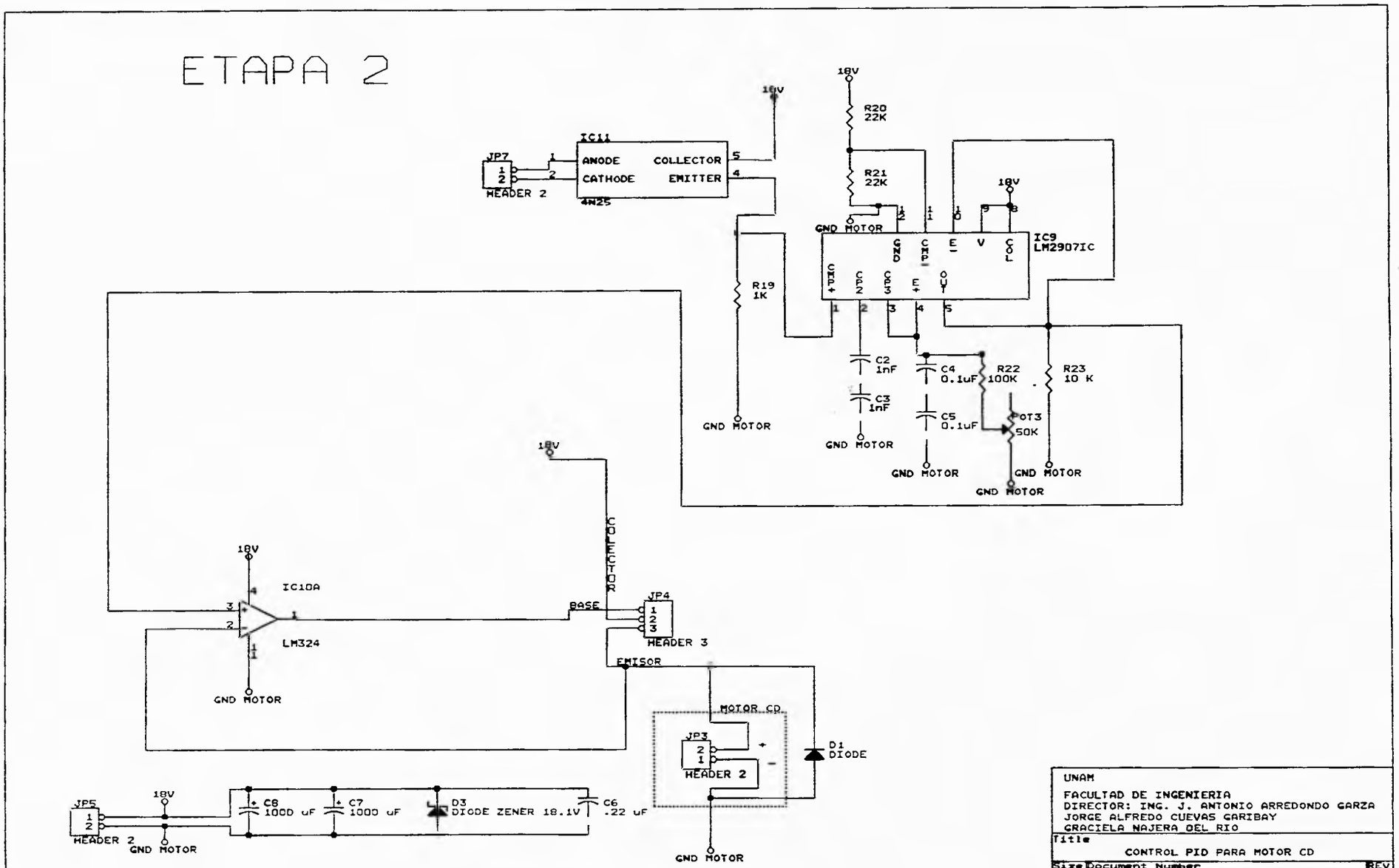


Figura 6.20 Diagrama esquematico del circuito de interfaz de salida (etapa 2).

$$n = KV_{to} + \alpha \qquad (6.14)$$

donde: K es la pendiente de la curva n contra V_{t0}

α es una constante

V_{t0} es el voltaje promedio en las terminales.

Por otra parte, el voltaje en las terminales del motor (Vt), para este caso, sería:

$$V_t = \begin{cases} V_{max} & para \quad t < T_A \\ V_a & para \quad T_A < t < T \end{cases} \qquad (6.15)$$

Donde V_a es la fuerza contraelectromotriz.

De tal forma que, el voltaje promedio en las terminales V_{t0} es:

$$V_{to} = \frac{1}{T}\int_0^{T_A} V_{max}\, dt + \frac{1}{T}\int_{T_A}^{T} V_a\, dt \qquad (6.16)$$

resolviendo las integrales:

$$V_{to} = \frac{T_A}{T} V_{max} + \frac{1}{T} V_a \,[T - T_A] \qquad (6.17)$$

Sustituyendo la ecuación 6.17 en la ecuación 6.14, resulta:

$$n = K\left[\frac{T_A}{T} V_{max}\right] + K\left(\frac{T-TA}{T} V_a\right) + \alpha \qquad (6.18)$$

Recordando que Va es función de la velocidad del motor:

$$n = K\left[\frac{T_A}{T} V_{max}\right] + K\left(\frac{T-TA}{T} V_a(n)\right) + \alpha \qquad (6.19)$$

De la ecuación 6.19, se deduce que la velocidad no guarda una relación lineal con respecto

a T_A por el efecto de la fuerza contraelectromotriz.

Esto es, un cambio en T_A, provocará un cambio en el V_{t0} aplicado a las terminales del motor, y ello traerá consigo un cambio en la velocidad del motor, lo cual, implica un cambio en la fuerza contraelectromotriz que provocará una variación no lineal de la velocidad con respecto a T_A.

Tomando en cuenta que el ciclo de trabajo está determinado en forma lineal por el voltaje de control Vc, se deduce que esta opción no es la adecuada, ya que la respuesta de velocidad con respecto al voltaje de control, sería como lo muestra la gráfica de la figura 6.22:

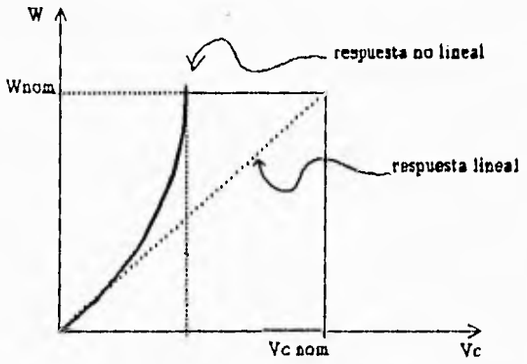


Figura 6.22 Respuesta no lineal de la velocidad del motor ante el voltaje de control.

Para un buen desempeño del controlador PID en cualquier punto de operación, sería necesario que la curva de respuesta de la velocidad del motor, ante un voltaje de control, fuera lineal, y como se aprecia en la figura 6.22, un motor controlado de esta forma, por un chopper, no responde en forma lineal.

La explicación anterior, justifica la razón por la que no se empleó esta opción en el circuito de interfaz de salida del sistema de control construído.

6.6 FUENTES DE ALIMENTACION

Para alimentar a los circuitos impresos, se diseñaron y construyeron 2 fuentes de corriente continua.

6.6.1 Fuente múltiple

Esta fuente entrega a su salida voltajes regulados de ±15V y 5V. El diseño de esta fuente se realizó tomando en cuenta que el consumo de corriente por parte de las tarjetas a las que alimenta (sistema multiprocesador, interfaz de entrada y etapa 1 de la interfaz de salida) no es mayor que 500 mA.

Según se muestra en el diagrama esquemático de la figura 6.23, esta fuente esta construída en base a un transformador 110V:30V, un puente rectificador de onda completa, 2 capacitores C_1 y C_2, cuya función es disminuir el voltaje rizo, tres reguladores integrados (IC1, IC2, IC3) que por sus características y por los capacitores externos que los acompañan, permiten obtener un voltaje rizo pico a pico casi nulo en las salidas reguladas. Con el fin de proteger a los reguladores, ante transitorios causados por cargas inductivas o capacitivas, se colocaron los diodos D1 a D6.

6.6.2 Fuente de 18V

La otra fuente que forma parte del sistema es una fuente regulada de 18V, 3A, la cual alimenta a la etapa 2 del circuito de interfaz de salida. El diseño de esta fuente involucra a un transformador 110V:15V, un rectificador onda completa (IC1), un capacitor de 4700 μF, C_1, el cual disminuye en forma significativa el voltaje rizo a la salida del rectificador. También se utilizó un regulador integrado (IC2), el cual en conjunto con los capacitores C_2, C_3 y C_4 proporcionan a la

salida del regulador un voltaje continuo con un rizo despreciable. Para el funcionamiento adecuado de este regulador (LM350), es necesario calcular los valores de R_1 y R_2, y tomando en cuenta que el LM350 mantiene exactamente 1.2 V entre sus terminales de salida y ajuste, un resistor R_1 de 240Ω permitirá un flujo de corriente especificado de 5 mA. El voltaje de salida del regulador se establece por el voltaje del resistor R_2 mas la caída de voltaje de 1.2V a través de R_1. Por lo tanto el valor de 18V de salida se establece por el ajuste de R_2 a un valor determinado por:

$$V_0 = 1.2V + (5mA)(R_2) \qquad (6.20)$$

entonces R_2 es igual a 3360.

Para calcular el tamaño y forma del disipador para este regulador LM350 (IC2) se ocuparon los siguientes datos tomados de las hojas de aplicación de este circuito:

Resistencia térmica entre la unión (j) y el encapsulado (c): θ_{jc}=2.3 °C/W

Resistencia térmica entre el encapsulado (c) y el disipador (s): θ_{CS}=5 °C/W

Temperatura máxima en la unión: T_j=150 °C

Por otro lado, se sabe que la potencia que disipa este circuito es P = VI; donde V es el voltaje de CD despues del capacitor C_1 (aproximadamente 21.3V), menos el voltaje que entrega este regulador (18V), por lo que V = 3.3V. I es la corriente máxima que entrega el regulador (3A), de forma que P = (3.3V)(3A) = 9.9W.

Con estas consideraciones, resolviendo el circuito térmico para una temperatura de unión máxima de 150°C, la resistencia térmica entre el disipador (s) y el aire (A) es:

$$\theta_{SA} = (150\text{-}25\text{-}9.9(2.3+5))/9.9) = 5.32\ °C/W$$

Según la curva θ_{SA} contra longitud del disipador, que proporciona los fabricantes de disipadores DESA, basta con 5 cm del perfil del disipador modelo 3526.

En la figura 6.24 se muestra el diagrama esquemático de esta fuente.

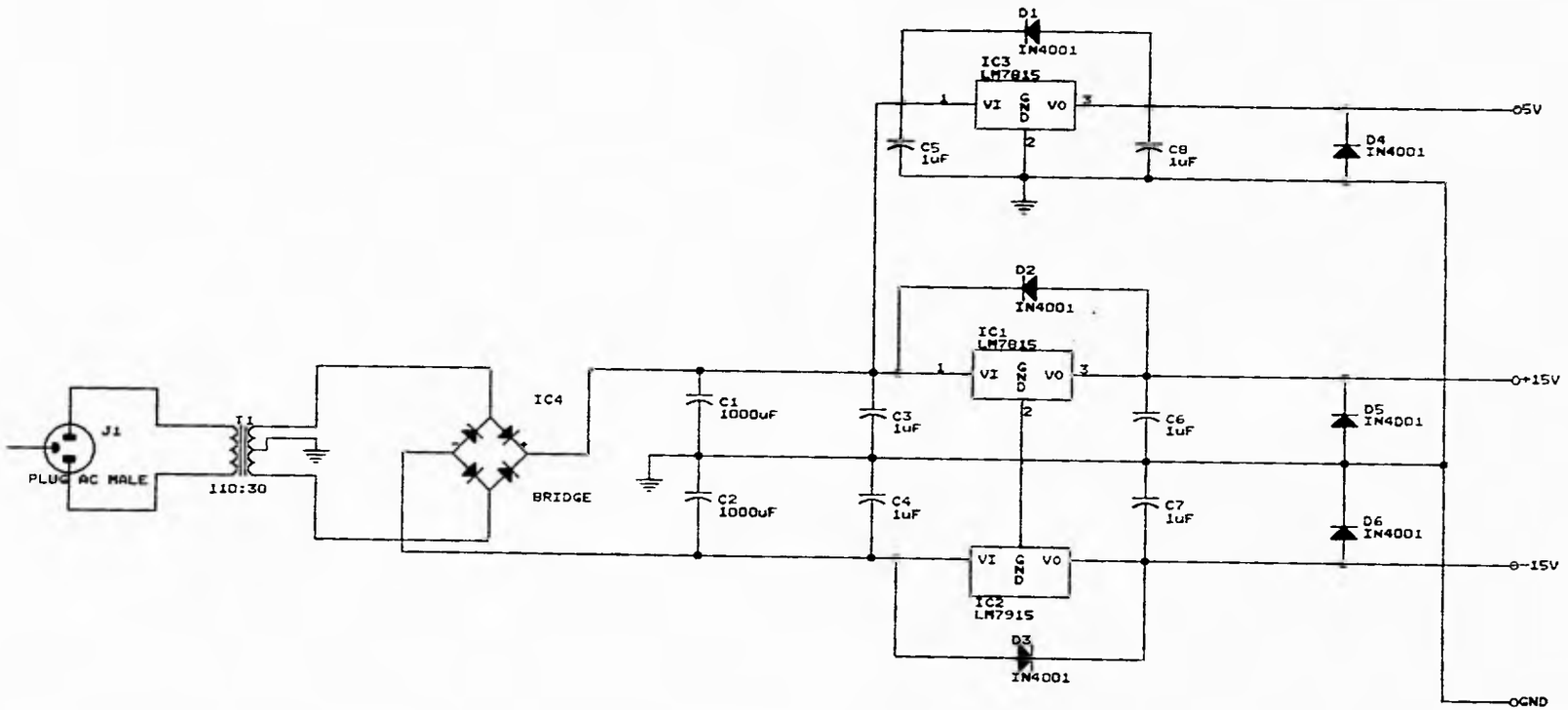


Figura 6.23 Diagrama esquematico de la fuente multiple.

UNAM
FACULTAD DE INGENIERIA
DIRECTOR: ING. J. ANTONIO ARREDONDO GARZA
JORGE ALFREDO CUEVAS GARIBAY
GRACIELA NAJERA DEL RIO

Title
CONTROL PID PARA MOTOR CD

Document Number
FUENTE MULTIPLE
REV

Date: October 30, 1995 Sheet of

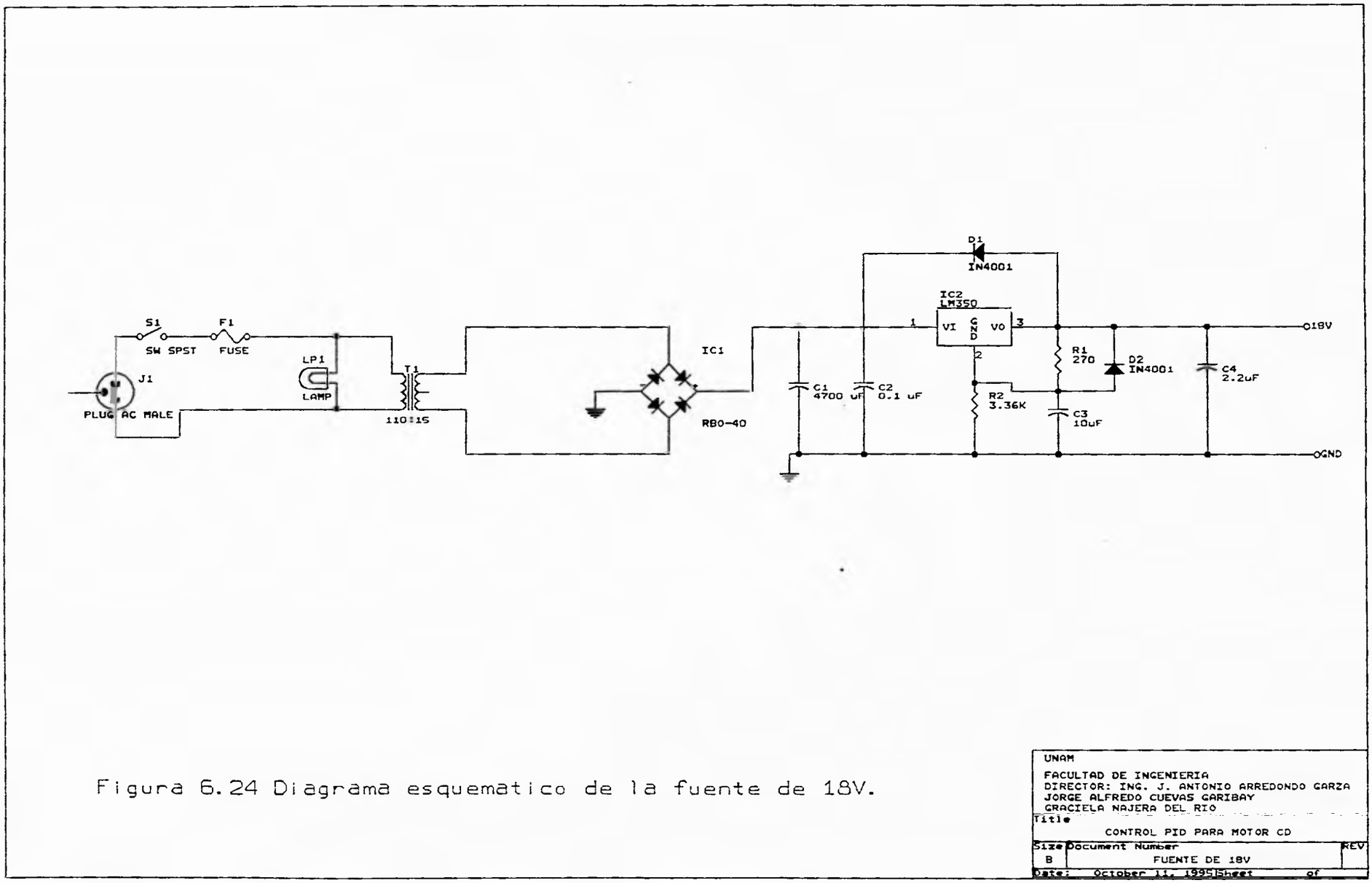


Figura 6.24 Diagrama esquematico de la fuente de 18V.

6.7 ELABORACION DE LOS CIRCUITOS IMPRESOS

Para el diseño y construcción de los circuitos impresos se utilizaron los paquetes de cómputo, auxiliares para la elaboración de circuitos impresos, OrCAD y TANGO. Con el paquete OrCAD se eleboraron los diagramas de conexiones eléctricas, y el listado de conexiones en los circuitos. Posteriormente, en el paquete TANGO, se asignó de forma gráfica la colocación y las dimensiones físicas de todos los dispositivos electrónicos, sobre el espacio asignado para el circuito impreso en cuestión. En todos los circuitos impresos realizados, se utilizaron bases para colocar los circuitos integrados y conectores para facilitar la conexión entre tarjetas.

6.7.1 Circuito impreso: sistema multiprocesador

Este circuito impreso se realizó a dos caras. En una de estas, las pistas son horizontales y en la otra, son verticales. Se utilizó la opción de ruteo automático de TANGO, y posteriormente tuvimos que completar en forma manual las pistas que el paquete no pudo realizar. El diagrama de distribución de componentes se muestra en la figura 6.25, la cara superior del circuito impreso en la figura 6.26 y la cara inferior en la 6.27. En todas las figuras la escala es 1:1.

6.7.2 Circuito impreso: interfaz (interfaz de entrada y etapa 1 de la interfaz de salida)

Este circuito impreso, esta dividido en dos partes, ambas sobre la misma tarjeta. Una parte contiene al circuito de interfaz de entrada y la otra, a la etapa 1 del circuito de interfaz de salida. Al igual que el circuito impreso: sistema multiprocesador, este también fue realizado a dos caras. La cara superior se muestra a escala 1:1 en la figura 6.29 y la cara inferior, también a escala 1:1 se presenta en la figura 6.30. El diagrama de distribución de componentes se encuentra en la figura 6.28.

6.7.3 Circuito impreso: etapa 2 de la interfaz de salida

Por la sencillez de este circuito, este fue realizado en una tarjeta de menor tamaño. El diagrama de distribución de componentes y ambas caras del circuito impreso, se muestran en las figuras 6.31, 6.32 y 6.33 respectivamente.

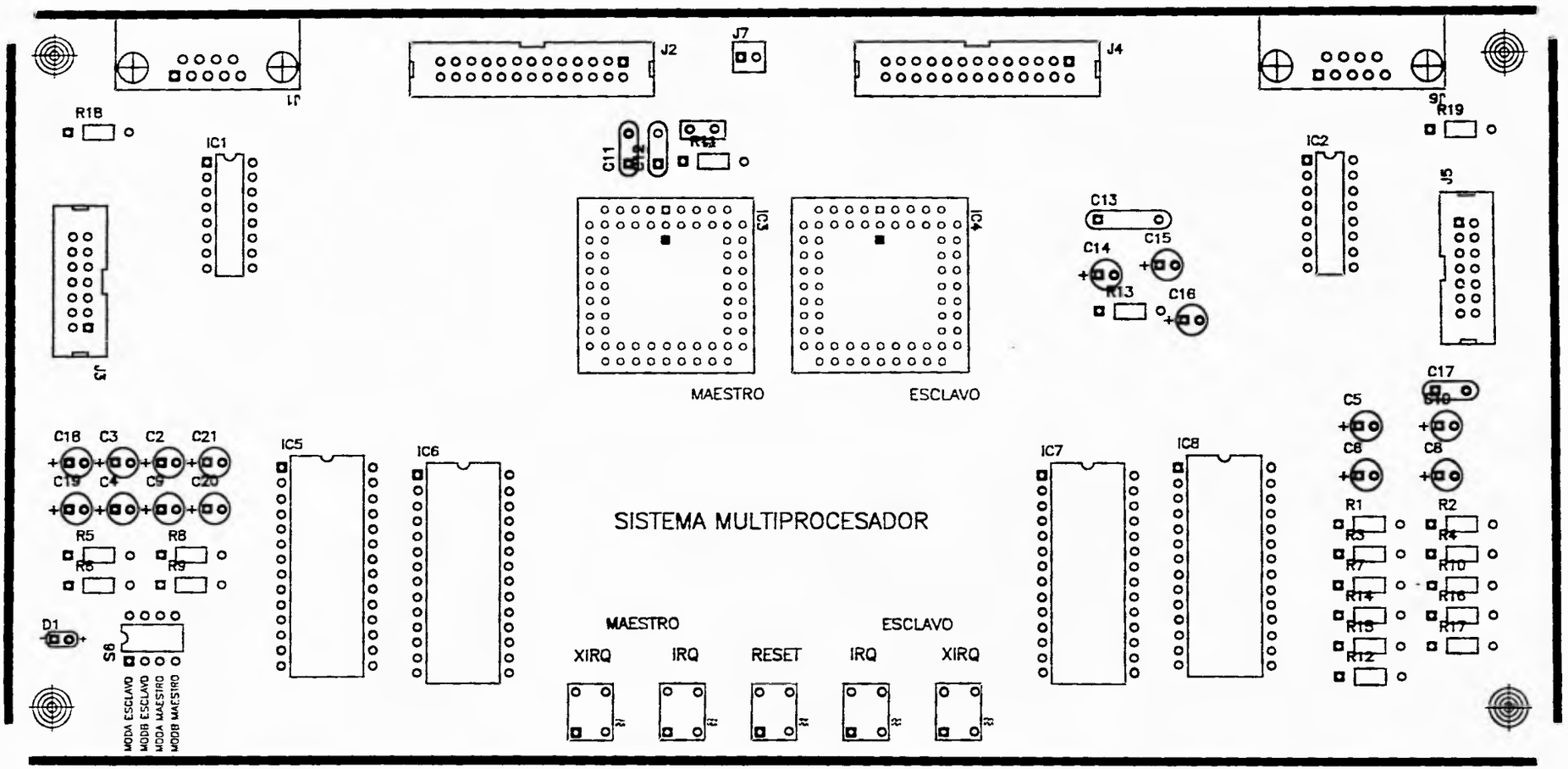


Figura 6.25 Diagrama de distribucion de componentes

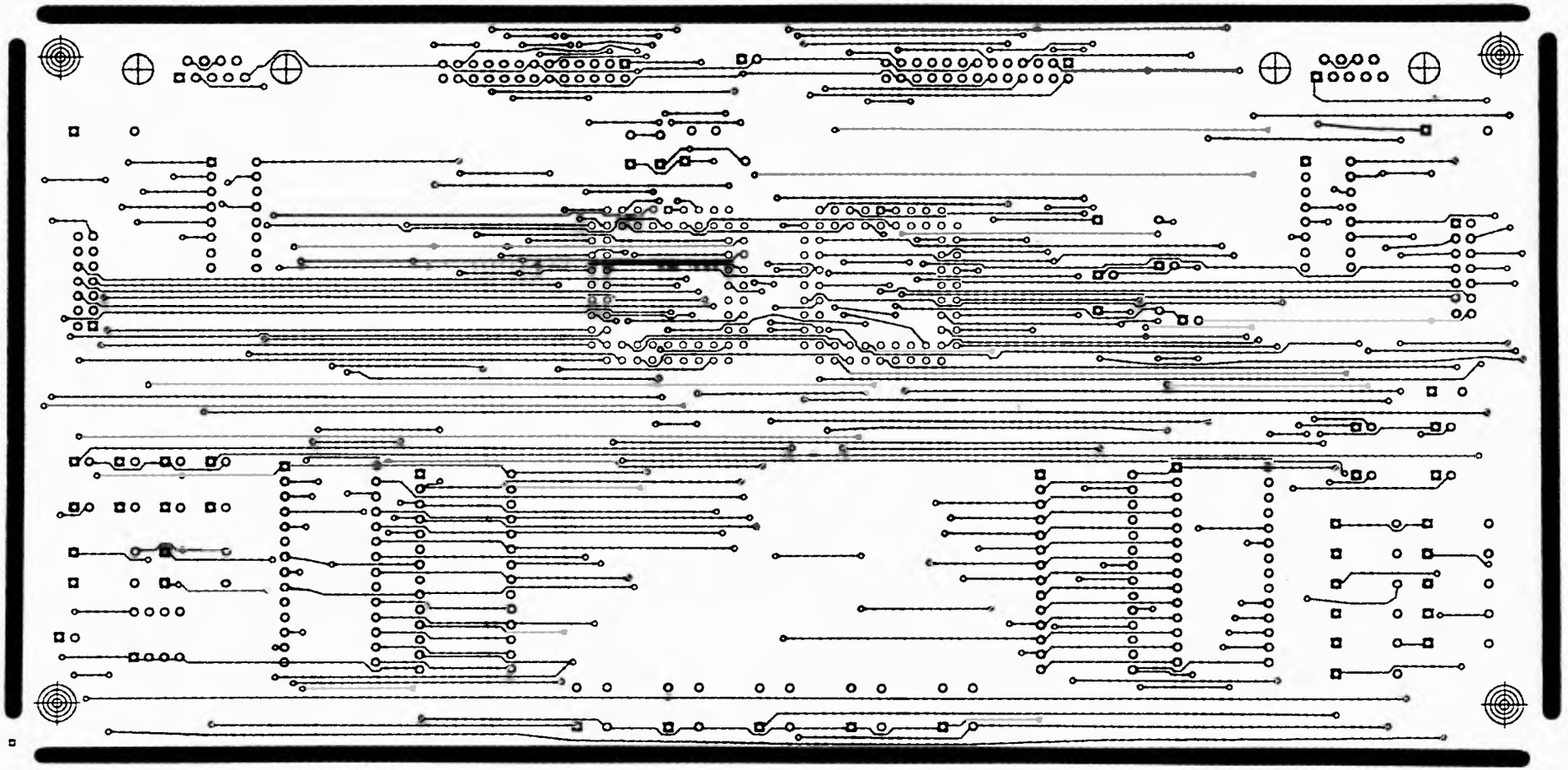


Figura 6.26 Circuito impreso, cara superior.

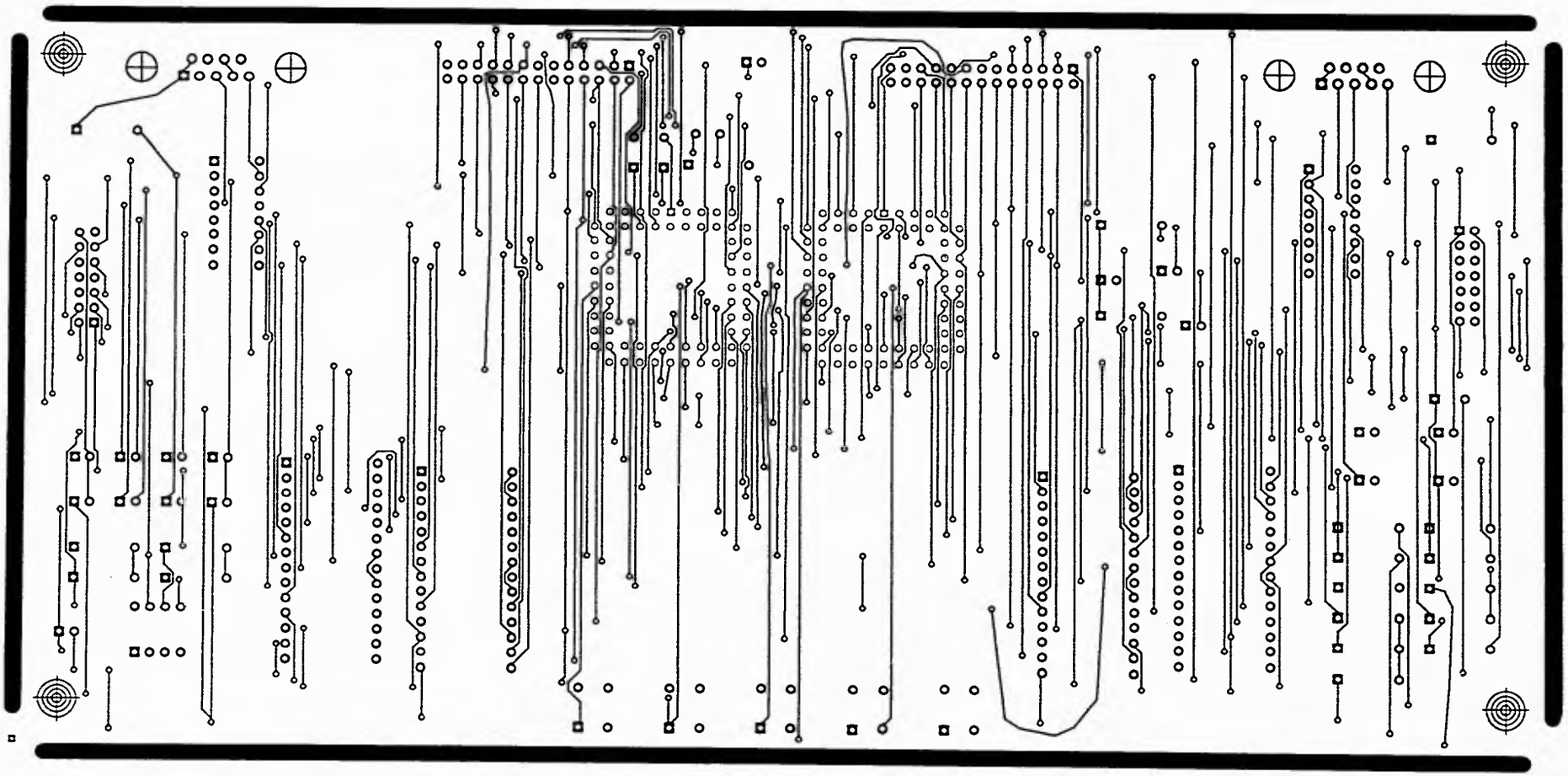


Figura 6.27 Circuito impreso, cara inferior.

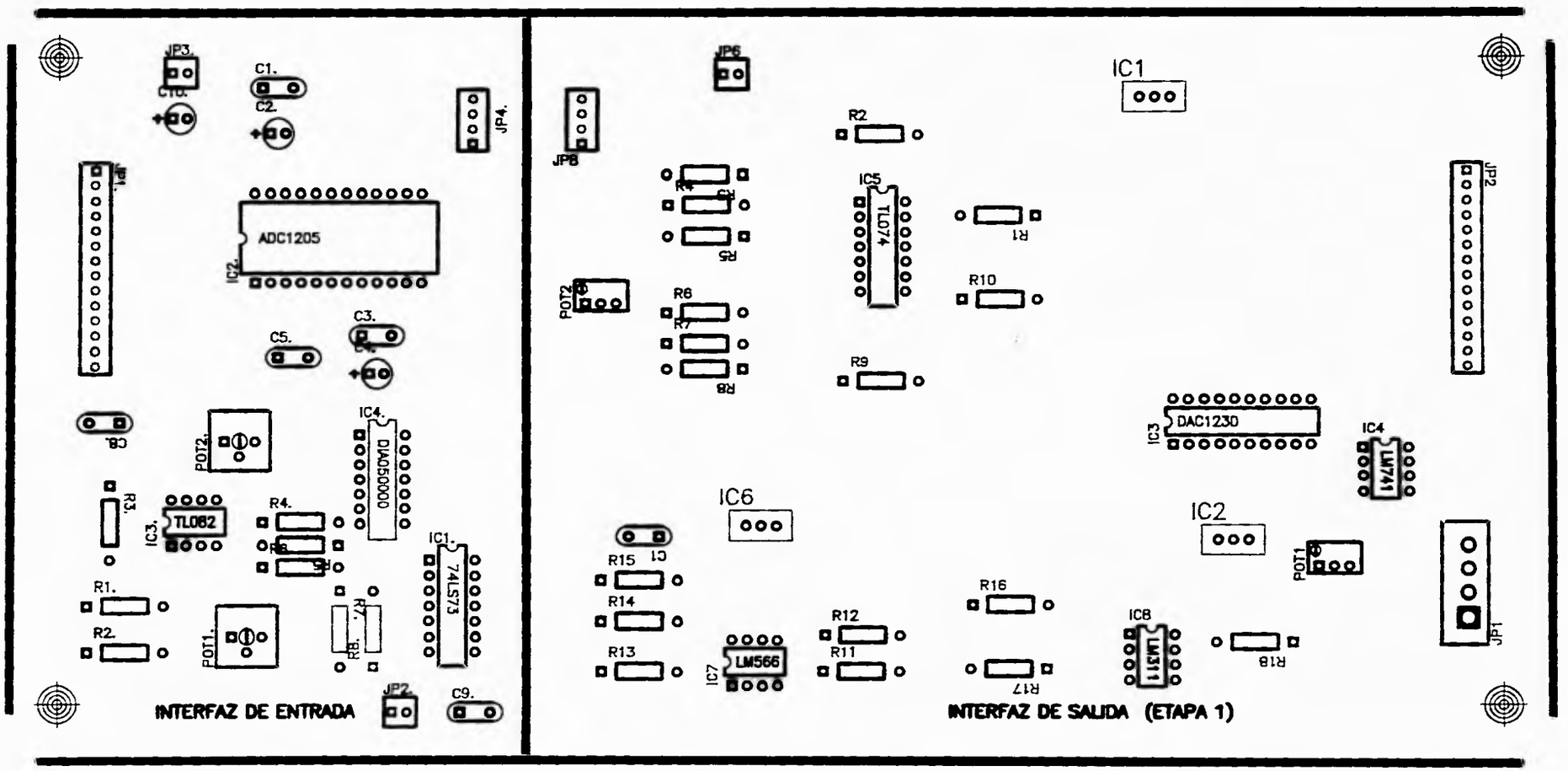


Figura 6.28 Diagrama de distribucion de componentes.

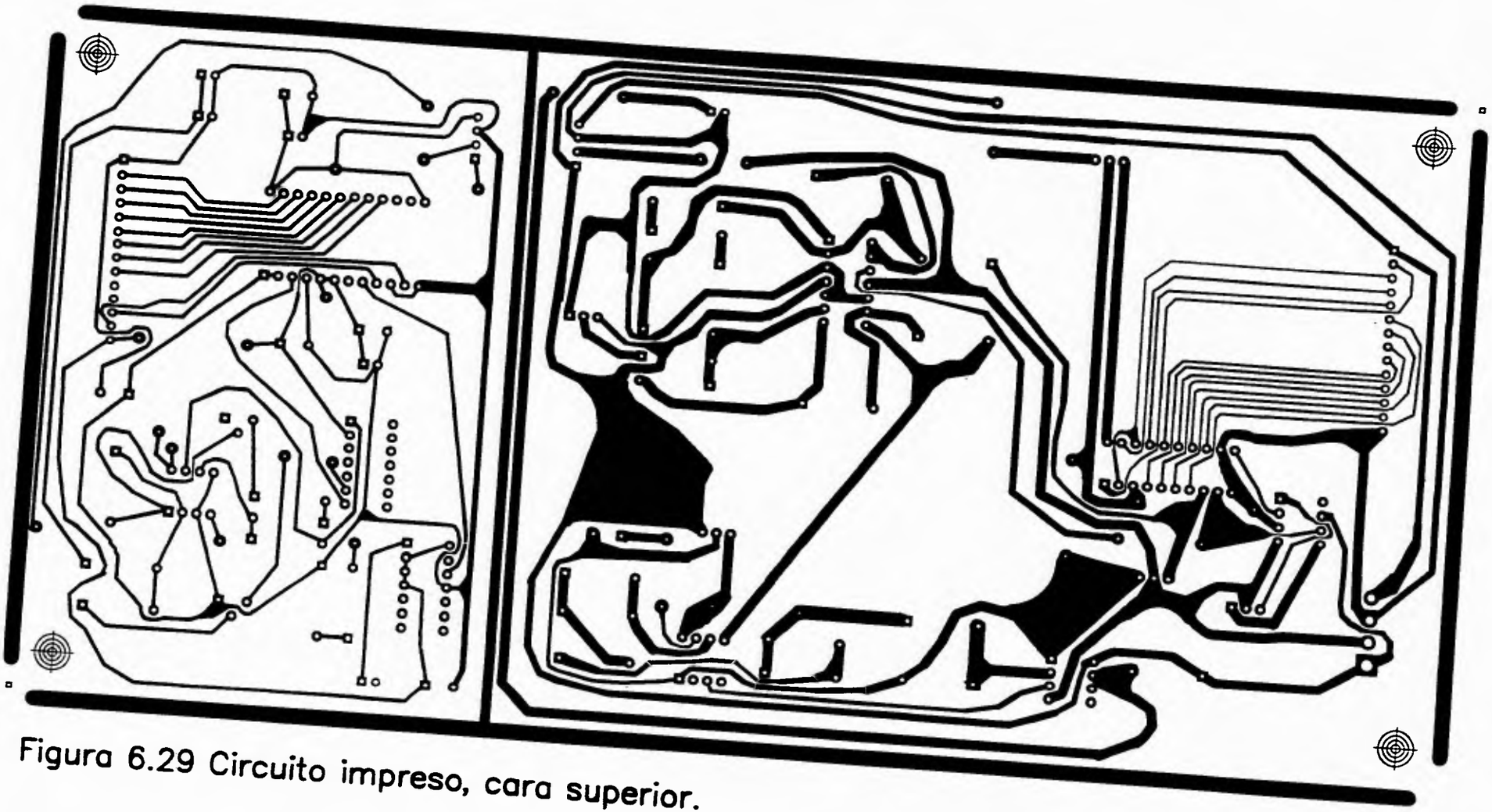


Figura 6.29 Circuito impreso, cara superior.

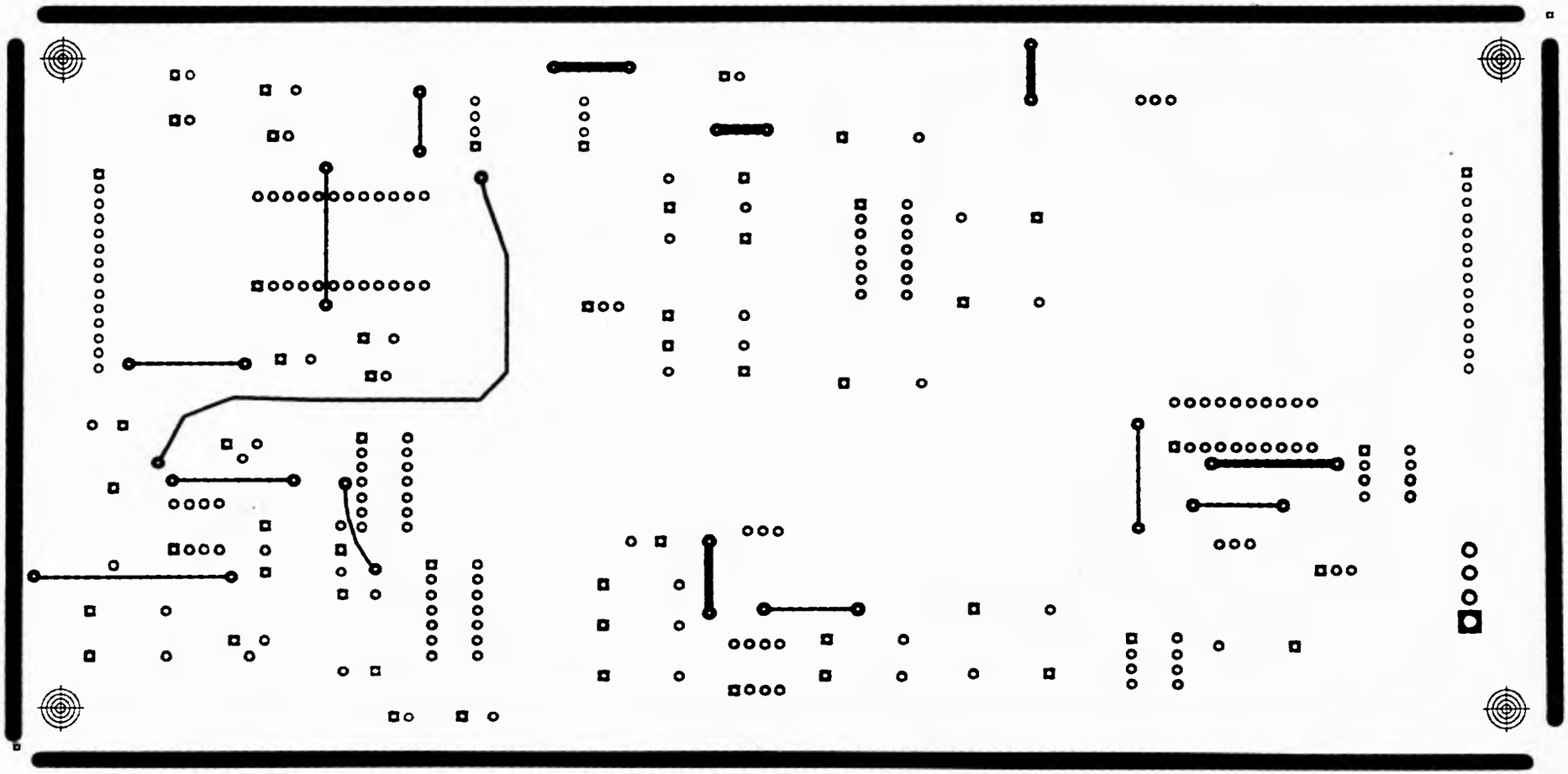


Figura 6.30 Circuito impreso, cara inferior.

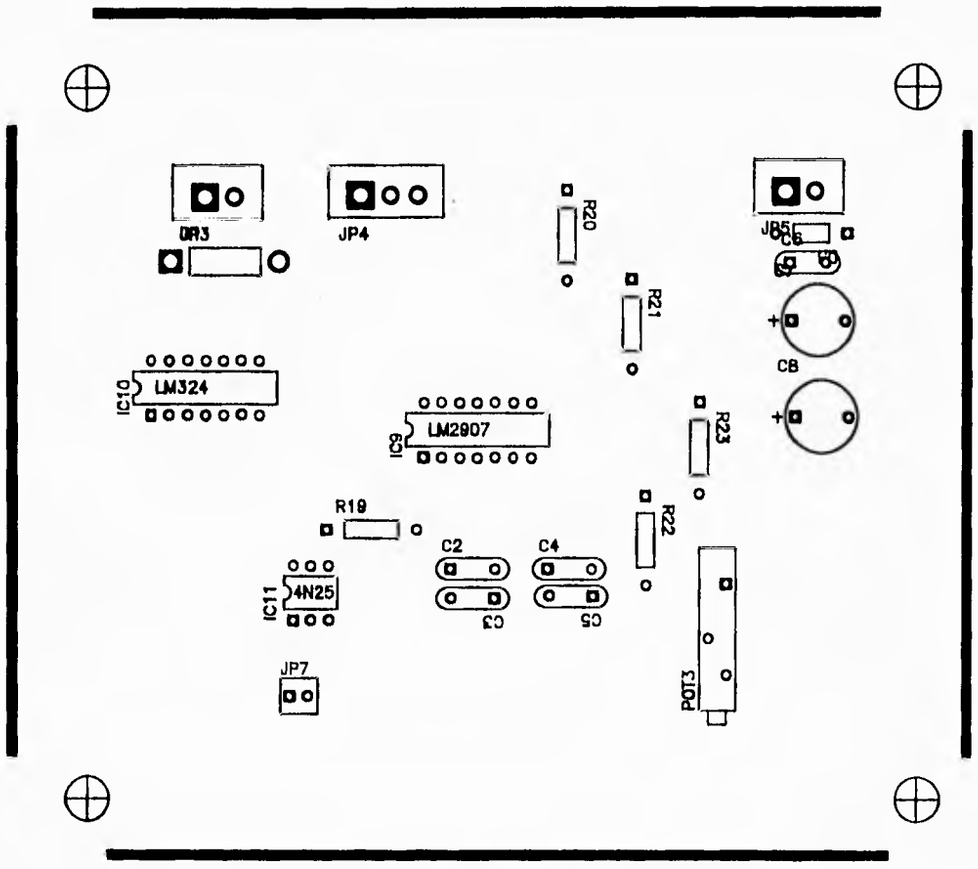


Figura 6.31 Diagrama de distribucion de componentes.

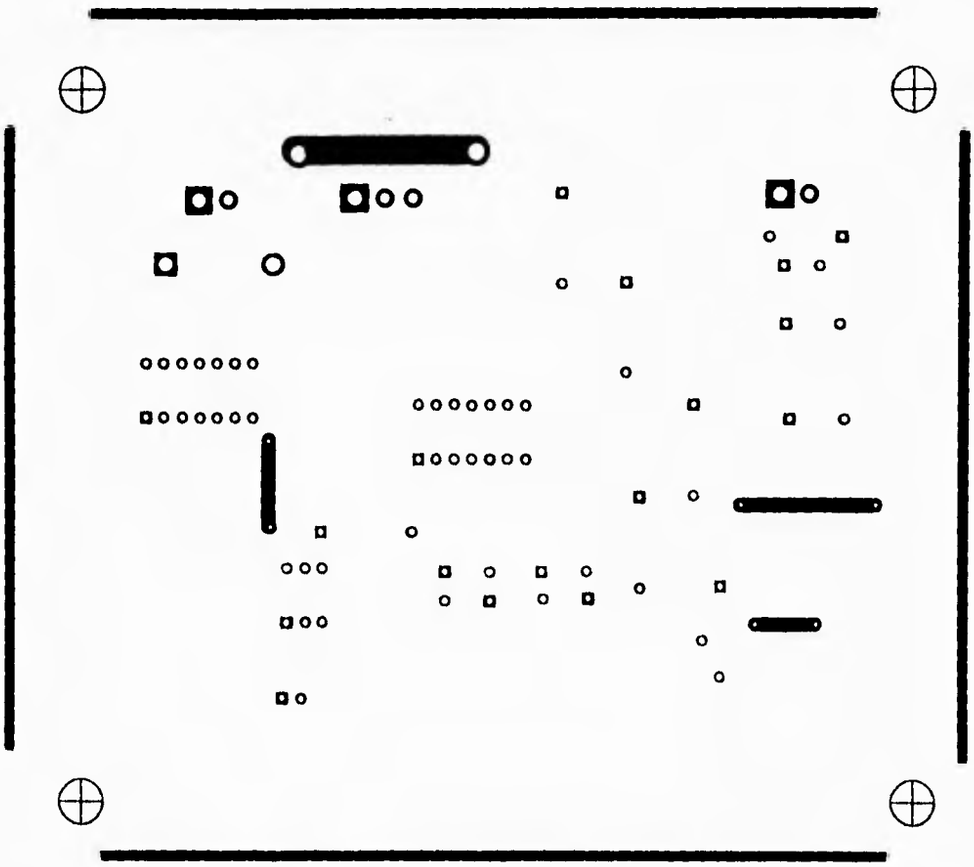


Figura 6.32 Circuito impreso, cara superior.

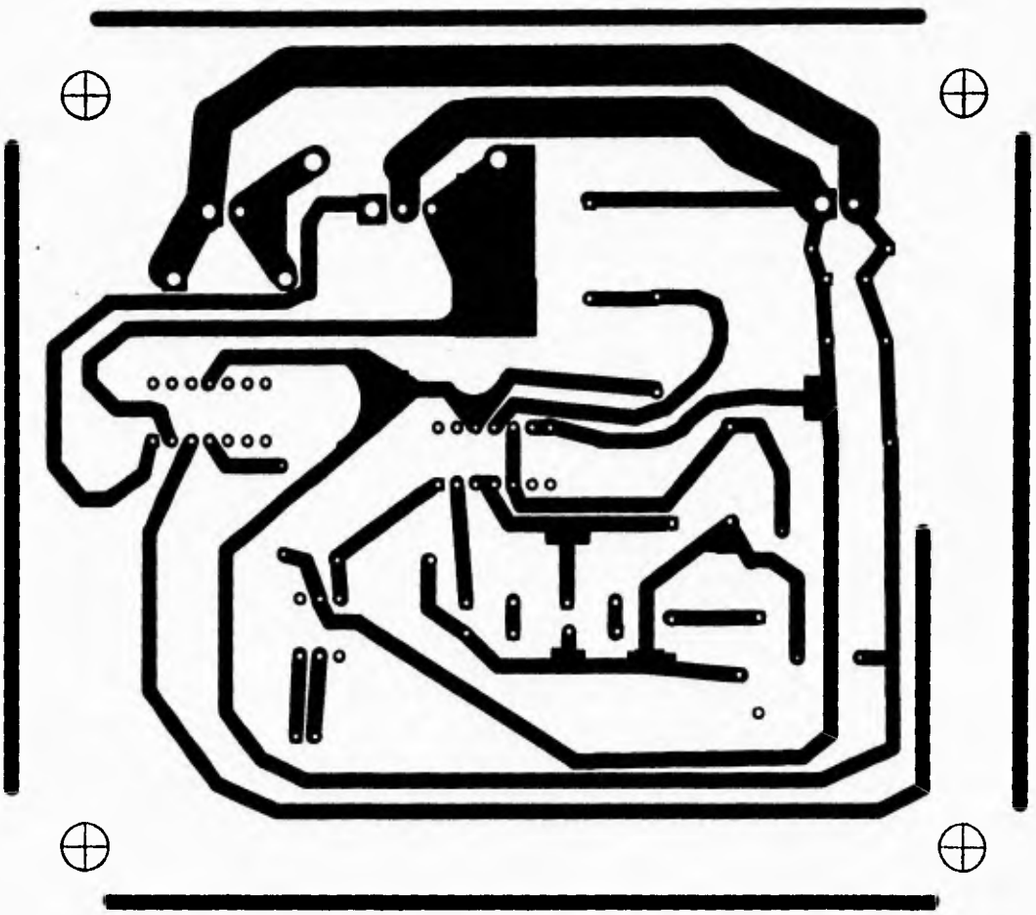


Figura 6.33 Circuito impreso, cara inferior.

VII. IMPLEMENTACION DEL SOFTWARE DEL SISTEMA

7.1 DESCRIPCION GENERAL DEL SOFTWARE DEL SISTEMA

El software en un sistema de control digital es la parte central del mismo, sin embargo, debemos tomar en cuenta que así se trate del software más sofisticado, elegante y preciso, si no se dispone del hardware que sea capaz de interpretar y emplear las ventajas del software, el sistema no funcionará correctamente, esto significa que el hardware y el software del sistema van de la mano y es necesaria una congruencia perfecta de ambos para un funcionamiento exitoso del sistema.

En este sistema de control, existen tres unidades que requieren de un software que les permita cumplir con su función dentro del mismo. Por ello, en términos generales el software de éste sistema de control puede dividirse en las siguientes partes:

1. Programación del MicroControlador Maestro (MCM)
2. Programación del MicroControlador Esclavo (MCE)
3. Programación en la Computadora Personal (PC) de un "Sistema de Interfaz con el Usuario (SIU-PID)".

Cada una de las partes antes mencionadas están formadas por un programa principal y un conjunto de rutinas y funciones asociadas.

Para la programación de los microcontroladores, se empleó el lenguaje ensamblador para el MC68HC11F1, y para la realización del sistema de interfaz con el usuario en la PC, se empleó Lenguaje C.

Es necesario señalar que para obtener la salida del controlador (sistema multiprocesador), la función de transferencia que describe al controlador PID, obtenida en el capítulo V y mostrada en la ecuación 7.1, tuvo que ser dividida en dos partes, a las que llamaremos DATOM (a la parte

del cálculo que le corresponde al MCM) y DATOE (a la parte del cálculo correspondiente al MCE), finalmente estas dos partes se suman para dar por resultado el valor de u(k) que es realmente la salida del controlador.

$$u(k) = \frac{1}{T_i(T_a + T)} [K_p [T_a(T_i - T) + T_d T_i] e(k-2) + \\ + K_p [-T_i(2T_a + T) + T(T_a + T) - 2T_d T_i] e(k-1) + \\ + K_p [T_i(T_a + T + T_d)] e(k) - [T_i T_a] u(k-2) + \\ + [T_i(2T_a + T)] u(k-1)] \qquad (7.1)$$

$$DATOM = Ae(k-2) + Be(k-1) + Ce(k) \qquad (7.2)$$

$$DATOE = Du(k-2) + Fu(k-1) \qquad (7.3)$$

$$u(k) = DATOM + DATOE \qquad (7.4)$$

donde:

$$A = \frac{1}{T_i(T_a + T)} [K_p [T_a(T_i - T) + T_d T_i]] \qquad (7.5)$$

$$B = \frac{1}{T_i(T_a + T)} [K_p [-T_i(2T_a + T) + T(T_a + T) - 2T_d T_i]] \qquad (7.6)$$

$$C = \frac{1}{T_i(T_a + T)} [K_p [T_i(T_a + T + T_d)]] \qquad (7.7)$$

$$D = \frac{1}{T_i(T_a + T)} [-T_i T_a] \qquad (7.8)$$

$$F = \frac{1}{T_i\,(T_a + T)}\,[\,T_i\,(2T_a + T)\,] \qquad (7.9)$$

En la figura 7.1 se muestra, de manera esquemática, el funcionamiento del sistema multiprocesador. Como se observa, en cada periodo de muestreo, el MCM y el MCE comienzan el cálculo de su DATOM y su DATOE, respectivamente. Tomando en cuenta que el cálculo del DATOE toma menor tiempo que el cálculo del DATOM, el MCE permanecerá esperando hasta la llegada del DATOM, y una vez que lo haya recibido, podrá continuar el cálculo de la salida u(k) del sistema multiprocesador.

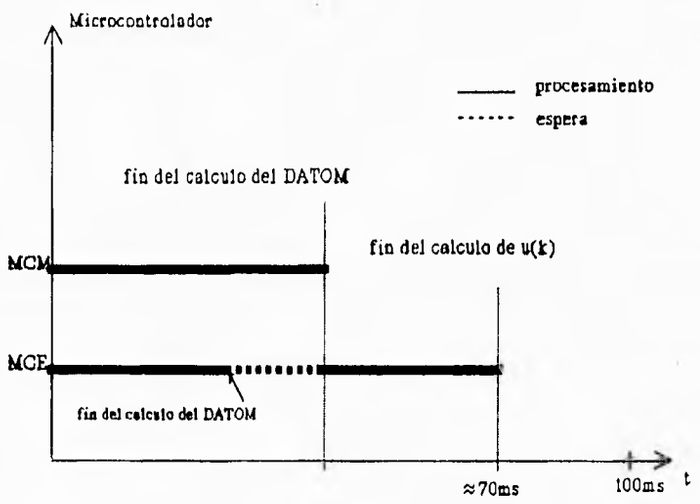


Figura 7.1 Diagrama de tiempos del sistema multiprocesador.

El MCM podrá recibir cualquier requerimiento de interrupción, excepto cuando se esté realizando alguna transmisión o recepción de datos, o cuando se encuentre dentro de alguna rutina de interrupción.

En este proyecto, se empleó un periodo de muestreo de 100[ms], tomando en cuenta las características del motor a controlar, sin embargo, éste tiempo podría reducirse, según los cálculos realizados del tiempo de procesamiento, hasta 70[ms].

7.2 PROGRAMACION DEL MICROCONTROLADOR MAESTRO

Para el cálculo y envío del DATOM, el microcontrolador maestro debe ser capaz de realizar las siguientes acciones:

- Tomar una muestra proveniente del sistema de interfaz de entrada.
- Interpretar la muestra tomada para tenerla en código ASCII.
- Enviar esta muestra a la PC.
- Conversión de la muestra a formato punto flotante
- Cálculo del DATOM empleando el paquete de aritmética de punto flotante.
- Transmisión del DATOM al MCE

En caso de que el MCM reciba alguna interrupción que indique algún requerimiento por parte de la PC, este requerimiento será atendido en el momento adecuado dentro del periodo de muestreo sin causar ningún problema, pero cuando se trate de grabar la velocidad de RAM en EEPROM o de cambiar los parámetros, después de atender a la interrupción será necesario reinicializar el sistema.

Antes de apagar el sistema, el MCM debe recibir una interrupción por hardware (IRQ) para deshabilitar la escritura a la EEPROM y evitar con esto posibles desconfiguraciones de los parámetros o la velocidad en el sistema de control.

A continuación se presentarán los diagramas de flujo, que de manera muy general, describen el programa principal y las rutinas empleadas por el MCM, sin embargo, para una mayor comprensión y análisis, en el Apéndice B se incluyen los listados del programa principal y todas las rutinas para el MCM.

7.2.1 Diagramas de Flujo

Diagrama de flujo del programa principal para el MCM

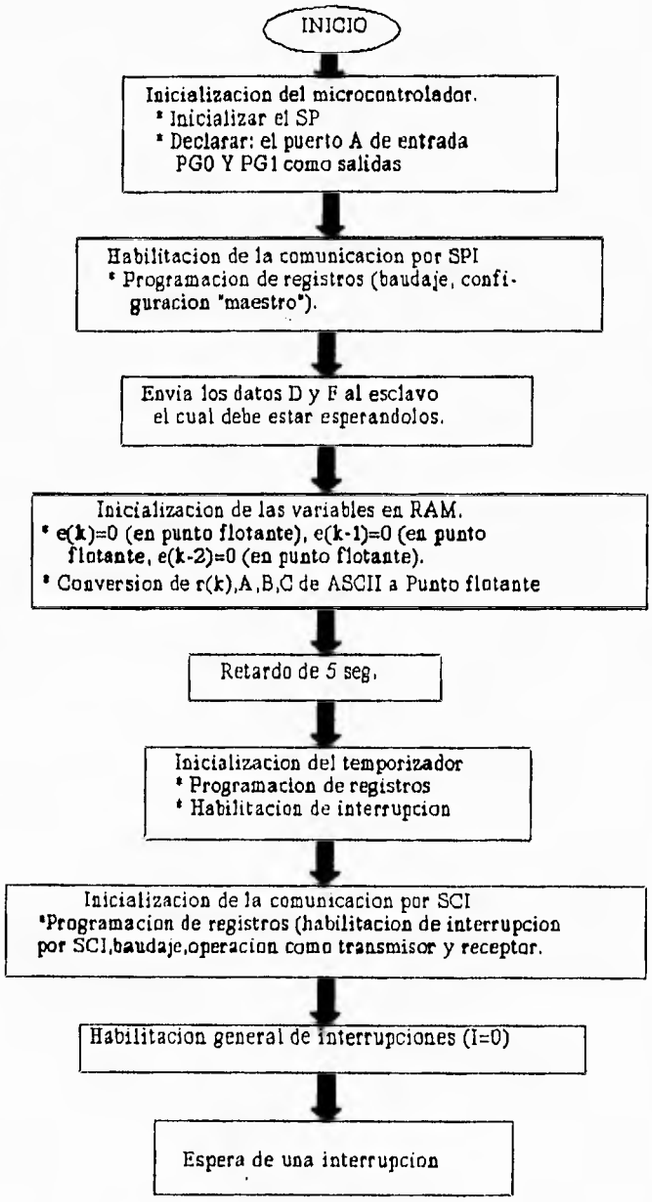


Diagrama de flujo de la rutina de servicio de interrupción por temporizador

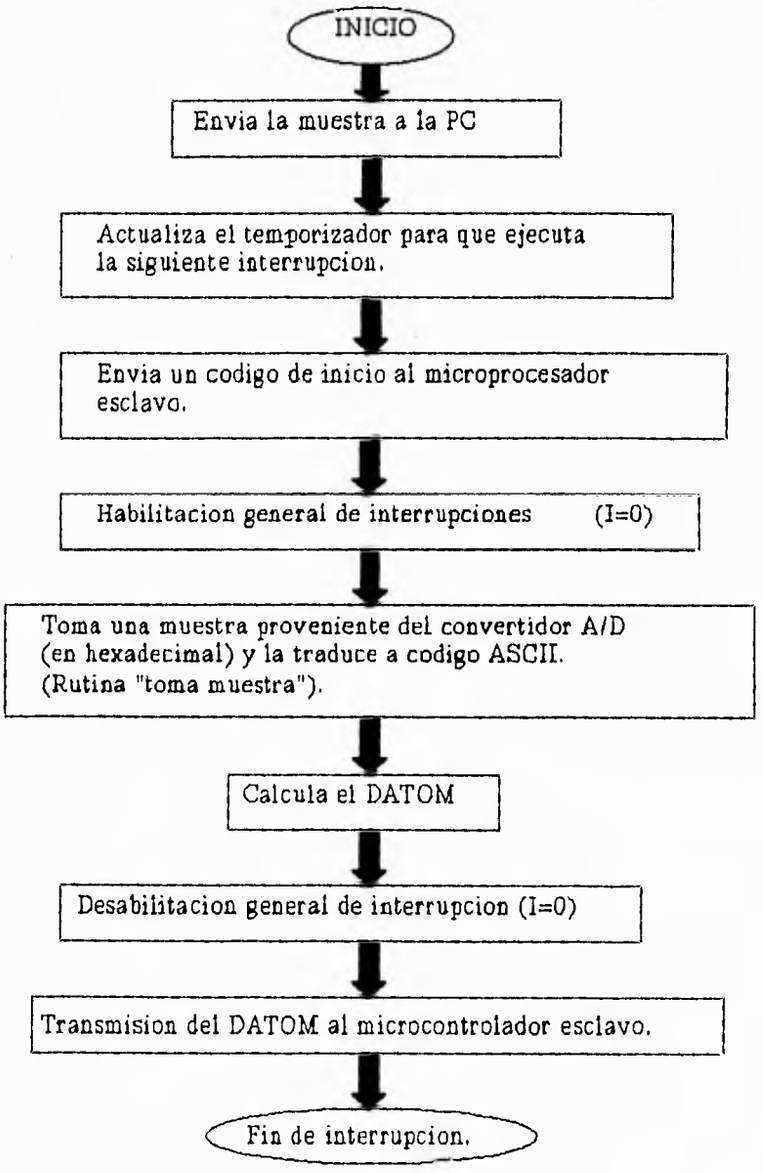
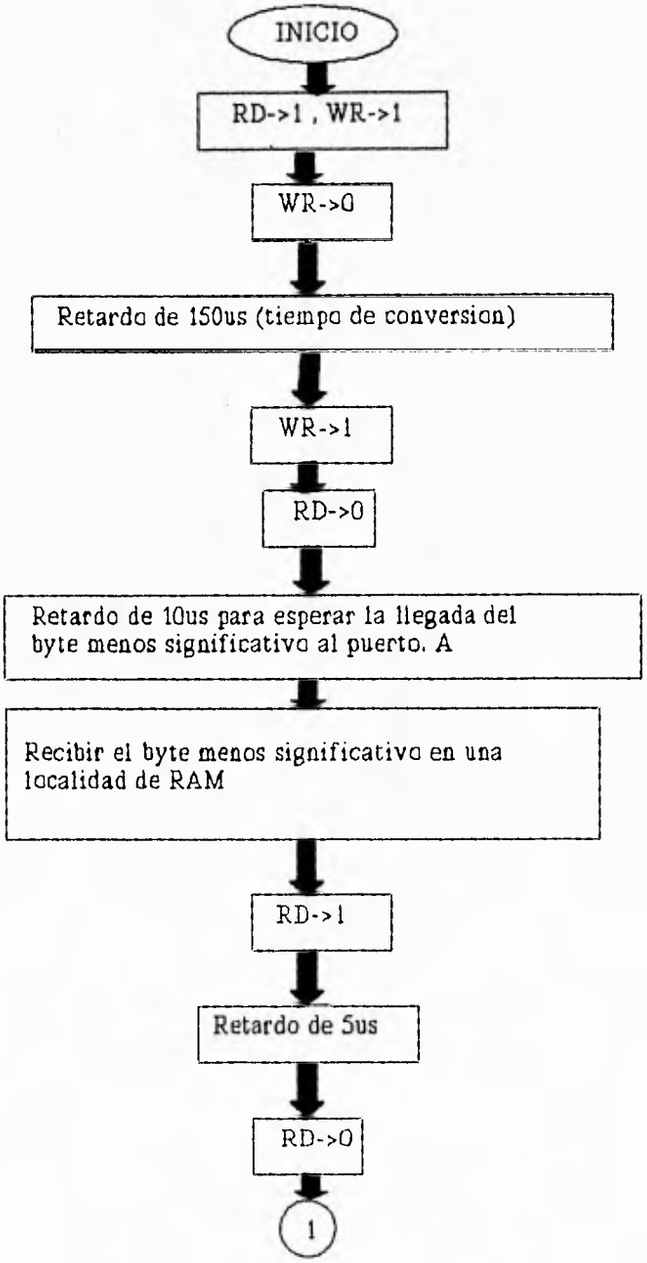


Diagrama de flujo de la rutina "toma muestra"



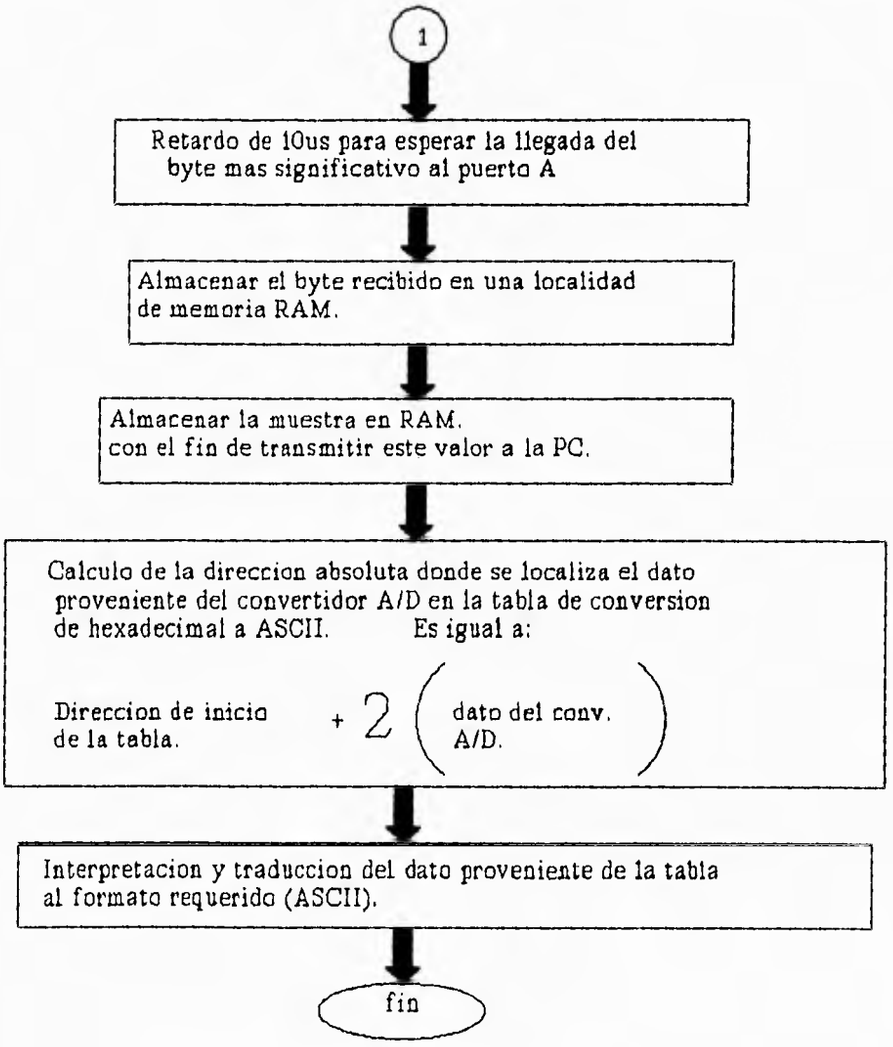

1
Retardo de 10us para esperar la llegada del byte mas significativo al puerto A
Almacenar el byte recibido en una localidad de memoria RAM.
Almacenar la muestra en RAM. con el fin de transmitir este valor a la PC.
Calculo de la direccion absoluta donde se localiza el dato proveniente del convertidor A/D en la tabla de conversion de hexadecimal a ASCII. Es igual a:
Direccion de inicio de la tabla.
+ 2 (dato del conv. A/D.)
Interpretacion y traduccion del dato proveniente de la tabla al formato requerido (ASCII).
fin

Diagrama de flujo de la rutina "calcula"

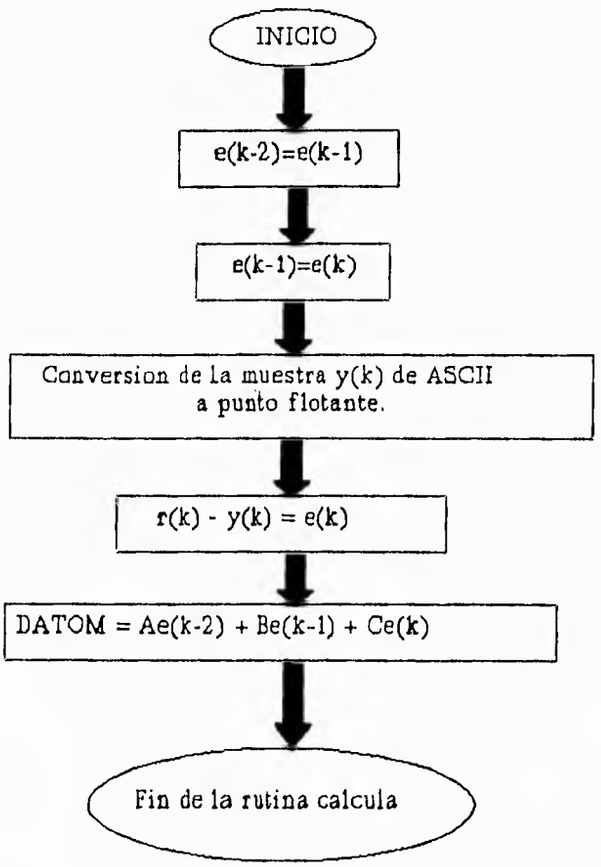
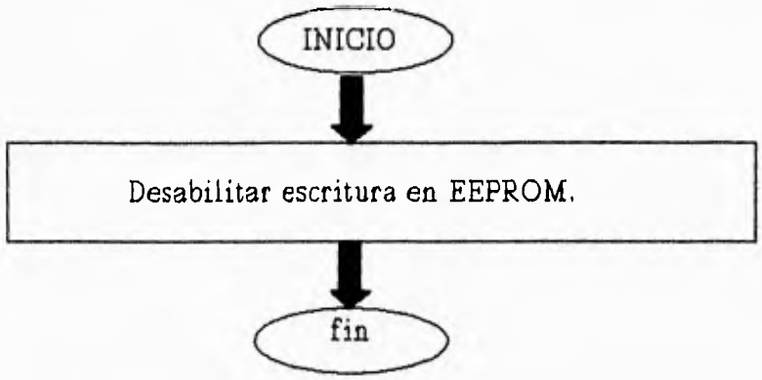
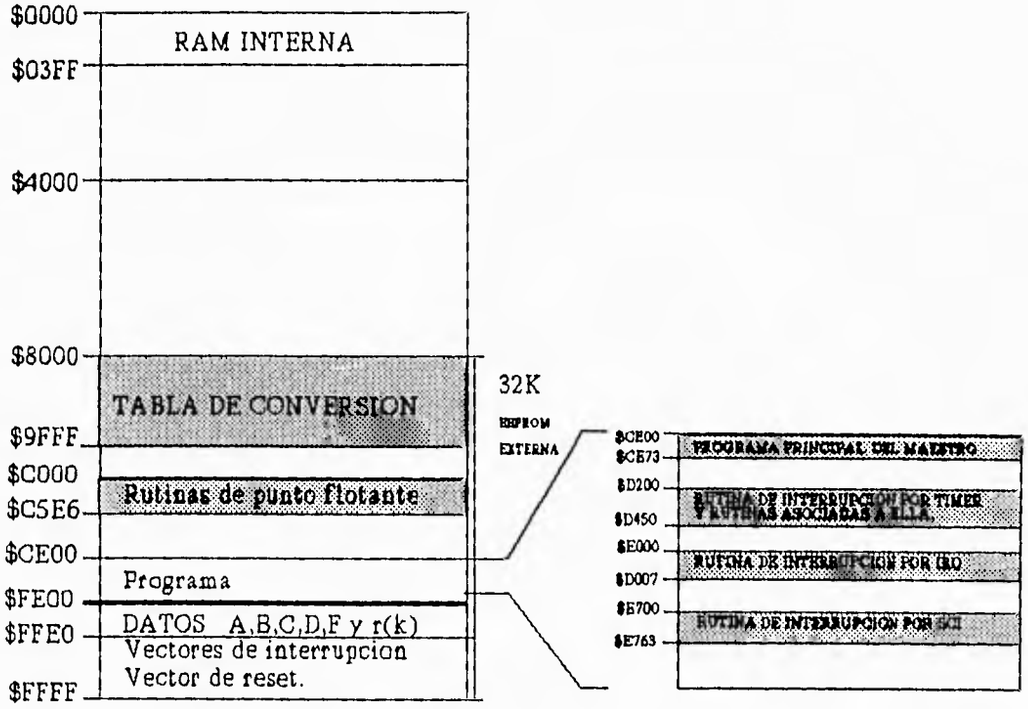


Diagrama de flujo de la rutina de interrupción por hardware (IRQ)



7.2.2 Mapa de Memoria



7.2.3 Localidades empleadas en RAM y EEPROM

Localidades empleadas en RAM

$0030 - $0031	Almacenan el dato de retardo.
$0032 - $0036	Ae(k-2) en flotante - DATOM
$0037 - $003B	Be(k-1) en flotante
$003C - $003D	REST
$0050 - $0051	Almacenan el dato proveniente del convertidor A/D (12 bits), en la $0050 el MSB y en la $0051 el LSB
$0052	Auxiliar para el acceso a la tabla de conversión.
$0053	RESA. Utilizado por la rutina de multiplicación.
$0054	RESB. Utilizado por la rutina de multiplicación.
$0055	REST. Utilizado por la rutina de multiplicación. Aquí se almacena el resultado de la multiplicación.
$005B - $005F	Número A en flotante.
$0060 - $0064	Número B en flotante.
$0065 - $0069	Número C en flotante.
$006A - $006E	e(k-2) en flotante.
$006F - $0073	e(k-1) en flotante.
$0074 - $0078	e(k) en flotante.
$0079 - $007D	r(k) en flotante (velocidad de referencia).
$007E - $0082	y(k) en flotante
$0083 - $0088	Número en ASCII procedente de la tabla de conversión.
$0100	Contador auxiliar para el envío del DATOM.
$140 - $145	Velocidad en RAM (ASCII).
$0150	Localidad testigo para actualizar A, B, C, D y F en EEPROM, puede haber un $77 o un $00.
$0151	Localidad testigo para actualizar velocidad en EEPROM, puede haber un

	$88 o un $00.
$0152 - $0159	A en ASCII en RAM
$015A - $0161	B en ASCII en RAM
$0162 - $0169	C en ASCII en RAM
$016A - $0171	D en ASCII en RAM
$0172 - $0179	F en ASCII en RAM
$0180	Número de datos a grabar en EEPROM.
$0181 - $0182	Dirección del dato en RAM que va a ser grabado en EEPROM
$0183 - $0184	Dirección de la 1er localidad en que se va a grabar en EEPROM.
$0185	No. de bytes que se van a recibir por SCI.
$0186 - $0187	Dirección a partir de la que se van a recibir los datos que lleguen por SCI.
$0188	No. de bytes a enviar por SCI en la rutina ENVIAR.
$0189 - $018A	Se almacena la dirección del 1er byte a enviar.
$018B	No. de bytes a enviar al MCE, correspondientes a los parámetros D y F.
$018C - $018D	Muestra que se envía a la PC.

Localidades empleadas en EEPROM

$FC1F - $FC20	T (Periodo de muestreo).
$FE01 - $FE08	Número A en ASCII, 7 bytes para el dato y un $00 como byte de fin.
$FE09 - $FE10	Número B en ASCII, 7 bytes para el dato y un $00 como byte de fin.
$FE11 - $FE18	Número C en ASCII, 7 bytes para el dato y un $00 como byte de fin.
$FE19 - $FE1E	r(k) (Velocidad de referencia) en ASCII. 5 bytes para el dato y un $00 como byte de fin.
$FE21 - $FE28	Número D en ASCII, 7 bytes para el dato y un $00 como byte de fin.
$FE29 - $FE30	Número F en ASCII, 7 bytes para el dato y un $00 como byte de fin.

7.3 PROGRAMACION DE MICROCONTROLADOR ESCLAVO

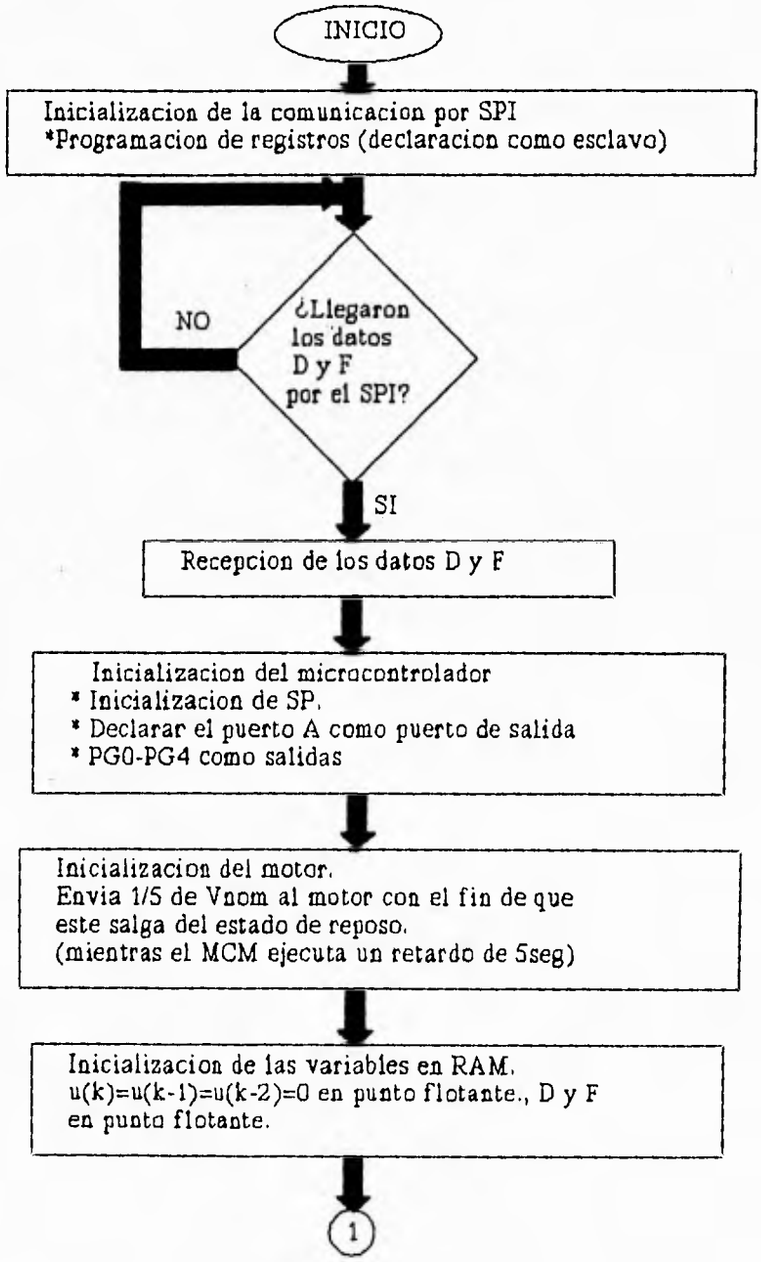
De manera general, el microcontrolador esclavo debe ser capaz de realizar las siguientes acciones para enviar, como salida del sistema multiprocesador, el valor u(k).

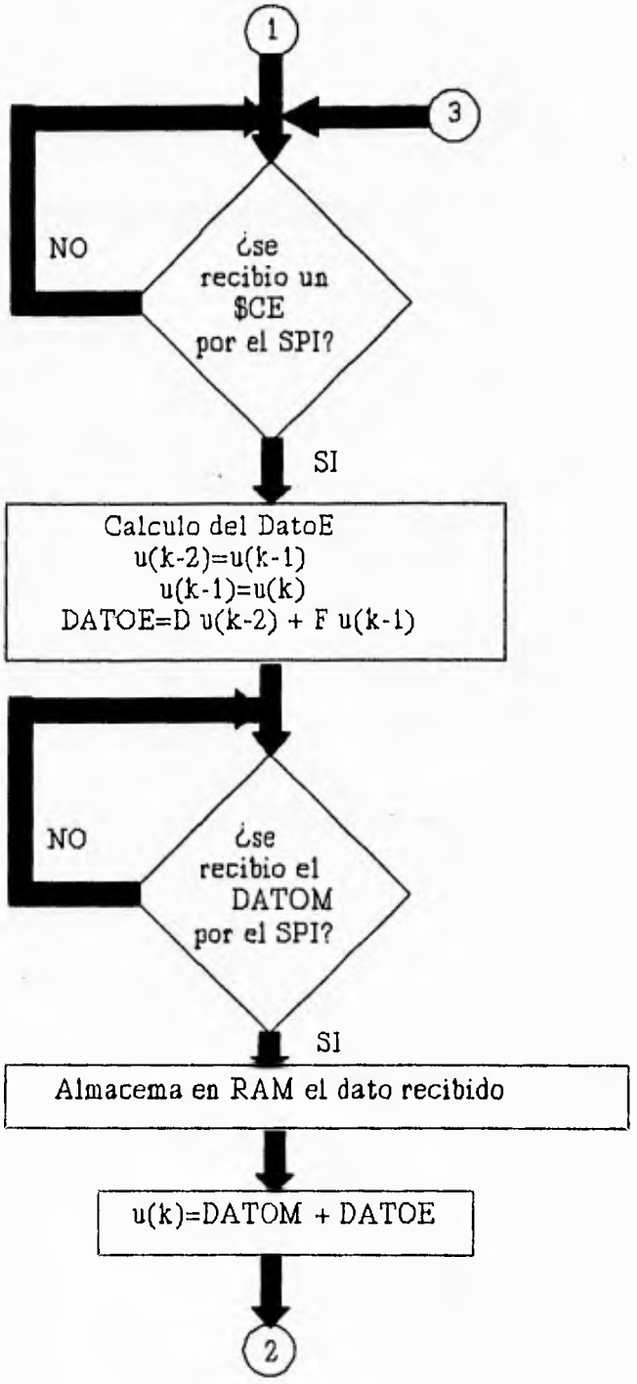
- Actualización de u(k-2)
- Actualización de u(k-1)
- Cálculo del DATOE
- Espera del DATOM
- Una vez recibido el DATOM, cálculo de u(k)
- Envío de u(k) al convertidor D/A.

A continuación se presentarán los diagramas de flujo que de manera muy general describen el programa principal y las rutinas empleadas por el MCE, sin embargo, para una mayor comprensión y análisis, en el Apéndice B se incluyen los listados del programa principal y todas las rutinas para el MCE.

7.3.1 Diagramas de Flujo

Diagrama de flujo del programa principal




1
3
NO
¿se recibio un $CE por el SPI?
SI
Calculo del DatoE
u(k-2)=u(k-1)
u(k-1)=u(k)
DATOE=D u(k-2) + F u(k-1)
NO
¿se recibio el DATOM por el SPI?
SI
Almacema en RAM el dato recibido
u(k)=DATOM + DATOE
2

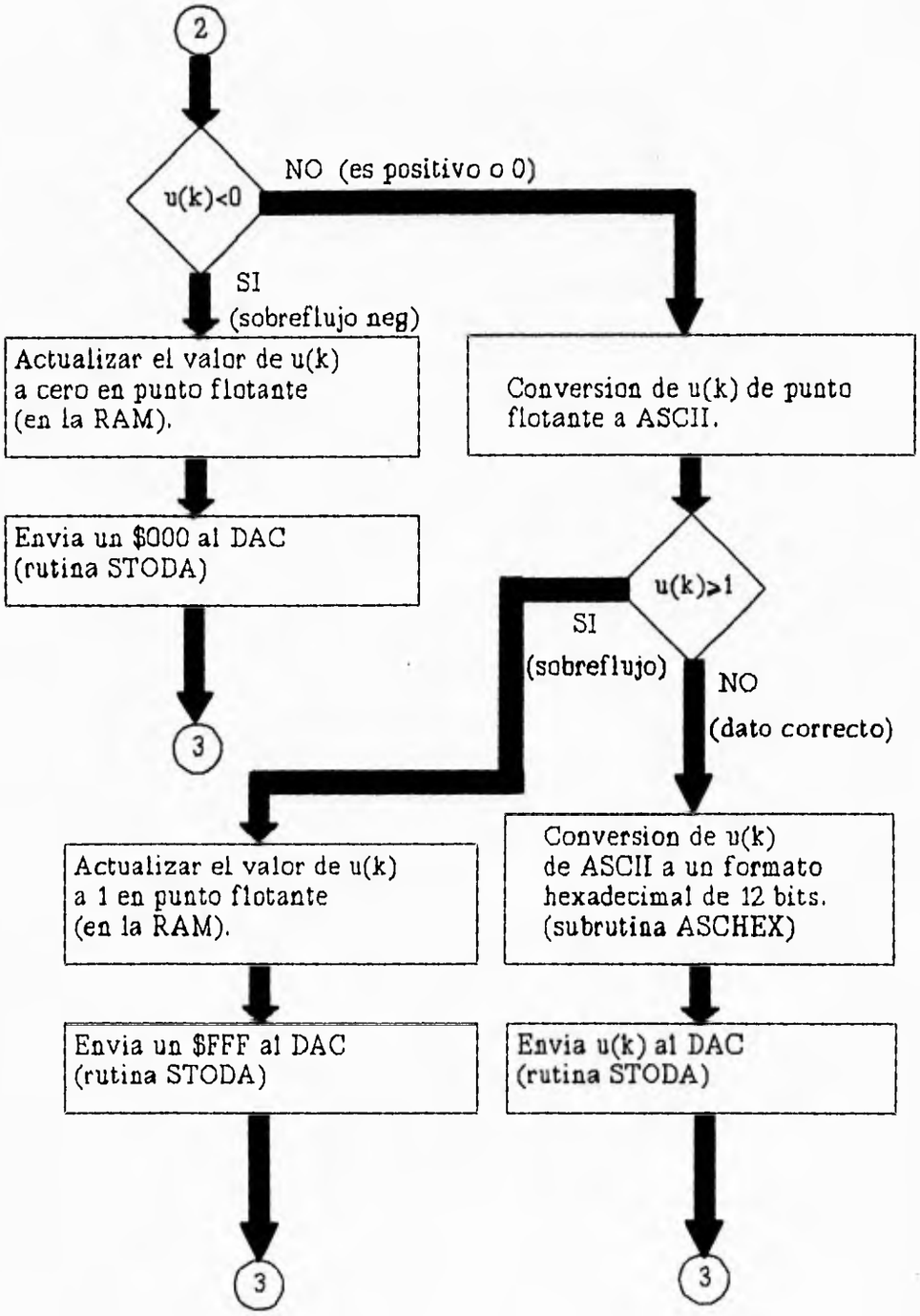
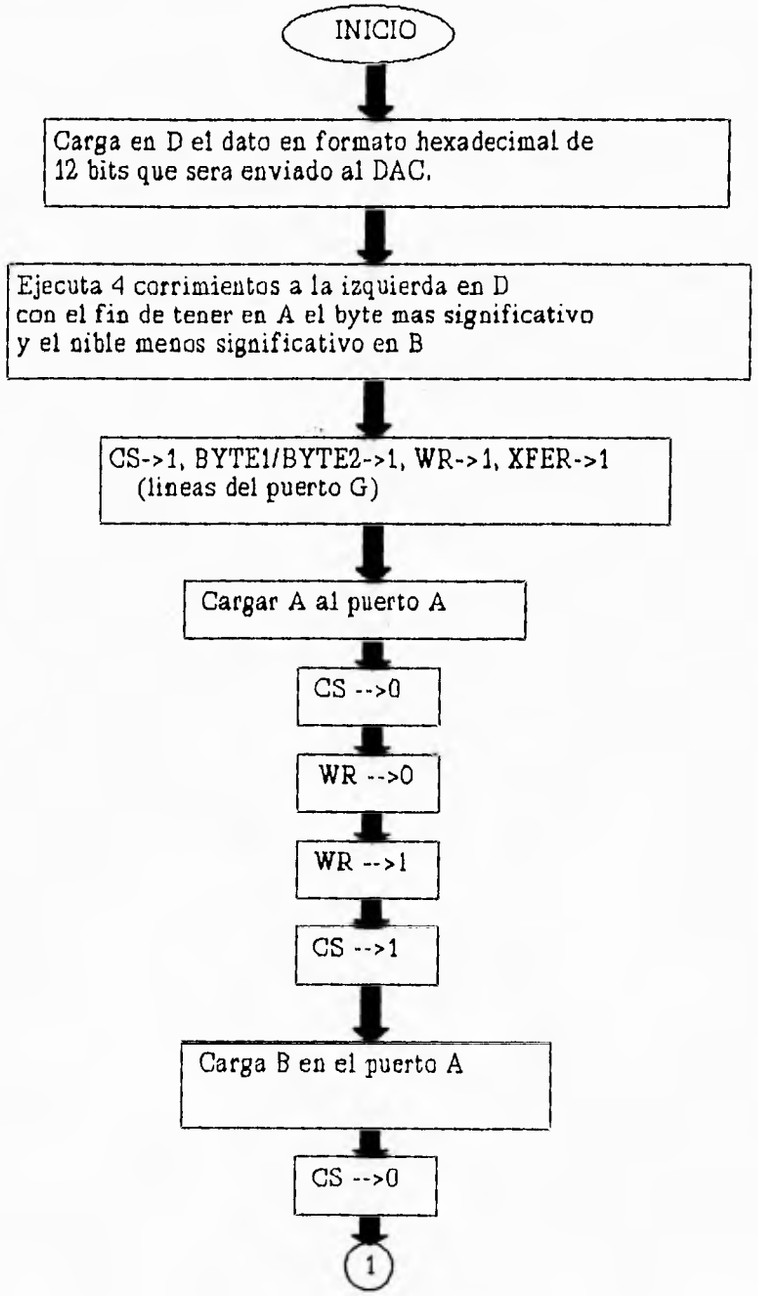
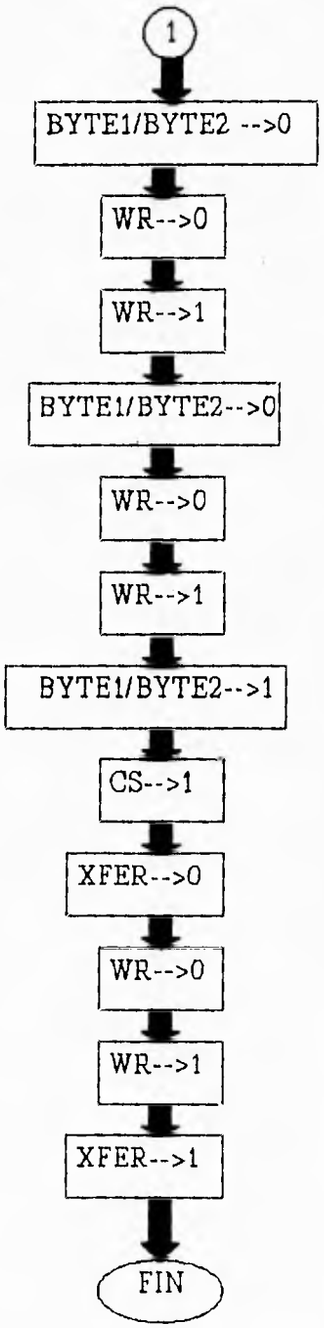
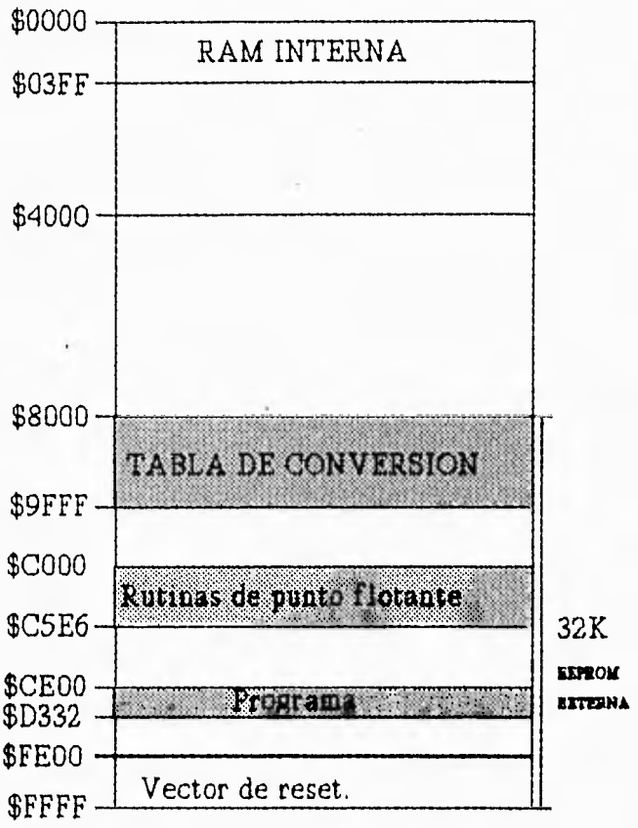

2
u(k)<0
NO (es positivo o 0)
SI
(sobreflujo neg)
Actualizar el valor de u(k)
a cero en punto flotante
(en la RAM).
Conversion de u(k) de punto
flotante a ASCII.
Envia un $000 al DAC
(rutina STODA)
u(k)>1
SI
(sobreflujo)
NO
(dato correcto)
3
Actualizar el valor de u(k)
a 1 en punto flotante
(en la RAM).
Conversion de u(k)
de ASCII a un formato
hexadecimal de 12 bits.
(subrutina ASCHEX)
Envia un $FFF al DAC
(rutina STODA)
Envia u(k) al DAC
(rutina STODA)
3
3

Diagrama de flujo de la rutina STODA (envío de la salida u(k) al convertidor digital/analógico (DAC))




1
BYTE1/BYTE2 -->0
WR-->0
WR-->1
BYTE1/BYTE2-->0
WR-->0
WR-->1
BYTE1/BYTE2-->1
CS-->1
XFER-->0
WR-->0
WR-->1
XFER-->1
FIN

7.3.2 Mapa de Memoria



7.3.3 Localidades empleadas en RAM

$0030 - $0031	Auxiliar para almacenar X.
$0032 - $0033	Dato a enviar al convertidor D/A.
$0034	Auxiliar para la conversión de ASCII a hexadecimal del dato a enviar al convertidor D/A.
$0050 - $0054	u(k) en flotante.
$0055 - $0059	u(k-1) en flotante.
$005A - $005E	u(k-2) en flotante.

$005F - $0063	Du(k-2) en flotante/DATOE.
$0064 - $0068	DATOM.
$0069 - $006A	u(k) en ASCII (Valor a enviar para conversión a hexadecimal).
$006B - $006F	D en flotante.
$0070 - $0074	F en flotante.
$0090	Bandera auxiliar para regresar a diferentes partes del programa después de enviar un dato al convertidor D/A.
$0100	u(k) en ASCII.
$150 - $157	D en ASCII.
$158 - $15F	F en ASCII.

7.4 PAQUETE DE ARITMETICA DE PUNTO FLOTANTE

Mientras muchas aplicaciones pueden ser implementadas empleando únicamente las operaciones con números enteros que el MC68HC11 es capaz de realizar, otras aplicaciones o algoritmos resultan muy difíciles o imposibles de implementar sin operaciones matemáticas de punto flotante. La meta fundamental de las rutinas de punto flotante que se emplearon en este proyecto es proveer un camino rápido y flexible para realizar operaciones de punto flotante para aplicaciones tales como controladores, entre muchas otras.

El paquete de rutinas de punto flotante proporcionado por Motorola, permite implementar no solo las operaciones aritméticas básicas (suma, resta, multiplicación y división), sino también contiene rutinas para convertir de formato ASCII a formato punto flotante y viceversa, así como las tres funciones trigonométricas básicas: seno, coseno y tangente, y rutinas para convertir de radianes a grados y de grados a radianes, además de la raíz cuadrada. Sin embargo, en éste proyecto, solamente se manejan las operaciones de suma, resta y multiplicación, así como las rutinas de conversión de formato ASCII a formato punto flotante y de formato punto flotante a formato ASCII.

El paquete completo de rutinas de punto flotante requiere aproximadamente 2Kbytes de memoria, sin embargo, para este proyecto, se requiere un poco menos puesto que no se emplearon todas las rutinas que contiene el paquete. Además de esta memoria EPROM requerida, las rutinas de punto flotante emplean los 10 primeros bytes de la página cero da la RAM, y todas las variables temporales utilizadas son almacenadas en el stack.

Los 10 bytes de la página cero de la RAM son usados por dos acumuladores de punto flotante, FPACC1 y FPACC2. Para cada uno de estos acumuladores, el primer byte indica el exponente, los 3 siguientes bytes la mantisa y un byte es empleado para el signo de la mantisa.

A continuación se presenta una breve descripción de cada una da las subrutinas a las que se invoca dentro de los programas del MCM y del MCE:

CONVERSION DE ASCII A PUNTO FLOTANTE.

Nombre de la subrutina: ASCFLT
Operación: ASCII(X) -> FPACC1
Tamaño: 352 Bytes
Espacio de Stack: 14 Bytes
Entrada: Registro X apunta a la cadena ASCII a convertir
Salida: FPACC1 contiene el número en punto flotante
Notas: Esta rutina convierte un número especificado en código ASCII, al formato requerido por todas las rutinas de punto flotante. La conversión se detiene ya sea cuando se encuentra un caracter no decimal antes del exponente o después de que se han leído dos dígitos del exponente.

MULTIPLICACION EN PUNTO FLOTANTE

Nombre de la subrutina: FLTMUL
Operación: FPACC1 x FPACC2 -> FPACC1

Tamaño: 169 Bytes

Espacio de Stack: 10 Bytes

Entrada: FPACC1 y FPACC2 contienen los números a multiplicar

Salida: FPACC1 contiene el producto de los dos acumuladores. FPACC2 permanece sin cambio.

SUMA EN PUNTO FLOTANTE

Nombre de la subrutina: FLTADD

Operación: FPACC1 x FPACC2 -> FPACC1

Tamaño: 194 Bytes

Espacio de Stack: 6 Bytes

Entrada: FPACC1 y FPACC2 contienen los números a sumar

Salida: FPACC1 contiene la suma de los dos acumuladores. FPACC2 permanece sin cambio

Notas: La rutina de suma en punto flotante realiza la suma con signo. Ambos acumuladores pueden tener mantisas con el mismo o con diferente signo.

RESTA EN PUNTO FLOTANTE

Nombre de la subrutina: FLTSUB

Operación: FPACC1 - FPACC2 -> FPACC1

Tamaño: 12 Bytes

Espacio de Stack: 8 Bytes

Entrada: FPACC1 y FPACC2 contienen los números a restar

Salida: FPACC1 contiene la diferencia de los dos acumuladores. FPACC2 permanece sin cambio.

Notas: Esta rutina únicamente invierte el signo de FPACC2, llama a FLTADD y posteriormente regresa a FPACC2 su signo original.

CONVERSION DE PUNTO FLOTANTE A ASCII.

Nombre de la subrutina: FLTASC

Operación: FPACC1 -> ASCII(X)

Tamaño: 370 Bytes

Espacio de Stack: 28 Bytes

Entrada: FPACC1 contiene el número a convertir a ASCII. El registro X apunta a un buffer de 14 bytes

Salida: El buffer al que apunta X contiene la cadena ASCII que representa el número en FPACC1. La cadena es terminada con un byte 00 y el registro X apunta al inicio de la cadena.

7.5 SISTEMA DE INTERFAZ CON EL USUARIO (SIU - PID)

El Sistema de Interfaz con el Usuario para el sistema de control PID construido (SIU - PID), forma una parte fundamental y muy valiosa, tanto para el manejo del sistema de control (cambios de velocidad y variación de parámetros del controlador PID), como para el monitoreo en tiempo real del funcionamiento del sistema, para el análisis de la respuesta transistoria y para corregir posibles desconfiguraciones del sistema en caso de fallas en el suministro de energía eléctrica.

Este sistema fue construido empleando la técnica de programación estructurada en lenguaje C de Borland. El listado completo de programa principal y las funciones auxiliares se encuentra en el Apéndice C, incluye algunos comentarios que permiten una mejor comprensión de su funcionamiento.

Es conveniente hacer notar que para un mejor funcionamiento, y con el fin de evitar problemas de graficación ante la gran diversidad de monitores que existen, el SIU-PID maneja una pantalla virtual que lo hace compatible con cualquier tipo de monitor.

El puerto empleado para la comunicación con el MCM es el COM1, el cual se inicializó de la siguiente forma:

Velocidad de transmisión: 9600 bits/seg.
8 bits de datos
1 bit de inicio
1 bit de paro
No paridad

7.5.1 Recomendaciones para un mejor fucionamiento

Tomando en cuenta que el SIU-PID funciona en un ambiente gráfico, es recomendable, disponer de un monitor a color, aunque esto no es un requerimiento del sistema, ya que por sus características de diseño, el SIU-PID puede ser ejecutado sin ningún problema de deformación de imágen en cualquier tipo de monitor.

Dado que el tiempo, en este sistema, es un factor muy importante, y gracias a que el acceso a disco duro es mucho más rápido que el acceso a un disco flexible, es conveniente ejecutar el SIU-PID desde disco duro, únicamente para evitar el posible retardo inicial al abrir los archivos que maneja el SIU-PID. Sin embargo, éste posible retardo sería el únioo problema si el sistema se ejecutara desde un disco flexible.

7.5.2 Acceso al sistema

Estando el sistema de control apagado, antes de encender la computadora, es necesario verificar que el cable de comunicación con el MCM esté conectado al puerto COM1 de la computadora personal y al conector J1 de la tarjeta del sistema multiprocesador. Una vez que esta

conexión haya sido hecha y verificada, debe encenderse la computadora.

Estando presente en la pantalla de la computadora, el *prompt* del sistema operativo MS-DOS, y estando dentro del directorio donde se localice el SIU-PID, solamente es necesario ejecutar el programa SIU.EXE, tecleando:

SIU
y posteriormente oprimiendo <ENTER>

Una vez hecho esto aparecerá la pantalla inicial del SIU-PID.

7.5.3 Módulos del SIU-PID

En la figura 7.2 se muestra la pantalla inicial del SIU-PID. Se puede ver que esta pantalla esta dividida en las siguientes partes:

- Menú principal
- Pantalla de texto
- Gráfica de la respuesta transitoria
- Gráfica en tiempo real
- Monitor numérico de velocidad

Las cuales se describirán en forma detallada.

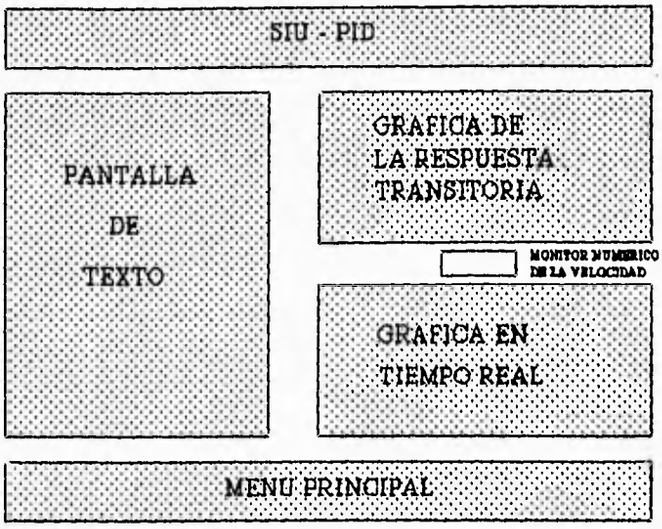


Figura 7.2 Pantalla principal del SIU - PID

Menú principal

El menú principal esta formado por 5 opciones:

<F1> Velocidad
<F2> Parámetros
<F3> Gráfica
<F4> Archivo
<F10> Salir

que se describen a continuación:

<F1> Velocidad.

Al oprimir <F1> aparecerá un segundo menú con 3 opciones:

<1> Cambiar velocidad

<2> Grabar velocidad
<ESC> Menú anterior

La opción <1> permite cambiar la velocidad del motor a cualquier otro valor dentro del rango permitido (800 rpm - 3500 rpm); si se intentara salir de este rango, el SIU-PID, por seguridad, no lo permitiría. Además, en caso de algún error al introducir la nueva velocidad, el usuario puede salir de esta opción (presionando <ESC>) en cualquier momento, antes de confirmar que la nueva velocidad es correcta (oprimiendo <ENTER>).

Una vez que el motor está trabajando a la velocidad deseada, la opción <2>, permite grabar, si se requiere, esta velocidad en la EEPROM del MCM, con el fin de que al encender o reinicializar nuevamente el sistema de control, el motor opere a la misma velocidad que fue grabada.

Debido a que el tiempo que toma el grabar la velocidad es aproximadamente de 150ms, y para evitar posibles descompensaciones en el sistema de control, dado que 150ms para un microcontrolador que trabaja a una frecuencia de 2MHz, es un tiempo muy grande, decidimos que por seguridad, es conveniente detener el motor antes de grabar velocidad, siendo necesario posteriormente reinicializar el sistema de control.

Para que el usuario no tenga problemas cuando desee grabar velocidad, si en algún momento olvida que la velocidad del motor se reducirá a la mínima, el SIU-PID, envía un mensaje de advertencia, al cual, el usuario responderá si desea realmente grabar la velocidad a la que está trabajando su motor, recordándole que para lograrlo, el motor se detendrá.

Durante el tiempo de grabación, aparecerá un mensaje en la pantalla que indica que es necesario esperar, y una vez que sea posible, aparecerá un mensaje al usuario para que éste apague o reinicialice, según lo desee, su sistema de control.

La opción <ESC> permite regresar al menú anterior.

<F2> Parámetros

Dentro de esta opción se tienen las siguientes:

<1> Cambiar parámetros
<2> Ver parámetros
<ESC> Menú anterior

La opción <1> del SIU-PID, da al sistema de control, la gran ventaja de poder modificar los parémetros K_p, T_d, y T_i del controlador PID, con el fin de sintonizar el controlador para cumplir con las especificaciones (Mp, ts, ta, etc.), que se requieran, según la aplicación.

A diferencia de muchos controladores PID analógicos que existen en la actualidad, en los cuales, es posible sintonizar el controlador, variando en forma independiente y manual cada uno de los parámetros del controlador, en este caso, es posible variar ya sea en forma simultánea o independiente, los parámetros, y posteriormente, observar y analizar la respuesta del motor; repitiendo este proceso tantas veces como sea necesario, para obtener la respuesta más adecuada.

Dado que la ecuación en diferencias del controlador PID está en función de los datos A,B,C,D y F, definidos anteriormente, los cueles, a su vez, estén en función de K_p, T_d y T_i, un cambio en K_p, T_d y/o T_i requiere un nuevo cálculo de A,B,C,D y F y une actualización de estos valores en las locelidades de EEPROM del MCM, por ello, de igual forma que el grebar velocidad, antes de hacer un cambio de parámetros, es necesario detener el motor, para lo cual, el SIU-PID envía un mensaje de advertencia al usuario, antes de enviar al sistema multiprocesador, la indicación necesarie pare reducir le velocidad hasta detener al motor.

Posteriormente, el SIU-PID, pide al usuario, la introducción de los nuevos parámetros pera el controlador (K_p, T_d y T_i) y una vez que el usuario confirma (oprimiendo <ENTER>), que los nuevos parámetros son correctos, el SIU-PID calcula los datos A,B,C,D y F y los envía en formato

ASCII al MCM, con el fin de que este microcontrolador grabe estos datos en su EEPROM. Durante este tiempo de grabación, el SIU-PID, despliega un mensaje que indica al usuario que espere, y una vez grabados los datos, se envía un mensaje para apagar o reinicializar el sistema de control.

La opción <2> ver parámetros, da al usuario la ventaja de poder ver en cualquier momento los parámetros con los que está trabajando el controlador PID, éstos se almacenan permanentemente en un archivo en el directorio de trabajo.

La opción <ESC> permite regresar al menú anterior.

<F3> Gráfica

Esta opción permite "congelar la gráfica", durante el tiempo deseado hasta oprimir <ENTER> para continuar el monitoreo. Esto permite hacer una lectura y visualización de la gráfica, en una forma más cómoda. Además, en caso de que se deseen imprimir las gráficas mostradas en pantalla, se puede detener la graficación en el momento que se considere conveniente, imprimir y posteriormente, reanudar el monitoreo del funcionamiento del sistema.

<F4> Archivo

La gráfica de la respuesta transitoria de la salida del sistema es una característica importante en todo sistema de control, razón por la cual, se decidió, no sólo graficarla y mantenerla visible en la pantalla, sino también almacenar en un archivo las muestras obtenidas y graficadas.

Esta opción <F4> del menú principal, permite observar, para un posible análisis, los datos del archivo que contiene las muestras de la respuesta del sistema durante los primeros 10 segundos después de encenderlo o reinicializarlo. A través del análisis de los datos en este archivo, es posible obtener de forma más precisa, datos tales como el sobrepaso, el tiempo de levantamiento, de asentamiento, de pico, etc.

<F10> Salir

Esta opción permite abandonar el SIU-PID y regresar al *prompt* del sistema operativo MS-DOS.

Pantalla de texto

En esta porción de la pantalla del SIU-PID se realizarán las siguientes acciones:

- Solicitud y adquisición de datos.
- Despliegue de mensajes de advertencia al usuario.
- Despliegue de mensajes sobre la acción que se está realizando.
- Despliegue del contenido del archivo mencionado anteriormente.

Gráfica de la respuesta transitoria

Esta gráfica se muestra en la parte superior derecha de la pantalla. Es una gráfica de velocidad en rpm, en función del tiempo en segundos, y nos permite observar el comportamiento del sistema de control en malla cerrada durante los primeros 10 segundos después de encender o reinicializar el sistema.

Con el fin de facilitar la lectura en cualquier punto de la gráfica, se dibujó una cuadrícula con línea punteada.

Gráfica en tiempo real

La gráfica en tiempo real del SIU-PID, es la mostrada en la parte inferior derecha de la pantalla, y permite tener un monitoreo continuo del funcionamiento del sistema de control, así como observar los transitorios ante cambios de velocidad en el momento exacto en que se presenten.

En esta gráfica, también se dibujó una cuadrícula para facilitar la lectura, y se visualizan tanto la última muestra (una pequeña línea roja), como las muestras tomadas durante los últimos 10 segundos.

Monitor numérico de velocidad

En la parte central, entre las dos gráficas descritas anteriormente, se muestra el valor numérico de la velocidad actual del motor en rpm. Este monitor numérico tiene la finalidad de facilitar aún más el monitoreo del sistema.

VIII. PRUEBAS Y RESULTADOS

8.1 PRESENTACION FINAL DEL SISTEMA DE CONTROL

Con el fin de facilitar el manejo del sistema de control, éste se dividió en 7 módulos:

1. Tarjeta del Sistema Multiprocesador

2. Tarjeta de Interfaz (Interfaz de Entrada y Etapa 1 de la Interfaz de Salida)

3. Tarjeta de la Etapa 2 de la Interfaz de Salida

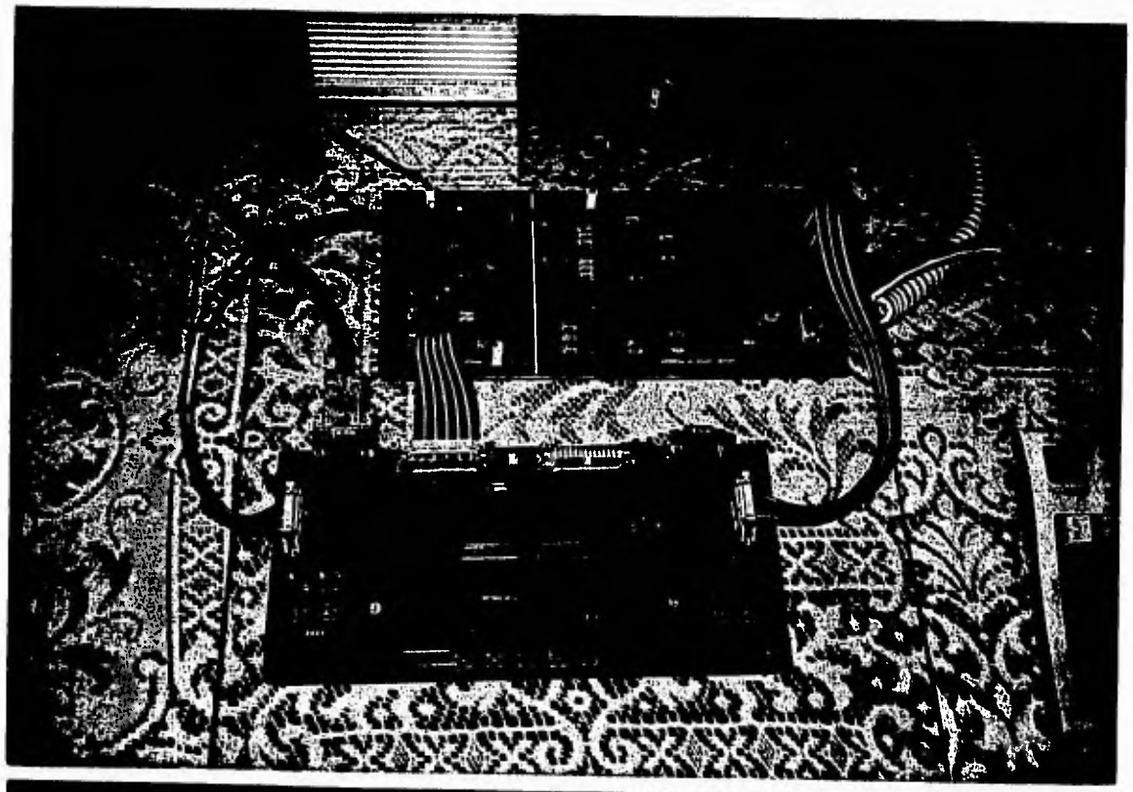
4. Motor de CD y tacogenerador

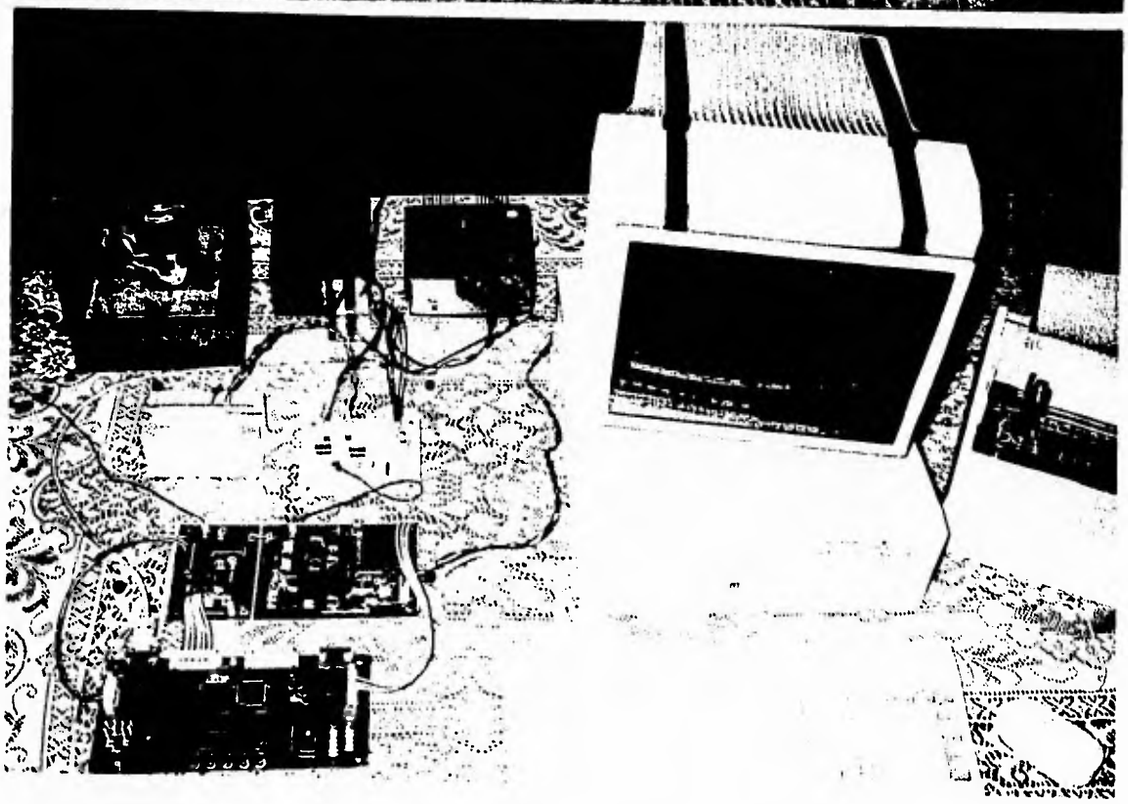
5. Fuente múltiple

6. Fuente de 18V

7. Computadora Personal

En las fotografías siguientes se puede apreciar la colocación y conexión final del sistema.





8.2 MANEJO DEL SISTEMA

8.2.1 Encendido, apagado y reinicialización

Siempre que se esté controlando la velocidad de un motor, es conveniente llevarlo a un estado de inicialización en su velocidad. Esta inicialización hace posible que la corriente en los embobinados del motor, no se eleve demasiado. Existen motores en los que debido a la gran inercia del rotor, este proceso de inicialización resulte indispensable.

El controlador digital que construímos, realiza esta inicialización de la velocidad del motor en forma automática, al momento de inicializar el sistema multiprocesador. Por ello, el procedimiento de encendido del sistema de control es el siguiente:

- Encender la fuente de 18V
- Encender la fuente múltiple.

En el momento en que el usuario así lo requiera, tendrá acceso al SIU-PID para monitorear el comportamiento del sistema o ejecutar cualquiera de las opciones del programa. Sin embargo, si se desea observar y analizar la respuesta transitoria del sistema de control, por medio del SIU-PID, es necesario que antes de encender o reinicializar el sistema, el programa SIU-PID haya sido ejecutado. Para mayores detalles acerca del acceso al SIU-PID, consultar el subtema 7.5.2.

Por otra parte, el procedimiento de apagado, es el siguiente:

- Apagar la fuente de 18V
- Oprimir IRQ del MCM
- Apagar la fuente múltiple

Para reinicializar el sistema:

Opción 1:

- Oprimir RESET

Opción 2:

- Oprimir IRQ
  - Apagar la fuente múltiple
  - Encender la fuente múltiple

La interrupción IRQ en el MCM, como ya se mencionó en el capítulo VII, permite deshabilitar la escritura a la EEPROM y evitar con esto posibles desconfiguraciones de los parámetros o la velocidad en el sistema de control.

8.2 PRUEBAS REALIZADAS

A continuación se presentan algunas gráficas de la respuesta del sistema en malla cerrada, en las cuales se pueden apreciar los cambios en las curvas de velocidad del motor al variar los parámetros K_p, T_d y T_i.

En el capítulo V "Modelado matemático del sistema", se hicieron simulaciones con el programa CC. Estas simulaciones se realizaron para los parámetros que se obtuvieron por el método de Ziegler y Nichols y también para los parámetros que finalmente fueron programados en el controlador. Ahora, se presentan aquí, las gráficas de respuesta que se obtuvieron en el SIU-PID para los mismos parámetros. La gráfica que se obtuvo cuando se programaron los parámetros entregados por el método de Ziegler y Nichols se muestra en la figura 8.1. La gráfica con los parámetros finales, se muestra en la figura 8.2.

Según se explicó anteriormente, el controlador digital que construímos ofrece al usuario la posibilidad de cambiar y ajustar los parámetros del controlador hasta que la respuesta del sistema le satisfaga por completo. Con el fin de visualizar como se refleja en la respuesta del sistema, un ajuste en los parámetros K_p, T_i y T_d, a continuación presentamos un conjunto de gráficas de respuesta del sistema. Para la realización de estas gráficas, se tomaron como base los parámetros con los que se realizó la gráfica de la figura 8.2. (Kp=1.5, Td=0.1, Ti=0.7), y posteriormente se fueron realizando ajustes en un parámetro a la vez, dejando a los dos restantes sin cambio.

En la figura 8.3 se muestra como al aumentar Kp (Kp=1.8), aumenta la velocidad de respuesta, pero también aumenta el sobrepaso.

En la figura 8.4 se aprecia como al disminuir Kp (Kp=0.5), el sistema responde en forma mas lenta, pero el sobrepaso disminuye (el sistema se vuelve sobreamortiguado).

En la figura 8.5 se muestra como al aumentar Td (Td=0.25), aumenta la amortiguación, y disminuye la velocidad de respuesta.

En la figura 8.6 se aprecia como al disminuir Td (Td=0.05), disminuye la amortiguación, por lo tanto aumenta el sobrepaso.

En la figura 8.7 se muestra como al aumentar Ti (Ti=2), disminuye la velocidad de respuesta, pero mejora la respuesta en estado estable.

En la figura 8.8 se aprecia como al disminuir Ti (Ti=0.2), la velocidad de respuesta del sistema aumenta, pero consecuentemente el tiempo de asentamiento también aumenta.

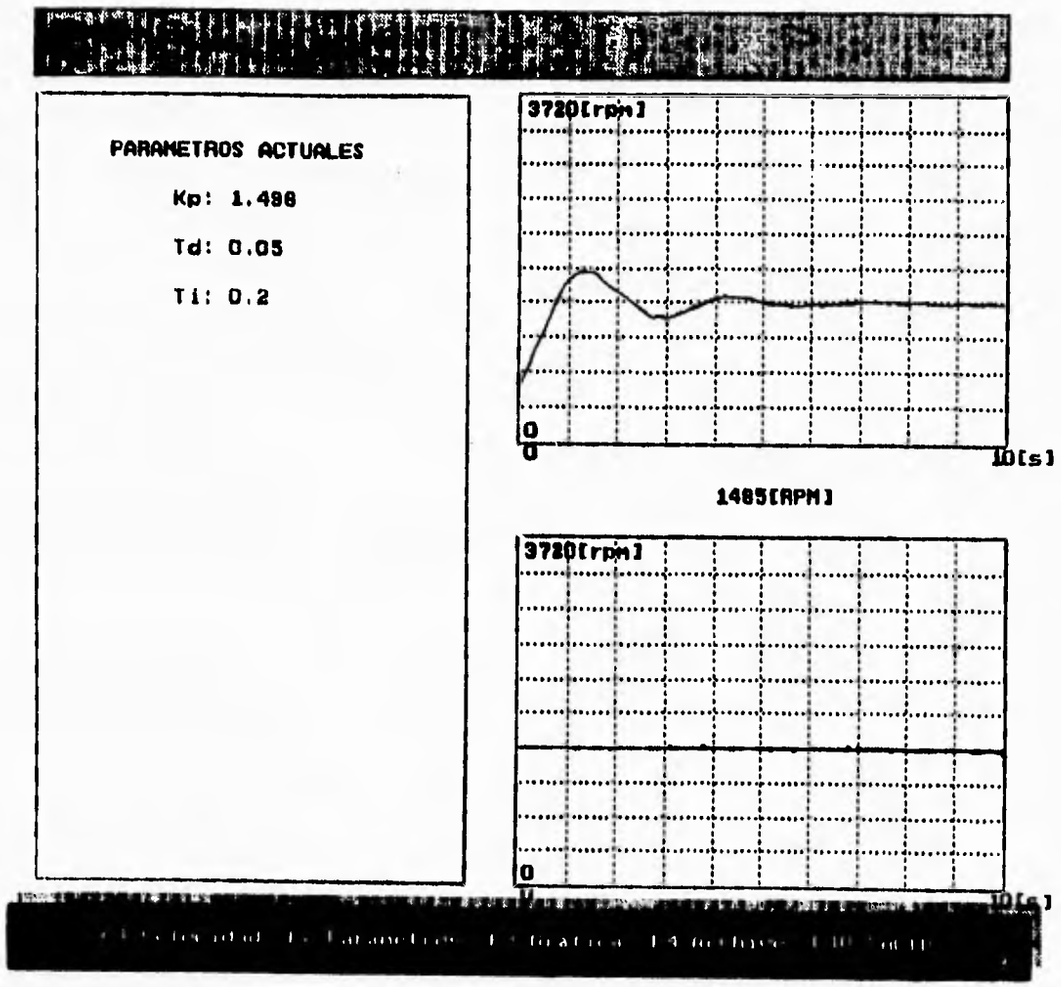

PARAMETROS ACTUALES
Kp: 1.498
Td: 0.05
Ti: 0.2
3720[rpm]
0
0
10[s]
1485[RPM]
3720[rpm]
0
10[s]

Figura 8.1

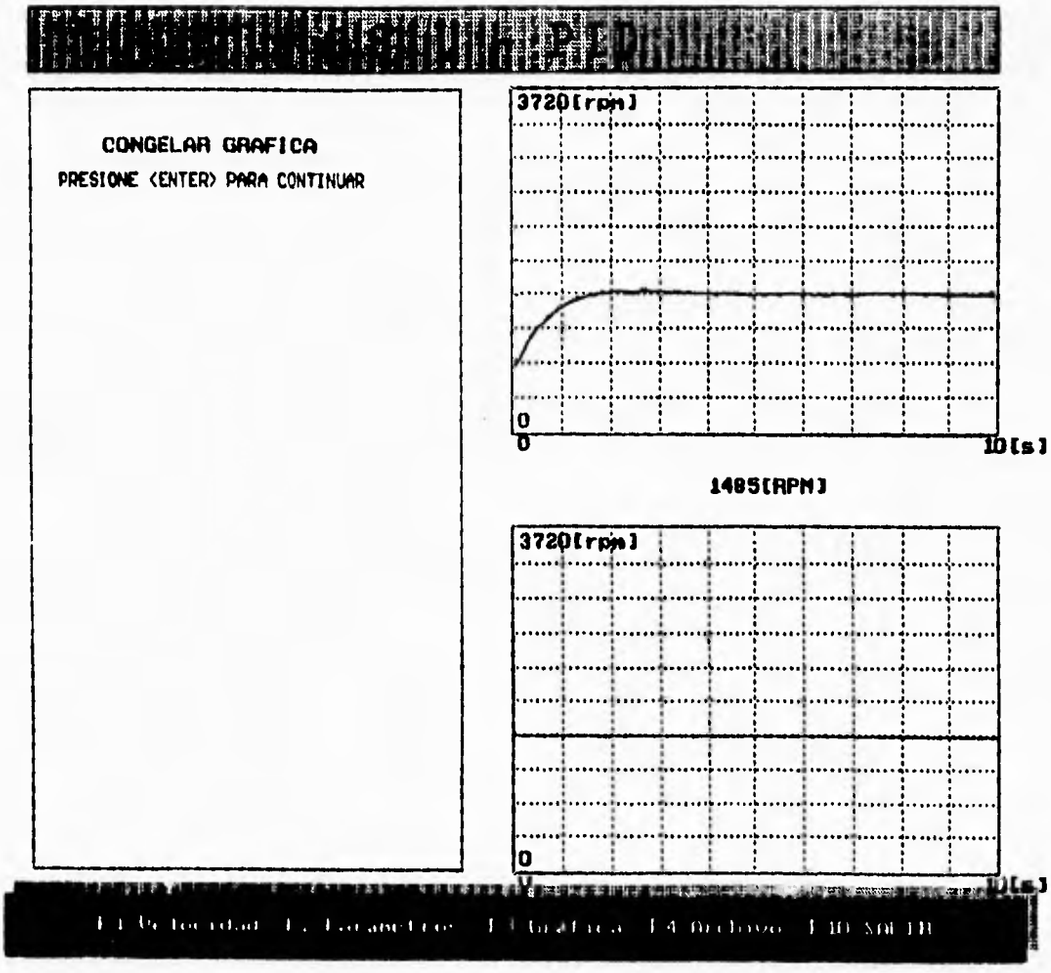

CONGELAR GRAFICA
PRESIONE <ENTER> PARA CONTINUAR
3720[rpm]
0
0
10[s]
1485[RPM]
3720[rpm]
0
10[s]

Figura 8.2

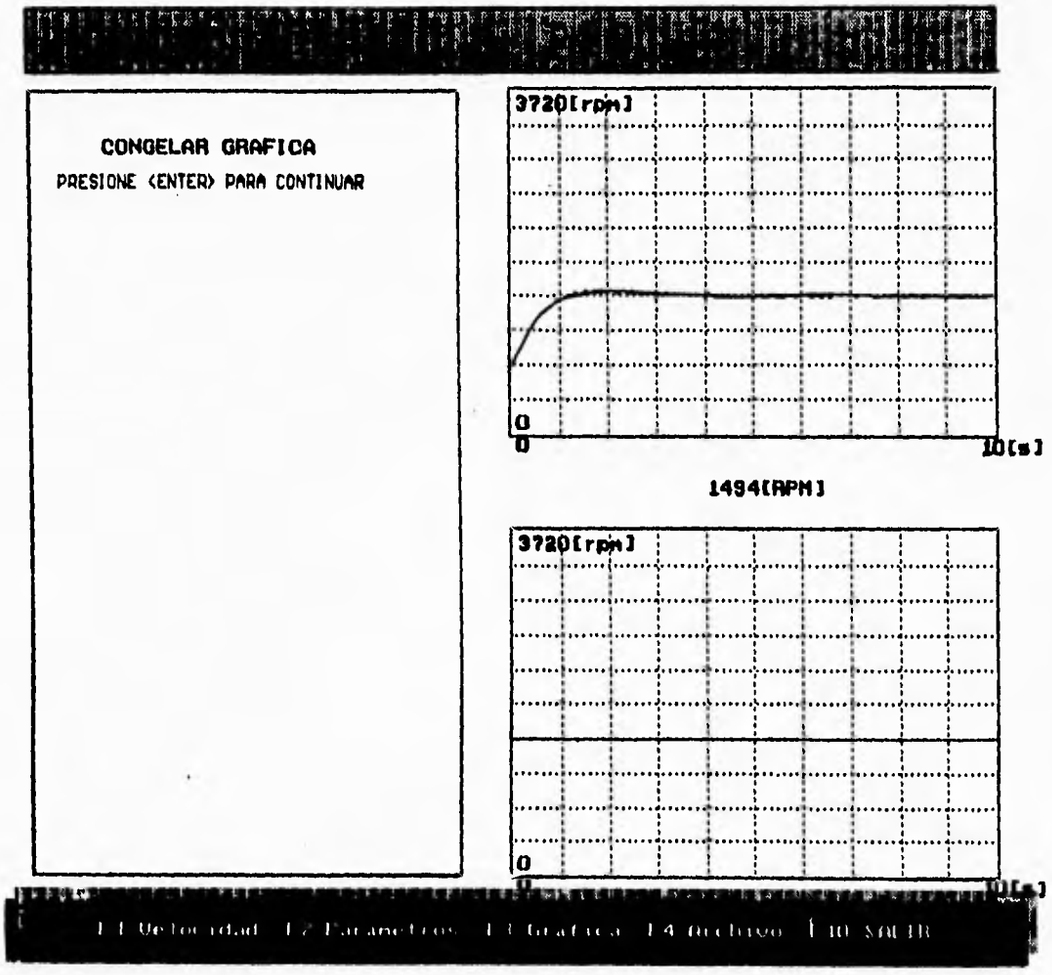

CONGELAR GRAFICA
PRESIONE <ENTER> PARA CONTINUAR
3720[rpm]
0
0
10[s]
1494[RPM]
3720[rpm]
0
0
10[s]

Figura 8.3

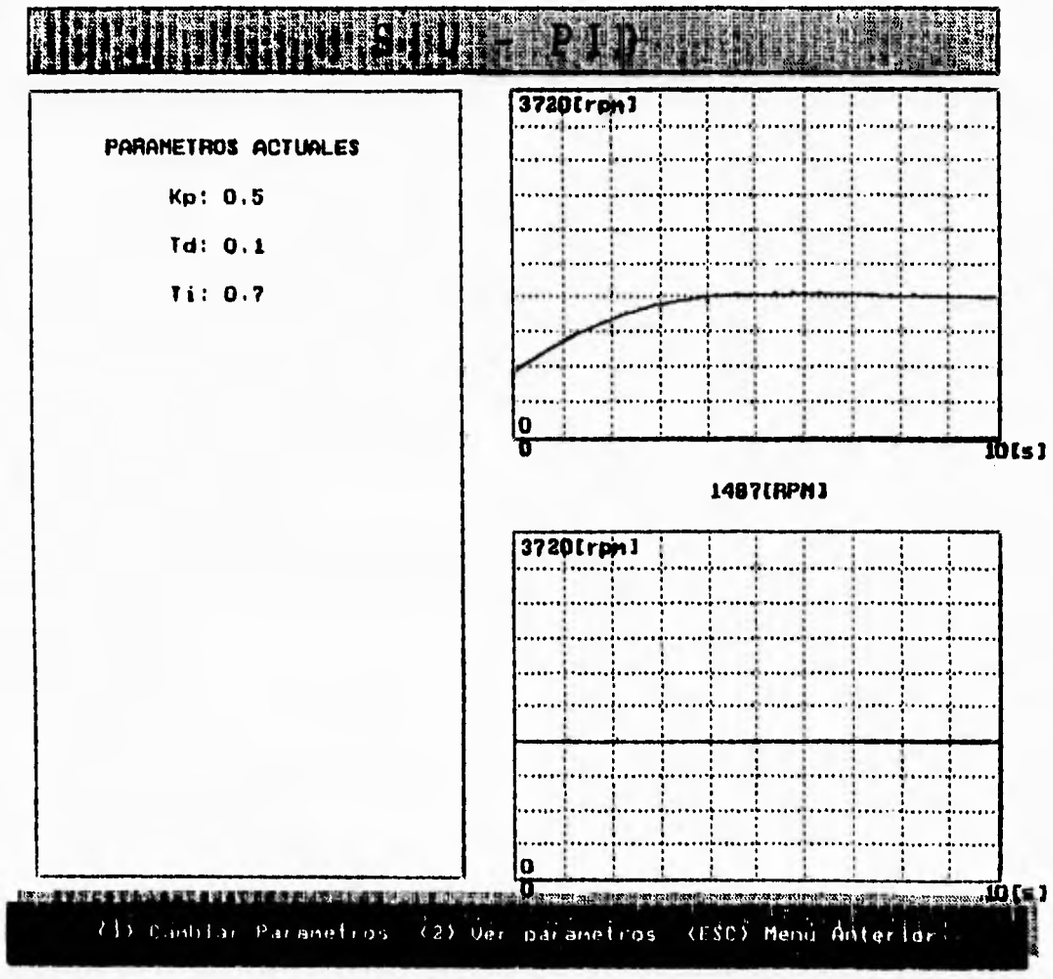

PARAMETROS ACTUALES
Kp: 0.5
Td: 0.1
Ti: 0.7
3720[rpm]
0
0
10[s]
1487[RPM]
3720[rpm]
0
0
10[s]
(1) Cambiar Parametros (2) Ver parametros (ESC) Menu Anterior

Figura 8.4

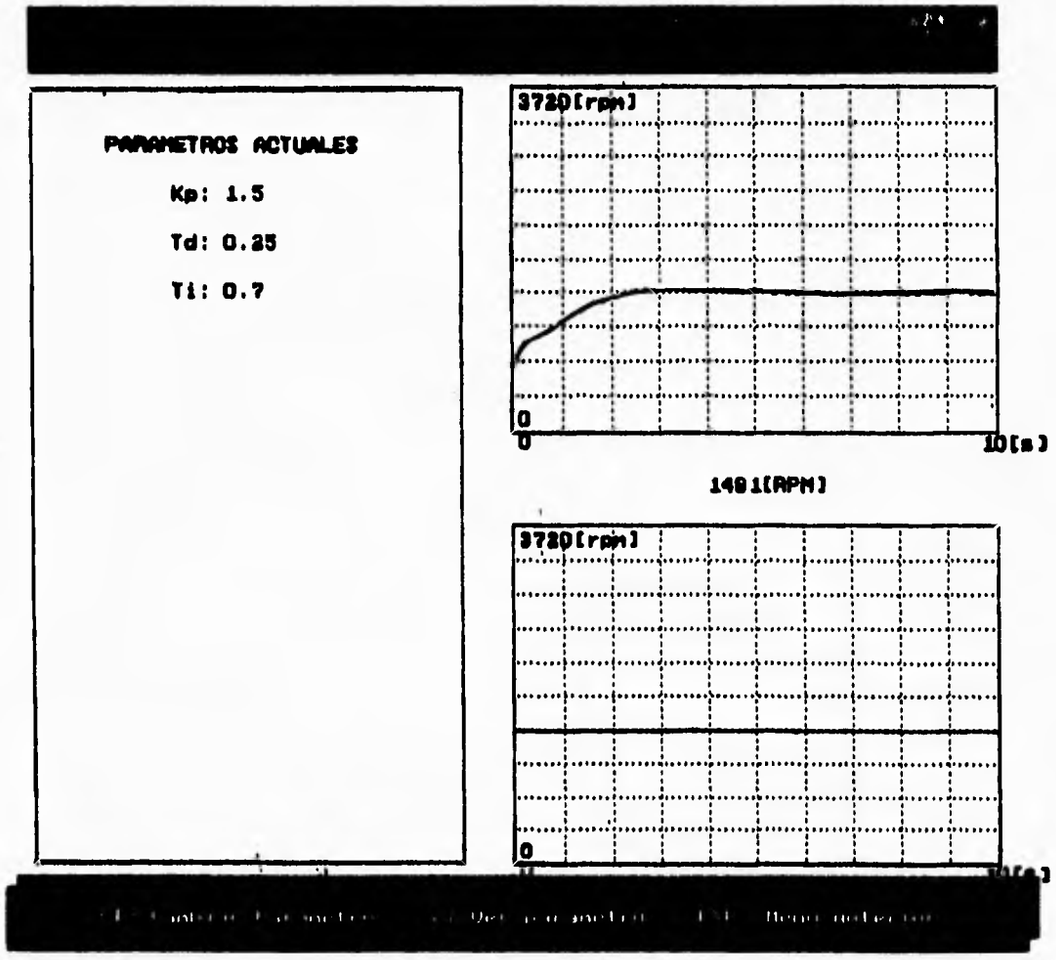

PARAMETROS ACTUALES
Kp: 1.5
Td: 0.25
Ti: 0.7
3720[rpm]
0
0
10[s]
1491[RPM]
3720[rpm]
0

Figura 8.5

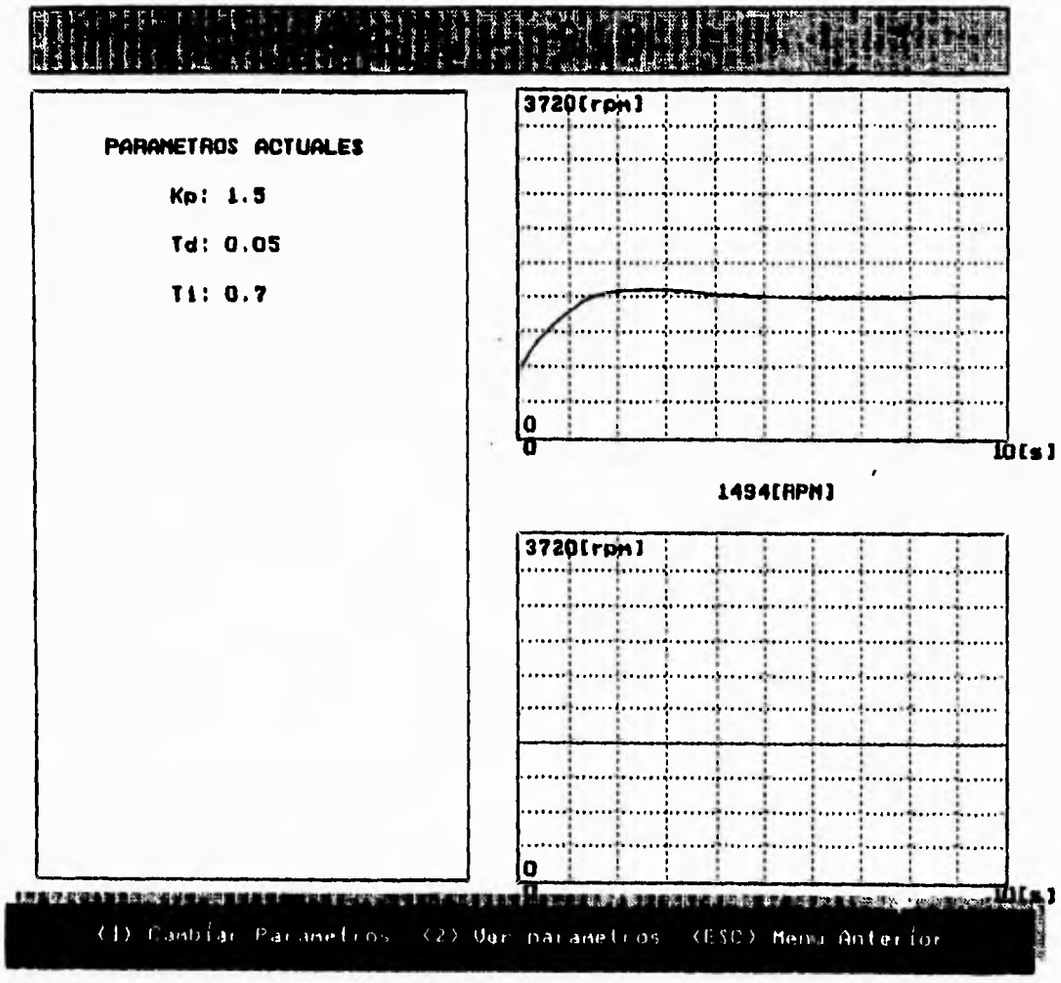

PARAMETROS ACTUALES
Kp: 1.5
Td: 0.05
Ti: 0.7
3720[rpm]
0
0
10[s]
1494[RPM]
3720[rpm]
0
0
10[s]
(1) Cambiar Parametros (2) Ver parametros (ESC) Menu Anterior

Figura 8.6

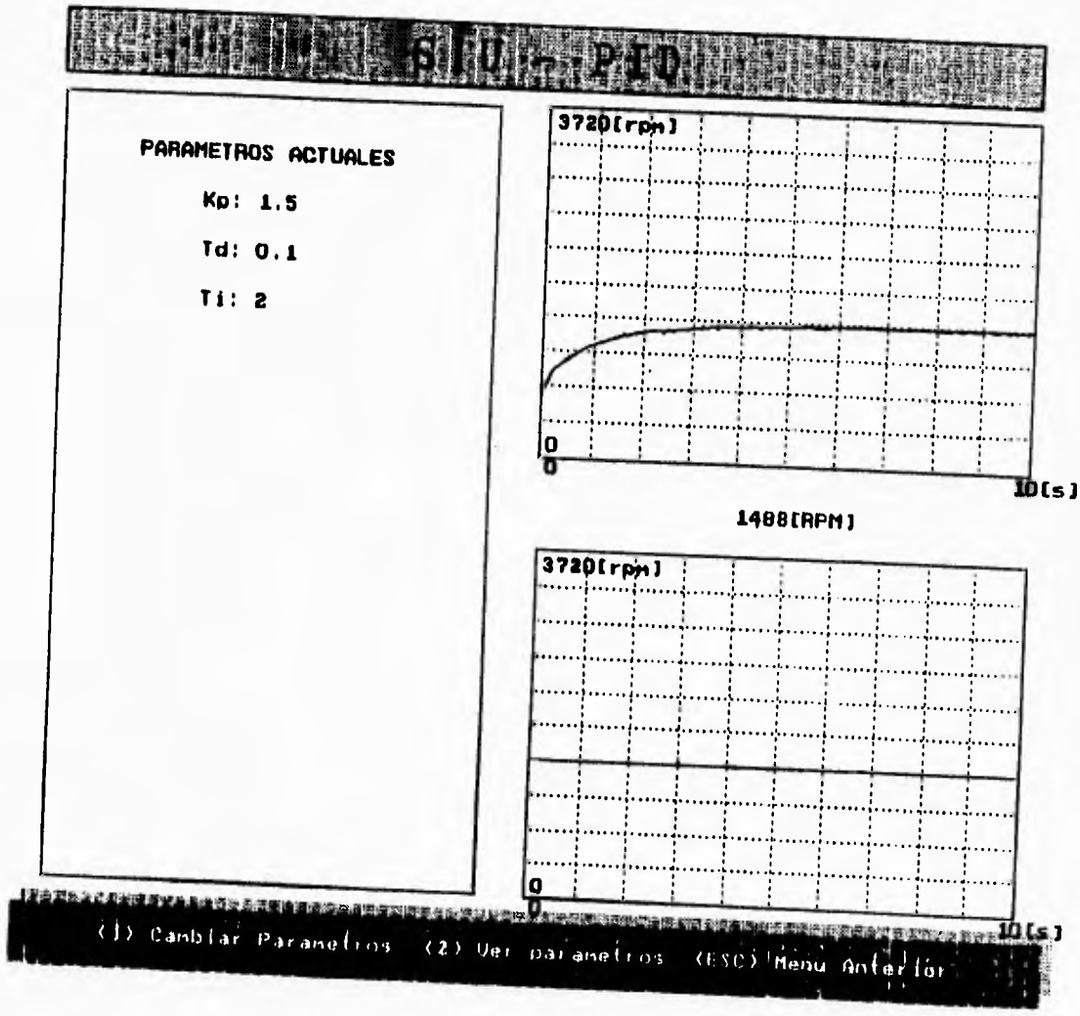

PARAMETROS ACTUALES
Kp: 1.5
Td: 0.1
Ti: 2
3720[rpm]
0
0
10[s]
1488[RPM]
3720[rpm]
0
0
10[s]
(1) Cambiar Parametros
(2) Ver parametros
(ESC) Menu Anterior

Figura 8.7

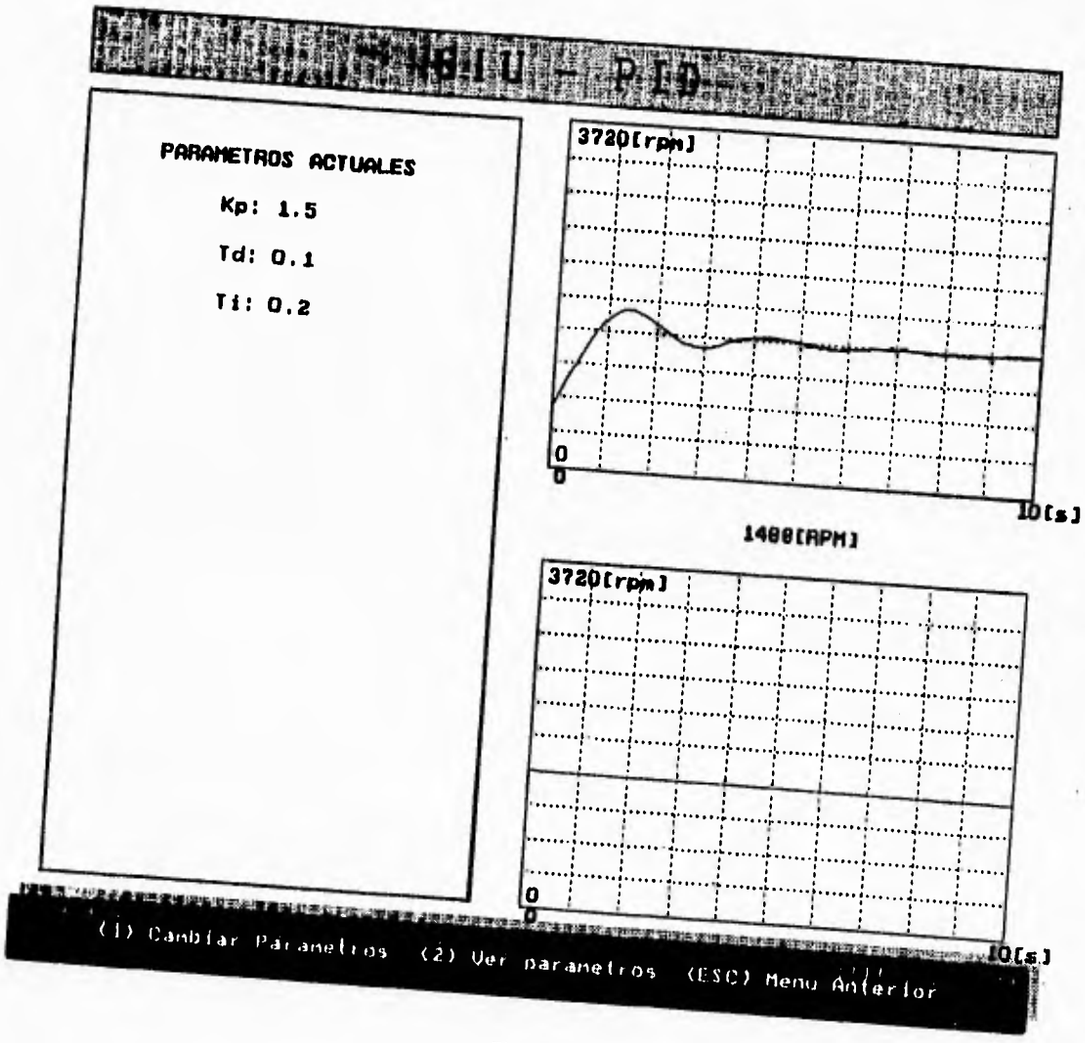

- PID -
PARAMETROS ACTUALES
Kp: 1.5
Td: 0.1
Ti: 0.2
3720[rpm]
0
0
10[s]
1400[RPM]
3720[rpm]
0
0
10[s]
(1) Cambiar Parametros (2) Ver parametros (ESC) Menu Anterior

Figura 8.8

8.3 ANALISIS DE RESULTADOS

De acuerdo con las pruebas realizadas, podemos decir, que el controlador PID diseñado y construído con el objetivo de controlar la velocidad de un motor de CD, posee las características a las que aspiraba, pues es preciso, modular, fácil de utilizar, permite el monitoreo, pero su mayor ventaja es que cumple con las especificaciones indicadas por el usuario.

Como se puede apreciar, las gráficas que muestran la respuesta del sistema, son muy parecidas a las gráficas que se obtuvieron de las simulaciones con el programa CC. Es necesario advertir que las diferencias que existen, son debidas, básicamente, a las discrepancias entre el modelo matemático del motor con el que se realizaron las simulaciones y el comportamiento verdadero del mismo. Sin embargo, la ventaja de un controlador digital como este, sobre los controladores analógicos, consiste en la facilidad con la que el desempeño del controlador puede ajustarse, mediante un cambio en los parámetros de control K_p, T_d y T_i. Es por ello que las ligeras diferencias, de lo simulado, con lo real, no constituyen un factor que opaque los resultados del controlador.

Para todas las gráficas mostradas en las figuras anteriores, se dió una velocidad de referencia de 1488 rpm, sin embargo, el valor de velocidad instantánea que aparece en el monitor numérico de velocidad, fluctúa alrededor de este valor ±6 rpm. Este error se mantiene para cualquier velocidad, de forma que el error relativo máximo se presenta cuando se pretende mantener una velocidad de referencia de 800 rpm, que corresponde, a la mínima velocidad que el SIU-PID, por las características de funcionamiento del sistema de control, permite introducir. Resultando un error de 0.75%.

8.4 POSIBLES MEJORAS

A continuación se mencionan algunas de las posibles mejoras del sistema de control construído:

1. Este proyecto constituye únicamente un prototipo, pero dado su diseño y construcción modular,

sería posible construir versiones integrales, colocando las tarjetas en un solo gabinete, optimizando espacio y minimizando conexiones.

2. Como ya se mencionó, la construcción de un *chopper* como etapa de potencia del circuito de interfaz de salida, trae consigo una mejora en cuanto al ahorro de energía, pero aumenta el tiempo de asentamiento y disminuye el rango de control preciso de la velocidad.

CONCLUSIONES

A continuación se mencionan en forma breve las conclusiones a las que llegamos al finalizar el presente proyecto:

1. Diseñamos y construimos un sistema de control de velocidad para un motor de CD, funcional, modular, preciso y de fácil manejo. Su característica de modularidad, permite el desarrollo de nuevas versiones de los diferentes módulos del sistema, para futuras aplicaciones.

2. Dado que el motor a controlar en este proyecto, es un motor con excitación independiente y por lo tanto, su modelo matemático es lineal, fue posible aplicar en forma directa todos los conceptos de la teoría de control digital. En caso de que el modelo matemático del motor a controlar hubiese sido no lineal, el controlador PID que construimos, podría controlar la velocidad del motor únicamente alrededor de un punto de operación, sin alterar las características dinámicas obtenidas al programar los parámetros K_p, T_d y T_i. En caso de que se requiera que el motor opere alrededor de un punto de operación diferente, sería necesario sintonizar el controlador para ese nuevo punto de operación, logrando con ello un control de velocidad en un rango diferente.

3. Con el uso de control digital logramos gran versatilidad en la implantación del algoritmo de control, así como facilidad en el proceso de sintonización.

4. Con el desarrollo de este proyecto pudimos observar muchas de las características más importantes de la retroalimentación, entre las que se encuentran las siguientes: permite estabilizar sistemas, mejora la respuesta en estado estacionario, facilita el rechazo a perturbaciones y disminuye la sensibilidad a las variaciones de parámetros.

5. Creemos que es conveniente mencionar que el método de Ziegler y Nichols arroja parámetros iniciales de sintonización, que son de gran utilidad, pero cuando se tiene una curva de respuesta de segundo orden sobreamortiguada que comienza a parecerse a una respuesta de primer orden, no son muy precisas las mediciones sobre la curva de respuesta y por ello este método sólo proporciona una base de la cual partir para sintonizar adecuadamente el sistema. Debido a esto es conveniente y muchas veces

necesario, realizar simulaciones hasta llegar a la respuesta deseada.

6. El sensor de velocidad utilizado es muy confiable y preciso, sin embargo, la falta de linealidad de éste a bajas velocidades nos ocasionó que no podamos tener un buen control a velocidades bajas (menores de 800 rpm). No obstante, después de analizar otro tipo de transductor de velocidad, (capitulo 5), hemos llegado a la conclusión de que es difícil tener un solo transductor de velocidad que opere en forma lineal en un rango tan amplio. Debido a esto hemos pensado en que si también se requiriera tener control a bajas velocidades, lo mejor sería implementar otro transductor y agregarlo al sistema, con el fin de tener un sistema transductor que opere linealmente en todo el rango de control.

7. En el capítulo 5 se planteó la posibilidad de utilizar un *chopper* como etapa de potencia de la interfaz de salida. El uso de un *chopper*, controlado por ancho de pulso, para llevar un voltaje de control a las terminales de un motor, cuya fuerza contraelectromotriz es elevada, debe de tener un sistema para compensar el voltaje en las terminales del motor debido a la fuerza contraelectromotriz. La fuerza contraelectromotriz, está presente en todo momento, aún en los instantes en que no conduce el TMOS o tiristor empleado para el *chopper*, y esto trae consigo una no linealidad de la velocidad, con respecto al ciclo de trabajo de la señal modulada por ancho de pulso utilizada para el *chopper*.

8. Una característica fundamental e innovadora en este proyecto es el uso del sistema multiprocesador, el cual constituye el módulo fundamental en el sistema de control. El uso de esta arquitectura, nos permitió implementar el controlador PID empleando microcontroladores de bajo costo y fácil adquisición, ya que de no haber empleado este sistema multiprocesador, hubiese sido necesario utilizar dispositivos tales como circuitos DSP, los cuales resultan muy difíciles de adquirir.

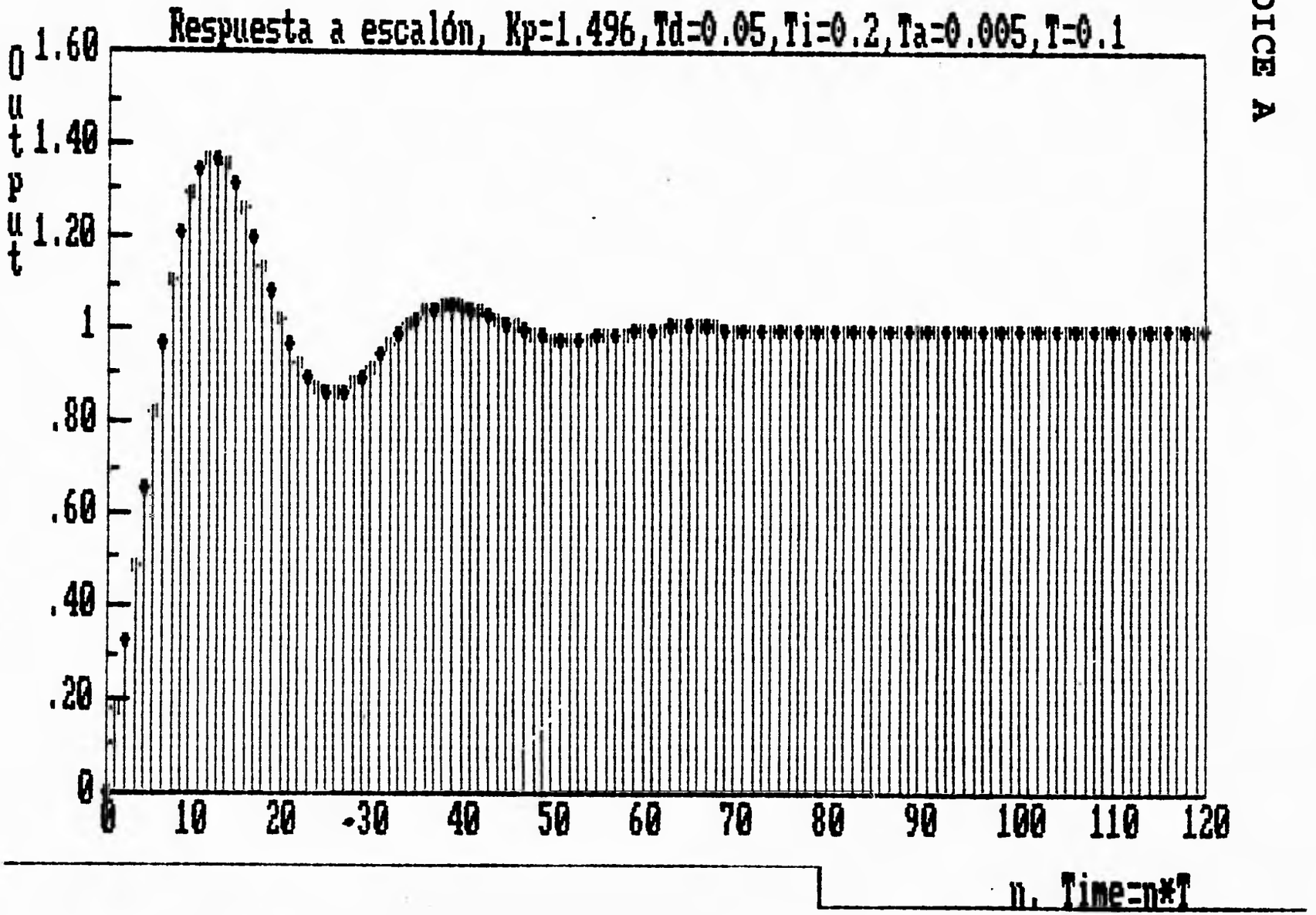
9. Es conveniente recalcar que para la solución del algoritmo de control se empleó el paquete de rutinas de punto flotante proporcionado por Motorola, el cual permite dar soluciones de alta precisión, pero en un tiempo de procesamiento elevado. Esta fue otra de las razones por las que el uso del sistema multiprocesador resultó necesario.

10. Una de las ventajas más sobresalientes del sistema de control construido, es que permite, a través del SIU-PID, una programación de los parámetros K_p, T_d y T_i en cualquier momento, dando al usuario la capacidad de modificar estos parámetros hasta obtener la respuesta deseada.

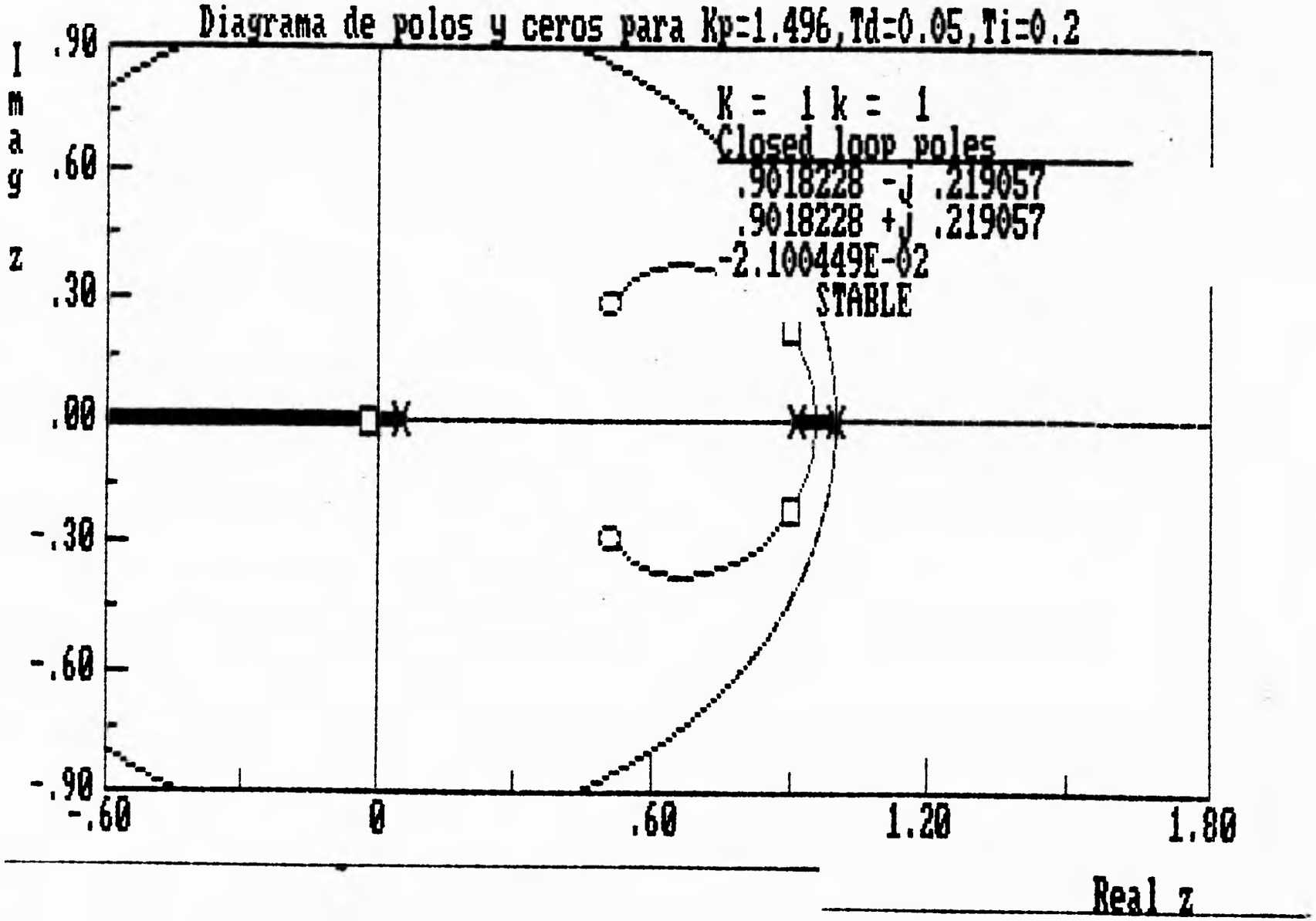
11. La confiabilidad del equipo demostró ser alta, pues después de varias pruebas de uso continuo conservó su desempeño normal. No hubo calentamiento excesivo, ni deterioro de componentes.

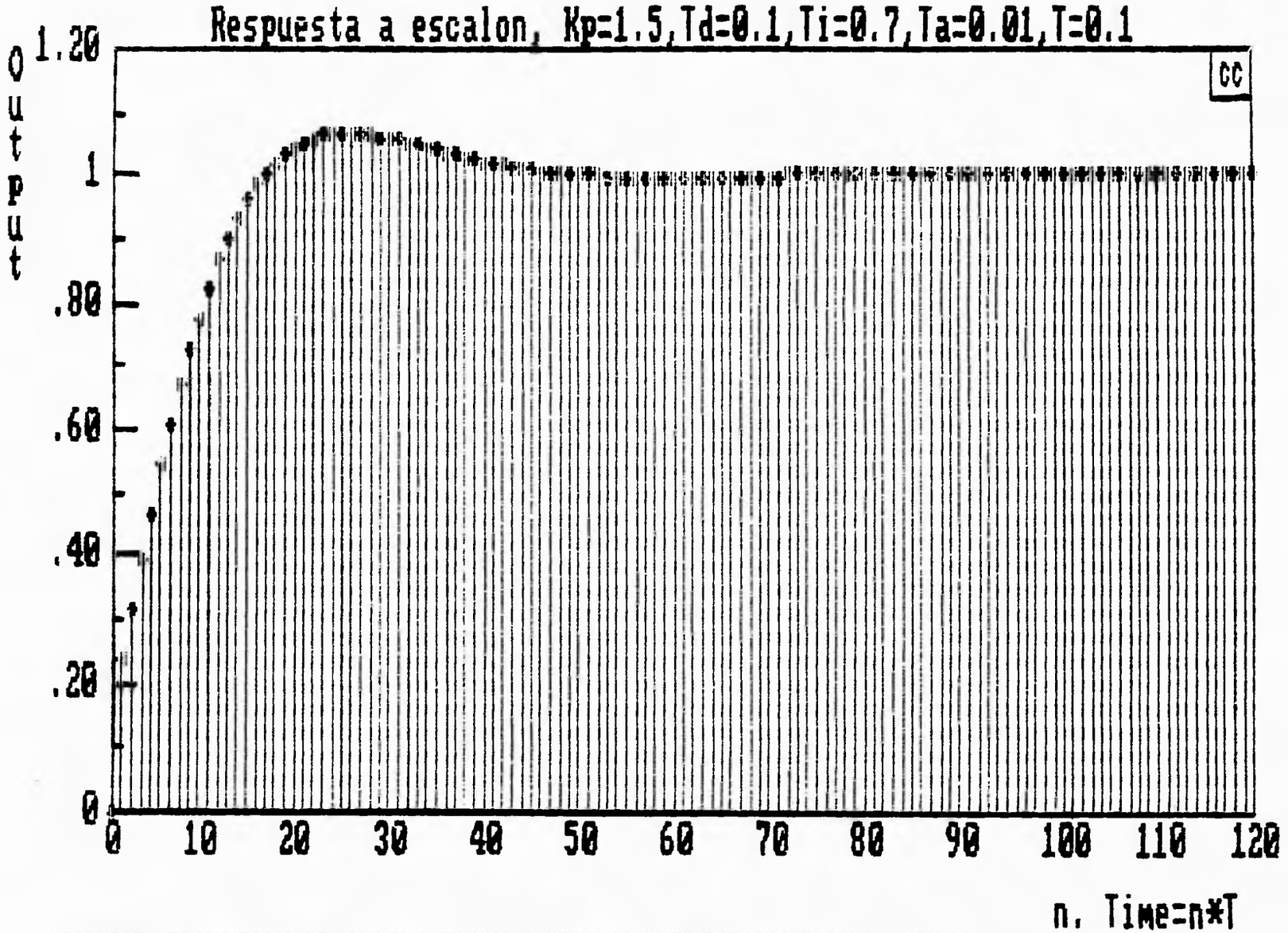
12. Finalmente, podemos decir que el desarrollo del presente proyecto fue una gran experiencia de trabajo interdisciplinario, en el que se combinaron los conocimientos que adquirimos a lo largo de nuestra formación profesional y que ahora se ven cristalizados.

RESPUESTA TRANSITORIA DEL SISTEMA EN MALLA CERRADA ANTE ENTRADA ESCALON, CON LOS PARAMETROS OBTENIDOS POR EL METODO DE ZIEGLER Y NICHOLS

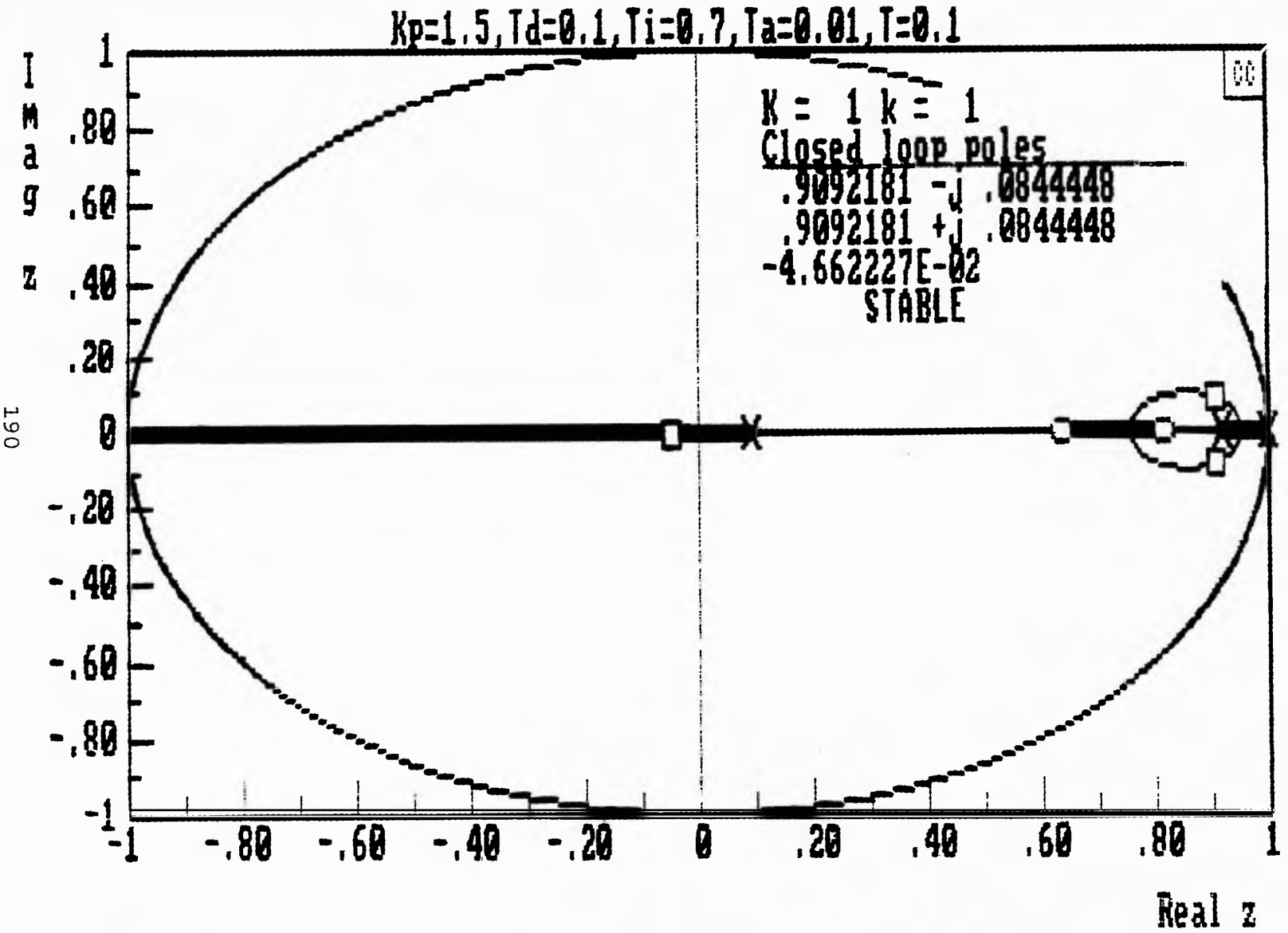


LUGAR GEOMETRICO DE LAS RAICES, CON LOS PARAMETROS OBTENIDOS POR EL METODO DE ZIEGLER Y NICHOLS





RESPUESTA TRANSITORIA DEL SISTEMA EN MALLA CERRADA ANTE ENTRADA ESCALON, CON LOS PARAMETROS OBTENIDOS DESPUES DE LA SINTONIZACION



LUGAR GEOMETRICO DE LAS RAICES, CON LOS PARAMETROS OBTENIDOS DESPUES DE LA SINTONIZACION

APENDICE B

Este apéndice contiene los listados de los programas y las subrutinas asociadas con ellos para el Microcontrolador Maestro (MCM) y para el Microcontrolador Esclavo (MCE)

Tanto para el MCM como para el MCE, se emplearon las siguientes etiquetas a lo largo de los programas.

```
PORTA     EQU     $00   ;I/O Port A
DDRA      EQU     $01   ;DDRA
PORTG     EQU     $02   ;Parallel I/O Control Register
DDRG      EQU     $03   ;Data Direction Register for Port G
PORTE     EQU     $04   ;I/O Port E
PORTF     EQU     $05   ;Alternate Latched Port C
PORTC     EQU     $06   ;I/O Port C
DDRC      EQU     $07   ;Data Direction for Port C
PORTD     EQU     $08   ;I/O Port D
DDRD      EQU     $09   ;Data Direction for Port D
PORTE     EQU     $0A   ;I/O Port E
TCNT      EQU     $0E   ;Timer Counter Register
TOC2      EQU     $18   ;Output Compare 2 Register
TMSK1     EQU     $22   ;Main Timer Interrupt Mask Register 1
TFLG1     EQU     $23   ;Main Timer Interrupt Flag Register 1
TMSK2     EQU     $24   ;Main Timer Interrupt Mask Register 2
TFLG2     EQU     $25   ;Main Timer Interrupt Flag Register 2
SPCR      EQU     $28   ;SPI Control Register
SPSR      EQU     $29   ;SPI Status Register
SPDR      EQU     $2A   ;SPI Data In/Out
BAUD      EQU     $2B   ;SCI Baud Rate Control
SCCR1     EQU     $2C   ;SCI Control Register 1
SCCR2     EQU     $2D   ;SCI Control Register 2
SCSR      EQU     $2E   ;SCI Status Register
SCDR      EQU     $2F   ;SCI Data (Read RDR, Write TDR)
BPROT     EQU     $35   ;Block Protect
OPTION    EQU     $39   ;System Configuration Options
PPROG     EQU     $3B   ;EEPROM Programming Control Register
CONFIG    EQU     $3F   ;COP, ROM & EEPROM Enables
FPACC1EX  EQU     $0000
FPACC1MN  EQU     $0001
MANTSGN1  EQU     $0004
FPACC2EX  EQU     $0005
FPACC2MN  EQU     $0006
MANTSGN2  EQU     $0009
PWR10EXP  EQU     $0001
EXPSIGN   EQU     $0000
MULTIA    EQU     $0050
MULTIB    EQU     $0052 ; Usado por la subrutina para obtener el valor de la muestra
RESA      EQU     $0053
RESB      EQU     $0054
REST      EQU     $003C ; Usado por la subrutina para obtener el valor de la muestra
```

Para el Microcontrolador Maestro:

```
*****************************************************************************
*                          PROGRAMA PRINCIPAL                               *
*****************************************************************************

          ORG  $CE00      ; Inicio de programa principal
          LDS  #$03FF     ; Tope del stack
          LDX  #$1000     ; Cargar X con $1000 para direccionamiento indexado
          BCLR  BPROT,X,$07   ; Habilitar escritura a la EEPROM,protegiendo config y vectores de
                              ; reset e interrupcion.
          BSET TMSK2,X,$01 ;PR1=0,PR0=1 para controlar el preescalador del temporizador principal
                           ; de forma que el contador de 16 bits se incremente cada 2 us y sea
                             ; posible generar una base de tiempo de 100 ms
          BSET OPTION,X,$20 ;
          CLR  DDRA,X     ; Declaración del puerto A como entrada
          BSET DDRG,X,$03  ; Declaración de PG0 y PG1 como salida
          jsr maestro  ; se declara como maestro.
          LDAA #$10    ; ----------------------------------------------
          STAA $0100   ; Contador auxiliar para enviar 16 bytes del MCM al MCE
          LDY  #$FE21  ; Carga Y con la dirección inicial de los datos D y F
PRIN2:    LDAB $00,Y   ; Carga B con el primer byte a enviar
          STAB SPDR,X  ; Carga B al SPDR y comienza la transmisión
PRIN3:    BRCLR SPSR,X,$80,PRIN3 ; Permanece en este ciclo hasta que finalice la transmisión
          LDAA  SPDR,X ; Acceso al SPDR para borrar la bandera SPIF del registro SPSR cuando se
                       ; ha completado la transmisión del byte en cuestion
          INY
          DEC  $0100
          BNE  PRIN2   ; Sigue transmitiendo hasta terminar con los 16 bytes que corresponden a
                       ; los datos D y F
          JSR  INICIA  ; Inicialización de las variables en RAM
          LDY #$FFFF   ;-----------------------------------------------
          STY $0030    ; Dirección auxiliar para retardo
```

```
            LDAA #$14     ; 14h = 20d   5 seg aproximadamente
PRIN1:      JSR  RETARDO  ; Durante este tiempo el motor permanece a una
            DBCA          ; velocidad constante que corresponde al estado de
            BNE PRIN1     ; inicialización ----------------------------------
            LDAA #$40
            STAA TMSK1,X    ; Habilitación de OC2I
            STAA TFLG1,X    ; Clarear la bandera OC2F
            LDAA #$30
            STAA BAUD,X     ; Baudaje de 9600 para comunicación por SCI
            LDAA #$2C
            STAA SCCR2,X    ; RIE -> 1, TE -> 1, RE -> 1
            LDY SCSR,X  ;envia un $cc como protocolo con la PC
            LDAA #$CC
            STAA SCDR,X  ;--------------------------------
            LDAA #$AA     ; Envía un $AA para indicarle a la PC que grafique en
PRIN4:      BRCLR SCSR,X,$80,PRIN4
            STAA SCDR,X   ; la grafica de respuesta dinámia.
            LDAA #$AA
PRIN5:      BRCLR SCSR,X,$80,PRIN5
            NOP
            NOP
            NOP
            NOP
            NOP
            NOP
            STAA SCDR,X
ESP:        BRCLR SCSR,X,$80,ESP  ;40
            CLR $150
            CLR $151
            JSR  MUESTRA  ; Sirve para inicializar la $18C-$18D para que la primer muestra a ser
                          ; enviada a la PC sea válida
            CLI           ; Habilitación general de interrupciones
HERE:       LDAA #$77
            CMPA $150     ; En caso de que en esta localidad haya un $77 indica
            BEQ  GRABAP   ; que se hará cambio de parámetros y que estos serán grabados
            LDAA #$88     ; En caso de que en esta localidad haya un $88 indica
            CMPA $151     ; que se requiere grabar velocidad
            BEQ  GRABAV
            BRA  HERE     ; Permanece en este loop hasta la siguiente interrupcion
GRABAP:     JMP GRABAPAR
GRABAV:     JMP GRABAVEL

*******************************************************************************
*                          SUBRUTINA INICIA                                   *
*******************************************************************************
* Inicialización de variables en RAM
INICIA:     PSHX
            PSHY
            LDX  #$0074     ; e(k) en flotante en RAM
            JSR  FLTCERO    ; se inicializa con 0 flotante
            LDX  #$006F     ; e(k-1) en flotante en RAM
            JSR  FLTCERO    ; se inicializa con 0 en flotante
            LDX  #$006A     ; e(k-2) en flotante en RAM
            JSR  FLTCERO    ; se inicializa con 0 flotante
            LDX  #$FE19     ; r(k) en ASCII en la EEPROM
            JSR  ASCFLTP    ; conversión de ASCII a flotante
            LDX  #$0000     ; FPACC1
            LDY  #$0079     ; r(k) en flotante en RAM
            JSR  TRANS      ; FPACC1 -> r(k) en flotante en RAM
            LDX  #$FE01     ; A en ASCII en la EEPROM
            JSR  ASCFLTP    ; conversión de ASCII a flotante
            LDX  #$0000     ; FPACC1
            LDY  #$005B     ; A en flotante en RAM
            JSR  TRANS      ; FPACC1 -> A en flotante en RAM
            LDX  #$FE09     ; B en EEPROM en ASCII
            JSR  ASCFLTP    ; conversión de ASCII a flotante
            LDX  #$0000     ; FPACC1
            LDY  #$0060     ; B en flotante en RAM
            JSR  TRANS      ; FPACC1 -> B en flotante en RAM
            LDX  #$FE11     ; C en EEPROM en ASCII
            JSR  ASCFLTP    ; conversión de ASCII a flotante
            LDX  #$0000     ; FPACC1
            LDY  #$0065     ; C en flotante en RAM
            JSR  TRANS      ; FPACC1 -> C en flotante en RAM
            PULY
            PULX
            RTS             ; Retorno de subrutina
```

```
*****************************************************************************
*          Subrutina de servicio de interrupción por Temporizador           *
*****************************************************************************

RS_TIMER:  ORG  $D200     ; Inicio de la subrutina de servicio de interrup. por TIMER
           LDX  #$1000    ; Cargar X con $1000 para direccionamiento indexado
           LDAA #$02      ; ------------------------------------------------
           STAA $188      ; Envia por SCI la mestra [y(k)] en dos bytes que están
           LDY  #$18C     ; almacenados a partir de la $18C
           STY  $189
           JSR ENVIAR     ; ------------------------------------------------
           LDD  $FC1F     ; Cargar en D el offset para la siguiente comparación, el cual se
                          ; encuentra en la localidad $FC1F
           ADDD TOC2,X    ; Definir el nuevo valor de comparación, sumandole al valor actual de
                          ; TOC2, el offset
           STD  TOC2,X    ; Se sobreescribe este nuevo valor en TOC2 para la sig. comparación
           LDAA #$40
           STAA TPLG1,X  ;Deshabilita la presente interrupción para evitar que el stack se sature.
           LDAB #$CE      ;Cargar a B un #$CE (código de Comienza Esclavo)
           STAB SPDR,X     ; Este dato se carga en el registro SPDR (dato a enviar)
RST1:      BRCLR SPSR,X,$80,RST1  ; Se espera hasta que termine la transmisión
           LDAA SPDR,X    ; Borra la bandera SPIF para poder realizar la siguiente transmisión.
                          ; Esta bandera se borra con una lectura+un acceso
           CLI            ; A partir de aqui pueden ocurrir interrupciones
           JSR  MUESTRA   ; Toma la muestra [y(k)]
           JSR  CALCULA   ; Calculo del valor DATOM para enviar el resultado al esclavo
           LDAA #$05      ; Contador auxiliar para enviar los 5 bytes del DatoM en flotante
           STAA $0100     ; Almacena el contador en la dirección 0100
           LDX  #$0032    ; Carga X con la dirección inicial de donde se encuentra DatoM
           LDY  #$1000    ; Carga Y con 1000 para indexar
           SEI            ; Deshablita interrupciones para transmitir por SPI
RST3:      LDAB $00,X     ; Carga B con el dato a enviar, indexado por X
           STAB SPDR,Y     ; Carga B al SPDR y comienza la transmisión
RST2:      BRCLR  SPSR,Y,$80,RST2  ; Permanece en este loop hasta que finalice la transmisión
           LDAA SPDR,Y    ; Borra la bandera SPIF del registro SPSR cuando se ha completado
                          ; la transmisión
           INX            ; Incrementa X para que apunte al siguiente dato a enviar
           DEC  $0100     ; Decrementa el contador de datos enviados
           BNE RST3       ; Mientras no se hayan enviado los 5 bytes continúa enviando
FIN:       RTI ; Una vez enviado el DatoM, finaliza la subrutina de interrupción por TIMER

*****************************************************************************
*                          SUBRUTINA MAESTRO                                *
*****************************************************************************
* Esta subrutina inicializa al microcontrolador como Maestro para comunicación por SPI, es llamada
*por la subrutina de servicio de interrupción por TIMER

MAESTRO:   PSHX           ; Alamcena X en el stack
           PSHA           ; Almacena A en el stack
           PSHB           ; Almacena B en el stack
           LDX  #$1000    ; Carga X con $1000 para indexar
           CLR  $28,X     ; Limpia el registro SPCR para su posterior uso
           LDD  #$3A57
           STAA $09,X ; Carga 3A en DDRD para habilitar como salida la terminal SS, SCK, MOSI y
                      ; TDX y como entradas MISO y RXD.
           BCLR $08,X,$20 ; Carga un cero en SS para despertar al esclavo.
           STAB $28,X     ; Carga un 57 en SPCR para: SPIE,MSTR,CPHA,SPR1,SPR0->1
                          ; y SPIE,DWOM,CPOL->0
           PULB           ; Restaura el valor de B
           PULA           ; Restaura el valor de A
           PULX           ; Restaura el valor de X
           RTS            ; Retorno de subrutina

*****************************************************************************
*                          SUBRUTINA CALCULA                                *
*****************************************************************************

CALCULA:   PSHX
           PSHY
           PSHA
           PSHB
           LDX #$006F     ; e(k-1) en flotante en RAM
           LDY #$006A     ; e(k-2) en flotante en RAM
           JSR TRANS      ; e(k-1) -> e(k-2)
           LDX #$0074     ; e(k) en flotante en RAM
           LDY #$006F     ; e(k-1) en flotante en RAM
           JSR TRANS      ; e(k) -> e(k-1)
           LDX #$0083     ; número en ASCII procedente de la tabla de conversión del dato del
                          ; convertidor A/D a ASCII. Corresponde a y(k)
```

```
        JSR ASCFLTP    ; Conversión del dato anterior de ASCII a flotante
        LDX #$0000
        LDY #$0005
        JSR TRANS
        LDX #$0079     ; r(k) en flotante en RAM
        LDY #$0000     ; FPACC1
        JSR TRANS      ; r(k) -> FPACC1
        JSR FLTSUBP    ; r(k) - y(k)
        LDX #$0000     ; FPACC1
        LDY #$0074     ; e(k)
        JSR TRANS      ; FPACC1 (r(k)-y(k)) -> e(k)
        LDX #$005B     ; A en flotante
        LDY #$0000     ; FPACC1
        JSR TRANS      ; A -> FPACC1
        LDX #$006A     ; e(k-2) en flotante en RAM
        LDY #$0005     ; FPACC2
        JSR TRANS      ; e(k-2) -> FPACC2
        JSR FLTMULP    ; Ae(k-2) -> FPACC1
        LDX #$0000     ; FPACC1
        LDY #$0032     ; Ae(k-2) en flotante
        JSR TRANS      ; FPACC1 -> Ae(k-2)
        LDX #$0060     ; B en flotante en RAM
        LDY #$0000     ; FPACC1
        JSR TRANS      ; B -> FPACC1
        LDX #$006F     ; e(k-1) en flotante en RAM
        LDY #$0005     ; FPACC2
        JSR TRANS      ; e(k-1) -> FPACC2
        JSR FLTMULP    ; Be(k-1)
        LDX #$0000     ; FPACC1
        LDY #$0037     ; Be(k-1) en flotante en RAM
        JSR TRANS      ; FPACC1 -> Be(k-1)
        LDX #$0065     ; C en flotante en RAM
        LDY #$0000     ; FPACC1
        JSR TRANS      ; C -> FPACC1
        LDX #$0074     ; e(k) en flotante en RAM
        LDY #$0005     ; FPACC2
        JSR TRANS      ; e(k) -> FPACC2
        JSR FLTMULP    ; Ce(k) -> FPACC1
        LDX #$0037     ; Be(k-1)
        LDY #$0005     ; FPACC2
        JSR TRANS      ; Be(k-1) -> FPACC2
        JSR FLTADDP    ; Ce(k) + Be(k-1)
        LDX #$0032     ; Ae(k-2)
        LDY #$0005     ; FPACC2
        JSR TRANS      ; Ae(k-2) -> FPACC2
        JSR FLTADDP    ; Ce(k) + Be(k-1) + Ae(k-2)
        LDX #$0000     ; FPACC1
        LDY #$0032     ; DATOM
        JSR TRANS      ; FPACC1 -> DATOM
        PULB
        PULA
        PULY
        PULX
        RTS

*******************************************************************************
*                          SUBRUTINA TRANS                                    *
*******************************************************************************
* Transfiere 5 bytes de la dirección almacenada en X a la dirección almacenada en Y

TRANS:  PSHA           ; almacena A en el stack
        PSHB           ; almacena B en el stack
        LDAB #$05      ; contador auxiliar para hacer la transferencia
TRANS1: LDAA $00,X     ; transfiere el valor al que apunta X
        STAA $00,Y     ; a la posición a la que apunta Y
        INX            ; incrementa X
        INY            ; incrementa Y
        DECB           ; decrementa B
        BNE TRANS1     ; mientras no se han transferido los 5 bytes, continua
        PULB           ; recupera el valor de B del stack
        PULA           ; recupera el valor de A del stack
        RTS            ; retorno de subrutina

*******************************************************************************
*                          SUBRUTINA FLTCERO                                  *
*******************************************************************************
* Llena con 00 los cinco bytes a partir de la dirección almacenada en X

FLTCERO:  PSHA         ; almacena A en el stack
```

```
          LDAA #$05    ; contador auxiliar para llenar 5 bytes con 00
FLTCERO1: CLR $00,X    ; limpia el byte al que apunta X
          INX          ; incrementa X
          DECA
          BNE FLTCERO1 ; lo hace 5 veces
          PULA         ; recupera el valor de A
          RTS          ; retorno de subrutina

*******************************************************************************
*                          SUBRUTINA TOMA MUESTRA                             *
*******************************************************************************

MUESTRA:  PSHX     ; Almacena X en el stack
          PSHY     ; Almacena Y en el stack
          PSHA     ; Almacena A en el stack
          PSHB     ; Almacena B en el stack
          LDX #$1000     ; Carga X con $1000 para indexar
          BSET PORTG,X,$03   ; RD -> 1,  WR -> 1  y
          BCLR PORTG,X,$02   ; WR -> 0  para establecer el protocolo de comunica
                             ; ción para leer el dato del convertidor A/D
          LDY #$0023   ; Valor necesario para un retardo de 150 us
          STY $0030    ; Se almacena en la dirección $0030
          JSR RETARDO  ; Se realiza el retardo de 150 us para esperar a que el convertidor realice
                       ; la conversión.
          BSET PORTG,X,$02 ; WR -> 1
          BCLR PORTG,X,$01 ; RD -> 0
          NOP              ; Retardo de 10 us para esperar a que el primer byte
          NOP              ; del dato llegue al puerto A
          NOP
          NOP
          NOP
          NOP
          NOP
          NOP
          NOP
          NOP
          LDAA PORTA,X  ; Lee el byte MENOS significativo del dato del A/D que ahora se encuentra
                        ; en el puerto A
          STAA $0051    ; Almacena este valor en la direción $0051
          BSET PORTG,X,$01   ; RD -> 1
          NOP           ; Retardo de 5 us
          NOP
          NOP
          NOP
          NOP
          BCLR PORTG,X,$01   ; RD -> 0
          NOP           ; Retardo de 10 us para esperar que el segundo byte
          NOP           ; del dato llegue al puerto A, de donde será leído por
          NOP           ; el ucontrolador maestro
          NOP
          NOP
          NOP
          NOP
          NOP
          NOP
          NOP
          LDAA PORTA,X  ; Se lee el byte MAS significativo del convertidor A/D
          STAA $0050    ; Se almacena esta lectura en la dirección $0050
          CMPA #$0F
          BHI  ES0
          BRA  CONTM
ES0:      CLR  $0050
          CLR  $0051
CONTM:    LDAA #$02     ; Número de cifras de cada valor en ASCII en la tabla y por el cual hay
                        ; que multiplicar para hacer una búsqueda dentro de la tabla
          STAA MULTIB   ; Almacena este 2 en MULTIB
          JSR  TAUX1    ; Se realiza (Dato de ADC) x (MULTIB) -> (REST). Entonces (REST) contiene
                        ; la dirección de donde se encuenta el dato del ADC en ASCII
          LDD  #$8000   ; Dirección de inicio de la tabal de conversión de hexadecimal a ASCII
          ADDD REST     ; 8000+(REST) -> D por ello, D contiene la dirección absoluta de donde se
                        ; localiza el dato del ADC en ASCII
          XGDX          ; Transfiere el valor de D a X
          LDAA #$2E     ; Punto decimal en ASCII
          STAA $0083    ; Colocar el punto decimal del dato en ASCII procedente de ADC en la
                        ; localidad $0083
          LDAA $00,X    ; Carga en A el primer byte procedente de la tabla de conversión este byte
                        ; contiene 2 dígitos en ASCII, mismos que serán separados adecuadamente.
          STAA $18C     ; Coloca en RAM el byte más significativo de la muestra
                        ; a enviarse a la PC
```

```
        ANDA #$F0     ; Separación de nible más significativo
        LSRA          ; Corrimiento hacia la derecha del dato en A
        LSRA          ; Corrimiento hacia la derecha del dato en A
        LSRA          ; Corrimiento hacia la derecha del dato en A
        LSRA          ; Corrimiento hacia la derecha del dato en A
        ADDA #$30     ; Se le suma un #$30 al dato en A
        STAA $0084    ; y se almacena este resultado en la dirección $0084
        LDAA $00,X    ; Se carga en A nuevamente el primer byte procedente de la tabla de
                      ; conversión, ahora para tomar el nibble menos significativo
        ANDA #$0F     ; Se separa el nible menos significativo
        ADDA #$30     ; Se le suman #$30 para convertir el número en A a ASCII
        STAA $0085    ; Se almacena este valor en la dirección $0085
        INX           ; Se incrementa el valor de X ahora para trabajar con el byte menos
                      ; significativo del dato en la tabla de conversión
        LDAA $00,X ; Carga en A el segundo byte procedente de la tabla de conversión este byte
                   ; contiene 2 dígitos en ASCII, mismos que serán separados adecuadamente
        STAA $18D  ; Coloca en RAM el byte menos significativo de la muestra a enviarse a la PC
        ANDA #$F0  ; Separación de niblo más significativo
        LSRA       ; Corrimiento hacia la derecha del dato en A
        LSRA       ; Corrimiento hacia la derecha del dato en A
        LSRA       ; Corrimiento hacia la derecha del dato en A
        LSRA       ; Corrimiento hacia la derecha del dato en A
        ADDA #$30  ; Se le suma un #$30 al dato en A
        STAA $0086 ; y se almacena este resultado en la dirección $0086
        LDAA $00,X ; Se carga en A nuevamente el primer byte procedente de tabla de
                   ; conversión, ahora para tomar el nibble menos significativo
        ANDA #$0F     ; Se separa el nible menos significativo
        ADDA #$30     ; Se le suman #$30 para convertir el número en A a ASCII
        STAA $0087    ; Se almacena este valor en la dirección $0087
        CLR $0088     ; $00 en la dirección $0088 para indicar el final del número en ASCII
        PULB     ; Restaura el valor de B
        PULA     ; Restaura el valor de A
        PULY     ; Restaura el valor de Y
        PULX     ; Restaura el valor de X
        RTS      ; Retorno de subsubrutina

*******************************************************************************
*                          SUBRUTINA TAUX1                                    *
*******************************************************************************
* Subrutina para multiplicar un número de 12 bits, localizado en la dirección
* MULTIA por un número de 1 nible localizado en MULTIB

TAUX1:  PSHX              ; Almacena X en el stack
        PSHA              ; Almacena A en el stack
        PSHB              ; Almacena B en el stack
        LDX    #$01       ; Carga X con 1 para indexar
        LDAA   MULTIA,X   ; Se obtiene el byte menos significativo de MULTIA
        LDAB   MULTIB     ; LLevar MULTIB a B
        MUL               ; A * B -> D
        STAA   RESA       ; Carga A a RESA
        STAB   RESB       ; Carga B a RESB
        LDAA   MULTIA     ; Separar el byte más significativo de MULTIA y llevarlo a A
        LDAB   MULTIB     ; MULTIB -> B
        MUL               ; A * B -> D
        ADDB   RESA       ; RESA + B -> B
        ASLD              ; Cuatro corrimientos a la izquierda
        ASLD
        ASLD
        ASLD
        ASLD              ; Cuatro corrimientos a la izquierda
        ASLD
        ASLD
        ASLD
        LDAB   RESB       ; RESA -> B
        STD    REST       ; D -> REST
        PULB              ; Saca B del stack
        PULA              ; Saca A del stack
        PULX              ; Saca X del stack
        RTS

*******************************************************************************
*                          RUNTINA RETARDO                                    *
*******************************************************************************
; Notas: la dirección $0030 y $0031 de RAM almacenan un valor correspondiente
; al retardo que se desea, el máximo retardo es de 0.26 seg aproximadamente,
; cuando ($0030) = #$FFFF

RETARDO:  PSHX            ; Introduce X al stack
          LDX  $0030      ; Carga X con lo que hay en la dirección $0384
```

```
VUELTA:   NOP              ; Retarda 2 ciclos de reloj
          DEX              ; Decrementa X
          BNE  VUELTA      ; Mientras X <> 0 salta a VUELTA (retarda)
          PULX             ; Saca X del stack
          RTS              ; Regresa de subrutina

*****************************************************************************
*            SUBRUTINA DE SERVICIO DE INTERRUPCION POR SCI                  *
*****************************************************************************

RISCI:    ORG  $B700
          LDX  #$1000
          LDAA SCSR,X ; Acceso a SCSR para borrar la bandera RDRF
          LDAA SCDR,X ; Se lee el dato recibido en A
          CMPA #$11   ; Actualización de velocidad en RAM
          BNE  RISCI1
          LDAB #$06   ; ------------------------------------------------------
          STAB $185   ; Se recibirán los 6 siguientes datos provenientes de la PC
          LDY  #$140  ; en las localidades de velocidad en RAM $140-$145
          STY  $186
          JSR  RECIBIR
          LDX  #$140  ; ------------------------------------------------------
          JSR  ASCFLTP ; Conversión del dato recibido de ASCII a flotante
          LDX  #$0000 ; y almacenado en las localidades de RAM correspondientes
          LDY  #$0079 ; a la velocidad en formato punto flotante
          JSR  TRANS
          JMP  FINRISCI
RISCI1:   CMPA #$55    ; Velocidad en RAM=0 para grabar parametros o velocidad
          BNE  RISCI3
          LDY  #$0000 ; ------------------------------------------------------
          STY  $0079  ; Velocidad en flotante en RAM = 0
          STY  $007B
          CLR  $007D  ; ------------------------------------------------------
          JMP  FINRISCI
RISCI3:   CMPA #$77    ; Almacena un $77 en la localidad testigo para grabar
          BNE  RISCI4  ; parametros al salir de la interrrupción
          LDAB #$28    ; ------------------------------------------------------
          STAB $185    ; Recibe los datos A,B,C,D y F en ASCII en RAM
          LDY  #$0152
          STY  $186
          JSR  RECIBIR ; ------------------------------------------------------
          STAA $150
          JMP  FINRISCI
RISCI4:   CMPA #$88    ; Almacena un $88 en la localidad testigo para grabar
          BNE  FINRISCI ; velocidad al salir de la interrupción
          STAA $151
FINRISCI: RTI

*****************************************************************************
*                            SUBRUTINA GRABAPAR                             *
*****************************************************************************
* Es empleada para grabar los datos A,B,C,D y F en EEPROM

GRABAPAR: SEI
          CLR  $150   ; Limpia la bandera de Grabar Parametros
          LDAA #$18   ; ------------------------------------------------------
          STAA $180   ; Graba los datos A, B y C
          LDX  #$152
          STX  $181
          LDX  #$FE01
          STX  $183
          JSR  GRABAR ; ------------------------------------------------------
          LDAA #$10   ; ------------------------------------------------------
          STAA $180   ; Graba los datos D y F
          LDX  #$16A
          STX  $181
          LDX  #$FE21
          STX  $183
          JSR  GRABAR ; ------------------------------------------------------
STAY1:    BRA  STAY1  ; Permanece en este ciclo hasta que se reinicialice el sistema

*****************************************************************************
*                          SUBRUTINA GRABAVEL                               *
*****************************************************************************
* Empleada para grabar la velocidad de referencia en EEPROM

GRABAVEL: SEI
          CLR $151   ; Limpia la bandera de Graba Velocidad
          LDAA #$06  ; ------------------------------------------------------
```

```
          STAA $180  ; Graba en EEPROM la velocidad de referencia a la que
          LDX  #$140 ; se encuentre trabajando el motor (en ASCII)
          STX  $181
          LDX  #$FE19
          STX  $183
          JSR  GRABAR ; ------------------------------------------------
STAY2:    BRA STAY2   ; Permanece en este ciclo hasta que se reinicialice el
                      ; sistema

*****************************************************************************
*                          SUBRUTINA GRABAR                                 *
*****************************************************************************
; Esta subrutina graba n bytes (n almacenado en la $180),los bytes a grabar se encuentran en
; RAM a partir de la direccion almacenada en la $181-$182. La direccion destino del primer
; byte a grabar en  EEPROM esta almacenada en la $183-$184

GRABAR:   LDX #$1000
          BSET OPTION,X,$90
          LDAB $180
          LDX $181
          LDY $183
GRAB1:    JSR BYTERASE ; Borrar el byte en EEPROM al que apunta Y
          LDAA $00,X
          JSR PROG ; Programa el byte contenido en A en la direccion a la que
                   ; apunta Y
          INY
          INX
          DECB
          BNE GRAB1
          RTS

*****************************************************************************
*                            SUBRUTINA BYTERASE                             *
*****************************************************************************
;Y apunta a la localidad a borrar

BYTERASE: PSHB
          PSHX
          LDX #$1000
          BSET PPROG,X,$18 ; Modo de borrado de 1 Byte
          BSET PPROG,X,$06 ; EELAT -> 1, ERASE -> 1.
          LDAB #$AA
          STAB $00,Y       ; Se escribe en la direccion del byte a borrar
          BSET PPROG,X,$01 ; EEPGM -> 1.
          LDX #$09C1
          STX $0030
          JSR RETARDO      ; Retardo de 10ms (tiempo de borrado)
          LDX #$1000
          BCLR PPROG,X,$01 ; EEPGM -> 0
          BCLR PPROG,X,$06 ; EELAT -> 0, ERASE -> 0.
          PULX
          PULB
          RTS

*****************************************************************************
*                              SUBRUTINA PROG                               *
*****************************************************************************
* Programa el byte contenido en A en la dirección de EEPROM a la que apunta Y

PROG:     PSHB
          PSHX
          LDX  #$1000
          BSET PPROG,X,$02  ; EELAT -> 1
          STAA $00,Y
          BSET PPROG,X,$01  ; EEPGM -> 1. Activa el alto voltaje
          LDX  #$09C1
          STX  $0030
          JSR  RETARDO      ; Retardo de 10ms (tiempo de programación)
          LDX #$1000
          BCLR PPROG,X,$01  ; EEPGM -> 0. Desactiva el alto voltaje
          CLR  PPROG,X      ; Selecciona modo de lectura de EEPROM
          PULX
          PULB
          RTS

*****************************************************************************
*                             RECIBIR                                       *
*****************************************************************************
; Recibe el número de bytes almacenado en la dirección $185 y los carga a
```

```
; partir de la dirección almacenada en la $186-$187

RECIBIR:  PSHA
          PSHB
          PSHX
          PSHY
          LDY  $186
          LDX  #$1000
          LDAB $185
RECIB1:   BRCLR SCSR,X,$20,RECIB1  ; Detecta la llegada de un dato
          LDAA SCDR,X
          STAA $00,Y ; Almacena el dato en la dirección a la que apunta Y
          INY
          DECB
          BNE  RECIB1
          PULY
          PULX
          PULB
          PULA
          RTS

*****************************************************************************
*                              SUBRUTINA ENVIAR                             *
*****************************************************************************
; Subrutina para enviar el número de bytes almacenado en la dirección $188 que
; que se encuentran a partir de la dirección almacenada en $189-$190

ENVIAR:   PSHA
          PSHB
          PSHX
          PSHY
          LDX  #$1000
                    ;--------------------------------
BEGIN:    LDY SCSR,X  ;envia un $cc como protocolo con la PC
          LDAA #$CC
          STAA SCDR,X  ;--------------------------------
          LDY  $189
ENVIAR2:  LDAA $00,Y
          LDAB SCSR,X
ENVIAR1:  BRCLR SCSR,X,$80,ENVIAR1
          STAA SCDR,X
          INY
          DEC  $188
          BNE  ENVIAR2
          PULY
          PULX
          PULB
          PULA
          RTS

*****************************************************************************
*                            SUBRUTINA  RIIRQ                               *
*****************************************************************************
* Subrutina de servicio de interrupción IRQ
          ORG   $E000
          LDX   #$1000
          LDAA  #$1F
          STAA  BPROT,X
          RTI

*****************************************************************************
*            TABLA DE CONVERSIoN DEL DATO DEL ADC A ASCII                   *
*****************************************************************************
* Esta tabla ocupa 8Kbytes de memoria EEPROM, a continuación se muestra una
* pequeña parte de ella para ilustrar la forma en la que esta construida

         ORG $8000
         FDB $0000
         FDB $0002
         FDB $0005
         FDB $0007
         FDB $0010
         FDB $0012
         FDB $0015
         FDB $0017
         FDB $0020
              .
              .
              .
```

```
        FDB $4041
        FDB $4043
        FDB $4046
        FDB $4048
             .
             .
             .
        FDB $9982
        FDB $9984
        FDB $9987
        FDB $9989
        FDB $9999

*****************************************************************************
*                    VECTORES DE RESET E INTERRUPCION                       *
*****************************************************************************

          ORG  $FFFE     ; Vector de RESET
          FDB  $CE00
          ORG  $FFE6     ; Vector de interrupción por TIMER
          FDB  $D200
          ORG  $FFD6     ; Vector de interrupción por SCI
          FDB  $E700
          ORG  $FFF2     ; Vector de interrupcion IRQ
          FDB  $E000
          ORG  $FC1F     ; Valor actual del offset para el TIMER (T = 100 ms)
          FDB  $C350     ; $(100ms/2us) = $C350
          ORG $FE19      ; r(k) en ASCII en EEPROM
          FCB $2E,$35,$30,$30,$30,$00
          ORG $FE01      ; A en ASCII en EEPROM
          FCB $32,$2E,$36,$31,$36,$30,$30,$00
          ORG $FE09      ; B en ASCII en EEPROM
          FCB $2D,$36,$2E,$31,$32,$30,$30,$00
          ORG $FE11      ; C en ASCII en EEPROM
          FCB $33,$2E,$36,$30,$30,$30,$30,$00
          ORG $FE21      ; D en ASCII en EEPROM
          FCB $2D,$2E,$32,$30,$30,$30,$30,$00
          ORG $FE29      ; F en ASCII en EEPROM
          FCB $31,$2E,$32,$30,$30,$30,$30,$00

          END
```

Para el Microcontrolador Esclavo:

```
*****************************************************************************
*                                 ETIQUETAS                                 *
*****************************************************************************

DIR1      EQU     $8000
DIR2      EQU     $80A0
DIR3      EQU     $8140
DIR4      EQU     $81E0
DIR5      EQU     $8280
DIR6      EQU     $8320
DIR7      EQU     $83C0
DIR8      EQU     $8460
DIR9      EQU     $8500
DIR10     EQU     $85A0
DIR11     EQU     $8640
DIR12     EQU     $86E0
DIR13     EQU     $8780
DIR14     EQU     $8820
DIR15     EQU     $88C0
DIR16     EQU     $8960
DIR17     EQU     $8A00
DIR18     EQU     $8AA0
DIR19     EQU     $8B40
DIR20     EQU     $8BE0
DIR21     EQU     $8C80
DIR22     EQU     $8D20
DIR23     EQU     $8DC0
DIR24     EQU     $8E60
DIR25     EQU     $8F00
DIR26     EQU     $8FA0
DIR27     EQU     $9040
DIR28     EQU     $90E0
DIR29     EQU     $9180
DIR30     EQU     $9220
```

```
DIR31    EQU     $92C0
DIR32    EQU     $9360
DIR33    EQU     $9400
DIR34    EQU     $94A0
DIR35    EQU     $9540
DIR36    EQU     $95E0
DIR37    EQU     $9680
DIR38    EQU     $9720
DIR39    EQU     $97C0
DIR40    EQU     $9860
DIR41    EQU     $9900
DIR42    EQU     $99A0
DIR43    EQU     $9A40
DIR44    EQU     $9AE0
DIR45    EQU     $9B80
DIR46    EQU     $9C20
DIR47    EQU     $9CC0
DIR48    EQU     $9D60
DIR49    EQU     $9E00
DIR50    EQU     $9EA0
DIR51    EQU     $9F40

*****************************************************************************
*                            PROGRAMA PRINCIPAL                             *
*****************************************************************************

         ORG $CE00      ; Origen del programa
         LDY #$1000     ; Y es auxiliar en indexación
         LDS #$03FF     ; Inicialización del apuntador de stack
         JSR ESCLAVO    ; Declaración del MCB como esclavo
         LDAA #$10              ; ---------------------------------------------
         LDX  #$150             ; Recibe los datos D y F que le envía el MCM y
DF1:     BRCLR SPSR,Y,$80,DF1 ; los almacena en RAM
         LDAB SPDR,Y
         STAB $00,X
         INX
         DECA
         BNE  DF1               ; ---------------------------------------------
         LDAA #$FF      ; $FF -> A
         STAA DDRA,Y    ; Declaración del puerto A como salida
         BCLR PORTG,Y,$0F ; PG0 a PG4 a cero
         BSET DDRG,Y,$0F  ; Declaración de PG0 a PG4 como salidas
         ;*********************
         LDX #$0333  ; Inicialización del motor con 2 Volts (fff/5)
         STX $0032
         STAA $0090  ; Al llamar a STODA esta subrutina regresa a REG1
         JMP STODA
         ;*********************
REG1:    JSR INICIA      ; Inicializa u(k-2), u(k-1), u(k), D y F
REGRESA: LDY #$1000
WAIT:    BRCLR SPSR,Y,$80,WAIT ;espera la llegada de un dato por SPI.
         LDAA SPDR,Y     ;recibe el dato en A
         CMPA #$CE       ;compara el dato de llegada con $CE
         BNE WAIT        ;Si no fue $CE regresa a wait.
         LDX #$0055      ;dirección donde se tiene almacenado u(k-1) en flotante
         LDY #$005A      ;dirección de u(k-2) en flotante
         JSR TRANS       ;u(k-1) -> u(k-2)
         LDX #$0050      ;dirección donde se tiene almacenado u(k) en flotante
         LDY #$0055      ;dirección de u(k-1) en flotante
         JSR TRANS       ;u(k) -> u(k-1)
         LDX #$006B      ; Localidad donde está D en flotante
         LDY #$0000      ; FPACC1
         JSR TRANS       ; D -> FPACC1
         LDX #$005A      ; Dirección donde se tiene almacenado u(k-2)
         LDY #$0005      ; FPACC2
         JSR TRANS       ; u(k-2) -> FPACC2
         JSR FLTMULP     ; Du(k-2) -> FPACC1
         LDX #$0000      ; FPACC1
         LDY #$005F      ; dirección de RAM donde se tiene almacenada Du(k-2)
         JSR TRANS       ; FPACC1 -> Du(k-2)
         LDX #$0070      ; F en flotante en RAM
         LDY #$0000      ; FPACC1
         JSR TRANS       ; F -> FPACC1
         LDX #$0055      ; dirección de RAM donde se tiene almacenada u(k-1)
         LDY #$0005      ; FPACC2
         JSR TRANS       ; u(k-1) -> FPACC2
         JSR FLTMULP     ; Fu(k-1) -> FPACC1
         LDX #$005F      ; dirección de RAM donde se tiene almacenada Du(k-2)
         LDY #$0005      ; FPACC2
```

```
        JSR TRANS       ; Du(k-2) -> FPACC2
        JSR FLTADDP     ; Fu(k-1)+Du(k-2) -> FPACC1
        LDX #$0000      ; dirección de RAM donde se tiene almacenada la oper. ant.
        LDY #$005F      ; DATOB
        JSR TRANS       ; Fu(k-1)+Du(k-2) -> DATOB
        LDY #$1000      ; Auxiliar para indexar
        LDAA #$05       ; Contador de los bytes recibidos
        LDX #$0064      ; Direción donde se va a almacenar el DATOM
REC1:   BRCLR SPSR,Y,$80,REC1  ; Espera la llegada de un dato por SPI.
        LDAB SPDR,Y     ; Recibe el dato en B
        STAB $00,X      ; Almacena B en la dirección que apunta X
        INX             ; Incrementa X
        DECA            ; Decrementa el contador de bytes recibidos
        BNE REC1        ; Cuando A = 0 continua hacia abajo
        LDX #$005F      ; DATOB
        LDY #$0000      ; FPACC1
        JSR TRANS       ; DATOB -> FPACC1
        LDX #$0064      ; DATOM
        LDY #$0005      ; FPACC2
        JSR TRANS       ; DATOM -> FPACC2
        JSR FLTADDP     ; DATOM+DATOB -> FPACC1
        LDX #$0000      ; FPACC1
        LDY #$0050      ; u(k) en flotante
        JSR TRANS       ; FPACC1 -> u(k)
        LDAA $0004      ; Signo de u(k)
        CMPA #$FF       ; ¿Es negativo?
        BEQ CERO1        ; Si es negativo  brinca a NEG1
        BRA ESCL1       ; Si es positivo brinca a ESCL1
CERO1:  JMP CERO
ESCL1:  LDX #$0100      ; X apunta a la dirección donde será almacenada
                        ; el valor de u(k) en ASCII
        JSR FLTASCP     ; convierte u(k) de flotante a ASCII y almacena el resultado
                        ; a partir de la dirección $0100
        LDX #$0101      ; X apunta al primer dato en ASCII
        STX $0030       ; Almacena la dirección a la que apunta X en la $0030
              ; Se revisa si no hay notación exponencial en el dato en ASCII
        LDAA $00,X      ; Carga en A el siguiente dato en ASCII
R45:    CMPA #$45       ; ¿ es una E ?
        BEQ CERO        ; Si hay notación exponencial, trunca el resultado
                        ; a cero porque son valores menores que 0.01
        INX
        LDAA $00,X      ; X apunta a la siguiente localidad.
        CMPA #$00       ; continúa revisando el número hasta que se acabe
        BNE R45         ; mientras no acabe el número, regresa a R45.
        LDX $0030      ; No fue exponencial, ahora revisa si es < 1
        LDAA $00,X     ; Carga en A el valor de la dirección a la que apunta X
        CMPA #$2E      ; ¿es un . ?
        BEQ MENUNO     ; si fue un . brinca a MENUNO, en caso contrario, el
                       ; número fue mayor que 1, por lo que continua
MAYUNO: LDY #$0FFF
        STY $0032       ;poner un 0FFF como dato para enviar al DAC
        LDY #$8180      ;**************
        STY $0050       ;*
        LDY #$0000      ;* PONE UN 1 EN FLOTANTE
        STY $0052       ;*
        LDAA #$00       ;*
        STAA $0054      ;**************
        JMP STODA       ;envia el número al DAC
CERO:   LDY #$0000
        STY $0032       ; Almacena un $0000 como dato a enviar al DAC
        STY $0050       ;**************
        STY $0052       ;* Pone un 0 en flotante en u(k)
        LDAA #$00       ;*
        STAA $0054      ;**************
        JMP  STODA      ; Envía el dato almacenado en $0032-$0033 al DAC
MENUNO: LDY #$0000      ; Limpia la localidad $0069-6A que es donde se encuentra
                        ; el dato a enviar a la tabla para convertirlo a hexa.
                        ; y que corresponde a u(k) en ASCII
        STY $0069
        LDX $0030       ; Carga X con la dirección en la que se había quedado,
                        ; cuando se revisó el 2D o 20 iniciales
        INX             ; Para quitar el 2E
        LDY #$0069      ; Cargar Y con la dirección de u(k) en ASCII que se enviará
                        ; para conversión a hexadecimal por medio de la tabla
        LDAB #$04  ; Se inicializa B como contador del número de decimales del valor en ASCII
MENUNO1: LDAA #$0F      ; Separa el nible menos significativo del dato al que
        ANDA $00,X      ; apunta X
        ASLA            ; Cuatro corrimientos hacia la izquierda
        ASLA
```

```
          ASLA
          ASLA
          STAA $0034      ; Almacena el valor de A en la $0034
          INX             ; Increm X para apuntar al siguiente valor en ASCII
          DECB            ; Decrementa el contador B
          LDAA $00,X      ; Carga en A el valor al que apunta X
          CMPA #$00       ; Verifica si ya acabó el número en ASCII para salir
          BEQ  FUE0       ; Cuando ya terminó el número brinca a FUE0
          LDAA #$0F       ; Se separa el nible menos significativo del número en ASCII
          ANDA $00,X
          ORAA $0034      ; A OR ($0034) -> A. Agrupa los dos nibles que se habían separado
                          ; anteriormente.
          STAA $00,Y      ; Almacena este valor en la dirección a la que apunta Y
          INY             ; Incrementa Y
          INX             ; Incrementa X
          DECB            ; Decrementa B
          LDAA $00,X      ; Carga en A el valor ASCII al que apunta X
          CMPA #$00       ; Verifica si ya acabó el número en ASCII para salir
          BEQ  FUE02      ; Cuando ya terminó el número brinca a FUE02
          CMPB #$00       ; Si ya se tomaron 4 decimales, es necesario salir
          BNE  MENUNO1    ; en caso contrario, continuar
          BRA  FUE02
FUE0:     LDAA $0034      ; Carga en A lo que hay en la dirección $0034
          STAA $00,Y      ; y lo almacena en la dirección a la que apunta Y
FUE02:    LDY  $0069      ; Carga en Y lo que hay en la dirección $0069
          JSR  ASCHEX     ; Convierte el dato en Y de ASCII a hexadecimal
          JMP  STODA      ; Envía este dato en hexa al DAC

*****************************************************************************
*                          SUBRUTINA ESCLAVO                                *
*****************************************************************************

ESCLAVO:  PSHX
          PSHA
          PSHB
          LDX  #$1000
          CLR  $28,X
          CLR  $29,X
          LDD  #$0647
          STAA $09,X
          STAB $28,X
          PULB
          PULA
          PULX
          RTS

*****************************************************************************
*                                STODA                                      *
*****************************************************************************
; Manda el dato almacenado en la dirección $0032-$0033 al convertidor D/A

STODA:    PSHX
          PSHY
          PSHA
          PSHB
          LDX #$1000
          LDD $0032    ; Carga en D el dato a enviar
          LSLD    ; Cuatro corrimientos a la izquierda para tener el MSB en A
          LSLD    ; y el nible menos significativo del dato en B
          LSLD
          LSLD
          BSET PORTG,X,$0F    ; CS -> 1, BYTE -> 1, WR -> 1, XFER -> 1
          STAA PORTA,X     ; Cargar el MSB en PORTA
          BCLR PORTG,X,$01  ; CS -> 0
          BCLR PORTG,X,$02  ; WR -> 0
          NOP  ; Retardo
          NOP
          BSET PORTG,X,$02    ; WR -> 1
          BSET PORTG,X,$01    ; CS -> 1
          STAB PORTA,X        ; LSB -> PORTA
          BCLR PORTG,X,$01    ; CS -> 0
          BCLR PORTG,X,$04    ; BYTE1/BITE2 -> 0
          BCLR PORTG,X,$02    ; WR -> 0
          NOP
          NOP
          BSET PORTG,X,$02    ; WR -> 1
          BSET PORTG,X,$04    ; BYTE -> 1
          BSET PORTG,X,$01    ; CS -> 1
          BCLR PORTG,X,$08    ; XFER -> 0
```

```
          BCLR PORTG,X,$02    ; WR -> 0
          NOP
          NOP
          BSET PORTG,X,$02    ; WR -> 1
          BSET PORTG,X,$08    ; XFER -> 1
          PULB
          PULA
          PULY
          PULX
          LDAA $0090  ; Bandera que indica a donde saltar de regreso
          CLR $0090
          CMPA #$FF
          BEQ  REG11
          JMP  REGRESA
REG11:    JMP  REG1

*******************************************************************************
*                            SUBRUTINA INICIA                                 *
*******************************************************************************
* Inicialización de variables en RAM

INICIA:     PSHX
            PSHY
            LDX  #$0050     ; u(k) en flotante en RAM
            JSR  FLTUNO     ; se inicializa con 0.2 en flotante
            LDX  #$0055     ; u(k-1) en flotante en RAM
            JSR  FLTUNO     ; se inicializa con 0.2 en flotante
            LDX  #$005A     ; u(k-2) en flotante en RAM
            JSR  FLTUNO     ; se inicializa con 0.2 flotante
            LDX  #$0150     ; D en ASCII en la EEPROM
            JSR  ASCFLTP    ; conversión de ASCII a flotante
            LDX  #$0000     ; FPACC1
            LDY  #$006B     ; D en flotante en RAM
            JSR  TRANS      ; FPACC1 -> D en flotante en RAM
            LDX  #$0158     ; F en EEPROM en ASCII
            JSR  ASCFLTP    ; conversión de ASCII a flotante
            LDX  #$0000     ; FPACC1
            LDY  #$0070     ; F en flotante en RAM
            JSR  TRANS      ; FPACC1 -> F en flotante en RAM
            PULY
            PULX
            RTS             ; Retorno de subrutina

*******************************************************************************
*                              SUBRUTINA TRANS                                *
*******************************************************************************
* Transfiere 5 bytes de la dirección almacenada en X a la dirección almacenada en Y

TRANS:      PSHA          ; almacena A en el stack
            PSHB          ; almacena B en el stack
            LDAB #$05     ; contador auxiliar para hacer la transferencia
TRANS1:     LDAA $00,X    ; transfiere el valor al que apunta X
            STAA $00,Y    ; a la posición a la que apunta Y
            INX           ; incrementa X
            INY           ; incrementa Y
            DECB          ; decrementa B
            BNE TRANS1    ; mientras no se han transferido los 5 bytes, continua
            PULB          ; recupera el valor de B del stack
            PULA          ; recupera el valor de A del stack
            RTS           ; retorno de subrutina

*******************************************************************************
*                             SUBRUTINA FLTCERO                               *
*******************************************************************************
* Llena con 00 los cinco bytes a partir de la dirección almacenada en X

FLTCERO:   PSHA         ; almacena A en el stack
           LDAA #$05    ; contador auxiliar para llenar 5 bytes con 00
FLTCERO1:  CLR $00,X    ; limpia el byte al que apunta X
           INX          ; incrementa X
           DECA
           BNE FLTCERO1 ; lo hace 5 veces
           PULA         ; recupera el valor de A
           RTS          ; retorno de subrutina

*******************************************************************************
*                             SUBRUTINA FLTUNO                                *
*******************************************************************************
* Llena con 0.2 en flotante a partir de la dirección almacenada en X
```

```
FLTUNO:  PSHY
         LDY #$7ECC
         STY $00,X
         INX
         INX
         LDY #$CCCD
         STY $00,X
         INX
         INX
         CLR $00,X
         PULY
         RTS

*******************************************************************************
*                              SUBRUTINA ASCHEX                               *
*******************************************************************************

* Subaubrutina de conversión de un número en ASCII a hexadecimal
* "Y" contiene el número a convertir en 2 bytes

ASCHEX:   PSHX
          PSHA
          PSHB
          LDD #DIR1
          LDX #DIR2
          CPY $00,X
          BHI A
          JMP ASCHEX1
A:        XGDX
          LDX #DIR3
          CPY $00,X
          BHI B
          JMP ASCHEX1
B:        XGDX
          LDX #DIR4
          CPY $00,X
          BHI C
          JMP ASCHEX1
C:        XGDX
          LDX #DIR5
          CPY $00,X
          BHI D
          JMP ASCHEX1
D:        XGDX
          LDX #DIR6
          CPY $00,X
          BHI E
          JMP ASCHEX1
E:        XGDX
          LDX #DIR7
          CPY $00,X
          BHI F
          JMP ASCHEX1
F:        XGDX
          LDX #DIR8
          CPY $00,X
          BHI G
          JMP ASCHEX1
G:        XGDX
          LDX #DIR9
          CPY $00,X
          BHI H
          JMP ASCHEX1
H:        XGDX
          LDX #DIR10
          CPY $00,X
          BHI I
          JMP ASCHEX1
I:        XGDX
          LDX #DIR11
          CPY $00,X
          BHI AA
          JMP ASCHEX1
AA:       XGDX
          LDX #DIR12
          CPY $00,X
          BHI AB
          JMP ASCHEX1
AB:       XGDX
```

```
        LDX #DIR13
        CPY $00,X
        BHI AC
        JMP ASCHEX1
AC:     XGDX
        LDX #DIR14
        CPY $00,X
        BHI AD
        JMP ASCHEX1
AD:     XGDX
        LDX #DIR15
        CPY $00,X
        BHI AE
        JMP ASCHEX1
AE:     XGDX
        LDX #DIR16
        CPY $00,X
        BHI AF
        JMP ASCHEX1
AF:     XGDX
        LDX #DIR17
        CPY $00,X
        BHI AG
        JMP ASCHEX1
AG:     XGDX
        LDX #DIR18
        CPY $00,X
        BHI AH
        JMP ASCHEX1
AH:     XGDX
        LDX #DIR19
        CPY $00,X
        BHI AI
        JMP ASCHEX1
AI:     XGDX
        LDX #DIR20
        CPY $00,X
        BHI BA
        JMP ASCHEX1
BA:     XGDX
        LDX #DIR21
        CPY $00,X
        BHI BB
        JMP ASCHEX1
BB:     XGDX
        LDX #DIR22
        CPY $00,X
        BHI BC
        JMP ASCHEX1
BC:     XGDX
        LDX #DIR23
        CPY $00,X
        BHI BD
        JMP ASCHEX1
BD:     XGDX
        LDX #DIR24
        CPY $00,X
        BHI BE
        JMP ASCHEX1
BE:     XGDX
        LDX #DIR25
        CPY $00,X
        BHI BF
        JMP ASCHEX1
BF:     XGDX
        LDX #DIR26
        CPY $00,X
        BHI BG
        JMP ASCHEX1
BG:     XGDX
        LDX #DIR27
        CPY $00,X
        BHI BH
        JMP ASCHEX1
BH:     XGDX
        LDX #DIR28
        CPY $00,X
        BHI BI
        JMP ASCHEX1
```

```
BI:     XGDX
        LDX #DIR29
        CPY $00,X
        BHI CA
        JMP ASCHEX1
CA:     XGDX
        LDX #DIR30
        CPY $00,X
        BHI CB
        JMP ASCHEX1
CB:     XGDX
        LDX #DIR31
        CPY $00,X
        BHI CC
        JMP ASCHEX1
CC:     XGDX
        LDX #DIR32
        CPY $00,X
        BHI CD
        JMP ASCHEX1
CD:     XGDX
        LDX #DIR33
        CPY $00,X
        BHI CE
        JMP ASCHEX1
CE:     XGDX
        LDX #DIR34
        CPY $00,X
        BHI CF
        JMP ASCHEX1
CF:     XGDX
        LDX #DIR35
        CPY $00,X
        BHI CG
        JMP ASCHEX1
CG:     XGDX
        LDX #DIR36
        CPY $00,X
        BHI CH
        JMP ASCHEX1
CH:     XGDX
        LDX #DIR37
        CPY $00,X
        BHI CI
        JMP ASCHEX1
CI:     XGDX
        LDX #DIR38
        CPY $00,X

        BHI DA
        JMP ASCHEX1
DA:     XGDX
        LDX #DIR39
        CPY $00,X
        BHI DB
        JMP ASCHEX1
DB:     XGDX
        LDX #DIR40
        CPY $00,X
        BHI DC
        JMP ASCHEX1
DC:     XGDX
        LDX #DIR41
        CPY $00,X
        BHI DD
        JMP ASCHEX1
DD:     XGDX
        LDX #DIR42
        CPY $00,X
        BHI DE
        JMP ASCHEX1
DE:     XGDX
        LDX #DIR43
        CPY $00,X
        BHI DF
        JMP ASCHEX1
DF:     XGDX
        LDX #DIR44
        CPY $00,X
```

```
          BHI DG
          JMP ASCHEX1
DG:       XGDX
          LDX #DIR45
          CPY $00,X
          BHI DH
          JMP ASCHEX1
DH:       XGDX
          LDX #DIR46
          CPY $00,X
          BHI DI
          JMP ASCHEX1
DI:       XGDX
          LDX #DIR47
          CPY $00,X
          BHI DJ
          JMP ASCHEX1
DJ:       XGDX
          LDX #DIR48
          CPY $00,X
          BHI DK
          JMP ASCHEX1
DK:       XGDX
          LDX #DIR49
          CPY $00,X
          BHI EA
          JMP ASCHEX1
EA:       XGDX
          LDX #DIR50
          CPY $00,X
          BHI EB
          JMP ASCHEX1
EB:       XGDX
          LDX #DIR51
ASCHEX1:  CPY $00,X
          BEQ IGUAL
          XGDX
          BRA ASCHEX2
ASCHEX3:  INX
          INX
ASCHEX2:  CPY $00,X
          BHI ASCHEX3
IGUAL:    XGDX    ; Cargar a D la dirección que tiene almacenada X
          SUBD #$8000 ; X-$8000 -> D
          LDX #$0002
          IDIV  ; D/2 -> X
          STX $0032
          PULB
          PULA
          PULX
          RTS

*******************************************************************************
*        TABLA DE CONVERSIóN DEL DATO DEL Convertidor A/D A ASCII              *
*******************************************************************************
* Esta tabla ocupa 8Kbytes de memoria EEPROM, a continuación se muestra una
* pequeña parte de ella para ilustrar la forma en la que esta construida

          ORG $8000
          FDB $0000
          FDB $0002
          FDB $0005
          FDB $0007
          FDB $0010
          FDB $0012
          FDB $0015
          FDB $0017
              .
              .
              .
          FDB $7791
          FDB $7793
          FDB $7796
          FDB $7798
          FDB $7801
              .
              .
              .
           FDB $9972
```

```
            FDB $9975
            FDB $9977
            FDB $9980
            FDB $9982
            FDB $9984
            FDB $9987
            FDB $9989
            FDB $999A

*******************************************************************************
*                          VECTOR DE RESET                                    *
*******************************************************************************

            ORG  $FFFE     ; Vector de RESET
            FDB  $CE00

            END
```

Para ambos microcontroladores:

```
*****************************************************************************************************
* RUTINAS DE SUMA, RESTA, MULTIPLICACION y CONVERSION DE ASCII A FLOTANTE Y DE FLOTANTE A ASCII
*
*      DEL PAQUETE DE ARITMETICA DE PUNTO FLOTANTE
* * Nota: No se incluyen las subrutinas auxiliares
      *
*****************************************************************************************************

*******************************************************************************
*                 Conversion de ASCII a Punto Flotante                        *
*******************************************************************************

                ORG $C000
ASCFLT:         EQU *
                PSHX
                JSR PSHFPAC2
                LDX #$0000
                PSHX
                STX $00                         ;FPACC1EX
                STX $02                         ;FPACC1EX+2
                CLR MANTSGN1
                TSY
                LDX $06,Y
ASCFLT1:        LDAA $00,X
                JSR NUMERIC
                BCS ASCFLT4
ASCFLT2:        CMPA #$2D
                BNE ASCFLT3
                COM MANTSGN1
                INX
                LDAA $00,X
                JSR NUMERIC
                BCS ASCFLT4
ASCFLT3:        CMPA #$2B
                BNE ASCFLT5
                INX
                LDAA $00,X
                JSR NUMERIC
                BCC ASCFLT5
                JMP ASCFLT11
ASCFLT5:        INS
                INS
                JSR PULFPAC2
                PULX
                LDAA #$01
                SEC
                RTS
ASCFLT4:        LDAA $00,X
                JSR NUMERIC
                BCC ASCFLT10
                JSR ADDNXTD
                INX
                BCC ASCFLT4
ASCFLT6:        INC FPACC1EX
```

```
                LDAA $00,X
                INX
                JSR NUMERIC
                BCS ASCFLT6
                CMPA #$2E
                BNE ASCFLT7
ASCFLT8:        LDAA $00,X
                JSR NUMERIC
                BCC ASCFLT7
                INX
                BRA ASCFLT8
ASCFLT7:        CMPA #$45
                BEQ ASCFLT13
                JMP FINISH
ASCFLT13:       INX
                LDAA $00,X
                JSR NUMERIC
                BCS ASCFLT9
                CMPA #$2D
                BEQ ASCFLT15
                CMPA #$2B
                BEQ ASCFLT16
                BRA ASCFLT5
ASCFLT15:       COM EXPSIGN,Y
ASCFLT16:       INX
                LDAA $00,X
                JSR NUMERIC
                BCC ASCFLT5
ASCFLT9:        SUBA #$30
                STAA PWR10EXP,Y
                INX
                LDAA $00,X
                JSR NUMERIC
                BCC ASCFLT14
                LDAB PWR10EXP,Y
                LSLB
                LSLB
                ADDB PWR10EXP,Y
                LSLB
                SUBA #$30
                ABA
                STAA PWR10EXP,Y
                CMPA #$26
                BHI ASCFLT5
ASCFLT14:       LDAA PWR10EXP,Y
                TST EXPSIGN,Y
                BPL ASCFLT12
                NEGA
ASCFLT12:       ADDA $00                ;FPACC1EX
                STAA $00                ;FPACC1EX
                BRA FINISH
ASCFLT10:       CMPA #$2E
                BNE ASCFLT7
                INX
ASCFLT11:       LDAA $00,X
                JSR NUMERIC
                BCC ASCFLT7
                BSR ADDNXTD
                INX
                BCS ASCFLT8
                DEC FPACC1EX
                BRA ASCFLT11
ADDNXTD:        LDAA $01                ;FPACC1EX
                STAA $06                ;FPACC2MN
                LDD  $02                ;FPACC1MN+1
                STD  $07                ;FPACC2MN+2
                LSLD
                ROL FPACC1MN
                BCS ADDNXTD
                LSLD
                ROL FPACC1MN
                BCS ADDNXTD1
                ADDD $07                ;FPACC2MN+1
                PSHA
                LDAA $01                ;FPACC1MN
                ADCA #$00
                ADDA $06                ;FPACC2MN
                STAA $01                ;FPACC1MN
                PULA
```

```
                BCS ADDNXTD1
                LSLD
                ROL FPACC1MN
                STD $02                 ;FPACC1MN+1
                BCS ADDNXTD1
                LDAB $00,X
                SUBB #$30
                CLRA
                ADDD $02                ;FPACC1MN+1
                STD  $02                ;FPACC1MN+1
                LDAA $01                ;FPACC1MN
                ADCA #$00
                BCS ADDNXTD1
                STAA $01                ;FPACC1MN
                RTS
ADDNXTD1:       LDD $07                 ;FPACC2MN+1
                STD $02                 ;FPACC1MN+1
                LDAA $06                ;FPACC2MN
                STAA $01                ;FPACC1MN
                RTS
FINISH:         STX $06,Y
                LDX #$0000              ;#FPACC1EX
                JSR CHCK0
                BEQ FINISH3
                LDAA $00                ;FPACC1EX
                STAA PWR10EXP,Y
                LDAA #$98               ;#$80+24
                STAA $00                ;FPACC1EX
                JSR FPNORM
                TST PWR10EXP,Y
                BEQ FINISH3
                BPL FINISH1
                LDX #$C18B              ;#CONSTP1
                JSR GETFPAC2
                NEG PWR10EXP,Y
                BRA FINISH2
FINISH1:        LDX #$C18F              ;#CONST10
                JSR GETFPAC2
FINISH2:        JSR FLTMUL
                DEC PWR10EXP,Y
                BNE FINISH2
FINISH3:        INS
                INS
                JSR PULFPAC2
                PULX
                RTS
NUMBRIC:        CMPA #$30
                BLO NUMBRIC1
                CMPA #$39
                BHI NUMBRIC1
                SEC
                RTS
NUMBRIC1:       CLC
                RTS

*******************************************************************************
*                              FLTMUL                                         *
*******************************************************************************

FLTMUL:         JSR PSHFPAC2            ;RUTINA PARA MULTIPLICACION EN P. FLOTANTE.
                LDX #$0000                  ;#FPACC1EX
                JSR CHCK0
                BEQ FPMULT3
                LDX #$0005                   ;#FPACC2EX
                JSR CHCK0
                BNE FPMULT4
                CLRA
                CLRB
                STD $00                 ;FPACC1EX
                STD $02                 ;FPACC1MN+1
                BRA FPMULT3
FPMULT4:        LDAA $04                ;MANTSGN1
                EORA $09                ;MANTSGN2
                STAA $04                ;MANTSGN1
                LDAA $00                ;FPACC1EX
                ADDA $05                ;FPACC2EX
                BPL FPMULT1
                BCC FPMULT2
FPMULT5:        LDAA #$02
```

```
                SBC
                BRA FPMULT6
FPMULT1:        BCS FPMULT2
                LDAA #$03
                SEC
                BRA FPMULT6
FPMULT2:        ADDA #$80
                STAA $00                ;FPACC1EX
                JSR UMULT
FPMULT3:        TST FPACC1EX
                BEQ FPMULT5
                CLC
FPMULT6:        JSR PULFPAC2
                RTS
UMULT:          LDX #$0000
                PSHX
                PSHX
                TSX
                LDAA #$18
                STAA $00,X
UMULT1:         LDAA $08                ;FPACC2MN+2
                LSRA
                BCC UMULT2
                LDD $02                 ;FPACC1MN+1
                ADDD $02,X
                STD $02,X
                LDAA $01                ;FPACC1MN
                ADCA $01,X
                STAA $01,X
UMULT2:         ROR $01,X
                ROR $02,X
                ROR $03,X
                ROR FPACC2MN
                ROR FPACC2MN+1
                ROR FPACC2MN+2
                DEC $00,X
                BNE UMULT1
                TST $01,X
                BMI UMULT3
                LSL FPACC2MN
                ROL $03,X
                ROL $02,X
                ROL $01,X
                DEC FPACC1EX
UMULT3:         TST FPACC2MN
                BPL UMULT4
                LDD $02,X
                ADDD #$0001
                STD $02,X
                LDAA $01,X
                ADCA #$00
                STAA $01,X
                BCC UMULT4
                ROR $01,X
                ROR $02,X
                ROR $03,X
                INC FPACC1EX
UMULT4:         INS
                PULX
                STX $01                 ;FPACC1MN
                PULA
                STAA $03                ;FPACC1MN+2
                RTS

*****************************************************************************
*                                FLTADD                                     *
*****************************************************************************

FLTADD:         JSR PSHFPAC2        ;RUTINA PARA SUMA EN PUNTO FLOTANTE.
                LDX #$0005              ;#FPACC2EX
                JSR CHCK0
                BNE FLTADD1
FLTADD6:        CLC
FLTADD10:       JSR PULFPAC2
                RTS
FLTADD1:        LDX #$0000              ;#FPACC1EX
                JSR CHCK0
                BNE FLTADD2
FLTADD4:        LDD $05                 ;FPACC2EX
```

```
                STD $00                 ;FPACC1EX
                LDD $07                 ;FPACC2MN+1
                STD $02                 ;FPACC1MN+1
                LDAA $09                ;MANTSGN2
                STAA $04                ;MANTSGN1
                BRA FLTADD6
FLTADD2:        LDAA $00                ;FPACC1EX
                CMPA $05                ;FPACC2EX
                BEQ FLTADD7
                SUBA $05                ;FPACC2EX
                BPL FLTADD3
                NEGA
                CMPA #$17
                BHI FLTADD4
                TAB
                ADDB $00                ;FPACC1EX
                STAB $00                ;FPACC1EX
                LDX #$0001              ;#FPACC1MN
                BRA FLTADD5
FLTADD3:        CMPA #$17
                BHI FLTADD6
                LDX #$0006              ;#FPACC2MN
FLTADD5:        LSR $00,X
                ROR $01,X
                ROR $02,X
                DECA
                BNE FLTADD5
FLTADD7:        LDAA $04                ;MANTSGN1
                CMPA $09                ;MANTSGN2
                BEQ FLTADD11
                TST MANTSGN1
                BPL FLTADD8
                LDX $06                 ;FPACC2MN
                PSHX
                LDX $01                 ;FPACC1MN
                STX $06                 ;FPACC2MN
                PULX
                STX $01                 ;FPACC1MN
                LDX $08                 ;FPACC2MN+2
                PSHX
                LDX $03                 ;FPACC1MN+2
                STX $08                 ;FPACC2MN+2
                PULX
                STX $03                 ;FPACC1MN+2
FLTADD8:        LDD $02                 ;FPACC1MN+1
                SUBD $07                ;FPACC2MN+1
                STD  $02                ;FPACC1MN+1
                LDAA $01                ;FPACC1MN
                SBCA $06                ;FPACC2MN
                STAA $01                ;FPACC1MN
                BCC FLTADD9
                LDAA $01                ;FPACC1MN
                COMA
                PSHA
                LDD $02                 ;FPACC1MN+1
                COMB
                COMA
                ADDD #$0001
                STD $02                 ;FPACC1MN+1
                PULA
                ADCA #$00
                STAA $01                ;FPACC1MN
                LDAA #$FF
                STAA $04                ;MANTSGN1
FLTADD9:        JSR FPNORM
                BCC FLTADD12
                LDAA #$03
                SEC
                JMP FLTADD10
FLTADD12:       JMP FLTADD6
FLTADD11:       LDD $02                 ;FPACC2MN+1
                ADDD $07                ;FPACC2MN+1
                STD  $02                ;FPACC1MN+1
                LDAA $01                ;FPACC1MN
                ADCA $06                ;FPACC2MN
                STAA $01                ;FPACC1MN
                BCC FLTADD12
                ROR FPACC1MN
                ROR FPACC1MN+1
```

```
            ROR FPACC1MN+2
            INC FPACC1EX
            BNE FLTADD12
            LDAA #$02
            SEC
            JMP FLTADD10

*******************************************************************************
*                              FLTSUB                                         *
*******************************************************************************

FLTSUB:     BSR FLTSUB1                 ;RESTA EN PUNTO FLOTANTE.
            JSR FLTADD
FLTSUB1:    LDAA $09                    ;MANTSGN2
            EORA #$FF
            STAA $09                    ;MANTSGN2
            RTS

*******************************************************************************
*                              FLTASC                                         *
*******************************************************************************

FLTASC:     PSHX                        ;CONVERSION DE PUNTO FLOTANTE A ASCII.
            LDX #$0000                  ;#FPACC1EX
            JSR CHCK0
            BNE FLTASC1
            PULX
            LDD #$3000
            STD $00,X
            RTS
FLTASC1:    LDX $00                     ;FPACC1EX
            PSHX
            LDX $02                     ;FPACC1MN+1
            PSHX
            LDAA $04                    ;MANTSGN1
            PSHA
            JSR PSHFPAC2
            LDX #$0000
            PSHX
            PSHX
            PSHX
            TSY
            LDX $0F,Y
            LDAA #$20
            TST MANTSGN1
            BEQ FLTASC2
            CLR MANTSGN1
            LDAA #$2D
FLTASC2:    STAA $00,X
            INX
            STX $00,Y
FLTASC5:    LDX #$C545
            JSR GETFPAC2
            JSR FLTCMP
            BHI FLTASC3
            LDX #$C541
            JSR GETFPAC2
            JSR FLTCMP
            BHI FLTASC4
            DEC $02,Y
            LDX #$C18F                  ;#CONST10
FLTASC6:    JSR GETFPAC2
            JSR FLTMUL
            BRA FLTASC5
FLTASC3:    INC $02,Y
            LDX #$C18B
            BRA FLTASC6
FLTASC4:    LDX #$C549                  ;#CONSTP5
            JSR GETFPAC2
            JSR FLTADD
            LDAB $00                    ;FPACC1EX
            SUBB #$81
            NEGB
            ADDB #$17
            BRA FLTASC17
FLTASC7:    LSR FPACC1MN
            ROR FPACC1MN+1
            ROR FPACC1MN+2
            DECB
```

```
FLTASC17:   BNE FLTASC7
            LDAA #$01
            STAA $03,Y
            LDAA $02,Y
            ADDA #$08
            BMI FLTASC8
            CMPA #$08
            BHS FLTASC8
            DECA
            STAA $03,Y
            LDAA #$02
FLTASC8:    SUBA #$02
            STAA $02,Y
            TST $03,Y
            BGT FLTASC9
            LDAA #$2E
            LDX $00,Y
            STAA $00,X
            INX
            TST $03,Y
            BEQ FLTASC18
            LDAA #$30
            STAA $00,X
            INX
FLTASC18:   STX $00,Y
FLTASC9:    LDX #$C52C              ;#DECDIG
            LDAA #$07
            STAA $05,Y
FLTASC10:   CLR $04,Y
FLTASC11:   LDD $02                 ;FPACC1MN+1
            SUBD $01,X
            STD $02                 ;FPACC1MN+1
            LDAA $01                ;FPACC1MN
            SBCA $00,X
            STAA $01                ;FPACC1MN
            BCS FLTASC12
            INC $04,Y
            BRA FLTASC11
FLTASC12:   LDD $02                 ;FPACC1MN+1
            ADDD $01,X
            STD $02                 ;FPACC1MN+1
            LDAA $01                ;FPACC1MN
            ADCA $00,X
            STAA $01                ;FPACC1MN
            LDAA $04,Y
            ADDA #$30
            PSHX
            LDX $00,Y
            STAA $00,X
            INX
            DEC $03,Y
            BNE FLTASC16
            LDAA #$2E
            STAA $00,X
            INX
FLTASC16:   STX $00,Y
            PULX
            INX
            INX
            INX
            DEC $05,Y
            BNE FLTASC10
            LDX $00,Y
FLTASC13:   DEX
            LDAA $00,X
            CMPA #$30
            BEQ FLTASC13
            INX
            LDAB $02,Y
            BEQ FLTASC15
            LDAA #$45
            STAA $00,X
            INX
            LDAA #$2B
            STAA $00,X
            TSTB
            BPL FLTASC14
            NEGB
            LDAA #$2D
```

```
            STAA $00,X
FLTASC14:   INX
            STX $00,Y
            CLRA
            LDX #$000A
            IDIV
            PSHB
            XGDX
            ADDB #$30
            LDX $00,Y
            STAB $00,X
            INX
            PULB
            ADDB #$30
            STAB $00,X
            INX
FLTASC15:   CLR $00,X
            PULX
            PULX
            PULX
            JSR PULFPAC2
            PULA
            STAA $04                ;MANTSGN1
            PULX
            STX $02                 ;FPACC1MN+1
            PULX
            STX $00                 ;FPACC1EX
            PULX
            RTS
DECDIG:     FCB $0F,$42,$40
            FCB $01,$86,$A0
            FCB $00,$27,$10
            FCB $00,$03,$E8
            FCB $00,$00,$64
            FCB $00,$00,$0A
            FCB $00,$00,$01
P9999999:   FCB $94,$74,$23,$FB
N9999999:   FCB $98,$18,$96,$7F
CONSTP5:    FCB $80,$00,$00,$00
```

APENDICE C

```
/*****************************************************************************/
/***********          FACULTAD DE INGENIERIA, UNAM                   ***********/
/***********  PROGRAMA DE INTERFAZ CON EL USUARIO PARA EL SISTEMA    ***********/
/***********          DE CONTROL PID PARA UN MOTOR DE CD             ***********/
/***********                 ( SIU - PID )                           ***********/
/***********                                                         ***********/
/***********  DIRECTOR DE TESIS: Ing. J. Antonio Arredondo Garza     ***********/
/***********                                                         ***********/
/***********           GRACIELA NAJERA DEL RIO                       ***********/
/***********         JORGE ALFREDO CUEVAS GARIBAY                    ***********/
/***********                                                         ***********/
/***********                  SEPTIEMBRE 1995                        ***********/
/*****************************************************************************/

/* LIBRERIAS */

#include<stdio.h>
#include<math.h>
#include<graphics.h>
#include<stdlib.h>
#include<conio.h>
#include<alloc.h>
#include<dos.h>
#include<bios.h>
#include<string.h>
#include<ctype.h>

/* DEFINICIONES */

#define COM1 0  /* puerto a emplear */
#define SETTINGS (0xE0|0x00|0x00|0x03) /* Baudaje=9600, no paridad, 8 bits de datos */
#define DATA_READY 0x100  /* Bandera de Dato Listo para ser leido del puerto */
#define TSRE 0x4000  /* Bandera de "Transmit Shift Register Empty" */
#define ESC 27
#define VIRTUAL 10000 /* maxima coordenada virtual para 'x' y para 'y' */
#define MAXVEL  3720  /* velocidad máxima del motor trabajando con 10 V en sus  terminales */
#define MINVEL  800   /* velocidad minima para garantizar una respuesta lineal del motor */

/* FUNCIONES AUXILIARES */

float Regla3(float uno,long dos);
void  Mundialesxy(int x1,int y1, int x2,int y2,float xmin,float ymin,float xmax,float ymax,
                  float xr,int yr,int *xs,int *ys,int af,int hf);
void  Virtual(long xv,long yv,long hv, long av,int *XS ,int *YS,long af,long hf);
void  Menu1();
void  Menu7(char *strold, char *strnew);
void  Mensaje(char *strmen);
void  Mensaje2(void);
char  Advertencia(void);
char  Advertencia2(void);
void  Marco(float xaco1,float yaco1,float xaco2,float yaco2,int xvirtual1,int yvirtual1,
      int xvirtual2,int yvirtual2,int af,int hf);
void  Marco2(int xvirtual1,int yvirtual1,int xvirtual2,int yvirtual2,int af,int hf);
void  clrMarco(int x1, int y1, int x2, int y2);
void  clrMarco1(int x1, int y1, int x2, int y2);
void  cuadricula(int x1, int y1, int x2, int y2);
int   getentero(int x, int y);
float getflot(int x, int y);
void  STRTOASC(char *cad, int bytes,int *);
void  VELRAM(void);
void  nuevos_parametros(FILE *);
void  transmitir1(int);
void  transmitir2(int *BYTES,int);
void  Mostrar_Archivo(FILE * stream);
int   Salir(FILE *,FILE *);

/* ESTRUCTURAS */

struct coord {int x;
              int y;
             };      /* Estructura empleada por el algoritmo de graficación */

/* VARIABLES GLOBALES */

int  reseto=0;

/* --------------------  FUNCION PRINCIPAL ------------------------------ */
void main(void){

  char tecla,tecla2;  /* opción elegida */
  char cadm[5];
  char str[3];
  char KP[12];  /*cadenas auxiliares para enviar los valores de Kp,Td y Ti a pantalla */
```

```
char TD[12];
char TI[12];
char ch;
char adv = 'N'; /* respuesta del usuario ante una advertencia */
char texty[12]; /* cadena auxiliar para enviar la velocidad actual a pantalla */
FILE *fp; /*apuntador al archivo TRANS.dat, que almacena las muestras de los 10 primeros seg */
FILE *fpp; /* apuntador al archivo PARAM.dat, que almacena los parametros actuales del sistema
                        de control */
float xaco1,xaco2,yaco1,yaco2; /* coordenadas de los ejes al dibujar los marcos*/
float escalax,escalay,factorp; /* variables empladas para graficación */
float x,dx; /* coordenada x en el eje definido por xaco1 y xaco2 e incremento en x para cada
               muestra */
int gdriver = DETECT, gmode, errorcode;  /* variables empleadas en la inicialización del modo
                                            grafico */
int N = 100;  /* número de puntos que se grafican y que determinan la rapidez y la resolución
              de una imagen */
int xvirtual1,xvirtual2,yvirtual1,yvirtual2;  /* coordenadas virtuales dentro de una pantalla
                                                 de 10000X10000 */
int af,hf;   /* ancho físico y alto físico del monitor */
int xs1,ys1; /* coordenadas en pantalla correspondientes a un punto en coordenadas virtuales */
int xs,ys;
int xs2,ys2; /* coordenadas en pantalla correspondientes a un punto en coordenadas virtuales */
int xmax,ymax,xt1,xt2,yt1,yt2;
int i; /* contador auxiliar en ciclos */
int y,yant; /* valor de la muestra recibida y la muestra inmediata anterior recibida */
int n=0;    /* contador de la muestra que esta graficando 0<=n<=100 */
int par=1;  /* bandera que indica en que arreglo se deben ir almacenando las muestras anteriores
               (XY1 o XY2) */
int abyte[2]={0x10,0x00};
int cont_muestra = 0; /* número de la muestra que se esta graficando */
int c = 0;  /* contador auxiliar para recepcion de un dato */
int ban1=0;  /* banderas auxiliares para control en la graficación */
int ban2 = 0;
int ban3=0;
int rec;
int salir; /* indicación para salir del SIU-PID */
int currx,curry; /* posiciones actuales de x y y */
int xsant = 0;  /* coordenada de x en pantalla para la muestra inmediata anterior */
int ysant = 0;  /* coordenada de y en pantalla para la muestra inmediata anterior */
int grafvel = 0;  /* bandera de indicación de requerimiento para grabar velocidad */
int cambiapar = 0; /* bandera de indicación de requerimiento pra cambiar parametros */
int numy; /* valor numerico de la muestra recibida en rpm */
int status;  /* variable empleada para conocer el estado del puerto serie (COM1) */
struct coord XY1[101]; /*almacenan las muestras anteriores para ir actualizando la grafica 2 */
struct coord XY2[101];

/* inicialización del modo gráfico */
initgraph(&gdriver, &gmode, "");

/* lectura del resultado de la inicialización */
errorcode = graphresult();

if(errorcode != grOk)  /* ocurrió un error */
{
    printf("Graphics error: %s\n", grapherrormsg(errorcode));
    printf("Press any key to halt:");
    getche();
    exit(1);             /* devuelve un mensaje de error y termina */
}

if((fp = fopen("trans.dat","w+")) == NULL){
   printf("Error al abrir el archivo de salida");
   exit(1);
}
if((fpp = fopen("param.dat","r+")) == NULL){
   printf("Error al abrir el archivo de parametros");
   exit(1);
}
af=getmaxx();     /* ancho físico del monitor */
hf=getmaxy();     /* alto físico del monitor */
setbkcolor(1);
Menu1();
/* MARCO SUPERIOR DERECHO */
/* coordenadas virtuales de los límites del rectangulo considerando una
   pantalla de 10000X10000 */
xvirtual1 = 5000;
yvirtual1 = 1000;
xvirtual2 = 9500;
yvirtual2 = 4500;
```

```
/* coordenadas para los ejes */
xaco1 = 0;
xaco2 = 10;  /* 10 seg */
yaco1 = 0;
yaco2 = VIRTUAL;  /* MAXVEL */
Marco(xaco1,yaco1,xaco2,yaco2,xvirtual1,yvirtual1,xvirtual2,yvirtual2,af,hf);
cuadricula(xvirtual1,yvirtual1,xvirtual2,yvirtual2);
/* MARCO INFERIOR DERECHO */
/* coordenadas virtuales de los límites del rectangulo considerando un
   monitor de 10000X10000 */
xvirtual1 = 5000;
yvirtual1 = 5500;   /* 5000 */
xvirtual2 = 9500;
yvirtual2 = 9000;
/* coordenadas para los ejes */
xaco1 = 0;
xaco2 = 10;  /* 10 seg */
yaco1 = 0;
yaco2 = VIRTUAL; /* MAXVEL */
Marco(xaco1,yaco1,xaco2,yaco2,xvirtual1,yvirtual1,xvirtual2,yvirtual2,af,hf);
cuadricula(xvirtual1,yvirtual1,xvirtual2,yvirtual2);
/* MARCO IZQUIERDO */
/* coordenadas virtuales de los límites del rectangulo considerando un
   monitor de 10000X10000 */
xvirtual1 = 500;
yvirtual1 = 1000;
xvirtual2 = 4500;
yvirtual2 = 9000;
Marco2(xvirtual1,yvirtual1,xvirtual2,yvirtual2,af,hf);

xvirtual1=5000;
yvirtual1=1000;
xvirtual2=9500;
yvirtual2=4500;
xaco1 = 0;
yaco1 = 0;
xaco2 = 10;
yaco2 = VIRTUAL;
N = xaco2*10; /* N=100 si xaco2 = 10seg */
dx = (float)(xaco2-xaco1)/(float)N; /* incremento en x */
if(dx<0) dx = -dx;
bioscom(0,SETTINGS,COM1);  /* Inicialización del COM1. Velocidad=9600 bauds, no paridad */
                           /* 1 bit de inicio, 8 bits de datos */
x = xaco1;

/* byte de inicio del protocolo (0xCC) */
rec=0xDD;
do{
  status=bioscom(3,0,COM1);
  if(status&DATA_READY)
    rec=bioscom(2,0,COM1)&0xFF;
    salir = Salir(fp,fpp);
    if(salir) exit(1);
}while(rec!=0xCC);

/* se lee la primera muestra */
c= 0;
while(c<2){
  status=bioscom(3,0,COM1);
  if(status&DATA_READY)
    abyte[c++]=bioscom(2,0,COM1)&0xFF;
}

/* si se recibe una muestra 0xCC,0xAA,0xAA indica que se reinicializo el sistema */
if(abyte[0]==0xAA){
  x=0;
  n=0;
  ban2=0;
  ban3=1;
  par=1;
  xsant=0;
  ysant=0;
  rec=0xDD;
  do{
    status=bioscom(3,0,COM1);
    if(status&DATA_READY)
        rec=bioscom(2,0,COM1)&0xFF;
  }while(rec!=0xCC);
  c= 0;
```

```
    while(c<2){
      status=bioscom(3,0,COM1);
      if(status&DATA_READY)
            abyte[c++]=bioscom(2,0,COM1)&0xFF;
    }
    cadm[0] = '\0';
    if(abyte[0]==0x00)
      strcpy(cadm,"00");
    else{
      itoa(abyte[0],str,16);
      strcpy(cadm,str);
    }
    if(abyte[1]==0x00)
      strcat(cadm,"00");
    else{
      itoa(abyte[1],str,16);
      if(abyte[1]<0xA){
            strcat(cadm,"0");
            strcat(cadm,str);
      }
      else
            strcat(cadm,str);
    }
    y = atoi(cadm);
    yant = y;
  }
  else{
    cadm[0] = '\0';
    if(abyte[0]==0x00)
      strcpy(cadm,"00");
    else{
      itoa(abyte[0],str,16);
      strcpy(cadm,str);
    }
    if(abyte[1]==0x00)
      strcat(cadm,"00");
    else{
      itoa(abyte[1],str,16);
      if(abyte[1]<0xA){
         strcat(cadm,"0");
         strcat(cadm,str);
      }
      else
       strcat(cadm,str);
    }

    /* "y"contiene el valor de la muestra leida */
    y = atoi(cadm);
    yant = y;
  }
  numy=(3720*(float)y)/(float)9999;
  itoa(numy,texty,10);
  Virtual(6800,5000,VIRTUAL,VIRTUAL,&xs1,&ys1,af,hf);
  strcat(texty," [RPM]");
  setcolor(WHITE);
  outtextxy(xs1,ys1,texty);
  Mundialesxy(xvirtual1,yvirtual1,xvirtual2,yvirtual2,xaco1,yaco1,xaco2,yaco2,x,y,&xs,&ys,af,hf);
  moveto(xs,ys);
  cont_muestra = 0;
  ban1 = 0;

  while(1){
    if(cont_muestra < N){
      xvirtual1=5000;
      yvirtual1=1000;
      xvirtual2=9500;
      yvirtual2=4500;
      x+=dx;
      /* protocolo */
      rec=0xDD;
      do{
            status=bioscom(3,0,COM1);
            if(status&DATA_READY)
              rec=bioscom(2,0,COM1)&0xFF;
          }while(rec!=0xCC);

      /* recepcion de una de las 100 primeras muestras */
      c= 0;
      while(c<2){
```

```
      status=bioscom(3,0,COM1);
      if(status&DATA_READY)
        abyte[c++]=bioscom(2,0,COM1)&0xFF;
}
cadm[0] = '\0';

/* si se recibe una muestra 0xCC,0xAA,0xAA indica que se reinicializo el sistema */
if(abyte[0]==0xAA){
    cont_muestra=0;
    x=0;
    n=0;
    ban1=0;
    ban2=0;
    ban3=1;
    par=1;
    xsant=0;
    ysant=0;
    /* MARCO SUPERIOR DERECHO */
    clrMarco(5000,1000,9500,4500);
    Marco(0,0,10,VIRTUAL,5000,1000,9500,4500,af,hf);
    cuadricula(5000,1000,9500,4500);
    /* MARCO INFERIOR DERECHO */
    clrMarco(5000,5500/*5000*/,9500,9000);
    Marco(0,0,10,VIRTUAL,5000,5500,9500,9000,af,hf);
    cuadricula(5000,5500,9500,9000);
    clrMarco1(500,1000,4500,9000);
    rec=0xDD;
    do{
      status=bioscom(3,0,COM1);
      if(status&DATA_READY)
        rec=bioscom(2,0,COM1)&0xFF;
    }while(rec!=0xCC);
    c= 0;
    while(c<2){
      status=bioscom(3,0,COM1);
      if(status&DATA_READY)
        abyte[c++]=bioscom(2,0,COM1)&0xFF;
    }
    cadm[0] = '\0';
    if(abyte[0]==0x00)
      strcpy(cadm,"00");
   else{
      itoa(abyte[0],str,16);
      strcpy(cadm,str);
    }
    if(abyte[1]==0x00)
      strcat(cadm,"00");
    else{
      itoa(abyte[1],str,16);
      if(abyte[1]<0xA){
        strcat(cadm,"0");
        strcat(cadm,str);
      }
      else
        strcat(cadm,str);
    }
    y = atoi(cadm);
    yant = y;
    Mundialesxy(5000,1000,9500,4500,xaco1,yaco1,xaco2,yaco2,x,y,&xs,&ys,af,hf);
    moveto(xs,ys);
    reseto++;
    continue;
 }

/* si no se recibio la muestra 0xCC,0xAA,0xAA continua aqui */
if(abyte[0]==0x00)
   strcpy(cadm,"00");
else{
   itoa(abyte[0],str,16);
   strcpy(cadm,str);
}
if(abyte[1]==0x00)
   strcat(cadm,"00");
else{
   itoa(abyte[1],str,16);
   if(abyte[1]<0xA){
     strcat(cadm,"0");
     strcat(cadm,str);
   }
```

```
        else
          strcat(cadm,str);
      }
      yant = y;
      y = atoi(cadm);

        /*   se   grafica   una   de   las   100   primeras   muestras   */
Mundialesxy(xvirtual1,yvirtual1,xvirtual2,yvirtual2,xaco1,yaco1,xaco2,yaco2,x,y,&xs,&ys,af,hf);
      lineto(xs,ys);
      xsant=xs;
      ysant=ys;

       /* Presentacion en pantalla de la velocidad actual */

      if(((yant>=(y+3))||(yant<=(y-3)))){
         numy=(3720*(float)y)/(float)9999;
         itoa(numy,texty,10);
         Virtual(6800,5000,VIRTUAL,VIRTUAL,&xs1,&ys1,af,hf);
         Virtual(7300,5400,VIRTUAL,VIRTUAL,&xs2,&ys2,af,hf);
         setviewport(xs1,ys1,xs2,ys2,1);
         clearviewport();
         setviewport(0,0,af,hf,1);
         outtextxy(xs1,ys1,texty);
         moveto(xsant,ysant);
      }

      /* almacena en el archivo una de las 100 primeras muestras */
      /* (la que corresponde a cont_muestra */

      while(strlen(texty)<4)
        strcat(texty," ");

       fputs(texty,fp);

       if(x>=xaco2) x=0;

     } /* end if cont_muestra < 100 */

     /* GRAFICA 2. De la muestra 101 en adelante */

     else{
       xvirtual1=5000;
       yvirtual1=5500;
       xvirtual2=9500;
       yvirtual2=9000;
       x+=dx;
       /* protocolo */
       rec = 0xDD;
       do{
          status=bioscom(3,0,COM1);
          if(status&DATA_READY)
            rec=bioscom(2,0,COM1)&0xFF;
       }while(rec!=0xCC);

       /* recepcion de una muestra */
       c= 0;
       while(c<2){
          status=bioscom(3,0,COM1);
          if(status&DATA_RDADY)
            abyte[c++]=bioscom(2,0,COM1)&0xFF;
       }

       cadm[0] = '\0';

       /* si la muestra recibida fue 0xCC,0xAA,0xAA se presento una reinicializacion */
       /* por lo cual es necesario empezar a graficar en la GRAFICA 1 */

       if(abyte[0]==0xAA){
          cont_muestra=0;
          x=0;
          n=0;
          ban1=0;
          ban2=0;
          ban3=1;
          par=1;
          xsant=0;
          ysant=0;
          /* MARCO SUPERIOR DERECHO */
          clrMarco(5000,1000,9500,4500);
```

```
    Marco(0,0,10,VIRTUAL,5000,1000,9500,4500,af,hf);
    cuadricula(5000,1000,9500,4500);
    /* MARCO INFERIOR DERECHO */
    clrMarco(5000,5500,9500,9000);
    Marco(0,0,10,VIRTUAL,5000,5500,9500,9000,af,hf);
    cuadricula(5000,5500,9500,9000);
    clrMarco1(500,1000,4500,9000);
    rec=0xDD;
    do{
      status=bioscom(3,0,COM1);
      if(status&DATA_READY)
        rec=bioscom(2,0,COM1)&0xFF;
    }while(rec!=0xCC);
    c=0;
    while(c<2){
      status=bioscom(3,0,COM1);
      if(status&DATA_READY)
        abyte[c++]=bioscom(2,0,COM1)&0xFF;
    }
    cadm[0] = '\0';
    if(abyte[0]==0x00)
      strcpy(cadm,"00");
  else{
      itoa(abyte[0],str,16);
      strcpy(cadm,str);
  }
    if(abyte[1]==0x00)
      strcat(cadm,"00");
    else{
      itoa(abyte[1],str,16);
      if(abyte[1]<0xA){
        strcat(cadm,"0");
        strcat(cadm,str);
      }
      else
        strcat(cadm,str);
    }
    y = atoi(cadm);
    yant = y;
    Mundialesxy(5000,1000,9500,4500,xaco1,yaco1,xaco2,yaco2,x,y,&xs,&ys,af,hf);
    moveto(xs,ys);
    resets++;
    continue;
}

/* si la muestra es diferente de 0xCC,0xAA,0xAA continua */

if(abyte[0]==0x00)
    strcpy(cadm,"00");
else{
    itoa(abyte[0],str,16);
    strcpy(cadm,str);
}
if(abyte[1]==0x00)
    strcat(cadm,"00");
  else{
      itoa(abyte[1],str,16);
      if(abyte[1]<0xA){
      strcat(cadm,"0");
      strcat(cadm,str);
    }
else
    strcat(cadm,str);
}

/* actualiza yant y "y" */
yant = y;
y = atoi(cadm);

/* Si ban1=0 indica que es necesario realizar un movimiento hacia (0,y) */
/* sin trazar linea, es decir al tiempo t=0 de la GRAFICA2 */
if(ban1==0){
    x = 0;
  Mundialesxy(xvirtual1,yvirtual1,xvirtual2,yvirtual2,xaco1,yaco1,xaco2,yaco2,x,y,
              &xs,&ys,af,hf);
    moveto(xs,ys);
    xsant=xs;
    ysant=ys;
    ban1=1;
```

```
    if((fmod((double)par,(double)2)==0)){
      XY2[n].x=xs;
      XY2[n].y=ys;
    }
    else{
      XY1[n].x=xs;
      XY1[n].y=ys;
    }
    ban3=1;
    n++;
  } /* end if ban==1 */

/* si ban3 = 1 no se debe dibujar ninguna linea, solamente realizar movimiento */
if(ban3==0){
  Mundialesxy(xvirtual1,yvirtual1,xvirtual2,yvirtual2,xaco1,yaco1,xaco2,yaco2,x,y,
              &xs,&ys,af,hf);
    /* si ban2=1 ya se tienen valores correctos en el arreglo XY1 para */
    /* continuar la graficación */
    if(ban2==1){
      setcolor(1);
      if((fmod((double)par,(double)2)==0)){
        if(n>1){
           /* borra la muestra correspondiente de la grafica anterior */
            moveto(XY1[n-1].x,XY1[n-1].y);
            lineto(XY1[n].x,XY1[n].y);
            /* restaura la linea con el color normal */
            setcolor(LIGHTGREEN);
            moveto(XY2[n-2].x,XY2[n-2].y);
            lineto(XY2[n-1].x,XY2[n-1].y);
        }
        else{
          moveto(XY1[n-1].x,XY1[n-1].y);
          lineto(XY1[n].x,XY1[n].y);
          moveto(XY1[99].x,XY1[99].y);
          lineto(XY1[100].x,XY1[100].y);
         setcolor(LIGHTGREEN);
          moveto(XY1[99].x,XY1[99].y);
         lineto(XY1[100].x,XY1[100].y);
        }
      }
      else{
        if(n>1){
            moveto(XY2[n-1].x,XY2[n-1].y);
            lineto(XY2[n].x,XY2[n].y);
            setcolor(LIGHTGREEN);
            moveto(XY1[n-2].x,XY1[n-2].y);
            lineto(XY1[n-1].x,XY1[n-1].y);
        }
        else{
            moveto(XY2[n-1].x,XY2[n-1].y);
            lineto(XY2[n].x,XY2[n].y);
            moveto(XY2[99].x,XY2[99].y);
            lineto(XY2[100].x,XY2[100].y);
            setcolor(LIGHTGREEN);
            moveto(XY2[99].x,XY2[99].y);
            lineto(XY2[100].x,XY2[100].y);
        }
      }
      /* grafica la muestra actual en color rojo */
      moveto(xsant,ysant);
      setcolor(LIGHTRED);
      lineto(xs,ys);
      setcolor(WHITE);
    } /* end if ban2 = 1 */

    /* al graficar por primera vez en la GRAPICA2, no es necesario */
    /* borrar muestras de graficas anteriores */
    else{
      if(n>1){
        setcolor(LIGHTGREEN);
        moveto(XY1[n-2].x,XY1[n-2].y);
        lineto(XY1[n-1].x,XY1[n-1].y);
      }
      moveto(xsant,ysant);
      setcolor(LIGHTRED);
      lineto(xs,ys);
      setcolor(WHITE);
    } /* end else if ban2=1 */
```

```
      /* actualizacion de valores para la graficación de la siguiente muestra */
      xsant=xs;
      ysant=ys;
      if(fmod((double)par,(double)2)==0){
        XY2[n].x=xsant;
        XY2[n].y=ysant;
      }
      else{
        XY1[n].x=xsant;
        XY1[n].y=ysant;
     }
      n++;
   } /* end if ban3 = 0 */

   ban3=0;

   /* En este caso ya se graficaron 100 muestras mas y es necesario iniciar */
   /* en x = 0 */

   if(x>=xaco2){
      /* Restaurar el espacio de trabajo de la grafica 2 */
      xvirtual1 = 5000;
      yvirtual1 = 5500;
      xvirtual2 = 9500;
      yvirtual2 = 9000;
      /* coordenadas para los ejes */
      xaco1 = 0;
      xaco2 = 10;  /* 10 seg */
      yaco1 = 0;
      yaco2 = VIRTUAL; /* MAXVEL */
      Marco(xaco1,yaco1,xaco2,yaco2,xvirtual1,yvirtual1,xvirtual2,yvirtual2,af,hf);
      cuadricula(xvirtual1,yvirtual1,xvirtual2,yvirtual2);
      x=-0.1;
      ban1=0;
      par++;
      n=0;
      ban2=1;
      ban3=1;
   }
} /* end else de la GRAFICA 2 */

cont_muestra++;

/* Si se solicito grabar velocidad, cuando y<10, envia los codigos necesarios
al microcontrolador para que se realice la grabación */
if(y<1000 && grafvel==1){
  Virtual(1100,1500,VIRTUAL,VIRTUAL,&xs1,&ys1,af,hf);
  outtextxy(xs1,ys1,"GRABAR VELOCIDAD");
  transmitir1(0x88);
  grafvel = 0;
  moveto(xsant,ysant);
  sleep(1); /* tiempo de grabación */
  Mensaje2(); /* puede apagar o reinicializar el sistema */
}

/* Si se solicito cambiar parametros, cuando y<10, la PC envia los codigos
necesarios al microcontrolador para que se graben los parametros en EEPROM */
if(y<1000 && cambiapar==1){
  nuevos_parametros(fpp);
  cambiapar=0;
  Menu7("[redacted]","F1 Velocidad F2
        Parámetros  F3 Gráfica  F4 Archivo  F10 SALIR");
  moveto(xsant,ysant);
  sleep(1);
  Mensaje2(); /* puede apagar o reinicializar el sistema */
}

/* Presentacion en pantalla de la velocidad actual */
if(((ysant>=(y+3))||(ysant<=(y-3)))&&ban1){
  numy=(3720*(float)y)/(float)9999;
  itoa(numy,texty,10);
  Virtual(6800,5000,VIRTUAL,VIRTUAL,&xs1,&ys1,af,hf);
  Virtual(7300,5400,VIRTUAL,VIRTUAL,&xs2,&ys2,af,hf);
  setviewport(xs1,ys1,xs2,ys2,1);
  clearviewport();
  setviewport(0,0,af,hf,1);
  outtextxy(xs1,ys1,texty);
  moveto(xsant,ysant);
}
```

```
/* En caso de presionar alguna tecla del menú se realiza la accion correspondiente */
if(kbhit()){
  tecla=getch();
  fflush(stdin);
  if(tecla==0){
    tecla=getch();
    fflush(stdin);
    switch(tecla){
      /* tecla F1  VELOCIDAD */
      case ';': Menu7("F1 Velocidad  F2 Parámetros  F3 Gráfica  F4 Archivo  F10 SALIR","<1>
                  Cambia Velocidad     <2> Graba Velocidad     <ESC> Menu Anterior");
                tecla2 = getch();
                fflush(stdin);
                switch(tecla2){
                  case '1': VELRAM();   /* Cambiar velocidad */
                            break;
                  case '2': adv=Advertencia2();
                            if(adv=='S'){
                              transmitir1(0x55);  /* Grabar velocidad */
                              grafvel=1;
                              Mensaje("Espere...");
                              adv='N';
                            }
                            break;
                }
                Menu7("[redacted]","F1
                  Velocidad  F2 Parámetros  F3 Gráfica  F4 Archivo  F10 SALIR");
                clrMarco1(500,1000,4500,9000);
                moveto(xsant,ysant);
                break;
      /* tecla F2  PARAMETROS */
      case '<': Menu7("F1 Velocidad  F2 Parámetros  F3 Gráfica  F4 Archivo  F10 SALIR","<1>
                   Cambiar Parametros  <2> Ver parametros  <ESC> Menu Anterior");
                tecla2 = getch();
                fflush(stdin);
                switch(tecla2){
                  case '1': adv=Advertencia();
                            if(adv=='S'){
                              transmitir1(0x55);
                              cambiapar=1;
                              Mensaje("Espere...");
                              adv='N';
                            }
                            break;
                  case '2': rewind(fpp);
                            strcpy(KP,"Kp: ");
                            strcpy(TD,"Td: ");
                            strcpy(TI,"Ti: ");
                            ch=fgetc(fpp);
                            i=4;
                            while(ch!='\n'){
                              KP[i++]=ch;
                              ch = fgetc(fpp);
                            }
                            KP[i]='\0';
                            ch=getc(fpp);
                            i=4;
                            while(ch!='\n'){
                              TD[i++]=ch;
                              ch = fgetc(fpp);
                            }
                            TD[i]='\0';
                            ch=getc(fpp);
                            i=4;
                            while(ch!='\n'){
                              TI[i++]=ch;
                              ch = fgetc(fpp);
                            }
                            TI[i]='\0';
                            Virtual(1200,1500,VIRTUAL,VIRTUAL,&xs1,&ys1,af,hf);
                            outtextxy(xs1,ys1,"PARAMETROS ACTUALES");
                            Virtual(1800,2000,VIRTUAL,VIRTUAL,&xs1,&ys1,af,hf);
                            outtextxy(xs1,ys1,KP);
                            Virtual(1800,2500,VIRTUAL,VIRTUAL,&xs1,&ys1,af,hf);
                            outtextxy(xs1,ys1,TD);
                            Virtual(1800,3000,VIRTUAL,VIRTUAL,&xs1,&ys1,af,hf);
                            outtextxy(xs1,ys1,TI);
                            break;
```

```
                    Menu7("[redacted]
                          ,"F1 Velocidad  F2 Parámetros  F3 Gráfica  F4 Archivo  F10 SALIR");
                    moveto(xsant,ysant);
                    break;

          /* tecla f3   GRAFICA */
          case '=':Virtual(1200,1500,VIRTUAL,VIRTUAL,&xs1,&ys1,af,hf);
                    outtextxy(xs1,ys1,"CONGELAR GRAFICA");
                    Virtual(800,1800,VIRTUAL,VIRTUAL,&xs1,&ys1,af,hf);
                    settextstyle(SMALL_FONT,HORIZ_DIR,4);
                    outtextxy(xs1,ys1,"PRESIONE <ENTER> PARA CONTINUAR");
                    getch();
                    settextstyle(DEFAULT_FONT,HORIZ_DIR,1);
                    clrMarco1(500,1000,4500,9000);
                    moveto(xsant,ysant);
                    break;
          /* tecla f4 */
          case '>': Mostrar_Archivo(fp);
                    moveto(xsant,ysant);
                    break;
          /* tecla F10 */
          case 'D': fclose(fp);
                    fclose(fpp);
                    restorecrtmode();
                    exit(1);
        } /* end del switch */
      } /* end del if(tecla==0) */
    }/* end del if kbhit() */

  } /* end del while(1) */

} /* end del main */


/* ---------------------------  REGLA3 ---------------------------------- */
/* Esta función sirve para obtener una escala adecuada para la graficación */

float Regla3(float uno,long dos)
{
        return ((float)dos/uno);
}

/* ------------------------- MUNDIALESXY ------------------------------- */
/* Con esta función se posiciona un punto en coordenadas mundiales, es decir, se define un
rectángulo de trabajo y unos ejes dentro del mismo, para graficar un punto en coordenadas reales
(xr,yr) y esta función toma en cuenta una pantalla virtual para entregarnos un punto físico que
quede dentro del rectángulo de trabajo y que corresponda al punto (xr,yr) */

void Mundialesxy(int x1,int y1,int x2,int y2,float xmin,float ymin,float xmax,
           float ymax,float xr,int yr,int *xs,int *ys,int af,int hf)
{
  /* x1,y1,x2,y2 definen las esquinas del rectángulo de trabajo */
  /* xmin,ymin,xmax,ymax definen los límites de los ejes */
  /* xr,yr son las coordenadas de un punto a graficar */
  /* xs,ys son las coordenadas físicas de la pantalla para graficar el punto */
  /* af, hf son el ancho físico y el alto físico de la pantalla */

  int apx,apy; /* Ancho del dispositivo en X y en Y */
  float amx,amy; /* Ancho Mundial (el deseado) en X y en Y  */
  int xs1,ys1;  /* Auxiliares para recibir coordenadas virtuales */
  float wx,wy;  /* Factores de escala en X y en Y */

  apx=x2-x1;
  apy=y2-y1;
  amx=xmax-xmin;
  amy=ymax-ymin;
  wx=(apx/amx);  /* Escalas en X y en Y */
  wy=(apy/amy);

  /* Funciones de Transformación */

  *xs=wx*(xr-xmin)+x1+0.5;
  *ys=wy*(ymax-yr)+y1+0.5;

  /* Ya teniendo el punto en coordenadas mundiales (dento del rectángulo
  de trabajo pero en coordenadas virtuales, entre 0 y 10000) se obtiene
  el punto físico en la pantalla virtual que se está manejando */

  Virtual(*xs,*ys,VIRTUAL,VIRTUAL,&xs1,&ys1,af,hf);
```

```
  *xs=xs1;
  *ys=ys1;
}

/* ---------------------------- VIRTUAL ---------------------------------- */
/* Esta función pasa de un punto en coordenas virtuales a coordenadas
       reales de la pantalla*/

void Virtual(long xv,long yv,long hv,long av,int *XS,int *YS,long af,long hf){

  long XSc,XSr,YSc,YSr;

  XSc=(long)((xv*af)/av);
  XSr=(long)(af*xv);
  XSr=XSr%av;
   if(VIRTUAL<2*XSr)
      *XS=++XSc;
   else
      *XS=XSc;
  YSc=(hf*yv)/hv;
  YSr=(hf*yv)%hv;
  if(VIRTUAL<2*YSr)
    *YS=++YSc;
  else
    *YS=YSc;
}

/* ------------------------------ MENU ----------------------------------- */

/* Menu inicial y titulo del sistema de interfaz con el usuario */

void Menu1(){

  int blanco = getcolor();
  int rojo = 8;
  char pattern[8];

  getfillpattern(pattern);
  setcolor(rojo);
  rectangle(getmaxx()/40+6,getmaxy()-(getmaxy()/14)-6,getmaxx()-(getmaxx()/40)+6,getmaxy()-10);
  setfillpattern(pattern,rojo);
  floodfill(getmaxx()/40+10,getmaxy()-getmaxy()/14-2,rojo);
  setcolor(blanco);
  rectangle(getmaxx()/40,getmaxy()-(getmaxy()/14),getmaxx()-(getmaxx()/40),getmaxy()-4);
  setfillpattern(pattern,blanco);
  floodfill(getmaxx()/2,getmaxy()-10,blanco);
  rectangle(getmaxx()/20,getmaxy()/55,getmaxx()-(getmaxx()/20),getmaxy()/50+30);
  setfillpattern(pattern,rojo);
  floodfill(getmaxx()/20+5,getmaxy()/55+5,blanco);
  setcolor(blanco);
  settextstyle(TRIPLEX_FONT,HORIZ_DIR,2);
  settextjustify(CENTER_TEXT,CENTER_TEXT);
  outtextxy(getmaxx()/2,20," S I U -  P I D ");
  settextstyle(DEFAULT_FONT,HORIZ_DIR,1);
  setcolor(0);
  settextjustify(CENTER_TEXT,CENTER_TEXT);
  outtextxy(getmaxx()/2,getmaxy()-20,"F1 Velocidad  F2 Parámetros  F3 Gráfica  F4 Archivo  F10
SALIR");
  setcolor(blanco);
  settextjustify(LEFT_TEXT,TOP_TEXT);
}

/* ---------------------------- MENU7 ---------------------------------- */

/* Función para cambiar el menu actual en el sistema */

void Menu7(char *strold,char *strnew){

  int blanco = getcolor();
  int rojo = 8;
  char pattern[8];

  getfillpattern(pattern);
  setcolor(rojo);
  rectangle(getmaxx()/40+6,getmaxy()-(getmaxy()/14)-6,getmaxx()-(getmaxx()/40)+6,getmaxy()-10);
  setfillpattern(pattern,rojo);
  floodfill(getmaxx()/40+10,getmaxy()-getmaxy()/14-2,rojo);
  setcolor(blanco);
  rectangle(getmaxx()/40,getmaxy()-(getmaxy()/14),getmaxx()-(getmaxx()/40),getmaxy()-4);
```

```
  setfillpattern(pattern,blanco);
  floodfill(getmaxx()/2,getmaxy()-10,blanco);
  settextstyle(DEFAULT_FONT,HORIZ_DIR,1);
  setcolor(blanco);
  settextjustify(CENTER_TEXT,CENTER_TEXT);
  outtextxy(getmaxx()/2,getmaxy()-20,strold);
  settextstyle(DEFAULT_FONT,HORIZ_DIR,1);
  setcolor(0);
  settextjustify(CENTER_TEXT,CENTER_TEXT);
  outtextxy(getmaxx()/2,getmaxy()-20,strnew);
  setcolor(blanco);
  settextjustify(LEFT_TEXT,TOP_TEXT);
}

/* ------------------------- MENSAJE ---------------------------------- */
/* Función para enviar el mensaje contenido en strmen a pantalla */

void Mensaje(char *strmen){

  int x1,y1,x2,y2,af,hf;

  af = getmaxx();
  hf = getmaxy();

  Virtual(1000,7000,VIRTUAL,VIRTUAL,&x1,&y1,af,hf);
  Virtual(4000,8500,VIRTUAL,VIRTUAL,&x2,&y2,af,hf);
  rectangle(x1,y1,x2,y2);
  rectangle(x1+1,y1+1,x2+1,y2+1);
  setviewport(x1+1,y1+1,x2-1,y2-1,1);
  clearviewport();
  setviewport(0,0,af,hf,1);
  setfillstyle(SOLID_FILL,LIGHTGRAY);
  floodfill(x1+2,y1+2,WHITE);
  setcolor(WHITE);
  settextstyle(DEFAULT_FONT,HORIZ_DIR,1);
  Virtual(1500,7400,VIRTUAL,VIRTUAL,&x1,&y1,af,hf);
  outtextxy(x1+5,y1+10,strmen);
  settextjustify(LEFT_TEXT,TOP_TEXT);
}

/* ------------------------ MENSAJE2 ---------------------------------- */
/* Función para enviar un mensaje al usuario */

void Mensaje2(void){

  int x1,y1,x2,y2,af,hf;

  af = getmaxx();
  hf = getmaxy();
  Virtual(1000,7000,VIRTUAL,VIRTUAL,&x1,&y1,af,hf);
  Virtual(4000,8500,VIRTUAL,VIRTUAL,&x2,&y2,af,hf);
  rectangle(x1,y1,x2,y2);
  rectangle(x1+1,y1+1,x2+1,y2+1);
  setviewport(x1+1,y1+1,x2-1,y2-1,1);
  clearviewport();
  setviewport(0,0,af,hf,1);
  setfillstyle(SOLID_FILL,LIGHTGRAY);
  floodfill(x1+2,y1+2,WHITE);
  setcolor(WHITE);
  settextstyle(DEFAULT_FONT,HORIZ_DIR,1);
  Virtual(1100,7300,VIRTUAL,VIRTUAL,&x1,&y1,af,hf);
  outtextxy(x1,y1,"AHORA debe apagar o");
  Virtual(1100,7600,VIRTUAL,VIRTUAL,&x1,&y1,af,hf);
  outtextxy(x1,y1,"reinicializar el");
  Virtual(1100,7900,VIRTUAL,VIRTUAL,&x1,&y1,af,hf);
  outtextxy(x1,y1,"sistema.");
  settextjustify(LEFT_TEXT,TOP_TEXT);
}

/* ------------------------ ADVERTENCIA ------------------------------- */
/* Función para enviar un mensaje de advertencia a pantalla cuando se van a
   cambiar parametros  */

char Advertencia(void){

  char resp;
  int x1,y1,x2,y2,af,hf;

  af = getmaxx();
```

```
  hf = getmaxy();
  Virtual(1000,7000,VIRTUAL,VIRTUAL,&x1,&y1,af,hf);
  Virtual(4000,8500,VIRTUAL,VIRTUAL,&x2,&y2,af,hf);
  rectangle(x1,y1,x2,y2);
  rectangle(x1+1,y1+1,x2+1,y2+1);
  setviewport(x1+1,y1+1,x2-1,y2-1,1);
  clearviewport();
  setviewport(0,0,af,hf,1);
  setfillstyle(SOLID_FILL,LIGHTGRAY);
  floodfill(x1+2,y1+2,WHITE);
  setcolor(WHITE);
  settextstyle(DEFAULT_FONT,HORIZ_DIR,1);
  Virtual(1100,7100,VIRTUAL,VIRTUAL,&x1,&y1,af,hf);
  outtextxy(x1,y1,"Advertencia:");
  Virtual(1100,7400,VIRTUAL,VIRTUAL,&x1,&y1,af,hf);
  outtextxy(x1,y1,"La velocidad actual");
  Virtual(1100,7600,VIRTUAL,VIRTUAL,&x1,&y1,af,hf);
  outtextxy(x1,y1,"se perdera a menos que");
  Virtual(1100,7800,VIRTUAL,VIRTUAL,&x1,&y1,af,hf);
  outtextxy(x1,y1,"haya sido previamente");
  Virtual(1100,8000,VIRTUAL,VIRTUAL,&x1,&y1,af,hf);
  outtextxy(x1,y1,"grabada");
  Virtual(1300,8300,VIRTUAL,VIRTUAL,&x1,&y1,af,hf);
  outtextxy(x1,y1,"¿Continuar?(S/N): ");
  resp=toupper(getch());
  clrMarco1(500,1000,4500,9000);
  settextjustify(LEFT_TEXT,TOP_TEXT);
  return resp;
}

/* ---------------------------- ADVERTENCIA2 ------------------------- */
/* Función para enviar un mensaje de advertencia a pantalla cuando se va a
   grabar velocidad */

char Advertencia2(void){

  char resp;
  int x1,y1,x2,y2,af,hf;

  af = getmaxx();
  hf = getmaxy();
  Virtual(1000,7000,VIRTUAL,VIRTUAL,&x1,&y1,af,hf);
  Virtual(4000,8500,VIRTUAL,VIRTUAL,&x2,&y2,af,hf);
  rectangle(x1,y1,x2,y2);
  rectangle(x1+1,y1+1,x2+1,y2+1);
  setviewport(x1+1,y1+1,x2-1,y2-1,1);
  clearviewport();
  setviewport(0,0,af,hf,1);
  setfillstyle(SOLID_FILL,LIGHTGRAY);
  floodfill(x1+2,y1+2,WHITE);
  setcolor(WHITE);
  settextstyle(DEFAULT_FONT,HORIZ_DIR,1);
  Virtual(1100,7100,VIRTUAL,VIRTUAL,&x1,&y1,af,hf);
  outtextxy(x1,y1,"Advertencia:");
  Virtual(1100,7400,VIRTUAL,VIRTUAL,&x1,&y1,af,hf);
  outtextxy(x1,y1,"La velocidad del");
  Virtual(1100,7600,VIRTUAL,VIRTUAL,&x1,&y1,af,hf);
  outtextxy(x1,y1,"motor se reducirá a");
  Virtual(1100,7800,VIRTUAL,VIRTUAL,&x1,&y1,af,hf);
  outtextxy(x1,y1,"0 RPM");
  Virtual(1300,8300,VIRTUAL,VIRTUAL,&x1,&y1,af,hf);
  outtextxy(x1,y1,"¿Continuar?(S/N): ");
  resp=toupper(getch());
  clrMarco1(500,1000,4500,9000);
  settextjustify(LEFT_TEXT,TOP_TEXT);
  return resp;
}

/* ---------------------------- MARCO ---------------------------------- */
/* Esta función se encarga de el dibujo del marco junto con los ejes */

void Marco(float xaco1,float yaco1,float xaco2,float yaco2,int xvirtual1,int yvirtual1,int
xvirtual2,
         int yvirtual2,int af,int hf){

  float escalax,escalay;
  int xt1,yt1,xt2,yt2;
  char *cadena,*buf;
  char buff[10];
```

```
  int x1 = xvirtual1;
  int x2 = xvirtual2;
  int y1 = yvirtual1;
  int y2 = yvirtual2;

  Virtual(x1,y1,VIRTUAL,VIRTUAL,&x1,&y1,af,hf);
  Virtual(x2,y2,VIRTUAL,VIRTUAL,&x2,&y2,af,hf);
  rectangle(x1-1,y1-1,x2+1,y2+1);
  escalax=Regla3(xaco2-xaco1,xvirtual2-xvirtual1);
  escalay=Regla3(yaco2-yaco1,yvirtual2-yvirtual1);
  buf=(char *)malloc(10);
  itoa(MAXVEL,buff,10);
  strcat(buff,"[rpm]");
  outtextxy(x1+3,y1+3,buff);
  cadena=gcvt(yaco1,3,buf);
  outtextxy(x1+3,y2-9,cadena);
  cadena=gcvt(xaco1,3,buf);
  outtextxy(x1+3,y2+2,cadena);
  cadena=gcvt(xaco2,3,buf);
  strcat(cadena,"[s]");
  outtextxy(x2-9,y2+3,cadena);
}

/* ------------------------- MARCO2 ------------------------------------ */
/* Función para dibujar un rectángulo en la pantalla virtual */

void Marco2(int xvirtual1,int yvirtual1,int xvirtual2,int yvirtual2,int af,int hf){

  int x1 = xvirtual1;
  int x2 = xvirtual2;
  int y1 = yvirtual1;
  int y2 = yvirtual2;

  Virtual(x1,y1,VIRTUAL,VIRTUAL,&x1,&y1,af,hf);
  Virtual(x2,y2,VIRTUAL,VIRTUAL,&x2,&y2,af,hf);
  rectangle(x1,y1,x2,y2);
}

/* ----------------------- CLRMARCO ---------------------------------- */
/* Limpia el espacio de trabajo limitado por x1,y1,x2,y2. Se usa para limiar
   los marcos de la derecha porque esta función restaura el rectangulo que
   delimita el espacio de trabajo en estos marcos */

void clrMarco(int x1, int y1, int x2, int y2){

  int af,hf;
  int xvirtual1=x1;
  int yvirtual1=y1;
  int xvirtual2=x2;
  int yvirtual2=y2;

   af = getmaxx();
   hf = getmaxy();
  Virtual(x1,y1,VIRTUAL,VIRTUAL,&x1,&y1,af,hf);
  Virtual(x2,y2,VIRTUAL,VIRTUAL,&x2,&y2,af,hf);
  setviewport(x1,y1,x2,y2,1);
  clearviewport();
  setviewport(0,0,af,hf,1);
  rectangle(x1-1,y1-1,x2+1,y2+1);
  cuadricula(xvirtual1,yvirtual1,xvirtual2,yvirtual2);
}

/* -------------------------- CLRMARCO1 ------------------------------- */
/* Limpia el espacio de trabajo limitado por x1,y1,x2,y2 sin restaurar el
   rectangulo que limita este espacio */

void clrMarco1(int x1, int y1, int x2, int y2){

  int af,hf;

   af = getmaxx();
   hf = getmaxy();
  Virtual(x1,y1,VIRTUAL,VIRTUAL,&x1,&y1,af,hf);
  Virtual(x2,y2,VIRTUAL,VIRTUAL,&x2,&y2,af,hf);
  setviewport(x1+1,y1+1,x2-1,y2-1,1);
  clearviewport();
  setviewport(0,0,af,hf,1);
}
```

```
/* ------------------------------ CUADRICULA ------------------------------ */
/* Función para dibujar una cuadricula para facilitar la lectura en las graficas */

void cuadricula(int x1, int y1, int x2, int y2){

  int af,hf;
  int xs1,ys1,xs2,ys2;
  int pos;

   af = getmaxx();
   hf = getmaxy();
  setlinestyle(1,1,1);

  /* lineas horizontales */
  for(pos=y1+350;pos<=y2-350;pos+=350){
      Virtual(x1,pos,VIRTUAL,VIRTUAL,&xs1,&ys1,af,hf);
      moveto(xs1,ys1);
      Virtual(x2,pos,VIRTUAL,VIRTUAL,&xs2,&ys2,af,hf);
      lineto(xs2,ys2);
  }
  /* lineas verticales */
  for(pos=x1+450;pos<=x2-450;pos+=450){
      Virtual(pos,y1,VIRTUAL,VIRTUAL,&xs1,&ys1,af,hf);
      moveto(xs1,ys1);
      Virtual(pos,y2,VIRTUAL,VIRTUAL,&xs2,&ys2,af,hf);
      lineto(xs2,ys2);
  }
  setlinestyle(0,1,1);
}

/* ------------------------ GETENTERO ---------------------------------- */

int getentero(int x, int y){

  int i;
  char entero[11];
  char caracter;
  int numero;
  char despliegue[2];

  moveto(x,y);
  i = 0;
  caracter = getch();
  while((caracter != '\r')&&(caracter != '.')&&(i<4)){
    if(caracter==ESC) return(-1);
    despliegue[0] = caracter;
    despliegue[1] = '\0';
    outtextxy(x+=10,y,despliegue);
    entero[i]=caracter;
    caracter = getch();
    i++;
  }
  entero[i]='\0';
  numero = atoi(entero);
  return numero;
}

/* ---------------------------- GETFLOT ------------------------------------ */
/* Función que permite leer un número flotante de teclado en modo grafico,
   la posición para comenzar la lectura del dato esta dada por (x,y) */

float getflot(int x, int y){

  int i;
  char flotante[11];
  char caracter;
  float numero;
  char despliegue[2];

  moveto(x,y);
  i = 0;
  caracter = getch();
  while((caracter != '\r')&&(i<9)){
    if(caracter==ESC) return(-9999);
    despliegue[0] = caracter;
    despliegue[1] = '\0';
    outtextxy(x+=10,y,despliegue);
    flotante[i]=caracter;
    caracter = getch();
```

```
    i++;
  }
  flotante[i]='\0';
  numero = atof(flotante);
  return numero;
}

/* ------------------------- STRTOASC ---------------------------------- */
/* Función que convierte una cadena de dígitos (cad) en un arreglo de enteros, correspondientes
a los números ASCII de cada dígito. El número de elementos del arreglo está determinado por bytes,
y siempre el último elemento que forme parte del conjunto de números ASCII sera un 00 */

void STRTOASC(char *cad, int bytes, int * registro){

  int i,j;

  i = 0;
  j=0;
  if(cad[0]=='0') i++;
  while(j<(bytes-1)){
    switch(cad[i]){
      case '0': registro[j] = 0x30;
                break;
      case '1': registro[j] = 0x31;
                break;
      case '2': registro[j] = 0x32;
                break;
      case '3': registro[j] = 0x33;
                break;
      case '4': registro[j] = 0x34;
                break;
      case '5': registro[j] = 0x35;
                break;
      case '6': registro[j] = 0x36;
                break;
      case '7': registro[j] = 0x37;
                break;
      case '8': registro[j] = 0x38;
                break;
      case '9': registro[j] = 0x39;
                break;
      case '.': registro[j] = 0x2E;
                break;
      case '-': registro[j] = 0x2D;
                break;
      case '\0':while(j<bytes) registro[j++]=0x30;
                break;
    }
    j++;
    i++;
  }
registro[bytes-1] = 0;
}

/* ------------------------- VELRAM ---------------------------------- */
/* Función empleada para cambiar la velocidad del motor. Este cambio es temporal, es decir, si se
desea que el motor opere a la velocidad modificada, es necesario grabar esta velocidad (opción 2
dentro del
menú de velocidad) */

void VELRAM(void){

  int af,hf;
  int xs1,ys1;
  int VEL;
  char strvel[10];
  int velascii[10];
  double num;
  char tecla;

  af = getmaxx();
  hf = getmaxy();
  do{
    clrMarco1(500,1000,4500,9000);
    Virtual(1800,1300,VIRTUAL,VIRTUAL,&xs1,&ys1,af,hf);
    outtextxy(xs1,ys1,"VELOCIDAD ");
    Virtual(800,1800,VIRTUAL,VIRTUAL,&xs1,&ys1,af,hf);
    outtextxy(xs1,ys1,"Nueva Velocidad: ");
    Menu7("<1> Cambia Velocidad      <2> Graba Velocidad      <BSC> Menu Anterior","<BSC> Para
```

```
Cancelar");
    VEL = getentero(xs1+strlen("Nueva Velocidad: ")*8,ys1);
    if(VEL==-1){ clrMarco1(500,1000,4500,9000); return; }
    if(VEL<MINVEL || VEL>(MAXVEL-220)){
      Virtual(1200,2300,VIRTUAL,VIRTUAL,&xs1,&ys1,af,hf);
      outtextxy(xs1,ys1,"VELOCIDAD NO VALIDA.");
      Virtual(1200,2600,VIRTUAL,VIRTUAL,&xs1,&ys1,af,hf);
      outtextxy(xs1,ys1,"RANGO: 800 - 3500 RPM");
      settextstyle(SMALL_FONT,HORIZ_DIR,4);
      Virtual(750,2900,VIRTUAL,VIRTUAL,&xs1,&ys1,af,hf);
      outtextxy(xs1,ys1,"PRESIONE <ENTER> PARA CONTINUAR");
      settextstyle(DEFAULT_FONT,HORIZ_DIR,1);
      getch();
    }
  }while(VEL<MINVEL || VEL>(MAXVEL-220));
  Virtual(900,3000,VIRTUAL,VIRTUAL,&xs1,&ys1,af,hf);
  outtextxy(xs1,ys1,"<ENTER> PARA CONTINUAR");
  Virtual(900,3300,VIRTUAL,VIRTUAL,&xs1,&ys1,af,hf);
  outtextxy(xs1,ys1,"<ESC>   PARA CANCELAR");
  tecla=getch();
  switch(tecla){
    case '\r': break;
    case '\x1B': VELRAM();
                 return;
  }
  num = (float)VEL/(float)MAXVEL;
  if(num==1) num = 0.99999;
  gcvt(num,7,strvel);
  STRTOASC(strvel,6,&velascii[0]);
  transmitir1(0x11);
  transmitir2(velascii,6);
return;
}

/* ------------------------ NUEVOS_PARAMETROS -------------------------- */
/* Función que permite modificar los parámetros del controlador PID desde la PC. Esta función pide
al usuario la introducción de los nuevos parametros, calcula los datos A,B,C,D y F que requiere
el controlador y transmite dichos datos vía el puerto COM1 */

void nuevos_parametros(FILE * fpp){

  int af,hf;
  int xs1,ys1;
  char strvel[10];
  char tecla;
  char cadena[15];
  double num;
  float Kp,Td,Ti,T,Ta;
  double a,b,c,d,f,j;
  char stra[15];
  char strb[15];
  char strc[15];
  char strd[15];
  char strf[15];
  int A[10];
  int B[10];
  int C[10];
  int D[10];
  int F[10];

  af = getmaxx();
  hf = getmaxy();
  clrMarco1(500,1000,4500,9000);
  Virtual(1300,1300,VIRTUAL,VIRTUAL,&xs1,&ys1,af,hf);
  outtextxy(xs1,ys1,"CAMBIO DE PARAMETROS");
  Virtual(1300,1600,VIRTUAL,VIRTUAL,&xs1,&ys1,af,hf);
  outtextxy(xs1,ys1,"T = 0.1 [seg]");
  T = 0.1;
  Virtual(800,1900,VIRTUAL,VIRTUAL,&xs1,&ys1,af,hf);
  outtextxy(xs1,ys1,"Kp: ");
  Menu7("F1 Velocidad  F2 Parámetros  F3 Gráfica  F4 Archivo  F10 SALIR","<ESC> Para Cancelar");
  Kp = getflot(xs1+32,ys1);
  if(Kp==-9999){clrMarco1(500,1000,4500,9000); return; }
  Virtual(800,2300,VIRTUAL,VIRTUAL,&xs1,&ys1,af,hf);
  outtextxy(xs1,ys1,"Td: ");
  Td = getflot(xs1+32,ys1);
  if(Td==-9999){ clrMarco1(500,1000,4500,9000); return; }
  Virtual(800,2700,VIRTUAL,VIRTUAL,&xs1,&ys1,af,hf);
  outtextxy(xs1,ys1,"Ti: ");
```

```
  Ti = getflot(xs1+32,ys1);
  if(Ti==-9999){ clrMarco1(500,1000,4500,9000); return; }
  Virtual(900,5000,VIRTUAL,VIRTUAL,&xs1,&ys1,af,hf);
  outtextxy(xs1,ys1,"<ENTER> PARA CONTINUAR");
  Virtual(900,5300,VIRTUAL,VIRTUAL,&xs1,&ys1,af,hf);
  outtextxy(xs1,ys1,"<ESC>   PARA CANCELAR");
  tecla=getch();
  switch(tecla){
    case '\r': break;   /* ENTER */
    case '\x1B': nuevos_parametros(fpp);  /* ESC */
                 return;
  }
  rewind(fpp);
  gcvt(Kp,5,cadena);
  strcat(cadena,"\n");
  fputs(cadena,fpp);
  gcvt(Td,5,cadena);
  strcat(cadena,"\n");
  fputs(cadena,fpp);
  gcvt(Ti,5,cadena);
  strcat(cadena,"\n");
  fputs(cadena,fpp);
  Ta=(float)Td/(float)10;
  j =  1/(Ti*(Ta+T));
  a = (Kp*(Ta*(Ti-T)+Td*Ti))*j;
  b = (Kp*(Ti*(-2*Ta-T)+T*(Ta+T)-2*Td*Ti))*j;
  c = (Kp*Ti*(Ta+T+Td))*j;
  d = -(Ti*Ta)*j;
  f = (Ti*(2*Ta+T))*j;
  stra[0]=strb[0]=strc[0]=strd[0]=strf[0] = '\0';
  gcvt((double)a,9,stra);
  gcvt((double)b,9,strb);
  gcvt((double)c,9,strc);
  gcvt((double)d,9,strd);
  gcvt((double)f,9,strf);
  STRTOASC(stra,8,&A[0]);
  STRTOASC(strb,8,&B[0]);
  STRTOASC(strc,8,&C[0]);
  STRTOASC(strd,8,&D[0]);
  STRTOASC(strf,8,&F[0]);
  transmitir1(0x77);
  transmitir2(A,8);
  transmitir2(B,8);
  transmitir2(C,8);
  transmitir2(D,8);
  transmitir2(F,8);
return;
}

/* ------------------------- TRANSMITIR1 ------------------------------ */
/* Función que permite transmitir el byte contenido en trans por el puerto COM1 */

void transmitir1(int trans){

  register int status;

  while(1){
    status = bioscom(3,0,COM1);
    if(status&TSRE){
      bioscom(1,trans,COM1);
      break;
    }
  }
return;
}

/* ------------------------- TRANSMITIR2 ------------------------------ */
/* Función que envia el número de bytes contenidos en cuantos por el COM1 */

void transmitir2(int * BYTES,int cuantos){

   register int status;
   int cont;
   char strin[10];

   cont=0;
   while(cont<cuantos){
     status=bioscom(3,0,COM1);
     if(status&TSRE)
```

```
        bioscom(1,BYTES[cont++],COM1);
    }
return;
}

/*  ------------------------ MOSTRAR_ARCHIVO --------------------------- */
/*  Función que muestra el archivo que contiene las muestras tomadas en los
    primeros 10 [s] despues de encender o reinicializar el sistema */

void Mostrar_Archivo(FILE * stream){

  char msg[6];
  char strt[12];
  char strk[28]; /* cadena a desplegar */
  float t = 0; /* tiempo de la muestra */
  int k=1;  /* número de la muestra */
  int xiv,yiv; /* posiciones de x y y en la pantalla virtual */
  int xip,yip; /* posiciones en x y en y para colocar cada punto en pantalla */
  int af,hf; /* ancho y alto físicos del monitor */
  long posicion;

  af = getmaxx();
  hf = getmaxy();
  clrMarco1(500,1000,4500,9000);
  Virtual(525,1200,VIRTUAL,VIRTUAL,&xip,&yip,af,hf);
  outtextxy(xip,yip,"MUESTRAS EN LOS PRIMEROS 10[seg]");
  Virtual(1350,1500,VIRTUAL,VIRTUAL,&xip,&yip,af,hf);
  outtextxy(xip,yip,"k    t [seg]    w[RPM]");
  xiv = 1300;
  yiv = 1800;
  Virtual(xiv,yiv,VIRTUAL,VIRTUAL,&xip,&yip,af,hf);
 msg[0]='\0';
 if(reseteo==0)
   rewind(stream);
 else{
   rewind(stream);
   posicion=(reseteo*100);
   fseek(stream,posicion*4,SEEK_SET);
 }
 do{
    fgets(msg,5,stream);
    itoa(k,strk,10);
    if(k==101) strcpy(strk,"  ");
    strcat(strk,"     ");
    gcvt(t,4,strt);
    strcat(strt,"       ");
    strcat(strk,strt);
    strcat(strk,msg);
    outtextxy(xip,yip,strk);
    yiv+=200;
    t+=0.1;
    k++;
    Virtual(xiv,yiv,VIRTUAL,VIRTUAL,&xip,&yip,af,hf);
    if(yiv > 8400) {
      settextstyle(SMALL_FONT,HORIZ_DIR,4);
      Virtual(700,8700,VIRTUAL,VIRTUAL,&xip,&yip,af,hf);
      outtextxy(xip,yip,"PRESIONE <ENTER> PARA CONTINUAR");
      getch();
      settextstyle(DEFAULT_FONT,HORIZ_DIR,1);
      clrMarco1(500,1700,4500,9000);
      yiv=1800;
      xiv=1300;
      Virtual(xiv,yiv,VIRTUAL,VIRTUAL,&xip,&yip,af,hf);
    }
 }while(!feof(stream));
 settextstyle(SMALL_FONT,HORIZ_DIR,4);
 Virtual(700,8700,VIRTUAL,VIRTUAL,&xip,&yip,af,hf);
 outtextxy(xip,yip,"PRESIONE <ENTER> PARA CONTINUAR");
 getch();
 settextstyle(DEFAULT_FONT,HORIZ_DIR,1);
 clrMarco1(500,1000,4500,9000);
 return;
}

/* ------------------------ SALIR ------------------------------------- */
/* Función que permite salir del sistema al oprimir <F10>  */

int Salir(FILE * fp, FILE * fpp){
```

```
char tecla;

   if(kbhit()){
     tecla=getch();
     tecla=getch();
     fflush(stdin);
     if(tecla=='D'){
          fclose(fp);
          fclose(fpp);
          restorecrtmode();
          return 1;
     }
   }
return 0;
}
```

GLOSARIO

Archivo: Conjunto de registros relacionados, tratado como una unidad. Un archivo puede contener datos, programas, etc.

ASCII American Standard Code for Information Interchange: código estándar americano para intercambio de información.

Bit: es la mínima unidad de almacenamiento de información, puede ser un 1 o un 0.

Bits por segundo: El número de bits de datos enviados por segundo entre dos sistemas.

Byte: Es un conjunto de 8 bits.

Buffer: Area de memoria de almacenamiento temporal.

Bus: Un grupo de cables utilizados para transmitir un conjunto de señales de información relacionadas, entre dispositivos de un sistema.

Cargar: Enviar datos a un sistema central.

Compatibilidad: Capacidad de una integración ordenada y eficiente para coordinar los elementos de un sistema.

Chip: Conjunto complejo de componentes electrónicos y sus interconexiones, implementado sobre una pequeña lámina de material (suele ser de silicio).

Chopper: Troceador. Circuito empleado para controlar el voltaje en las terminales de una carga (dispositivo que consume energía eléctrica).

Deshabilitar: Negar permiso.

Direccionar: Acción de asignar o acceder una dirección de datos.

Diagrama de flujo: Representación gráfica de una secuencia de operaciones, usando un grupo convencional de símbolos. Puede ser general o detallado.

Dirección: Indicación en forma numérica de una información en la memoria o en otro dispositivo de almacenamiento.

Hardware: Componentes físicos de un sistema electrónico.

Implementar: Realizar.

Inicializar: Iniciar la operación de los circuitos.

Interfaz: Intercambio de información entre dos dispositivos, o el mecanismo que hace posible dicho intercambio.

LED (Light Emmiting Diode): Diodo luminoso. Elemento que se ilumina cuando se el aplica un voltaje.

Microcontrolador: Este dispositivo integra un microprocesador, unidad(es) de memoria y la interfaz entrada/salida en un solo chip.

Microprocesador: Elemento lógico complejo que realiza operaciones aritméticas, lógicas y de control, las cuales son incorporadas en un solo circuito integrado.

Monitoreo: Función que realiza cualquier dispositivo al examinar el estado de un sistema para indicar cualquier desviación que se produzca, con respecto a las condiciones de funcionamiento deseadas.

Multiplexar: La acción que realiza un multiplexor.

Multiplexor: Elemento controlado por un selector de dirección, que dirige una de sus muchas entradas a su salida.

Multiplexado: Adjetivo empleado para calificar modos de transmisión en los que se emplea un multiplexor.

Nibble: es un conjunto de 4 bits.

Offset: Compensación o desplazamiento.

Optoacoplador: Dispositivo de acoplamiento en el que el medio de acoplo es un haz de luz.

Optointerruptor: Interruptor accionado por un haz de luz.

Pila: Estructura de datos en la que la inserción o supresión de alguno de sus elementos, solo puede realizarse en alguno de sus extremos, de forma que el primer dato en entrar es el último en salir.

Preescalador: Circuito divisor de su frecuencia de entrada.

Prompt: Indicador de que el sistema operativo está listo para recibir ordenes.

Protocolo: Conjunto de reglas que regulan algún tipo de comunicación entre sistemas.

Puerto: Canal o interfaz entre un microprocesador o microcontrolador y dispositivos periféricos.

Reinicialización o RESET: Proceso de inicializar el sistema.

Ruteo: Proceso consistente en encontrar trayectorias entre puntos definidos, con el fin de ser conectados.

Set: Conjunto.

Simulación: Representación tanto abstracta como física de las características de un sistema mediante operaciones informáticas.

Sintonizar: Acción de ajustar.

Sistema: Es la combinación de elementos o componentes que actúan de manera conjunta para realizar una función perfectamente definida.

Software: Término que hace referencia a los programas y técnicas informáticas propiamente dichas.

Stack: Pila.

Subrutina: Un segmento de programa que se caracteriza por un principio y un fin y que puede ser incorporado dentro de una labor más grande.

Tarjeta: Placa de circuito impreso.

Voltaje de offset: Voltaje de CD que es sumado a una señal de interés, en ocasiones es deseable y en otras no.

BIBLIOGRAFIA

LIBROS

BOYLESTAD, NASHELSKY, Electrónica. Teoría de Circuitos, 4a. ed., Ed. Prentice Hall Hispanoamericana, S.A., México, 1989.

COUGHLIN, DRISCOLL, Circuitos Integrados Lineales y Amplificadores Operacionales, 2a. ed., Ed. Prentice Hall Hispanoamericana, S.A., México, 1987.

DEL TORO, Vincent, Fundamentos de Ingeniería Eléctrica, 2a. ed., Ed. Prentice Hall Hispanoamericana, S.A., México, 1988.

FRANKLIN, POWEL, EMAMI-NAEINI, Control de Sistemas Dinámicos con Retroalimentación, Ed. Addison-Wesley Iberoamericana, S.A., E.U.A., 1991.

OGATA, Katsuhiko, Ingeniería de Control Moderna, Ed. Prentice Hall Hispanoamericana, S.A., México, 1986.

PEREZ AMADOR, Víctor, Generadores, Motores y Transformadores Eléctricos, Facultad de Ingeniería, UNAM, México, 1994.

RODRIGUEZ RAMIREZ, Francisco J., Apuntes de Sistemas Dinámicos, Facultad de Ingeniería, U.N.A.M., México, 1986.

TOCCI, Ronald J., Sistemas Digitales. Principios y Aplicaciones, 3a. ed., Ed. Prentice Hall Hispanoamericana, S.A., México, 1987.

MANUALES

MOTOROLA, M68HC11 Reference Manual, 2a. ed., Motorola Inc., U.S.A., 1991.

MOTOROLA, MC68HC11F1 Technical Summary, Motorola Inc., U.S.A., 1991.

DEPARTAMENTO DE INGENIERIA DE CONTROL, Manual del Laboratorio de Control Digital, Facultad de Ingeniería, México, 1991.

OrCAD, Users Manual, U.S.A., 1990.

Tango PCB Plus, Users Manual., U.S.A., 1990.

OTRAS FUENTES

GARIBAY JIMENEZ, Ricardo, Apuntes de Control Digital, Facultad de Ingeniería, U.N.A.M., México, 1994.

GRAY JAMES W., PID Routines for MC68HC11K4 and MC68HC11N4 Microcontrollers, Aplication Note AN1215, Motorola Inc., U.S.A, 1992.

MOTOROLA, MC68HC11 Floating-Point Package, Aplication Note AN974, Motorola Inc., U.S.A, 1987.