CONTROL DE MOTORES DE PASOS

LUIS AGUSTIN ALVAREZ ICAZA LONGORIA

Tesis

Presentada a la División de Estudios de

Posgrado de la

FACULTAD DE INGENIERIA

de la

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

como requisito para obtener

el grado de

MAESTRO EN INGENIERIA

(CONTROL)

CIUDAD UNIVERSITARIA

Octubre de 1987

CONTROL DE MOTORES DE PASOS

Luis Alvarez Icaza Longoria
Instituto de Ingeniería, UNAM
Apdo. Postal 70-472
04510, Coyoacán D.F.
México

RESUMEN

Este trabajo trata acerca del problema de control de motores de pasos. Para ello se presenta en primera instancia una revisión de sus características más importantes, seguida de una sección con modelos matemáticos de su comportamiento. Se discuten en especial las principales técnicas que se emplean para abordar el control de los motores de pasos, según la naturaleza de las cargas acopladas a los mismos. Se propone una alternativa para controlarlos, que se basa en la simulación en lazo cerrado del sistema para control de los motores. Como resultado del empleo de esta técnica se derivan secuencias de movimiento de los motores que permiten seguir patrones arbitrarios de movimiento. Se presentan además condiciones para que dichas secuencias de movimiento sean robustas frente a variaciones en los valores de los parámetros del sistema motor-carga. La validez de la metodología propuesta se demuestra tanto teórica como experimentalmente.

El trabajo concluye sobre las condiciones de aplicación de los motores de pasos y las técnicas de control más adecuadas en cada caso.

Los programas empleados para implantar los modelos matemáticos usados y la descripción de los dispositivos que se diseñaron para llevar a cabo los experimentos con buen éxito se incluyen en anexos al final de reporte.

1. INTRODUCCION

Antecedentes

Durante 1981 y 1982 en el Instituto de Ingeniería se diseñó y construyó una máquina para cortar tubos. El dispositivo puede clasificarse como un manipulador robótico de cuatro grados de libertad y resuelve el problema de habilitar, mediante corte por soplete oxiacetilénico, los tubos que se deben unir por soldadura en la construcción de estructuras tubulares de gran tamaño. Los actuadores empleados en todos los casos fueron motores de pasos. Los resultados obtenidos se pueden considerar ampliamente satisfactorios. (En las Refs. 1 a 3 se encuentra ampliamente descrito el trabajo realizado para la construcción de este dispositivo).

A partir de la experiencia descrita se diseñó un segundo manipulador robótico, en este caso de tres grados de libertad, enfocado a resolver el problema de trasladar hileras de envases de vidrio desde una banda transportadora hacia un horno de tratamiento térmomecánico. Se decidió en este caso emplear también motores de pasos como actuadores. (En la Ref. 4 se pueden encontrar más detalles de este dispositivo)

Las condiciones de operación necesarias para esta aplicación presentaron dos variaciones importantes en relación a las de la primera: las

inercias involucradas y las velocidades de movimiento de los motores de pasos eran de mayor magnitud.

Al realizar las pruebas de funcionamiento del dispositivo se encontró que no era posible extrapolar las técnicas de control usadas en la primera aplicación, por lo que era necesario derivar una nueva metodología que tomase en cuenta las variaciones del caso particular. Se propuso entonces realizar estudios para conocer con mayor detalle el comportamiento de los motores de pasos. En este trabajo se describen las actividades realizadas y los resultados obtenidos en dichos estudios.

Contenido

El trabajo se ha dividido en once capítulos y cuatro anexos, su contenido es como sigue. El capítulo dos contiene una descripción general de los motores de pasos: sus tipos y características más importantes. En el tres se presentan los modelos matemáticos más empleados para describir su comportamiento.

En el capítulo cuatro se discuten las técnicas más usadas para el control de motores de pasos y las condiciones en que se las emplean. El capítulo cinco se derivan las condiciones bajo las cuales se puede garantizar que una secuencia de pulsos enviada a un motor de pasos sea robusta y el sexto presenta el esquema de control propuesto en este trabajo.

El capítulo número siete describe los modelos que se emplearon para las simulaciones y el octavo el banco de pruebas que se construyó para realizar los experimentos. En el capítulo nueve se muestran los resultados obtenidos de las simulaciones y experimentos realizados.

El décimo capítulo presenta las conclusiones del trabajo y por último se proporcionan las referencias bibliográficas pertinentes.

El escrito incluye cuatro anexos que contienen respectivamente: los programas para computadora digital con los que se realizaron las simulaciones, los programas para manejar el banco de pruebas, los diagramas electrónicos de los acoplamientos construidos o empleados y los datos de los experimentos realizados.

Agradecimientos

El presente trabajo no se hubiera podido realizar sin la colaboración de Roberto Canales Ruiz, que participó en la dirección de todas las etapas del mismo. Se agradece a Juan Martínez García su amable revisión de los acoplamientos diseñados y su participación en la construcción de uno de ellos. Finalmente se desea reconocer el apoyo que brindó el Instituto de Ingeniería.

2. DESCRIPCION DE LOS MOTORES DE PASOS

Funcionamiento general

Los motores de pasos se pueden clasificar como dispositivos de conversión de energía electromecánica. Las dos características más importantes que tienen son que proporcionan movimientos discretos y que se pueden mover en ambas direcciones.

La naturaleza de la forma como se producen los movimientos discretos ha propiciado su uso controlado por dispositivos digitales. De ahí el que se utilicen profusamente en la construcción de periféricos para equipos de cómputo, o en la de dispositivos que deben funcionar en un entorno supervisado por computadoras (Ref. 5). Las unidades para disquetes, las impresoras y algunas máquinas de control numéricos son buenos ejemplos de aplicaciones de los motores de pasos.

Desde el punto de vista del método de control, estos dispositivos se utilizan en general en malla abierta, lo cual implica una instalación más sencilla y económica. Sin embargo, su correcto funcionamiento en estas circunstancias debe ser garantizado con la adopción de ciertos criterios de diseño y selección.

Tipos de motores de pasos

Existen diferentes tipos de motores de pasos, a continuación se describen los de uso más frecuente. La característica común a todos ellos es que cuentan con embobinados alambrados de manera independiente. Se requieren al menos dos fases para garantizar direccionalidad.

Motor de solenoide y trinquete.- en la Fig. 2.1 se muestra un bosquejo de este tipo de motor. Se componen de una rueda dentada, un trinquete y un par de solenoides. Cuando se energiza alguno de los solenoides, el desplazamiento de su núcleo impulsa la rueda dentada en la dirección correspondiente. El trinquete detiene el movimiento de la rueda cuando ha transcurrido un desplazamiento angular igual al que separa los dientes de esta. Este tipo de motor no puede alcanzar altas velocidades, es ruidoso y en general de gran tamaño. Su uso se encuentra en franca desaparición.

Motor de rotor flexible.- se construye con un estator dentado por el interior y un rotor ovalado dentado por el exterior (véase la Fig. 2.2) El rotor incluye polos magnéticos y el estator algunos electroimanes. El movimiento se produce cuando se fuerzan alinamientos sucesivos entre el núcleo del rotor y los campos magnéticos del estator. Estos motores poseen una gran resolución (de 2000 a 5000 incrementos por revolución),pero no pueden entregar un gran par en su flecha de salida.

Motor de imán permanente.- su diagrama se muestra en la Fig. 2.3, como puede notarse en ella, se construye con un rotor que incluye polos magnéticos de polaridad contraria colocados uno junto al otro. El estator contiene bobinas alineadas de tal forma que su energización secuencial provoca que el rotor se desplace a las posiciones de mínima reluctancia magnética. Se trata de un tipo poco usado, principalmente por el escaso par que pueden proporcionar en sus flechas de salida.

Motor de reluctancia variable.- este tipo de motor era hasta hace 10 años el de uso más difundido. Su estructura básica se describe con ayuda de la Fig. 2.4. Como puede notarse se construye con un rotor que consta de varias secciones dentadas, las que se encuentran distribuidas de manera coplanar con secciones dentadas del estator. El paso angular entre los dientes de las secciones del rotor y del estator es el mismo, sin embargo, estas últimas se encuentran defasadas en la misma proporción entre sí. El movimiento se produce energizando una a una todas las secciones del estator. El rotor gira en cada caso hasta alcanzar la posición en que el circuito magnético tiene mínima reluctancia, razón a la que deben su nombre. El número de secciones del rotor y del estator debe ser mayor que tres con el fin de garantizar

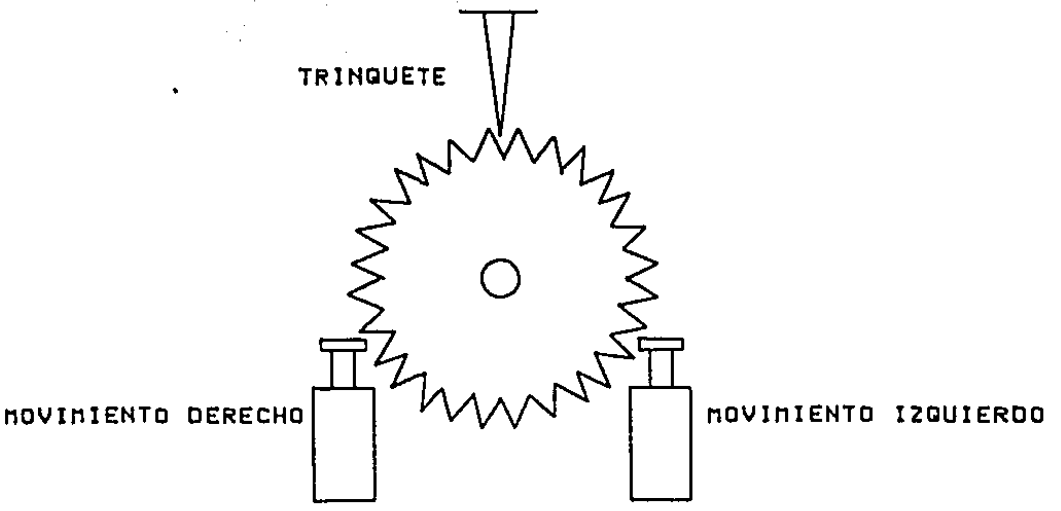


FIG. 2.1 MOTOR DE SOLENOIDE Y TRINQUETE

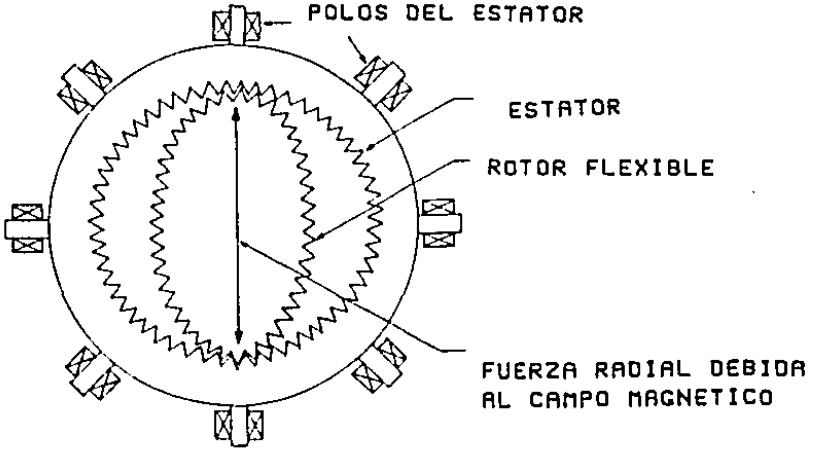


FIG. 2.2 MOTOR DE ROTOR FLEXIBLE

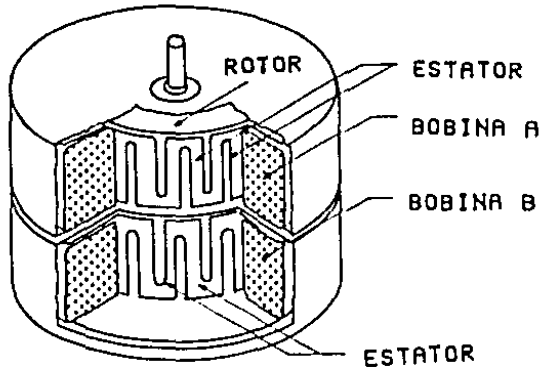
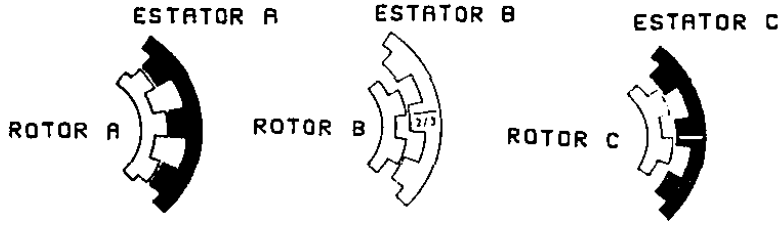


FIG. 2.3 MOTOR DE IMAN PERMANENTE



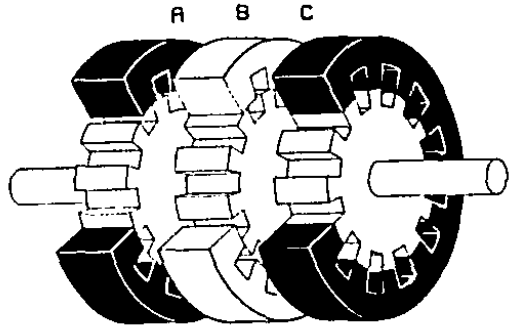


FIG. 2.4 MOTOR DE RELUCTANCIA VARIABLE

direccionalidad y determina el valor del defasamiento angular entre las mismas. Existen algunas variaciones en las construcción de estos motores como son la de contar con una sola sección tanto en el rotor como en el estator, pero magnetizar los dientes de este como si perteneciesen a secciones distintas.

Motor híbrido.- este es el tipo más común de motores de pasos y el de uso más difundido. Se construye con un rotor que tiene dos secciones dentadas, defasada la una respecto a la otra, y un estator también con dos secciones dentadas, pero en este caso con los dientes alineados (ver Fig. 2.5). Los dientes del estator se energizan con dos o más bobinas independientes, cuyo número determina el número de fases que tiene el motor. En la Fig. 2.6 se ilustra el funcionamiento de un motor muy sencillo de este tipo. Las altas velocidades de operación que pueden alcanzar (hasta 10,000 pasos por segundo) y el alto par de retención que proporcionan cuando se encuentran en reposo ha propiciado que sea el tipo más usado, razón por la cual los desarrollos y modelos presentados en este trabajo se realizaron alrededor del mismo.

Formas de manejo de los motores de pasos

La manera en que tradicionalmente se manejan los motores de pasos consiste en enviar secuencias de pulsos hacia ellos y obtener a cambio de cada pulso un movimiento angular equivalente a un paso. Dado que se ha mencionado que los motores son bidireccionales, los motores se dotan de dos puertos que reciben secuencias de pulsos. A cada puerto corresponde recibir órdenes para movimientos en una sola dirección.

De la descripción del inciso anterior se desprende que los movimientos de los motores de pasos dependen de secuencias apropiadas de energización de los embobinados de los motores, por lo que hace falta un acoplamiento que convierta las secuencias de pulsos en secuencias de energización. Para el efecto se han desarrollado lo que se llaman controladores de motores de pasos. El nombre sin embargo resulta semánticamente incorrecto en el contexto de este trabajo, por lo aquí se les ha denominado acoplamientos de potencia.

En todos los casos estos acoplamientos incluyen un autómata finito que convierte las secuencias de pulsos a los estados apropiados de energía en los distintos embobinados. El flujo de corriente en éstos se maneja a través de transistores de potencia, también en todos los casos (véase Fig. 2.7).

Existen diferentes tipos de acoplamientos de potencia, la descripción que se presenta a continuación se refiere a los tres más usados. Su uso

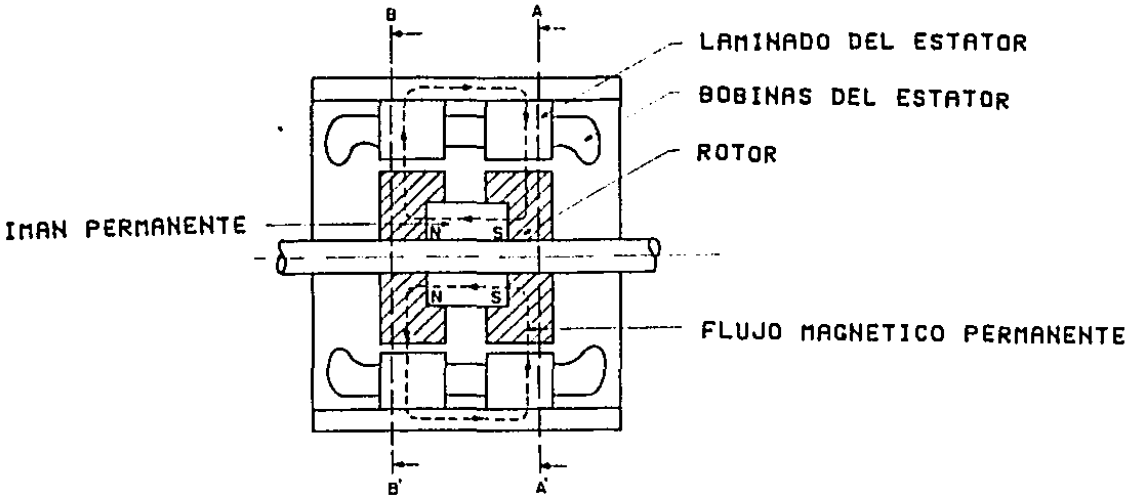


FIG. 2.5 MOTOR HIBRIDO

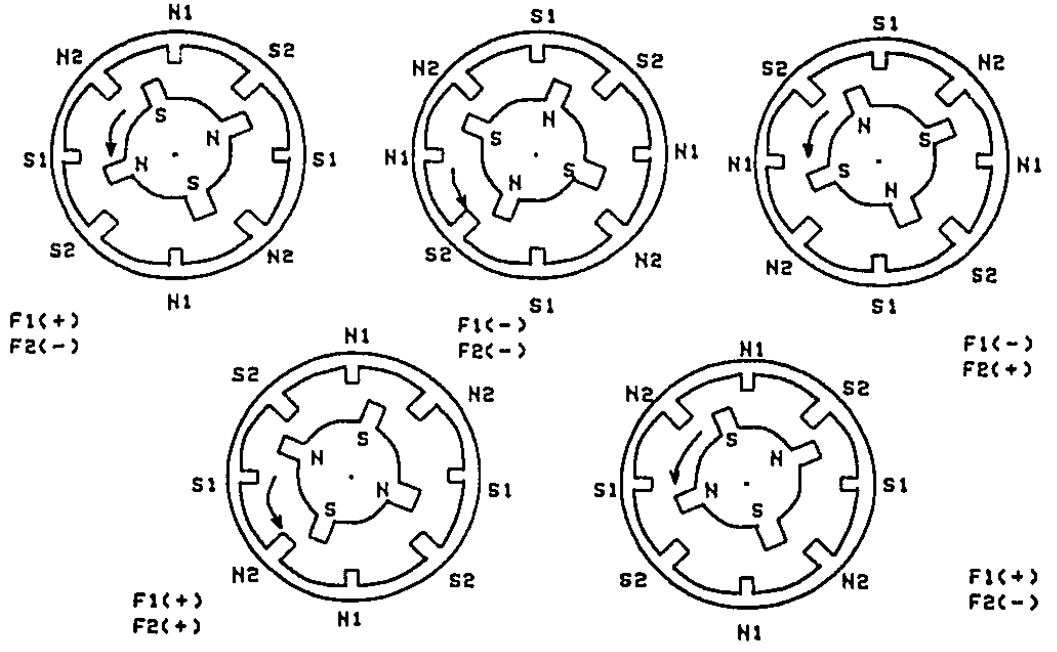


FIG. 2.6 FUNCIONAMIENTO DE UN MOTOR DE PASOS HIBRIDO

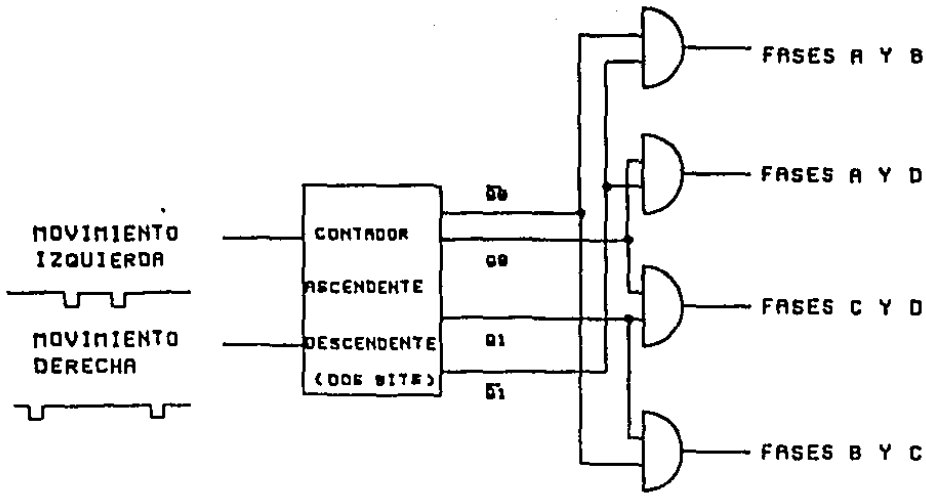
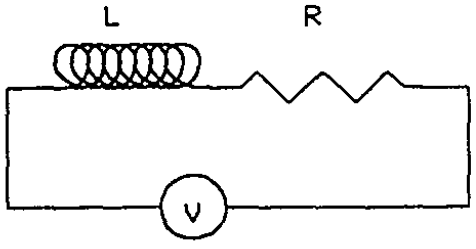
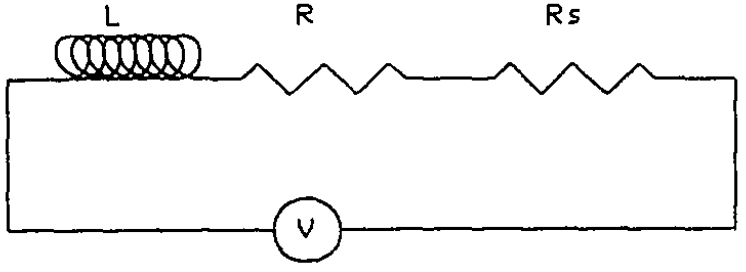


FIG. 2.7 AUTOMATA FINITO PARA MANEJAR LA SECUENCIA DE ENERGIZACION



L - INDUCTANCIA EMBOBINADO
R - RESISTENCIA EMBOBINADO
V - FUENTE DE VOLTAJE

FIG. 2.8 DIAGRAMA ELECTRICO PARA UNA FASE DEL MOTOR



L - INDUCTANCIA EMBOBINADO
R - RESISTENCIA EMBOBINADO
Rs - RESISTENCIA EN SERIE
V - FUENTE DE VOLTAJE

FIG. 2.9 DIAGRAMA CON RESISTENCIA EN SERIE

se describe, como ya se mencionó, particularizado a los motores híbridos.

Acoplamiento de potencia por resistencia inductancia

El estudio de la respuesta de los motores que ocupan este tipo de acoplamiento se basa en el circuito simplificado que se ilustra en la Fig. 2.8. Como se sabe el tiempo de levantamiento para este circuito depende de la constante de tiempo del mismo, en este caso el cociente L/R. Dada una combinación de inductancia y resistencia de los embobinados del motor existe una velocidad máxima alcanzable, que corresponde aproximadamente al inverso del tiempo de levantamiento. La relación mencionada permite alcanzar en la mayoría de los casos velocidades del orden de los 200 pasos por segundo, que son insatisfactorias para la mayoría de las aplicaciones.

Para resolver este problema se incluyen resistencias en serie con el motor (Fig. 2.9), las cuales permiten reducir la constante de tiempo de manera considerable. La desventaja consiste, evidentemente, en que para mantener la corriente nominal en los motores se debe aumentar el voltaje de alimentación. En la práctica es común que la relación entre la resistencia que se coloca en serie y la de los embobinados del motor sea de aproximadamente 10, ello implica que de la potencia consumida por el motor el 90% se disipe inútilmente en la resistencia en serie. Además cuando se tratan de alcanzar velocidades superiores a los 1000 pasos por segundo la relación anterior debe elevarse haciendo incosteable usar este tipo de acoplamientos de potencia.

Acoplamientos de fuente de voltaje dual

Este tipo de acoplamiento, como su nombre lo indica, funciona con base en dos fuentes de voltaje. La primera proporciona un voltaje cuyo valor corresponde al necesario para mantener la corriente de estado estable en los motores. La segunda alimenta al circuito con un voltaje mucho mayor al realmente necesario. El funcionamiento grosso modo es como sigue: al energizar un embobinado se le alimenta inicialmente con la fuente de alto voltaje, en forma paralela funciona un detector de corriente que verifica el instante en que esta alcanza el valor nominal. Cuando ocurre esto último, se desconecta la fuente de alto voltaje y se conecta la de bajo valor. Con esta estrategia se puede reducir enormemente el tiempo de levantamiento real y es posible alcanzar velocidades del orden de los 5,000 pasos por segundo. La relación entre los voltajes de las fuentes es más o menos de treinta. En la Fig. 2.10 se muestra el diagrama que correspondería a una fase del motor.

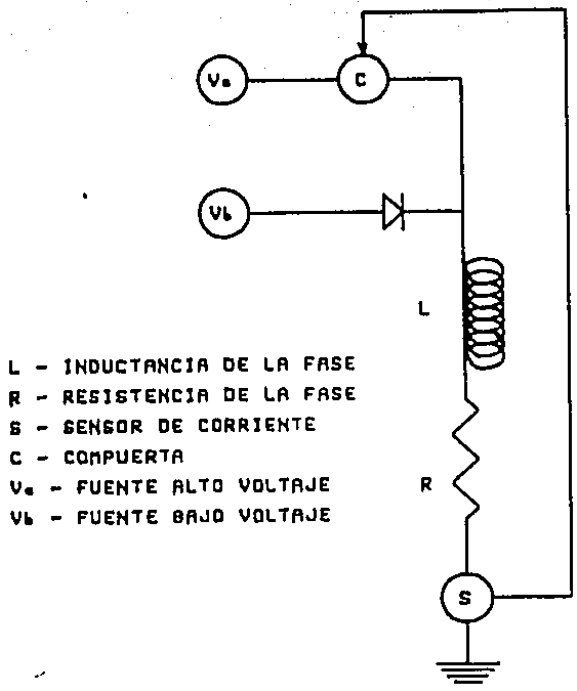


FIG. 2.10 DIAGRAMA DE UNA FASE CON ACOPLAMIENTO DE VOLTAJE DUAL

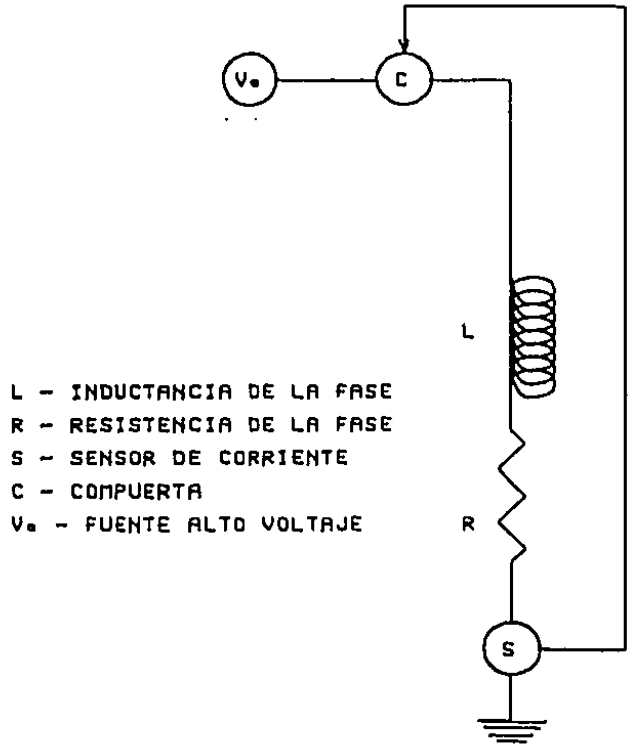


FIG. 2.11 DIAGRAMA DE UNA FASE CON ACOPLAMIENTO POR CORTE DE CORRIENTE

Acoplamiento por corte de corriente

En este tipo de acoplamientos se utiliza únicamente una fuente de alimentación de alto valor. Existen detectores de corriente que alternativamente conectan y desconectan la fuente de alimentación según el valor de la corriente esté por abajo o arriba de la corriente nominal. El voltaje de la fuente excede en un factor de hasta 50 al voltaje nominal necesario en el motor, lo que permite que con este tipo de acoplamientos de potencia se alcancen velocidades del orden de los 10,000 pasos por segundo. La Fig. 2.11 muestra un diagrama para una fase del motor.

3. MODELOS MATEMATICOS DE LOS MOTORES DE PASOS

En este capítulo se describirán los modelos matemáticos empleados para describir el comportamiento de los motores de pasos. Se presentarán modelos para describir los elementos mecánicos, eléctricos y magnéticos de los motores. Los modelos magnéticos se basan en un par de suposiciones, cuya validez se discute en un inciso por separado. Por último, se incuye un modelo para explicar el problema de la resonancia de los motores a bajas velocidades.

Los modelos se presentan de manera constructiva: se parte de los elementos básicos y se incorporan después los de mayor complejidad. Se procede de esta forma pues el trabajo pone especial énfasis en el empleo de modelos simplificados para el diseño de las técnicas de control.

Todas las descripciones se basarán en el motor híbrido de dos fases bipartidas, pues como ya se explicó, es el tipo más difundido y para el cual se realizaron los experimentos de validación de los algoritmos propuestos en este trabajo.

FUNCIONAMIENTO BASICO DEL MOTOR HIBRIDO

En la Fig. 3.1 se muestra el diagrama que corresponde a un motor de pasos híbrido. En la parte mecánica se distingue el rotor, con una inercia J, un amortiguador viscoso, con constante B y un par aplicado τ. En la parte electromagnética se muestran las resistencias e inductancias correspondientes a las cuatro fases del motor. Los valores de aquellas se consideran iguales para todas las fases. Puede notarse que las fases A y C, y las fases B y D tienen puntos de salida comunes.

Existen dos maneras para conseguir que este tipo de motor se mueva de manera discreta. La primera consiste en tratar las fases A y C como una sola fase AC, y las fases B y D como otra sola fase, BD. El funcionamiento bajo este tratamiento coincide con el que se expuso en el capítulo anterior (ver Fig. 2.6). La otra forma se consigue si se energiza a la vez solamente una de las fases de cada pareja A-C y B-D. Bajo este esquema el movimiento del motor se produce cuando se sigue una secuencia como la que describe la Tabla 3.1. Esta segunda forma de manejo es la más usada en la actualidad, a pesar de que implica una pérdida en el par total que puede suministrar el motor. La razón para su uso estriba en la simplicidad de la electrónica del acoplador de potencia que maneja los trenes de pulsos que se envían al motor.

Existe una tercera posibilidad para mover los motores que consiste en dividir en dos etapas el proceso de apagar una fase y prender la otra (este proceso se realiza en una sola etapa en la Tabla 3.1). Cuando se adopta este esquema el motor se mueve únicamente la mitad de un paso, pero se pierde de nueva cuenta la mitad del par disponible para las posiciones en que está energizada una sola fase. La Tabla 3.2 describe las secuencias de energización para el movimiento a medio paso.

MODELO MECANICO

El modelo mecánico para un motor de pasos y el sistema de cargas que se le acople se obtiene al aplicar la segunda ley de Newton al movimiento del motor. La ecuación que se deriva es:

$$J\ddot{\theta} + B\dot{\theta} + F\frac{\dot{\theta}}{|\dot{\theta}|} = \tau \qquad (3.1)$$

$$\tau = -T \operatorname{sen}\left(\frac{N_p}{N_f}\theta + K(t)\frac{\pi}{2}\right)$$

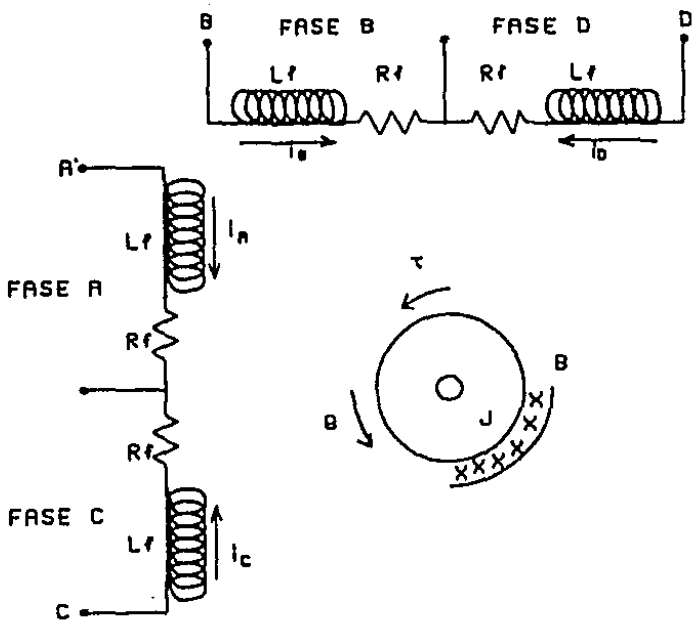


FIG. 3.1 DIAGRAMA DE UN MOTOR DE PASOS HIBRIDO

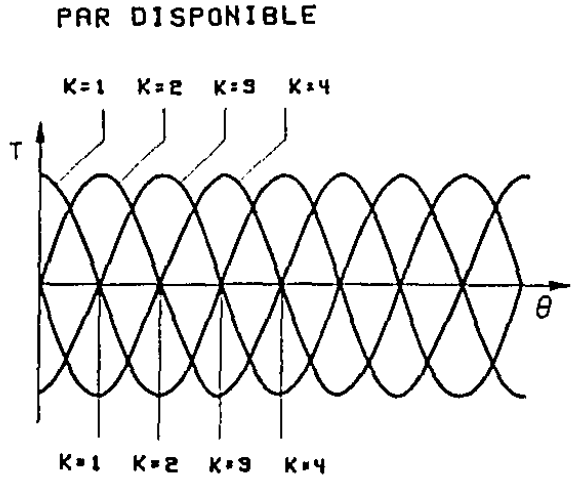
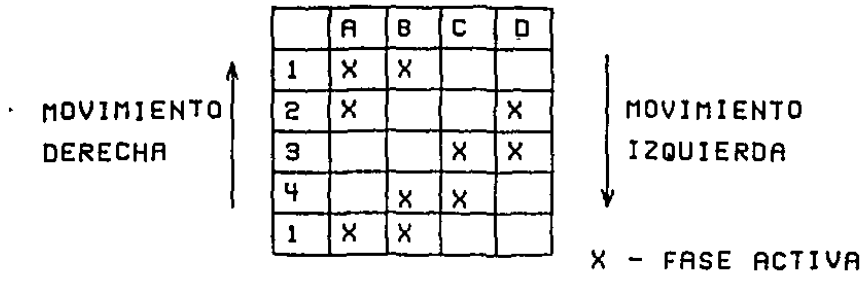
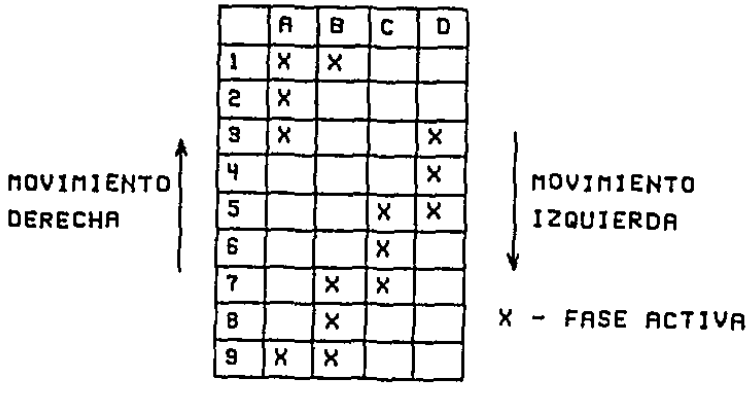


FIG. 3.2 CURVAS PAR-POSICION ANGULAR



	A	B	C	D
1	X	X		
2	X			X
3			X	X
4		X	X	
1	X	X		

TABLA 3.1 SECUENCIA PARA MOVIMIENTO DE UN MOTOR DE PASOS



	A	B	C	D
1	X	X		
2	X			
3	X			X
4				X
5			X	X
6			X	
7		X	X	
8		X		
9	X	X		

TABLA 3.2 SECUENCIA DE ACTIVACION PARA MOVIMIENTO A MEDIO PASO

donde θ =posición angular del rotor; $\dot{\theta}$ =velocidad del rotor; $\ddot{\theta}$ = aceleración del rotor; J=inercia del rotor-carga; B=coeficiente de amortiguamiento viscoso; F=coeficiente de fricción de Coulomb; τ=par aplicado por el motor de pasos; T=máximo par que puede aplicar el motor de pasos; N_p =número de pasos por revolución que proporciona el motor; N_f=número de fases del motor; K(t) es una función de conmutación que cumple con las siguientes condiciones:

$$K(t) = \{1,2,3,4\} \tag{3.2}$$

donde además el tiempo t se puede dividir en un conjunto de subintervalos disjuntos, cuyo límites están dados por los tiempos

$$t_1 < t_2 < \ldots\ldots < t_n \tag{3.3}$$

llamados tiempos de conmutación. Los valores de k(t) satisfacen:

$$K(t) = K(t_i) \quad ; \quad \forall\, i \;\; [t_i,\, t_{i+1}) \qquad i = 1,2,\ldots,n-1 \tag{3.4}$$

La ecuación (3.1) indica que el par que proporcionan los motores de pasos varía de manera senoidal con la posición angular del rotor. Esto implica que cuando el motor ha alcanzado la posición que corresponde a un paso entero, el par que aplica es nulo, mientras que cuando se encuentra alejado una distancia angular igual a la magnitud del paso, el par aplicado es máximo y su efecto es el de acercar al motor de pasos hacia la posición de par nulo. A esta última se le conoce como posición de equilibrio estable, y al efecto de restitución, par de retención.

La Fig. 3.2 muestra las curvas par posición que corresponden a las cuatro posiciones estables que se indicaron en la Tabla 3.1. Puede notarse que se trata de curvas senoidales desplazadas noventa grados una con respecto a la otra.

En el inciso que describe los modelos electromágneticos se retoma esta suposición sobre la variación del par, y se da una comprobación cualitativa para la misma.

Si la ecuación (3.1) se escribe en términos de variables de estado se llega a:

$$\frac{d}{dt} = \omega$$

$$\frac{d\omega}{dt} = \frac{B}{J}\,\omega - \frac{F}{J}\cdot\frac{\theta}{|\dot{\theta}|} - \tau \qquad (3.5)$$

MODELO ELECTRICO

El modelo mecánico del inciso anterior no toma en cuenta la forma en que la variación de las corrientes en las fases del motor de pasos puede influir sobre el par generado. Para tomar en cuenta dicho efecto se requiere proponer modelos eléctricos para cada fase del motor.

Cuando se energiza cualquier fase la corriente en la misma satisface la ecuación:

$$V_j = R_f\, I_j + L_f \frac{dI_j}{dt} \qquad j = A,\ldots,D \qquad (3.6)$$

donde j=fase que se describe; V=voltaje aplicado; I=corriente de fase; R_f=resistencia eléctrica de cada fase; L_f=inductancia de cada fase. La Fig. 3.3 describe el circuito eléctrico para una fase energizada.

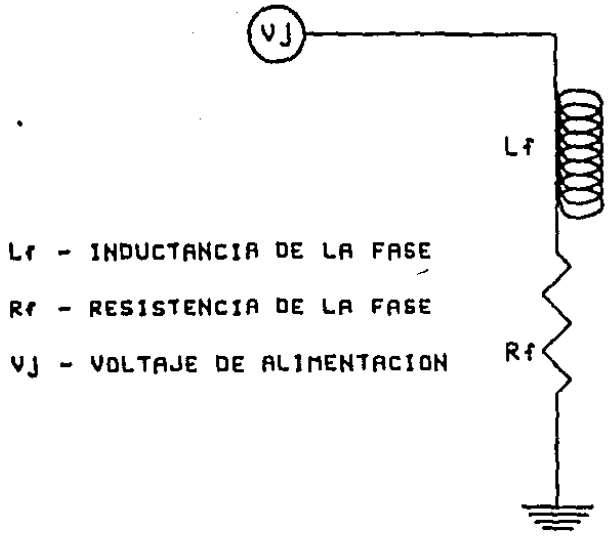
En el instante en que se desconecta una fase, la corriente en la misma satisface la ecuación:

$$0 = (R_D + R_f)\, I_j + L_f \frac{dI_j}{dt} \qquad (3.7)$$

donde R_D es la resistencia para descarga. La Fig. 3.4 describe el circuito eléctrico de descarga. Este circuito está dotado de un diodo que impide el paso de corriente durante el ciclo de energización de los embobinados, pero que permite el paso de corriente durante la desconexión del embobinado para evitar sobrevoltajes innecesarios.

La constante de tiempo del circuito de carga es L_f/R_f, y la del circuito de descarga $L_f/(R_f+R_D)$, de donde se concluye que si:

$$R_D >> R_f \qquad (3.8)$$

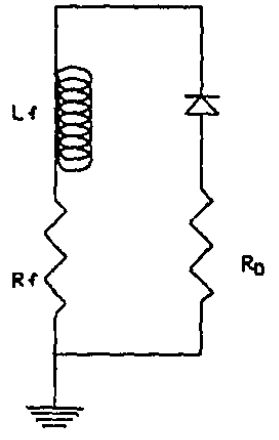


Lf - INDUCTANCIA DE LA FASE

Rf - RESISTENCIA DE LA FASE

VJ - VOLTAJE DE ALIMENTACION

FIG. 3.3 CIRCUITO PARA UNA FASE ENERGIZADA



Lf - INDUCTANCIA DE LA FASE

Rf - RESISTENCIA DE LA FASE

RD - RESISTENCIA DE DESCARGA

FIG. 3.4 CIRCUITO DE DESCARGA DE UNA FASE

el tiempo de descarga es despreciable y basta con tomar en cuenta los circuitos relacionados con la energización. La Fig. 3.5 muestra la forma de la corriente para la energización y desenergización de una fase.

Tomar todas las ecuaciones posibles en (3.6) implica formar un sistema de ecuaciones diferenciales de cuarto orden. Se pueden hacer consideraciones para reducir esta dimensión. La primera es que como sólo se energizan a la vez una de las fases A y C, y otra de las B y D, es suficiente tomar dos ecuaciones para describir eléctricamente al motor, una para la fase A y otra para la B.

El proceso de conmutación se modela a través de un cambio de condiciones iniciales, este implica resolver la ecuación (3.6) para cada energización de alguna de las fases asociada con ella considerando que la corriente inicial es nula. Otra forma de tomar en cuenta dicho proceso de conmutación es la de modificar la ecuación (3.6) para hacerla de la forma:

$$V_j - V_{nom}\,\delta(t - t_c) \cdot R_f\, I_j + L_f \frac{dI_j}{dt} \qquad j = A,\ldots,D \qquad (3.9)$$

donde V_{nom}=voltaje nominal, siempre y cuando los tiempos de conmutación cumplan con:

$$|t_{i+1} - t_i| > \frac{L_f}{R_f} \quad ; \quad i = 1,2,\ldots,n-1 \qquad (3.10)$$

El efecto de la variación de la corriente sobre el par que proporciona el motor se modela a través de

$$\tau = -\frac{T(I_A + I_B)}{2I_{nom}} \operatorname{sen}\left[N_r - K(t)\,\frac{\pi}{2}\right] \qquad (3.11)$$

donde $Nr=N_p/N_f$ e I_{nom}=corriente nominal.

MODELO ELECTROMAGNETICO

El desarrollo de este modelo se debe a que la variación de la posición del rotor sobre el campo magnético que producen las diferentes fases,

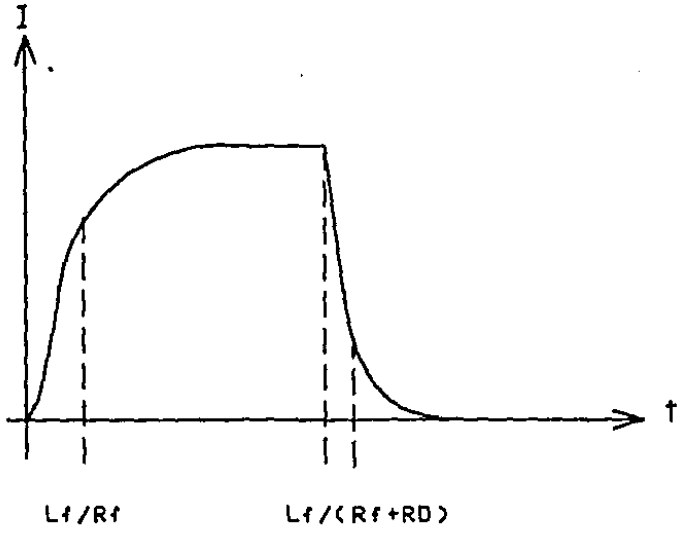


FIG. 3.5 CORRIENTE DE CARGA Y DESCARGA DE UNA FASE

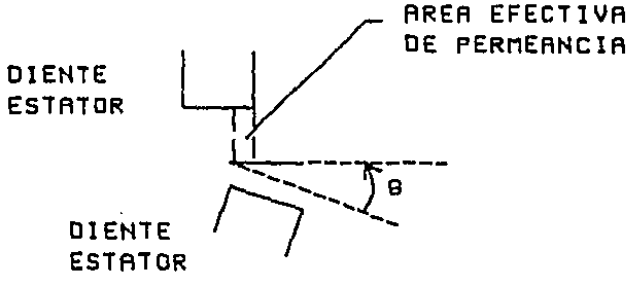


FIG. 3.6 VARIACION DE LA PERMEANCIA CON LA POSICION ANGULAR

produce a su vez una variación en la inductancia de los embobinados. Esta última se consideró constante en el inciso anterior. Si se desea tomar en cuenta dicha variación se debe modificar la ecuación (3.6) para incluir el efecto del acoplamiento magnético sobre el comportamiento de la corriente en los embobinados. El modelo que se presenta fue desarrollado por Kuo (Ref. 6).

Bajo las circunstancias anteriores el modelo que decribe cada fase es:

$$V_j = R_f\ I_j + \frac{d\lambda_j}{dt} \qquad j = A,\ldots,D \tag{3.12}$$

donde λ=acoplamiento magnético. Desde el punto de vista electromagnético se supone que las fases A y C tienen un 100% de acoplamiento magnético y que las corrientes de cada una producen efectos magnéticos contrarios. La suposición es válida también para las fases B y D. Adicionalmente se supone que las fases A y C están en cuadratura magnética con las fases B y D, por lo que no existen efectos electromágneticos entre cada par de fases.

El acoplamiento magnético de cada fase tiene dos componentes: una debida a la acción del imán permanente del rotor y la otra debido a la autoinductancia. La componente debida al imán se supone que varía según la expresión:

$$\lambda_{im} = K_\lambda \cos\ [Nr\ \theta - (i-1)\tfrac{\pi}{2}]\ ;\ i = 1,2,..,4 \tag{3.13}$$

con K_λ=constante de proporcionalidad. La autoinductancia de cada fase se considera constante e igual a L. De acuerdo con lo anterior los acoplamientos magnéticos están dados por:

$$\lambda_A = L(I_A - I_C) + K_\lambda \cos\ (N_r\ \theta)$$

$$\lambda_B = L(I_B - I_D) + K_\lambda \cos\ (N_r\ \theta - \pi/2) \tag{3.14}$$

$$\lambda_C = -\lambda_A$$

$$\lambda_D = -\lambda_B$$

Para este modelo el par que proporciona el motor es:

$$\tau = \frac{K_T}{L} [(I_C - I_A) \text{ sen } (N_r \theta) + (I_D - I_B) \cos (N_r \theta)] \qquad (3.15)$$

con K_T=constante de propocionalidad, o bien en términos de los acoplamientos, la ecuación (3.15) sería:

$$\tau = \frac{K_T}{L} [\lambda_A \text{ sen } (N_r \theta) + \lambda_B \cos (N_r \theta)] \qquad (3.16)$$

En variables de estado el modelo completo es:

$$\frac{d\lambda_A}{d_t} = \frac{1}{2} [V_A - V_C] - \frac{R}{2L} \lambda_A + \frac{RK_\lambda}{2L} \cos (N_r \theta)$$

$$\frac{d\lambda_B}{d_t} = \frac{1}{2} [V_B - V_D] - \frac{R}{2L} \lambda B + \frac{RK_\lambda}{2L} \text{ sen } (N_r \theta) \qquad (3.17)$$

$$\frac{d\theta}{dt} = \omega$$

$$\frac{d\omega}{dt} = - \frac{B}{J} \omega - \frac{K_T}{JL} \lambda_A \text{ sen } (N_r \theta) + \frac{K_T}{JL} \lambda_B \cos (N_r \theta)$$

HIPOTESIS BASICAS PARA LOS MODELOS MATEMATICOS

La ecuación (3.13) incluye el origen de la suposición básica sobre la variación del par versus la posición angular del rotor. Para justificar esta hipótesis considérese la Fig. 3.6. En ella se muestra un esquema simplificado de un diente del rotor, que se mueve para alinearse con el diente correspondiente del estator. La permeancia del flujo magnético está dominada por la proporción del área del diente del rotor que coincide con el diente del estator, por lo que la permeancia está dada por:

$$P = P_1 + P_0 \cos(N_r \theta) \qquad (3.18)$$

el término P_1 está asociado con la permeancia mínima, que corresponde con un defasamiento total de los dientes del rotor y estator, mientras que el término P_0 tiene que ver con la ganancia en la permeancia debido a la coincidencia de los dientes.

Se sabe además (Ref. 7) que en la posición de máxima permeancia se da también el máximo acoplamiento magnético, pero se dispone del mínimo par

en la flecha del motor. Esto se debe a que el par se produce por la tendencia de los circuitos magnéticos con partes móviles a encontrar las posiciones de mínima reluctancia o máxima permeancia.

El otro supuesto importante que se realiza en la Ref. 6 para derivar el modelo electromagnético, es que los embobinados de las fases están distribuidos senoidalmente sobre la periferia del estator. A partir de esta suposición se determina el número de vueltas efectivas que acopla cada fase con cada diente del estator. La justificación al respecto se encuentra si se considera que la rotación de un embobinado con respecto a un diente fijo es equivalente, desde el punto de vista del flujo magnético, al efecto de una bobina perpendicular pero con un número menor de vueltas. La Fig. 3.7 ilustra este hecho.

RESONANCIA EN LOS MOTORES DE PASOS

Este inciso describe las causas del problema de resonancia de los motores de pasos. Este fenómeno se presenta a velocidades de movimiento relativamente bajas. Las primeras explicaciones para su presencia se referían a una pérdida brusca en el par que proporciona el motor de pasos (Ref. 8). El desarrollo que se presenta a continuación está basado en el trabajo de Pollack (Ref. 9), quien postula que la resonancia se debe a una pérdida relativamente brusca de amortiguamiento.

Si se desea mover un motor a una velocidad fija ω_p, es posible establecer un sistema de coordenadas angulares que se mueva a dicha velocidad. En este sistema el problema de resonancia de un motor se puede modelar a través de un oscilador amortiguado, es decir, por la expresión:

$$\theta_m = \theta_{om} e^{-Dt} \cos \omega_o t \qquad (3.19)$$

donde θ_m =posición del rotor en sistema de coordenadas móviles; θ_{om} =valor inicial de la perturbación en la posición por alteración en la velocidad; ω_o =frecuencia de oscilación en el sistema rotatitivo; D=coeficiente de amortiguamiento.

La tesis de Pollack es que el coeficiente de amortiguamiento es una función de la velocidad de movimiento es decir:

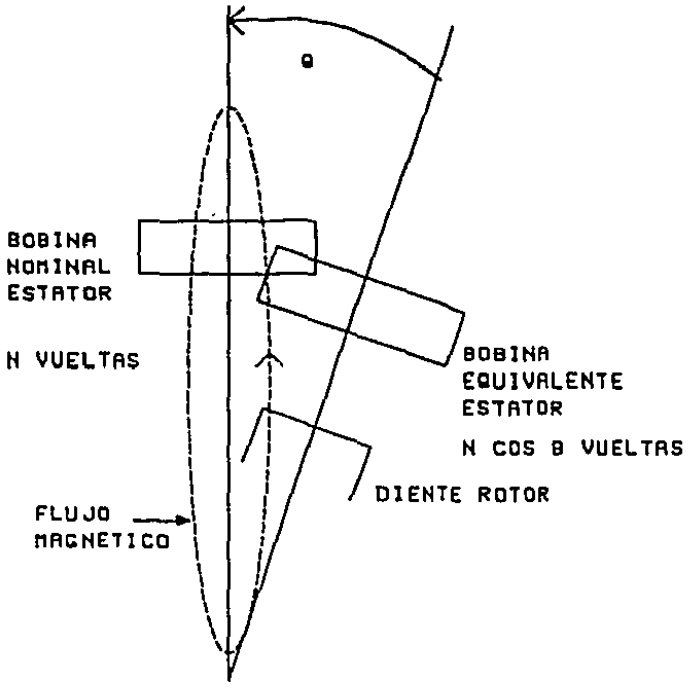


FIG. 3.7 VARIACION DEL ACOPLAMIENTO MAGNETICO CON LA POSICION ANGULAR

$$D = D(\omega_p) \qquad (3.20)$$

Para demostrar la validez de esta hipótesis se realizaron mediciones de el coeficiente de amortiguamiento. La Fig. 3.8 muestra dicho coeficiente en función de la velocidad de desplazamiento del motor, para un motor determinado. Se puede notar una rápida disminución en el valor de D para la zona de velocidades entre 950 y 1050 pasos por segundo. Esta zona corresponde con la de resonancia observada experimentalmente para el motor bajo prueba.

La demostración analítica de la hipótesis (3.20) se basa en el modelo presentado en el inciso anterior (ver ecuación 3.17). Se obtiene una solución que corresponde a velocidad invariante, y a partir de ello se muestra que para obtener error de velocidad nulo, la variación de los voltajes de alimentación de las distintas fases debe tener forma senoidal. Este resultado es de esperarse pues el motor de pasos híbrido se diseñó originalmente como un motor síncrono de baja velocidad.

Una vez mostrado lo anterior el modelo de (3.17) se linealiza respecto a la solución para velocidad constante. Para este modelo linealizado se propone que la variación de la velocidad tenga la misma estructura que la variación de la posición angular en el sistema de coordenadas móviles mencionado en la ecuación (3.19).

Pollack demuestra que el sistema linealizado tiene solución para la forma propuesta. Nótese que (3.19) es de la forma:

$$\theta_m = e^{\alpha t} \cos \gamma t \qquad (3.21)$$

los parámetros α y γ de (3.20) cumplen con:

$$\alpha = -\frac{B}{J} + \frac{RK_\lambda N_p K_T}{2 JL^2(\alpha^2 + \gamma^2)} \left[\frac{(R/2L + \alpha)\ \alpha - (N_r \omega\theta + \gamma)\ \gamma}{(N_r \omega\theta + \gamma)^2 + (R/2L + \alpha)^2}\right] + \qquad (3.22)$$

$$+ \left[\frac{(R/2L + \alpha)\ \alpha + (N_r \omega_o - \gamma)\ \gamma}{(N_r \omega_o \gamma)^2 + (R/2L + \alpha)^2}\right]$$

$$-\gamma = \frac{R K_\lambda N_r K_t}{2J L^2(\alpha^2 + \gamma^2)} \left[\frac{(N_r \omega_o + \gamma)\alpha + (R/2L + \alpha)\gamma}{(N \omega_o + \gamma)^2 + (R/2L + \alpha)^2}\right] + \tag{3.23}$$

$$+ \left[\frac{(R/2L + \alpha)\gamma - (\omega_o - \gamma)\alpha}{(\omega_o - \gamma)^2 + (R/2L + \alpha)^2}\right]$$

El primer término de (3.21) depende del amortiguamiento mecánico, mientras que el segundo de la interacción mecánica eléctrica. La conclusión del trabajo es que la resonancia de los motores de pasos se debe efectivamente a la pérdida brusca de amortiguamiento viscoso.

4. TECNICAS PARA CONTROL DE MOTORES DE PASOS

En este capítulo se describirán las técnicas más empleadas para el control de motores de pasos. Estas pueden agruparse según la naturaleza de las cargas, viscosas o inerciales, que se acoplan a los motores. Las que se describirán en este escrito son:

Técnica para cargas viscosas.

Técnicas para cargas inerciales.
- del plano de fase
- de Venkataratnam.
- del ángulo de adelanto o conmutación.
- de control realimentado.
- de Leenhouts.
- de Kuo.

Todas las técnicas, salvo las del ángulo de adelanto o conmutación y la de control realimentado se implantan en malla abierta.

TECNICA PARA CONTROL DE CARGAS VISCOSAS

Cuando los motores de pasos se ocupan para mover cargas de tipo viscoso, la ecuación que describe la relación entre la posición angular de rotor (θ) y el par τ está dada por:

$$B\,\dot{\theta} = \tau \qquad (4.1)$$

$$\tau = T \cos \theta$$

donde B y, T han sido definidos con anterioridad y θ se ha definido de forma tal que un paso equivale a $\pi/2$ radianes.

Del análisis de las ecuaciones (4.1) se desprende en primera instancia que cuando el par que se aplica es cero, la velocidad resultante también lo es, por lo que el motor estará en reposo en esta circunstancia. Esto implica que para cada paso el sistema carga-motor alcanza un punto de equilibrio estable.

El movimiento que describirá el rotor entre cada paso se puede describir a partir de la solución exacta a (4.1) que está dada por:

$$\theta(t) = 2 \operatorname{arctg} \left(\exp \left(\frac{T}{B} t\right)\right) - \pi/2 \qquad (4.2)$$

Cuando las velocidades son bajas, el movimiento que se describe tiene una forma similar a la que se ilustra en la Fig. 4.1. Conforme la velocidad aumenta, el perfil del movimiento se parece al que se presenta en la Fig. 4.2. Resulta claro que no es posible incrementar indefinidamente la velocidad del motor. Más específicamente, la máxima velocidad que se puede alcanzar está determinada por el tiempo de levantamiento(t_1) del sistema (en este contexto se define a dicho tiempo como el necesario para que el sistema alcance el 95% de su valor final) y que se puede aproximar por (Ref. 10):

$$t_1 = \frac{3.25}{T/B} \qquad (4.3)$$

Cuando se desea mover un motor acoplado a cargas viscosas y los tiempos de conmutación están separados por tiempos menores al de levantamiento, el motor no podrá seguir el movimiento que se le impone y perderá sincronía.

La aplicación de los conceptos anteriores al control de cargas viscosas se puede plantear así. Si se desea que el motor describa un patrón de movimiento prescrito de antemano y que se especifica como una función del tiempo $\theta_D(t)$, se debe seguir el siguiente proceso.

En primer término se divide al movimiento en pasos enteros. Acto seguido se encuentran los tiempos que corresponden a dichos pasos y que

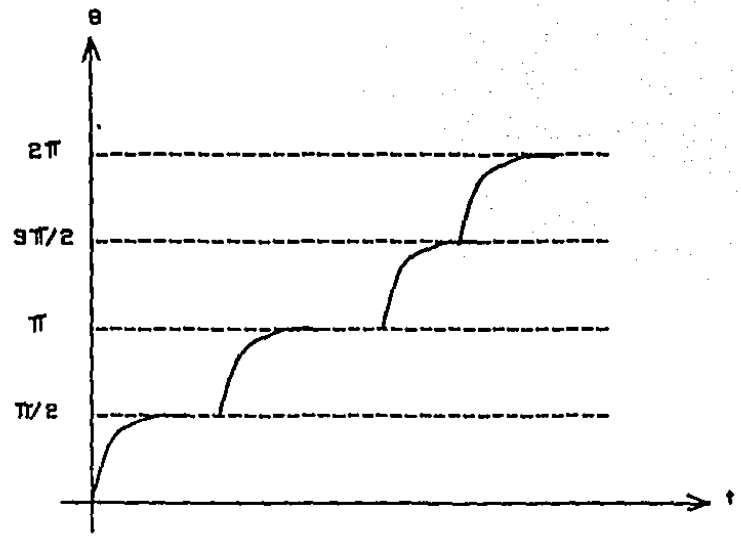


FIG. 4.1 RESPUESTA A PASOS CON CARGAS VISCOSAS Y VELOCIDAD BAJA

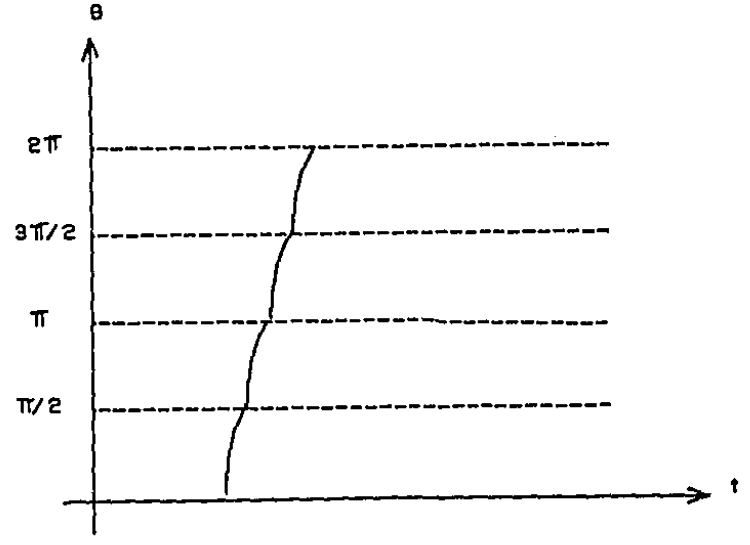


FIG. 4.2 RESPUESTA A PASOS CON CARGAS VISCOSAS Y VELOCIDAD ALTA

correspondenrán a los tiempos de conmutación de energía en los embobinados del motor. El conjunto de tiempos de conmutación $\{t_c\}$ se puede hallar de:

$$t_{ci} = \min_{t} (t: |\theta_D(i) - \theta_D(t_{Ci-1})| \geq \pi/2 \ , \ t > t_{Ci-1}) \qquad i = 1,... \tag{4.4}$$

$$t_{Co} = 0$$

La ecuación (4.4) se resuelva tantas veces como sea necesario hasta que se agote el patrón de movimiento deseado $\theta_D(t)$.

El movimiento que se ha prescrito al sistema motor-carga viscosa se podrá realizar si se cumple la siguiente condición para todos los tiempos de conmutación:

$$|t_{ci} - t_{c_{i-1}}| \geq t_1 \quad ;i=1,..,NP \tag{4.5}$$

con NP el número de pasos que se obtiene como el máximo i necesario para discretizar toda la función $\theta_D(t)$ según (4.4).

Cuando la condición (4.5) no se satisface totalmente se debe modificar el patrón de movimiento deseado $\theta_D(t)$ de tal forma que se cumpla la condición.

Las conclusiones más importantes que se pueden derivar para este tipo de movimiento son que: primero, no es posible seguir referencias con dinámicas complejas, más específicamente sólo se podrán seguir movimientos cuya derivada esté acotada por:

$$|\theta_{max}| \leq 1/t_1 \tag{4.6}$$

Segundo, aquellos movimientos cuya magnitud sea inferior a la resolución del motor empleado, no podrán realizarse.

En la Fig. 4.3 se muestra esquemáticamente el proceso que se ha descrito en estos párrafos.

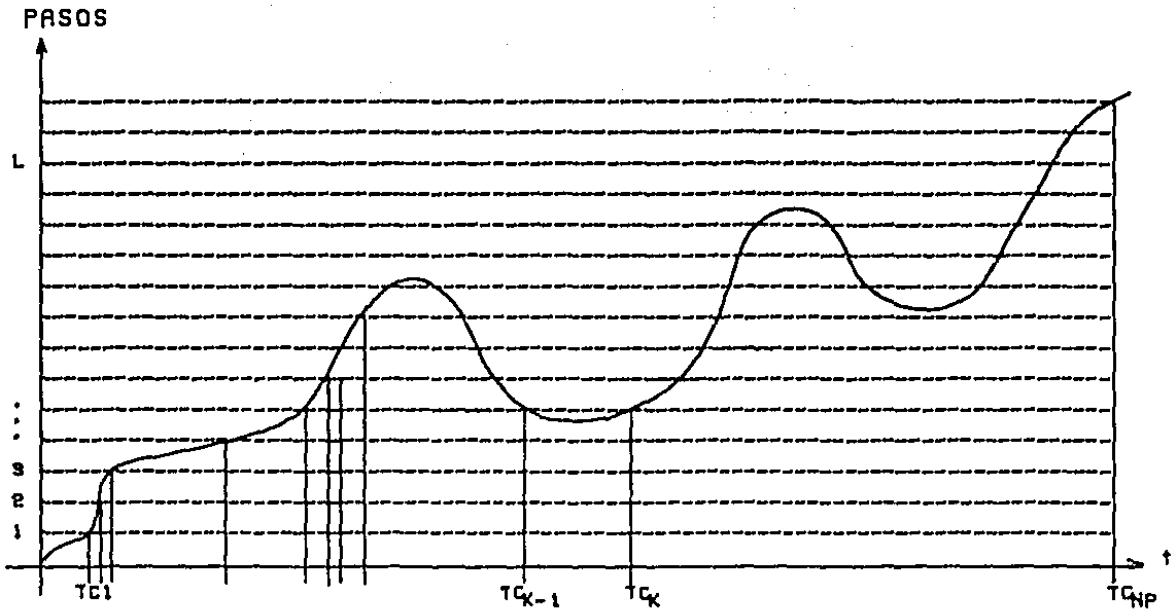


FIG. 4.3 TIEMPOS DE CONMUTACION PARA CONTROL DE CARGAS VISCOSAS

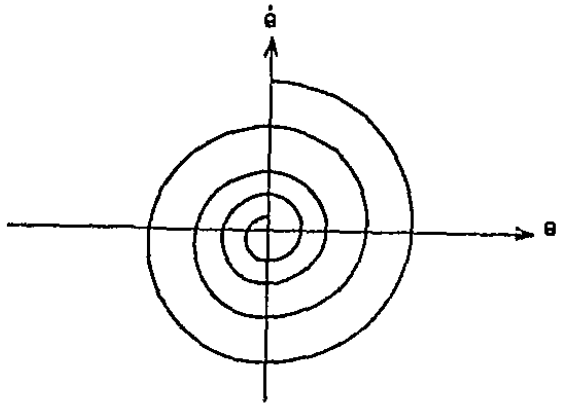


FIG. 4.4 RESPUESTA A PASO EN EL PLANO DE FASE

TECNICA DEL PLANO DE FASE

La aplicación del plano de fase al control de motores de pasos se describe con detalle en las Refs. 11 a 13. La idea básica detrás de esta técnica es la de aprovechar la fácil visualización que proporciona el plano de fase sobre la dinámica del motor de pasos. En la Fig. 4.4 se muestra la respuesta a un paso en dicho plano.

Si se tiene como modelo simplificado del comportamiento del motor de pasos el que describe la ecuación

$$J\ddot{\theta} + B\dot{\theta} + K_T\, i\, \text{sen}\,(N_r\,\theta) = F\,\frac{\dot{\theta}}{|\dot{\theta}|} - K_c\,\dot{\theta}\,\text{sen}^2(N_r\,\theta) \qquad (4.7)$$

es posible modificar esta ecuación para incluir el efecto de las conmutaciones de energía en los embobinados si se hace que:

$$\text{sen}\,(N_r\,\theta) = \text{sen}\,(N_r\,\theta - K\,\frac{\pi}{2}) \qquad (4.8)$$

donde K es un parámetro que domina las conmutaciones y que indica el número de pasos que se ha enviado en un momento dado. Si el valor de K se incrementa en la unidad al tiempo t, el motor se moverá un paso hacia adelante, y se decrementa en uno, lo hará hacia atrás. Incrementos o decrementos en K de 2 o 3 unidades implican ese número de conmutaciones instantáneas y corresponderían a secuencias de conmutación inapropiadas, en principio.

La implicación de este modelo para las conmutaciones es que cuando el valor de K se incrementa, la posición de equilibrio se desplaza $\pi/2$ radianes hacia la derecha. Este desplazamiento puede hacerse relativo a la posición actual de equilibrio del motor si se altera la posición angular en $-\pi/2$ radianes. La Fig. 4.5 muestra una secuencia de conmutación de cuatro pasos bajo esta última consideración.

La Fig. 4.6 ilustra un plano de fase que contiene tres posiciones de equilibrio. Se puede notar que existen muchas líneas por arriba de los movimientos que se realizan hacia los tres focos. Esta líneas

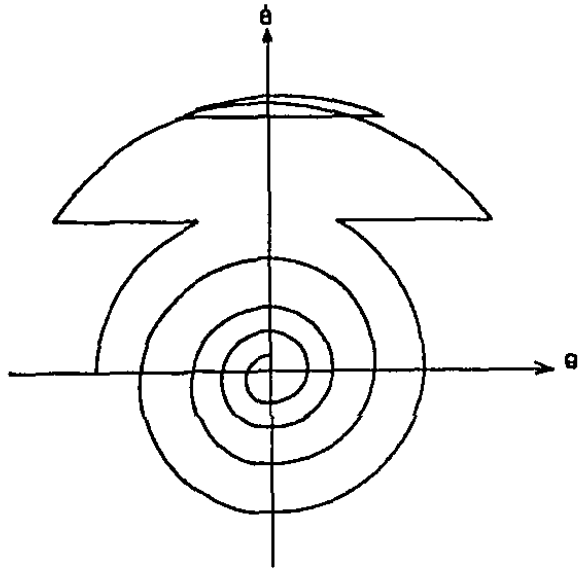


FIG. 4.5 SECUENCIA DE CONMUTACION EN EL PLANO DE FASE (4 PASOS)

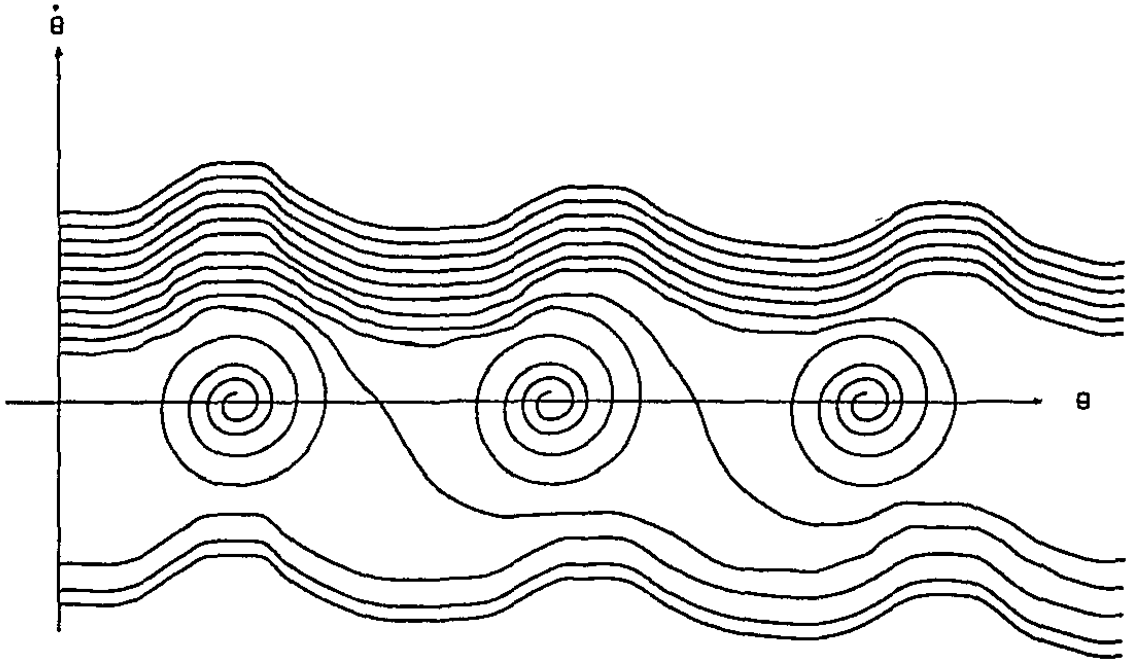


FIG. 4.6 PLANO DE FASE CON TRES POSICIONES DE EQUILIBRIO

representan movimientos cuya velocidad les impide llegar a los puntos de equilibrio mostrados y que llegarían a puntos de equilibrio situados más a la derecha.

Lo que Gauthier y equipo hicieron fue proponer una lista inicial de tiempos de conmutación y simular después el comportamiento del motor con base en la ecuación (4.8). Una vez simulado se representa el movimiento en el plano de fase y se encuentran por inspección aquellos puntos en los que la dinámica del motor difiera de la deseada. A partir de estos puntos se modifica la lista de tiempos de conmutación, adelantándolos o atrasándolos hasta conseguir que la simulación indique que el motor realiza adecuadamente el movimiento prescrito.

De la inspección del plano de fase que corresponde a un movimiento se pueden derivar condiciones más benignas para la aceleración y frenado, que es uno de los puntos críticos en los patrones de movimiento para motores de pasos. La Fig. 4.7 muestra la misma secuencia que la Fig. 4.5, pero con modificaciones en los tiempos de conmutación. Puede notarse que las oscilaciones que corresponden al último paso han desaparecido.

La técnica presenta limitaciones que la hacen poco manejable para condiciones normales. Las principales son las siguientes:

1) Se requieren conocer con precisión los valores de los parámetros del sistema.
2) Las secuencias de tiempos de conmutación son muy sensibles a pequeñas variaciones en las trayectorias.
3) Sólo se pueden manejar movimientos de pequeña magnitud (máximo decenas de pasos).

TECNICA DE VENKATARATNAM

Esta técnica esta diseñada para determinar las condiciones bajo las cuales es posible que un motor de pasos en reposo se sincronice con una secuencia de conmutación de una frecuencia dada. En tanto, únicamente considera el control de motores a velocidad constante. A diferencia de las demás técnicas, no hace diferencia entre los movimientos de arranque y de paro, con los de deslizamiento a velocidad constante.

El desarrollo que se presenta en la Ref. [27] se basa en un modelo eléctrico de un motor de pasos híbrido (ver Fig. 4.8), en el que se consideran los efectos de variación de permeancia. Por otro lado se

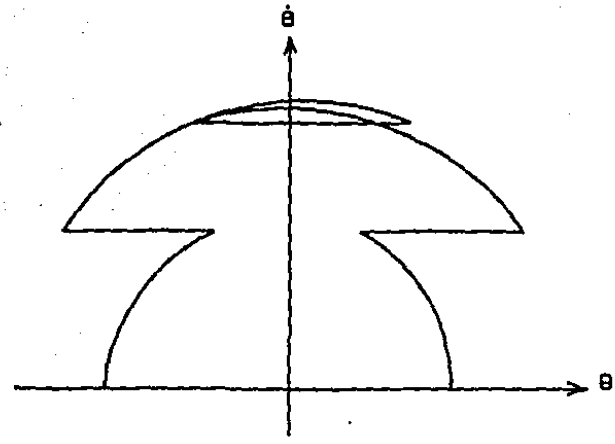


FIG. 4.7 SECUENCIA DE CONMUTACION MODIFICADA (4 PASOS)

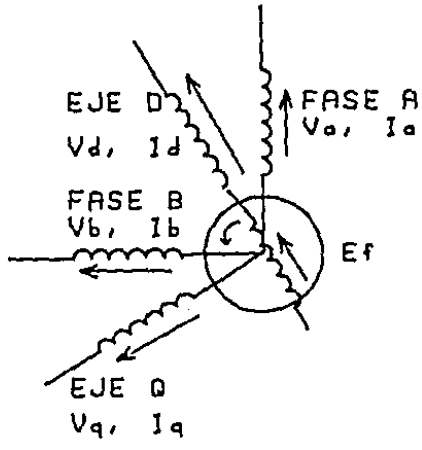


FIG. 4.8 MODELO ELECTRICO EMPLEADO POR VENKATARATNAM

deriva el modelo mecánico del movimiento. El modelo total resultante es de cuarto orden (ver capítulo 3 para más detalles).

A partir de este modelo, se deriva un sistema de ecuaciones diferenciales de cuarto orden. Este sistema se normaliza y se encuentran las coordenadas de los puntos singulares, que corresponden a los puntos de equilibrio estables e inestables.

El problema de arranque y sincronía de un de un motor de pasos se resuelve por analogía del comportamiento de estos con las máquinas eléctricas síncronas, por lo que se proponen ángulos de cuadratura y de deslizamiento. El sistema anterior se convierte en uno en el espacio fase y se procede entonces a derivar las condiciones bajo las cuales el motor puede seguir un movimiento propuesto, dada una velocidad de referencia.

Esta derivación se basa en una solución analítica por series de potencias de las ecuaciones en el espacio de fase, a partir de la cual se obtienen dos resultados básicos. El primero es que existe una velocidad crítica por arriba de la cual el motor no podrá sincronizarse con el movimiento que se le pide, y que cumple con:

$$2\zeta_e \sigma_c \cos \beta + \frac{2\zeta_m \sigma_c + J_\ell}{\cos \beta} = 1 \qquad (4.9)$$

donde σ_c =velocidad crítica, β =ángulo que depende de la constante de tiempo del circuito eléctrico del motor, ζ_e , ζ_m =los coeficientes normalizados de amortiguamiento interno y externo, J =carga normalizada.

El segundo resultado proporcionado se refiere al error de estado estable (δ) que se presenta cuando se pide al motor que se mueva a una velocidad por abajo de la crítica, error que cumple con:

$$\text{sen } (\delta - \beta) = 2\zeta_e \sigma \cos \beta + \frac{2\zeta_m \sigma + J_\ell}{\cos \beta} \qquad (4.10)$$

donde σ =velocidad de movimiento deseada tal que

$$|\sigma| < |\sigma_c| \qquad (4.11)$$

Los resultados obtenidos se validan experimentalmente y muestran buena concordancia con los esperado. En la Fig. 4.9 se muestra un ejemplo de una curva velocidad crítica versus constante de tiempo normalizada.

La principal limitación de la técnica estriba en que se limita a resolver el problema de movimientos a velocidad constante, y no puede manejar otro tipo de cinemática.

TECNICA DEL ANGULO DE ADELANTO O CONMUTACION

Esta técnica de lazo cerrado se diseñó para resolver el problema de mover un motor de pasos según patrones de movimiento similares a los que se muestran en la Fig. 4.10 (para más detalles ver la Ref. 6)

La idea en que se basa es la siguiente: cuando un motor de pasos se mueve a velocidad constante, se produce sincronía entre el tren de pulsos que ordena el movimiento y la velocidad a que se mueve el motor.

La sincronía descrita implica que las posiciones angulares en que ocurre la conmutación son invariantes en relación con los puntos de equilibrio estables. A esta posición angular invariante se le conoce como ángulo de conmutación (α), cuando se hace relativa a la posición de equilibrio desde la cual se hizo la comutación. En caso de que la posición se haga relativa a la siguiente posición de equilibrio, se le llama ángulo de adelanto (δ). En la Fig. 4.11 se describe gráficamente la interpretación de los ángulos de adelanto y atraso. Estos ángulos están relacionados por la siguiente expresión:

$$\alpha = 2\,\theta_p - \delta \qquad (4.12)$$

con θ_p = tamaño angular de un paso del motor.

Para la implantación de esta técnica es necesario dotar al motor de pasos con un sensor de posición. Se usa para ello un detector optoelectrónico y un disco que contiene tantas marcas como pasos por revolución proporcione el motor.

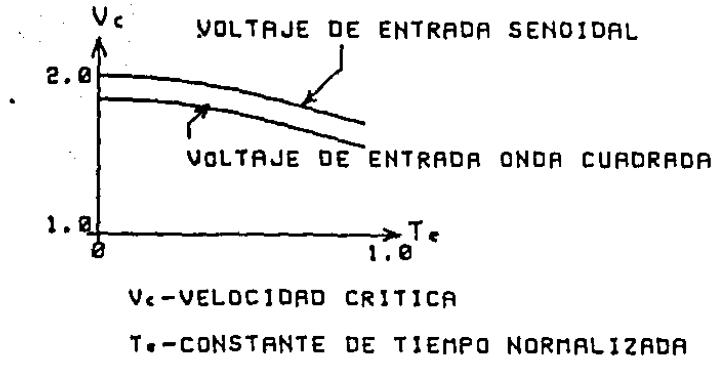


FIG. 4.9 VARIACION DE LA VELOCIDAD CRITICA CON LA CONSTANTE DE TIEMPO

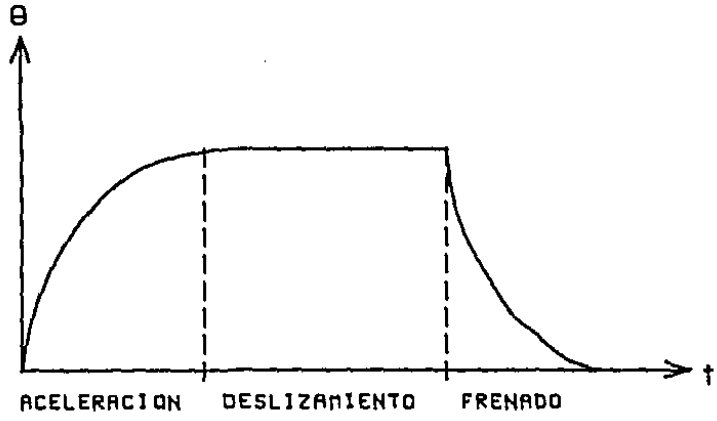


FIG. 4.10 PATRON DE MOVIMIENTO SIMPLE

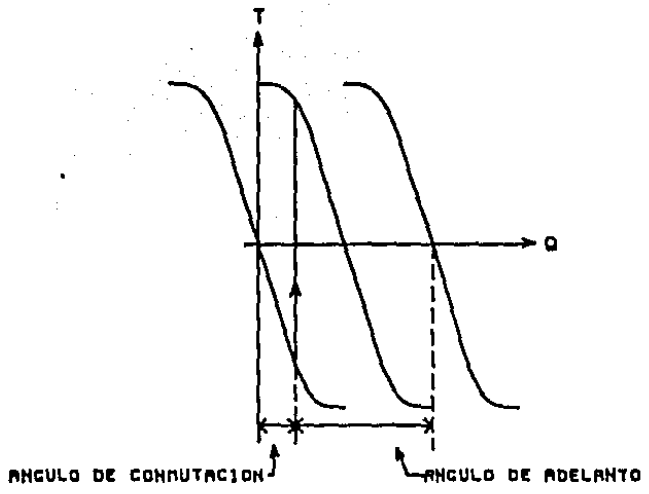


FIG. 4.11 ANGULOS DE CONMUTACION Y ADELANTO

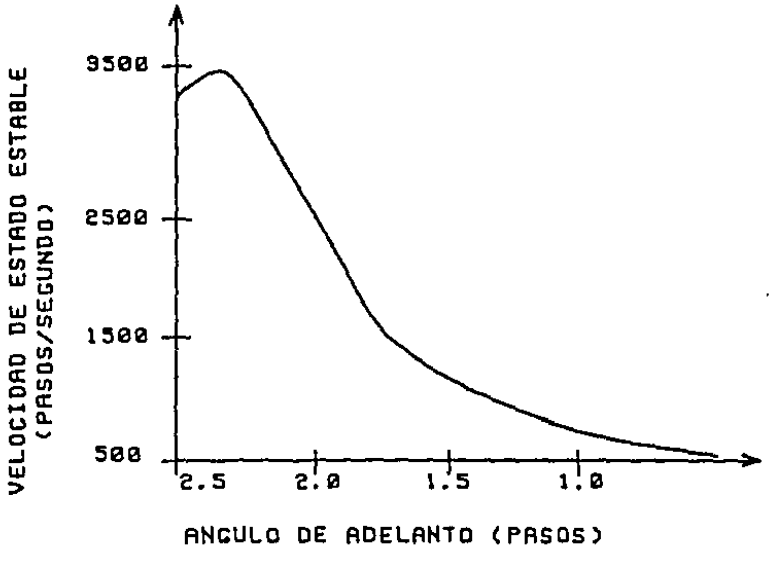


FIG. 4.12 VELOCIDAD VS. ANGULO DE ADELANTO

La velocidad de estado estable que se desea alcanzar en un motor de pasos es una función del ángulo de conmutación. En la Fig. 4.12 se muestra una curva típica para esta relación.

Para sincronizar el movimiento del motor en lazo cerrado, el disco con marcas se debe desplazar un valor igual al ángulo de conmutación que corresponda a la velocidad deseada. De esta forma las transiciones del optodetector por las perforaciones determinan precisamente el envío de nuevos pulsos.

Si se desea acelerar o frenar el motor, es necesario alterar la sincronía descrita. Para lograrlo se debe modificar el tiempo entre pulsos. Si el tiempo se alarga, se produce un frenado, y en caso contrario, aceleración del motor. El efecto práctico de estos retrasos y adelantos de tiempo sobre el ángulo de conmutación en las curvas par-posición angular, es el de adelantar o atrasar el ángulo de conmutación. En esta técnica a los pulsos que se envían fuera de sincronía se les conoce como pulsos inyectados.

Estos pulsos inyectados corresponden a pulsos adicionales y su efecto se puede apreciar en la Fig. 4.13 que muestra dos secuencias de conmutación, una que corresponde a un movimiento a velocidad constante y otra a un movimiento después de la inyección de un pulso. Puede notarse que en este caso la inyección del pulso produce un frenado del motor pues se aplica un par en promedio negativo.

Como resulta evidente, es totalmente impráctico mover el disco con marcas cada vez que se cambia la velocidad de estado estable. Para evitar este problema se coloca el disco en una posición fija y se introduce un retraso para enviar los pulsos cuya magnitud depende de la velocidad angular deseada. Este retraso se calcula a partir del tiempo que tarda el motor en pasar desde la posición de equilibrio hasta la del ángulo de conmutación, cuando el recorrido se realiza a la velocidad deseada.

Aunque la técnica ha sido diseñada para emplearse en lazo cerrado, resulta claro que su aplicación se puede también llevar a cabo en lazo abierto. Para este último caso se debe poner especial énfasis en los movimentos de arranque y paro, pero una vez alcanzada la velocidad de estado estable deseada, el proceso resulta bastante seguro.

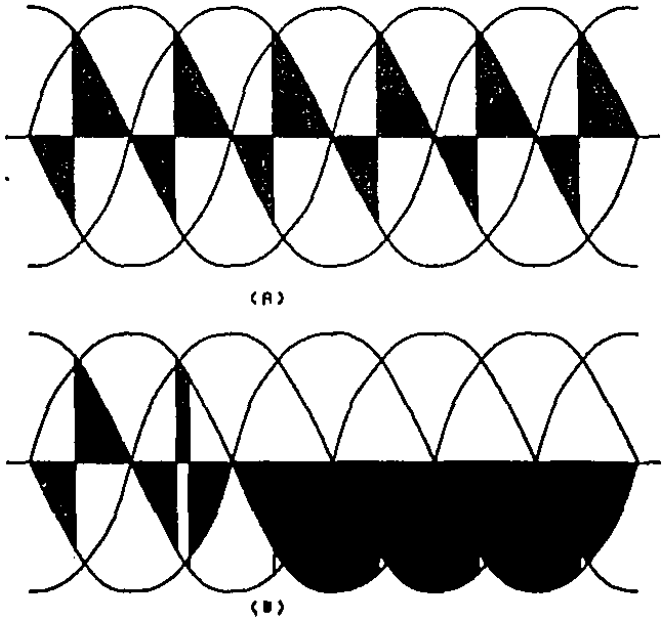


FIG. 4.13 PULSOS INYECTADOS (A) SECUENCIA SIN INYECCION
(B) SECUENCIA CON INYECCION

EFECTO SOBRE EL MOVIMIENTO	PUNTOS DE EQUILIBRIO			
	1	2	3	4
DERECHA LENTO	B	C	D	A
IZQUIERDA LENTO	A	B	C	D
DERECHA MEDIO	C	D	A	B
IZQUIERDA MEDIO	D	A	B	C
PARO	AB	CB	CD	AD
DERECHA RAPIDO	CB	CD	AD	AB
IZQUIERDA RAPIDO	AD	AB	CB	CD
MUY RAPIDO (*)	CD	AD	AB	CB

(*)EN AMBAS DIRECCIONES, SOLO PARA VELOCIDAD NO NULA

TABLA 4.1 DIAGRAMA DE TRANSICION DE ESTADOS PARA LA TECNICA DE CONTROL REALIMENTADO

TECNICA DE CONTROL REALIMENTADO

Esta técnica fue diseñada por Frediksen (Ref. 14) y como se mencionó en la introducción a este capítulo se trata de una técnica que funciona en lazo cerrado. Se pretende que a través de la realimentación el motor de pasos puede funcionar como un servomotor que responda en tiempo mínimo. Se consideran únicamente las dinámicas de primero y segundo orden (posición y velocidad). Su descripción grosso modo es como sigue.

A partir de una análisis de los diagramas de transición de estados de las conmutaciones de energía en los embobinados, y de las curvas par-posición angular mencionadas en el capítulo 3, se puede deducir que dichas conmutaciones equivalen a la aplicación de controles discretos para el movimiento del motor. En la Tabla 4.1 se muestra el diagrama de transición de estados, incluyendo su interpretación en términos del efecto que produce sobre el movimiento del motor. Cabe aclarar que Frediksen utiliza todas las posibles conmutaciones de energía en los embobinados, lo que arroja un total de ocho controles diferentes.

El trabajo de Frediksen consistió en la implantación de un sistema de realimentación de la posición y de un tacómetro digital sencillo (ver Fig. 4.14), a partir del cual se puede obtener información sobre la transición de estados que más conviente para minimizar el error de seguimiento. Las aplicaciones que se muestran en la referencia citada se refieren únicamente a movimientos punto a punto con velocidad máxima restringida.

El sistema implantado es equivalente a un autómata finito que divide los movimientos en cuatro zonas siempre iguales: dos zonas de aceleración y dos de frenado. A cada zona corresponde una secuencia de conmutación que puede incluir una o dos fases energizadas. Cuando se energiza una fase se trata de un movimiento de aceleración o frenado lento. Cuando se energizan las dos fases se trata de aceleraciones de frenado de la máxima magnitud posible (ver Fig. 4. 15).

La política de control que se sigue consiste en analizar instantáneamente el error de posición y de velocidad que existe. Si existe un gran error de velocidad, se usa un movimiento a máxima aceleración. Cuando el error de velocidad disminuye en magnitud, se cambia a los movimentos de aceleración mínima.

Para conocer la posición se cuenta con un contador con el número de pasos ordenados inicialmente, cuyo valor decrece según la señal de los optoemisores (Fig. 4.16). La velocidad se estima con base en una

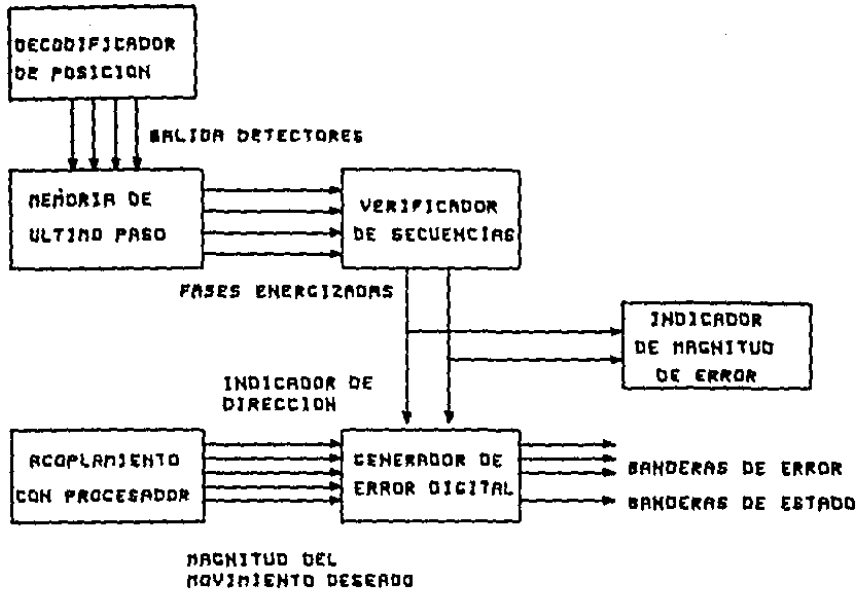


FIG. 4.14 TACOMETRO DIGITAL PARA CONTROL REALIMENTADO

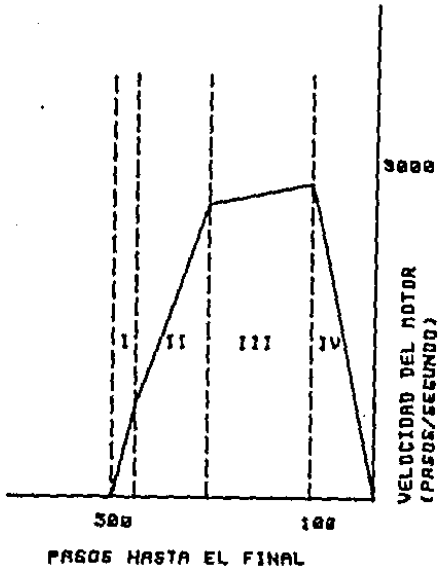


FIG. 4.15 CURVAS DE ACELERACION Y FRENADO
PARA CONTROL REALIMENTADO

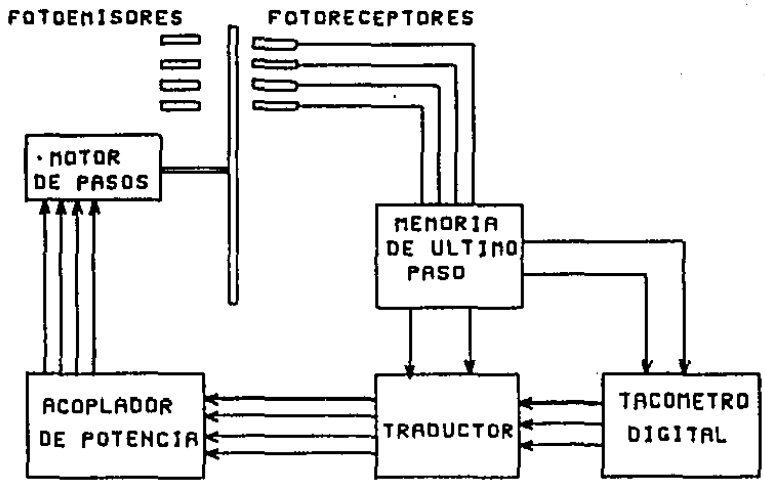


FIG. 4.16 ESQUEMA GENERAL DEL SISTEMA DE CONTROL REALIMENTADO

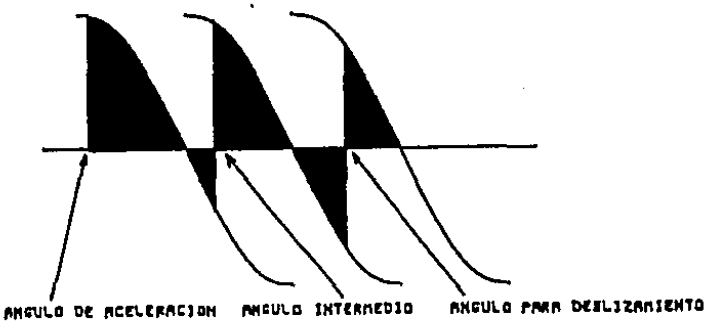


FIG. 4.17 ANGULO INTERMEDIO EN TRANSICION ACELERACION-DESLIZAMIENTO

interpolación lineal entre los dos últimos pasos ejecutados, pero para la instrumentación del control realimentado sólo es necesario tener una idea gruesa de la misma, ya que el error de velocidad sólo determina el tipo de aceleración a emplear.

El trabajo de Frediksen presenta la aplicación de los motores de pasos como servomotores, la cual coincide con la tendencia más reciente en el uso de los motores de pasos, en este sentido se trata de una experiencia precursora

Se pueden notar, sin embargo, las siguientes desventajas: se propone el uso de todas las formas de conmutación posible, lo que implica que no es posible garantizar la robustez de los esquemas de control, y que podrán presentarse errores importantes en el desempeño del seguimiento cuando exista variación en la carga, o problemas de alineamiento en los sensores ópticos.

TECNICA DE LEENHOUTS

Esta técnica de lazo abierto, que se reporta en la Ref. 15, está diseñada para lograr que un motor de pasos pueda seguir algún perfil de velocidad. Como resultado del empleo de la técnica se obtiene la tabla de los tiempos de conmutación de energía en los embobinados.

En la descripción de esta técnica se introduce el concepto de par promedio ($\bar{T}$), como aquel que se obtiene al promediar el par que proporciona un motor de pasos dado un desplazamiento angular determinado, la expresión para calcularlo es:

$$\bar{T} = - \frac{T}{\theta_p} \int_{\theta_1}^{\theta_2} \text{sen } \theta \, d\theta \qquad (4.13)$$

donde θ_1 y θ_2 representan los ángulos inicial y final del desplazamiento dado, y los demás términos se han definido con anterioridad. La integración de (4.13) conduce a:

$$\bar{T} = \frac{T}{\theta_2 - \theta_1} (\cos \theta_2 - \cos \theta_1) \qquad (4.14)$$

Una vez hallado el par promedio se puede aproximar la aceleración promedio disponible como:

$$\ddot{\bar{\theta}} = \frac{\bar{T} - F}{J} \tag{4.15}$$

Adicionalmente, Leenhouts introduce un factor de utilización F_t, que equivale a limitar la zona de las curvas par-posición angular sobre la que se pueden realizar conmutaciones. Este factor debe cumplir con:

$$-\frac{\theta_p}{2} - \text{sen}^{-1}(F_t) \leq \theta \leq \frac{\theta_p}{2} + \text{sen}^{-1}(F_t) \tag{4.16}$$

(intervalo de validez para Ft).

El tipo de movimientos para los que se aplica esta técnica se muestra en la Fig. 4.10. El movimiento total se divide en cinco etapas: arranque, aceleración, deslizamiento, desaceleración y paro. A continuación se describe brevemente cada una.

El arranque se maneja bajo la suposición de que el motor se encuentra en reposo y que se debe realizar una conmutación de la posición de equilibrio a la siguiente fase.

La aceleración se realiza siempre con el máximo valor posible según lo indiquen las ecuaciones (4.15) y (4.16). En seguida se calcula la velocidad con la que el motor alcanza el siguiente punto de equilibrio estable (θ_o). Para ello se calcula primero la posición de este último, que está dada por:

$$\theta_o = -\text{sen}^{-1}\left[\frac{F}{T}\right] \tag{4.17}$$

La velocidad en dicho punto (ω_o) se calcula considerando que el movimiento del motor se realiza con aceleración uniforme, su valor es:

$$\omega_o = \sqrt{2 \cdot \ddot{\bar{\theta}} \frac{\theta_o - \theta}{\theta_p} + \omega_i^2} \tag{4.18}$$

donde ω_i es la velocidad inicial.

El tiempo para la conmutación se encuentra de:

$$t = \frac{\theta_o - \theta}{\theta_p} \cdot \frac{2}{\omega_o + \omega_i} \qquad (4.19)$$

Si la velocidad alcanzada es menor que la de deslizamiento propuesta se continúa con la rutina de aceleración. Si es aproximadamente igual, se pasa a la rutina de deslizamiento. En el último caso, cuando la velocidad es mayor que la deseada, se pasa a la rutina de frenado.

En la rutina de deslizamiento se calcula el ángulo de conmutación (α) como el necesario para que el motor venza la fuerza de fricción. Este ángulo se encuentra de:

$$\alpha = \frac{\theta_p}{2 \operatorname{sen} \frac{\theta_p}{2}} \operatorname{sen}^{-1} \left(\frac{-F}{T}\right) + \frac{\theta_s}{2} \qquad (4.20)$$

Existe un problema adicional que proviene del hecho de que el ángulo que se calcula en la ecuación (4.20) corresponde con el que se debe mantener cuando el motor se desliza a la velocidad deseada (ω_D), pero no toma en cuenta que cuando se pasa de la rutina de aceleración a la de deslizamiento, se debe encontrar un ángulo de conmutación intermedio que permita garantizar que al empezar con la secuencia de pulsos de deslizamiento el motor se mueva a la velocidad deseada, y no a la que tiene al terminar de acelerar. Para lograr lo anterior se usan de nueva cuenta las ecuaciones (4.13) y (4.14), y se considera que el desplazamiento en cuestión se realiza desde la última posición de aceleración hasta una situada dos pasos adelante. El ángulo para el paso intermedio (θ_i) se calcula de (ver Fig. 4.17):

(4.21)

$$\theta_i = \frac{\theta_p}{2} + \operatorname{sen}^{-1}\left[\frac{\theta_p(\cos\theta_c - \cos\theta_o)\,\theta_p T - F(\theta_c - \theta_o + \theta_p) - (\omega_c - \omega_o)\theta_p^{\,2} J}{2 \operatorname{sen}\left(\frac{\theta_p}{2}\right) T\,\theta_p}\right]$$

El movimiento de desaceleración se realiza de manera inversa al de aceleración. Se considera que se frena al motor con el par más negativo que se pueda obtener de (4.15) y (4.13).

La rutina de paro se basa en la ecuación (4.21), valuada para velocidad final nula.

Los resultados experimentales indican buena concordancia con los esperados. La técnica es apropiada para resolver movimientos de pequeña magnitud y con patrones como el que describe la Fig. 4.10, pero resulta de poca utilidad en otras circunstancias.

TECNICA DE KUO

La técnica de Kuo se basa en la simulación detallada del movimiento de un motor de pasos y en las ideas de ángulo de conmutación y de adelanto, y de pulsos inyectados, que se describieron en un inciso anterior (ver Ref. 6 para más detalles).

El modelo que usa Kuo es equivalente al que usa Venkataratnam y al que se deriva en el capítulo 3. La técnica consiste en simular el comportamiento del motor de pasos y hallar por aproximaciones sucesivas los ángulos y tiempos de conmutación, y los tiempos para los pulsos inyectados.

El número de pulsos a inyectar, la forma de frenado y demás características importantes de los desarrollos de Kuo, requieren de experiencia y conocimiento profundo de los motores de pasos. Los resultados que se obtienen son muy buenos, aunque se limitan a movimientos de pequeña magnitud.

5. ROBUSTEZ DE LAS SECUENCIAS DE CONMUTACION

Del análisis de las técnicas descritas en el capítulo anterior se desprenden las siguientes conclusiones generales:

a) En ninguno de los métodos propuestos se consideran los efectos de la variación de las cargas nominales sobre el movimiento del motor de pasos.

b) Salvo la técnica de Leenhouts en todas las demás se permiten conmutaciones en cualquier región de las curvas par-posición angular.

A partir de los puntos anteriores surge la necesidad de estudiar la robustez de las conmutaciones frente a variaciones de los parámetros del motor y su sistema de cargas.

En este capítulo se deducen las condiciones de las que depende que una secuencia de conmutación en la energía de los embobinados sea robusta.

El análisis que se propone se puede realizar a partir de modelos completos o simplificados de los motores de pasos. Se consideró que esta última alternativa era más atractiva desde el punto de vista de diseño, pues los datos básicos que se requieren para este tipo de modelos se

pueden obtener de los datos del fabricante y de mediciones sencillas de los parámetros del sistema de cargas. De esta manera sería posible que un usuario potencial del método pudiera usar los resultados sin conocer profundamente el funcionamiento de los motores de pasos. Además se decidió tomar en cuenta únicamente sistemas de cargas inerciales, dado que la técnica descrita en el capítulo anterior para cargas viscosas, establece claramente las condiciones para un buen desempeño.

El capítulo se divide en dos partes, la primera presenta un análisis de estabilidad del comportamiento del motor de pasos considerando únicamente una posición fija en los voltajes de los embobinados. La segunda deriva las condiciones bajo las cuales se puede garantizar la robustez de las secuencias de conmutación.

ESTABILIDAD DE LOS MOTORES DE PASOS

La estabilidad de los motores de pasos se puede estudiar considerando el modelo simplificado que sigue:

$$\begin{aligned} J\ddot{\theta} + B\dot{\theta} &= \tau \\ \tau &= - T \operatorname{sen} [\theta] \end{aligned} \qquad (5.1)$$

Si se integran en una sola ecuación se llega a:

$$J\ddot{\theta} + B\dot{\theta} + T \operatorname{sen} \theta = 0 \qquad (5.2)$$

como puede notarse la ecuación (5.2) coincide con la ecuación que describe el movimiento de un péndulo amortiguado, que es una ecuación diferencial no lineal de segundo orden.

La estabilidad en el sentido Lyapunov (Ref. 16) de dicho sistema se garantiza si dado un conjunto de condiciones iniciales θo y ωo, el sistema llega a una posición de equilibrio estable.

La ecuación (5.2) reescrita en variables de estado es:

$$\begin{aligned} \dot{\theta} &= \omega \\ \dot{\omega} &= - \frac{B}{J} \omega - \frac{\tau}{J} \operatorname{sen} \theta \end{aligned} \qquad (5.3)$$

La pendiente de las curvas posición-velocidad angular en el plano de fase para el sistema (5.3) está dada por:

$$\frac{\dot{\omega}}{\dot{\theta}} = \frac{-\frac{B}{J}\omega - \frac{T}{J}\operatorname{sen}\theta}{\omega} \tag{5.4}$$

Los puntos de equilibrio se obtienen cuando la derivada es nula en (5.3) y son:

$$\theta = n\pi \qquad n \in E \tag{5.5}$$

Para determinar cuales son los puntos de equilibrio estables se utiliza el segundo método de Lyapunov (Ref. 17) que se puede plantear a través del siguiente:

Teorema 5.1.- Un sistema dinámico $\dot{\theta} = f(\theta)$ es asintóticamente estable en la vecindad de un punto de equilibrio θo si existe una función escalar $V(\theta)$, tal que:

$$\begin{aligned}
&\text{i) } V(\theta) \text{ es continua y tiene primeras derivadas parciales continuas en una vecindad S del punto de equilibrio.}\\
&\text{ii) } V(\theta) > \emptyset \;;\; \theta \neq o\\
&\text{iii) } V(\theta o) = \emptyset\\
&\text{iv) } \dot{V}(\theta) < 0 \;;\; \theta \neq o
\end{aligned} \tag{5.6}$$

las funciones $V(\theta)$ se conocen como funciones de Lyapunov.

Para sistemas mecánicos, la energía total del sistema cumple con las condiciones del Teorema 5.1 para ser considerada como una función de Lyapunov. En efecto si la energía total del sistema se define como:

$$V(\theta) = \frac{1}{2} J \dot{\theta}^2 + T(1 - \cos\theta) \tag{5.7}$$

su derivada está dada por:

$$\dot{V}(\theta) = \dot{\theta}\ddot{\theta} + T \operatorname{sen}\theta\,\dot{\theta} = -\frac{B}{J}\dot{\theta}^2 \tag{5.8}$$

que se puede comprobar cumple con dichas condiciones.

La interpretación de la elección de (5.7) como función de Lyapunov es que para que un motor de pasos vaya hacia una posición de equilibrio estable debe encontrarse en la zona de las curvas par-posición angular que poseen pendiente negativa (ver Fig. 5.1).

El teorema 5.1 garantiza que el movimiento de un motor de pasos lo llevará al punto de equilibrio estable sólo si se encuentra en la vecindad del mismo. Sin embargo, no dice nada respecto a los casos en que esa condición no se cumple. Al respecto se puede plantear la siguiente condición.

La máxima energía de frenado (E) que puede proporcionar un motor de pasos para una posición fija de los embobinados está dada por:

$$E = T \int_{o}^{\pi/2} \text{sen } \theta \, d\theta = T \tag{5.9}$$

Si se considera que el motor tiene una posición angular θ_o y una velocidad $\dot{\theta}$, entonces se puede asegurar que el motor de pasos llegará a la posición de equilibrio más próxima si y sólo si se cumple que:

$$\frac{1}{2} J \dot{\theta}_o^{\,2} + T[1 - \cos \theta_o] \leq T \tag{5.10}$$

La interpretación de (5.10) en el movimiento de los motores de pasos justifica el fenómeno de pérdida o ganancia de pasos durante el frenado, cuando los motores de pasos se acoplan a cargas inerciales, pues para velocidades altas de movimiento la energía cinética del motor es mucho mayor que la capacidad de frenado de este.

Además si se dividen ambos lados de la desigualdad entre J, la inercia, se concluye que la posibilidad de que la desigualdad se siga cumpliendo depende de la relación T/J. Esto es, que cuando se cuenta con un motor grande para mover una carga relativamente pequeña es posible acelerar o frenar dicha carga con brusquedad. Este hecho explica algunas de las experiencias descritas en la literatura y que se detallaron en el capítulo anterior.

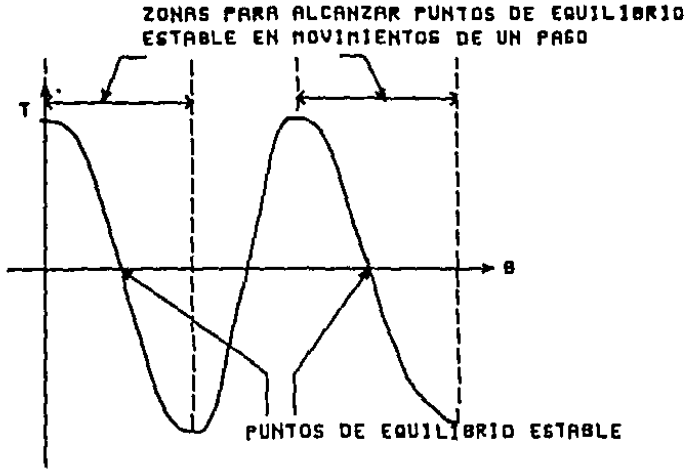


FIG. 3.1 ESTABILIDAD EN MOVIMIENTOS DE UN PASO

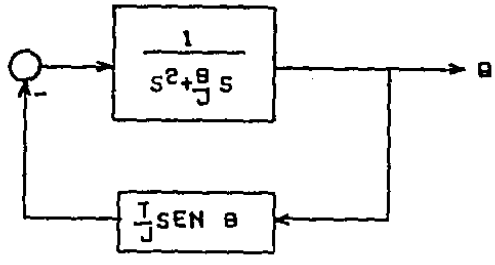


FIG. 5.2 MOTOR DE PASOS COMO UN SISTEMA LINEAL CON REALIMENTACION NO LINEAL

ROBUSTEZ DE LAS SECUENCIAS DE CONMUTACION

En este inciso se pretende tomar en cuenta el efecto de la variación en los parámetros del sistema de cargas de un motor, cuando ya se toman en cuenta las conmutaciones de energía en los embobinados.

Supóngase que se tiene una secuencia de conmutación dada por la función K(t), donde K es una función entera de argumento real. Los valores de K(t) cumplen con:

$$K(t) = \{1,2,3,4\} \tag{5.11}$$

y el argumento real, o tiempo, se puede partir en un conjunto de subintervalos $t_1, t_2, \ldots, t_n$ tal que:

$$t_1 < t_2 < \ldots\ldots < t_n \tag{5.12}$$

y que además:

$$K(t) = K(t_i) \quad \forall \quad t_i \leq t < t_{i+1} \quad ; \; i=1, \ldots, t_{n-1} \tag{5.13}$$

Considérese que el sistema motor-carga se puede especificar dados los parámetros J_1, B_1, T_1. El efecto de K(t) sobre este sistema se puede describir a través de la función $\theta_1(t)$ que satisface la ecuación:

$$J_1 \ddot{\theta}_1(t) + B_1 \dot{\theta}_1(t) = T_1 \operatorname{sen} [\alpha\theta_1(T) + \beta K\omega] \tag{5.14}$$

donde α y β se obtienen de (3.1).

Propóngase ahora un nuevo juego de parámetros J_2, B_2, T_2, la misma función de conmutación K(t) producirá ahora una trayectoria $\theta_2(t)$ descrita por:

$$J_2 \ddot{\theta}_2(t) + B_2 \dot{\theta}_2(t) = T_2 \text{ sen } [\alpha \theta_2(t) + \beta K(t)] \qquad (5.15)$$

defínase el error como:

$$\varepsilon(t) = \theta_2(t) - \theta_1(t) \qquad (5.16)$$

y las diferencias paramétricas como:

$$\begin{aligned} \delta J &= J_2 - J_1 \\ \delta B &= B_2 - B_1 \\ \delta T &= T_2 - T_1 \end{aligned} \qquad (5.17)$$

De (5.14) a (5.17) se puede determinar que el error satisface la ecuación:

$$(J_1 + \delta J)\ (\ddot{\theta}_1(t) + \ddot{\varepsilon}(t)) + (B_1 + \delta B)\ (\dot{\theta}_1(t) + \dot{\varepsilon}(t)) = \qquad (5.18)$$
$$= (T_1 + \delta T) \text{ sen } [\alpha(\theta_1(t) + \varepsilon(t)) + \beta K(t)]$$

Si se supone que los cocientes de (5.17) a los parámetros originales son pequeños, se pueden ignorar los términos de segundo orden en la diferencia de (5.18) y (5.14), lo cual proporciona la siguiente ecuación diferencial lineal variante con el tiempo:

$$J_1 \ddot{\varepsilon}(t) + B_1 \dot{\varepsilon}(t) + \varepsilon(t) \cos [\alpha \theta_1(t) + \beta K(t)] =$$
$$\qquad (5.19)$$
$$\delta J \ddot{\theta}_1(t) - \delta B \dot{\theta}_1(t) + T_1 \text{ sen } [\alpha \theta_1(t) + \beta K(t)]$$

Para estudiar la estabilidad de (5.19) se recurre al Criterio del Círculo (Ref. 16). Este se desarrolló para estudiar la estabilidad de sistemas con realimentación no lineal. Como puede notarse de la Fig. 5.2 que muestra un diagrama de bloques del modelo de un motor de pasos, el efecto de la variación del par con la posición angular se puede tratar considerando la realimentación de un término no lineal.

El Criterio del Círculo se puede enunciar a través del siguiente

Teorema 5.2.- El sistema dinámico descrito por la ecuación

$$\ddot{x}(t) + a\,\dot{x}(t) + g(t)\,K(t) = \gamma(t) \tag{5.20}$$

con a > 0 es estable en el sentido entrada-salida, si existe un valor $\gamma > 0$, tal que:

$$\gamma^2 \leq g(t) \leq (\gamma + a)^2 \tag{5.21}$$

El resultado básico del Criterio del Círculo indica que la ecuación (5.19) tendrá una solución acotada cuando se cumpla que:

$$\gamma^2 \leq -\frac{T}{J} \cos\left[\alpha\,\theta_1(t) + \beta\,K(t)\right] \leq (\gamma + \frac{B}{J})^2 \tag{5.22}$$

se define la función m(t) como:

$$m(t) = \alpha\,\theta_1(t) + \beta\,K(t) \tag{5.23}$$

la desigualdad se puede reescribir como:

$$\cos\left[m(t)\right] \geq -\frac{J}{T}\,(\gamma + \frac{B}{J})^2 \tag{5.24}$$

El análisis se realiza ahora desde un punto de equilibrio estable de las curvas par-posición, lo que implica hacer:

$$m(t) = (2n+1)\pi \quad ; \quad n \in E \tag{5.25}$$

Para la primera desigualdad de (5.24) esto implica que:

$$-1 \geq -\frac{J}{T}\,(\gamma + \frac{B}{J})^2 \tag{5.26}$$

o bien en términos de γ :

$$\gamma \geq [\sqrt{\frac{T}{J}} - \frac{B}{J}] \tag{5.27}$$

Enseguida se averiguan los límites para los cuales se cumple que $\gamma > 0$, lo que se consigue valuando (5.27) para $\gamma = 0$, que es:

$$\frac{B}{\sqrt{JT}} \geq 1 \tag{5.28}$$

En (5.28) se define el coeficiente de amortiguamiento relativo del sistema como:

$$\xi = \frac{B}{2\sqrt{JT}} \tag{5.29}$$

de (5.29) en (5.28) se concluye que la primera se satisface siempre que cumpla con:

$$\zeta > 1/2 \quad \Rightarrow \quad \gamma = 0 \tag{5.30}$$

y la desigualdad (5.22) se satisface para cualquier valor de m(t) tal que cos(m(t)) sea negativo, lo que implica situarse en la parte negativa de las curvas par-posición angular. Este resultado coincide con el que se obtiene de la ecuación (5.8).

Si se toma ahora la otra parte de la desigualdad (5.24) y se substituye (5.27) en ella se llega a:

$$\cos [m(t)] \leq -(1 - \frac{B}{\sqrt{JT}})^2 \tag{5.31}$$

la cual expresada según (5.29) da:

$$\cos[m(t)] \leq -(1 - 2\xi)^2 \tag{5.32}$$

La ecuación (5.32) impone un límite al valor del cos(m(t)), que depende de los parámetros α y β de (5.23), es decir, ξ no puede disminuir arbitrariamente so pena de no satisfacer (5.32). Para el caso

que se estudia, el de un motor híbrido de cuatro fases el mínimo valor de ξ aceptable es 0.08. Para este valor las conmutaciones deben ocurrir cuando m(t).cumple con:

$$m(t) = \pi \ (n + 1/4) \tag{5.33}$$

además de que:

$$\cos (m(t)) < 0 \tag{5.34}$$

lo que implica que para conmutaciones robustas se debe conmutar únicamente en ese ángulo.

Conforme el valor de ξ pasa de 0.08 a 1/2, la zona en la que es posible realizar conmutaciones crece. La Fig. 5.3 muestra cuatro casos de zonas de conmutación para diferentes valores de ξ .

Cuando el valor de ξ es mayor que 1/2, las conmutaciones son robustas siempre que se cumpla con (5.34).

La Fig. 5.4 muestra dos secuencias de conmutación, una robusta y la otra no.

Existe una interpretación cualitativa para los resultados obtenidos que se puede resumir en los siguientes puntos:

i) Para que en presencia de conmutaciones se preserve la estabilidad, es necesario que los movimientos realizados en cada paso cumplan con las condiciones que se derivaron para movimientos de un sólo paso, que indican que para permanecer en las curvas par posición se debe permanecer en la zona de pendiente negativa.

ii) Cuando las conmutaciones se realizan en la zona de pendiente negativa se presenta un efecto de corrección. La situación se plantea en la Fig. 5.5 y se explica a continuación.

Supóngase que por la presencia de perturbaciones la velocidad del motor ha disminuido en relación a la deseada. En esta situación el ángulo en el cual se presentará la conmutación se retrasará con respecto al originalmente previsto. Este retraso producirá un aumento neto en la

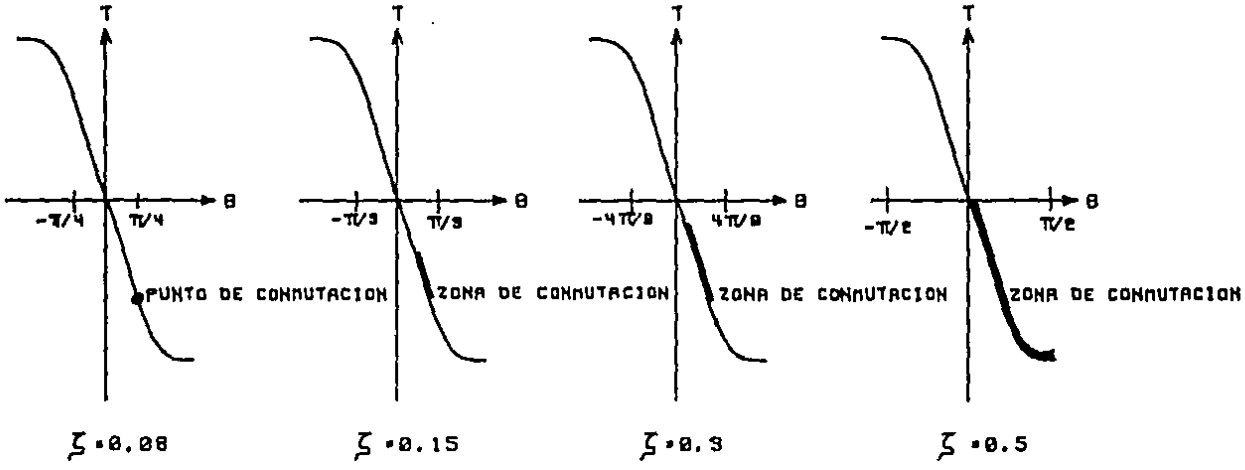


FIG. 5.9 ZONAS PARA CONMUTACIONES ESTABLES VS, COEFICIENTE DE AMORTIGUAMIENTO RELATIVO

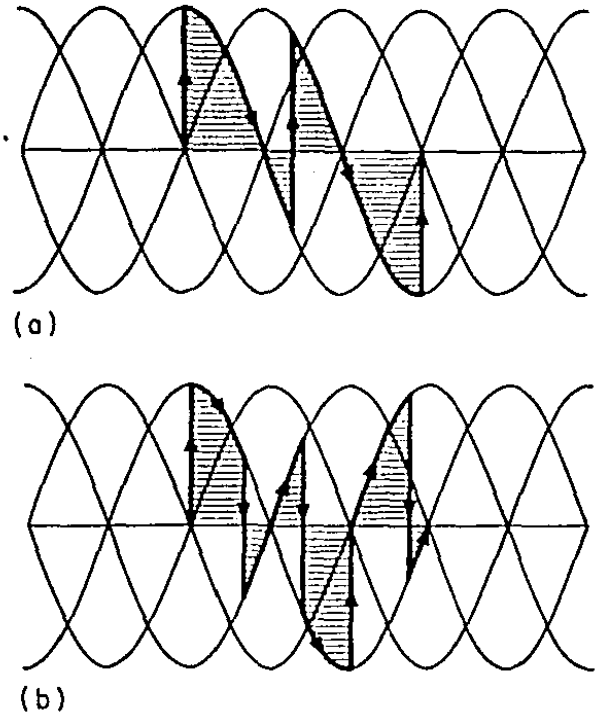


FIG. 5.4 SECUENCIAS DE CONMUTACION:
(a) ROBUSTA
(b) NO ROBUSTA

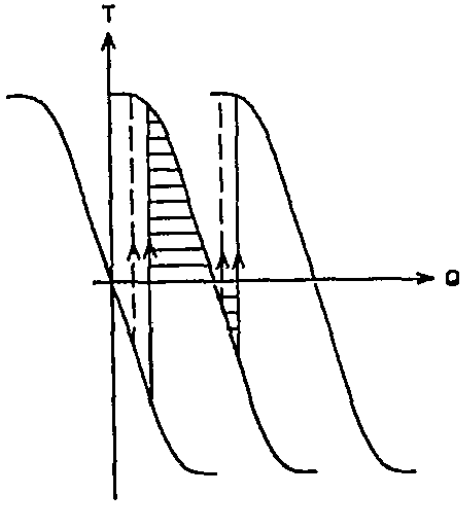


FIG. 5.5 EFECTO CORRECTIVO DEL PAR EN LA ZONA DE PENDIENTE NEGATIVA

cantidad de energía que recibe el motor durante el trayecto del siguiente paso (ver la líneas punteadas en la Fig. 5.5). Con el aumento de energía .la velocidad aumentará hasta hacerse igual a la deseada. La explicación se puede reproducir para los casos en que la velocidad es mayor que la deseada y se notará que en dicho caso la energía neta proporcionada es menor que la necesaria, lo que frenará al motor.

Los desarrollos presentados son independientes de la dirección del movimiento, por lo que los resultados son válidos también para desplazamientos en la dirección negativa de θ .

Los resultados presentados en este capítulo se publicaron primeramente en la Ref. 28 .

6. METODOLOGIA DE CONTROL PROPUESTA

Una vez que en el capítulo anterior se derivaron las condiciones para encontrar secuencias robustas de conmutación para el movimiento de los motores de pasos, hace falta proponer un esquema de control que las incorpore y que ofrezca además ventajas en relación con los que se plantearon en el capítulo 4.

Las conclusiones generales a las que se puede llegar del análisis de las técnicas de control para motores de pasos que se han descrito son:

a) Las técnicas han sido diseñadas para patrones de movimiento sencillos: máxima aceleración, deslizamiento y frenado.

b) Sólo se consideran movimiento de pequeña magnitud angular, del orden de las decenas de pasos.

c) Las técnica que da mejor resultados se basa en un proceso iterativo regido por reglas heurísticas.

En este capítulo se pretende proponer una técnica de control para motores de pasos que sea robusta y que además considere la posibilidad de que el motor de pasos siga patrones arbitrarios de movimiento. La técnica no debe estar basada en procedimientos heurísticos, ni implicar un conocimiento profundo de los motores de pasos.

Como resultado del empleo de esta técnica se debe obtener una secuencia de tiempos de conmutación que hagan que el motor de pasos siga la trayectoria especificada.

El desarrollo de la técnica se presenta en cuatro partes, en la primera se derivan las condiciones bajo las cuales se puede sustituir el par instántaneo que proporciona el motor de pasos, por un par promedio. La segunda parte se dedica a calcular el ángulo de conmutación que corresponde a un par promedio dado. En la tercera parte se deriva la ley de control que se empleará en la técnica que se propone, y por último se relaciona el cálculo del ángulo de conmutación con el del tiempo de conmutación.

EQUIVALENCIA PAR INSTANTANEO-PAR PROMEDIO

Para conseguir que la secuencia de tiempos que se obtendrá como resultado del empleo de la técnica sea manejable, su tamaño debe minimizarse. Para lograr esto se ha considerado que sólo debe efectuarse una conmutación por cada recorrido del motor sobre las zonas de conmutación que se describieron en el capítulo anterior, lo que equivale a realizar una sola conmutación por paso del motor.

Para poder justificar adecuadamente esta restricción se debe garantizar que existe una equivalencia entre el efecto de un par que varía a lo largo de un cierto desplazamiento angular y un par promedio constante que se aplica durante el mismo (ver Fig. 6.1).

Cónsiderese de nueva cuenta la energía total del motor de pasos:

$$V(\theta) = \frac{1}{2} J \dot{\theta}^2 + T[1 - \cos \theta] \qquad (6.1)$$

y la máxima energía que se puede proporcionar durante un paso del motor, que está dada por:

$$E = -T \int_0^{\pi/2} \operatorname{sen} \theta \, d\theta = T \qquad (6.2)$$

Supóngase ahora que el motor se encuentra en un punto de par nulo, en estas condiciones se puede anular el último término del lado derecho de (6.1), y además que la energía cinética en dicho punto es un múltiplo de

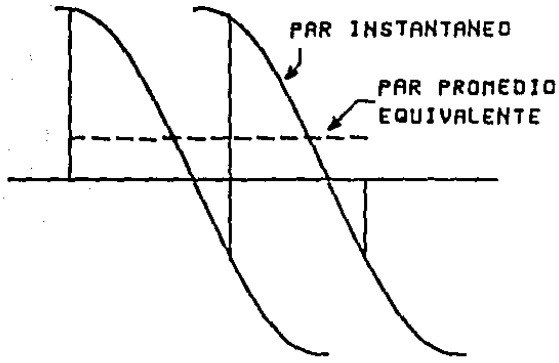


FIG. 6.1 EQUIVALENCIA PARA PROMEDIO-PAR INSTANTANEO

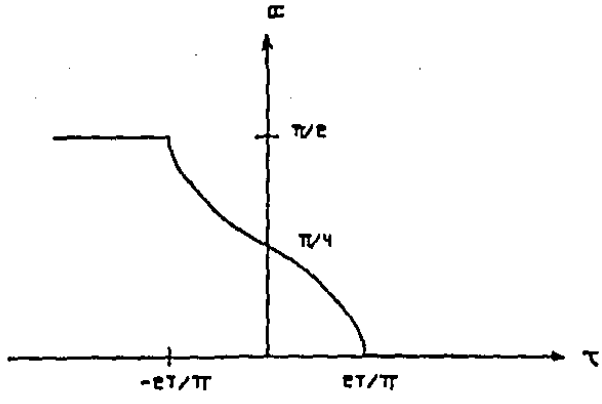


FIG. 6.2 PAR REQUERIDO VS. ANGULO DE CONMUTACION

la máxima que puede proporcionar el motor, es decir:

$$\frac{1}{2} J \dot{\theta}_1^2 = K \quad T \tag{6.3}$$

si a partir de dicho punto de equilibrio no se realiza ninguna conmutación, y despreciando los efectos por pérdidas viscosas, al final de la parte negativa de la curva par-posición se tendría como energía cinética:

$$\frac{1}{2} J \dot{\theta}_1^2 = (K-1) \; T \tag{6.4}$$

el cociente entre (6.4) y (6.3) es:

$$\frac{\dot{\theta}_2}{\dot{\theta}_1} = \frac{K-1}{K} \tag{6.5}$$

si se desea que la velocidad no cambie más del 10% en el intervalo en cuestión se debe cumplir que:

$$(K-1) \; / \; K \; = 0.9^2 \quad => \; K = 5.26 \tag{6.6}$$

De (6.5) en (6.3) y despejando para $\dot{\theta}_1$:

$$\dot{\theta}_1 \geq \frac{2KT}{J} \tag{6.7}$$

Se puede presentar un segundo análisis si se considera cual es la variación de velocidad en la que se incurre si se desea mover el motor a una velocidad constante dada. Para el análisis se despreciarán de nueva cuenta las pérdidas por efectos viscosos.

La energía cinética para una velocidad $\dot{\theta}_1$ está dada por (6.2), si se conmuta cuando el ángulo ha aumentado $\pi /4$ radianes, la energía ha disminuido en una cantidad dada por:

$$\Delta V = 2T \; (1 - \frac{1}{\sqrt{2}}) \tag{6.8}$$

que se obtiene de la ecuación (4.13) desde un punto de equilibrio hasta $\pi/4$ radianes adelante.

En esta condición la velocidad cumplirá con:

$$\frac{1}{2} J \dot{\theta}_2 = T (K - 2 + \sqrt{2}) \tag{6.9}$$

el cociente entre (6.9) y (6.8) es:

$$\frac{\dot{\theta}_2^2}{\dot{\theta}_1^2} = \frac{K - 2 + \sqrt{2}}{K} \tag{6.10}$$

si se desea que la velocidad no varía más del 10% de nueva cuenta se tiene que K debe cumplir con:

$$(K-2 + \sqrt{2})/K = 0.81 \Rightarrow K = 3.06 \tag{6.11}$$

De (6.11) se puede derivar que la condición para que el par instantáneo se pueda aproximar por el par promedio es que la velocidad cumpla con:

$$\dot{\theta}_1 \geq \sqrt{\frac{3T}{J}} \tag{6.12}$$

CALCULO DEL PAR PROMEDIO

Una vez que se propuso una condición para que el par instantáneo se pueda aproximar por el par promedio, en necesario resolver el problema inverso, es decir, como calcular la posición angular de conmutación, o ángulo de conmutación, simplemente, que correponde a un par predeterminado.

Para conseguirlo considérese que un tiempo $t\theta_c$ que corresponde a una cierta posición angular . Llámese $t\theta_c+\pi/2$ al tiempo que correspondería a una posición un paso adelante. Supóngase además que se conoce el par promedio T que se desea aplicar.

El ángulo de conmutación α es un ángulo que cumple con:

$$\theta_c \leq \alpha \leq \theta_c + \pi/2 \tag{6.13}$$

y con:

$$\tau = \int_{t_\theta}^{t_{\theta c+\pi/2}} T(\sigma)d\sigma \qquad (6.14)$$

Según la equivalencia encontrada (6.14) se puede aproximar por:

$$T = \int_{\alpha-\pi/2}^{\alpha} T \operatorname{sen}\theta \, d\theta \qquad (6.15)$$

cuya solución está dada por:

$$\tau = \frac{2}{\pi} T \left[\cos \alpha - \cos (\alpha - \pi/2)\right] \qquad (6.16)$$

que conduce a:

$$\alpha = \pi/4 - \operatorname{sen}^{-1} \left[\frac{\tau\pi}{2T}\right] \qquad (6.17)$$

La ecuación (6.17) determina el ángulo de conmutación siempre que se cumpla que:

$$|\tau| < \frac{2\pi}{\pi} \qquad (6.18)$$

cuando la desigualdad (6.18) no se cumple el ángulo de conmutación está dado por:

$$\alpha = 0 \; ; \; \tau > 2T/\pi \qquad (6.19)$$

y

$$\alpha = \pi/2 \; \tau < -2T/\pi \qquad (6.20)$$

La Fig. 6.2 muestra el ángulo de conmutación vs. el par requerido.

LEY DE CONTROL

Una vez que se han presentado los resultados de los dos incisos anteriores, resta por derivar una ley de control que permita que un motor de pasos describa un movimiento arbitrario predeterminado de antemano.

Sean $\ddot{\theta}_D(t)$, $\dot{\theta}_D(t)$ y $\theta_D(t)$ las funciones que describen la la aceleración, velocidad y posición angular deseadas en la flecha del motor de pasos. Consíderese que el comportamiento del motor está descrito por la ecuación:

$$\ddot{\theta} + f(\dot{\theta}, \theta) = \tau \tag{6.21}$$

A partir de la trayectoria deseada y del modelo del motor dado por (6.21) se desea el problema de calcular el par τ_D que se debe aplicar al motor de pasos para que describa dicha trayectoria. Este problema se resuelve como un problema inverso de cinética. Para conseguirlo se propone que el par deseado satisfaga:

$$\tau_D = \ddot{\theta}_D + g_D (\dot{\theta}_D - \dot{\theta}, \theta_D - \theta) + f(\dot{\theta}, \theta) \tag{6.22}$$

donde el segundo término del lado derecho de (6.22) se interpreta como una corrección el valor del par aplicado según el error de seguimiento.

Si se considera que el par deseado y el requerido coinciden y de la diferencia de (6.22) y (6.21) se llega a:

$$\ddot{\theta}_D - \ddot{\theta} - g_N (\dot{\theta}_N - \dot{\theta}, \theta_D - \theta) = 0 \tag{6.23}$$

o bien si se define la función de error:

$$\varepsilon(t) = \theta_D(t) - \theta(t) \tag{6.24}$$

se tiene que el error satisface la ecuación:

$$\ddot{\varepsilon} - g_D(\dot{\varepsilon}, \varepsilon) = 0 \qquad (6.25)$$

Existen muchas posibilidades para elegir la función g_D. Para el desarrollo del trabajo se decidió elegirla como una función lineal del error de posición y de velocidad, esto es:

$$g_D(\dot{\varepsilon}, \varepsilon) = \alpha_1 \dot{\varepsilon} + \alpha_2 \varepsilon \qquad (6.26)$$

La elección tomada implica la resolución de un problema de asignación de polos para la función de error. De esta manera los valores de α_1 y α_2 se eligen para conseguir que la ecuación:

$$\ddot{\varepsilon} + \alpha_1 \dot{\varepsilon} + \alpha_2 \varepsilon = 0 \qquad (6.27)$$

tenga una solución asintóticamente estable.

Lo anterior se resume en la expresión:

$$\tau_D = \ddot{\theta}_D + \alpha_1[\dot{\theta}_D - \dot{\theta}] + \alpha_2(\theta_D - \theta) + f(\dot{\theta}, \theta) \qquad (6.28)$$

que permite calcular el par deseado.

La forma en que se propone calcular el par deseado para el motor de pasos difiere de las que se proponen en otras técnicas en que se incorpora la posible diferencia entre el comportamiento deseado y el que efectivamente puede describir el motor de pasos.

CALCULO DEL TIEMPO DE CONMUTACION

Para que la técnica que se describe en los párrafos anteriores sea completa, falta relacionar los ángulos de conmutación con los tiempos en que deben ocurrir. Para conseguir esto último se debe calcular el par deseado en cada momento, asociar dicho par con el par promedio que debe proporcionar el motor para el siguiente paso y finalmente encontrar el tiempo que corresponde al ángulo de conmutación determinado. Este último paso implica integrar la ecuación diferencial que describe el comportamiento del motor de pasos (6.21). Por la naturaleza no lineal este modelo se debe recurrir a técnicas numéricas para su solución.

El algoritmo completo de solución es el que sigue:

i) Se considera que el motor parte del reposo.

ii) Se calcula el par deseado según la ecuación (6.28)

iii) Se calcula el ángulo de conmutación que corresponde con el par deseado calculado en (ii).

iv) Se integra la ecuación (6.21) hasta que se alcanza el ángulo de conmutación que se calculó en (iii).

v) Si pasado un cierto número de pasos de integración no se ha llegado al ángulo de conmutación se regresa a (ii).

vi) Cuando se cumple (iv) se iguala el tiempo corriente en la integración con el tiempo de conmutación.

vii) Si no se ha terminado con la trayectoria deseada se regresa a (ii).

7. DESCRIPCION DEL MODELO USADO PARA LAS SIMULACIONES

En este capítulo se describen los programas empleados para simular el comportamiento de los motores de pasos. El objeto de este proceso de simulación es generar las tablas de tiempos de conmutación y verificar el efecto que estos tiempos de conmutación producen sobre un sistema motor-carga que tenga un juego de parámetros diferente al que se usó para generar la tabla de tiempos mencionada.

El capítulo se divide en tres incisos, el primero describe la estructura y funcionamiento del programa que genera la tabla de tiempos de conmutación. El segundo se avoca a describir la forma en que se simula el efecto de dicha tabla. Finalmente se incluye una descripción de un programa auxiliar a las simulaciones que permite generar los patrones de movimiento que se desea que siga el motor de pasos y de otro para modificar los parámetros de las simulaciones.

PROGRAMA DE GENERACION DE TIEMPOS DE CONMUTACION

Este programa realiza las siguientes tareas:

i) Calcular las condiciones deseadas de movimiento
ii) Calcular los parámetros variables
iii) Calcular el par requerido

iv) Integrar numéricamente las ecuaciones de movimiento
v) Verificar el ángulo de conmutación
vi) Elaborar la tabla de tiempos de conmutación

que se describen brevemente a continuación.

Condiciones deseadas de movimiento

En esta parte del programa se calculan los movimientos deseados en el motor de pasos. Para hacerlo se supone que la aceleración del motor es de la forma:

$$\ddot{\theta}_D = C_1 + C_2 t + C_3 \text{ sen } (C_4 t) \tag{7.1}$$

la velocidad se encuentran integrando (6.1) y proporcionando las condiciones iniciales para la misma, es decir:

$$\dot{\theta}_D = \omega_o + C_1 t + C_2 \frac{t^2}{2} - \frac{C_3}{C_4} (C_4 t) \tag{7.2}$$

Finalmente la posición corresponde a la integral de (7.2) más la posición inicial:

$$\theta_D = \theta_o + \omega_o t + C_1 \frac{t^2}{2} + C_2 \frac{t^3}{6} - \frac{C_3}{C_4^2} \text{ sen } (C_4 t) \tag{7.3}$$

Las constantes C_1, C_2, C_3 y C_4 de (7.1) sólo son válidas para un intervalo de tiempo arbitrario. Lo anterior significa que dado un conjunto de tiempos iniciales para dichos intervalos, la posición, velocidad y aceleración deseadas se pueden calcular si se conocen las constantes para cada intervalo.

Los patrones de aceleración que se pueden imponer a través de (7.1) son de una variedad mucho más extensa que la que se considera en la literatura. Es posible, además, aproximar un patrón arbitrario de aceleración por interpolación, si se elige un número adecuado de intervalos y se ajustan apropiadamente los valores de las constantes para cada uno de ellos.

Parámetros del sistema de cargas

La mayoría de los parámetros del sistema motor-cargas son invariantes durante la simulación de su movimiento, sin embargo, el efecto de la masa y la magnitud del par máximo disponible pueden variar con la posición y velocidad angular, respectivamente.

El efecto de la masa se calcula de:

$$J\ddot{\theta} = - B\dot{\theta} - T - (m_1 gr_1 + m_2 gr_2) \text{ sen } \theta \qquad (7.4)$$

donde J está dada por:

$$J = \frac{m_o r_o^2}{2} + \frac{m_1 r_1^2}{3} + m_2 r_2^2 \qquad (7.5)$$

que corresponde a suponer que la inercia se integra con un disco, un brazo y una masa concentrada al final del mismo (ver Fig. 7.1).

El par máximo disponible decrece al aumentar la velocidad de movimiento. Este efecto es de esperarse debido al amortiguamiento viscoso del sistema. A partir de las curvas que proporcionan los fabricantes de motores de pasos para la combinación acoplamiento de potencia-motor se puede encontrar por interpolación lineal una constante de decrecimiento de dicho par de tal forma que el par disponible obedezca a:

$$\begin{aligned} T &= T_{max} - K_v |\dot{\theta}| \; ; \; T_{max} > K |\dot{\theta}| \\ T &= 0 \; ; \; T_{max} < |\dot{\theta}| \end{aligned} \qquad (7.6)$$

donde Tmax=par de retención.

Par requerido

El par requerido se calcula según la ley de control que se derivó en el capítulo anterior (ver ecuación 6.28), a la que se añadió el efecto sobre el par del brazo y la masa concentrada al final del mismo.

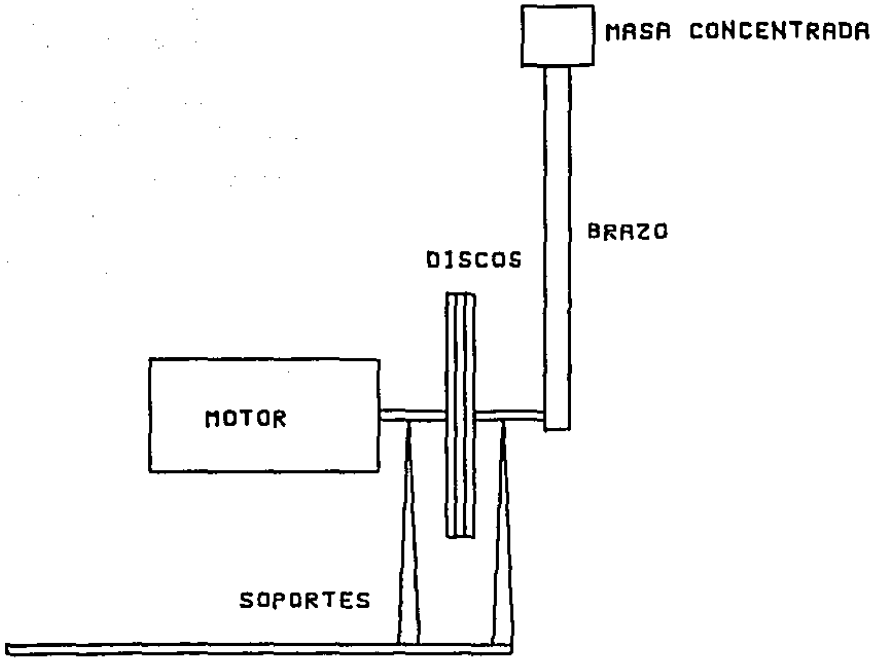


FIG. 7.1 ARREGLO DE CARGAS DEL SISTEMA

Integración de las ecuaciones de movimiento

La integración de las ecuaciones de movimiento se realizó con un método Runge-Kutta de segundo orden (Refs. 18 y 19).El paso de integración puede variar cada vez que se cambian las constantes C_1 a C_4 que determinan los patrones de aceleración.

Para seleccionar el tamaño del incremento de integración se usó como criterio que, a la máxima velocidad de operación prevista para el intervalo de validez de las constantes mencionadas, fuera posible integrar las ecuaciones al menos diez veces para cada paso del motor. Cuando la velocidad prevista estaba por abajo de un umbral (1000 pasos/segundo) se tomaba como paso de integración el correspondiente a dicho umbral.

Este criterio se validó realizando pruebas con pasos de integración menores y observando que los resultados obtenidos no cambiaban de manera significativa.

Con el fin de reducir el tiempo de cómputo, las condiciones deseadas de movimiento no se calculan cada paso de integración, sino que se toma un tiempo múltiplo de este último para hacerlo.

Los resultados de la integración se conservan en un archivo para su posterior análisis. Tampoco en este caso se guardan los resultados de todas las integraciones.

Angulo de conmutación

El ángulo de conmutación se calcula según las ecuaciones (6.18) a (6.20). Normalmente el cálculo se realiza cuando se detecta un cruce por cero en las curvas par-posición angular. Sin embargo, debido a problemas de tipo numérico se incluyeron otras condiciones para calcular dicho ángulo. A continuación se describe el proceso completo:

a) Si el motor se mueve a velocidades mayores que un umbral determinado, se calcula el ángulo de conmutación normalmente.

b) Una vez que se ha calculado el ángulo de conmutación se levanta una

bandera para indicar su cálculo.

c) Si pasado un cierto número de iteraciones no se ha alcanzado el ángulo de conmutación determinado, y la bandera mencionada en (b) está levantada, se pregunta si el movimiento del motor se encuentra en alguna de las siguientes situaciones:

- La velocidad es menor que el umbral mencionado en (a)
- Ha cambiado la dirección del movimiento

d) Si se cumplen las condiciones de (c) y las condiciones deseadas indican que el motor se debería estar moviendo, se recalcula el ángulo de conmutación.

Tiempo de conmutación

El tiempo de conmutación se registra cuando el ángulo de desplazamiento real es mayor o igual,en valor absoluto, que el ángulo de conmutación que se haya calculado.

Cuando en el proceso descrito en el párrafo anterior se pasa varias veces por los puntos (c) y (d), el par requerido aumenta, hasta que se hace igual al máximo par que puede proporcionar el motor. En estos casos el ángulo de conmutación coincidirá con el ángulo corriente y la conmutación ocurrirá instáneamente. Esta situación se presenta para todos los arranques del motor.

SIMULACION DE LAS TRAYECTORIAS

Este programa es una reproducción de las partes de cálculo de condiciones deseadas, parámetros e integración de las ecuaciones de movimiento descritas en el inciso anterior. Su funcionamiento grosso modo es como sigue.

La integración de las ecuaciones de movimiento se realiza con un paso de integración menor o igual que el menor que se utilizó en la generación de la tabla de tiempos de conmutación. Para cada paso de integración se revisa que el tiempo de simulación no sea mayor que el tiempo de la siguiente conmutación por realizar. En caso contrario se realiza la conmutación. Los resultados de la simulación se guardan con el mismo

criterio que en el programa anterior.

Cuando la simulación se realiza para verificar la robustez de la secuencia de tiempos de conmutación para un juego de parámetros diferente del original, los parámetros correspondientes deben cambiarse en el archivo de datos.

PROGRAMAS AUXILIARES

Existe un programa auxiliar que permite generar las tablas para las condiciones deseadas de movimiento. Este programa pregunta por los valores de las constantes de aceleración, el tiempo a partir de cual son válida y el el paso de integración que se usará durante el intervalo de validez de dichas constantes.

Existe otro programa para modificar el archivo de parámetros de la simulación que al ejecutarse permite cambiar con facilidad cualquier parámetro.

Todos los programas descritos en este capítulo se codificaron en Pascal (Ref. 20) y se implantaron en una microcomputadora Apple II (Refs. 21 y 22). Los textos se incluyen en el anexo A.

8. BANCO DE PRUEBAS.

En este capítulo se describe el banco de pruebas para motores de pasos diseñado y construído para validar experimentalmente los resultados obtenidos. El banco se controla desde una microcomputadora y permite medir la respuesta dinámica de motores sujetos a distintas técnicas de control. El funcionamiento del banco se realiza a través de un conjunto de programas que ejecutan las siguientes tareas:

1) Envío de pulsos a los motores de pasos según los tiempos de conmutación.
2) Medición de posición y velocidad en movimientos de pequeña o gran magnitud.
3) Pruebas de los dispositivos de medición.
4) Presentación gráfica de resultados.

El banco se desarrolló con base en tres subsistemas: mecánico, de control y de medición. En la Fig. 8.1 se muestra un diagrama de bloques de la instalación, los diferentes subsistemas están integrados como sigue:

Subsistema mecánico.- lo integran aquellos bloques que están en movimiento, es decir, el rotor del motor, el juego de cargas y el detector de posición.

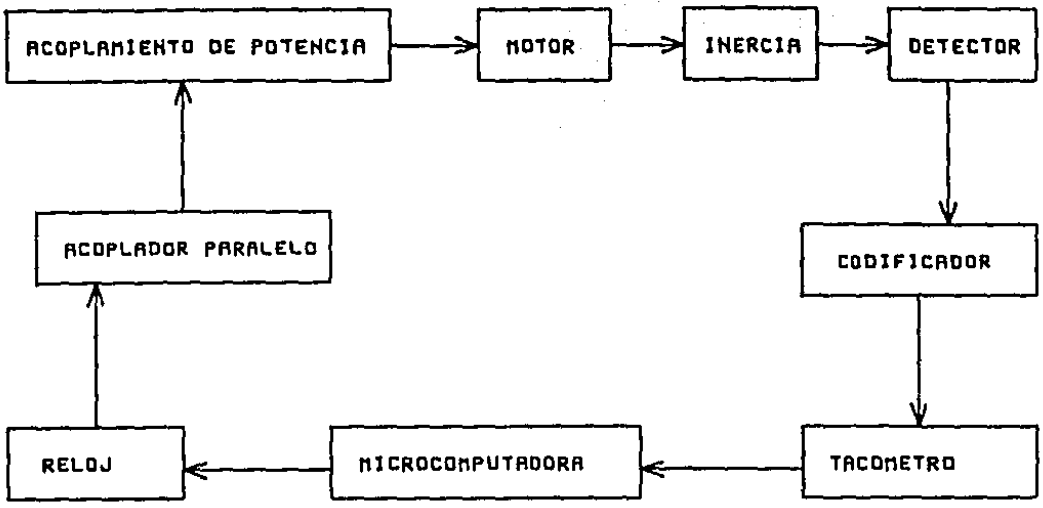


FIG. 8.1 DIAGRAMA DE BLOQUES DEL BANCO DE PRUEBAS

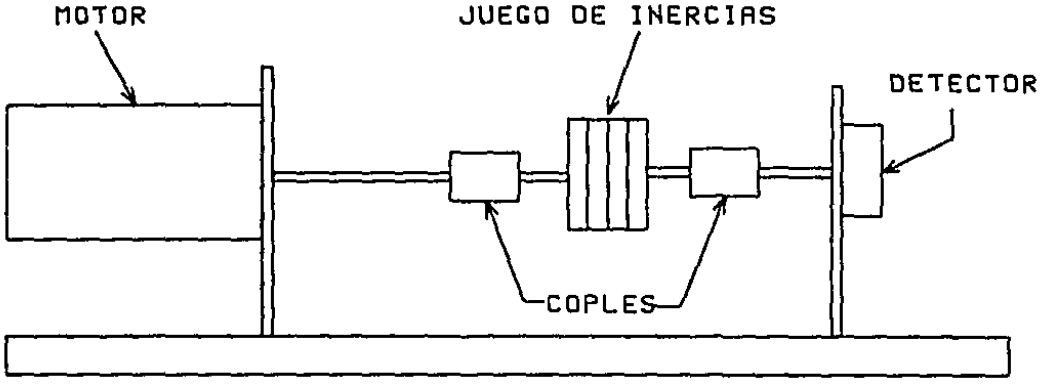


FIG. 8.2 ARREGLO DEL SUBSISTEMA MECANICO

Subsistema de control.- lo forman todos aquellos componentes dirigidos a producir las señales eléctricas que permiten el movimiento del motor. En este caso lo integran un reloj, un acoplador paralelo y el controlador del motor de pasos.

Subsistema de medición.- es áquel que permite obtener información acerce del movimiento del motor. Esta constituido por un codificador y un tacómetro digital.

Los subsistemas de control y medición se manejan a través de un microcomputador Apple II en la que se implantaron los programas necesarios para manejarlos. En los siguientes incisos se describe con mayor amplitud cada subsistema, así como también los distintos programas que ejecutan los experimentos con los motores de pasos.

SUBSISTEMA MECANICO

El subsistema mecánico se coloca sobre un banco. Sus diferentes componentes se unen por medio de coples. El banco puede soportar pruebas de motores que proporcionen pares menores que 19 Nm.

El juego de cargas consiste de un conjunto de discos de acero y aluminio colocados sobre una flecha de fácil desmonte que varía la inercia que debe mover el motor. Es posible también acoplar la flecha del motor con un reductor de velocidad y colocar a la salida de éste discos de acero que cumplan la función descrita. En el anexo D se encuentran las especificaciones de las componentes mecánicas del banco de pruebas.

La salida del sistema de cargas se acopla con un detector de posición relativa de tipo optoelectrónico. Se utilizó el modelo HEDS-5000 de Hewlett-Packard. Este detector entrega quinientos pulsos eléctricos por revolución del motor a través de dos canales defasados 90^{o} eléctricos. En la Fig. 8.2 se muestra un diagrama del arreglo del subsistema mecánico.

SUBSISTEMA DE CONTROL

Como se mencionó, este subsistema está integrado por el reloj, el acoplador paralelo y el acoplamiento de potencia para los motores de pasos. A continuación se describe brevemente cada componente.

Reloj

El reloj que genera señales según los tiempos en que deban enviarse pulsos a los motores de pasos se construyó con base en la componente MC6840 de Motorola (Ref. 23) que tiene tres contadores de 16 bits cuyo valor se decrementa con una señal externa de sincronía. Se utilizó un arreglo en cascada de dos de estos contadores. El primero se alimenta con el reloj de sincronía de la microcomputadora (que posee un periodo aproximado de 1 microsegundo) y produce señales de sincronía con periodos de uno, diez, cien y mil microsegundos que alimentan al segundo contador. De esta forma se pueden lograr señales de tiempo separadas por periodos desde 1 microsegundo hasta 65 segundos, aproximadamente. En la Tabla 8.1 se muestran los rangos de las señales producidas incluyendo su precisión.

La señal del reloj produce una interrupción al microprocesador que al detectarla ejecutará una rutina para enviar por el acoplador en paralelo el pulso eléctrico correspondiente.

Acoplador paralelo

El acoplador paralelo se implantó con base en un alimentador de 2 bits que mantiene estable el valor de la salida que se controla con operaciones de escritura del microprocesador. Se utilizaron como biestables dos de los anunciadores que provee la Apple II (Ref. 24).

Tanto el reloj como el acoplador paralelo están colocados en las ranuras para interfase de la microcomputadora (Ref. 24).

Acoplamiento de potencia

El acoplamiento de potencia que se usó para el motor de pasos es comercial. Se utilizó el modelo TBM-105 de Superior Electric Co. Es de tipo voltaje dual y permite mover los motores bajo prueba hasta 5000 pasos por segundo. El acoplamiento impone como limitación el empleo de motores de pasos de tipo imán permanente con dos fases bipartidas, que es el tipo más usado en la actualidad. Es posible sustituir este acoplamiento por cualquier otro que requiera para su manejo de pulsos electrícos TTL. En la Ref. 6 se describen con amplitud los diversos tipos de acoplamientos existentes.

DESDE	HASTA	UNIDADES	ERROR
1	65 536	MICROSEGUNDOS	+/- 1
10	655 360	MICROSEGUNDOS	+/- 10
100	6 553 600	MICROSEGUNDOS	+/- 100
1	65 536	MILISEGUNDOS	+/- 1

TABLA 8.1 PRECISION DE LOS PERIODOS PARA ENVIO DE PULSOS A LOS MOTORES DE PASOS

B \ A	0	1
0->1	I	D
1->0	D	I

A \ B	0	1
0->1	D	I
1->0	I	D

A Y B SENALES DEL CODIFICADOR

I - IZQUIERDA

D - DERECHA

TABLA 8.2 INTERPRETACION DE LAS SENALES DEL CODIFICADOR

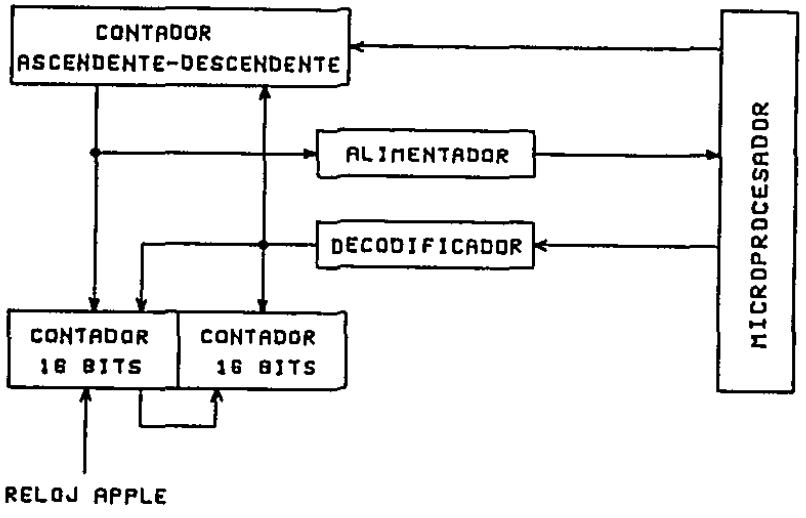


FIG. 8.3 DIAGRAMA DE BLOQUES DEL TACOMETRO DIGITAL

En el anexo C se encuentran los diagramas de conexiones que corresponden a los componentes de este subsistema y las especificaciones más importantes del acoplamiento de potencia.

SUBSISTEMA DE MEDICION

Está integrado por el codificador y el tacómetro digital, la descripción grosso modo de los mismos se presenta a continuación.

Codificador

El codificador recibe los pulsos eléctricos de los dos canales del detector de posición. A partir de estas señales genera otras dos que indican movimientos a la izquierda o a la derecha del motor también a través de pulsos eléctricos. En la Tabla 8.2 se muestran los diagramas de interpretación de las señales del detector. Como puede notarse su diagrama de transición de estados corresponde a un autómata finito.

La resolución del codificador es de 1/2000 de revolución en ambas direcciones que corresponde a todas las transiciones que se presentan para dos bandas ópticas con 500 marcas cada uno.

En en el anexo C se encuentran las hojas de especificaciones del detector de posición usado y del automáta finito que se construyó para codificar las salidas del primero.

Tacómetro digital

El tacómetro digital es el dispositivo más importante de todos los que se contruyeron. Su función es recibir las señales provenientes del codificador y miden el tiempo que transcurre entre una o varias de ellas. Consta de dos unidades idénticas, por lo que en adelante sólo se describirá una de ellas. La razón para esta duplicación estriba en que bajo ciertas condiciones anormales, el motor se moverá en la dirección contraria a la esperada y no es posible predecir el momento en que ello sucederá.

En la Fig. 8.3 se muestra un diagrama de bloques de una unidad del tacómetro digital que consta de un contador ascendente-descendente de 12 bits, de dos contadores descendentes de 16 bits y de un conjunto de decodificadores, biestables y compuertas tres estados que lo conectan al microprocesador.

Los contadores ascendente-descendentes se forman con tres circuitos

74LS193 en cascada. Las señales de salida del codificador se conectan a las señales cuenta arriba y cuenta abajo de los contadores invirtiendo su papel en cada tacómetro. De esta forma el contenido de los contadores aumenta en un tacómetro mientras que disminuye en el otro. La señal de cuenta mínima se conecta a los contadores de 16 bits mencionados y además produce una interrupción al microprocesador cuando se filtra a través de un biestable.

La medición de tiempo se realiza en una componente INTEL 8253 (Ref. 25). Nuevamente se utilizan dos contadores de 16 bits en cascada. El primero de ellos se alimenta también con el reloj de sincronía de la Apple II y su salida sirve de sincronía al segundo.

El resto de los elementos del tacómetro permite que el microprocesador accese a los contadores o relojes, o bien interrogue para conocer cual tacómetro produjo la señal de interrupción.

El funcionamiento del tacómetro es como sigue: se precargan los contadores ascendente-descendentes con el número de eventos que se desea medir. Cada evento correponde a 1/2000 de revolución de detector de posición (que corresponde a 1/10 de paso para los motores que se probaron). Acto seguido se desinhibe el funcionamiento de los contadores de 16 bits. Cuando se presenta una señal de cuenta mínima, ésta inhibe a los contadores que miden el tiempo y avisa al microprocesador. De esta forma el tiempo medido en los contadores de tiempo es el que transcurrió para que se midiesen la cantidad de eventos deseada, sin tomar en cuenta las oscilaciones de pequeña amplitud que es posible se presenten en el movimiento de los motores de pasos y que se filtran a través de los contadores ascendente-descendentes.

Para las pruebas del conjunto codificador-tacómetro se construyó un emulador electrónico del detector de posición con el que fue posible verificar el funcionamiento.

De nueva cuenta en el anexo C se encuentra la descripción del tacómetro digital y el emulador del detector de posición.

PROGRAMAS PARA CONTROLAR EL BANCO DE PRUEBAS.

Los programas para el manejo del banco de pruebas se implantaron en el

sistema operativo de Pascal de la Apple II y se codificaron en lenguajes Pascal y ensamblador. A continuación se encuentra su descripción.

Programa para medición de movimientos grandes

A partir de la lista de tiempos que generó el programa de simulación del movimiento que se describió en el capítulo anterior, este programa controla simultáneamente los subsistemas de control y medición. Para el primero, carga el primer y segundo contador de tal forma que produzcan una señal de interrupción al microprocesador en el momento de envío de un pulso a los motores de pasos. En el segundo subsistema el programa precarga los contadores ascendente-descendente según la magnitud de las muestras que se desea tomar, arranca los relojes al recibir una señal de inicio de experimento y coloca en zonas conocidas de memoria el resultado de las mediciones.

El programa itera sobre la lista de tiempo hasta agotarlo o bien concluye cuando ha tomado un número de muestras predeterminado. Adicionalmente, verifica una señal que indica paro forzoso del experimento. Al concluir las mediciones avisa sobre la forma en que concluyó el experimento y almacena en disco los resultados.

Programa para medición de eventos pequeños

Este programa se realizó para medir la respuesta del motor en un sólo paso con el fin de determinar la validez de las suposiciones efectuadas en el modelo matemático. Se envía un paso al motor y se mide el tiempo de cualquier evento que exceda la resolución del detector en ambas direcciones. Los resultados se almacenan también en disco.

Programa de pruebas

Está dirigido a comprobar el funcionamiento del reloj y del conjunto codificador-tacómetro. Para el primero, lo obliga a producir un tren de pulsos de frecuencia conocida que puede ser analizado en un osciloscopio o en un frecuencímetro de alta resolución.

El codificador y tacómetro se prueban de dos maneras. En la primera se emplea el emulador del detector de posición y se lo hace funcionar a una velocidad conocida. La segunda prueba consiste en alimentar el motor de pasos con un generador de pulsos a una frecuencia predeterinada. En ambos casos se toman mediciones y se contrastan los resultados obtenidos

contra las velocidades de las fuentes que los produjeron.

La precisión obtenida para todos los casos fué menor que 0.1%.

Programa para graficación de resultados

Este programa tiene por objeto elaborar gráficas con base en los archivos en disco que se produjeron en los programas anteriores. Se utiliza el sistema para gráficas de alta resolución de la Apple II, sobre el que se implantó una unidad (conjunto de programas) para facilitar el manejo de la graficación y reducir en todos los casos la producción de una gráfica al llamado de un procedimiento. Por otro lado se producen copias en papel de estas gráficas en una impresora ATI II. En el capítulo 9 se muestran ejemplos de los resultados obtenidos.

En el anexo B de este escrito se encuentran los listados de todos los programas para computadora digital que aquí se describieron.

9. RESULTADOS

En este capítulo se presentan los resultados obtenidos al aplicar la técnica descrita en los capítulos 5 y 6. Se muestran siete ejemplos significativos de los experimentos realizados. Los datos para cada experimentos se encuentran en el anexo D.

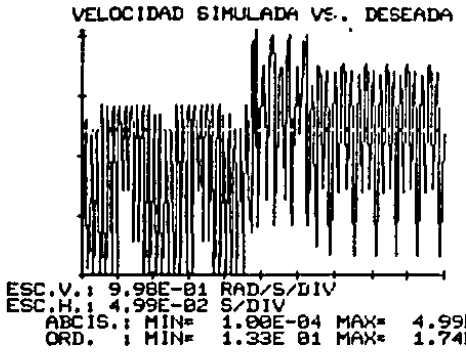
En todos los casos se presentan las siguientes gráficas:

- posición real y posición deseada vs. tiempo
- velocidad real y velocidad deseada vs. tiempo
- posición simulada y posición deseada vs. tiempo
- velocidad simulada y velocidad real vs. tiempo

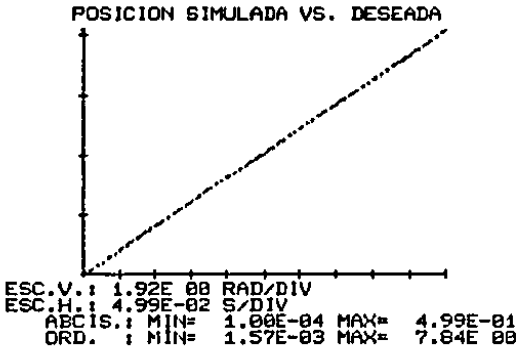
Las posiciones y velocidades reales corresponden a las que se midieron en el banco de pruebas, mientras que las simuladas se obtuvieron del programa de simulación descrito en el capítulo 7. Las líneas punteadas en las gráficas corresponden siempre a las condiciones deseadas.

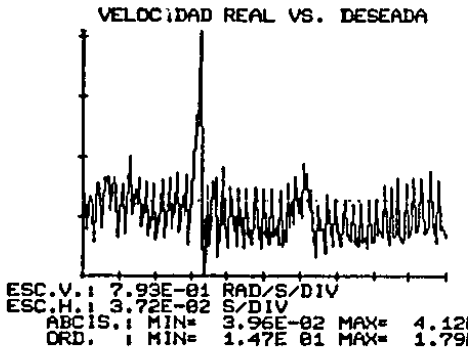
Experimento 1: movimiento a velocidad constante

El objetivo del experimento era mover el motor de pasos a una velocidad de 1500 pasos/segundo durante medio segundo. La Fig. 9.1 muestra los resultados obtenidos. El desempeño del motor es correcto y los hechos más notables son: las oscilaciones de velocidad reales son menores que



(a)





(c)

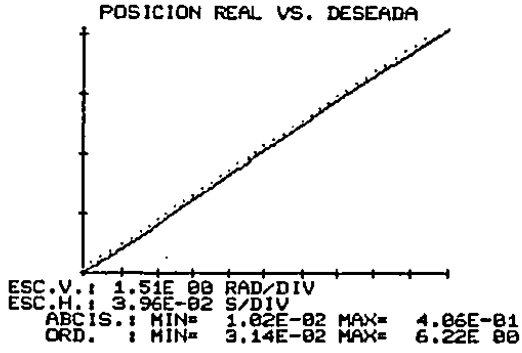


Fig. 9.1 Resultados del Experimento 1

a) Velocidad simulada b) Posición simulada
c) Velocidad real d) Posición real

las simuladas y que existe un pequeño error de estado estable en el seguimiento de posición.

Experimento 2: movimiento uniformemente acelerado y a velocidad constante

En este experimento se pretendía acelerar durante 0.1 segundos un motor hasta llevarlo a 3000 pasos/segundo y mantenerlo en esa velocidad durante 0.4 segundos.

La Fig. 9.2 muestra que el resultado es adecuado. En este caso se observa que las oscilaciones de velocidad reales son mayores que las simuladas.

Experimento 3: movimiento uniformemente acelerado, a velocidad constante y uniformemente desacelerado.

En este experimento se parte de tener el motor en reposo, acelerarlo durante 0.1 segundos, mantener su velocidad por 0.3 segundos y finalmente frenarlo en 0.1 segundos.

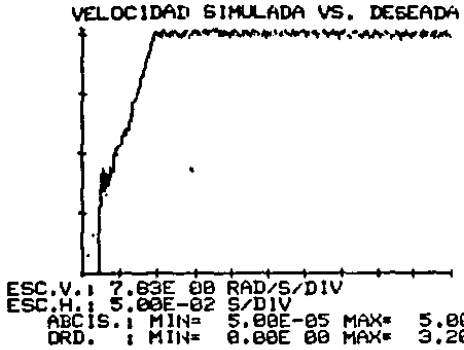
La Fig. 9.3 muestra los resultados del experimento. De nueva cuenta se observa que las oscilaciones de velocidad son mayores en la realidad que durante la simulación.

Experimento 4: movimiento uniformemente acelerado y desacelerado

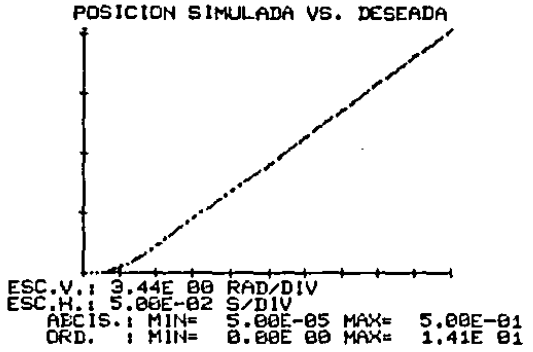
En este caso se aceleró el motor durante 0.15 segundos y se le frenó en el mismo lapso hasta llevarlo al reposo de nueva cuenta. Los resultados se grafican en la Fig. 9.4. El comportamiento simulado y real concuerdan básicamente.

Experimento 5: movimiento uniformemente acelerado, desacelerado y uniformente acelerado

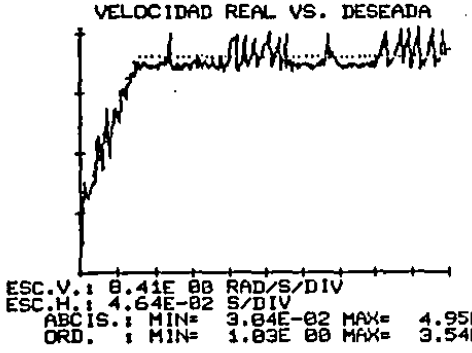
En este caso se aplican aceleraciones positivas y negativas. Se acelera al motor durante 0.15 segundos, se deacelera 0.3 segundos y se acelera de nuevo 0.15 segundos. Las posiciones inicial y final deben coincidir, por lo que debe ocurrir un cambio de dirección a la mitad del movimiento. Los resultados están contenidos en la Fig. 9.5.



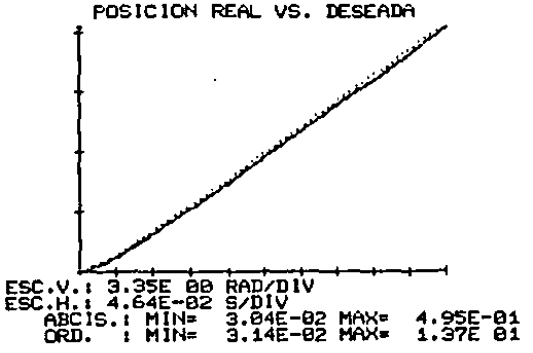
(a)



(b)



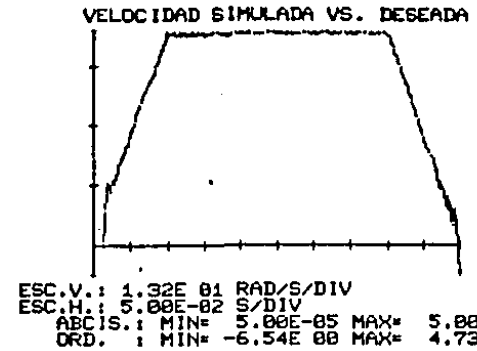
(c)



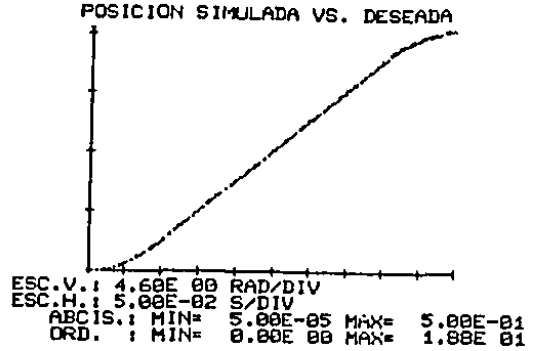
(d)

Fig. 9.2 Resultados del experimento 2

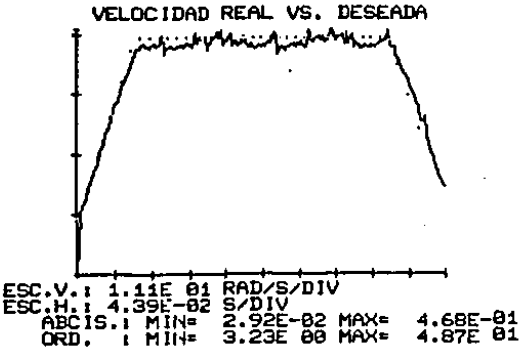
a) Velocidad simulada b) Posición simulada
c) Velocidad real d) posición real



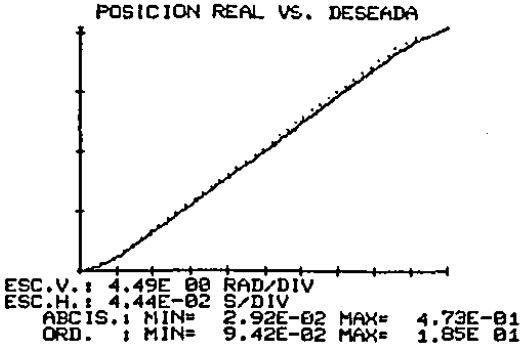
(a)



(b)



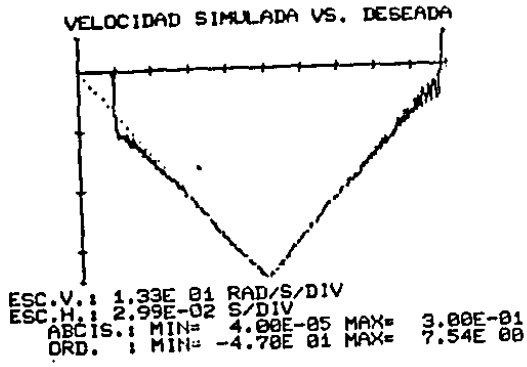
(c)



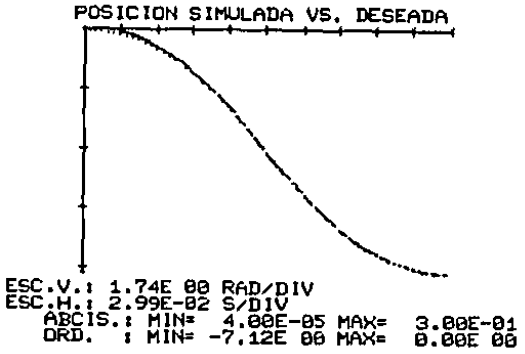
(d)

Fig. 9.3 Resultados del experimento 3

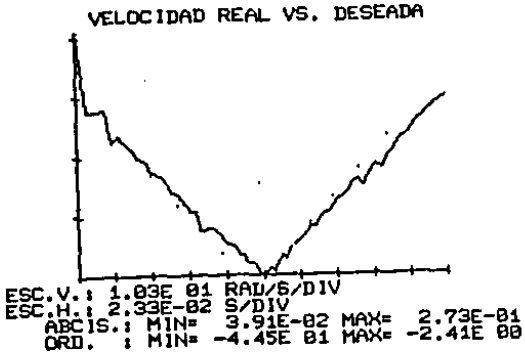
a) Velocidad simulada b) Posición simulada
c) Velocidad real d) Posición real



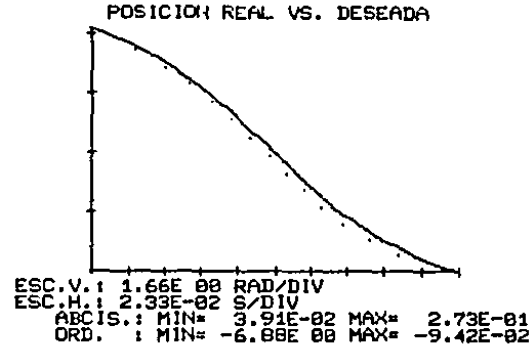
(a)



(b)



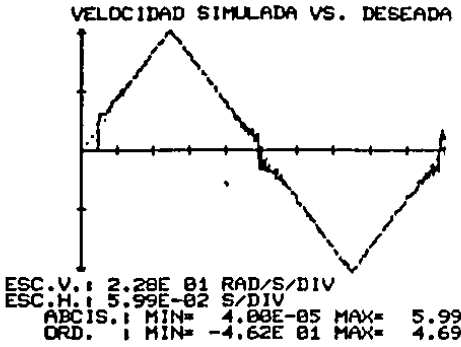
(c)



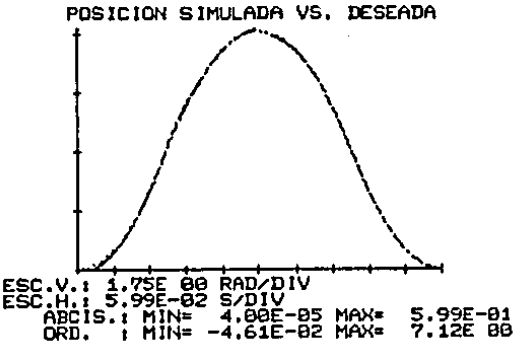
(d)

Fig. 9.4 Resultados del experimento 4

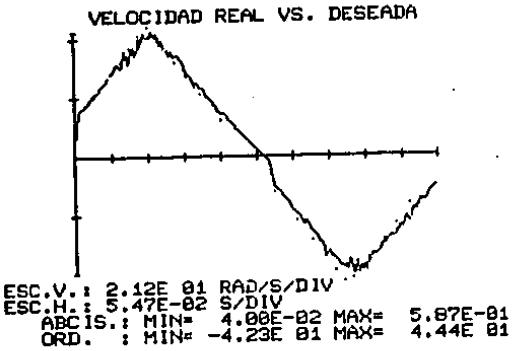
a) Velocidad simulada b) Posición simulada
c) Velocidad real d) Posición real



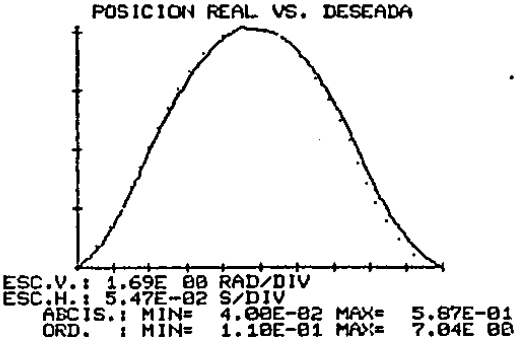
(a)



(b)



(c)



(d)

Fig. 9.5 Resultados del experimento 5

a) Velocidad simulada b) Posición simulada
c) Velocidad real d) Posición real

Experimento 6: movimiento con aceleración lineal

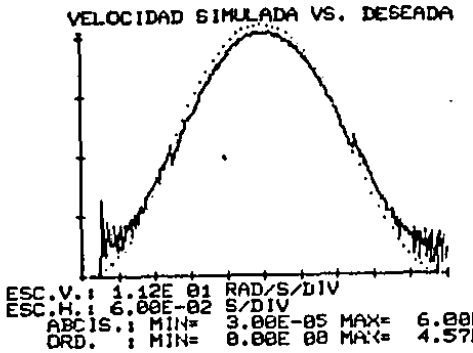
Para este experimento se introdujo una aceleración de forma más compleja. La velocidad debe describir una parábola y la posición una cúbica. Los resultados se muestran en las Fig. 9.6.

Experimento 7: movimiento con aceleración lineal y cambio de inercia

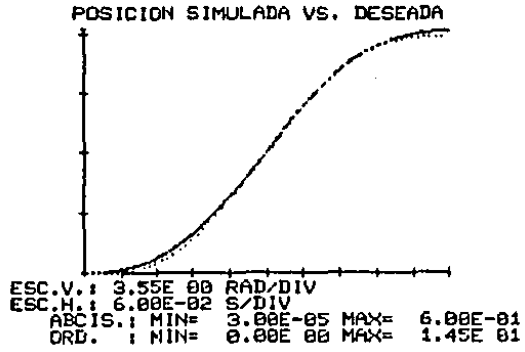
En este caso se prescribió al motor un movimiento igual al que se impuso en el experimento 6, pero se realizaron cambios en la inercia real, en relación con la que se usó para generar la tabla de tiempos. La Fig. 9.7 muestra los resultados.

Experimento 8: respuesta a paso

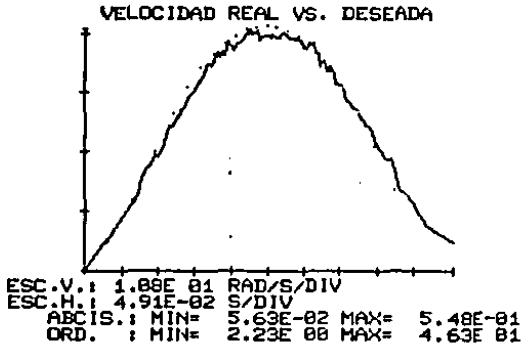
Este experimento tenía como objetivo obtener la curva de respuesta de un motor de pasos a una conmutación simple. Se pretendía verificar si el modelo de segundo orden propuesto para el comportamiento del motor correspondía con el respuesta real. La Fig. 9.8 muestra el resultado obtenido. Puede notarse que la forma de la respuesta coincide con la esperada, aunque se detectaron dos problemas. El primero fue una leve asimetría entre los dos canales de detector de posición empleado, que explica que los puntos se asocien por parejas. La segunda anomalía detectada fue una zona de transición, o de cambio de pendiente, cuya presencia se debe a la influencia del tipo de acoplamiento de potencia sobre la forma de la respuesta. En este caso se usó un acoplador de votaje dual, y la zona de transición corresponde a la desconexión de la fuente de alto voltaje y la conexión de la de bajo.



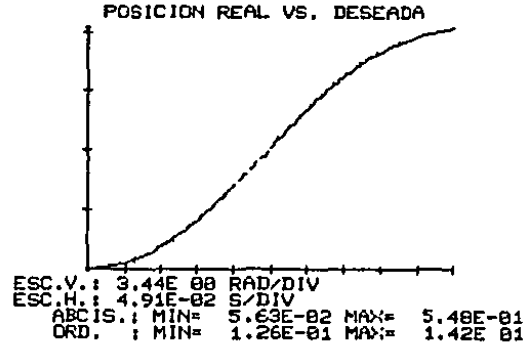
(a)



(b)



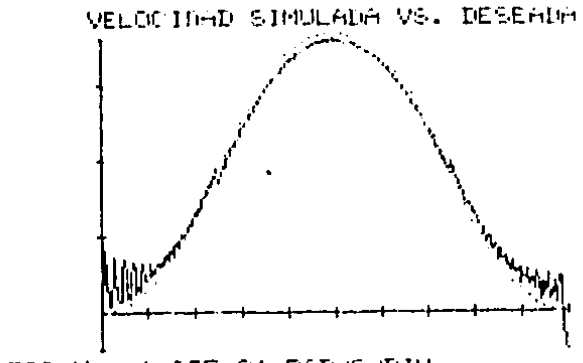
(c)



(d)

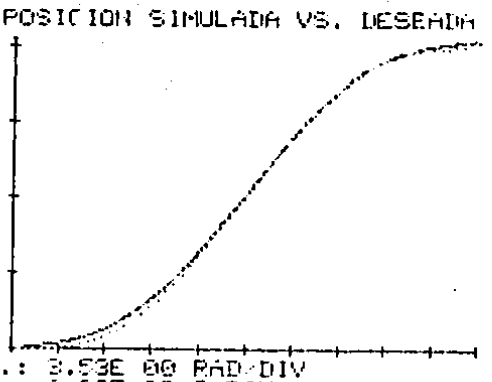
Fig. 9.6 Resultados del experimento 6

a) Velocidad simulada b) Posición simulada
c) Velocidad real d) Posición real



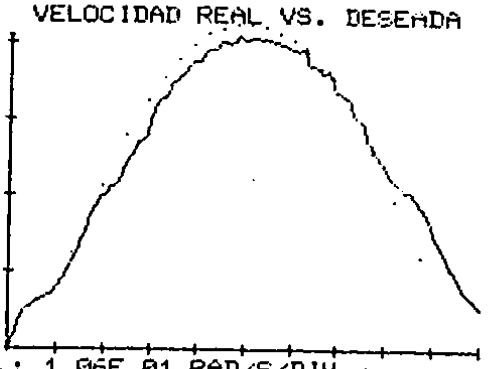
(a)

ESC.V.: 1.28E 01 RAD/S/DIV
ESC.H.: 6.00E-02 S/DIV
ABCIS.: MIN= 3.00E-05 MAX= 6.00E-01
ORD. : MIN= -6.29E 00 MAX= 4.60E 01



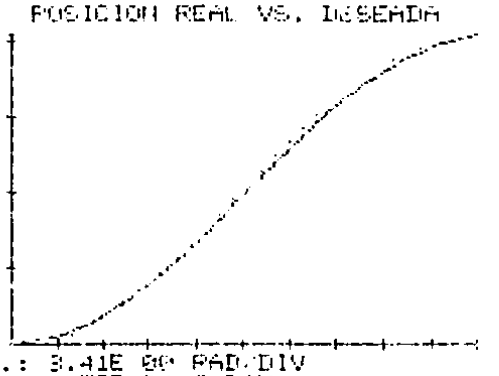
(b)

ESC.V.: 3.53E 00 RAD/DIV
ESC.H.: 6.00E-02 S/DIV
ABCIS.: MIN= 3.00E-05 MAX= 6.00E-01
ORD. : MIN= 0.00E 00 MAX= 1.44E 01



(c)

ESC.V.: 1.06E 01 RAD/S/DIV
ESC.H.: 4.79E-02 S/DIV
ABCIS.: MIN= 5.92E-02 MAX= 5.38E-01
ORD. : MIN= 2.12E 00 MAX= 4.53E 01



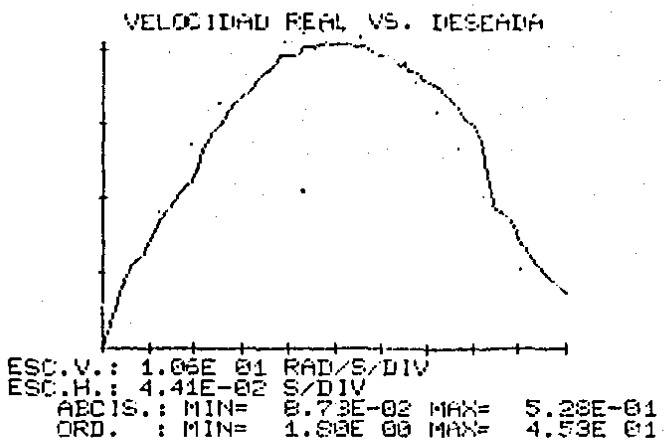
(d)

ESC.V.: 3.41E 00 RAD/DIV
ESC.H.: 4.79E-02 S/DIV
ABCIS.: MIN= 5.92E-02 MAX= 5.38E-01
ORD. : MIN= 1.26E-01 MAX= 1.41E 01

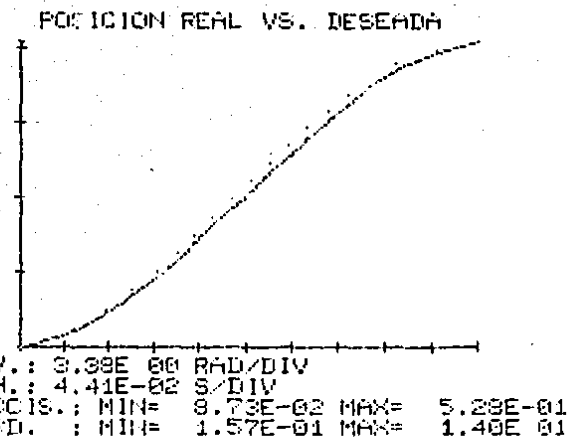
Fig. 9.7 Resultados del experimento 7

a) Velocidad simulada b) Posición simulada

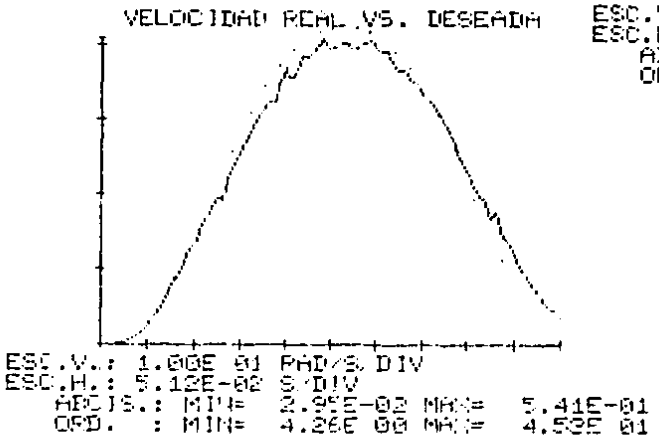
c) Velocidad real d) Posición real



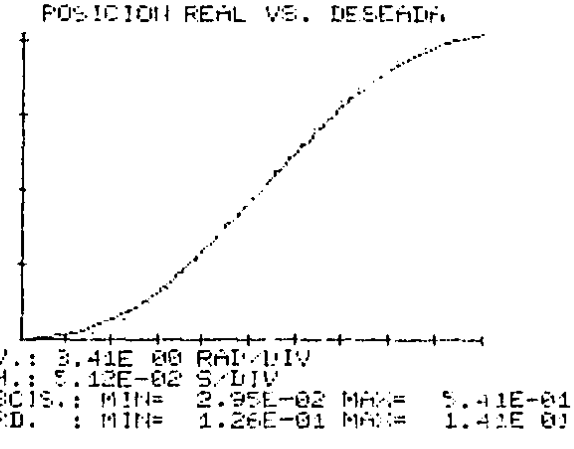
Δj + 16



Δj + 16



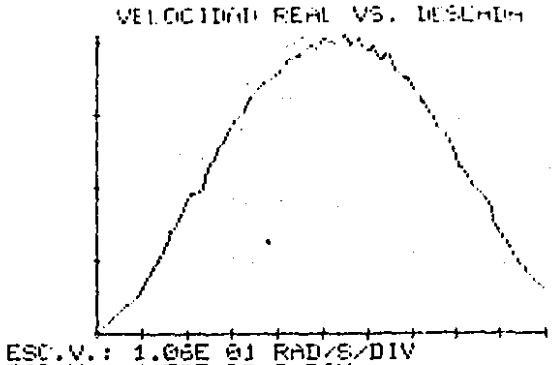
Δj + 31



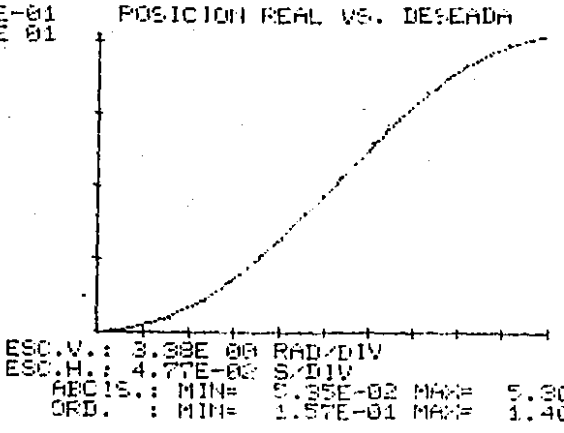
Δj + 31

Fig. 9.7 Resultados del experimento 7 (continuación)

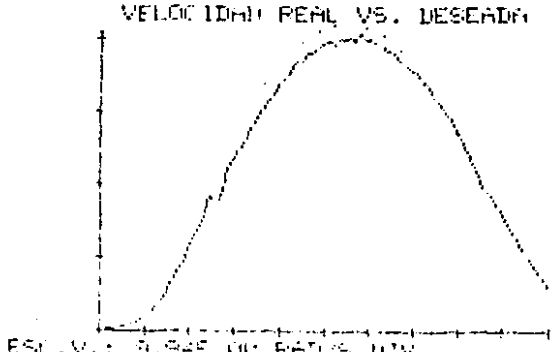
a) Velocidad real con incremento inercia Δj = + 16% b) Posición real con incremento inercia Δj = + 16% c) Velocidad real con incremento inercia Δj = + 31% d) Posición real con incremento inercia Δj = + 31%.



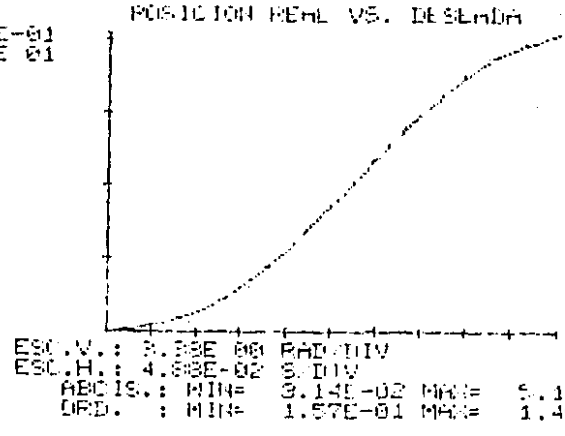
a) $\Delta j = -14$



b) $\Delta j = -14$



c) $\Delta j = -29$



d) $\Delta j = -29$

Fig. 9.7 Resultados experimento 7 (valor final)
a) Velocidad real; decremento de inercia $\Delta j = -14\%$ b) Posición real; decremento de inercia $\Delta j = -14\%$ c) Velocidad real; decremento de inercia $\Delta j = -29\%$ d) Posición real; decremento de inercia $\Delta j = -29\%$.

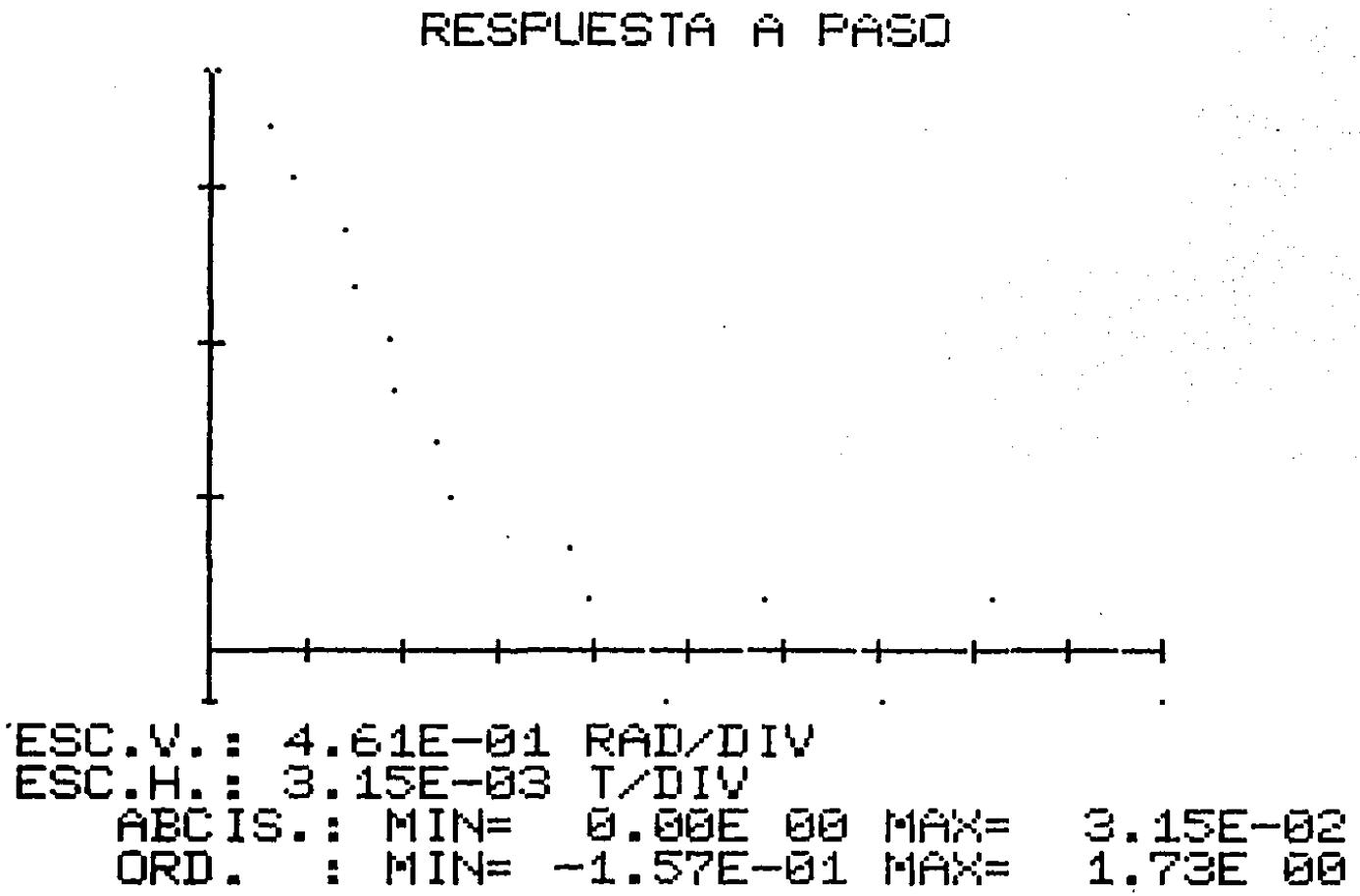


Fig. 9.8 Resultados experimento 8. Respuesta a paso

10. CONCLUSIONES

Este capítulo presenta las conclusiones del trabajo, estas se pueden dividir en las relativas a la evaluación de la técnica de control propuesta y aquellas que se refieren a condiciones generales para el empleo de motores de pasos. Se incluye además recomendaciones sobre líneas de trabajo que conviene desarrollar para un mejor aprovechamiento de los motores de pasos.

Técnica de control propuesta

Del análisis de los resultados mostrados en el capítulo anterior se puede concluir que la técnica propuesta cumple con las expectativas que motivaron el trabajo aquí descrito. Se pudo imponer al motor de pasos patrones de movimiento más complicados y de mayor magnitud que los que se reportan en la literatura. La coincidencia de los resultados reales con los deseados está dentro de márgenes de tolerancia apropiados.

La técnica permite manejar de manera natural el problema del arranque de los motores de pasos, que es una de las limitantes importantes que presentan las demás técnicas que se describieron en el capítulo 4. Estos problemas en el arranque de los motores se deben a la alta aceleración que se obtiene cuando se aplica el primer paso. De cualquier forma, bajo el criterio de control empleado, se obtiene el mejor resultado posible.

La presencia en algunos de los experimentos de errores de estado estable entre las posiciones reales y las deseadas, que resulta definitivamente inapropiada, se puede eliminar si se incluye un término integral en la ley de control, o bien si se da mayor valor a la constante que pesa el error de posición.

Condiciones para el empleo de los motores de pasos

Las conclusiones generales sobre el empleo de los motores de pasos son:

i) Si se deben mover cargas viscosas debe emplearse la técnica descrita en el primer inciso del capítulo 4.

ii) Si se desea mover cargas a velocidad constante, se debe usar el criterio de Venkantaratnam.

iii) Si se quieren mover cargas inerciales con patrones de movimiento sencillos, la Técnica de Leenhouts es recomendable por que se basa en datos fácilmente asequibles.

iv) Si se desea mover cargas inerciales con patrones arbitrarios de movimiento se debe utilizar la técnica descrita en los capítulos 5 y 6 de este trabajo.

v) En todos los casos es conveniente que la relación par-inercia, o coeficiente de amortiguamiento-inercia para cargas viscosas, tenga un valor mayor que la unidad. Entre mayor este valor, mayor la seguridad de completar los movimientos prescritos sin pérdida de sincronía.

Recomendaciones generales

Las técnicas aquí descritas permiten resolver el problema de control de motores de pasos en malla abierta. A últimas fechas se han empezado a utilizar estos dispositivos en esquemas de malla cerrada (Ref. 26). La razón para ello estriba en el descenso en el costo de la electrónica de control y en que es posible obtener información sobre el desplazamiento de los motores si se mide apropiadamente el voltaje en los embobinados, lo que ahorra los detectores de posición.

Se considera por tanto necesario realizar estudios conducentes al empleo de los motores de pasos como servomecanismos, que pueden competir favorablemente en costo y precisión con los motores de corriente directa.

11. REFERENCIAS

1. L. Alvarez, M. Dovalí, R. Canetti y J. Nieto. Cortadora automática de tubos. Memorias del IX Congreso de la Academia Nacional de Ingeniería. León, Gto. Septiembre 1983. pp. 107-112.

2. L. Alvarez y R. Canales. Diseño y construcción de una cortadora automática de tubos. Informe del Instituto de Ingeniería, UNAM. Abril 1981. 169 pp.

3. L. Alvarez, R. Canales y otros. Modificaciones a la cortadora automática de tubos. Informe del Instituto de Ingeniería, UNAM. Junio 1982. 156 pp.

4. L. Alvarez, R. Canales y otros. Empujador electro-electrónico de envases. Informe del Instituto de Ingeniería, UNAM. Agosto 1982. 349 pp.

5. A. Leenhouts. Stepping motors in industrial motion control. Proceedings of the Joint Automatic Control Conference. San Francisco, USA. 1980. pp. WP10-A.

6. B. C. Kuo. Incremental Motion Systems. SRL Publishing Company. Champaign, Illinois, USA. 1979. pp. 1-45, 114-133.

7. S. Papaioannou. Stepping motors a review. Computer and Electrical Engineering Vol. 7. Great Britain. 1980. pp. 243-266.

8. A. Hughes, P.J. Lawrenson and T.S. Davies. Factors determining high speed torque in hybrid motors. Proceedings, International Conference on Stepping Motors and Systems. Leeds, England. 1976. pp. 150-157.

9. S.H. Pollack. On stability characteristics of permanent-magnet step motors. Proceedigns of the Seventh Annual Symposium on Incremental Motion Systems and Devices. Illinois, USA. 1978. pp. 55-62.

10. R. Canales y R. Barrera. Análisis de Sistemas Dinámicos y Control Automático. Ed. Limusa. México. 1980. pp. 193-195.

11. R. Gauthier and others. Stepping motor dynamics and the phase plane. Proceedings of the Joint Automatic Control Conference. San Francisco, USA. 1980. pp. WP10-A.

12. F.M. DiNuzzo. Development of preprogrammed sequences for stepping motors. Proceedings of the Joint Automatic Control Conference. San Francisco, USA. 1980. pp. WP10-D.

13. T.J. Harned. An unique measurement system for experimental phase plane generation. Proceedings of the Joint Automatic Control Conference. San Francisco, USA. 1980. pp. WP10-E.

14. T.R. Frediksen. Applications of the close-loop stepping motor. IEEE Transactions on Automatic Control. Vol. AC-13 No. 5. USA. October 1968. pp. 464-474.

15. A. C. Leenhouts and C. S. Wilson. Torque control of P.M. step motors in high performance open loop applications. Proceedings of the Symposium on Incremental Motion Systems. Illinois, USA. 1979. pp. 1-15.

16. J.L. Willems. Stability Theory of Dynamical Systems. Nelson. London, England. 1970. pp. 145-169.

17. J.J. D´Azzo and C. Houpis. Linear Control Analysis and Design. Mc. Graw-Hill. USA. 1981. pp. 485-522.

18. B. Carnahan, H. A. Luther and J. O. Wilkes, Applied Numerical Methods. John Wiley 1969. U.S.A.

19. J. Stoer and R. Burlisch. Introduction to Numerical Analysis. Springer-Verlag. New York, USA. 1980. pp. 410-423.

20. K. Jensen and N. Wirth. Pascal: User Manual and Report. Springer Verlag. New York, USA. 1978. 167 pp.

21. Apple Computer, Inc. Apple Pascal: System Reference Manual. Apple Computer, Inc. Cupertino, USA. 1980. 209 pp.

22. Apple Computer, Inc. Apple Pascal: Language Reference Manual. Apple Computer, Inc. Cupertino, USA. 1980. 298 pp.

23. Motorola Inc. The Complete Microcomputer Data Library. Technical Information Center. Motorola Inc. USA. 1978, pp. 1-107 a 1-118.

24. Apple Computer, Inc. Apple Reference Manual. Apple Computer, Inc. Cupertino, USA. 1979. 196 pp.

25. Intel Corp. Component Data Catalog. Literature Department Intel Corp. Santa Clara, USA. 1981. pp. 8-59 a 8-69.

26. S.J. Bailey. Lessening the gap between incremental and continuos motion control. Control Engineering. USA. February 1987. pp. 72-78.

27. K. Venkataratnam and others. Stability of a stepping motor. Proceedings of the IEE. Vol. 113 No. 6, June 1971 pp. 805-812.

28. R. Canales and L. Alvarez-Icaza. Stepping motors control. Proceedings of the IFAC/IFIP Symposium on Real Time Digital Control Applications. Pergamon-Press. USA. 1983. pp. 527-533.

ANEXO A

PROGRAMAS PARA SIMULAR EL COMPORTAMIENTO DE LOS MOTORES DE PASOS

Este anexo contiene los listados de los siguientes programas:

MOTOR: genera la lista de tiempos de conmutación.

SIMULA: simula el efecto de la lista sobre el motor con un juego de parámetros distinto.

CURVAS: genera las tablas de condiciones deseadas de movimiento, tiempos iniciales de validez de las constantes e intervalos de integración.

PARAMETRO: modifica cualquier parámetro de los que se emplean para en MOTOR y SIMULA.

SERVICIO: consulta cualquier archivo de texto, enteros o reales y lo presenta de manera flexible.

GRAFSIM: grafica los resultados de las simulaciones

INTCPAR: simula la respuesta del motor a una sola conmutación

PARTETA: grafica las curvas par-posición angular que corresponden a los resultados de MOTOR o SIMULA.

```
(*$S+*)
(* MOTOR.TEXT *)
(*****************************************)
(*                                       *)
(* PROGRAMA PARA SIMULAR EL CONTROL Y EL COMPORTA- *)
(* MIENTO DE UN MOTOR DE PASOS.          *)
(*                                       *)
(* EL PROGRAMA TIENE POR OBJETO GENERAR UNA LISTA  *)
(* DE LOS TIEMPOS DE CONMUTACION QUE PRODUCIRAN EN *)
(* EN EL MOTOR DE PASOS UN MOVIMIENTO PREDETERMI-  *)
(* NADO.                                 *)
(* LOS PULSOS SE GENERAN CON BASE EN EL SIGUIENTE  *)
(* PROCEDIMIENTO:                        *)
(*      -SE SIMULA EL COMPORTAMIENTO DE MOTOR AL   *)
(*       APLICAR UN PASO                 *)
(*      -SE VALUA EL ERROR ENTRE EL COMPORTAMIENTO *)
(*       REAL Y EL DESEADO               *)
(*      -CON BASE EN EL ERROR SE DETERMINA EL MO-  *)
(*       MENTO EN QUE SE DEBE ENVIAR EL SIGUIENTE  *)
(*       PULSO                           *)
(*                                       *)
(*              LAIL      JULIO-1983     *)
(*              PROYECTO        2136     *)
(*                                       *)
(*****************************************)

PROGRAM MOTOR;

USES TRANSCEND;{*LIBRERIA DE FUNCIONES TRASCENDENTES*}

(* CATALOGO DE VARIABLES *)
(* --------------------- *)

CONST PI=3.1415926;
      DATOSMAX=50;{*NO. MAXIMO DE PUNTOS EN LAS CONDICIONES DESEADAS*}

VAR

    {*VARIABLES RELACIONADOS CON EL MOTOR DE PASOS*}

    NFASES{*NUMERO DE FASES DEL MOTOR DE PASOS*}
          :INTEGER;
    RESOLUCION,{*DIMENSION EN RADIANES DEL PASO*}
    R,{*RELACION DEL REDUCTOR DE VELOCIDAD*}
    TMAX,{*PAR MAXIMO QUE PUEDE APLICAR EL MOTOR*}
    AUXEL,{*PENDIENTE DE LA RECTA DE DISMINUCION DE PAR SEGUN VELOCIDAD*}
    P1,P2,{*VARIABLES AUXILIARES PARA CALCULAR EL PAR DISPONIBLE*}
    CONT,{*CONTADOR DEL NUMERO DE PASOS ENVIADOS*}

    {*VARIABLES RELACIONADOS CON EL SISTEMA DE CARGAS *}

    INERCIA,{*INERCIA DEL SISTEMA*}
    D,{*COEFICIENTE DE FRICCION VISCOSA*}
    FRICCION,{*FRICCION DE COULOMB*}
    M0,{*MASA DEL DISCO DE INERCIA*}
    R0,{*RADIO DEL DISCO D INERCIA*}
    M1,{*MASA DEL BRAZO*}
    L1,{*LONGITUD DEL BRAZO*}
    M2,{*MASA CONCENTRADA*}
    L2,{*RADIO DE LA MASA CONCENTRADA*}

    {*VARIABLES RELACIONADAS CON LA SIMULACION*}

    TAO,{*PAR REQUERIDO*}
    T,{*PAR DISPONIBLE*}
    TNETO,{*PAR DISPONIBLE MAXIMO*}
    ALFAD,{*ACELERACION ANGULAR DESEADA*}
    OMEGAR,{*VELOCIDAD ANGULAR REAL*}
    OMEGANT,{*VELOCIDAD ANGULAR ANTERIOR*}
    OMEGAD,{*VELOCIDAD ANGULAR DESEADA*}
    TETAR,{*POSICION ANGULAR REAL*}
    TETAD,{*POSICION ANGULAR DESEADA*}
    TETACON,{*ANGULO PARA LA CONMUTACION*}
    TIEMPO,{*TIEMPO CORRIENTE*}
    DELTATIE,{*TIEMPO CORRIENTE MENOS TIETAB[NACTUAL]*}
    TOL,{*TOLERANCIA PARA EL CALCULO DE LOS TIEMPOS DE CONMUTACION*}
    SIGANT,SIGACT,{*SIGNO DEL PAR DISPONIBLE:ANTERIOR Y CORRIENTE*}
    FRACPASO,{*FRACCION DE PASO ENTERO*}
    REDTIE,{*FACTOR DE REDUCCION DEL TIEMPO DE SIMULACION*}
    KP,KV{*CONSTANTES DE AMPLIFICACION DEL ERROR DE POSICION Y VELOCIDAD*}
         :REAL;

    C1,C2,C3,C4,{*CONSTANTES PARA DETERMINAR LA ACELERACION*}
    OMEGATAB,{*TABLA DE VELOCIDADES*}
    TETATAB,{*TABLA DE POSICIONES*}
    HTAB,{*TABLA DE PASOS DE INTEGRACION*}
    TIEMTAB{*TABLA DE TIEMPOS*}
           :ARRAY[1..DATOSMAX] OF REAL;

    TIPO,{*TIPO DE CONMUTACION*}
    NITER,{*NUMERO DE ITERACIONES DESDE LA ULTIMA CONMUTACION*}
    NMAX,{*NUMERO MAXIMO DE ITERACIONES SIN CONMUTACION*}
    IMPRESION,{*FRECUENCIA PARA LA SALIDA DE DATOS AL ARCHIVO DE DATOS*}
    CONTIMPRESION,{*ITERACIONES DESDE LA ULTIMA IMPRESION*}
    NACTUAL,{*REGISTRO ACTUAL DE LAS TABLAS DE CONDICIONES*}
    NUMDATOS,{*NO. DE REGISTROS DE LAS TABLAS DE CONDICIONES*}
    DESEADAS,{*FRECUENCIA PARA EL CALCULO DE LAS CONDICIONES DESEADAS*}
    NDESEADAS,{*ITERACIONES DESDE EL CALCULO DE CONDICONES DESEADAS*}
    BANDERA,{*INDICADOR DE CONDICION DE CONMUTACION*}
    I{*CONTADOR PARA REGISTROS DE TABLAS DE CONDICIONES*}
             :INTEGER;

    VELBAJ: BOOLEAN;{*INDICADOR DE VELOCIDAD BAJA*}

    {*VARIABLES RELACIONADAS CON LA INTEGRACION NUMERICA*}

    OMEGA1,{*PRIMERA ESTIMACION DE LA VELOCIDAD ANGULAR*}
    TETA1,{*PRIMERA ESTIMACION DE LA POSICION ANGULAR*}
    H,{*PASO DE INTEGRACION*}
    AK11,AK12,AK21,AK22,{*VARIABLES AUXILIARES EN LA INTEGRACION*}
    FACH{*FACTOR DE AMPLIFICACION DEL PASO DE INTEGRACION*}
        :REAL;

    {*VARIABLES AUXILIARES Y APUNTADORES A ARCHIVOS*}

    MUESTRAS,{*NUMERO DE MUESTRAS DEL RESULTADO DE LA SIMULACION*}
    PASOS{*NUMERO DE PASOS ENVIADOS*}
         : INTEGER;

    A0,B0,C0: REAL;{*VARIABLES AUXILIARES PARA LECTURA*}
```

```
    AE1:INTERACTIVE;

    AE2,(*ARCHIVO QUE CONTIENE LAS TABLAS DE CONDICIONES DESEADAS*)
    AS1,(*   "    CON LAS MUESTRAS DEL RESULTADO DE LA SIMULACION*)
    AS2,(*   "     "  LOS TIEMPOS DE CONMUTACION*)
    AS3 (*   "     "  ANGULOS "     "        *)
       :FILE OF REAL;

{*******.*********}
(* LECTURA DE DATOS *)
{*********************}

(*$I %5:LECTURA.TEXT *) (*INCLUYE LOS PROCEDIMIENTOS PARA LECTURA Y *)
                       (*CALCULO DE CONDICIONES DESEADAS  *)

(* FUNCION SIGNO*)

FUNCTION SGN(OMEGA:REAL):REAL;

BEGIN(*SGN*)

 IF OMEGA>0
  THEN SGN:=1
  ELSE IF OMEGA <0
       THEN SGN:=-1
       ELSE SGN:=0;
END;(*SGN*)

(* CALCULA EL PAR NECESARIO EN FUNCION DE LOS PARAMETROS *)
(* DEL SISTEMA Y DE LOS ERRORES EN VELOCIDAD Y POSICION  *)
(* LAS CONSTANTES KP Y KV DETERMINAN LA UBICACION DE LOS *)
(* POLOS DE LA FUNCION DE ERROR. *)

PROCEDURE PARREQUERIDO;

BEGIN(*PARREQUERIDO*)

  TAO:=INERCIA*(KP*(TETAD-TETAR)+KV*(OMEGAD-OMEGAR)+ALFAD)+B*OMEGAR
       +FRICCION*SGN(OMEGAR);

END;(*PARREQUERIDO*)

(* CALCULA EL DISPONIBLE EN FUNCION DE LA POSICION Y VELOCIDAD *)
(* ANGULAR CORRIENTE *)

PROCEDURE PARDISPONIBLE(OMEGA,TETA:REAL);

BEGIN(*PARDISPONIBLE*)

  T:=-(TMAX-ABS(OMEGA)*AKVEL)*SIN(K)*TETA-CONT*K2);
  IF ABS(OMEGA*AKVEL) > TMAX THEN T:=0;(*ANULA EL PAR A VELOCIDADES ALTAS *)

END;(*PARDISPONIBLE*)

(* INTEGRACION POR EL METODO DE RUNGE-KUTTA DE SEGUNDO ORDEN CON PARAMETROS *)
(* IGUALES (INTEGRACION TRAPEZOIDAL) .*)

PROCEDURE F1(OMEGA:REAL;VAR AK: REAL);

BEGIN(*F1*)

  AK:=OMEGA;

END;(*F1*)

PROCEDURE F2(OMEGA,TETA: REAL;VAR AK: REAL);

BEGIN(*F2*)

  PARDISPONIBLE(OMEGA,TETA);

  AK:=(T-B*OMEGA-FRICCION*SGN(OMEGA))/INERCIA;

END;(*F2*)

PROCEDURE RUNGE;

BEGIN(*RUNGE*)

  F1(OMEGAR,AK11);
  F2(OMEGAR,TETAR,AK12);

  TETA1:=TETAR+H*AK11;(*ESTIMACION DE LOS VALORE DE TETA Y OMEGA *)
  OMEGA1:=OMEGAR+H*AK12;

  F1(OMEGA1,AK21);
  F2(OMEGA1,TETA1,AK22);

  TETAR:=TETAR+H/2*(AK11+AK21);(* VALORES FINALES DE TETA Y OMEGA*)
  OMEGAR:=OMEGAR+H/2*(AK12+AK22);

END;(*RUNGE*)

(* ESCRIBE EN LOS ARCHIVOS DE SALIDA EL TIEMPO, EL ANGULO Y EL *)
(* TIPO(IZQ. O DER.) DE LA ULTIMA CONMUTACION*)

PROCEDURE ESCRIBE;

BEGIN(*ESCRIBE*)

  AS1^:=TIEMPO;
  PUT(AS1);
  AS1^:=TIPO;
  PUT(AS1);

  AS3^:=TETAR;
  PUT(AS3);
  AS3^:=CONT;
  PUT(AS3);
  PASOS:=PASOS+1;(*ACTUALIZA INDICADOR DE PASOS ENVIADOS*)

END;(*ESCRIBE*)

(* TOMA MUESTRAS DE LOS RESULTADOS DE LA SIMULACION Y LOS IMPRIME *)
(* EN UN ARCHIVO DE SALIDA. LA FRECUENCIA DE MUESTREO DEPENDE DEL *)
(* PRODUCTO IMPRESION*H . GUARDA VELOCIDAD Y POSICION;REALES Y DESEADAS.*)

PROCEDURE GRAFICA;
```

```
BEGIN(*GRAFICA*)

  WRITELN('GRAFICA');

  AS2^:=TIEMPO;
  PUT(AS2);
  AS2^:=TETAD;
  PUT(AS2);
  AS2^:=TETAR;
  PUT(AS2);
  AS2^:=OMEGAD;
  PUT(AS2);
  AS2^:=OMEGAR;
  PUT(AS2);
  MUESTRAS:=MUESTRAS+1;(* ACTUALIZA INDICADOR DE MUESTAS TOMADAS *)

END;(*GRAFICA*)

(* CALCULA EL MAXIMO PAR PROMEDIO EN FUNCION DE LA VELOCIDAD *)
(* CORRIENTE.*)

PROCEDURE PARNETO;

BEGIN(*PARNETO*)

  TNETO:=(TMAX-AKVEL*ABS(OMEGAR));
  IF TNETO< 0 THEN TNETO:=0;(*ANULA EL PAR A VELOCIDADES ALTAS *)
  TNETO:=TNETO*2/PI;

END;(*PARNETO*)

(* ZONA PARA EL CALCULO DE LAS CONDICIONES DE CONMUTACION *)

(* CALCULA EL ANGULO DE CONMUTACION PARA MOVIMIENTOS POSITIVOS *)

PROCEDURE CALTETA1;

BEGIN(*CALTETA1*)

  PARNETO;(* PAR PROMEDIO DISPONIBLE *)

  (* CALCULA LA DIFERENCIA ENTRE LA POSICION CORRIENTE Y LA DE EQUILIBRIO *)
  (* MAS CERCANA.*)
  FRACPASO:=(K1*TETAR-CONT*K2)/2/PI;
  FRACPASO:=(FRACPASO-TRUNC(FRACPASO))*2*PI;
  IF FRACPASO < 0 THEN FRACPASO:=2*PI+FRACPASO;

  IF ABS(TAO) < TNETO

    THEN(*PAR REQUERIDO MENOR QUE MAXIMO PAR PROMEDIO *)

      BEGIN(*THEN ABS(TAO)*)

        (*TETACON EN ANGULO ELECTRICO *)
        TETACON:=TAO/SQRT(2)/TNETO;
        TETACON:=PI/4-ATAN(TETACON/SQRT(1-SQR(TETACON)));
        TETACON:=TETACON-FRACPASO;

      END(*THEN ABS(TAO)*)

    ELSE

      IF TAO > 0

        THEN(*TAO>0*)  TETACON:=0
        ELSE(*TAO<=0*)  TETACON:=PI/2-TOL-FRACPASO;
    (*ELSE ABS(TAO)*)

  TETACON:=TETACON/K1+TETAR;(*TETACON A ANGULO MECANICO*)

END;(*CALTETA1*)

(* CALCULA EL ANGULO DE CONMUTACION PARA MOVIMIETOS NEGATIVOS *)

PROCEDURE CALTETA2;

BEGIN(*CALTETA2*)

  PARNETO;(* PAR PROMEDIO DISPONIBLE *)

  (*DIFERENCIA ENTRE LA POSICION CORRIENTE Y LA DE EQUILIBRIO DEL MOTOR *)
  FRACPASO:=(K1*TETAR-CONT*K2)/2/PI;
  FRACPASO:=(FRACPASO-TRUNC(FRACPASO))*2*PI;
  IF FRACPASO > 0 THEN FRACPASO:=FRACPASO-2*PI;

  IF ABS(TAO) < TNETO(* PAR REQUERIDO MENOR QUE EL PAR PROMEDIO *)

    THEN

      BEGIN(*THEN ABS(TAO)*)

        (* TETACON EN ANGULO ELECTRICO *)
        TETACON:=TAO/SQRT(2)/TNETO;
        TETACON:=-PI/4-ATAN(TETACON/SQRT(1-SQR(TETACON)));
        TETACON:=TETACON-FRACPASO;

      END(*THEN ABS(TAO)*)

    ELSE

      IF TAO < 0

        THEN(*TAO<0*)  TETACON:=0
        ELSE(*TAO>=0*)  TETACON:=-PI/2+TOL-FRACPASO;
    (*ELSE ABS(TAO)*)

  TETACON:=TETACON/K1+TETAR;(*TETACON A ANGULO MECANICO*)

END;(*CALTETA2*)

(* VERIFICA LA OCURRENCIA DE LAS CONDICIONES DE CONMUTACION  *)
(* ESTAS SON: CRUCE POR CERO EN LAS CURVAS PAR VS. POSICION .*)
(* NUMERO EXCESIVO DE ITERACIONES SIN CONMUTACION Y VELOCIDAD*)
(* DEMASIADO BAJA(MOTOR PARADO) *)

PROCEDURE VERIFICA;

BEGIN(*VERIFICA*)

 IF (BANDERA=0) OR (NITER)=NMAX)(*LAS CONDICIONES DE ENTRADA SON:CONMUTACION*)
```

```
                        (*EN EL PERIODO ANTERIOR Y NO. DE ITERACIONES*)
THEN

 BEGIN(*THEN BANDERA*)

   PARREQUERIDO;(*PAR NECESARIO*)

   SIGACT:=-SIN(K)*TETAR-K2*CONT);(*SIGNO DE LA CURVA T VS. TETA*)
   IF ABS(OMEGAR)<6E-4 THEN VELBAJ:=FALSE ELSE VELBAJ:=TRUE;

   (*CONDICIONES COMPLETAS PARA CONMUTACION POSITIVA*)
   IF(OMEGAR > 0) OR (NOT(VELBAJ) AND (OMEGAD>0)) OR
     ((OMEGAR(=0) AND (OMEGANT)=0) AND (OMEGAD>0) AND (SIGACT)=0))

       THEN

         (*VERIFICA NECESIDAD DE CALCULO DE COND. DE CONMUTACION*)
         BEGIN(*THEN OMEGA*)

           IF ((SIGANT)=0) AND (SIGACT(=0)) OR
              ((NITER)NMAX) AND (SIGACT(=0)) OR
              ((OMEGAR(=0) AND (OMEGANT)=0) AND (SIGACT)=0) AND (OMEGAD>0))

                THEN

                  BEGIN(*THEN SIGANT*)

                    BANDERA:=1;(*CALCULO DE CONDICIONES DE LA PROXIMA  *)
                    CALTETA1;  (*CONMUTACION PARA MOVIMIENTOS POSITIVOS*)
                    IF NITER)NMAX THEN NITER:=0;(*ACTUALIZA NO. ITERAC.*)

                  END;(*THEN SIGANT*)

         END;(*THEN OMEGA*)

   (*CONDICIONES COMPLETAS PARA CONMUTACION NEGATIVA*)
   IF(OMEGAR < 0) OR (NOT(VELBAJ) AND (OMEGAD<0)) OR
     ((OMEGAR)=0) AND (OMEGANT(=0) AND (OMEGAD(0) AND (SIGACT(=0))

     THEN

         (*VERIFICA LA NECESIDAD DE CALCULAR CONDICIONES DE CONMUTACION*)
         BEGIN(*THEN OMEGA<0*)

           IF ((SIGANT(=0) AND (SIGACT)=0)) OR
              ((NITER)NMAX) AND (SIGACT)=0)) OR
              ((OMEGAR)=0) AND (OMEGANT(=0) AND (SIGACT(=0) AND (OMEGAD(0))

                THEN

                  BEGIN(*THEN SIGANT*)

                    BANDERA:=-1;
                    CALTETA2;
                    IF NITER)NMAX THEN NITER:=0;

                  END;(*THEN SIGANT*)

         END;(*THEN OMEGA<0*)
```

```
         SIGANT:=SIGACT;(*ACTUALIZA SIGNO Y VELOCIDAD ANTERIORES*)
         OMEGANT:=OMEGAR;

       END;(*THEN BANDERA*)

 END;(*VERIFICA*)

 (* REALIZA UNA CONMUTACION HACIA ADELANTE *)

 PROCEDURE INYECTAPOS;

 BEGIN(*INYECTAPOS*)

   WRITELN('         INYECTAPOS');

   TIPO:=1;(*DEFINE APUNTADORES Y ACTUALIZA INDICADORES*)
   BANDERA:=0;
   CONT:=CONT+1;
   ESCRIBE;
   NITER:=0;
   IF SIGANT<0 THEN SIGANT:=1;

 END;(*INYECTAPOS*)

 (* REALIZA UNA CONMUTACON HACIA ATRAS *)

 PROCEDURE INYECTANEG;

 BEGIN(*INYECTANEG*)

   WRITELN('         INYECTANEG');

   TIPO:=-1;(*DEFINE APUNTADORES Y ACTUALIZA INDICADORES*)
   CONT:=CONT-1;
   ESCRIBE;
   BANDERA:=0;
   NITER:=0;
   IF SIGANT > 0 THEN SIGANT:=-1;

 END;(*INYECTANEG*)

 (* VERIFICA QUE SE CUMPLAN LAS CONDICIONES PARA LA CONMUTACION *)

 PROCEDURE CONMUTA;

 BEGIN(*CONMUTA*)

   NITER:=NITER+1;(*ACTUALIZA NO. DE ITERACIONES*)

   IF BANDERA <> 0(*VERIFICA NECESIDAD DE ENVIAR PULSO*)

     THEN

       BEGIN(*THEN BANDERA<>0*)

         IF BANDERA=1

           THEN

             BEGIN(*THEN BANDERA=1*)
```

```
            (*VERIFICA QUE EL ANGULO CUMPLA CON LA CONDICION *)
            IF TETAR)=TETACON THEN INYECTAPOS;

          END(*THEN BANDERA=1*)

        ELSE

          BEGIN(*ELSE BANDERA=1*)

            (*VERIFICA QUE EL ANGULO CUMPLA CON LA CONDICION *)
            IF TETAR(=TETACON THEN INYECTANEG;

          END;(*ELSE BANDERA=1*)

      END;(*THEN BANDERA()0*)

END;(*CONMUTA*)

(*-------------------------------*)
(* INICIA EL PROGRAMA PRINCIPAL *)
(*-------------------------------*)

BEGIN(*MOTOR*)

  LECTURA;
  INICIALIZA;
  (*ABRE ARCHIVOS DE DATOS Y REPOSICIONA APUNTADORES *)
  (*LOS ARCHIVOS DEBEN EXISTIR EN DISCO              *)


  RESET(AS1,'#5:SENALES.DATA');
  SEEK(AS1,0);
  RESET(AS2,'#5:PLANOS.DATA');
  SEEK(AS2,0);
  RESET(AS3,'#5:PARESTETA.DATA');
  SEEK(AS3,0);

  (*ITERA MIENTRAS NO SE AGOTE EL TIEMPO DE SIMULACION*)
  WHILE TIEMPO < TIEMTAB(NUMDATOS)/REDTIE DO

    BEGIN(*WHILE TIEMPO*)

      IF NDESEADAS >= DESEADAS(*CALCULO DE CONDICIONES DESEADAS*)

        THEN
          BEGIN(*THEN NDESEDAS*)

            VALDESEADOS;
            WRITELN('DIF POS: ',TETAD-TETAR);
            WRITELN('DIF VEL: ',OMEGAD-OMEGAR);
            WRITELN('TIEMPO: ',TIEMTAB(NUMDATOS)/REDTIE-TIEMPO);
            WRITELN;
            NDESEADAS:=0;

          END;(*THEN NDESEADAS*)

      NDESEADAS:=NDESEADAS+1;(*INDICADOR DE CALCULO DE COND. DESEADAS*)

      RUNGE;(*INTEGRACION NUMERICA*)

      TIEMPO:=TIEMPO+H;(*ACTUALIZA TIEMPO*)

      VERIFICA;(*VERIFICA COND. DE CONMUTACION*)
      CONMUTA;(*REALIZA CONMUTACION DE SER NECESARIO*)

      (*ACTUALIZA APUNTADOR A TABLAS*)
      IF TIEMPO > TIEMTAB(NACTUAL+1) THEN NACTUAL:=NACTUAL+1;

      IF CONTIMPRESION=IMPRESION(*TOMA MUESTRAS DE RESULTADOS*)

        THEN

          BEGIN(*THEN CONTIMP*)

            GRAFICA;
            CONTIMPRESION:=0;

          END(*THEN CONTIMP*)

        ELSE

          (*ACTULIZA INDICADOR PARA TOMA DE MUESTRAS*)
          CONTIMPRESION:=CONTIMPRESION+1;

    END;(*WHILE TIEMPO*)

  (*IMPRIME NO. DE REGISTROS*)
  WRITELN('NO. REG EN PLANOS: ',MUESTRAS);
  WRITELN('NO. REG EN SENALES Y PARESTETA:',PASOS);

  CLOSE(AS1);(*CIERRA ARCHIVOS*)
  CLOSE(AS2);
  CLOSE(AS3);

  REWRITE(AS1,'#5:NUMSENALES.DATA');
  AS1^:=PASOS;PUT(AS1);(*GUARDA NO. DE PASOS*)
  AS1^:=MUESTRAS;PUT(AS1);(*GUARDA NO. DE MUESTRAS*)
  CLOSE(AS1,LOCK);

END.(*MOTOR*)
```

```
(* LECTURA.TEXT*)
(* SUBPROGRAMA DE MOTOR,TEXT                          *)
(* EN EL SE DEFINEN LAS CURVAS DE ACELERACION PO-*)
(* SIBLES.ESTAS DEBEN SER DE LA FORMA:           *)
(*              ALFA=C1+C2*T+C3*SEN(C4*T)        *)
(*                                               *)
(* LAS TRAYECTORIAS DESEADAS SE PARTEN  EN INTER-*)
(* VALOS DE TIEMPO. LOS VALORES DE C1 A C4 SON   *)
(* VALIDOS DURANTE TODO EL INTERVALO.            *)
(* ADICIONALMENTE SE PROPORCIONA LA VELOCIDAD  Y *)
(* POSICION ANGULAR AL INICIO DE ESTOS INTERVALOS*)

(*=====================*)
(* LECTURA DE DATOS *)
(*=====================*)

(*CADA REGISTRO CONTIENE LA ACELERACION(EN TERMINOS DE C1 A C5),VELOCIDAD,*)
(*POSICION Y PASO DE INTEGRACION VS. TIEMPO EN QUE COMIENZAN A SER VALIDOS*)

PROCEDURE LEE;

BEGIN(*LEE*)

  C1[I]:=AE2^;GET(AE2);(*CONSTANTES QUE DEFINEN LA ACELERACION*)
  C2[I]:=AE2^;GET(AE2);
  C3[I]:=AE2^;GET(AE2);
  C4[I]:=AE2^;GET(AE2);

  OMEGATAB[I]:=AE2^;GET(AE2);(*VELOCIDAD AL INICIO DEL INTERVALO*)

  TETATAB[I]:=AE2^;GET(AE2);(*POSICION AL INICIO DEL INTERVALO*)

  HTAB[I]:=AE2^;GET(AE2);(*PASO DE INTEGRACION PARA EL INTERVALO*)

  TIEMTAB[I]:=AE2^;GET(AE2);(*TIEMPO EN QUE INICIA EL INTERVALO*)

END;(*LEE*)

(*LEE LOS PARAMETROS PARA LA SIMULACION*)

PROCEDURE LECTURA;

BEGIN(*LECTURA*)

  WRITELN('LECTURA DE DATOS');

  RESET(AE1,'#5:PARAMETROS.DATA');
  READ(AE1,AO,RESOLUCION,TMAX,AKVEL,R);
  NFASES:=TRUNC(AO);
  READ(AE1,TOL,KP,KV);
  READ(AE1,FRICCION,B);
  READ(AE1,RO,MO,L1,M1,L2,M2);
  READ(AE1,AO,BO,CO);
  IMPRESION:=TRUNC(AO);
  DESEADAS:=TRUNC(BO);
  NMAX:=TRUNC(CO);
  CLOSE(AE1);

  RESET(AE2,'#5:CURVAS.DATA');
  NUMDATOS:=TRUNC(AE2^);
  GET(AE2);
  FOR I:=1 TO NUMDATOS DO LEE;
  CLOSE(AE2);

END;(*LECTURA*)

(*=============================*)
(* INICIALIZACION DE VARIABLES *)
(*=============================*)

PROCEDURE INICIALIZA;

BEGIN(*INICIALIZA*)

  WRITELN('INICIALIZACION:');

  (* EL FACTOR DE REDUCCION DE TIEMPO SIRVE PARA SIMULAR UNICAMENTE UNA *)
  (* FRACCION DEL TIEMPO TOTAL*)

  WRITE('FACTOR DE REDUCCION DEL TIEMPO: ');
  READLN(REDTIE);

  (* EL FACTOR DE AMPLIFICACION DEL PASO DE INTEGRACION CAMBIA EL VALOR DEL *)
  (* PASO DE INTEGRACION *)

  WRITE('FACTOR DE AMPLIFICACION DE H: ');
  READLN(FACH);

  (* INICIALIZA VARIABLES PARA LA SIMULACION *)

  NACTUAL:=1;
  TIEMPO:=TIEMTAB[NACTUAL];
  H:=HTAB[NACTUAL]*FACH;

  CONTIMPRESION:=IMPRESION;
  MUESTRAS:=0;
  PASOS:=0;
  NDESEADAS:=DESEADAS;

  (* CALCULA LA INERCIA EQUIVALENTE. CONSIDERA TRES TIPO DE MASAS: DISCOS,*)
  (* BRAZOS DE MASA UNIFORME Y MASAS CONCENTRADAS EN ALGUN PUNTO. LA NATU-*)
  (* RALEZA DE LAS CARGAS DEL SISTEMA DEBE DARSE EN FORMA EQUIVALENTE A   *)
  (* ESTA *)

  INERCIA:=M0*SQR(R0)/2+M1*SQR(L1)/3+M2*SQR(L2);

  (* MODIFICA LAS CONSTANTES DEL MOTOR SEGUN LA REDUCCION EMPLEADA *)

  TMAX:=TMAX*R;
  AKVEL:=AKVEL*SQR(R);

  K1:=2*PI/NFASES/RESOLUCION*R;
  K2:=2*PI/NFASES;

  NITER:=0;

  (* ACTUALIZA LA POSICION INICAL DEL MOTOR PARA HACERA COINCIDIR CON UNA *)
  (* POSICION DE EQUILIBRIO *)

  FRACPASO:=K1*TETATAB[NACTUAL]/2/PI;
```

```
  FRACPASO:=(FRACPASO-TRUNC(FRACPASO))/K1*2*PI;

  TETAR:=TETATAB[NACTUAL]-FRACPASO;

  (* INICIALIZA INDICADORES Y AUXILIARES DEL TIPO DE CONMUTACION *)

  CONT:=0;
  BANDERA:=0;
  FRACPASO:=0;
  OMEGANT:=OMEGATAB[NACTUAL];
  OMEGAR:=OMEGATAB[NACTUAL];

  WRITELN('           TIEMPO: ',TIEMPO);
  WRITELN('           NUMDATOS: ',NUMDATOS);

END;(*INICIALIZA*)

(* CALCULA LAS CONDICIONES DESEADAS *)

(* POSICION DESEADA *)

PROCEDURE POSICION;

BEGIN(*POSICION*)

  TETAD:= TETATAB[NACTUAL]+(OMEGATAB[NACTUAL] +C3[NACTUAL]/C4[NACTUAL])*
          DELTATIE + C1[NACTUAL]*SQR(DELTATIE)/2 + C2[NACTUAL]*
          SQR(DELTATIE)*DELTATIE/6 - C3[NACTUAL]/SQR(C4[NACTUAL])*
          SIN(C4[NACTUAL]*DELTATIE);

END;(*POSICION*)

(* VELOCIDAD DESEADA *)

PROCEDURE VELOCIDAD;

BEGIN(*VELOCIDAD*)

  OMEGAD:= OMEGATAB[NACTUAL]+C3[NACTUAL]/C4[NACTUAL]+C1[NACTUAL]*DELTATIE+
           C2[NACTUAL]*SQR(DELTATIE)/2-C3[NACTUAL]/C4[NACTUAL]*
           SIN(C4[NACTUAL]*DELTATIE);

END;(*VELOCIDAD*)

(* ACELERACION DESEADA *)

PROCEDURE ACELERACION;

BEGIN(*ACELERACION*)

  ALFAD:=C1[NACTUAL]+C2[NACTUAL]+C3[NACTUAL]*
         SIN(C4[NACTUAL]*DELTATIE);

END;(*ACELERACION*)

(* CONDICIONES CINEMATICAS DESEADAS *)

PROCEDURE VALDESEADOS;

BEGIN(*VALDESEADOS*)

  DELTATIE:=TIEMPO-TIEMTAB[NACTUAL];(*DEFINE EL TIEMPO CON RELACION AL *)
                                    (*INICIO DE INTERVALO*)
  POSICION;
  VELOCIDAD;
  ACELERACION;

END;(*VALDESEADOS*)
```

```
(*$S+*)
(* SIMULA.TEXT *)
(*****************************************************)
(*                                                   *)
(* PROGRAMA PARA SIMULAR EL CONTROL Y EL COMPORTA-   *)
(* MIENTO DE UN MOTOR DE PASOS.                      *)
(*                                                   *)
(* EL PROGRAMA TIENE POR OBJETO COMPROBAR EL EFEC-   *)
(* DE LA LISTA DE TIEMPOS DE CONMUTACION QUE SE      *)
(* PRODUCE EN EL PROGRAMA MOTOR CUANDO LAS CONDI-    *)
(* CIONES DE CARGA DEL SISTEMA CAMBIAN.  SE PERSI-   *)
(* QUE PROBAR LA ROBUSTEZ DE ESTA LISTA.             *)
(*                                                   *)
(*                LAIL       JUL10-1983              *)
(*                PROYECTO         2136              *)
(*                                                   *)
(*****************************************************)

PROGRAM SIMULA;

USES TRANSCEND;(*LIBRERIA DE FUNCIONES TRASCENDENTES*)

(* CATALOGO DE VARIABLES *)
(* --------------------- *)

CONST PI=3.14159;
      DATOSMAX=50;(*NO. MAXIMO DE PUNTOS EN LAS CONDICIONES DESEADAS*)

VAR

    (*VARIABLES RELACIONADOS CON EL MOTOR DE PASOS*)

    NFASES(*NUMERO DE FASES DEL MOTOR DE PASOS*)
          :INTEGER;
    RESOLUCION,(*DIMENSION EN RADIANES DEL PASO*)
    R,(*RELACION DEL REDUCTOR DE VELOCIDAD*)
    TMAX,(*PAR MAXIMO QUE PUEDE APLICAR EL MOTOR*)
    ALVEL,(*PENDIENTE DE LA RECTA DE DISMINUCION DE PAR SEGUN VELOCIDAD*)
    P1,P2: (*VARIABLES AUXILIARES PARA CALCULAR EL PAR DISPONIBLE*)
    CONT,(*CONTADOR DEL NUMERO DE PASOS ENVIADOS*)

    (*VARIABLES RELACIONADOS CON EL SISTEMA DE CARGAS *)

    INERCIA,(*INERCIA DEL SISTEMA*)
    D,(*COEFICIENTE DE FRICCION VISCOSA*)
    FRICCION,(*FRICCION DE COULOMB*)
    MO,(*MASA DEL DISCO DE INERCIA*)
    RO,(*RADIO DEL DISCO D INERCIA*)
    MI,(*MASA DEL BRAZO*)
    L1,(*LONGITUD DEL BRAZO*)
    M2,(*MASA CONCENTRADA*)
    L2,(*RADIO DE LA MASA CONCENTRADA*)

    (*VARIABLES RELACIONADAS CON LA SIMULACION*)

    T,(*PAR DISPONIBLE*)
    ALFAD,(*ACELERACION ANGULAR DESEADA*)
    OMEGAR,(*VELOCIDAD ANGULAR REAL*)
    OMEGAD,(*VELOCIDAD ANGULAR DESEADA*)
    TETAR,(*POSICION ANGULAR REAL*)
    TETAD,(*POSICION ANGULAR DESEADA*)
    TIEMPO,(*TIEMPO CORRIENTE*)
    TCONMUTACION,(*TIEMPO DE LA SIGUIENTE CONMUTACION*)
    DELTATIE,(*TIEMPO CORRIENTE MENOS TIETAB[NACTUAL]*)
    FRACPASO,(*FRACCION DE PASO ENTERO*)
    REDTIE(*FACTOR DE REDUCCION DEL TIEMPO DE SIMULACION*)
        :REAL;

    C1,C2,C3,C4,(*CONSTANTES PARA DETERMINAR LA ACELERACION*)
    OMEGATAB,(*TABLA DE VELOCIDADES*)
    TETATAB,(*TABLA DE POSICIONES*)
    HTAB,(*TABLA DE PASOS DE INTEGRACION*)
    TIEMTAB(*TABLA DE TIEMPOS*)
          :ARRAY[1..DATOSMAX] OF REAL;

    TIPO,(*TIPO DE CONMUTACION*)
    IMPRESION,(*FRECUENCIA PARA LA SALIDA DE DATOS AL ARCHIVO DE DATOS*)
    CONTIMPRESION,(*ITERACIONES DESDE LA ULTIMA IMPRESION*)
    NACTUAL,(*REGISTRO ACTUAL DE LAS TABLAS DE CONDICIONES*)
    NUMDATOS,(*NO. DE REGISTROS DE LAS TABLAS DE CONDICIONES*)
    DESEADAS,(*FRECUENCIA PARA EL CALCULO DE LAS CONDICIONES DESEADAS*)
    NDESEADAS,(*NO. ITERACIONES DESDE EL CALCULO DE CONDICONES DESEADAS*)
    I(*CONTADOR PARA REGISTROS DE TABLAS DE CONDICIONES*)
             :INTEGER;

    (*VARIABLES RELACIONADAS CON LA INTEGRACION NUMERICA*)

    OMEGA1,(*PRIMERA ESTIMACION DE LA VELOCIDAD ANGULAR*)
    TETA1,(*PRIMERA ESTIMACION DE LA POSICION ANGULAR*)
    H,(*PASO DE INTEGRACION*)
    AK11,AK12,AK21,AK22,(*VARIABLES AUXILIARES EN LA INTEGRACION*)
    FACH(*FACTOR DE AMPLIFICACION DEL PASO DE INTEGRACION*)
        :REAL;

    (*VARIABLES AUXILIARES Y APUNTADORES A ARCHIVOS*)

    MUESTRAS(*NUMERO DE MUESTRAS DEL RESULTADO DE LA SIMULACION*)
         : INTEGER;

    AO,BO,CO: REAL;(*VARIABLES AUXILIARES PARA LECTURA*)

    AE1:INTERACTIVE;

    AE2,(*ARCHIVO QUE CONTIENE LAS TABLAS DE CONDICIONES DESEADAS*)
    AE3,(*   "     "     "     LA TABLA DE TIEMPOS DE CONMUTACION*)
    AS1(*   "     CON LAS MUESTRAS DEL RESULTADO DE LA SIMULACION*)
       :FILE OF REAL;

(*********************)
(* LECTURA DE DATOS *)
(*********************)

(* EN EL SE DEFINEN LAS CURVAS DE ACELERACION PO-*)
(* SIBLES,ESTAS DEBEN SER DE LA FORMA:           *)
(*              ALFA=C1+C2*T+C3*SEN(C4*T)        *)
(*                                               *)
(* LAS TRAYECTORIAS DESEADAS SE PARTEN  EN INTER-*)
(* VALOS DE TIEMPO. LOS VALORES DE C1 A C4 SON   *)
(* VALIDOS DURANTE TODO EL INTERVALO.            *)
(* ADICIONALMENTE SE PROPORCIONA LA VELOCIDAD  Y *)
```

```
(* POSICION ANGULAR AL INICIO DE ESTOS INTERVALOS*)

(*=====================*)
(* LECTURA DE DATOS *)
(*=====================*)

(*CADA REGISTRO CONTIENE LA ACELERACION(EN TERMINOS DE C1 A C5),VELOCIDAD,*)
(*POSICION Y PASO DE INTEGRACION VS, TIEMPO EN QUE COMIENZAN A SER VALIDOS*)

PROCEDURE LEE;

BEGIN(*LEE*)

  C1[I]:=AE2^;GET(AE2);(*CONSTANTES QUE DEFINEN LA ACELERACION*)
  C2[I]:=AE2^;GET(AE2);
  C3[I]:=AE2^;GET(AE2);
  C4[I]:=AE2^;GET(AE2);

  OMEGATAB[I]:=AE2^;GET(AE2);(*VELOCIDAD AL INICIO DEL INTERVALO*)

  TETATAB[I]:=AE2^;GET(AE2);(*POSICION AL INICIO DEL INTERVALO*)

  HTAB[I]:=AE2^;GET(AE2);(*PASO DE INTEGRACION PARA EL INTERVALO*)

  TIEMTAB[I]:=AE2^;GET(AE2);(*TIEMPO EN QUE INICIA EL INTERVALO*)

END;(*LEE*)

(*LEE LOS PARAMETROS PARA LA SIMULACION*)

PROCEDURE LECTURA;

BEGIN(*LECTURA*)

  WRITELN('LECTURA DE DATOS');

  RESET(AE1,'#5:PARAMETROS.DATA');
  READ(AE1,NFASES,RESOLUCION,TMAX,AKVEL,R);
  READ(AE1,A0,B0,C0);
  READ(AE1,FRICCION,B);
  READ(AE1,R0,M0,L1,M1,L2,M2);
  READ(AE1,A0,B0,C0);
  IMPRESION:=TRUNC(A0);
  DESEADAS:=TRUNC(B0);
  CLOSE(AE1);

  RESET(AE3,'#5:SENALES.DATA');(*TABLAS DE TIEMPOS DE CONMUTACION*)
  TCONMUTACION:=AE3^;GET(AE3);(*PRIMER TIEMPO*)
  TIPO:=TRUNC(AE3^);GET(AE3);(*PRIMER TIPO DE CONMUTACION*)

  RESET(AE2,'#5:CURVAS.DATA');(*TABLAS DE CONDICIONES DESEADAS*)
  NUMDATOS:=TRUNC(AE2^);
  GET(AE2);
  FOR I:=1 TO NUMDATOS DO LEE;
  CLOSE(AE2);

END;(*LECTURA*)

(*=====================================*)
(* INICIALIZACION DE VARIABLES *)
(*=====================================*)

PROCEDURE INICIALIZA;

BEGIN(*INICIALIZA*)

  WRITELN('INICIALIZACION');

  (* EL FACTOR DE REDUCCION DE TIEMPO SIRVE PARA SIMULAR UNICAMENTE UNA *)
  (* FRACCION DEL TIEMPO TOTAL*)

  WRITE('FACTOR DE REDUCCION DEL TIEMPO: ');
  READLN(REDTIE);

  (* EL FACTOR DE AMPLIFICACION DEL PASO DE INTEGRACION CAMBIA EL VALOR DEL *)
  (* PASO DE INTEGRACION *)

  WRITE('FACTOR DE AMPLIFICACION DE H: ');
  READLN(FACH);

  (* INICIALIZA VARIABLES PARA LA SIMULACION *)

  NACTUAL:=1;
  TIEMPO:=TIEMTAB[NACTUAL];
  H:=HTAB[NACTUAL]*FACH;

  CONTIMPRESION:=IMPRESION;
  MUESTRAS:=0;
  NDESEADAS:=DESEADAS;

  (* CALCULA LA INERCIA EQUIVALENTE, CONSIDERA TRES TIPO DE MASAS: DISCOS,*)
  (* BRAZOS DE MASA UNIFORME Y MASAS CONCENTRADAS EN ALGUN PUNTO, LA NATU-*)
  (* RALEZA DE LAS CARGAS DEL SISTEMA DEBE DARSE EN FORMA EQUIVALENTE A   *)
  (* ESTA *)

  INERCIA:=M0*SQR(R0)/2+M1*SQR(L1)/3+M2*SQR(L2);

  (* MODIFICA LAS CONSTANTES DEL MOTOR SEGUN LA REDUCCION EMPLEADA *)

  TMAX:=TMAX*R;
  AKVEL:=AKVEL*SQR(R);

  K1:=2*PI/NFASES/RESOLUCION*R;
  K2:=2*PI/NFASES;

  (* ACTUALIZA LA POSICION INICAL DEL MOTOR PARA HACERA COINCIDIR CON UNA *)
  (* POSICION DE EQUILIBRIO *)

  FRACPASO1:=K1*TETATAB[NACTUAL]/2/PI;
  FRACPASO:=(FRACPASO-TRUNC(FRACPASO))/K1*2*PI;

  TETAR:=TETATAB[NACTUAL]-FRACPASO;

  (* INICIALIZA INDICADORES Y AUXILIARES DEL TIPO DE CONMUTACION *)

  CONT:=0;
  OMEGAR:=OMEGATAB[NACTUAL];

  WRITELN('          TIEMPO: ',TIEMPO);
  WRITELN('          NUMDATOS: ',NUMDATOS);
```

```
END;(*INICIALIZA*)

(* CALCULA LAS CONDICIONES DESEADAS *)

(* POSICION DESEADA *)

PROCEDURE POSICION;

BEGIN(*POSICION*)

  TETAD:= TETATAB[NACTUAL]+(OMEGATAB[NACTUAL] +C3[NACTUAL]/C4[NACTUAL])*
          DELTATIE + C1[NACTUAL]*SQR(DELTATIE)/2 + C2[NACTUAL]*
          SQR(DELTATIE)*DELTATIE/6 - C3[NACTUAL]/SQR(C4[NACTUAL])*
          SIN(C4[NACTUAL]);

END;(*POSICION*)

(* VELOCIDAD DESEADA *)

PROCEDURE VELOCIDAD;

BEGIN(*VELOCIDAD*)

  OMEGAD:= OMEGATAB[NACTUAL]+C3[NACTUAL]/C4[NACTUAL]+C1[NACTUAL]*DELTATIE+
           C2[NACTUAL]*SQR(DELTATIE)/2-C3[NACTUAL]/C4[NACTUAL]*
           SIN(C4[NACTUAL]*DELTATIE);

END;(*VELOCIDAD*)

(* ACELERACION DESEADA *)

PROCEDURE ACELERACION;

BEGIN(*ACELERACION*)

  ALFAD:=C1[NACTUAL]+C2[NACTUAL]+C3[NACTUAL]*
         SIN(C4[NACTUAL]*DELTATIE);

END;(*ACELERACION*)

(* CONDICIONES CINEMATICAS DESEADAS *)

PROCEDURE VALDESEADOS;

BEGIN(*VALDESEADOS*)

  DELTATIE:=TIEMPO-TIEMTAB[NACTUAL];(*DEFINE EL TIEMPO CON RELACION AL *)
                                    (*INICIO DE INTERVALO*)
  POSICION;
  VELOCIDAD;
  ACELERACION;

END;(*VALDESEADOS*)

(* CALCULA EL DISPONIBLE EN FUNCION DE LA POSICION Y VELOCIDAD *)
(* ANGULAR CORRIENTE *)

PROCEDURE PARDISPONIBLE(OMEGA,TETA:REAL);

BEGIN(*PARDISPONIBLE*)

  T:=-(TMAX-ABS(OMEGA)*AKVEL)*SIN(K1*TETA-CONT*K2);
  IF ABS(OMEGA*AKVEL) > TMAX THEN T:=0;(*ANULA EL PAR A VELOCIDADES ALTAS *)

END;(*PARDISPONIBLE*)

(* INTEGRACION POR EL METODO DE RUNGE-KUTTA DE SEGUNDO ORDEN CON PARAMETROS *)
(* IGUALES (INTEGRACION TRAPEZOIDAL) .*)

PROCEDURE F1(OMEGA:REAL;VAR AK: REAL);

BEGIN(*F1*)

  AK:=OMEGA;

END;(*F1*)

PROCEDURE F2(OMEGA,TETA: REAL;VAR AK: REAL);

BEGIN(*F2*)

  PARDISPONIBLE(OMEGA,TETA);

  AK:=(T-B*OMEGA)/INERCIA;

END;(*F2*)

PROCEDURE RUNGE;

BEGIN(*RUNGE*)

  F1(OMEGAR,AK11);
  F2(OMEGAR,TETAR,AK12);

  TETA1:=TETAR+H*AK11;(*ESTIMACION DE LOS VALORE DE TETA Y OMEGA *)
  OMEGA1:=OMEGAR+H*AK12;

  F1(OMEGA1,AK21);
  F2(OMEGA1,TETA1,AK22);

  TETAR:=TETAR+H/2*(AK11+AK21);(* VALORES FINALES DE TETA Y OMEGA*)
  OMEGAR:=OMEGAR+H/2*(AK12+AK22);

END;(*RUNGE*)

(* LEE UN NUEVO TIEMPO DE CONMUTACION Y SU TIPO *)

PROCEDURE CAMBIOS;

BEGIN(*CAMBIOS*)

  TCONMUTACION:=AE3^:GET(AE3);(*LEE TIEMPO*)
  TIPO:=TRUNC(AE3^);GET(AE3);(*LEE TIPO*)

END;(*CAMBIOS*)

(* TOMA MUESTRAS DE LOS RESULTADOS DE LA SIMULACION Y LOS IMPRIME *)

PROCEDURE GRAFICA;
```

```
BEGIN(*GRAFICA*)

  WRITELN('GRAFICA');

  ASI^:=TIEMPO;
  PUT(ASI);
  ASI^:=TETAD;
  PUT(ASI);
  ASI^:=TETAR;
  PUT(ASI);
  ASI^:=OMEGAD;
  PUT(ASI);
  ASI^:=OMEGAR;
  PUT(ASI);
  MUESTRAS:=MUESTRAS+1;(* ACTUALIZA INDICADOR DE MUESTAS TOMADAS *)

END;(*GRAFICA*)

(* REALIZA UNA CONMUTACION HACIA ADELANTE *)

PROCEDURE INYECTAPOS;

BEGIN(*INYECTAPOS*)

  WRITELN('        INYECTAPOS');

  CONT:=CONT+1;(*ACTUALIZA INDICADOR*)

END;(*INYECTAPOS*)

(* REALIZA UNA CONMUTACON HACIA ATRAS *)

PROCEDURE INYECTANEG;

BEGIN(*INYECTANEG*)

  WRITELN('        INYECTANEG');

  CONT:=CONT-1;(*ACTUALIZA INDICADOR*);

END;(*INYECTANEG*)

(*--------------------------------*)
(* INICIA EL PROGRAMA PRINCIPAL *)
(*--------------------------------*)

BEGIN(*SIMULA*)

  LECTURA;
  INICIALIZA;
  (*ABRE ARCHIVOS DE DATOS Y REPOSICIONA APUNTADORES *)
  (*LOS ARCHIVOS DEBEN EXISTIR EN DISCO              *)


  RESET(ASI,'B5:PLANOS.DATA');
  SEEK(ASI,0);

  (*TIENA MIENTRAS NO SE AGOTE EL TIEMPO DE SIMULACION*)
  WHILE TIEMPO < TIEMTAB[NUMDATOS]/REDTIE DO
```

```
BEGIN(*WHILE TIEMPO*)

  IF NDESEADAS >= DESEADAS(*CALCULO DE CONDICIONES DESEADAS*)

    THEN
      BEGIN(*THEN NDESEDAS*)

        VALDESEADOS;
        WRITELN('DIF POS: ',TETAD-TETAR);
        WRITELN('DIF VEL: ',OMEGAD-OMEGAR);
        WRITELN('');
        NDESEADAS:=0;

      END;(*THEN NDESEADAS*)

  NDESEADAS:=NDESEADAS+1;(*INDICADOR DE CALCULO DE COND. DESEADAS*)

  RUNGE;(*INTEGRACION NUMERICA*)
  TIEMPO:=TIEMPO+H;(*ACTUALIZA TIEMPO*)

  (*VERIFICA SI ES TIEMPO DE ENVIAR UN PULSO*)
  IF TIEMPO >=TCONMUTACION

    THEN

      BEGIN(*THEN TIEMPO*)

        IF TIPO >= 0

          THEN(*TIPO*) INYECTAPOS
          ELSE(*TIPO*) INYECTANEG;

        CAMBIOS;(*LEE NUEVO TIEMPO Y TIPO DE CONMUTACION*)

      END;(*THEN TIEMPO*)

  (*ACTUALIZA APUNTADOR A TABLAS*)
  IF TIEMPO > TIEMTAB[NACTUAL+1] THEN NACTUAL:=NACTUAL+1;

  WRITELN('TIEMPO: ',TIEMTAB[NUMDATOS]-TIEMPO);

  IF CONTIMPRESION=IMPRESION(*TOMA MUESTRAS DE RESULTADOS*)

    THEN

      BEGIN(*THEN CONTIMP*)

        GRAFICA;
        CONTIMPRESION:=0;

      END(*THEN CONTIMP*)

    ELSE

      (*ACTULIZA INDICADOR PARA TOMA DE MUESTRAS*)
      CONTIMPRESION:=CONTIMPRESION+1;

END;(*WHILE TIEMPO*)
```

```
  (*IMPRIME NO. DE REGISTROS*)
  WRITELN('NO. REG EN PLANOS: ',MUESTRAS);

  CLOSE(AS1);(*CIERRA ARCHIVOS*)
  CLOSE(AE3);

  REWRITE(AS1,'#5:NUMSENALES.DATA');
  AS1^:=0;PUT(AS1);(*GUARDA NO. DE PASOS*)
  AS1^:=MUESTRAS;PUT(AS1);(*GUARDA NO. DE MUESTRAS*)
  CLOSE(AS1,LOCK);

END.(*SIMULA*)
```

```
(*CURVAS.TEXT*)
(* PROGRAMA PARA GENERAR EL ARCHIVO CURVAS.DATA    *)
(* ESTE ARCHIVO DEFINE LOS PERFILES DEL MOVIMIENTO *)
(* DESEADO EN EL MOTOR DE PASOS.                   *)
(*                                                 *)
(* LAS CURVAS DE ACELERACION SON DE LA FORMA:      *)
(*                                                 *)
(*    ALFA=C1+C2*T+C3*SEN(C4*T)                    *)
(*                                                 *)
(* LAS TRAYECTORIAS DESEADAS SE PARTEN EN INTERVA- *)
(* LOS DE TIEMPO. LOS VALORES DE C1 A C4 SON VALI- *)
(* DOS DURANTE TODO UN INTERVALO.                  *)
(* ADICIONALMENTE Y PARA EVITAR ERRORES NUMERICOS  *)
(* SE INCLUYEN LA VELOCIDAD Y POSICION ANGULAR, EL *)
(* PASO PARA LA INTEGRACION NUMERICA Y EL TIEMPO   *)
(* EN QUE COMIENZA EL INTERVALO.                   *)
(* EL NUMERO DE JUEGOS DE DATOS SE ENCUENTRA EN EL *)
(* PRIMER REGISTRO DEL ARCHIVO.                    *)
(*                                                 *)
(*              INSTITUTO DE INGENIERIA            *)
(*              PROYECTO          2136             *)
(*                                                 *)
(*              LUIS ALVAREZ ICAZA LONGORIA        *)
(*              OCTUBRE  1983                      *)

CONST   NMAX = 100;(*NO. MAXIMO DE PUNTOS EN EL ARCHIVO*)

(* ZONA DE TIPOS *)

TYPE  DATOS = RECORD (*REGISTRO CON LA ESTRUCTURA DE LOS DATOS*)

        C1,C2,C3,C4,  (*CONSTANTES PARA DEFINIR ACELERACION*)
        TETA,         (*POSICION ANGULAR*)
        OMEGA,        (*VELOCIDAD ANGULAR*)
        H,            (*PASO DE INTEGRACION*)
        TIEMPO        (*TIEMPO DE INICIO DEL INTERVALO*)
              :REAL
      END;(*DATOS*)

(* ZONA DE VARIABLES *)

VAR     ARREGLO: ARRAY[1..NMAX] OF DATOS;(*ARREGLO PARA LOS DATOS*)

        NREG,(*NO. DE PUNTOS DESEADO*)
        I: INTEGER;(*AUXILIAR PARA ITERACIONES*)

        R,S: CHAR;(*AUXILIARES PARA PREGUNTAS*)

        DAT: FILE OF REAL;(*ARCHIVO DE DATOS*)

(*PROCEDIMIENTO DE LECTURA*)

PROCEDURE LECTURA;

BEGIN(*LECTURA*)

 WITH ARREGLO[1] DO

  BEGIN(*WITH*)
```

```
    C1:=DAT^;GET(DAT);(*SALVA DATOS EN ARREGLO*)
    C2:=DAT^;GET(DAT);
    C3:=DAT^;GET(DAT);
    C4:=DAT^;GET(DAT);
    OMEGA:=DAT^;GET(DAT);
    TETA:=DAT^;GET(DAT);
    H:=DAT^;GET(DAT);
    TIEMPO:=DAT^;GET(DAT);

   END;(*WITH*)

END;(*LECTURA*)

(*PROCEDIMIENTO DE ESCRITURA*)

PROCEDURE ESCRIBE;

BEGIN(*ESCRIBE*)

 WITH ARREGLO[I] DO

  BEGIN(*WITH*)

   DAT^:=C1;PUT(DAT);(*SALVA DATOS EN ARREGLO*)
   DAT^:=C2;PUT(DAT);
   DAT^:=C3;PUT(DAT);
   DAT^:=C4;PUT(DAT);
   DAT^:=OMEGA;PUT(DAT);
   DAT^:=TETA;PUT(DAT);
   DAT^:=H;PUT(DAT);
   DAT^:=TIEMPO;PUT(DAT);

  END;(*WITH*)

END;(*ESCRIBE*)

(*PROCEDIMIENTO PARA PEDIR JUEGOS DE DATOS POR CONSOLA*)

PROCEDURE PEDIDO;

BEGIN(*PEDIDO*)

 WITH ARREGLO[I] DO

  BEGIN(*WITH*)

   WRITE('C1[',I,']= ');    READLN(C1);
   WRITE('C2[',I,']= ');    READLN(C2);
   WRITE('C3[',I,']= ');    READLN(C3);
   WRITE('C4[',I,']= ');    READLN(C4);
   WRITE('TETA[',I,']= ');  READLN(TETA);
   WRITE('OMEGA[',I,']= '); READLN(OMEGA);
   WRITE('H[',I,']= ');     READLN(H);
   WRITE('TIEMPO[',I,']= ');READLN(TIEMPO);

  END;(*WITH*)

END;(*PEDIDO*)

BEGIN(*CURVAS*)

 WRITE('ARCHIVO N(UEVO O M)ODIFICACION: ');READ(R);WRITELN;
 (*TIPO DE MODIFICACION*)

 IF R = 'N'
  THEN
   BEGIN(*THEN R*)
    REWRITE(DAT,'#5:CURVAS.DATA');(*ABRE ARCHIVO NUEVO*)
    WRITE('CUANTOS JUEGOS DE DATOS: ');READLN(NREG);
    FOR I:=1 TO NREG DO PEDIDO;(*PIDE DATOS*)
   END(*THEN R*)
  ELSE
   BEGIN(*ELSE R*)
    RESET(DAT,'#5:CURVAS.DATA');
    NREG:=TRUNC(DAT^);GET(DAT);(*ABRE ARCHIVO Y OBTIENE NO. REGISTROS*)
    FOR I:=1 TO NREG DO LECTURA;(*LEE JUEGOS DE DATOS EN EL ARCHIVO*)
    WRITE('MODIFICA EL NO. DE JUEGOS(S/N): ');READ(S);WRITELN;
    IF S = 'S'
     THEN BEGIN(*THEN S*) WRITE('CUANTOS JUEGOS: ');READLN(NREG);END;(*THEN*)
    CLOSE(DAT);(*CIERRA ARCHIVO Y ABRE NUEVO PARA LECTURA*)
    REWRITE(DAT,'#5:CURVAS.DATA');
   END;(*ELSE R*)

 REPEAT (*UNTIL S<>'S'*)

  WRITE('MODIFICA ALGUN JUEGO DE DATOS(S/N): ');READ(S);WRITELN;
  (*PREGUNTA POR MODIFICACIONES*)
  IF S = 'S' THEN
   BEGIN(*THEN S*)
    WRITE('QUE NO. DE JUEGO(1 A ',NREG,')');READLN(I);
    IF ((I<=0) OR (I>NREG))
     THEN(*1*)WRITELN('NUMERO INVALIDO!!!!.REPITA')
     ELSE(*1*)PEDIDO;
   END;(*THEN S*)

 UNTIL S <> 'S';(*FIN REPEAT*)

 DAT^:=NREG;PUT(DAT);(*GUARDA EL NO. DE REGISTROS*)

 FOR I:=1 TO NREG DO ESCRIBE;(*ESCRIBE DATOS*)

 CLOSE(DAT,LOCK);(*SALVA ARCHIVO*)

END.(*CURVAS*)
```

```
(*PARAMETRO.TEXT*)
(* PROGRAMA PARA GENERAR EL ARCHIVO PARAMETROS.DATA , ESTE ARCHIVO *)
(* CONTIENE LOS PARAMETROS NECESARIOS PARA LA SIMULACION DEL COM- *)
(* PORTAMIENTO DEL MOTOR DE PASOS.                                *)
(* EL ARCHIVO CONTIENE CINCO REGISTROS CON LOS SIGUIENTES DATOS:  *)
(*  PRIMER REGISTRO:                                              *)
(*      NFASES=NO. DE FASES DEL MOTOR DE PASOS                    *)
(*      RESOLUCION=VALOR EN RADIANES DE UN PASO DEL MOTOR         *)
(*      TMAX=PAR MAXIMO DISPONIBLE                                *)
(*      A*VEL=CONSTANTE EN QUE DECRECE EL PAR SEGUN LA VELOCIDAD  *)
(*      R=RELACION DE REDUCCION EMPLEADA                          *)
(*  SEGUNDO REGISTRO:                                             *)
(*      TOL=TOLERANCIA PARA CONMUTACIONES                         *)
(*      KP=CONSTANTE DE REALIMENTACION DEL ERROR DE POSICION      *)
(*      KV=IDEM PARA VELOCIDAD                                    *)
(*  TERCER REGISTRO:                                              *)
(*      FRICCION=COEFICIENTE DE FRICCION DE COULOMB               *)
(*      D=COEFICIENTE DE FRICCION VISCOSA                         *)
(*  CUARTO REGISTRO:                                              *)
(*      RO=RADIO DEL DISCO DE INERCIA EQUIVALENTE                 *)
(*      MO=MASA     "    "    "    "        "                     *)
(*      L1=LONGITUD DEL BRAZO EQUIVALENTE                         *)
(*      M1=MASA     "    "    "                                   *)
(*      L2=LONGITUD A LA QUE SE HAYA LA MASA CONCENTRADA          *)
(*      M2=MASA DE LA MISMA                                       *)
(*  QUINTO REGISTRO:                                              *)
(*      IMPRESION=FRECUENCIA DE ESCRITURA EN EL ARCHIVO DE SALIDA *)
(*      DESEADAS=FRECUENCIA DE INTERPOLACION DE COND. DESEADAS    *)
(*      NMAX=NO. MAX. DE ITERACIONES SIN CALCULO DE CONDICIONES   *)
(*           DE CONMUTACION                                       *)
(*(ESTE REGISTRO SE CONVIERTE A VARIABLES ENTERAS AL LEERSE)      *)
(*                                                               *)
(*              INSTITUTO DE INGENIERIA                          *)
(*              PROYECTO          2136                           *)
(*                                                               *)
(*              LUIS ALVAREZ ICAZA LONGORIA                      *)
(*              OCTUBRE  1983                                    *)

CONST   NMAX =  19;(*NO. MAXIMO DE PUNTOS EN EL ARCHIVO*)

(* ZONA DE VARIABLES *)

VAR     DATO: ARRAY[1..NMAX] OF REAL; (*ARREGLO PARA LOS DATOS*)

        I: INTEGER;(*AUXILIAR PARA ITERACIONES*)

        R,S: CHAR;(*AUXILIARES PARA PREGUNTAS*)

        DAT: INTERACTIVE ;(*ARCHIVO DE DATOS*)

(*PROCEDIMIENTO DE LECTURA*)

PROCEDURE LECTURA;

BEGIN(*LECTURA*)

   READLN(DAT,DATO[1],DATO[2],DATO[3],DATO[4],DATO[5]);
   READLN(DAT,DATO[6],DATO[7],DATO[8]);
   READLN(DAT,DATO[9],DATO[10]);
   READLN(DAT,DATO[11],DATO[12],DATO[13],DATO[14],DATO[15],DATO[16]);
   READLN(DAT,DATO[17],DATO[18],DATO[19]);

END;(*LECTURA*)

(*PROCEDIMIENTO DE ESCRITURA*)

PROCEDURE ESCRIBE;

BEGIN(*ESCRIBE*)

   WRITELN(DAT,DATO[1],DATO[2],DATO[3],DATO[4],DATO[5]);
   WRITELN(DAT,DATO[6],DATO[7],DATO[8]);
   WRITELN(DAT,DATO[9],DATO[10]);
   WRITELN(DAT,DATO[11],DATO[12],DATO[13],DATO[14],DATO[15],DATO[16]);
   WRITELN(DAT,DATO[17],DATO[18],DATO[19]);

END;(*ESCRIBE*)

(*PROCEDIMIENTO PARA PEDIR JUEGOS DE DATOS POR CONSOLA*)

PROCEDURE PEDIDO;

BEGIN(*PEDIDO*)

 CASE I OF (*LEE DATO*)

  1:  BEGIN(*1*)  WRITE('NO. DE FASES: ');READLN(DATO[1]); END;(*1*)
  2:  BEGIN(*2*)  WRITE('RESOLUCION:   ');READLN(DATO[2]); END;(*2*)
  3:  BEGIN(*3*)  WRITE('PAR MAXIMO:   ');READLN(DATO[3]); END;(*3*)
  4:  BEGIN(*4*)  WRITE('CTE. VELOC.:  ');READLN(DATO[4]); END;(*4*)
  5:  BEGIN(*5*)  WRITE('REDUCCION:    ');READLN(DATO[5]); END;(*5*)
  6:  BEGIN(*6*)  WRITE('TOL. CONMUTA.:');READLN(DATO[6]); END;(*6*)
  7:  BEGIN(*7*)  WRITE('CTE. POSIC.:  ');READLN(DATO[7]); END;(*7*)
  8:  BEGIN(*8*)  WRITE('CTE. VELOC.:  ');READLN(DATO[8]); END;(*8*)
  9:  BEGIN(*9*)  WRITE('FRICCION COUL:');READLN(DATO[9]); END;(*9*)
  10: BEGIN(*10*) WRITE('VISCOSA:      ');READLN(DATO[10]);END;(*10*)
  11: BEGIN(*11*) WRITE('RADIO DISCO:  ');READLN(DATO[11]);END;(*11*)
  12: BEGIN(*12*) WRITE('MASA DISCO:   ');READLN(DATO[12]);END;(*12*)
  13: BEGIN(*13*) WRITE('LONG. BRAZO:  ');READLN(DATO[13]);END;(*13*)
  14: BEGIN(*14*) WRITE('MASA BRAZO:   ');READLN(DATO[14]);END;(*14*)
  15: BEGIN(*15*) WRITE('LONG. MASA:   ');READLN(DATO[15]);END;(*15*)
  16: BEGIN(*16*) WRITE('MASA CONCEN.: ');READLN(DATO[16]);END;(*16*)
  17: BEGIN(*17*) WRITE('FREC. IMPRES.:');READLN(DATO[17]);END;(*17*)
  18: BEGIN(*18*) WRITE('FREC. DESEA.: ');READLN(DATO[18]);END;(*18*)
  19: BEGIN(*19*) WRITE('ITER. CALCULO:');READLN(DATO[19]);END;(*19*)

 END;(*CASE I*)

END;(*PEDIDO*)

(*PROCEDIMIENTO PARA DESPLEGAR LOS DATOS POR CONSOLA*)

PROCEDURE DESPLIEGA;

BEGIN(*DESPLIEGA*)

 CASE I OF (*LEE DATO*)

  1:  BEGIN(*1*)  WRITE('NO. DE FASES: ');WRITELN(DATO[1]); END;(*1*)
  2:  BEGIN(*2*)  WRITE('RESOLUCION:   ');WRITELN(DATO[2]); END;(*2*)
```

```
  3: BEGIN(*3*) WRITE('PAR MAXIMO:   ');WRITELN(DATO[3]); END;(*3*)
  4: BEGIN(*4*) WRITE('CTE. VELOC.:  ');WRITELN(DATO[4]); END;(*4*)
  5: BEGIN(*5*) WRITE('REDUCCION:    ');WRITELN(DATO[5]); END;(*5*)
  6: BEGIN(*6*) WRITE('TOL. CONMUTA.:');WRITELN(DATO[6]); END;(*6*)
  7: BEGIN(*7*) WRITE('CTE. POSIC.:  ');WRITELN(DATO[7]); END;(*7*)
  8: BEGIN(*8*) WRITE('CTE. VELOC.:  ');WRITELN(DATO[8]); END;(*8*)
  9: BEGIN(*9*) WRITE('FRICCION COUL:');WRITELN(DATO[9]); END;(*9*)
 10: BEGIN(*10*) WRITE('VISCOSA:      ');WRITELN(DATO[10]);END;(*10*)
 11: BEGIN(*11*) WRITE('RADIO DISCO:  ');WRITELN(DATO[11]);END;(*11*)
 12: BEGIN(*12*) WRITE('MASA DISCO:   ');WRITELN(DATO[12]);END;(*12*)
 13: BEGIN(*13*) WRITE('LONG. BRAZO:  ');WRITELN(DATO[13]);END;(*13*)
 14: BEGIN(*14*) WRITE('MASA BRAZO:   ');WRITELN(DATO[14]);END;(*14*)
 15: BEGIN(*15*) WRITE('LONG. MASA:   ');WRITELN(DATO[15]);END;(*15*)
 16: BEGIN(*16*) WRITE('MASA CONCEN.: ');WRITELN(DATO[16]);END;(*16*)
 17: BEGIN(*17*) WRITE('FREC. IMPRES.:');WRITELN(DATO[17]);END;(*17*)
 18: BEGIN(*18*) WRITE('FREC. DESEA.: ');WRITELN(DATO[18]);END;(*18*)
 19: BEGIN(*19*) WRITE('ITER. CALCULO:');WRITELN(DATO[19]);END;(*19*)

 END;(*CASE I*)

END;(*DESPLIEGA*)

BEGIN(*PARAMETROS*)

 WRITE('ARCHIVO N)UEVO O M)ODIFICACION: ');READ(R);WRITELN
 (*TIPO DE MODIFICACION*)

 IF R = 'N'
  THEN
   BEGIN(*THEN R*)
    REWRITE(DAT,'#5:PARAMETROS.DATA');(*ABRE ARCHIVO NUEVO*)
    FOR I:=1 TO NMAX DO PEDIDO;(*PIDE DATOS*)
   END(*THEN R*)
  ELSE
   BEGIN(*ELSE R*)
    RESET(DAT,'#5:PARAMETROS.DATA');(*ABRE ARCHIVO*)
    LECTURA;(*LEE DATOS EN EL ARCHIVO*)
    CLOSE(DAT);(*CIERRA ARCHIVO Y ABRE NUEVO PARA LECTURA*)
    REWRITE(DAT,'#5:PARAMETROS.DATA');
   END;(*ELSE R*)

  REPEAT (*UNTIL S<>'S'*)

   WRITE('MODIFICA ALGUN DATO(S/N): ');READ(S);WRITELN;
   (*PREGUNTA POR MODIFICACIONES*)
   IF S = 'S' THEN
    BEGIN(*THEN S*)
     WRITE('QUE NO. DE DATO(1 A ',NMAX,')');READLN(I);
     IF ((I<=0) OR (I>NMAX))
      THEN(*I*)WRITELN('NUMERO INVALIDO!!!,REPITA')
      ELSE
       BEGIN(*ELSE I*)
        WRITELN('VALOR ACTUAL');
        DESPLIEGA;
        WRITELN('NUEVO VALOR');
        PEDIDO;
       END;(*ELSE I*)
    END;(*THEN S*)

  UNTIL S <> 'S';(*FIN REPEAT*)

  ESCRIBE;(*ESCRIBE DATOS*)

  CLOSE(DAT,LOCK)(*SALVA ARCHIVO*)

END.(*PARAMETROS*)
```

```
(*SERVICIO.TEXT*)
{*-------------------------------------------------------*}
{*                                                       *}
{*  PROGRAMA DE SERVICIO PARA EL DESPLIGUE O IMPRESION   *}
{*  DE ARCHIVOS DE LOS SIGUIENTES TIPOS: INTERACTIVOS,   *}
{*  DE TEXTO, REALES O ENTEROS. SE PERMITE ESCOGER EL    *}
{*  ARCHIVO DE ENTRADA Y EL DE SALIDA. SE PUEDE FIJAR    *}
{*  TANTO EL NUMERO DE REGISTROS A LEER COMO EL REGIS-   *}
{*  TRO INICIAL.                                         *}
{*                                                       *}
{*      LUIS ALVAREZ ICAZA LONGORIA   PROYECTO 2136      *}
{*      INSTITUTO DE INGENIERIA              UNAM        *}
{*                                                       *}
{*-------------------------------------------------------*}

PROGRAM SERVICIO;

(*DECLARACION DE VARIABLES*)

VAR
        SALIDA,(*NOMBRE DE LA UNIDAD DE SALIDA*)
        DATOS  (*  "    DEL ARCHIVO DE DATOS  *)
            :STRING[40];

        R: CHAR;(*AUXILIAR PARA PREGUNTAS*)

        INT,SAL: INTERACTIVE;(*APUNTADORES A ARCHIVOS INTERACTIVOS*)
        TEX:     TEXT       ;(*APUNTADOR   "     "   DE TEXTO     *)
        FE:      FILE OF REAL;(*    "       "     "   REALES       *)
        ENT: FILE OF INTEGER;(*    "       "     "   ENTEROS      *)

        I,   (*AUXILIAR PARA ITERACIONES*)
        CREG,(*CONTADOR DE REGISTROS*)
        NREG,(*NO. DE REGISTROS A LEER*)
        REGIN,(*REGISTRO DESDE EL QUE SE INICIA LA LECTURA*)
        IMP,  (*NO. DE IMPRESIONES POR RENGLON A LA SALIDA*)
        PAUSA (*INDICADOR DE PAUSA PARA LA SALIDA*)
            :INTEGER;

        MUDA: STRING[80];(*VARIABLE PARA LECTURA DE TEXTO O INTERACTIVOS*)

(*PROCEDIMIENTO PARA LECTURA DE REALES*)

PROCEDURE LEEREAL;

BEGIN(*LEEREAL*)

  GET(RE);(*LEE REAL*)
  IF EOF(RE) (*PREGUNTA POR FIN DE ARCHIVO*)

   THEN  CREG:=NREG

   ELSE
    BEGIN(*ELSE EOF*)

      IF ((CREG+1) MOD IMP) = 0
       THEN(*CREG+1MOD*) WRITELN(SAL,RE^)(*IMPRIME DATOS*)
       ELSE(*CREG+1MOD*) WRITE  (SAL,RE^,' ');

      IF (CREG+1)MOD(IMP*PAUSA) = 0
       THEN
        BEGIN(*CREG+1)DIV*)
         WRITE('<RETURN> PARA CONTINUAR');
         READLN;
        END;(*CREG+1)DIV*)

    END;(*ELSE EOF*)

END;(*LEEREAL*)

(*PROCEDIMIENTO PARA LECTURA DE ENTEROS*)

PROCEDURE LEEINT;

BEGIN(*LEEINT*)

  GET(ENT);(*LEE ENTERO*)
  IF EOF(ENT) (*PREGUNTA POR FIN DE ARCHIVO*)

   THEN  CREG:=NREG

   ELSE
    BEGIN(*ELSE EOF*)

      IF ((CREG+1) MOD IMP) = 0
       THEN(*CREG+1MOD*) WRITELN(SAL,ENT^)(*IMPRIME DATOS*)
       ELSE(*CREG+1MOD*) WRITE  (SAL,ENT^,' ');

      IF (CREG+1)MOD(IMP*PAUSA) = 0
       THEN
        BEGIN(*CREG+1)DIV*)
         WRITE('<RETURN> PARA CONTINUAR');
         READLN;
        END;(*CREG+1)DIV*)

    END;(*ELSE EOF*)

END;(*LEEINT*)

(*PROCEDIMIENTO PARA LECTURA DE ARCHIVOS INTERACTIVOS*)

PROCEDURE REGISTROS;

BEGIN(*REGISTROS*)

  READLN(INT,MUDA);(*LEE REGISTRO*)
  IF EOF(INT)
   THEN CREG:=NREG
   ELSE
     BEGIN
       WRITELN(SAL,MUDA);(*IMPRIME REGISTRO*)
       IF((CREG+1) MOD PAUSA) = 0 THEN
        BEGIN
         WRITE('<RETURN> PARA CONTINUAR');READLN;(*PAUSA POR PAGINA*)
        END;
     END;

END;(*REGISTRO*)

(*PROCEDIMIENTO PARA LECTURA DE ARCHIVOS DE TEXTO*)
```

```
PROCEDURE TEXTUAL;

BEGIN(*TEXTUAL*)

  READLN(TEX,MUDA);(*LEE REGISTRO*)
  IF EOF(TEX)
   THEN CREG:=NREG
   ELSE
     BEGIN
       WRITELN(SAL,MUDA);(*IMPRIME REGISTRO*)
       IF((CREG+1) MOD PAUSA) = 0 THEN
        BEGIN
         WRITE('(RETURN) PARA CONTINUAR');READLN;(*PAUSA POR PAGINA*)
        END;
     END;

END;(*TEXTUAL*)

BEGIN(*SERVICIO*)

 REPEAT (*UNTIL R <> 'S'*)

  REPEAT (*UNTIL R*)

   WRITELN('QUE TIPO DE ARCHIVO DE DATOS');
   WRITE  ('(T(EXTO,I)NTERACTIVO,R)EAL,E)NTERO); ');
   READ(R);WRITELN;(*LEE TIPO DE ARCHIVO*)

  UNTIL R IN['T', 'I','R','E'];

  WRITE('ARCHIVO DE SALIDA: ');
  READLN(DATOS);(*ARCHIVO DE SALIDA*)
  RESET(SAL,DATOS);(*LO ABRE*)

  WRITE('ARCHIVO DE ENTRADA: ');
  READLN(DATOS);(*ARCHIVO DE ENTRADA*)

  WRITE('REGISTROS A LEER: ');
  READLN(NREG);
  WRITE('REGISTRO INICIAL: ');
  READLN(REGIN);REGIN:=REGIN-1;(*REGISTRO INICIAL Y NO. DE ELLOS*)
  WRITE('NO. IMPRESIONES POR LINEA: ');
  READLN(IMP);
  WRITE('TAMAND DE LA PAGINA(EN LINEAS): ');
  READLN(PAUSA);(*FORMATOS PARA SALIDA*)

  CASE R OF (*ABRE ARCHIVO DE ENTRADA*)

   'T': BEGIN(*T*)RESET(TEX,DATOS);
                FOR I:=1 TO REGIN DO READLN(TEX,MUDA);END;(*T*)
   'I': BEGIN(*I*)RESET(INT,DATOS);
                FOR I:=1 TO REGIN DO READLN(INT,MUDA);END;(*I*)
   'R': BEGIN(*R*)RESET(RE, DATOS); SEEK(RE, REGIN);END;(*R*)
   'E': BEGIN(*E*)RESET(ENT,DATOS); SEEK(ENT,REGIN);END;(*E*)

  END;(*CASE R*)

  CREG:=0;(*CONTADOR DE REGISTROS*)

  WHILE (CREG <= NREG) DO

   BEGIN(*WHILE CREG*)

    CASE R OF (*LEE DATOS*)

     'T': TEXTUAL;
     'I': REGISTRO;
     'R': LEEREAL;
     'E': LEEINT;

    END;(*CASE R*)

    CREG:=CREG+1;

   END;(*WHILE CREG*)

   CASE R OF(*CIERRA ARCHIVOS*)

    'T': CLOSE(TEX);
    'I': CLOSE(INT);
    'R': CLOSE(RE);
    'E': CLOSE(ENT);

   END;(*CASE R*)

   CLOSE(SAL);(*CIERRA ARCHIVO DE SALIDA*)
   WRITELN;WRITE('OTRO ARCHIVO(S/N): ');
   READ(R);WRITELN;(*DECIDE SI LEER OTRO ARCHIVO*)

 UNTIL R <> 'S';(*END UNTIL R <> 'S'*)

END.(*SERVICIO*)
```

```
{$I+}
{*GRAFSIM.TEXT*}

PROGRAM GRAFSIM;

{* DECLARA UNIDADES DE LIBRERIA *}

USES TRANSCEND,TURTLEGRAPHICS,GRAFICASXY;

CONST MUESTRA = 5;
      MUESTMAX = 60;

{* ZONA DE VARIABLES *}
VAR  X,Y:ARREGLODATOS;{*ARREGLOS PARA GRAFICACION*}

     XX,YY:ARRAY[1..MUESTMAX]OF REAL;{*MUESTRAS DE VALORES DE DATOS*}

     UNY,{*UNIDADES DE LAS ORDENADAS*}
     ROT:STRING[40];{*ROTULO DE LA GRAFICA*}

     R: CHAR;{*AUXILIAR PARA PREGUNTAS*}

     FACTOR,{*FACTOR PARA LECTURA*}
     ACUM:REAL;{*AUXILIAR PARA LECTURA*}

     I,L,{*AUXILIARES PARA LECTURA*}
     REGIN,{*REGISTRO INICIAL*}
     REGFIN:INTEGER;

{* PROCEDIMIENTO PARA LEER LA CURVA DE CONDICIONES REALES *}

SEGMENT PROCEDURE UNO;

{* DEFINE FORMATO DEL ARCHIVO DE ENTRADA *}

TYPE DATOS = RECORD OMR,OMD,TR,TD,TIE:REAL;END;

{* ARCHIVOS DE ENTRADA *}

VAR  AE1: FILE OF REAL;
     AE2:FILE OF DATOS;
BEGIN
 {*PREGUNTA POR TIPO DE GRAFICA*}

 WRITE('GRAFICA DE V)ELOCIDAD O P)OSICION ');READ(R);WRITELN;

 {*OBTIENE NO. DE REGISTROS EN EL ARCHIVO*}
 RESET(AE1,'#5:NUMSENALES.DATA');
 GET(AE1);I:=TRUNC(AE1^);
 WRITELN('EXISTEN: ',I, ' REGISTROS');
 CLOSE(AE1);

 {*PREGUNTA POR REGITROS INICIALES Y FINALES*}
 WRITE('REGISTRO INICIAL ');READLN(REGIN);
 WRITE('REGISTRO FINAL   ');READLN(REGFIN);

 {*OBTIENE FACTOR PARA LECTURAS*}
 IF REGFIN-REGIN <= 300 THEN FACTOR:=1 ELSE FACTOR:= 1.0*(REGFIN-REGIN)/300;

 {*LEE DATOS*}
 RESET(AE2,'#5:PLANOS.DATA');
 ACUM:=FACTOR;I:=REGIN-1;SEEK(AE2,I);GET(AE2);
 FOR I:=1 TO TRUNC((REGFIN-REGIN)/FACTOR) DO
  BEGIN
   X^[I]:=AE2^.TIE;
   IF R= 'P' THEN Y^[I]:=AE2^.TR
             ELSE Y^[I]:=AE2^.OMR;

   {*TOMA MUESTRAS DE CONDICIONES DESEADAS*}
   IF (I MOD MUESTRA) = 0
    THEN
     BEGIN
      XX[(I DIV MUESTRA)]:=AE2^.TIE;
      IF R = 'P' THEN YY[(I DIV MUESTRA)]:=AE2^.TD
                 ELSE YY[(I DIV MUESTRA)]:=AE2^.OMD;
     END;

   {*NO. DEL SIGUIENTE REGISTRO A LEER*}
   ACUM:=ACUM+FACTOR;L:=TRUNC(ACUM)-1;
   SEEK(AE2,L);
   GET(AE2);
  END;

 {*ROT SEGUN ELECCION DE GRAFICA*}
 IF R = 'P' THEN UNY:='RAD' ELSE UNY:='RAD/S';
 IF R = 'P' THEN ROT:='POSICION SIMULADA VS. DESEADA'
            ELSE ROT:='VELOCIDAD SIMULADA VS. DESEADA';
 CLOSE(AE2);

 {*NO. DE PUNTOS A GRAFICAR*}
 L:=TRUNC((REGFIN-REGIN)/FACTOR);

END;

{*PROCEDIMIENTO PARA HALLAR COPIA EN PAPEL*}

PROCEDURE PASAPAPEL(TAMANO: INTEGER);
EXTERNAL;

BEGIN{*GRAFICA*}

 {*ABRE ESPACIO PARA ARREGLOS DE GRAFICACION*}
 NEW(X);NEW(Y);

 {*LEE DATOS*}
 UNO;

 {*CURVA DE CONDICIONES REALES*}
 DEFAULT(X,Y,L,CUR,'S',UNY,ROT);

 {*CURVA DE CONDICIONES DESEADAS*}
 L:=TRUNC(((REGFIN-REGIN) DIV MUESTRA )/FACTOR);

 {*TRASPASA DATOS DE CONDICIONES DESEADAS A ARREGLOS DE GRAFICACION*}
 FOR I:= 1 TO L DO BEGIN X^[I]:=XX[I];Y^[I]:=YY[I];END;
 DIBUJA(X,Y,L,PUN);
 FIJAIMAGEN;

 {*LIBERA ESPACIO DE VARIABLES*}
 RELEASE(Y);RELEASE(X);
```

```
(*PREGUNTA POR COPIA EN PAPEL*)
WRITE('COPIA EN PAPEL(S/N)? ');READ(R);WRITELN;
IF R = 'S' THEN PASAPAPEL(0);

END.(*GRAFICA*)
```

```
(*$N+,$A-,E+,I-*)

        (*EL PROGRAMA INTEGRA LA ECUACION QUE  *)
        (*MODELA LA RESPUESTA DEL MOTOR DE PASO*)
        (*A UN PASO.                           *)
        (*                                     *)
        (*          LUIS ALVARES ICAZA         *)
        (*                                     *)
        (*           PROYECTO  2136            *)
        (*                                     *)
        (*         INSTITUTO DE INGENIERIA     *)
PROGRAM INTEGRA;
USES TRANSCEND;

VAR
      A,B,C,    (*PARAMETROS DE LA INTEGRACION*)
      TET,      (*VALOR ANTERIOR DE LA POSICION*)
      OME,      (*VALOR ANTERIOR DE LA VELOCIDAD*)
      ALF,      (*VALOR DE LA ACELERACION*)
      TET1,     (*VALOR ACTUAL DE LA POSICION*)
      OME1,     (*VALOR ACTUAL DE LA VELOCIDAD*)
      T,        (*TIEMPO TOTAL DE INTEGRACION*)
      TS,       (*TIEMPO ACTUAL DE INTEGRACION*)
      DTE,      (*INCREMENTO EN TIEMPO*)
      NIT,      (*NUMERO DE INTERVALOS*)
      P:REAL;   (*NUMERO DE DATOS AL ARCHIVO*)
      VALORES:FILE OF REAL;
      I,J,K,L:INTEGER;
      APELLE:STRING[20];

FUNCTION SIGN(RY:REAL):INTEGER;  (*FUNCION SIGNO*)
  BEGIN  (*SIGNO*)
    IF RY>0 THEN SIGN:=1
    ELSE IF RY<0 THEN SIGN:=-1
         ELSE SIGN:=0;
  END;   (*SIGNO*)


(* AQUI SE RESUELVE LA ECUACION DIF.*)

BEGIN    (*INTEGRA*)
 J:=0;TS:=0;I:=0; (*INICIALIZA*)

        (*LECTURAS*)
 WRITELN ('PARAMETROS DE LA ECUACION');
 READLN (A,B,C);
 WRITELN ('VALORES INICIALES (OMEGA TETA) ');
 READLN (OME,TET);
 WRITELN ('NUMERO DE INTERVALOS');
 READLN (NIT);
 WRITELN ('NUMERO DE DATOS AL ARCHIVO');
 READLN (L);
 WRITELN ( TIEMPO FINAL');
 READLN (T);
 WRITELN ( ARCHIVO DE SALIDA');
 READLN (APELLE);

        (*FIN DE LECTURAS*)
 DTE:=T/NIT;
 ALF:=A*OME*E-B*SIN(TET)+C*SIGN(OME);
 REWRITE (VALORES,APELLE);
 P:=NIT/L;
 REPEAT
   OME1:=OME+ALF*DTE;
   TET1:=TET+(OME1+OME)*DTE/2;
   IF (J=K*P)    (*ESCOGE LOS DATOS*)
      THEN       (*LOS ESCRIBE*)
         BEGIN(*THEN*)
           VALORES^:=TET1;
           PUT(VALORES);
           VALORES^:=DTE*P;
           PUT(VALORES);
           K:=K+1;
         END;(*THEN*)
        (*CALCULA EL SIGUIENTE VALOR*)
   J:=J+1;
   OME:=OME1;
   TS:=TS+DTE;
   TET:=TET1;
```

```
    ALF:=A*ONE1+E*SIN(TET)+C*SIGN(ONE1);

  UNTIL (TS:=LT*DTE));
  CLOSE(VALORES,LOCK);

END.  (* INTEGRACION *)
```

```
(*$S+*)
PROGRAM MOTOR;
USES TURTLEGRAPHICS,TRANSCEND;
CONST
    AMPLITUD=20;
      BLOQUE=128;
         POR=50;
          PI=3,141592653;
VAR
    DELTATETA,
      NUMERO1,
      NUMERO2,
       BUFFER,
       FACTOR:REAL;
    PRINCIPIO,
    REGISTRO1,
     DESPLAZA,
      COLUMNA,
      CONTROL,
      RENGLON,
      CUANTOS,
      MUEVETE,
       INICIO,
       CUENTA,
       ORIGEN,
       RENACT,
       RENANT,
        DESDE,
        HASTA,
        FINAL,
         CASO:INTEGER;
    NODIBUJES,
     SIGNOANT,
     SIGNOACT,
      BANDERA,
        CRECE:BOOLEAN;
     PREGUNTA,
     CUESTION:CHAR;
      RE: FILE OF REAL;
      ARCHIVO:FILE OF RECORD
                        PAR1,
                       TETA1:REAL;
                        END;
       OFFSET:ARRAY [0..2] OF INTEGER;
         TETA,
          PAR:ARRAY [1..BLOQUE] OF REAL;
            X:0..279;
            Y:0..191;

(* PROCEDURE PARA PRODUCIR UNA COPIA EN PAPEL DE LA GRAFICA *)

PROCEDURE PASAPAPEL;
EXTERNAL;

(* PROCEDURE DE LECTURA DONDE SE PREGUNTA CUAL ES EL ARCHIVO DE DATOS  *)
(* QUE SE TIENE QUE PROCESAR PONIENDOLE UN: (DRIVE) DONDE SE ENCUENTRA)*)
(* TAMBIEN DA EL NUMERO DE REGISTROS QUE CONTIENE EL ARCHIVO Y PREGUNTA*)
(* DESDE DONDE COMIENZA LA LECTURA Y A PARTIR DE AHI CUANTOS REGISTROS *)
(* TIENE QUE PROCESAR, ADEMAS DE ACTUALIZAR VARIABLES Y BANDERAS.      *)
```

```
PROCEDURE LECTURA;
VAR
    NREGISTROS:INTEGER;
       PRIMERO,
       SEGUNDO:REAL;
        NOMBRE:STRING[21];
BEGIN
    WRITELN(CHR(12));
    WRITELN('COMO SE LLAMA TU ARCHIVO DE DATOS ?');
    READLN(NOMBRE);
    RESET(ARCHIVO,NOMBRE);
    RESET(RE,'#5:NUMSENALES.DATA');
    NREGISTROS:=TRUNC(RE^);
    CLOSE(RE);
    SEEK(ARCHIVO,0);
    HASTA:=-1;
    DESDE:=-1;
    WHILE (DESDE < 0) OR (DESDE > NREGISTROS) OR
          (HASTA >= NREGISTROS) OR (HASTA < DESDE) OR
          (FACTOR <= 0.0) DO
          BEGIN
          WRITELN(CHR(12));
          WRITELN('TIENES ',NREGISTROS,' REGISTROS');
          WRITELN('DESDE CUAL EMPIEZO ?');
          READLN(DESDE);
          WRITELN('CUANTOS GRAFICO ?');
          READLN(HASTA);
          DESDE:=DESDE-1;
          HASTA:=DESDE+HASTA-1;
          WRITELN(CHR(12));
          WRITELN('CUANTOS CICLOS POR RENGLON QUIERES ?');
          READLN(FACTOR);
          END;
    WRITELN('QUIERES SOMBREADA LA CURVA (S/N) ?');
    READLN(CUESTION);
    PRINCIPIO:=DESDE;
    INICIO:=DESDE+1;
    FACTOR:=279/12*PI*FACTOR;
    DELTATETA:=1/(FACTOR*POR)*15;
    SEEK(ARCHIVO,DESDE);
    GET(ARCHIVO);
    PRIMERO:=ARCHIVO^.TETA1;
    NUMERO1:=PRIMERO;
    SEEK(ARCHIVO,DESDE+1);
    GET(ARCHIVO);
    SEGUNDO:=ARCHIVO^.TETA1;
    IF (PRIMERO < 0.0) OR (SEGUNDO < 0.0) THEN
       BEGIN
       SIGNOANT:=FALSE;
       ORIGEN:=279;
       END
    ELSE
       BEGIN
       SIGNOANT:=TRUE;
       ORIGEN:=0;
       END;
    IF PRIMERO > SEGUNDO THEN
       BEGIN
       DELTATETA:=-DELTATETA;
       CRECE:=FALSE;
       END
    ELSE
       CRECE:=TRUE;
    SIGNOACT:=SIGNOANT;
    BANDERA:=FALSE;
    OFFSET[0]:=146;
    OFFSET[1]:=83;
    OFFSET[2]:=20;
    CUENTA:=-1;
    RENGLON:=0;
    RENANT:=0;
    RENACT:=0;
    CASO:=1;
    INITTURTLE;
    X:=0;
END;


(* PROCEDURE QUE PONE LOS LETREROS SOBRE CADA RENGLON GRAFICADO Y NOS  *)
(* DICE DE QUE REGISTRO A QUE REGISTRO SE GRAFICARON.                  *)

PROCEDURE LETRERO;
VAR
        ST1,
        ST2:STRING;
    MIENTRAS:REAL;
BEGIN
    NUMERO2:=TETA1[REGISTRO1-CASO]/POR;
    MIENTRAS:=NUMERO2+DELTATETA;
    FINAL:=CUENTA*BLOQUE+REGISTRO1+PRINCIPIO-CASO-CUENTA;
    STR(INICIO,ST2);
    ST1:=CONCAT('REGISTRO ',ST2,' AL ');
    STR(FINAL,ST2);
    ST1:=CONCAT(ST1,ST2);
    STR(ABS(TRUNC(NUMERO1)),ST2);
    ST1:=CONCAT(ST1,'(');
    IF NUMERO1 < 0.0 THEN
       ST1:=CONCAT(ST1,'-',ST2,',')
    ELSE
       ST1:=CONCAT(ST1,ST2,',');
    NUMERO1:=ABS(NUMERO1-TRUNC(NUMERO1))*10000;
    STR(TRUNC(NUMERO1),ST2);
    ST1:=CONCAT(ST1,ST2,'°');
    STR(ABS(TRUNC(NUMERO2)),ST2);
    IF NUMERO2 < 0.0 THEN
       ST1:=CONCAT(ST1,'-',ST2,',')
    ELSE
       ST1:=CONCAT(ST1,ST2,',');
    NUMERO2:=ABS(NUMERO2-TRUNC(NUMERO2))*10000;
    STR(TRUNC(NUMERO2),ST2);
    ST1:=CONCAT(ST1,ST2,')');
    ST1:=CONCAT(ST1,'RAD.');
    MOVETO(0,OFFSET[RENGLON MOD 3]+30);
    PENCOLOR(WHITE);
    WSTRING(ST1);
    PENCOLOR(NONE);
    INICIO:=FINAL+1;
    NUMERO1:=MIENTRAS;
END;
```

```
(* PROCEDURE QUE PONE UNA FLECHA IZQUIERDA A LA MITAD DE LA PANTALLA   *)
(* Y A LA ALTURA SEGUN SE LE ESPECIFIQUE EN LA VARIABLE "REN" SALE CON *)
(* LA PLUMA SIN PINTAR PARA PODER CAMBIAR DE RENGLON E IMPRIMIR POSTE- *)
(* RIORMENTE EL LETRERO.                                               *)

PROCEDURE FLECHAIZQ(REN:INTEGER);
VAR
    V1:INTEGER;
BEGIN
    PENCOLOR(NONE);
    V1:=REN+21;
    MOVETO(131,V1);
    PENCOLOR(WHITE);
    WSTRING('<-');
    PENCOLOR(NONE);
    LETRERO;
END;

(* PROCEDURE QUE PONE UNA FLECHA DERECHA  A LA MITAD DE LA PANTALLA    *)
(* Y A LA ALTURA SEGUN SE LE ESPECIFIQUE EN LA VARIABLE "REN" SALE CON *)
(* LA PLUMA SIN PINTAR PARA PODER CAMBIAR DE RENGLON E IMPRIMIR POSTE- *)
(* RIORMENTE EL LETRERO.                                               *)

PROCEDURE FLECHADER(REN:INTEGER);
VAR
    V1:INTEGER;
BEGIN
    PENCOLOR(NONE);
    V1:=REN+21;
    MOVETO(131,V1);
    PENCOLOR(WHITE);
    WSTRING('->');
    PENCOLOR(NONE);
    LETRERO;
END;

(* PROCEDURE QUE IMPRIME LOS EJES X A TRES DIFERENTES ALTURAS SEGUN SE *)
(* LE ESPECIFIQUE EN LA VARIABLE OFFSET ESTE ES ACCIONADO CUANDO COMIEN*)
(* ZA EL PROGRAMA Y CADA VEZ QUE YA IMPRIMIO LA ANTERIOR PANTALLA LLENA*)

PROCEDURE EJESX(REN:INTEGER);
BEGIN
    PENCOLOR(NONE);
    IF REN > 2 THEN
       BEGIN
       PENCOLOR(NONE);
       MOVETO(140,OFFSET[0]);
       EXIT(EJESX);
       END;
    MOVETO(0,OFFSET[REN]);
    PENCOLOR(WHITE);
    MOVETO(279,OFFSET[REN]);
    EJESX(REN+1);
END;

(* PROCEDURE QUE IMPRIME UNA LINEA VERTICAL SOBRE EL EJE X QUE SE EN - *)
(* CUENTRA Y PONIENDO TANTOS PUNTOS HACIA ARRIBA Y ABAJO COMO SE LE IN-*)
(* DICO EN LA VARIABLE "CUANTOS" ESTO SIRVE PARA PONER MARCAS Y ADEMAS *)
(* PONER EL EJE Y CADA VEZ QUE SE NECESITE.                            *)

PROCEDURE MARCAS(COL,REN,CUANTOS:INTEGER);
VAR
    ARRIBA,
    ABAJO:INTEGER;
BEGIN
    PENCOLOR(NONE);
    ARRIBA:=REN+CUANTOS;
    ABAJO:=REN-CUANTOS;
    MOVETO(COL,ARRIBA);
    PENCOLOR(WHITE);
    MOVETO(COL,ABAJO);
    MOVETO(COL,REN);
    PENCOLOR(NONE);
END;

(* PROCEDURE QUE SIRVE PARA IR TOMANDO DE BLOQUE EN BLOQUE DATOS DEL   *)
(* DISCO VACIANDOLE ESTOS SOBRE LOS VECTORES TETA Y PAR ESTO ES PARA   *)
(* QUE SE HAGA EL PROCESO MAS RAPIDO Y NO ACCESAR TAN SEGUIDO EL DISCO *)

PROCEDURE CACHE;
BEGIN
    SEEK(ARCHIVO,DESDE);
    GET(ARCHIVO);
    CUANTOS:=1;
    WHILE (DESDE <= HASTA) AND (CUANTOS <= BLOQUE) DO
          BEGIN
          TETA[CUANTOS]:=ARCHIVO^.TETA1*POR;
          PAR[CUANTOS]:=ARCHIVO^.PAR1;
          CUANTOS:=CUANTOS+1;
          DESDE:=DESDE+1;
          GET(ARCHIVO);
          END;
    CUANTOS:=CUANTOS-1;
    CUENTA:=CUENTA+1;
    DESDE:=DESDE-1;
END;

PROCEDURE PINTA;
BEGIN
    DESPLAZA:=TRUNC(1.0*ROUND(FACTOR*TETA[REGISTRO1]/1.0);
    MUEVETE:=TRUNC(1.0*TETA[REGISTRO1]*180/PI)*10;
    IF DESPLAZA < 0 THEN
       BEGIN
       SIGNOACT:=FALSE;
       ORIGEN:=279;
       CONTROL:=-DESPLAZA;
       X:=ORIGEN-(CONTROL MOD 280);
       RENACT:=CONTROL DIV 280;
       END
    ELSE
       IF DESPLAZA <> 0 THEN
          BEGIN
          SIGNOACT:=TRUE;
          ORIGEN:=0;
          X:=DESPLAZA MOD 200+ ORIGEN;
          RENACT:=DESPLAZA DIV 280;
          END;
    IF REGISTRO1 > 1 THEN
       IF CRECE THEN
          BEGIN
```

```
      IF TETA[REGISTRO1] < TETA[REGISTRO1-1] THEN
         BEGIN
         CASO:=1;
         DELTATETA:=-DELTATETA;
         FLECHADER(OFFSET[RENGLON MOD 3]);
         RENGLON:=RENGLON+1;
         CRECE:=FALSE;
         END;
      END
   ELSE
      IF TETA[REGISTRO1] > TETA[REGISTRO1-1] THEN
         BEGIN
         CASO:=1;
         DELTATETA:=-DELTATETA;
         FLECHAIZQ(OFFSET[RENGLON MOD 3]);
         RENGLON:=RENGLON+1;
         CRECE:=TRUE;
         END;
IF (MUEVETE MOD 450) = 0 THEN (*MARCA CADA 45 GRADOS*)
   MARCAS(X,OFFSET[RENGLON MOD 3],2);
IF RENANT <> RENACT THEN
   BEGIN
   CASO:=0;
   IF CRECE THEN
      FLECHADER(OFFSET[RENGLON MOD 3])
   ELSE
      FLECHAIZQ(OFFSET[RENGLON MOD 3]);
   RENGLON:=RENGLON+1;
   PENCOLOR(NONE);
   RENANT:=RENACT;
   END;
IF SIGNOANT <> SIGNOACT THEN
   BEGIN
   CASO:=1;
   IF CRECE THEN
      FLECHADER(OFFSET[RENGLON MOD 3])
   ELSE
      FLECHAIZQ(OFFSET[RENGLON MOD 3]);
   MARCAS(ABS(ORIGEN-279),OFFSET[RENGLON MOD 3],AMPLITUD);
   RENGLON:=RENGLON+1;
   END;
IF ((RENGLON MOD 3) = 0) AND BANDERA THEN
   BEGIN
   TEXTMODE;
   WRITELN('QUIERES COPIA EN PAPEL (S/N) ?');
   WRITELN('PARA CONTINUAR VIENDO LA GRAFICA');
   WRITELN('OPRIMA LA TECLA <RETURN>');
   READLN;
   GRAFMODE;
   READLN(PREGUNTA);
   IF PREGUNTA='S' THEN
      PASAPAPEL;
   BANDERA:=FALSE;
   INITTURTLE;
   EJES(0);
   END;
IF SIGNOANT <> SIGNOACT THEN
   BEGIN
   CASO:=1;
   MARCAS(ORIGEN,OFFSET[RENGLON MOD 3],AMPLITUD);
      SIGNOANT:=SIGNOACT;
      END;
   IF DESPLAZA = 0 THEN
      MARCAS(ORIGEN,OFFSET[RENGLON MOD 3],AMPLITUD);
   BUFFER:=TETA[REGISTRO1]-PAR[REGISTRO1]*PI/2;
   Y:=TRUNC(-AMPLITUD*SIN(BUFFER)+OFFSET[RENGLON MOD 3]);
   MOVETO(X,Y);
   PENCOLOR(WHITE);
   MOVETO(X,Y);
   IF CUESTION = 'S' THEN
      BEGIN
      Y:=OFFSET[RENGLON MOD 3];
      MOVETO(X,Y);
      END;
   IF ((RENGLON MOD 3) = 2) THEN
      BANDERA:=TRUE;
END;

BEGIN (*PRINCIPAL*)
REPEAT
   LECTURA;
   EJES(X(0));
   MARCAS(ORIGEN,146,AMPLITUD);
   WHILE DESDE < HASTA DO
         BEGIN
         CACHE;
         FOR REGISTRO1:=1 TO CUANTOS-1 DO
            BEGIN
            NODIBUJES:=FALSE;
            REPEAT
              PINTA;
              IF CRECE THEN
                 BEGIN
                 IF (TETA[REGISTRO1] + DELTATETA) >= TETA[REGISTRO1 + 1] THEN
                    NODIBUJES:=TRUE
                 ELSE
                    TETA[REGISTRO1]:=TETA[REGISTRO1] + DELTATETA;
                 END
              ELSE
                 IF (TETA[REGISTRO1] + DELTATETA) <= TETA[REGISTRO1 + 1] THEN
                    NODIBUJES:=TRUE
                 ELSE
                    TETA[REGISTRO1]:=TETA[REGISTRO1] + DELTATETA;
            UNTIL NODIBUJES;
            END;
         END;
   CLOSE(ARCHIVO);
   CASO:=0;
   IF CRECE THEN
      FLECHADER(OFFSET[RENGLON MOD 3])
   ELSE
      FLECHAIZQ(OFFSET[RENGLON MOD 3]);
   MARCAS(ORIGEN,OFFSET[RENGLON MOD 3],AMPLITUD);
   TEXTMODE;
   WRITELN('QUIERES COPIA EN PAPEL (S/N) ?');
   WRITELN('PARA CONTINUAR VIENDO LA GRAFICA');
   WRITELN('OPRIMA LA TECLA <RETURN>');
   READLN;
   GRAFMODE;
   READLN(PREGUNTA);
```

```
    IF PREGUNTA='S' THEN
       PASAPAPEL;
    TEXTMODE;
    WRITELN('QUIERES REGRESAR A LEER OTRO ARCHIVO (S/N) ?');
    READLN(PREGUNTA);
UNTIL PREGUNTA <>'S';
END.
```

ANEXO B

PROGRAMAS PARA CONTROLAR LAS MEDICIONES DEL MOVIMIENTO DE LOS MOTORES DE PASOS

Este anexo contiene los listados de los programas que manejan el banco de prueba de los motores de pasos.

CONTROL: aplica la tabla de tiempos de conmutación a los motores de pasos y mide los resultados obtenidos.

PASO: mide la respuesta del motor de pasos a una sola conmutación.

PRUEBA: permite comprobar el funcionamiento del tacómetro digital.

RELOJ: prueba el reloj que se encarga de enviar los pulsos según la tabla generada en MOTOR.

GRAFREAL: grafica los resultados de los experimentos realizados en CONTROL.

GRAFPASO: grafica los resultados de PASO.

DATOS: genera un archivo con algunos parámetros necesarios para la ejecución de CONTROL.

TIEMPOS: transforma los archivos generados en MOTOR, que contienen números reales, en archivos compatibles con las necesidades de CONTROL, que maneja números enteros.

LINEAL2: realiza un ajuste lineal por mínimos cuadrados de los parámetros del modelo de comportamiento de los motores de pasos, según los resultados de PASO.

PARABOLA: igual que LINEAL2, pero con un ajuste cuadrático.

PASAPAPEL: imprime las gráficas obtenidas en cualquiera de los programas de graficación.

```
(*$CONTROL,TEXT*)
(*************************************************)
(*                                               *)
(*    PROGRAMA PARA TOMAR MEDICIONES DEL COMPORTA-*)
(*    MIENTO DE UN MOTOR DE PASOS CONTROLADO BAJO *)
(*    DISTINTOS TIPOS DE CARGA.                  *)
(*    EL PROGRAMA CONTROLA TANTO DEL TIMER QUE CON-*)
(*    TROLA EL ENVIO DE PULSOS EN LA FORMA QUE INDI-*)
(*    CAN LOS RESULTADOS DE LA SIMULACION COMO LOS *)
(*    DISPOSITIVOS MEDIDORES DE VELOCIDAD, PAR,ETC.*)
(*                                               *)
(*            LUIS ALVAREZ ICAZA LONGORIA        *)
(*            PROYECTO 2136      II.UNAM         *)
(*                                               *)
(*************************************************)

PROGRAM CONTROL;

(*-----------------------------------------------*)
(* ZONA DE DECLARACION DE VARIABLES Y CONSTANTES *)
(*-----------------------------------------------*)

CONST   NREGMAX = 1000;     (* NO. MAXIMO DE PULSOS A ENVIAR *)
        NEVMAX  = 4095;     (*  "     "    " EVENTOS POR MUESTRA *)
        MUESTMAX= 1000;     (*  "     "    " MUESTRAS *)
        CONTO   =32767;     (* MODULO DE CUENTA EN EL TACOMETRO DIGITAL *)
        MUERTO  =  140;     (* TIEMPO MUERTO EN LAS MEDICIONES DE VELOCIDAD*)
        RELAPPLE= 1.0205E6;(*FRECUENCIA DEL RELOJ DE LA APPLE EN HZ *)
        RESOL   =3.14159E-3(*RESOLUCION DEL CODIFICADOR*);

(*--------*)

VAR     MASCARA: PACKED ARRAY [1..NREGMAX] OF 0..255; (* PULSO CODIFICADO *)
        FACTOR : PACKED ARRAY [1..NREGMAX] OF 0..255; (* ESCALA DE TIEMPO *)
        SIGNO  : PACKED ARRAY [1..MUESTMAX] OF 0..255; (* TABLA PARA SIGNO *)
        TIEMPO :        ARRAY [1..NREGMAX] OF INTEGER; (*TIEMPO DEL PULSO *)
        VELBAJ :        ARRAY [1..MUESTMAX] OF INTEGER; (* BPS DE MEDICION *)
        VELALT :        ARRAY [1..MUESTMAX] OF INTEGER; (* BMS  "    "     *)

        I,J, (*INDICES PARA ITERACIONES*)
        NUMREG, (*NO. DE REGISTROS *)
        FRECUENCIA, (* NO. DE EVENTOS POR MEDICION EN 1/10 PASO *)
        MUESTRAS, (* NO. DE MUESTRAS *)
        ERROR  (*INDICADOR DE ERROR EN LA EJECUCION*)

        : INTEGER;

        R.S: CHAR;(* VARIABLES PARA INTERROGAR *)

        TIEANT,(*TIEMPO ANTERIOR*)
        TETANT,(*POSICION ANTERIOR*)
        VELREAL: REAL;(*VARIABLE AUXILIAR PARA EL CALCULO DE VELOCIDADES*)

        AEI: FILE OF INTEGER;(*ARCHIVOS DE ENTRADA*)

        ASI: FILE OF REAL;   (*ARCHIVO DE SALDA*)

        RESPUESTA: SET OF 0..255;

(*--------*)
```

```
(* PROCEDIMIENTO EXTERNO QUE CONTIENE LOS PROGRAMAS PARA     *)
(* CONTROLAR EL TACOMETRO Y EL ACOPLADOR DE ENVIO DE PULSOS *)

PROCEDURE TIEMPOREAL;

EXTERNAL; (* SE DECLARA EXTERNO *)

(*---------*)

(* PROCEDIMIENTO PARA LEER LOS DATOS DE LOS ARCHIVOS EN DISQUETE *)
(* EL ARCHIVO ESTA DIVIDO EN REGISTROS, CADA UNO CONTIENE DOS -  *)
(* VARIABLES ENTERAS. LA PRIMERA INDICA EL TIEMPO ENTRE PULSOS Y *)
(* LA SEGUNDA EL FACTOR DE ESCALAMIENTO DE TIEMPO Y LA MASCARA   *)
(* PARA EL ENVIO DEL PULSO POR EL ACOPLADOR PARALELO.            *)

PROCEDURE LECTURA;

BEGIN (*LECTURA*)

  RESET(AE1,'#5:TIEMPOS.DATA'); (*ABRE ARCHIVO*)
  NUMREG:=AE1^; (*OBTIENE NO. DE REGISTROS DEL ARCHIVO*)

  IF NUMREG < NREGMAX

   THEN

     BEGIN(*THEN NUMREG*)

      FOR I:=1 TO NUMREG DO

       BEGIN(*FOR I*)

        GET(AE1);
        TIEMPO[I]:=AE1^

        GET(AE1);
        J:=AE1^DIV 256;(*TOMA B DMS*)
        J:=J MOD 4;(*TOMA FACTOR DE ESCALA*)
        FACTOR[I]:=J;

        J:=AE1^ MOD 256;(*TOMA B DPS*)
        J:=J MOD 64;(*TOMA MASCARA*)
        MASCARA[I]:=J;

       END;(*FOR I*)

       CLOSE(AE1);(*CIERRA ARCHIVO DATOS*)

     END(*THEN NUMREG*)

   ELSE

     BEGIN(*ELSE NUMREG*)

      WRITELN('DEMASIADOS REGISTROS EN EL ARCHIVO');
      CLOSE(AE1);(*CIERRA ARCHIVO DATOS*)
      EXIT(PROGRAM);(*ABORTA EJECUCION*)

     END;(*ELSE NUMREG*)

END;(*LECTURA*)

(*---------*)

(* PROCEDIMIENTO DE LECTURA DE LOS PARAMETROS PARA LAS MEDICIONES*)

PROCEDURE MEDICION;

BEGIN(*MEDICION*)

  RESET(AE1,'#5:DATOS.DATA');
  FRECUENCIA:=AE1^;(*TOMA NO. EVENTOS POR MEDICION*)

  GET(AE1);
  MUESTRAS:=AE1^;(*TOMA NO. DE MUESTRAS*)

  CLOSE(AE1);(*CIERRA ARCHIVO*)

END;(*MEDICION*)

(*---------*)

(* PROCEDIMIENTO QUE DESPLIEGA LOS DATOS DE LAS MEDICIONES*)

PROCEDURE DESPLIEGA;

BEGIN(*DESPLIEGA*)

  WRITELN('LOS VALORES ACTUALES PARA LAS');
  WRITELN('MEDICIONES SON:');
  WRITELN('');

  WRITELN('PASOS POR MEDICION:',FRECUENCIA DIV 10,'.',
          FRECUENCIA MOD 10);
  WRITELN('NO. DE MUESTRAS : ',MUESTRAS);
  WRITELN('NO. DE REGISTROS: ',NUMREG);

END;(*DESPLIEGA*)

(*---------*)

(*PROCEDIMIENTO PARA MODIFICAR ALGUN PARAMETRO*)

PROCEDURE MODIFICA;

   (*PROCEDIMIENTO LOCAL PARA FACILITAR LOS CAMBIOS*)

   PROCEDURE CORRIGE(N,FACTOR: INTEGER;VAR M:INTEGER);

   VAR A:REAL;

   BEGIN (*CORRIGE*)

     WRITE('VALOR: ');(*PIDE VALOR*)
     READLN(A);(*LEE VALOR CORREGIDO*)
     A:=TRUNC(A*FACTOR);(*LO ESCALA*)

     IF (A<=0) OR (A)N)
```

```
      THEN(*IF A*) WRITELN('VALOR FUERA DE RANGO')
      ELSE(*IF A*) M:=TRUNC(A);

   END;(*CORRIGE*)

LEGIN(*MODIFICA*)

 REPEAT

  BEGIN(*REPEAT R*)

   DESPLIEGA;(*VALORES ACTUALES*)
   WRITE('QUIERES CAMBIAR ALGUNO(S/N)? ');
   READ(R); WRITELN;

   IF R = 'S'

    THEN

     BEGIN(*THEN R*)

      WRITELN('QUE DATO: ');
      WRITELN('NO. EVENTOS/MUESTRA=E');
      WRITELN('NO. MUESTRAS       =M');
      WRITE('NO. REGISTROS      =R   ');
      READ(S); WRITELN('');

      CASE S OF

       'E': CORRIGE(NEVMAX,10,FRECUENCIA);

       'M': CORRIGE(MUESTMAX,1,MUESTRAS);

       'R': CORRIGE(NREGMAX,1,NUMREG);

      END;(*CASE S*)

    END;(*THEN R*)

  END;(*REPEAT R*)

UNTIL R = 'N';

END;(*MODIFICA*)

(*---------*)

(* PROCEDIMIENTO PARA TOMAR LOS DATOS NECESARIOS PARA LA EJECUCION *)

PROCEDURE DATOS;

BEGIN(*DATOS*)

  LECTURA;
  MEDICION;
  MODIFICA;
  FRECUENCIA:=FRECUENCIA-1;(*CORRIGE EL NO. DE MUESTRAS*)

END;(*DATOS*)
```

```
(*---------*)

(*PROCEDIMIENTO PARA EL MANEJO DEL RESULTADO DE LA PRUEBA*)

PROCEDURE RESULTA;

BEGIN(*RESULTA*)

 RESPUESTA:=IO(63);

 IF ERROR IN RESPUESTA

  THEN

    CASE ERROR OF

     0 : WRITELN('CORRIDA NORMAL, CONCLUIDA!!!!!!!');
     63: WRITELN('INTERRUPCION DE FUENTE DESCONOCIDA');

    END(*CASE ERROR*)

  ELSE

    WRITELN('ERROR NO IDENTIFICADO, REPITA');

END;(*RESULTA*)

(*---------*)

(*PROCEDIMIENTO PARA CONVERTIR LAS TABLAS DE VELOCIDAD CONTRA TIEMPO*)

PROCEDURE INTERPRETA;

BEGIN(*INTERPRETA*)

 REWRITE(ASI,'#5:VELOCIDAD.DATA');(*ABRE ARCHIVO DE SALIDA*)
 IF (MUESTRAS=((FRECUENCIA+1)DIV 10)) > NUMREG
  THEN ASI^:=NUMREG DIV ((FRECUENCIA+1) DIV 10)
  ELSE ASI^:=MUESTRAS;(*ESCRIBE NO. DE REGISTROS Y DE PASOS POR MEDICION*)
 PUT(ASI);
 ASI^:=FRECUENCIA+1;
 PUT(ASI);
 TIEANT:=0;(*INICIALIZA TIEMPO Y POSICION*)
 TETANT:=0;

 FOR I:=1 TO NUMREG DO

  BEGIN(*FOR I*)

   IF I = 1 THEN (*THEN I*) VELREAL:=CONTO-VELALT[1]
            ELSE (*ELSE I*) VELREAL:=VELALT[I-1]-VELALT[I];

   IF SIGNO[I]=0 THEN (*THEN SIGNO*) J:=1
                 ELSE (*ELSE SIGNO*) J:=-1;

   VELREAL:=(1.0*VELREAL*CONTO+CONTO-VELBAJ[I]+MUERTO)/RELAPPLE;
   TIEANT:=TIEANT+VELREAL;ASI^:=TIEANT;PUT(ASI);(*ESCRIBE TIEMPO*)

   TETANT:=TETANT+(FRECUENCIA+1)*RESOL*J;
```

```
    ASI^:=TETANT;PUT(ASI);(*INCREMENTO DE TETA*)

    VELREAL:=(FRECUENCIA*J)*J*RESOL/VELREAL;(*VELOCIDAD*)
    ASI^:=VELREAL;PUT(ASI);(*ESCRIBE VELOCIDAD*)

  END;(*FOR I*)

 CLOSE(ASI,LOCK);(*CIERRA ARCHIVO DE RESULTADOS*)

END;(*INTERPRETA*)

(* PRINCIPIO DEL PROGRAMA PRINCIPAL*)

BEGIN(*CONTROL*)

  DATOS;(*TOMA DATOS PARA EJECUCION*)

  ERROR:=0;(*INICIALIZA INDICADOR DE ERROR*)

  TIEMPOREAL;(*RUTINA EN ENSAMBLADOR*)

  RESULTA;(*ANALIZA RESULTADOS*)

  INTERPRETA;(*TABLAS DE RESULTADOS*)

END.(*CONTROL*)
```

```
;TIEMPOREAL.TEXT
;-----------------------------------------------------
;
;   PROGRAMA PARA REALIZAR PRUEBAS DEL COMPORTAMIENTO
;   DE LOS MOTORES DE PASOS BAJO DIVERSOS REGIMENES
;   CARGA.
;   ESTE PROGRAMA FUNCIONA COMO UNA SUBRUTINA DEL
;   PROGRAMA CONTROL.TEXT, CONSTA DE TRES PARTES:
;        -INICIALIZACION
;        -SERVICIO AL RELOJ PARA ENVIO DE PULSOS
;        -SERVICIO AL TACOMETRO DIGITAL
;   SUS TAREAS MAS IMPORTANTES SON:
;        -PROGRAMACION INICIAL DEL RELOJ Y DEL TACOMETRO
;        -ATENCION A LA INTERRUPCIONES PROVACADAS POR
;         AMBOS DISPOSITIVOS
;        -REPROGRAMACION DESPUES DE LAS INTERRUPCIONES.
;        -CONTROL DEL NUMERO DE MEDICIONES.
;        -ENVIO DE PULSOS A LOS MOTORES DE PASOS.
;
;        LUIS ALVAREZ ICAZA LONGORIA, PROYECTO 2136
;        INSTITUTO DE INGENIERIA, UNAM.
;
;-----------------------------------------------------

        .PROC TIEMPOREAL
;-----------------------
;-----------------------
;
; INCLUYE LAS MACROINSTRUCCIONES
;-------------------------------

        .INCLUDE        #5:MACRO.TEXT
;
;-----------------------------------------------------
;
; DECLARACION DE VARIABLES COMUNES CON EL PROGRAMA EN PASCAL
;
;-----------------------------------------------------
;
        .PUBLIC         NUMREG,FRECUENCIA,MUESTRAS,ERROR
                                ;NO. DE REGISTROS,NO. DE EVENTOS
                                ;POR MUESTRA,NO. DE MUESTRAS E
                                ;INDICADOR DE ERROR
        .PUBLIC         VELBAJ,VELALT
                                ;TABLAS QUE CONTENDRAN LOS RESULTADOS DE LAS
                                ;MEDICIONES
        .PUBLIC         MASCARA,FACTOR;TABLAS QUE CONTIENEN LA MASCARA PARA EL
        .PUBLIC         TIEMPO,SIGNO  ;ENVIO DE PULSOS, EL FACTOR DE DIVISION
                                      ;PARA EL PRIMER RELOJ, EL TIEMPO ENTRE
                                      ;PULSOS ESCALADO SEGUN EL FACTOR ANTERIOR
                                      ;Y EL SIGNO DEL MOVIMIENTO

; FIN DE LA ZONA DE VARIABLES PUBLICAS
;-------------------------------------

; ZONA PARA DECLARACION DE DIRECCIONES DE PERIFERICOS Y DE VARIABLES
; DE DIRECCION FIJA EN MEMORIA
;-----------------------------------------------------------------

RETORNO .EQU    00      ;DIRECCION DE RETORNO A PASCAL
```

```
INDMAS  .EQU    02      ;AUXILIAR PARA TABLA DE MASCARAS
INDFAC  .EQU    04      ;IDEM TABLA DE FACTORES
INDSIG  .EQU    06      ;IDEM TABLA DE SIGNOS
INDTIE  .EQU    08      ;IDEM TABLA DE TIEMPOS
INDVB   .EQU    0A      ;IDEM TABLA DE VELOCIDADES BAJAS
INDVA   .EQU    0C      ;IDEM TABLA DE VELOCIDADES ALTAS
APIRQ   .EQU OFFFE      ;DIRECCION DEL VECTOR DE INTERRUPCIONES

;  ***IMPORTANTE***     SE HA SUPUESTO LA SIGUIENTE COLOCACION DE LOS PERIFER-
;                       RICOS: EL TIMER ESTARA EN LA RANURA 3, EL TACOMETRO
;                       DIGITAL ESTARA EN LA RANURA 4 Y EL CONVERSOR ANALO-
;                       GICO-DIGITAL ESTARA EN LA RANURA 5.
;!!!CUALQUIER CAMBIO !!!DE UBICACION DE ESTOS PERIFERICOS PROVOCARA FUNCIONA-
;                       MIENTO INCORRECTO.
;                       PARA CAMBIAR DE UBICACION CUALQUIERA DE ELLOS SERIA NE-
;                       CESARIO MODIFICAR LAS DIRECCIONES SEGUN LAS INDICACI-
;                       CIONES DEL MANUAL DE REFERENCIA DE LA APPLE II.

; LOCALIDADES NECESARIAS PARA MANEJAR EL TAC. DIG. (TD)

TIM1    .EQU    0C0C0   ;DIRECCION RELOJ 1 TD
TIM2    .EQU    0C0C4   ;    "       "   2 TD

UDC1    .EQU    0C0C8   ;DIRECCION UDC1
UDC2    .EQU    0C0CC   ;    "     " 2

FFS1    .EQU    0C0CA   ;DIRECCION PARA LEVANTAR FLIP-FLOP 1
FFS2    .EQU    0C0CB   ;    "       "    "    "    "  2

FFR1    .EQU    0C0CE   ;DIRECCION PARA BAJAR FLIP-FLOP 1
FFR2    .EQU    0C0CF   ;    "       "    "    "   "  2

; LOCALIDAD PARA MANEJAR EL RELOJ QUE CALCULA EL TIEMPO ENTRE PULSOS

RELOJ   .EQU    0C0B0   ;DIRECCION DEL RELOJ

; LOCALIDAD PARA EL ACOPLADOR DE ENVIO DE PULSOS

PULN0   .EQU    0C058   ;DIRECCION DEL ANUNCIADOR CERO SALIDA CERO
PULS1   .EQU    0C059   ;IDEM SALIDA UNO

;LOCALIDADES PARA CONTROL DE PARO Y ARRANQUE DEL BANCO DE PRUEBAS

PARO    .EQU    0C062   ;DIRECCION DE ENTRADA PARA PARO FORZOSO
ARRAN   .EQU    0C061   ;DIRECCION DE ENTRADA PARA INICIO DE PRUEBAS

; FIN DE LA ZONA DE LOCALIDADES FIJAS DE MEMORIA
;-----------------------------------------------
;
;  INICIO DEL PROGRAMA PRINCIPAL
;
;-------------------------------
;-------------------------------

FRIN:   TOMA    RETORNO ;GUARDA DIRECCION PARA REGRESAR A PASCAL

;----------------------

AFUN:   GUARDA  APSERV  ;OBTIENE DIR. RUTINA SERV. A INTERRUPCIONES
        TOMA    APIRQ
```

```
;--------------------

; RUTINAS DE INICIALIZACION DE LOS PERIFERICOS
;-----------------------------------------------

INIC:   LDY #00 ;INICIALIZA ANUNCIADORES(AMBOS ALTOS)
        STA PULS1,Y
        INY
        INY
        STA PULSI,Y

; SALVA VALORES DE LOS APUNTADORES A LAS TABLAS Y DATOS DE LA EJECUCION

        MUEVE MUESTRAS,NUMUEST
        MUEVE NUMREG,NOREG
        MUEVE APBAJA,INDVB
        MUEVE APALTA,INDVA
        MUEVE APFACT,INDFAC
        MUEVE APMASC,INDMAS
        MUEVE APTIEM,INDTIE
        MUEVE APSIG,INDSIG

; PROGRAMA TIMER

        INTRES 1  ;RESET INTERNO AL TIMER(RELOJES PARADOS)

        LEEFACTOR INDFAC,FACACT       ;LEE EL PRIMER FACTOR
        LEEMASCARA INDMAS,MASACT      ; "  LA PRIMERA MASCARA
        LEETIEMPO INDTIE,TIEACT       ; "  "   "    TIEMPO

        QUITA NOREG     ;ACTUALIZA EL NO. DE REGISTROS POR LEER

        LDA MODO1       ;PROGRAMA RELOJ 2 APUNTANDO AL RELOJ 3
        STA RELOJ+1

        PROGRAMA FACACT,TIEACT ;PROGRAMA TIMERS 2 Y 3 SEGUN FACTOR

        INTRES 1        ;PRECARGA CONTADORES(RELOJES AUN PARADOS)

; PROGRAMA TAC-DIG

        UDC FRECUENCIA,UDC1,UDC2
        ;PROGRAMA UDC'S CON NUMERO DE EVENTOS

        STA FFR1        ;ASEGURA QUE LOS RELOJES DEL TD ESTEN PARADOS

        MODO TIM1,MODTD0        ;PROGRAMA MODO DE OPERACION TIMERTD 1 CONT 0
        MODO TIM1,MODTD1        ;IDEM CONT 1

        CICLO                   ;INICIALIZA LOS RELOJES DEL TD

        UDC FRECUENCIA,UDC1,UDC2
        ;PROGRAMA UDC'S CON NUMERO DE EVENTOS

        ; ESPERA QUE SE DE LA ORDEN DE INICIO DE PRUEBAS (CONTROLADA POR UN
; INTERRUPTOR EN EL BANCO DE PRUEBAS
;-------------------------------------------------------------

ESPERA: LDA ARRAN               ;VERIFICA ESTADO DEL INTERRUPTOR
```

```
        DPL ESPERA              ;ESPERA A LA ORDEN DE ARRANQUE

; HABILITA INTERRUPCIONES
;-------------------------

HAB:    INTRES 0                ;ARRANCA RELOJ

        STA FFS1                ;ARRANCA TD
        STA FFS2

        LDX FRECUENCIA          ;CARGA DPS DE FRECUENCIA EN INDICE X
        CLI                     ;HABILITA INTERRUPCIONES AL MPU

; VERIFICA FIN DE EJECUCION
;---------------------------

LOOP:   LDA ERROR       ;VERIFICA ALGUN ERROR DE EJECUCION
        BNE DESHAB      ;FINALIZA SI OCURRIO ALGUNO

        LDA PARO        ;VERIFICA SENAL DE PARO FORZOSO
        BMI DESHAB      ;FINALIZA SI ESTA PUESTA

        LDA NOREG+1     ;VERIFICA SI HAY INFORMACION POR ENVIAR
        BMI DESHAB      ;FINALIZA SI NO LA HAY

        LDA NUMUEST+1   ;VERIFICA SI CONCLUYERON LAS MEDICIONES
        BPL LOOP        ;CONTINUA SI NO ES CIERTO

; DESHABILITA INTERRUPCIONES
;----------------------------

DESHAB: SEI             ;DESHABILITA INT. AL MPU

        STA FFR1        ;PARA RELOJES
        STA FFR2

        ;INTRES 1
        ;PARA RELOJ PROGRAMABLE

;REGRESA A PASCAL
;----------------

        GUARDA RETORNO  ;DIRECCION DE RETORNO AL STACK

        RTS             ;REGRESA A PASCAL

; FIN DEL PROGRAMA PRINCIPAL EN ENSAMBLADOR
;-------------------------------------------
;-------------------------------------------


; RUTINA DE SERVICIO A INTERRUPCIONES
;-------------------------------------
;-------------------------------------

SERV:   PHA             ;SALVA ACUMULADO

; VERIFICA LA FUENTE DE INTERRUPCION
;------------------------------------

        LDA RELOJ+1     ;PREGUNTA SI LA INTERRUPCION ES DEL RELOJ
        BPL VETD        ;INTERRUPCION DEL TD

        JMP SEREL       ;SIRVE AL RELOJ

; VERIFICA QUE LA INTERRUPCION SEA DEL RELOJ
;--------------------------------------------

VETD:   LDA UDC1        ;CARGA STATUS DEL TD
        AND #03         ;LIMPIA BITS CON BASURA
        STA TSTATUS     ;LO GUARDA

        EOR #03         ;VERIFICA
        BNE SITD        ;LA INTERRUPCION SI ES DEL TD

; INTERRUPCION PROVENIENTE DE FUENTE DESCONOCIDA
;------------------------------------------------

DESCO:  LDA #3F         ;ERROR:FUENTE DESCONOCIDA DE INTERRUPCION
        STA ERROR

        PLA             ;RECUPERA ACUMULADOR
        RTI             ;FIN DE INTERRUPCION

; INTERRUPCION DEL TD, LA SIRVE
;-------------------------------

SITD:   STX UDC1        ;PROGRAMA DPS DE UDC'S
        STX UDC2

        LDA FRECUENCIA+1;CARGA ACUMULADOR BMS PARA UDC'S
        STA UDC1+1      ;LOS PONE
        STA UDC2+1

        STA FFR1                ;PARA RELOJES

        LECTURA TIM1,INDVB,0    ;LEE CONTENIDO CONTADORES 0 Y 1

        STA FFS1        ;ARRANCA RELOJES
        STA FFS2

        MODO TIM1,MODMOV;LEE CONTADOR 1 EN MOVIMIENTO

        LECTURA TIM1,INDVA,1

        LDA TSTATUS     ;CARGA STATUS TD
        AND #01         ;PREGUNTA SI EL MOVIMIENTO FUE POSITIVO

        DIREC INDSIG    ;GUARDA DIRECCION DEL MOVINIENTO

        QUITA NUMUEST   ;ACTUALIZA EL NO. DE MUESTRAS QUE FALTAN

        PLA     ;RESTITUYE ACUMULADOR

        RTI     ;FIN DE INTERRUPCION

;SERVICIO AL RELOJ
;-----------------

SEREL:  LDA #01         ;DESHABILITA INTERRUPCIONES RELOJ 3
```

```
        STA RELOJ
        LDA MODO1       ;APUNTA AL RELOJ 3
        STA RELOJ+1

        LDY MASACT      ;CARGA MASCARA
        STY MASANT      ;GUARDA MASCARA
        STA PULNO,Y     ;MANDA PARTE BAJA DE PULSOS SEGUN MASCARA

        LEEFACTOR INDFAC,FACACT         ;LEE FACTOR
        LEEMASCARA INDMAS,MASACT        ; "  MASCARA
        LEETIEMPO INDTIE,TIEACT         ; "  TIEMPO

        PROGRAMA FACACT,TIEACT ;PROGRAMA TIMERS 2 Y 3 SEGUN FACTOR

        INTRES 1        ;CARGA CONTADORES A NUEVO VALOR
        LDA #00         ;ARRANCA RELOJES
        STA RELOJ

        LDY MASANT      ;TERMINA DE FORMAR PULSO
        STA PULSI,Y

        QUITA NOREG     ;ACTUALIZA EL NO. DE REGISTROS POR LEER

        PLA     ;RECUPERA ACUMULADOR
        RTI     ;FIN DE INTERRUPCION

; FIN DE LA RUTINA DE SERVICIO A INTERRUPCIONES
;---------------------------------------------
;---------------------------------------------

; ZONA PARA LA DECLARACION DE VARIABLES CON VALOR INICIAL Y DE APUNTADORES
;-------------------------------------------------------------------------

APSERV  .WORD   SERV    ;APUNTADOR A LA RUTINA DE SERV. A INT.
APBAJA  .WORD   VELBAJ  ;   "       "  " TABLA DE LOS 1o DPS DE MEDICIONES
APALTA  .WORD   VELALT  ;   "       "  "   "   "   "   " DMS "       "
APFACT  .WORD   FACTOR  ;   "       "  "   "   "   "  FACTORES DE ESCALA
APMASC  .WORD   MASCARA ;   "       "  "   "   "   "  MASCARAS PARA PULSOS
APTIEM  .WORD   TIEMPO  ;   "       "  "   "   "   "  TIEMPOS
APSIG   .WORD   SIGNO   ;   "       "  "   "   "   "  PARES
TIEACT  .WORD   0000    ;VALOR ACTUAL DE TIEMPO
FACACT  .BYTE   0000    ; "     "     "  FACTOR
MACACT  .BYTE   OFF     ; "     "     "  LA MASCARA
MACANT  .BYTE   OFF     ; "   ANTERIOR"  "    "
ISTATUS .BYTE   00      ;  "     "  TD
MODO1   .BYTE    86     ;MODO DE OPERACION DEL RELOJ2 APUNTANDO AL 3
MODO2   .BYTE    87     ;  "   "     "      "     "  2 APUNTANDO AL 1
MODO3   .BYTE   0C0     ;  "   "     "      "     "  3 (SINCRONIA RELOJ2)
MODO4   .BYTE   0C2     ;  "   "     "      "     "  3 (SINCRONIA APPLE)
DIEZ1   .WORD   0104    ;EQUIVALENTE A 10 DEC.HACIENDO (DMS+1)*(DPS+1)
DIEZ2   .WORD   1203    ;EQUIVALENTE A 100 DEC. IDEM
DIEZ3   .WORD   0F203   ;EQUIVALENTE A 1000 DEC. IDEM
CONT0   .WORD   7FFF    ;EQUIVALENTE A 32767 DEC.
CONT1   .WORD   7FFF    ;   "     "  "   "   "
DUMMY1  .WORD   0006    ;   "     "  " 6 DEC. PARA INICIALIZAR TD1 CONT1
NUMOV   .BYTE    40     ;PERMITE LECTURAS EN MOV DEL CONT 1
MODTD0  .BYTE    34     ;MODO DE PROGRAMACION CONT 0 TD
MODTD1  .BYTE    70     ;  "   "     "        CONT 1 TD
NUMMEST .WORD   0000    ;LOC. TEMP. PARA MUESTRAS
NOREG   .WORD   0000    ;  "    "   "   NUMREG

;FIN DE LA ZONA DE VARIABLES Y APUNTADORES
;-----------------------------------------

        .END

; FIN DEL PROGRAMA EN ENSAMBLADOR
;-------------------------------
;-------------------------------
```

```
;-------------------------------------------------------------------
;
; MACROINSTRUCCIONES DEL PROGRAMA PARA REALIZAR LAS PRUEBAS DEL COMPOR-
; TAMIENTO DE LOS MOTORES DE PASOS.
;
; ESTAS MACROINSTRUCCINES SON INSERTADAS EN EL PROGRAMA TIEMPOREAL.TEXT
; CUANDO ESTE ES ENSAMBLADO
;
;    LUIS ALVAREZ ICAZA LONGORIA . PROYECTO 2136
;    INSTITUTO DE INGENIERIA, UNAM
;
;-------------------------------------------------------------------

; MACRO PARA OBTENER DOS BYTES DEL STACK Y GUARDARLOS EN UNA LOCALIDAD DADA

        .MACRO TOMA

        PLA     ;OBTIENE EL BPS
        STA %1  ;LO GUARDA EN BPS DE LOCALIDAD

        PLA     ;OBTIENE EL BMS
        STA %1+1;LO GUARDA EN BMS DE LOCALIDAD

        .ENDM

; MACRO PARA PONER DOS BYTES DE MEMORIA EN EL STACK

        .MACRO GUARDA

        LDA %1+1;OBTIENE BMS
        PHA     ;LO PASA AL STACK

        LDA %1  ;OBTIENE BPS
        PHA     ;AL STACK

        .ENDM

; MACRO PARA CARGAR EL REGISTRO DE CONTROL DE LOS TIMERS DEL TD.

        .MACRO MODO

        LDA %2  ;CARGA MODO DE CONTROL
        STA %1+3;LO PASA AL REGISTRO DE CONTROL DEL TIMER DESEADO

        .ENDM

; MACRO PARA PROGRAMAR LOS RELOJES DEL TD. ESCRIBE DOS BYTES SOBRE
; ALGUNO DE LOS CONTADORES (ES MUY IMPORTANTE ESCRIBIR SIEMPRE LOS
; DOS BYTES).

        .MACRO TP

        LDA %2   ;CARGA BPS
        STA %1+%3;LO GUARDA EN EL CONTADOR DESEADO (0 A 2,SEGUN %3)

        LDA %2+1 ;CARGA BMS
        STA %1+%3;AL CONTADOR

        .ENDM

; MACRO PARA PROGRAMAR LOS CONTADORES HACIA ARRIBA Y ABAJO (UDC)

        .MACRO UDC

        LDA %1  ;CARGA BPS
        STA %2  ;AL UDC #1
        STA %3  ;"   "  #2

        LDA %1+1;CARGA BMS
        STA %2+1;AL UCD #1(SOLO LOS 4 BITS MENOS SIGNIFICATIVOS)
        STA %3+1;IGUAL PARA EL UDC #2

        .ENDM

; MACRO PARA LEER EL VALOR DE LOS TIMERS DEL TD

        .MACRO LECTURA

        LDY #00    ;INICIALIZA INDICE Y

        LDA %1+%3  ;CARGA BPS DEL TIMER
        STA @%2,Y  ;LOS GUARDA

        INY        ;INCREMENTA Y

        LDA %1+%3  ;CARGA BMS
        STA @%2,Y  ;LOS GUARDA

        SUM2 %2    ;ACTUALIZA APUNTADOR

        .ENDM

; MACRO PARA MOVER DOS BYTES CONSECUTIVOS A OTRA LOCALIDAD

        .MACRO MUEVE

        LDA %1          ;CARGA BPS
        STA %2          ;LOS MUEVE

        LDA %1+1        ;CARGA BMS
        STA %2+1        ;LOS MUEVE

        .ENDM

;MACRO PARA INCREMENTAR EL VALOR DE UNA VARIABLE EN LA UNIDAD

        .MACRO SUM1

        INC %1  ;INCREMENTA EL VALOR DE LA VARIABLE

        BNE $01 ;PREGUNTA POR CRUCE DE PAGINA

        INC %1+1;ACTUALIZA SI OCURRIO EL CRUCE

$01     NOP     ;DE LO CONTRARIO CONTINUA

        .ENDM

;MACRO PARA INCREMENTAR EL VALOR DE UNA VARIABLE DE 16 BITS EN DOS UNIDADES
```

```
        .MACRO SUM2

        CLC     ;BORRA CARRY

        LDA %1  ;CARGA BPS DE ARGUMENTO
        ADC #02 ;LOS INCREMENTA
        STA %1  ;LOS SALVA

        LDA %1+1;CARGA AHORA BMS
        ADC #00 ;LOS ACTUALIZA EN CASO DE CRUCE DE PAGINA
        STA %1+1;LOS SALVA

        .ENDM

; MACRO PARA OBTENER EL DIVISOR DE FRECUENCIA DEL SEGUNDO RELOJ

        .MACRO LEEFACTOR

        LDY #00         ;BORRA INDICE Y

        LDA @%1,Y       ;CARGA VALOR DE TABLAS

        AND #03         ;CONSERVA LOS DOS PRIMEROS BITS

        STA %2          ;LOS GUARDA

        SUMI %1         ;ACTUALIZA APUNTADOR (DE 1 EN 1 BYTE)

        .ENDM

; MACRO PARA LEER EL VALOR A CARGAR AL SEGUNDO CONTADOR

        .MACRO LEETIEMPO

        LDA @%1,Y       ;CARGA BPS DEL TIEMPO
        STA %2          ;LOS GUARDA (Y=0 EN LEEFACTOR)

        INY             ;INCREMENTA INDICE Y

        LDA @%1,Y       ;IDEM PARA LOS BMS
        STA %2+1

        SUM2 %1         ;ACTUALIZA APUNTADOR (DE 2 EN 2 BYTES)

        .ENDM

; MACRO PARA LEER LA MASCARA QUE INDICA EL MOTOR AL CUAL ENVIAR EL PULSO

        .MACRO LEEMASCARA

        LDA @%1,Y       ;CARGA VALOR DE TABLAS (Y=0 EN LEEFACTOR)

        STA %2          ;GUARDA VALOR LEIDO

        SUMI %1         ;ACTUALIZA APUNTADOR (DE 1 EN 1 BYTE)

        .ENDM

; MACRO PARA DAR RESET INTERNO AL TIMER

        .MACRO INTRES

        LDA MODO2       ;PROGRAMA MODO DE OPERACION
        STA RELOJ+1     ;APUNTA A RELOJ 1

        LDA #%1         ;PARA RELOJES(RESET INTERNO)
        STA RELOJ       ;O LOS ARRANCA

        .ENDM

; MACRO PARA PROGRAMAR LOS RELOJES 2 Y 3 SEGUN FACTOR

        .MACRO PROGRAMA

        LDA %1  ;PASA FACTOR A ACUMULADOR

        BEQ $01 ;PREGUNTA POR PRIMER FACTOR

        CMP #01 ;PREGUNTA POR SEGUNDO FACTOR
        BEQ $02

        CMP #02 ;   "       "  TERCER     "
        BEQ $03

        ; PROGRAMA PARA FACTOR=1E3

        LDA MODO3       ;PROGRAMA MODO DE OPERACION RELOJ 3
        STA RELOJ

        LDA DIEZ2+1     ;CARGA BMS DE FACTOR
        STA RELOJ+4
        LDA DIEZ3       ;  "   BPS
        STA RELOJ+5

        JMP $04

        ;PROGRAMA FACTOR = 1E0

$01     LDA MODO4       ;PROGRAMA RELOJ 3
        STA RELOJ

        JMP $04

        ;PROGRAMA FACTOR = 1E1

$02     LDA MODO3       ;PROGRAMA RELOJ 3
        STA RELOJ

        LDA DIEZ1+1     ;CARGA BMS DE FACTOR(RELOJ2)
        STA RELOJ+4
        LDA DIEZ1       ;IDEM BPS
        STA RELOJ+5

        JMP $04

        ;PROGRAMA FACTOR = 1E2

$03     LDA MODO3       ;PROGRAMA RELOJ 3
        STA RELOJ
```

```
        LDA DIEZ2+1     ;CARGA DMS DE FACTOR(RELOJ2)
        STA RELOJ+4
        LDA DIEZ2       ;  "   DPS
        STA RELOJ+5

        ;CARGA RELOJ 3

604     LDA %2+1        ;CARGA DMS DE TIEMPO (RELOJ3)
        STA RELOJ+6
        LDA %2          ;  "   DPS
        STA RELOJ+7

        .ENDM

; MACRO PARA INICIALIZAR LOS RELOJES DEL TACOMETRO DIGITAL

        .MACRO CICLO

        TP TIM1,DUMMY1,0        ;PROGRAMA TIMER1D1 CONTO PARA INIC. CONT1
        TP TIM1,CONT1,1         ;PROGRAMA   "        "  1

        STA FFS1        ;ARRANCA RELOJES

        NOP     ;RETRASO PARA COMPLETAR INICIALIZACION DE CONTADORES
        NOP

        STA FFR1        ;PARA RELOJES

        TP TIM1,CONTO,0 ;PROGRAMA TIMER1D1 CONTO

        STA FFS1        ;ARRANCA RELOJES
        STA FFR1        ;PARA RELOJES

        .ENDM

; MACRO PARA DECREMENTAR UNA NUMERO DE 16 BITS EN UNO DE OCHO

        .MACRO QUITA

        SEC             ;LEVANTA CARRY
        LDA %1          ;CARGA DPS
        SBC #01         ;DECREMENTA EN EL NO. DESEADO

        STA %1          ;GUARDA RESULTADO

        LDA %1+1        ;CARGA DMS
        SBC #00         ;DECREMENTA EN CASO DE CRUCE DE PAGINA

        STA %1+1        ;GUARDA RESULTADO

        .ENDM

; MACRO PARA OBTENER LA DIRECCION DE MOVIMIENTO DE LOS MOTORES

        .MACRO DIREC

        LDY #00         ;BORRA INDICE Y

        STA @%1,Y       ;GUARDA VALOR DEL STATUS DEL UDC

        SUMI %1         ;ACTUALIZA APUNTADOR

        .ENDM

; FIN DEL ARCHIVO DE MACROINSTRUCCIONES
;-------------------------------------
;-------------------------------------
```

```
(*$A:PASO,TEXT*)
(**********************************************)
(*                                            *)
(*    PROGRAMA PARA ENCONTRAR LA RESPUESTA    *)
(*    DE UN MOTOR DE PASOS CUANDO SE APLICA   *)
(*    UN SOLO PASO. A PARTIR DE AQUELLA SE    *)
(*    PRETENDE ESTIMAR LOS PARAMETROS DE LA   *)
(*    ECUACION DIFERENCIAL QUE DESCRIBE LA    *)
(*    DINAMICA DEL MOTOR DE PASOS.            *)
(*                                            *)
(*            PROYECTO     2136               *)
(*                LAIL                        *)
(*            NOVIEMBRE 1, 1982               *)
(*                                            *)
(**********************************************)

PROGRAM PASO;

(* ZONA PARA DECLARACION DE CONSTANTES Y VARIABLES *)

CONST   MEDMAX = 500; (*NUMERO MAXIMO DE MEDICIONES QUE TOMARA EL TACOMETRO*)
        EVMAX = 4095; (*NUMERO MAXIMO DE EVENTOS POR MEDICION*)
        RESOLUCION = 0.15707963; (*RESOLUCION DEL MOTOR DE PASOS EN RADI.*)
        RELOJ=1.0205E6;(*FRECUENCIA DEL RELOJ DE LA APPLE II*)

TYPE    ARREGLO = ARRAY[1..MEDMAX] OF INTEGER;
                  (*TIPO DE ARREGLO PARA GUARDAR RESULTADOS DE MED.*)

        BITIOS  = PACKED ARRAY[1..MEDMAX] OF 0..255;
                  (*ARREGLO DE BYTES PARA GUARDAR DIRECCION*)


        REALES = ARRAY[1..MEDMAX] OF REAL;
                  (*ARREGLOS CON LOS TIEMPOS Y VELOCIDADES FINALES *)

VAR     J: BOOLEAN; (*INDICADOR PARA LETREROS*)

        NMED,(*NUMERO DE MEDICIONES DESEADAS*)
        NEVENTOS,(*NUMERO DE EVENTOS POR MEDICION*)
        CONT0,(*ARGUMENTO QUE SE DECRECERA EN EL TIMER 0 *)
        CONT1,(*  "         "  "     "     "  "    "  1 *)
        ERROR,(*INDICADOR DE EJECUCION DE LA RUTINA EN ENSAMBLADOR*)
        I: INTEGER;(*VARIABLE PARA ITERACIONES*)

        R: CHAR; (*VARIABLE DE CONTROL PARA ACEPTAR COMANDOS*)

        VELBAJ,(*ARREGLO CON LOS 16 BITS MAS SIGNIFICATIVOS DE LA MEDICION*)
        VELALT: ARREGLO;(*COMO VELBAJ PERO LOS BITS MENOS SIGNIFICATIVOS  *)

        SIGNO: BITIOS;(*SIGNO DEL MOVIMIENTO*)

        POSICION,(*ARREGLO CON LAS POSICIONES ANGULARES*)
        TIEMPO: REALES;(*ARREGLO CON LOS INCREMENTOS DE TIEMPO*)

        NOMBRE: STRING[40];(*NOMBRE DEL ARCHIVO DE SALIDA*)

        AP: FILE OF REAL;(*APUNTADOR AL ARCHIVO DE SALIDA*)

(* DECLARACION DE LA RUTINA EN ENSAMBLADOR *)
```

```
PROCEDURE MIDEPASO;

EXTERNAL; (*SE DECLARA EXTERNA *)


(*INICIO PROGRAMA PRINCIPAL*)

BEGIN(*PASO*)

  WRITELN('PROGAMA PARA HALLAR LA RESPUESTA A');
  WRITELN('PULSO DE UN MOTOR DE PASOS');

  CONT0:=32767;(*ASIGNA VALOR AL TIMER 0*)
  CONT1:=CONT0;(*  "      "    "   "   1*)

  (*INICIA RUTINA DE MEDICIONES*)

  REPEAT(*UNTIL R<>'S'*)

    REPEAT(*NMED*)

      WRITE('CUANTAS MEDICIONES(MAX=500):  ');
      READLN(NMED);
      IF NMED <=0 THEN NMED:=MEDMAX+1;

    UNTIL NMED <=MEDMAX;(*NMED*)

    REPEAT(*NEVENTOS*)

      WRITE('CUANTOS EVENTOS POR MEDICION(MAX=4096): ');
      READLN(NEVENTOS);
      NEVENTOS:=NEVENTOS-1;(*CORRIGE EL NO. DE EVENTOS*)
      IF NEVENTOS < 0 THEN NEVENTOS:=EVMAX+1;

    UNTIL NEVENTOS <=EVMAX;(*NEVENTOS*)

    ERROR:=0;(*INICIALIZA INDICADOR DE ERROR*)
    J:=TRUE;(*INDICADOR DE CABEZA DE PAGINA*)

    (* RUTINA EN ENSAMBLADOR *)

    MIDEPASO;

    (* FIN RUTINA EN ENSAMBLADOR *)

    IF ERROR = 0 (*ERROR*)

      THEN(*ERROR*)

        BEGIN(*THEN*)

         FOR I:=1 TO NMED DO(*MODIFICA EL ARCHIVO DE DIRECCIONES*)
             IF SIGNO[I] =2 THEN SIGNO[I]:=0 ELSE SIGNO[I]:=2;

         (*VALORES INICIALES DE TIEMPO Y POSICION*)
         TIEMPO[1]:=((CONT0-VELALT[1])*CONT0+ CONT0- VELBAJ[1]+ 109)/RELOJ;
         POSICION[1]:=RESOLUCION*9;
         IF SIGNO[1] = 1 THEN POSICION[1]:= POSICION[1]*(-1);
```

```
    FOR I:=2 TO NMED DO

     BEGIN(*FOR I*)

       IF J THEN WRITELN(' POSICION         TIEMPO');

       TIEMPO[I]:=VELALT[I-1]-VELALT[I];

       TIEMPO[I]:=((TIEMPO[I]*CONTO+ CONTO -VELMAJE[I] + 109)/RELOJ;
       (*OBTIENE TIEMPO REAL*)

       POSICION[I]:=POSICION[I-1]+RESOLUCION*(SIGNO[I]-1);

       WRITELN(POSICION[I]:15,TIEMPO[I]:15);

       IF I MOD 23 = 0 THEN J:=TRUE ELSE J:=FALSE;(*CABEZA DE PAGINA*)

     END(*FOR I*)

   END(*THEN*)

  ELSE(*ERROR*)

    BEGIN (*ELSE*)

      IF ERROR = 129(*ERROR=129*)

       THEN

         WRITELN('INTERRUPCION MUY RAPIDA')

       ELSE

         WRITELN('INTERRUPCION DE FUENTE DESCONOCIDA');

    END;(*ELSE*)

  WRITE('NOMBRE DEL ARCHIVO DE SALIDA: ');READLN(NOMBRE);(*NOMBRE ARCHIVO DE SAL.*)
  REWRITE(AP,NOMBRE);(*LO ABRE*)
  FOR I:=1 TO NMED DO
   BEGIN(*FOR I*) AP^:=POSICION[I];PUT(AP);AP^:=TIEMPO[I];PUT(AP);END;(*FOR*)
  CLOSE(AP,LOCK);(*LO CIERRA Y GUARDA*)

  WRITE('CONTINUO?(S/N): ');(*PREGUNTA SI CONCLUYEN LAS MEDICIONES*)
  READ(R);
  WRITELN('');

 UNTIL R <>'S';(*UNTIL R <>'S'*)

END.(*PASO*)
```

```
;MIDEPASO.TEXT
;*********************************************
;
;  RUTINA EN ENSAMBLADOR PARA CONTROLAR EL TA-
;  COMETRO DIGITAL
;
;              PROYECTO      2126
;              LAIL NOVIEMBRE  82
;
;  ESTA RUTINA ES LLAMADA COMO UN PROCEDURE
;  DEL PROGRAMA  PASO.CODE
;
;  SE INCLUYERON MODIFICACIONES PARA INICIAR LAS
;  PRUEBAS A PETICION DEL USUARIO, MANDAR EL PUL-
;  SO A TRAVES DE LOS ANUNCIADORES Y FORZAR EL
;  FIN DEL EXPERIMENTO.
;
;*********************************************

        .PROC  MIDEPASO

; ZONA PARA LA DECLARACION DE MACROINSTRUCCIONES
;-----------------------------------------------

; MACRO PARA OBTENER DOS BYTES DEL STACK Y GUARDARLOS EN UNA LOCALIDAD DADA

        .MACRO TOMA

        PLA     ;OBTIENE EL DPS
        STA %1  ;LO GUARDA

        PLA     ;OBTIENE EL PMS
        STA %1+1;LO GUARDA

        .ENDM

; MACRO PARA PONER DOS BYTES DE MEMORIA EN EL STACK

        .MACRO GUARDA

        LDA %1+1;OBTIENE PMS
        PHA     ;LO PONE EN EL STACK

        LDA %1  ;OBTIENE DPS
        PHA     ;LO PONE EN EL STACK

        .ENDM

; MACRO PARA CARGAR EL REGISTRO DE CONTROL DE LOS TIMERS

        .MACRO MODO

        LDA %2          ;CARGA MODO DE CONTROL
        STA %1+3        ;APUNTA AL REGISTRO DE CONTROL DEL TIMER EN CUESTION

        .ENDM

; MACRO PARA PROGRAMAR LOS RELOJES DEL TACOMETRO.
; ESCRIBE DOS BYTES SOBRE ALGUNO DE LOS CONTADORES
; (ES IMPORTANTE ESCRIBIR SIEMPRE LOS DOS BYTES)
```

```
        .MACRO IP

        LDA %2          ;CARGA DPS
        STA %1+%3       ;LO GUARDA EN EL CONTADOR

        LDA %2+1        ;CARGA DMS
        STA %1+%3       ;LO GUARDA EN EL CONTADOR

        .ENDM

; MACRO PARA SALVAR EN EL STACK EL VALOR DE LOS ACUMULADORES
; E INDICES AL OCURRIR INTERRUPCION

        .MACRO SALVA

        PHA     ;GUARDA ACUMULADOR
        TYA     ;PASA INDICE Y A ACUMULADOR
        PHA     ;SALVA INDICE Y

        .ENDM

; MACRO PARA RECUPERAR EL VALOR DEL ACUMULADOR E INDICES AL
; CONCLUIR LA INTERRUPCION

        .MACRO RECUPERA

        PLA     ;OBTIENE INDICE Y
        TAY
        PLA     ;OBTIENE ACUMULADOR

        .ENDM

; MACRO PARA PROGRAMAR LOS CONTADORES HACIA ARRIBA Y HACIA ABAJO

        .MACRO UDC

        LDA %1  ;CARGA DPS
        STA %2  ;LO GUARDA EN EL UDC #1
        STA %3  ;  "    "  " "  "  #2

        LDA %1+1;CARGA DMS
        STA %2+1;LO GUARDA EN EL UDC #1(SOLO LOS 4 BITS MENOS SIGNIFICATIVOS)
        STA %3+1"   "  "  " "  #2 "   "  " "  "       "       )

        .ENDM

; MACRO PARA LEER EL VALOR DE LOS TIMERS

        .MACRO LECTURA

        LDY #00         ;PONGA INDICE Y

        LDA %1+%3       ;LEE DPS DEL TIMER
        STA @%2,Y       ;LOS GUARDA EN TABLAS

        INY             ;INCREMENTA INDICE Y

        LDA %1+%3       ;LEE DMS DEL TIMER
        STA @%2,Y       ;LOS GUARDA

        CLC             ;ACTUALIZA APUNTADOR
                        ;SIGUE SI NO HAY CRUCE DE PAGINA
        LDA %2          ;ACTUALIZA EN CASO DE CRUCE
        ADC #02

        STA %2

        LDA %2+1
        ADC #00
        STA %2+1

        .ENDM

; MACRO PARA MOVER DOS BYTES

        .MACRO MUEVE

        LDA %1  ;CARGA DPS
        STA %2  ;LOS GUARDA

        LDA %1+1;CARGA DMS
        STA %2+1;LOS GUARDA

        .ENDM

; MACRO PARA GUARDAR EL SIGNO DEL MOVIMIENTO

        .MACRO DIREC

        LDY #00         ;PONGA Y

        STA @INDIR3,Y   ;GUARDA SIGNO

        INC INDIR3      ;ACTUALIZA APUNTADOR

        BNE $01         ;PREGUNTA POR CRUCE DE PAGINA
        INC INDIR3+1

$01:    NOP

        .ENDM

; FIN DE LA ZONA DE MACROINSTRUCCIONES
;-------------------------- --------

; ZONA PARA LA DECLARACION DE LAS VARIABLES DE LA RUTINA QUE SE COMPARTEN
; CON EL PROGRAMA EN PASCAL
;-------------------------------------------------------------------------

        .PUBLIC CONTO,CONT1;INDICADORES DEL VALOR A DECREMENTAR EN LOS TIMERS
                           ;1 Y 2
        .PUBLIC NMED,NEVENTOS;NUMERO DE MEDICIONES Y DE EVENTOS POR MEDICION
        .PUBLIC ERROR;INDICADOR DE EJECUCION CORRECTA DE LA RUTINA

        .PUBLIC VELALT,VELTAB;TABLAS PARA GUARDAR LOS VALORES LEIDOS DE LOS
                             ;TIMERS (SE LEEN 32 BITS)
        .PUBLIC SIGNO;TABLA CON LA DIRECCION DEL MOVIMIENTO
```

```
;FIN DE LA ZONA DE VARIABLE PUBLICAS
;-----------------------------------

; ZONA PARA LA DECLARACION DE DIRECCIONES DE PERIFERICOS
; Y DE VARIABLES DE DIRECCION FIJA EN MEMORIA
;-------------------------------------------- ------

RETORNO .EQU    00      ;DIRECCION DE RETORNO A PASCAL
INDIR1  .EQU    02      ;AUXILIAR PARA DIRECCIONAMIENTO INDIRECTO
INDIR2  .EQU    04      ; IDEM
INDIR3  .EQU    06      ; IDEM

F_INO   .EQU    0C059   ;DIRECCION PARA APAGAR EL ANUNCIADOR 0
FSI0    .EQU    0C059   ;   "       "  PRENDER "       "    0
F_IN1   .EQU    0C05A   ;DIRECCION PARA APAGAR EL ANUNCIADOR 1
FSI1    .EQU    0C05E   ;   "       "  PRENDER "       "    1
ENT0    .EQU    0C061   ;   "     ENTRADA BINARIA 0
ENT1    .EQU    0C062   ;   "       "     "     1

APINT   .EQU OFFFE      ;DIRECCION DEL VECTOR DE INTERRUPCIONES

;                       IMPORTANTE: SE HA SUPUESTO QUE EL TACOMETRO DIGITAL
;                       SERA COLOCADO EN LA RANURA 4, CUALQUIER CAMBIO DE
;                       UBICACION PROVOCARA FUNCIONAMIENTO INCORRECTO.
;                       SI SE DESEA CAMBIAR LA RANURA SE DEBE CAMBIAR EL
;                       SEGUNDO DIGITO (C) DE LAS DIRECCIONES QUE SIGUEN POR
;                       EL ADECUADO A LA NUEVA RANURA (VER MANUAL APPLE)

TIM1    .EQU 0C0C0      ;DIRECCION TIMER 1

UDC1    .EQU 0C0C8      ;DIRECCION UDC 1
UDC2    .EQU 0C0CC      ;   "      UDC 2

FFS1    .EQU 0C0CA      ;DIRECCION PARA LEVANTAR FLIP-FLOP 1
FFS2    .EQU 0C0CB      ;   "        "     "    FLIP-FLOP 2

FFR1    .EQU 0C0CE      ;DIRECCION PARA BAJAR FLIP-FLOP 1
FFR2    .EQU 0C0CF      ;   "        "    "   FLIP-FLOP 2

;FIN ZONA DECLARACION DE DIRECCIONES
;-----------------------------------


; INICIO DEL PROGRAMA EN ENSAMBLADOR
;-----------------------------------

FRIM:   TOMA RETORNO ;GUARDA DIRECCION DE REGRESO A PASCAL
        STA FOSI     ;GARANTIZA EL ESTADO DE LOS ANUNCIADORES
        STA FISI     ;(DEBE SER ALTO)

AFUN:   GUARDA APSERV;OBTIENE DIRECCION DE LA RUTINA DE SERV. A INTERRUPCIONES
        TOMA APINT   ;LA PASA AL VECTOR DE INTERRUPCIONES

PAOG:   LDA NMED;CARGA DPS DE NUMERO DE MEDICIONES
        STA NUMERO; LOS PASA A NUMERO
        LDA NMED+1;LO MISMO PARA LOS IMS
        STA NUMERO+1

        MUEVE APBAJA,INDIR1;SALVA LA DIRECCION DE LAS TABLAS
        MUEVE APALTA,INDIR2;EN VARIABLES DE PAGINA CERO

        MUEVE APSIG,INDIR3

        UDC NEVENTOS,UDC1,UDC2;PROGRAMA UDC CON NUMERO DE EVENTOS POR MEDICION

        STA FFR1        ;ASEGURA QUE LOS RELOJES ESTEN PARADOS

        MODO TIM1,MODO0         ;PROGRAMA MODO DE OPERACION TIMER 1 CONT 0
        MODO TIM1,MODO1         ;   "       "  "    "       "    " CONT 1

        TP TIM1,DOWN1,0;PROGRAMA TIMER1 CONTADOR 0
        TP TIM1,CONT1,1;PROGRAMA TIMER1 CONTADOR 1

        STA FFS1 ;LEVANTA BANDERA DE CUENTA MINIMA EN UDC1

        NOP     ;RETRASO PARA COMPLETAR LA INICIALIZACION DE LOS CONTADORES 1
        NOP

        STA FFR1        ;BAJA BANDERA DE CUENTA MINIMA(PARA RELOJES TIMER 1)

        TP TIM1,CONT0,0;PROGRAMA TIMER 1 CONTADOR 0

        STA FFS1        ;ARRANCA RELOJES TIMER 1
        STA FFR1        ;LOS PARA DESPUES DE INICIALIZAR CONTADOR 0

        UDC NEVENTOS,UDC1,UDC2;PROGRAMA UDC CON NUMERO DE EVENTOS POR MEDICION

AUNNO:  LDA ENT0        ;VERIFICA INDICADOR DE PRINCIPIO DE PIEZA
        BMI AUNNO       ;NO SIGUE SI NO SE ESTA LISTO

        STA FON0        ;FORMA PARTE BAJA DE PULSO
        STA FFS1        ;HABILITA RELOJES DESPUES DE INICIALIZACION
        STA FFS2

        STA FOS1        ;FORMA PARTE ALTA PULSO

HAB:    LDA NEVENTO:    ;CARGA DPS DE NUMERO DE EVENTOS EN ACUMULADOR
        CLI     ;HABILITA INTERRUPCIONES AL MPU

LOOP:   LDX ENT1  ;VERIFICA QUE NO SE DESEE PARAR EL EXPERIMENTO

        BMI DESHAB;FINALIZA SI ASI ES

        LDX NUMERO+1;PREGUNTA SI YA CONCLUYO CON TODAS LAS MEDICIONES

        BPL LOOP  ;CONTINUA SI NO ES CIERTO

DESHAB: SEI     ;DESHABILITA INTERRUPCIONES AL MPU

        STA FFR1        ;PARA RELOJES
        STA FFR2

        GUARDA RETORNO; CARGA EN STACK LA DIRECCION DE RETORNO A PASCAL

        RTS     ;REGRESA A PASCAL

; FIN DE LA RUTINA EN ENSAMBLADOR
;--------------------------------

; PRINCIPIA LA RUTINA DE SERVICIO A INTERRUPCIONES
;-------------------------------------------------
```

```
INI:    STA UDC1        ;REPROGRAMA DEL DE UDC1
        STA UDC2        ;IDEM UDC2

        LDA NEVENTOS+1  ;AHORA LOS BMS
        STA UDC1+1
        STA UDC2+1

        LDA UDC1        ;OBTIENE DIRECCION DE MOVIMIENTO

        STA FFF1        ;PARA TIMER 1

        AND #03         ;ENMASCARA DIRECCION
        DIREC           ;LA GUARDA

        LECTURA TIM1,INDIR1,0;LEE CONTENIDO CONTADOR 0

        STA FFS1        ;ARRANCA TIMER 1
        STA FFS2        ;ARRANCA TIMER 2

        MODO TIM1,MODO2;PROGRAMA LECTURA DE CONTADOR 1 EN MOV.

        LECTURA TIM1,INDIR2,1

        DEC NUMERO      ;DECREMENTA BYTE DE NUMERO
        LDA #FF
        CMP NUMERO      ;VERIFICA CRUCE DE PAGINA
        BNE SIGUE

        DEC NUMERO+1    ;DECREMENTA EN CASO DE CRUCE

SIGUE:  LDA NEVENTO

        RTI     ;REGRESA DE INTERRUPCION

; FIN DE LA RUTINA DE SERVICIO A INTERRUPCIONES
;----------------------------------------------

; ZONA DE DECLARACION DE VARIABLES CON VALOR INICIAL Y DE APUNTADORES
;-------------------------------  ---------------------------------

APSERV  .WORD   SERV ;APUNTADOR A LA RUTINA DE SERVICIO A INTERRUPCIONES
APBAJA  .WORD   VELBAJ;APUNTADOR A LA TABLA DE LOS 16 BYTES DE LAS MEDICIONES
APALTA  .WORD   VELALT;    "     "  "  "   "   "  BMS "   "     "
APSIG   .WORD   SIGNO ;    "     "  "  "   "   " SIGNO
NUMERO  .WORD   0000  ;REGISTRO TEMPORAL DEL NO. DE MEDICIONES
DUMMY1  .WORD   0005  ;VARIABLE PARA INICIALIZAR CONTADOR 1 TIMER 1
MODO0   .BYTE   24    ;MODO DE OPERACION DEL CONTADOR 0
MODO1   .BYTE   70    ;MODO DE OPERACION DEL CONTADOR 1
MODO2   .BYTE   40    ;LECTURA CONTADOR 1 EN MOVIMIENTO

; FIN DE ZONA DE VARIABLES Y APUNTADORES
;---------------------------------------


        .END

; FIN DEL RUTINA EN ENSAMBLADOR
;******************************
```

```
(*****************************************)
(*                                       *)
(*    PROGRAMA PARA PROBAR EL FUNCIONAMIENTO  *)
(*    DEL TACOMETRO DIGITAL QUE SE EMPLEARA   *)
(*    EN LAS PRUEBAS DE LOS MOTORES DE PASOS  *)
(*                                       *)
(*          PROYECTO    2136             *)
(*             LAIL                      *)
(*          OCTUBRE 20,1982              *)
(*                                       *)
(*****************************************)

PROGRAM PRUEBA;

(* ZONA PARA DECLARACION DE CONSTANTES Y VARIABLES *)

CONST   MEDMAX = 500; (*NUMERO MAXIMO DE MEDICIONES QUE TOMARA EL TACOMETRO*)
        EVMAX = 4095; (*NUMERO MAXIMO DE EVENTOS POR MEDICION*)

TYPE    ARREGLO = ARRAY[1..MEDMAX] OF INTEGER;
                  (*TIPO DE ARREGLO PARA GUARDAR RESULTADOS*)

VAR     J: BOOLEAN; (*INDICADOR PARA LETREROS*)

        NMED,(*NUMERO DE MEDICIONES DESEADAS*)
        NEVENTOS,(*NUMERO DE EVENTOS POR MEDICION*)
        CONTO,(*ARGUMENTO QUE SE DECRECERA EN EL TIMER 0 *)
        CONT1,(*  "     "  "  "    "  "  "   1 *)
        ERFOR,(*INDICADOR DE EJECUCION DE LA RUTINA EN ENSAMBLADOR*)
        I: INTEGER;(*VARIABLE PARA ITERACIONES*)

        R: CHAR; (*VARIABLE DE CONTROL PARA ACEPTAR COMANDOS*)

        VELBAJ,(*ARREGLO CON LOS 16 BITS MAS SIGNIFICATIVOS DE LA MEDICION*)
        VELALT: ARREGLO;(*COMO VELBAJ PERO LOS BITS MENOS SIGNIFICATIVOS  *)

        VELDESEADA,(*VELOCIDAD PROGRAMADA EN EL EMULADOR *)
        VELREAL,(*    "       MEDIDA CON EL TACOMETRO *)
        VELERROR: REAL;(*ERROR EN LA VELOCIDAD *)

(* DECLARACION DE LA RUTINA EN ENSAMBLADOR *)

PROCEDURE MUESTRA;

EXTERNAL; (*SE DECLARA EXTERNA *)


(*INICIO PROGRAMA PRINCIPAL*)

BEGIN(*PRINCIPAL*)

  WRITELN('PROGAMA DE PRUEBA DEL TACOMETRO DIGITAL');

  CONTO:=32767;(*ASIGNA VALOR AL TIMER 0*)
  CONT1:=CONTO;(*   "       "   "   "  1*)

  (*INICIA RUTINA DE MEDICIONES*)

  REPEAT(*UNTIL R='S'*)
```

```
REPEAT(*NMED*)

  WRITE('CUANTAS MEDICIONES(MAX=500): ');
  READLN(NMED);
  IF NMED <>0 THEN NMED:=MEDMAX+1;

UNTIL NMED <=MEDMAX;(*NMED*)

REPEAT(*NEVENTOS*)

  WRITE('CUANTOS EVENTOS POR MEDICION(MAX=4096): ');
  READLN(NEVENTOS);
  NEVENTOS:=NEVENTOS-1;(*CORRIGE EL NO. DE EVENTOS*)
  IF NEVENTOS <>0 THEN NEVENTOS:=EVMAX+1;

UNTIL NEVENTOS <=EVMAX;(*NEVENTOS*)

WRITE('VELOCIDAD DESEADA(EV/S): ');(*PIDE VELOCIDAD DEL EMULADOR*)
READLN(VELDESEADA);

ERROR:=0;(*INICIALIZA INDICADOR DE ERROR*)
J:=TRUE;(*INDICADOR DE CABEZA DE PAGINA*)

(* RUTINA EN ENSAMBLADOR *)

MUESTRA;

(* FIN RUTINA EN ENSAMBLADOR *)

IF ERROR = 0 (*ERROR*)

  THEN(*ERROR*)

    FOR I:=1 TO NMED DO

      BEGIN(*FOR I*)

        IF J THEN WRITELN(' VELOCIDAD REAL   ERROR DE VEL.');

        IF I = 1 THEN VELREAL:=CONTO-VELALTI[1]
                 ELSE VELREAL:=VELALTI[I-1]-VELALTI[I];
        VELREAL:=1.0205E6*(NEVENTOS+1)/(1.0*ABS(VELREAL)*CONTO-
                + CONTO -VELBAJI[I] + 82);
        (*OBTIENE VELOCIDAD REAL*)

        VELERROR:=ABS(ABS(VELREAL)-VELDESEADA);

        WRITELN(VELREAL:15,VELERROR:15);

        IF I MOD 23 = 0 THEN J:=TRUE ELSE J:=FALSE;(*CABEZA DE PAGINA*)

      END(*FOR I*)

  ELSE(*ERROR*)

    BEGIN (*ELSE*)

      IF ERROR = 128(*ERROR=128*)
        THEN
          WRITELN('INTERRUPCION MUY RAPIDA')
        ELSE
          WRITELN('INTERRUPCION DE FUENTE DESCONOCIDA');

    END;(*ELSE*)

  WRITE('CONTINUO?(S/N): ');(*PREGUNTA SI CONCLUYEN LAS MEDICIONES*)
  READ(R);
  WRITELN('');

UNTIL R <>'S';(*UNTIL R <>'S'*)

END.(*FIN PROGRAMA PRINCIPAL*)
```

```
;*****************************************
;
; RUTINA EN ENSAMBLADOR PARA CONTROLAR EL TA-
; COMETRO DIGITAL
;
;              PROYECTO    2136
;              LAIL OCTUBRE  82
;
; ESTA RUTINA ES LLAMADA COMO UN PROCEDURE
; DEL PROGRAMA  PRUEBA.CODE
;
;*****************************************

        .PROC  MUESTRA

; ZONA PARA LA DECLARACION DE MACROINSTRUCCIONES
;-----------------------------------------------

; MACRO PARA OBTENER DOS BYTES DEL STACK Y GUARDARLOS EN UNA LOCALIDAD DADA

        .MACRO TOMA

        PLA     ;OBTIENE EL BPS
        STA %1  ;LO GUARDA

        PLA     ;OBTIENE EL BMS
        STA %1+1;LO GUARDA

        .ENDM

; MACRO PARA PONER DOS BYTES DE MEMORIA EN EL STACK

        .MACRO GUARDA

        LDA %1+1;OBTIENE BMS
        PHA     ;LO PONE EN EL STACK

        LDA %1  ;OBTIENE BPS
        PHA     ;LO PONE EN EL STACK

        .ENDM

; MACRO PARA CARGAR EL REGISTRO DE CONTROL DE LOS TIMERS

        .MACRO MODO

        LDA %2          ;CARGA MODO DE CONTROL
        STA %1+3        ;APUNTA AL REGISTRO DE CONTROL DEL TIMER EN CUESTION

        .ENDM

; MACRO PARA PROGRAMAR LOS RELOJES DEL TACOMETRO.
; ESCRIBE DOS BYTES SOBRE ALGUNO DE LOS CONTADORES
; (ES IMPORTANTE ESCRIBIR SIEMPRE LOS DOS BYTES)

        .MACRO TP

        LDA %2          ;CARGA BPS
        STA %1+%3       ;LO GUARDA EN EL CONTADOR

        LDA %2+1        ;CARGA BMS
        STA %1+%3       ;LO GUARDA EN EL CONTADOR

        .ENDM

; MACRO PARA SALVAR EN EL STACK EL VALOR DE LOS ACUMULADORES
; E INDICES AL OCURRIR INTERRUPCION

        .MACRO SALVA

        PHA     ;GUARDA ACUMULADOR
        TYA     ;PASA INDICE Y A ACUMULADOR
        PHA     ;SALVA INDICE Y

        .ENDM

; MACRO PARA RECUPERAR EL VALOR DEL ACUMULADOR E INDICES AL
; CONCLUIR LA INTERRUPCION

        .MACRO RECUPERA

        PLA     ;OBTIENE INDICE Y
        TAY
        PLA     ;OBTIENE ACUMULADOR

        .ENDM

; MACRO PARA PROGRAMAR LOS CONTADORES HACIA ARRIBA Y HACIA ABAJO

        .MACRO UDC

        LDA %1  ;CARGA BPS
        STA %2  ;LO GUARDA EN EL UDC #1
        STA %3  ;  "    "  "  "  "  #2

        LDA %1+1;CARGA BMS
        STA %2+1;LO GUARDA EN EL UDC #1(SOLO LOS 4 BITS MENOS SIGNIFICATIVOS)
        STA %3+1"    "  "  "  "  #2(  "   "  "  "     "       "  )

        .ENDM

; MACRO PARA LEER EL VALOR DE LOS TIMERS

        .MACRO LECTURA

        LDY #00         ;BORRA INDICE Y

        LDA %1+%2       ;LEE BPS DEL TIMER
        STA @%2,Y       ;LOS GUARDA EN TABLAS

        INY             ;INCREMENTA INDICE Y

        LDA %1+%3       ;LEE BMS DEL TIMER
        STA @%2,Y       ;LOS GUARDA

        CLC             ;ACTUALIZA APUNTADOR
                        ;SIGUE SI NO HAY CRUCE DE PAGINA
        LDA %2          ;ACTUALIZA EN CASO DE CRUCE
        ADC #02
```

```
        STA %2

        LDA %2+1
        ADC #00
        STA %2+1

        .ENDM

; MACRO PARA MOVER DOS BYTES

        .MACRO MUEVE

        LDA %1  ;CARGA BPS
        STA %2  ;LOS GUARDA

        LDA %1+1;CARGA BMS
        STA %2+1;LOS GUARDA

        .ENDM

* FIN DE LA ZONA DE MACROINSTRUCCIONES
;-------------------------------------

* ZONA PARA LA DECLARACION DE LAS VARIABLES DE LA RUTINA QUE SE COMPARTEN
; CON EL PROGRAMA EN PASCAL
;-------------------------------------------------------------------------

        .PUBLIC CONTO,CONT1;INDICADORES DEL VALOR A DECREMENTAR EN LOS TIMERS0
                           ;Y 1
        .PUBLIC NMED,NEVENTOS;NUMERO DE MEDICIONES Y DE EVENTOS POR MEDICION
        .PUBLIC ERROR;INDICADOR DE EJECUCION CORRECTA DE LA RUTINA

        .PUBLIC MEALT,MEBAJA;TABLAS PARA GUARDAR LOS VALORES LEIDOS DE LOS
                            ;TIMERS (SE LEEN 32 BITS)

;FIN DE LA ZONA DE VARIABLE PUBLICAS
;-----------------------------------

; ZONA PARA LA DECLARACION DE DIRECCIONES DE PERIFERICOS
* Y DE VARIABLES DE DIRECCION FIJA EN MEMORIA
;-----------------------------------------------------

RETORNO .EQU    00      ;DIRECCION DE RETORNO A PASCAL
INDIR1  .EQU    02      ;AUXILIAR PARA DIRECCIONAMIENTO INDIRECTO
INDIR2  .EQU    04      ; IDEM
APIRQ   .EQU OFFFE      ;DIRECCION DEL VECTOR DE INTERRUPCIONES

*                       IMPORTANTE: SE HA SUPUESTO QUE EL TACOMETRO DIGITAL
;                       SERA COLOCADO EN LA RANURA 4, CUALQUIER CAMBIO DE
*                       UBICACION PROVOCARA FUNCIONAMIENTO INCORRECTO.
;                       SI SE DESEA CAMBIAR LA RANURA SE DEBE CAMBIAR EL
;                       SEGUNDO DIGITO (C) DE LAS DIRECCIONES QUE SIGUEN POR
;                       EL ADECUADO A LA NUEVA RANURA (VER MANUAL APPLE)

TIM1    .EQU OC0C0      ;DIRECCION TIMER 1

UDC1    .EQU OC0C8      ;DIRECCION UDC 1
UDC2    .EQU OC0CC      ;   "     UDC 2

FFS1    .EQU OC0CA      ;DIRECCION PARA LEVANTAR FLIP-FLOP 1
FFS2    .EQU OC0CB      ;   "         "     "    FLIP-FLOP 2

FFR1    .EQU OC0CE      ;DIRECCION PARA BAJAR FLIP-FLOP 1
FFR2    .EQU OC0CF      ;   "       "   "  FLIP-FLOP-2

;FIN ZONA DECLARACION DE DIRECCIONES
;-----------------------------------

; INICIO DEL PROGRAMA EN ENSAMBLADOR
;-----------------------------------

PRIN:   TOMA RETORNO ;GUARDA DIRECCION DE REGRESO A PASCAL

APUN:   GUARDA APSERV;OBTIENE DIRECCION DE LA RUTINA DE SERV. A INTERRUPCIONES
        TOMA APIRQ   ;LA PASA AL VECTOR DE INTERRUPCIONES

PROG:   LDA NMED;CARGA BPS DE NUMERO DE MEDICIONES
        STA NUMERO; LOS PASA A NUMERO
        LDA NMED+1;LO MISMO PARA LOS BMS
        STA NUMERO+1

        MUEVE APBAJA,INDIR1;SALVA LA DIRECCION DE LAS TABLAS
        MUEVE APALTA,INDIR2;EN VARIABLES DE PAGINA CERO

        UDC NEVENTOS,UDC1,UDC2;PROGRAMA UDC CON NUMERO DE EVENTOS POR MEDICION

        STA FFR1        ;ASEGURA QUE LOS RELOJES ESTEN PARADOS

        MODO TIM1,MODO0         ;PROGRAMA MODO DE OPERACION TIMER 1 CONT 0
        MODO TIM1,MODO1         ;   "     "  "       "     "  " CONT 1

        TP TIM1,DUMMY1,0;PROGRAMA TIMER1 CONTADOR 0
        TP TIM1,CONT1,1;PROGRAMA TIMER1 CONTADOR 1

        STA FFS1 ;LEVANTA BANDERA DE CUENTA MINIMA EN UDC1

        NOP      ;RETRASO PARA COMPLETAR LA INICIALIZACION DE LOS CONTADORES 1
        NOP

        STA FFR1        ;BAJA BANDERA DE CUENTA MINIMA(PARA RELOJES TIMER 1)

        TP TIM1,CONT0,0;PROGRAMA TIMER 1 CONTADOR 0

        STA FFS1        ;ARRANCA RELOJES TIMER 1
        STA FFR1        ;LOS PARA DESPUES DE INICIALIZAR CONTADOR 0

        UDC NEVENTOS,UDC1,UDC2;PROGRAMA UDC CON NUMERO DE EVENTOS POR MEDICION

        STA FFS1        ;HABILITA RELOJES DESPUES DE INICIALIZACION
        STA FFS2

HAB:    LDA NEVENTOS    ;CARGA BPS DE NUMERO DE EVENTOS EN ACUMULADOR
        CLI     ;HABILITA INTERRUPCIONES AL MPU

LOOP:   LDA ERROR ;VERIFICA QUE NO OCURRIERON INTERRUPCIONES DE FUENTE NO
                  ;CONOCIDA
        BNE DESHAB;FINALIZA SI OCURRIERON
```

```
        LDA NUMERO+1;PREGUNTA SI YA CONCLUYO CON TODAS LAS MEDICIONES

        BNE LOOP  ;CONTINUA SI NO ES CIERTO

FINAL:  CLI       ;DESHABILITA INTERRUPCIONES AL MPU

        STA FFR1        ;PARA RELOJES
        STA FFR2

        GUARDA RETORNO; CARGA EN STACK LA DIRECCION DE RETORNO A PASCAL

        RTS       ;REGRESA A PASCAL

; FIN DE LA RUTINA EN ENSAMBLADOR
;------ -------------------------

; DEFINICION LA RUTINA DE SERVICIO A INTERRUPCIONES
;-------------------------------------------------

SERV:   STA UDC1        ;REPROGRAMA CPS DE UDC1
        STA UDC2        ;IDEM UDC2

        LDA NEVENTOS+1  ;AHORA LOS IMS
        STA UDC1+1
        STA UDC2+1

        STA FFR1        ;PARA TIMER 1

        LECTURA TIM1,INDIR1,0;LEE CONTENIDO CONTADOR 0

        STA FFS1        ;ARRANCA TIMER 1
        STA FFS2        ;ARRANCA TIMER 2

        MODO TIM1,MODO2;PROGRAMA LECTURA DE CONTADOR 1 EN MOV.

        LECTURA TIM1,INDIR2,1

        DEC NUMERO      ;DECREMENTA CPS DE NUMERO
        LDA #0FF
        CMP NUMERO      ;VERIFICA CAUCE DE PAGINA
        BNE SIGUE

        DEC NUMERO+1    ;DECREMENTA EN CASO DE CAUCE

SIGUE:  LDA NEVENTOS

        RTI       ;REGRESA DE INTERRUPCION INDICANDO ERROR

; FIN DE LA RUTINA DE SERVICIO A INTERRUPCIONES
;- ------------ -------  ----------------------

; ZONA DE DECLARACION DE VARIABLES CON VALOR INICIAL Y DE APUNTADORES
;------------------------------------------------------------------

APSERV  .WORD   SERV ;APUNTADOR A LA RUTINA DE SERVICIO A INTERRUPCIONES
APBAJA  .WORD   VELBAJ;APUNTADOR A LA TABLA DE LOS 16 CPS DE LAS MEDICIONES
APALTA  .WORD   VELALT;    "     "  "  "   "  "  "  IMS "   "      "
NUMERO  .WORD   0000  ;REGISTRO TEMPORAL DEL NO. DE MEDICIONES

DUMMY1  .WORD   0006  ;VARIABLE PARA INICIALIZAR CONTADOR 1 TIMER 1
TESTIGO .BYTE   0FF   ;TESTIGO DE SERVICIO A INTERRUPCIONES
MODO0   .BYTE   34    ;MODO DE OPERACION DEL CONTADOR 0
MODO1   .BYTE   76    ;MODO DE OPERACION DEL CONTADOR 1
MODO2   .BYTE   40    ;LECTURA CONTADOR 1 EN MOVIMIENTO

; FIN DE ZONA DE VARIABLES Y APUNTADORES
;--------------------------------------


        .END

; FIN DEL RUTINA EN ENSAMBLADOR
;*****************************
```

```
{*RELOJ.TEXT*}
{*****************************************************}
{*                                                   *}
{*      PROGRAMA PARA PROBAR EL RELOJ QUE GENERA LOS  *}
{*      TIEMPOS ENTRE PULSOS.                        *}
{*                                                   *}
{*      EL OBJETO DEL PROGRAMA ES GENERAR PULSOS BASADO *}
{*      EN UN ARCHIVO DE DATOS CON EL FORMATO DEL ARCHI-*}
{*      VO TIEMPOS.DATA. LOS PULSOS DEBERAN VERSE EN UN *}
{*      OSCILOSCOPIO.                                *}
{*                                                   *}
{*              LUIS ALVAREZ ICAZA LONGORIA          *}
{*              PROYECTO 2136       II,UNAM          *}
{*                                                   *}
{*****************************************************}

PROGRAM RELOJ;

{*-------------------------------------------------*}
{* ZONA DE DECLARACION DE VARIABLES Y CONSTANTES *}
{*-------------------------------------------------*}

CONST   NREGMAX = 1000;     (* NO. MAXIMO DE PULSOS A ENVIAR *)

{*---------*}

VAR     MASCARA: PACKED ARRAY [1..NREGMAX] OF 0..255; (* PULSO CODIFICADO *)
        FACTOR : PACKED ARRAY [1..NREGMAX] OF 0..255; (* ESCALA DE TIEMPO *)
        TIEMPO :        ARRAY [1..NREGMAX] OF INTEGER; (*TIEMPO DEL PULSO *)

        I,J, (*INDICES PARA ITERACIONES*)
        NUMREG, (*NO. DE REGISTROS *)
        ERROR  (*INDICADOR DE ERROR EN LA EJECUCION*)

        : INTEGER;

        R,S: CHAR;(* VARIABLES PARA INTERROGAR *)

        AE1: FILE OF INTEGER;(*ARCHIVOS DE ENTRADA*)

        RESPUESTA: SET OF 0..245;

{*---------*}

{* PROCEDIMIENTO EXTERNO QUE CONTIENE LOS PROGRAMAS *}
{* PARA CONTROLAR EL ACOPLADOR DE ENVIO DE PULSOS   *}

PROCEDURE PRUEBELOJ;

EXTERNAL; (* SE DECLARA EXTERNO *)

{*---------*}

{* PROCEDIMIENTO PARA LEER LOS DATOS DE LOS ARCHIVOS EN DISQUETE *}
{* EL ARCHIVO ESTA DIVIDIDO EN REGISTROS, CADA UNO CONTIENE DOS  *}
{* VARIABLES ENTERAS, LA PRIMERA INDICA EL TIEMPO ENTRE PULSOS Y *}
{* LA SEGUNDA EL FACTOR DE ESCALAMIENTO DE TIEMPO Y LA MASCARA   *}
{* PARA EL ENVIO DEL PULSO POR EL ACOPLADOR PARALELO.            *}

PROCEDURE LECTURA;

BEGIN {*LECTURA*}

  RESET(AE1,'#5:TIEMPOS.DATA'); (*ABRE ARCHIVO*)
  NUMREG:=AE1^; (*OBTIENE NO. DE REGISTROS DEL ARCHIVO*)

  IF NUMREG < NREGMAX

   THEN

     BEGIN(*THEN NUMREG*)

      FOR I:=1 TO NUMREG DO

       BEGIN(*FOR I*)

        GET(AE1);
        TIEMPO[I]:=AE1^;

        GET(AE1);
        J:=AE1^DIV 256;(*TOMA 8 BMS*)
        J:=J MOD 4;(*TOMA FACTOR DE ESCALA*)
        FACTOR[I]:=J;

        J:=AE1^ MOD 256;(*TOMA 8 BPS*)
        J:=J MOD 64;(*TOMA MASCARA*)
        MASCARA[I]:=J;

       END;(*FOR I*)

       CLOSE(AE1);(*CIERRA ARCHIVO DATOS*)

     END(*THEN NUMREG*)

    ELSE

     BEGIN(*ELSE NUMREG*)

      WRITELN('DEMASIADOS REGISTROS EN EL ARCHIVO');
      CLOSE(AE1);(*CIERRA ARCHIVO DATOS*)
      EXIT(PROGRAM);(*ABORTA EJECUCION*)

     END;(*ELSE NUMREG*)

END;(*LECTURA*)

{*---------*}

{* PROCEDIMIENTO QUE DESPLIEGA LOS DATOS DE LAS MEDICIONES*}

PROCEDURE DESPLIEGA;

BEGIN(*DESPLIEGA*)

  WRITELN('LOS VALORES ACTUALES PARA LAS');
  WRITELN('MEDICIONES SON:');
  WRITELN('');

  WRITELN('NO. DE REGISTROS: ',NUMREG);
```

```
END;(*DESPLIEGA*)

(*---------*)

(*PROCEDIMIENTO PARA MODIFICAR ALGUN PARAMETRO*)

PROCEDURE MODIFICA;

    (*PROCEDIMIENTO LOCAL PARA FACILITAR LOS CAMBIOS*)

    PROCEDURE CORRIGE(N,FACTOR: INTEGER;VAR M:INTEGER);

    VAR A:REAL;

    BEGIN (*CORRIGE*)

      WRITE('VALOR: ');(*PIDE VALOR*)
      READLN(A);(*LEE VALOR CORREGIDO*)
      A:=TRUNC(A)*FACTOR;(*LO ESCALA*)

      IF (A<=0) OR (A>N)

        THEN(*IF A*) WRITELN('VALOR FUERA DE RANGO')
        ELSE(*IF A*) M:=TRUNC(A);

    END;(*CORRIGE*)

BEGIN(*MODIFICA*)

 REPEAT

  BEGIN(*REPEAT R*)

   DESPLIEGA;(*VALORES ACTUALES*)
   WRITE('QUIERES CAMBIAR ALGUNO(S/N)? ');
   READ(R); WRITELN;

   IF R = 'S'

    THEN

     BEGIN(*THEN R*)

      WRITE('NO. REGISTROS      =R');
      READ(S); WRITELN(' ');

      CASE S OF

       'R': CORRIGE(NREGMAX,1,NUMREG);

      END;(*CASE S*)

     END;(*THEN R*)

  END;(*REPEAT R*)

UNTIL R = 'N';

END;(*MODIFICA*)

(*---------*)

(* PROCEDIMIENTO PARA TOMAR LOS DATOS NECESARIOS PARA LA EJECUCION *)

PROCEDURE DATOS;

BEGIN(*DATOS*)

  LECTURA;
  MODIFICA;

END;(*DATOS*)

(*---------*)

(*PROCEDIMIENTO PARA EL MANEJO DEL RESULTADO DE LA PRUEBA*)

PROCEDURE RESULTA;

BEGIN(*RESULTA*)

 RESPUESTA:=[0,63];

 IF ERROR IN RESPUESTA

  THEN

    CASE ERROR OF

     0 : WRITELN('CORRIDA NORMAL, CONCLUIDA!!!!!!!');
     63: WRITELN('INTERRUPCION DE FUENTE DESCONOCIDA');

    END;(*CASE ERROR*)

  ELSE

    WRITELN('ERROR NO IDENTIFICADO, REPITA');

END;(*RESULTA*)

(*---------*)

(* PRINCIPIO DEL PROGRAMA PRINCIPAL*)

BEGIN(*CONTROL*)

  DATOS;(*TOMA DATOS PARA EJECUCION*)

  ERROR:=0;(*INICIALIZA INDICADOR DE ERROR*)

  PRUERELOJ;(*RUTINA EN ENSAMBLADOR*)

  RESULTA;(*ANALIZA RESULTADOS*)

END.(*CONTROL*)
```

```
*PRUERELOJ,TEXT
;-------------------------------------------------
;
;   PROGRAMA PARA REALIZAR PRUEBAS DEL RELOJ QUE
;   GENERA LOS PULSOS PARA LOS MOTORES DE PASOS.
;
;   ESTE PROGRAMA FUNCIONA COMO UNA SUBRUTINA DEL
;   PROGRAMA RELOJ,TEXT. CONSTA DE TRES PARTES:
;       -INICIALIZACION
;       -SERVICIO AL RELOJ PARA ENVIO DE PULSOS
;   SUS TAREAS MAS IMPORTANTES SON:
;       -PROGRAMACION INICIAL DEL RELOJ
;       -ATENCION A LA INTERRUPCIONES PROVOCADAS POR
;        EL RELOJ
;       -REPROGRAMACION DESPUES DE LAS INTERRUPCIONES.
;       -ENVIO DE PULSOS A LOS MOTORES DE PASOS.
;
;       LUIS ALVAREZ ICAZA LONGORIA, PROYECTO 2136
;       INSTITUTO DE INGENIERIA, UNAM.
;
;-------------------------------------------------

        .PROC PRUERELOJ
;-----------------------
;-----------------------
;
; INCLUYE LAS MACROINSTRUCCIONES
;-------------------------------

        .INCLUDE        #S:MACROREL.TEXT
;
;-------------------------------------------------
;
; DECLARACION DE VARIABLES COMUNES CON EL PROGRAMA EN PASCAL
;
;-------------------------------------------------
;
        .PUBLIC         NUMREG,ERROR
                        ;NO. DE REGISTROS E INDICADOR DE ERROR
        .PUBLIC         MASCARA,FACTOR;TABLAS QUE CONTIENEN LA MASCARA PARA EL
        .PUBLIC         TIEMPO          ;ENVIO DE PULSOS, EL FACTOR DE DIVISION
                                        ;PARA EL PRIMER RELOJ, EL TIEMPO ENTRE
                                        ;PULSOS ESCALADO SEGUN EL FACTOR ANTERIOR

; FIN DE LA ZONA DE VARIABLES PUBLICAS
;-------------------------------------

; ZONA PARA DECLARACION DE DIRECCIONES DE PERIFERICOS Y DE VARIABLES
; DE DIRECCION FIJA EN MEMORIA
;-------------------------------------------------------------

RETORNO .EQU    00      ;DIRECCION DE RETORNO A PASCAL
INDMAS  .EQU    02      ;AUXILIAR PARA TABLA DE MASCARAS
INDFAC  .EQU    04      ;IDEM TABLA DE FACTORES
INDTIE  .EQU    06      ;IDEM TABLA DE TIEMPOS
APIRO   .EQU OFFFE      ;DIRECCION DEL VECTOR DE INTERRUPCIONES

;  ***IMPORTANTE***     SE HA SUPUESTO LA SIGUIENTE QUE EL TIMER ESTARA
;                       COLOCADO EN LA RANURA 3.
;
```

```
;!!!CUALQUIER CAMBIO !!!DE UBICACION PROVOCARA FUNCIONAMIENTO INCORRECTO.
;                       PARA CAMBIAR DE UBICACION SERIA NECESARIO
;                       MODIFICAR LAS DIRECCIONES SEGUN LAS INDICACI-
;                       CIONES DEL MANUAL DE REFERENCIA DE LA APPLE II.

; LOCALIDAD PARA MANEJAR EL RELOJ QUE CALCULA EL TIEMPO ENTRE PULSOS

RELOJ   .EQU    0C0D0   ;DIRECCION DEL RELOJ

; LOCALIDAD PARA EL ACOPLADOR DE ENVIO DE PULSOS

PULNO   .EQU    0C058   ;DIRECCION DEL ANUNCIADOR CERO SALIDA CERO
PULSI   .EQU    0C059   ;IDEM SALIDA UNO

;LOCALIDADES PARA CONTROL DE PARO Y ARRANQUE DEL BANCO DE PRUEBAS

PARO    .EQU    0C062   ;DIRECCION DE ENTRADA PARA PARO FORZOSO
ARRAN   .EQU    0C061   ;DIRECCION DE ENTRADA PARA INICIO DE PRUEBAS

; FIN DE LA ZONA DE LOCALIDADES FIJAS DE MEMORIA
;-----------------------------------------------
;
;  INICIO DEL PROGRAMA PRINCIPAL
;
;--------------------------------
;--------------------------------

PRIN:   TOMA    RETORNO ;GUARDA DIRECCION PARA REGRESAR A PASCAL

;----------  --------

APUN:   GUARDA  APSERV  ;OBTIENE DIR. RUTINA SERV. A INTERRUPCIONES
        TOMA    APIRO

;--------------------

; RUTINAS DE INICIALIZACION DE LOS PERIFERICOS
;---------------------------------------------

INIC:   LDY #00 ;INICIALIZA ANUNCIADORES(AMBOS ALTOS)
        STA PULSI,Y
        INY
        INY
        STA PULSI,Y

; SALVA VALORES DE LOS APUNTADORES A LAS TABLAS Y DATOS DE LA EJECUCION

        MUEVE NUMREG,NOREG
        MUEVE APFACT,INDFAC
        MUEVE APMASC,INDMAS
        MUEVE APTIEM,INDTIE

; PROGRAMA TIMER

        INITRES 1  ;RESET INTERNO AL TIMER(RELOJES PARADOS)

        LEEFACTOR INDFAC,FACACT      ;LEE EL PRIMER FACTOR
        LEEMASCARA INDMAS,MASACT     ; "  LA PRIMERA MASCARA
        LEETIEMPO INDTIE,TIEACT      ; "  "  "     TIEMPO
```

```
        QUITA NOREG     ;ACTUALIZA EL NO. DE REGISTROS POR LEER

        LDA MODOI       ;PROGRAMA RELOJ 2 APUNTANDO AL RELOJ 3
        STA RELOJ+1

        PROGRAMA FACACT,TIEACT ;PROGRAMA TIMERS 2 Y 3 SEGUN FACTOR

        INTRES 1        ;PRECARGA CONTADORES(RELOJES AUN PARADOS)

; ESPERA QUE SE DE LA ORDEN DE INICIO DE PRUEBAS (CONTROLADA POR UN
; INTERRUPTOR EN EL BANCO DE PRUEBAS
;-------------------------------------------------------------------

ESPERA: LDA ARRAN               ;VERIFICA ESTADO DEL INTERRUPTOR
        BFL ESPERA              ;ESPERA A LA ORDEN DE ARRANQUE

; HABILITA INTERRUPCIONES
;------------------------

HAB:    INTRES 0                ;ARRANCA RELOJ
        LDA MODOI               ;PROGRAMA RELOJ 2 APUNTANDO AL RELOJ 3
        STA RELOJ+1

        CLI                     ;HABILITA INTERRUPCIONES AL MPU

; VERIFICA FIN DE EJECUCION
;--------------------------

LOOP:   LDA ERROR       ;VERIFICA ALGUN ERROR DE EJECUCION
        BNE DESHAB      ;FINALIZA SI OCURRIO ALGUNO

        LDA PARO        ;VERIFICA SENAL DE PARO FORZOSO
        BMI DESHAB      ;FINALIZA SI ESTA PUESTA

        LDA NOREG+1     ;VERIFICA SI HAY INFORMACION POR ENVIAR
        BFL LOOP        ;FINALIZA SI NO LA HAY

; DESHABILITA INTERRUPCIONES
;---------------------------

DESHAB: SEI             ;DESHABILITA INT. AL MPU

        INTRES 1        ;PARA RELOJ PROGRAMABLE

;REGRESA A PASCAL
;----------------

        GUARDA RETORNO  ;DIRECCION DE RETORNO AL STACK

        RTS             ;REGRESA A PASCAL

; FIN DEL PROGRAMA PRINCIPAL EN ENSAMBLADOR
;------------------------------------------
;------------------------------------------


; RUTINA DE SERVICIO A INTERRUPCIONES
;------------------------------------
;------------------------------------

SERV:   PHA             ;SALVA ACUMULADOR

; VERIFICA LA FUENTE DE INTERRUPCION
;-----------------------------------

VEREL:  LDA RELOJ+1     ;CARGA STATUS DEL RELOJ
        BMI SEREL       ;SIRVE AL RELOJ

; LEVANTA BANDERA POR INTERRUPCION NO CONOCIDA
;---------------------------------------------

DESCO:  LDA #3F         ;ERROR: FUENTE DESCONOCIDA DE INTERRUPCION
        STA ERROR

        PLA             ;RECUPERA ACUMULADOR
        RTI             ;FIN DE INTERRUPCION

;SERVICIO AL RELOJ
;-----------------

SEREL:  INTRES 1        ;PARA RELOJES E INHABILITA INTERRUPCIONES
        LDA MODOI       ;APUNTA AL RELOJ 3
        STA RELOJ+1

        LDY MASACT      ;CARGA MASCARA
        STY MASANT      ;GUARDA MASCARA
        STA PULNO,Y     ;MANDA PARTE BAJA DE PULSOS SEGUN MASCARA

        LEEFACTOR INDFAC,FACACT         ;LEE FACTOR
        LEEMASCARA INDMAC,MASACT        ; "  MASCARA
        LEETIEMPO INDTIC,TIEACT         ; "  TIEMPO

        PROGRAMA FACACT,TIEACT ;PROGRAMA TIMERS 2 Y 3 SEGUN FACTOR

        INTRES 1        ;CARGA CONTADORES A NUEVO VALOR
        INTRES 0        ;ARRANCA RELOJES
        LDA MODOI       ;APUNTA AL RELOJ 3
        STA RELOJ+1

        LDY MASANT      ;TERMINA DE FORMAR PULSO
        STA PULSI,Y

        QUITA NOREG     ;ACTUALIZA EL NO. DE REGISTROS POR LEER

        PLA     ;RECUPERA ACUMULADOR
        RTI     ;FIN DE INTERRUPCION

; FIN DE LA RUTINA DE SERVICIO A INTERRUPCIONES
;----------------------------------------------
;----------------------------------------------

; ZONA PARA LA DECLARACION DE VARIABLES CON VALOR INICIAL Y DE APUNTADORES
;-------------------------------------------------------------------------

APSERV  .WORD   SERV    ;APUNTADOR A LA RUTINA DE SERV. A INT.
APFACT  .WORD   FACTOR  ; "     "  " TABLA DE FACTORES DE ESCALA
APMASC  .WORD   MASCARA ; "     "  "  "   "  MASCARAS PARA PULSOS
APTIEM  .WORD   TIEMPO  ; "     "  "  "   "  TIEMPOS
TIEACT  .WORD   0000    ;VALOR ACTUAL DE TIEMPO
FACACT  .BYTE   0000    ; "     "  " FACTOR
```

```
MASACT  .BYTE   OFF     ; "     "     " LA MASCARA
MASANT  .BYTE   OFF     ; "   ANTERIOR " "     "
MODO1   .BYTE    06     ;MODO DE OPERACION DEL RELOJ2 APUNTANDO AL 3
MODO2   .BYTE    87     ; "   "     "     "   " 2 APUNTANDO AL 1
MODO3   .BYTE   0C0     ; "   "     "     "   " 3 (SINCRONIA RELOJ2)
MODO4   .BYTE   0C2     ; "   "     "     "   " 3 (SINCRONIA APPLE)
DIEZ1   .WORD   0104    ;EQUIVALENTE A 10 DEC.HACIENDO (DMS+1)*(DPS+1)
DIEZ2   .WORD   1803    ;EQUIVALENTE A 100 DEC. IDEM
DIEZ3   .WORD  0F903    ;EQUIVALENTE A 1000 DEC. IDEM
NUMREG  .WORD   0000    ; "     "     "   NUMREG

;FIN DE LA ZONA DE VARIABLES Y APUNTADORES
;----------------------------------------

        .END

; FIN DEL PROGRAMA EN ENSAMBLADOR
;-------------------------------
*-------------------------------

;MACROREL.TEXT
;-- ------------------------------------------------------ ------------------
;
; MACROINSTRUCCIONES DEL PROGRAMA PARA REALIZAR LAS PRUEBAS DEL RELOJ
; QUE ENVIA LOS PULSOS A LOS MOTORES DE PASOS.
;
; ESTAS MACROINSTRUCCIONES SON INCERTADAS EN EL PROGRAMA PRUERELOJ.TEXT
; CUANDO ESTE ES ENSAMBLADO
;
;    LUIS ALVAREZ ICAZA LONGORIA , PROYECTO 2130
;    INSTITUTO DE INGENIERIA, UNAM
;
;-- ------------------------------------------------------------ --------

; MACRO PARA OBTENER DOS BYTES DEL STACK Y GUARDARLOS EN UNA LOCALIDAD DADA

        .MACRO TOMA

        PLA     ;OBTIENE EL DPS
        STA %1  ;LO GUARDA EN DPS DE LOCALIDAD

        PLA     ;OBTIENE EL DMS
        STA %1+1;LO GUARDA EN DMS DE LOCALIDAD

        .ENDM

; MACRO PARA PONER DOS BYTES DE MEMORIA EN EL STACK

        .MACRO GUARDA

        LDA %1+1;OBTIENE DMS
        PHA     ;LO PASA AL STACK

        LDA %1  ;OBTIENE DPS
        PHA     ;AL STACK

        .ENDM

; MACRO PARA MOVER DOS BYTES CONSECUTIVOS A OTRA LOCALIDAD

        .MACRO MUEVE

        LDA %1          ;CARGA DPS
        STA %2          ;LOS MUEVE

        LDA %1+1        ;CARGA DMS
        STA %2+1        ;LOS MUEVE

        .ENDM

;MACRO PARA INCREMENTAR EL VALOR DE UNA VARIABLE EN LA UNIDAD

        .MACRO SUM1

        INC %1  ;INCREMENTA EL VALOR DE LA VARIABLE

        BNE $01 ;PREGUNTA POR CRUCE DE PAGINA

        INC %1+1;ACTUALIZA SI OCURRIO EL CRUCE

$01     NOP     ;DE LO CONTRARIO CONTINUA

        .ENDM

;MACRO PARA INCREMENTAR EL VALOR DE UNA VARIABLE DE 16 BITS EN DOS UNIDADES

        .MACRO SUM2

        CLC     ;BORRA CARRY

        LDA %1  ;CARGA DPS DE ARGUMENTO
        ADC #02 ;LOS INCREMENTA
        STA %1  ;LOS SALVA

        LDA %1+1;CARGA AHORA DMS
        ADC #00 ;LOS ACTUALIZA EN CASO DE CRUCE DE PAGINA
        STA %1+1;LOS SALVA

        .ENDM

; MACRO PARA OBTENER EL DIVISOR DE FRECUENCIA DEL SEGUNDO RELOJ

        .MACRO LEEFACTOR

        LDY #00         ;BORRA INDICE Y

        LDA @%1,Y       ;CARGA VALOR DE TABLAS

        AND #03         ;CONSERVA LOS DOS PRIMEROS BITS

        STA %2          ;LOS GUARDA

        SUM1 %1         ;ACTUALIZA APUNTADOR (DE 1 EN 1 BYTE)

        .ENDM

; MACRO PARA LEER EL VALOR A CARGAR AL SEGUNDO CONTADOR

        .MACRO LEETIEMPO
```

```
        LDA @%1,Y       ;CARGA BPS DEL TIEMPO
        STA %2          ;LOS GUARDA (Y=0 EN LEEFACTOR)

        INY             ;INCREMENTA INDICE Y

        LDA @%1,Y       ;IDEM PARA LOS BMS
        STA %2+1

        SUM2 %1         ;ACTUALIZA APUNTADOR (DE 2 EN 2 BYTES)

        .ENDM

; MACRO PARA LEER LA MASCARA QUE INDICA EL MOTOR AL CUAL ENVIAR EL PULSO

        .MACRO LEEMASCARA

        LDA @%1,Y       ;CARGA VALOR DE TABLAS (Y=0 EN LEEFACTOR)

        STA %2          ;GUARDA VALOR LEIDO

        SUM1 %1         ;ACTUALIZA APUNTADOR (DE 1 EN 1 BYTE)

        .ENDM

; MACRO PARA DAR RESET INTERNO AL TIMER

        .MACRO INTRES

        LDA MODO2       ;PROGRAMA MODO DE OPERACION
        STA RELOJ+1     ;APUNTA A RELOJ 1

        LDA #%1         ;PARA RELOJES(RESET INTERNO)
        STA RELOJ       ;O LOS ARRANCA

        .ENDM

; MACRO PARA PROGRAMAR LOS RELOJES 2 Y 3 SEGUN FACTOR

        .MACRO PROGRAMA

        LDA %1  ;PASA FACTOR A ACUMULADOR

        BEQ $01 ;PREGUNTA POR PRIMER FACTOR

        CMP #01 ;PREGUNTA POR SEGUNDO FACTOR
        BEQ $02

        CMP #02 ;   "       "  TERCER     "
        BEQ $03

        ; PROGRAMA PARA FACTOR=1E3

        LDA MODO3       ;PROGRAMA MODO DE OPERACION RELOJ 3
        STA RELOJ

        LDA DIEZ3+1     ;CARGA BMS DE FACTOR
        STA RELOJ+4
        LDA DIEZ3       ;   "   BPS
        STA RELOJ+5

        JMP $04

        ;PROGRAMA FACTOR = 1E0 ;

$01     LDA MODO4       ;PROGRAMA RELOJ 3
        STA RELOJ

        JMP $04

        ;PROGRAMA FACTOR = 1E1

$02     LDA MODO3       ;PROGRAMA RELOJ 3
        STA RELOJ

        LDA DIEZ1+1     ;CARGA BMS DE FACTOR(RELOJ2)
        STA RELOJ+4
        LDA DIEZ1       ;IDEM BPS
        STA RELOJ+5

        JMP $04

        ;PROGRAMA FACTOR = 1E2

$03     LDA MODO3       ;PROGRAMA RELOJ 3
        STA RELOJ

        LDA DIEZ2+1     ;CARGA BMS DE FACTOR(RELOJ2)
        STA RELOJ+4
        LDA DIEZ2       ;   "   BPS
        STA RELOJ+5

        ;CARGA RELOJ 3

$04     LDA %2+1        ;CARGA BMS DE TIEMPO (RELOJ3)
        STA RELOJ+6
        LDA %2          ;   "   BPS
        STA RELOJ+7

        .ENDM

; MACRO PARA DECREMENTAR UNA NUMERO DE 16 BITS EN UNO DE OCHO

        .MACRO QUITA

        SEC             ;LEVANTA CARRY
        LDA %1          ;CARGA BPS
        SBC #01         ;DECREMENTA EN EL NO. DESEADO

        STA %1          ;GUARDA RESULTADO

        LDA %1+1        ;CARGA BMS
        SBC #00         ;DECREMENTA EN CASO DE CRUCE DE PAGINA

        STA %1+1        ;GUARDA RESULTADO

        .ENDM

; FIN DEL ARCHIVO DE MACROINSTRUCCIONES
;-------------------------------------
```

```
(*GRAFREAL.TEXT*)
(*$+*)

(* PROGRAMA PARA GRAFICAR LAS CURVAS QUE RESULTAN DE LAS *)
(* MEDICIONES EN EL BANCO DE PRUEBAS DEL COMPORTAMIENTO  *)
(* DE LOS MOTORES DE PASOS.                              *)

(*              LUIS  ALVAREZ ICAZA  LONGORIA            *)
(*              PROYECTO 2136.    ENERO  1984            *)
(*              INSTITUTO DE INGENIERIA, UNAM            *)

PROGRAM GRAFREAL;

(*DECLARA LIBRERIAS*)

USES TRANSCEND,TURTLEGRAPHICS,GRAFICASXY;

CONST   MUESTRA = 5;(*PERIODO DE MUESTREO PARA CONDICIONES DESEADAS*)
        MUESTMAX=60;(*MAXIMO NO. DE MUESTRAS PARA COND. DESEADAS*)
        RECMAX =3E2;(*MAXIMO NO. DE REGISTROS*)

(*ZONA DE VARIABLES*)

VAR     ORD,TIEMPO:ARREGLODATOS;(*ARREGLOS PARA GRAFICACION*)

        ORDM,TIEMPOM:ARRAY[1..MUESTMAX]OF REAL;(*ARREGLOS PARA MUESTRAS*)

        UNORD,(*UNIDADES DE LAS ORDENADAS*)
        ROT:STRING[40];(*ROTULO DE LA GRAFICA*)

        R:CHAR;(*AUXILIAR PARA RESPUESTAS*)

        FACTOR,(*FACTOR PARA LECTURA*)
        ACUM:REAL;(*AUXILIAR PARA LECTURA*)

        I,J,(*AUXILIARES*)
        NUMREG,(*NO. DE REGISTROS*)
        REGIN,(*REGISTRO INICIAL DE GRAFICACION*)
        REGFIN:INTEGER;(*REGISTRO FINAL*)

(* PROCEDIMIENTO PARA LECTURA *)

SEGMENT PROCEDURE UNO;

(*ARCHIVO DE ENTRADA*)

VAR     C1,C2,C3,C4,(*PARAMETROS PARA ACELERACION ACTUAL*)
        C1P,C2P,C3P,C4P,(*IDEM PARA ACELERACION SIGUIENTE*)

        OMEGA,TETA,INTERVALO,(*VALORES INICIALES PARA CONDICIONES DESEADAS*)
        OMEGAP,TETAP,SIGUIENTE,(*IDEM PARA EL SIGUIENTE INTERVALO*)
        DELTA (*TIEMPO EFECTIVO PARA LAS CONDICIONES DESEADAS*)
            :REAL;

        AE1: FILE OF REAL;

(*PROCEDIMIENTO PARA LECTURA DE CONDICIONES DESEADAS*)

  PROCEDURE LECTURA;

    BEGIN(*LECTURA*)

     (*ACELERACION*)
     C1P:=AE1^;GET(AE1);C2P:=AE1^;GET(AE1);
     C3P:=AE1^;GET(AE1);C4P:=AE1^;GET(AE1);

     (*VELOCIDAD,POSICION Y TIEMPO*)
     OMEGAP:=AE1^;GET(AE1);TETAP:=AE1^;GET(AE1);
     GET(AE1);SIGUIENTE:=AE1^;GET(AE1);

    END;(*LECTURA*)

(*PROCEDIMIENTO PARA MOVER ARGUMENTOS DE CONDICIONES DESEADAS*)

 PROCEDURE MUEVE;

  BEGIN(*MUEVE*)

   C1:=C1P;C2:=C2P;C3:=C3P;C4:=C4P;(*MUEVE ARGUMENTOS DE ACELERACION*)

   OMEGA:=OMEGAP;TETA:=TETAP;INTERVALO:=SIGUIENTE;(*MUEVE LOS DEMAS*)

   LECTURA;(*LEE NUEVOS ARGUMENTOS*)

  END;(*MUEVE*)

 (*FUNCION PARA CALCULAR LA POSICION*)
 FUNCTION POSICION(TIEMPO:REAL):REAL;

  BEGIN(*POSICION*)

   IF TIEMPO <= SIGUIENTE THEN MUEVE;(*LEE NUEVO REGISTRO DE NECESITARSE*)
   DELTA:=TIEMPO-INTERVALO;(*TIEMPO DENTRO DEL INTERVALO*)
   POSICION:=TETA+(OMEGA+C3/C4)*DELTA+C1*SQR(DELTA)/2+C2*SQR(DELTA)*DELTA/6-
             C3/SQR(C4)*SIN(C4*DELTA);
  END;(*POSICION*)

 (*FUNCION PARA CACULAR LA VELOCIDAD*)
 FUNCTION VELOCIDAD(TIEMPO:REAL):REAL;

  BEGIN(*VELOCIDAD*)

   IF TIEMPO <= SIGUIENTE THEN MUEVE;(*LEE NUEVO REGISTRO DE NECESITARSE*)
   DELTA:=TIEMPO-INTERVALO;(*TIEMPO DENTRO DEL INTERVALO*)
   VELOCIDAD:=OMEGA+C3/C4*C1*DELTA+C2*SQR(DELTA)/2-C3/C4*SIN(C4*DELTA);

  END;(*VELOCIDAD*)

 BEGIN(*UNO*)

  (*PREGUNTA POR TIPO DE GRAFICA*)

  WRITE('GRAFICA DE V)ELOCIDAD O P)OSICION? ');READ(R);WRITELN;

  (*ABRE ARCHIVO Y OBTIENE NO. DE REGISTROS*)
  RESET(AE1,'#5:VELOCIDAD.DATA');
  NUMREG:=TRUNC(AE1^);GET(AE1);GET(AE1);(*NO. DE REGISTROS*)

  WRITELN('EXISTEN: ',NUMREG,' REGISTROS');
  (*PREGUNTA POR REGISTROS INICIALES Y FINALES*)
```

```
WRITE('REGISTRO INICIAL? '); READLN(REGIN);
WRITE('REGISTRO FINAL? '); READLN(REGFIN);

(*OBTIENE FACTOR PARA LECTURAS*)
IF (REGFIN-REGIN) > REGMAX
 THEN FACTOR:=1
 ELSE FACTOR:=1.0*(REGFIN-REGIN)/REGMAX;

I:=(REGIN*3-1); (*REGISTRO INICIAL EN EL ARCHIVO*)
SEEK(AE1,I); GET(AE1); (*MUEVE APUNTADOR*)
ACUM:=REGIN; (*INICIALIZA ACUMULADOR PARA LECTURAS*)

TIEMPO^[1]:=AE1^; GET(AE1); (*VALORES INICIALES*)
IF R = 'P'
 THEN BEGIN (*THEN R*) ORD^[1]:=AE1^; GET(AE1); GET(AE1); END (*THEN R*)
 ELSE BEGIN (*ELSE R*) GET(AE1); ORD^[1]:=AE1^; GET(AE1); END; (*ELSE R*)

FOR J:=2 TO TRUNC((REGFIN-REGIN)/FACTOR) DO

 BEGIN (*FOR J*)

   (*NO. DEL SIGUIENTE REGISTRO A LEER*)
   ACUM:=ACUM+FACTOR; I:=TRUNC(ACUM)*3-1;
   SEEK(AE1,I); GET(AE1);

   TIEMPO^[J]:=AE1^; GET(AE1); (*VALORES INICIALES*)
   IF R = 'P'
    THEN BEGIN (*THEN R*) ORD^[J]:=AE1^; GET(AE1); GET(AE1); END (*THEN R*)
    ELSE BEGIN (*ELSE R*) GET(AE1); ORD^[J]:=AE1^; GET(AE1); END; (*ELSE R*)

 END; (*FOR J*)

CLOSE(AE1); (*CIERRA ARCHIVO DE ENTRADA*)


RESET(AE1,'B5:CURVAS.DATA'); (*ABRE ARCHIVO PARA CONDICIONES DESEADAS*)
GET(AE1);

(*ACELERACION*)
C1:=AE1^; GET(AE1); C2:=AE1^; GET(AE1); C3:=AE1^; GET(AE1); C4:=AE1^; GET(AE1);

(*VELOCIDAD, POSICION Y TIEMPO*)
VMEDA:=AE1^; GET(AE1); TETA:=AE1^; GET(AE1); GET(AE1); INTERVALO:=AE1^; GET(AE1);

(*LEE SIGUIENTE REGISTRO*)
LECTURA;

 (*LEE LOS REGISTROS NECESARIOS PARA LLEGAR AL TIEMPO INICIAL*)
WHILE SIGUIENTE < TIEMPO^[1] DO MUEVE;

TIEMPOM[1]:=TIEMPO^[1]; (*TIEMPO INICIAL PARA LAS MUESTRAS*)
IF R = 'P' (*ORDENADA INICIAL PARA LAS MUESTRAS*)
 THEN
  ORDM[1]:=POSICION(TIEMPOM[1])
 ELSE
  ORDM[1]:=VELOCIDAD(TIEMPOM[1]);

FOR J:=2 TO TRUNC((REGFIN-REGIN)/FACTOR/MUESTRA) DO

 BEGIN (*FOR J*)

   TIEMPOM[J]:=TIEMPO^[(J-1)*MUESTRA+1];
   IF R = 'P'
    THEN
     ORDM[J]:=POSICION(TIEMPOM[J])
    ELSE
     ORDM[J]:=VELOCIDAD(TIEMPOM[J]);

  END; (*FOR J*)

 CLOSE(AE1); (*CIERRA ARCHIVO DE CONDICIONES DESEADAS*)

END; (*UNO*)

(*PROCEDIMIENTO PARA COPIAS EN PAPEL*)

PROCEDURE PASAPAPEL(TAMANO: INTEGER);
EXTERNAL;


BEGIN (*GRAFREAL*)

 NEW(TIEMPO); NEW(ORD); (*ESPACIO PARA ARREGLOS DE GRAFICACION*)

 (*LEE DATOS*)
 UNO;

 (*ROTULOS PARA LA GRAFICA*)
 IF R = 'P' THEN UNORD:='RAD' ELSE UNORD:='RAD/S';

 IF R = 'P' THEN ROT:='POSICION REAL VS. DESEADA'
            ELSE ROT:='VELOCIDAD REAL VS. DESEADA';

 (*CURVA DE CONDICIONES REALES*)
 I:=TRUNC((REGFIN-REGIN)/FACTOR)-1;
 DEFAULT(TIEMPO,ORD,I,CUR,'S',UNORD,ROT);

 (*TRASPASA DATOS DE CONDICIONES DESADAS*)
 I:=(I+1) DIV MUESTRA ;
 FOR J:=1 TO I DO BEGIN TIEMPO^[J]:=TIEMPOM[J]; ORD^[J]:=ORDM[J]; END;

 (*DIBUJA CONDICIONES DESEADAS*)
 DIBUJA(TIEMPO,ORD,I,FLW);
 FIJAIMAGEN;

 (*LIBERA ESPACIO DE VARIABLES*)
 RELEASE(ORD); RELEASE(TIEMPO);

 (*PREGUNTA POR COPIA EN PAPEL*)
 WRITE('COPIA EN PAPEL(S/N)? '); READ(R); WRITELN;
 IF R = 'S' THEN PASAPAPEL(0);

END. (*GRAFREAL*)
```

```
{ ...
{*GRAFPASO.TEXT*}

  PROGRAMA PARA GRAFICAR LA RESPUESTA A PULSO DE UN *)
(* MOTOR DE PASOS. LOS DATOS SE GENERAN EL EL PROGRA-*)
 * MA  PASO.CODE.                                     *)
 *                                                    *)
 *                 PROYECTO    2136                   *)
 *                 NOVIEMBRE   1983                   *)
 *                    LAIL                            *)


PROGRAM GRAFPASO;

(* DECLARA LIBRERIAS *)

USES TRANSCEND,TURTLEGRAPHICS,GRAFICASYX;

 * ZONA DE VARIABLES *)

VAR A,T:ARREGLODATOS;(*ARREGLOS A GRAFICAR*)
    I,J:INTEGER;(*AUXILIARES*)
    S:CHAR;(*AUXILIAR PARA RESPUESTAS*)
    R:REAL;(*DEFASAMIENTO PARA LOS ARCHIVOS*)
    NOMBRE:STRING[20];(*NOMBRE ARCHIVO DE DATOS*)
    AP:FILE OF REAL;(*ARCHIVO DE ENTRADA*)

(*PROCEDIMIENTO PARA COPIAS EN PAPEL*)

PROCEDURE PASAPAPEL;
EXTERNAL;

BEGIN(*GRAFPASO*)

 REPEAT(*UNTIL S*)

  (*OBTIENE DATOS DE LECTURA*)

  WRITE('QUE ARCHIVO? ');
  READLN(NOMBRE);
  WRITE('CUANTOS PUNTOS? ');
  READLN(I);
  WRITE('DEFASAMIENTO INICIAL? ');
  READLN(R);
  RESET(AP,NOMBRE);(*ABRE ARCHIVO*)

  NEW(A);NEW(T);(*DECLARA ARREGLOS DE GRAFICACION*)

  (*LLENA ARREGLOS DE GRAFICACION*)
  A^[1]:=1.5703-R;
  T^[1]:=0;
  FOR J:=2 TO I+1 DO
   BEGIN
    A^[J]:=AP^-R;GET(AP);
    T^[J]:=T^[J-1]+AP^;GET(AP);
   END;
  CLOSE(AP);(*CIERRA ARCHIVO*)

  (*EJECUTA GRAFICACION*)
  DEFAULT(T,A,I+1,PUN,'SEG','RAD','RESPUESTA A PASO');

  (*LIBERA ARREGLOS DE GRAFICACION*)
  RELEASE(T);RELEASE(A);

  WRITE('COPIA EN PAPEL(S/N)? ');READ(S);WRITELN;
  IF S = 'S' THEN PASAPAPEL;

  WRITE('CONTINUO(S/N)? ');READ(S);WRITELN;

 UNTIL S = 'N';(*END UNTIL*)

END.(*GRAFPASO*)
```

```
(*DATOS.TEXT*)

(*PROGRAMA QUE LLENA EL ARCHIVO DATOS.DATA CON DOS *)
(*VALORES: EL NO. DE EVENTOS POR MEDICION Y EL NO. *)
(*DE MUESTRAS A TOMAR.                             *)
(*       LUIS  ALVAREZ ICAZA  LONGORIA             *)
(*       INSTITUTO DE INGENIERIA, UNAM             *)
(*       PROYECTO 2136, DICIEMBRE 1983             *)

PROGRAM DATOS;

(*ZONA DE VARIABLES*)

VAR  I:INTEGER;(*AUXILIAR*)
     AS1:FILE OF INTEGER;(*ARCHIVO DE SALIDA*)

BEGIN(*DATOS*)

 REWRITE(AS1,'#5:DATOS.DATA');(*ABRE ARCHIVO*)

 WRITE  ('NO. DE EVENTOS POR MEDICION(EN 1/10 DE PASO)? ');
 READLN(I);AS1^:=I;PUT(AS1);(*TOMA NO. DE EVENTOS*)

 WRITE  ('NO. DE MUESTRAS EN EL EXPERIMENTO? ');
 READLN(I);AS1^:=I;PUT(AS1);(*TOMA NO. DE MUESTRAS*)

 CLOSE(AS1,LOCK);(*CIERRA ARCHIVO*)

END;(*DATOS*)
```

```
(*TIEMPOS.TEXT*)

(* PROGRAMA PARA TRANSFORMAR EL ARCHIVO SENALES.DATA *)
(* EN UN ARCHIVO COMPATIBLE CON LOS REQUERIMIENTOS   *)
(* DEL PROGRAMA CONTROL. EL ARCHIVO SE LLAMA         *)
(* TIEMPOS.DATA                                      *)
(* LOS REGISTROS DE SENALES.DATA CONTIENEN DOS VARIA- *)
(* BLES REALES: TIEMPO DE LA CONMUTACION Y SU TIPO,  *)
(* EL NO. DE REGISTROS SE INDICA EN NUM:ENALES.DATA. *)
(* EL ARCHIVO TIEMPOS.DATA CONTIENE DOS ENTEROS POR  *)
(* REGISTRO: EL PRIMERO INDICA EL TIEMPO PARA LA     *)
(* SIGUIENTE CONMUTACION Y EL SEGUNDO SE PARTE EN    *)
(* DOS FRACCIONES DE OCHO BITS, LA PRIMERA FRACCION  *)
(* (LOS BITS MENOS SIGNIFICATIVOS) INDICA EL LOGARIT- *)
(* MO BASE 10 A QUE SE ELEVARA LA FRECUENCIA DEL RE- *)
(* LOJ DE LA AFTLEII (QUE FUNCIONA A 1 MHZ); SOLO PUE- *)
(* DE TOMAR LOS VALORES (0,1,2,3), LA SEGUNDA FRAC- *)
(* CION INDICA LA DIRECCION DEL MOVIMIENTO: SI SU VA- *)
(* LOR ES CERO INDICA MOVIMIENTO A LA DERECHA Y SI ES *)
(* DOS, INDICA MOVIMIENTO A LA IZQUIERDA.            *)

(*         LUIS  ALVAREZ ICAZA  LONGORIA             *)
(*         INSTITUTO DE INGENIERIA, UNAM             *)
(*         PROYECTO 2136,DICIEMBRE 1983              *)

PROGRAM TIEMPOS;

(*ZONA DE VARIABLES*)

VAR     TIEMPO,(*TIEMPO DE CONMUTACION*)
        TIEANT,(*TIEMPO ANTERIOR DE CONMUTACION*)
        TIPO,(*DIRECCION DE MOVIMIENTO*)
        ACUM (*TEMPORAL PARA OPERACIONES*)
            :REAL;

        DELTA,(*DIFERENCIA DE TIEMPO*)
        FACTOR,(*FACTOR DE MULTIPLICACION DE DELTA*)
        MASCARA,(*DIRECCION DEL MOVIMIENTO*)
        I,(*AUXILIAR PARA ITERACIONES*)
        NUMREG (*NO. DE REGISTROS*)
              :INTEGER;

        AE1: FILE OF REAL;(*ARCHIVO DE ENTRADA*)
        AS1: FILE OF INTEGER;(*ARCHIVO DE SALIDA*)

(*PROCEDIMIENTO PARA LECTURA DE INFORMACION*)

PROCEDURE LECTURA;

BEGIN(*LECTURA*)

 TIEMPO:=AE1^;GET(AE1);(*LEE TIEMPO*)
 TIPO:=AE1^;  GET(AE1);(*LEE TIPO*)

END;(*LECTURA*)

(*PROCEDIMIENTO PARA ESCRITURA DE LOS RESULTADOS*)

PROCEDURE ESCRIBE;
```

```
BEGIN(*ESCRIBE*)

 AS1^:=DELTA;PUT(AS1);(*ESCRIBE TIEMPO*)
 AS1^:=FACTOR*256+MASCARA;PUT(AS1);(*ESCRIBE FACTOR Y MASCARA*)

END;(*ESCRIBE*)

(*PROCEDIMIENTO PARA CONVERSION DE LOS DATOS LEIDOS*)

PROCEDURE TRANSFORMA;

(*CONSTANTES PARA LA CONVERSION DE LOS ARCHIVOS*)

CONST   RELOJ = 1.0205E6  ;(*FRECUENCIA DEL RELOJ DE LA APPLE *)
        FACTOR0=   32767.0;(*CONSTANTE PARA FACTOR 0*)
        FACTOR1=  327670.0;(*IDEM FACTOR 1*)
        FACTOR2= 3276700.0;(*IDEM FACTOR 2*)
        FACTOR3=32767000.0;(*IDEM FACTOR 3*)
        MUERTO = 1.76384E-4;(*TIEMPO MUERTO EN LA EJECUCION DEL PROGRAMA CONTROL*)

BEGIN(*TRANSFORMA*)

 ACUM:=(TIEMPO-TIEANT-MUERTO)*RELOJ;(*NO. DE CICLOS DEL RELOJ*)

 IF ACUM <> 0
  THEN
   BEGIN(*IF ACUM*)
    WRITELN('TIEMPO DEMASIADO CORTO');
    EXIT(PROGRAM);
   END;(*IF ACUM*)

 IF ACUM <= FACTOR0
  THEN
   BEGIN(*FACTOR0*)(*CICLOS EN MICROSEGUNDOS*)
    FACTOR:=0;
    DELTA:=TRUNC(ACUM);
   END  (*FACTOR0*)
  ELSE
   IF ACUM <= FACTOR1
    THEN
     BEGIN(*FACTOR1*)(*CICLOS EN DECENAS DE MICROSEGUNDOS*)
      FACTOR:=1;
      ACUM:=ACUM/10;
      DELTA:=TRUNC(ACUM);
     END (*FACTOR1*)
    ELSE
     IF ACUM <= FACTOR2
      THEN
       BEGIN(*FACTOR2*)(*CICLOS EN CENTENAS DE MICROSEGUNDOS*)
        FACTOR:=2;
        ACUM:=ACUM/100;
        DELTA:=TRUNC(ACUM);
       END  (*FACTOR2*)
      ELSE
       IF ACUM <= FACTOR3
        THEN
         BEGIN(*FACTOR3*)(*CICLOS EN MILISEGUNDOS*)
          FACTOR:=3;
          ACUM:=ACUM/1000;
          DELTA:=TRUNC(ACUM);
         END
        ELSE
         BEGIN(*FACTOR3*)(*TIEMPO DE ENTRADA POSIBLEMENTE INCORRECTO*)
          WRITELN('TIEMPO EXCESIVO, NO ES POSIBLE GENERARLO');
          CLOSE(AS1);
          EXIT (PROGRAM);
         END;

 IF TIPO = 1(*DEFINE DIRECCION DE MOVIMIENTO*)
  THEN MASCARA:=0
  ELSE MASCARA:=2;

 TIEANT:=TIEMPO;(*REACTUALIZA TIEMPO*)

END;(*TRANSFORMA*)

BEGIN(*TIEMPOS*)

 RESET(AE1,'#5:NUMSENALES.DATA');(*OBTIENE NO. DE REGISTROS DE SENALES.DATA*)
 NUMREG:=TRUNC(AE1^);
 CLOSE(AE1);(*CIERRA ARCHIVO*)

 RESET(AE1,'#5:SENALES.DATA');(*ABRE ARCHIVO DE DATOS*)
 REWRITE(AS1,'#5:TIEMPOS.DATA');(*ABRE ARCHIVO DE SALIDA*)
 AS1^:=NUMREG;PUT(AS1);(*ESCRIBE NO. DE REGISTROS*)

 TIEANT:=0;(*INICIALIZA VARIABLES*)

 FOR I:=1 TO NUMREG DO
  BEGIN(*FOR I*)

   LECTURA;(*LEE DATOS*)
   TRANSFORMA;(*LOS CAMBIA DE FORMATO*)
   ESCRIBE;(*A ARCHIVO DE SALIDA*)

  END;

 CLOSE(AS1,LOCK);

END.(*TIEMPOS*)
```

```
{$L-}
{*LINEAL.TE$T*}
                {*EL PROGRAMA AJUSTA 3 PARAMETROS DE   *}
                {*UNA ECUACION DIFERENCIAL A LOS DATOS*}
                {*DE UN ARCHIVO ESPECIFICADO DURANTE   *}
                {*LA EJECUCION,POR MEDIO DE RECTAS,DE *}
                {*ACUERDO AL CRITERIO DE MINIMOS CUA --*}
                {*DRADOS.                              *}
                {*                                     *}
                {*          LUIS ALVAREZ ICAZA         *}
                {*                                     *}
                {*           PROYECTO  2136            *}
                {*                                     *}
                {*        INSTITUTO DE INGENIERIA      *}
PROGRAM AJUSTE1;
USES TRANSCEND;

CONST DIM=1000; {*NUMERO MAXIMO DE DATOS*}

TYPE  INDICE=0..DIM;
      VECTOR=ARRAY[1..3] OF REAL;
      VECTOREAL=ARRAY[INDICE] OF REAL;
      MATRIZ=ARRAY[1..3,1..3] OF REAL;

VAR   TETA,  {*DESPLAZAMIENTO ANGULAR*}
      OMEGA, {*VELOCIDAD ANGULAR*}
      ALFA,  {*ACELERACION ANGULAR*}
      DT,    {*INTERVALOS DE TIEMPO*}
      T      {*TIEMPO*}:VECTOREAL;
      B,     {*TERMINOS INDEPENDIENTES*}
      P      {*PARAMETROS DE LA ECUACION*},
      V1,V2  {*VECTORES AUXILIARES*}:VECTOR;
      A:MATRIZ;
      SOMEGA,SOMEGA2,SSENO,SOMEGASENO,
      SALFA,SALFA2,SALFAOMEGA,SALFASENO,
      SENOM,SUMEST,SUMREAL,
      VILR,A0,A1,A3:REAL;
      J,M,I,SUMSIGN:INTEGER;
      NOMBREESTR1145[20]
      AP:FILE OF REAL;

FUNCTION SIGN(R:REAL):INTEGER; {*FUNCION SIGNO*}
  BEGIN {*SIGNO*}
    IF R>0 THEN SIGN:=1
    ELSE IF R<0 THEN SIGN:=-1
         ELSE SIGN:=0;
  END; {*SIGNO*}

PROCEDURE RESUELVE(A:MATRIZ;E:VECTOR);

VAR
    AC:MATRIZ;
    D:VECTOR;
    I,J:INTEGER;
    DEL:REAL;

PROCEDURE DET33 (M:MATRIZ;VAR DELTA:REAL);
      {*CALCULA EL DETERMINANTE DE 3X3*}
BEGIN  (* DET33 *)

  DELTA:=M[1,1]*(M[2,2]*M[3,3]-M[2,3]*M[3,2])+
         M[1,2]*(M[2,3]*M[3,1]-M[2,1]*M[3,3])+
         M[1,3]*(M[2,1]*M[3,2]-M[2,2]*M[3,1]);
END;  (* DET33 *)

PROCEDURE INTERCO (AS:MATRIZ;BE:VECTOR;N:INTEGER;VAR CS:MATRIZ);
      {*INSERTA COLUMNA N A LA MATRIZ*}
VAR
    I,J:INTEGER;

BEGIN    (* INTERCO *)
  FOR I:=1 TO 3 DO
    FOR J:=1 TO 3 DO
      BEGIN {*FOR*}
        IF J=N
          THEN CS[I,J]:=DE[I]
          ELSE CS[I,J]:=AS[I,J];
      END; {*FOR*}
END;    (* INTERCO  *)

BEGIN    (*  RESUELVE   *)

  DET33 (A,DEL);      (* LLAMA A DET33 *)

  IF ABS(DEL)>1E-30
     THEN
      BEGIN {*THEN*}
        FOR I:=1 TO 3 DO
          BEGIN {*FOR*}
            INTERCO (A,E,I,AC); (* LLAMA A INTERCO *)
            DET33 (AC,DE[I]);    (* LLAMA A DET33 *)
          END; {*FOR*}
        P[1]:=DE[1]/DEL;
        P[2]:=DE[2]/DEL;
        P[3]:=DE[3]/DEL;
      END  {*THEN*}
     ELSE
      BEGIN {*ELSE*}
        WRITELN( EL DETERMINANTE VALE CERO );
        EXIT (*PROGRAM*);
      END; {*ELSE*}
END;     {*  RESUELVE  *}

PROCEDURE UNO;

BEGIN  (*  UNO  *)

WRITE ( DIMENSION DEL VECTOR TETA= );
READLN (M);
WRITE( ARCHIVO  );
READLN(NOMBRE);
RESET(AP,NOMBRE); {*LECTURA DEL ARCHIVO*}
FOR I:=1 TO M DO
  BEGIN {*FOR*}
    TETA[I]:=AP^;
    GET (AP);
    DT[I]:=AP^;
    GET(AP);
  END; {*FOR*}
CLOSE(AP);
T[0]:=0;
FOR I:=1 TO M DO {*FORMA EL VECTOR TIEMPO*}
   T[I]:=T[I-1]+DT[I];

            {* SE OBTIENEN LAS DERIVADAS *}

FOR I:=1 TO M-1 DO
   OMEGA[I]:=(TETA[I+1]-TETA[I])/DT[I+1];
FOR I:=1 TO M-2 DO
   ALFA[I]:=(OMEGA[I+1]-OMEGA[I])/DT[I+1];

SOMEGA:=0;
SOMEGA2:=0;
SENOM:=0;
SSENO:=0;
SALFA2:=0;
SOMEGASENO:=0;
SUMEST:=0;
SUMREAL:=0;
SALFA:=0;
SALFAOMEGA:=0;
SALFASENO:=0;
SUMSIGN:=0;

END;  (*  UNO  *)

PROCEDURE DOS;    (* CORRELACION *)

BEGIN {*DOS*}
  FOR I:=2 TO M-2 DO
    BEGIN {*FOR*}
      SUMEST:=SUMEST+SQR(P[1]*OMEGA[I]-P[2]*ALFA[I]+
                P[3]*SIGN(OMEGA[I])-SENOM);
      SUMREAL:=SUMREAL+SQR(SIN(TETA[I])-SENOM);
    END; {*FOR*}
  VILR:=SQRT(SUMEST/SUMREAL);
  WRITELN ( COEFICIENTE DE CORRELACION =  ,VILR:4:3);
END; {*DOS*}

BEGIN{*PARAMETROSDELMOTOR*}  (*  AJUSTE  *)
```

```
UNO;

FOR I:=2 TO M-2 DO (*SUMATORIAS*)
  BEGIN (*FOR*)
      SOMEGA:=SOMEGA+OMEGA[I]*SIGN(OMEGA[I]);
      SOMEGA2:=SOMEGA2+OMEGA[I]*OMEGA[I];
      SSENO:=SSENO+SIN(TETA[I])*SIGN(OMEGA[I]);
      SALFA2:=SALFA2+SQR(ALFA[I]);
      SOMEGASENO:=SOMEGASENO+OMEGA[I]*SIN(TETA[I]);
      SALFA:=SALFA+ALFA[I]*SIGN(OMEGA[I]);
      SALFAOMEGA:=SALFAOMEGA+OMEGA[I]*ALFA[I];
      SALFASENO:=SALFASENO+ALFA[I]*SIN(TETA[I]);
      SUMSIGN:=SUMSIGN+SQR(SIGN(OMEGA[I]));
      SENOM:=SENOM+SIN(TETA[I]);
  END; (*FOR*)

A[1,1]:=SOMEGA2;
A[1,2]:=-SALFAOMEGA;
A[1,3]:=SOMEGA;
A[2,1]:=SALFAOMEGA;
A[2,2]:=-SALFA2;
A[2,3]:=SALFA;
A[3,1]:=SOMEGA;
A[3,2]:=-SALFA;
A[3,3]:=SUMSIGN;
B[1]:=-SOMEGASENO;
B[2]:=-SALFASENO;
B[3]:=-SSENO;

WRITELN;
RESUELVE(A,B);
WRITELN;
AUX:=B[1]/B[2];A1:=B[3]/B[2];A3:=1/B[2];
WRITELN (' B/J=',AUX,' T1/J=',A1,'  F1/J=',A3);

END;

END.   *  AJUSTE  *
```

```
(*$I-*)
(*PARABOLA.IE(T*)
            (*AJUSTA LOS PARAMETROS DE UNA ECUACION*
            (*DIFERENCIAL A UN CONJUNTO DE DATOS DE*
            (*UN ARCHIVO ESPECIFICADO DURANTE LA   *
            (*ASIGNANDO PARABOLAS,TOMANDO PUNTOS DE*
            (*TRES EN TRES,DE ACUERDO AL ERROR MI- *
            (*NIMO AL CUADRADO                     *)
            (*                                     *)
            (*         LUIS ALVARES ICAZA          *)
            (*                                     *)
            (*           PROYECTO 2136             *)
            (*                                     *)
            (*        INSTITUTO DE INGENIERIA      *)
PROGRAM AJUSTE;
USES TRANSCEND;

CONST DIM=1000;

TYPE  INDICE=1..DIM;
      VECTOR=ARRAY[1..3] OF REAL;
      VECTOREAL=ARRAY[INDICE] OF REAL;
      MATRIZ=ARRAY[1..3,1..3] OF REAL;

VAR   TETA,  (*DESPLAZAMIENTO ANGULAR*)
      OMEGA, (*VELOCIDAD ANGUALAR*)
      ALFA,  (*ACELERACION ANGULAR*)
      DT,    (*INTERVALOS DE TIEMPO*)
      T      (*TIEMPO*):VECTOREAL;
      B,     (*TERMINOS INDEPENDIENTES*
      P,     (*PARAMETROS DE LA ECUACION*)
      V1,V2  (*VECTORES AUXILIARES*):VECTOR;
      A:MATRIZ;
      SOMEGA,SOMEGA2,SSENO,SSENO2,SOMEGASENO,SALFA,
      SALFAOMEGA,SALFASENO,ALFAM,SUMEST,SUMREAL,VILR,
      AO,A1,A2:REAL;
      M,I,SUMSIGN:INTEGER;
      NOMBRE:STRING[20];
      AP:FILE OF REAL;

FUNCTION SIGN(RY:REAL):INTEGER;  (*FUNCION SIGNO*)
    BEGIN (*SIGNO*)
      IF RY>0
        THEN SIGN:=1
        ELSE IF RY<0
               THEN SIGN:=-1
               ELSE SIGN:=0;
    END; (*SIGNO*)

PROCEDURE RESUELVE(A:MATRIZ;B:VECTOR);

VAR
    AC:MATRIZ;
    D:VECTOR;
    I,J:INTEGER;
    DEL:REAL;              (*DETERMINANTE*)
PROCEDURE DET33 (M:MATRIZ;VAR DELTA:REAL);

BEGIN  (* DET33 *)
  DELTA:=M[1,1]*(M[2,2]*M[3,3]-M[2,3]*M[3,2])+
         M[1,2]*(M[2,3]*M[3,1]-M[2,1]*M[3,3])+
         M[1,3]*(M[2,1]*M[3,2]-M[2,2]*M[3,1]);
END;  (* DET33 *)

                (*INTERCALA COLUMNA*)
PROCEDURE INTERCO (AS:MATRIZ;BE:VECTOR;N:INTEGER;VAR CS:MATRIZ);

VAR
    I,J:INTEGER;

BEGIN (* INTERCO *)
  FOR I:=1 TO 3 DO
    FOR J:=1 TO 3 DO
      BEGIN (*FOR*)
        IF J=N
          THEN CS[I,J]:=BE[I]
          ELSE CS[I,J]:=AS[I,J];
      END; (*FOR*)
END; (* INTERCO  *)
```

```
BEGIN (* RESUELVE *)

  DET33 (A,DEL);              (* LLAMA A DET33 *)

  IF ABS(DEL)>1E-30
     THEN
       BEGIN (*THEN*)
         FOR I:=1 TO 3 DO
           BEGIN (*FOR*)
             INTERCO (A,D,I,AC); (* LLAMA A INTERCO *)
             DET33 (AC,D[I]);    (* LLAMA A DET33 *)
           END; (*FOR*)
         P[1]:=D[1]/DEL;
         P[2]:=D[2]/DEL;
         P[3]:=D[3]/DEL;
       END  (* THEN *)
     ELSE
       BEGIN (*ELSE*)
         WRITELN('EL DETERMINANTE VALE CERO');
         EXIT(PROGRAM);
       END; (*ELSE*)
END; (* RESUELVE *)

PROCEDURE CARGAMATRIZ(X,Y:VECTOR);

VAR
    AC:MATRIZ;
    BC:VECTOR;
    I,J:INTEGER;

BEGIN   (*  CARGAMATRIZ  *)
  FOR I:=1 TO 3 DO
    BEGIN (*FOR*)
      AC[I,1]:=X[I]*X[I];
      AC[I,2]:=X[I];
      AC[I,3]:=1;
      BC[I]:=Y[I];
    END; (*FOR*)

  RESUELVE (AC,BC);

  A0:=P[1];A1:=P[2];A2:=P[3];
END;   (*  CARGAMATRIZ  *)

PROCEDURE UNO;

BEGIN  (*  UNO  *)

WRITE ('DIMENSION DEL VECTOR TETA= ');
READLN (M);
WRITE('ARCHIVO ');
READLN(NOMBRE);
RESET(AP,NOMBRE);
FOR I:=1 TO M DO
  BEGIN
    TETA[I]:=AP^;
    GET(AP);
    DT[I]:=AP^;
    GET(AP);
  END;
CLOSE(AP);
T[0]:=0;
FOR I:=1 TO M DO (*FORMA EL VECTOR TIEMPO*)
   T[I]:=T[I-1]+DT[I];

                (*SE ENCUENTRAN LAS DERIVADAS*)

V1[1]:=T[1];V1[2]:=T[2];V1[3]:=T[3];
V2[1]:=TETA[1];V2[2]:=TETA[2];V2[3]:=TETA[3];

CARGAMATRIZ(V1,V2);

OMEGA[1]:=0;
ALFA[1]:=A2;
FOR I:=2 TO M-1 DO
   BEGIN (*FOR*)
      V1[1]:=T[I-1];V1[2]:=T[I];V1[3]:=T[I+1];
      V2[1]:=TETA[I-1];V2[2]:=TETA[I];V2[3]:=TETA[I+1];
      CARGAMATRIZ(V1,V2);
      OMEGA[I]:=2*A0*T[I]+A1;
      ALFA[I]:=2*A0;
   END; (*FOR*)

SUMEGA1:=0;
SUMEGA2:=0;
SSEN01:=0;
SSEN02:=0;
SUMEGASEN01:=0;
SALFA1:=0;
SALFAOMEGA1:=0;
SALFASEN01:=0;
SUMSIGN1:=0;
ALFAM:=0;
SUMEST:=0;
SUMREAL:=0;

END; (* UNO *)

PROCEDURE DOS; (*CORRELACION*)

 BEGIN (*DOS*)
    FOR I:=2 TO M-2 DO
       BEGIN (*FOR*)
          SUMEST:=SUMEST+SQR(P[1]*OMEGA[I]+P[2]*SIN(TETA[I])
                  +P[3]-SIGN(OMEGA[I]))-ALFAM);
          SUMREAL:=SUMREAL+SQR(ALFA[I]-ALFAM);
       END; (*FOR*)
    VILR:=SQRT(SUMEST/SUMREAL);
    WRITELN (' COEF. DE CORRELACION = ',VILR:4:3);
 END; (* DOS *)

BEGIN(*PARAMETROSDELMOTOR*)  (*  AJUSTE  *)

UNO;

FOR I:=2 TO M-2 DO
   BEGIN (*FOR*)
      SUMEGA1:=SUMEGA+SIGN(OMEGA[I]);
      SUMEGA2:=SUMEGA2+OMEGA[I]*OMEGA[I];
      SSEN01:=SSEN0+SIN(TETA[I])*SIGN(OMEGA[I]);
      SSEN02:=SSEN02+SQR(SIN(TETA[I]));
      SUMEGASEN01:=SUMEGASEN0+OMEGA[I]*SIN(TETA[I]);
      SALFA1:=SALFA+ALFA[I]*SIGN(OMEGA[I]);
      SALFAOMEGA1:=SALFAOMEGA+OMEGA[I]*ALFA[I];
      SALFASEN01:=SALFASEN0+ALFA[I]*SIN(TETA[I]);
      SUMSIGN1:=SUMSIGN+SQR(SIGN(OMEGA[I]));
      ALFAM:=ALFAM+ALFA[I];
   END; (*FOR*)
A[1,1]:=SUMEGA2;
A[1,2]:=SUMEGASEN0;
A[1,3]:=SUMEGA1;
A[2,1]:=SUMEGASEN0;
A[2,2]:=SSEN02;
A[2,3]:=SSEN01;
A[3,1]:=SUMEGA1;
A[3,2]:=SSEN01;
A[3,3]:=SUMSIGN1;
B[1]:=-SALFAOMEGA1;
B[2]:=-SALFASEN01;
B[3]:=-SALFA1;

RESUELVE(A,B);

WRITELN;
WRITELN('TETA= ',TETA[2],' OMEGA= ',OMEGA[2]);
WRITELN('A= ',P[1]:8:3,' B=',P[2]:8:3,' C=',P[3]:8:3);

DOS;

END.  (*   AJUSTE   *)
```

```
        .IFDC PASAPAPEL,1
;-------------------------------------------------------------------------
;
;     PROCEDURE PASAPAPEL;
;
; PROCEDIMIENTO PARA TRANSFERIR UNA GRAFICA DE ALTA RESOLUCION A LA
; IMPRESORA ATI-2.
;
; LA GRAFICA DEBE RESIDIR EN LA PAGINA UNO. LA IMPRESORA DEBE ESTAR
; COLOCADA EN LA RANURA 1.
;
; LA RESOLUCION DE LA GRAFICA PERMANECE IGUAL PERO EL AREA SE CUADRU-
; PLICA.
;
; EL CODIGO DE ESTE PROGRAMA SE DEBE LIGAR CON EL PROGRAMA EN PASCAL
; DESDE EL QUE SE DESEA CONTROLAR LA TRANSFERENCIA.
;
; LA FORMA DE LLAMADO ES:
;
;       PASAPAPEL(ESCALA);
;       ------------------
; EL PARAMETRO ESCALA INDICA SI SE QUIERA
; AMPLIFICAR AL DOBLE LA GRAFICA EN PAPEL
; (ESCALA <>0) O SE DESEA A ESCALA UNITARIA
; (ESCALA = 0).
;
;-------------------------------------------------------------------------

;*************************************************************************

; MACRO PARA SACAR DOS BYTES DEL STACK

        .MACRO GUARDA

        PLA     ;SALVA BPS
        STA %1
        PLA     ;SALVA BMS
        STA %1+1

        .ENDM

;*************************************************************************

; MACRO PARA PONER DOS BYTES EN EL STACK

        .MACRO PONE

        LDA %1+1;PONE BMS
        PHA
        LDA %1  ;PONE BPS
        PHA

        .ENDM

;*************************************************************************

;MACRO PARA LEER UNA COLUMNA DE PUNTOS

        .MACRO LECTURA

        LDY #2F             ;INICIALIZA CONTROLES DE ITERACION
        LDX #07

$01:    LDA BLOQUE,X        ;BMS DE DIRECCION DE BLOQUE DE RENGLONES
        STA INDIR+1

        DEX                 ;DECREMENTA X

        LDA BLOQUE,X        ;BPS DE DIRECCION
        STA INDIR           ;DIRECCION DEL RENGLON

        CLC                 ;BORRA CARRY
        LDA INDICE,Y
        ADC INDIR+1         ;DIRECCION DE BLOQUE Y RENGLON CONJUNTAMENTE
        STA INDIR+1

        LDA %2              ;AGREGA EL NO. DE COLUMNA
        ADC INDIR           ;INDIR E INDIR+1 CONTIENEN LA DIRECCION CORRECTA
        STA INDIR           ;(BLOQUE, RENGLON Y COLUMNA)

        STY TEMP1           ;SALVA INDICE Y
        STX TEMP2           ;SALVA INDICE X

        TXA                 ;OBTIENE INDICE EN LAS TABLAS
        ASL A               ;LO MULTIPLICA POR 4
        ASL A

        CLC                 ;BORRA CARRY
        ADC TEMP1           ;INDICE CORRECTO EN ACUMULADOR
        TAX                 ;  "     "    "  X

        LDY #00             ;BORRA INDICE Y
        LDA @INDIR,Y        ;OBTIENE VALOR DE MEMORIA
        STA %1,X            ;LO SALVA ADECUADAMENTE

        LDY TEMP1           ;RECUPERA INDICES
        LDX TEMP2

        DEY                 ;ACTUALIZA INDICADORES

        BMI $02             ;CONTINUA CON EL BLOQUE DE RENGLONES

        INY                 ;DEJA APUNTADOR A BLOQUE INTACTO
        JMP $01             ;LEE OTRO RENGLON

$02:    DEX                 ;APUNTA AL SIGUIENTE BLOQUE
        CPX #00             ;PREGUNTA SI TERMINA LA LECTURA

        BMI $03             ;TERMINA EN $02

        LDY #07             ;REINICIALIZA X
        JMP $01             ;CONTINUA

$03:    DEC %2              ;ACTUALIZA NO. DE COLUMNA

        .ENDM

;*************************************************************************
```

```
;MACRO PARA TIRAR DE BUFF EL BMS

        .MACRO TIRA

        LDX #00F            ;INICIALIZA CONTROL DE ITERACION

$01:    ASL BUFF,X          ;TIRA 7BIT DE BUFF

        DEX                 ;ACTUALIZA INDICADOR

        CPX #0FF            ;VA AL SIGUIENTE ELEMENTO DEL ARREGLO
        BNE $01

        NOP

        .ENDM

;*****************************************************************

; MACRO PARA PASAR EL BMS DE BUFFER COMO EL BFS DE ACUM, EL BMS DE ACUM S
* SE PIERDE

        .MACRO ROTA

        LDX #00F            ;APUNTA AL ULTIMO ELEMENTO DE ACUM Y BUFF

$01:    ASL BUFF,X          ;OBTIENE 7BIT DE BUFF
        ROL ACUM,X          ;LO PASA A ACUM

        DEX                 ;ACTUALIZA INDICADOR

        CPX #0FF            ;VA AL SIGUIENTE ELEMENTO DEL ARREGLO
        BNE $01

        NOP

        .ENDM

;*****************************************************************

; MACRO PARA TIRAR LOS BMS DE ACUM

        .MACRO ROTA3

        LDA #02             ;INDICE PARA NO. DE BITS TIRADOS
        STA TEMP3

$01:    LDA #06             ;PREGUNTA SI ES TURNO DE LEER UNA NUEVO COLUMNA
        CMP ACARREO

        BFL $02             ;BRINCA SI NO ES TIEMPO DE LEER

        LECTURA BUFF,COL    ;LEE NUEVA COLUMNA
        TIRA                ;QUITA 7BMS A BUFF

        LDA #00             ;REINICIALIZA ACARREOS
        STA ACARREO

$02:    ROTA                ;TIRA 1 BIT

        INC ACARREO         ;ACTUALIZA NO. DE ACARREOS

        DEC TEMP3
        BFL $01             ;SIGUE SI AUN NO CONCLUYE

        NOP

        .ENDM

;*****************************************************************

; MACRO PARA OBTENER EL CARACTER A IMPRIMIR

        .MACRO CARACTER

        LDY INDICA          ;CONTROL DE ITERACION

        LDX %1              ;PASA RENGLON A X
        LDA ACUM,X          ;OBTIENE ELEMENTO DE ACUM
        STA TEMP4           ;LO GUARDA

$01:    ASL TEMP4           ;OBTIENE PRIMER BIT
        BCC $02             ;BRINCA SI ES CERO

        ROR IMP             ;PONE PRIMER UNO
        LDA ESCALA          ;PREGUNTA POR ESCALA DE IMPRESION
        BEQ $03             ;BRINCA SI NO ES DUPLICACION

        SEC
        ROR IMP             ;PONE SEGUNDO UNO

        JMP $03

$02:    ROR IMP             ;PONE PRIMER CERO
        LDA ESCALA          ;PREGUNTA POR ESCALA DE IMPRESION
        BEQ $03             ;BRINCA SI NO ES DUPLICACION

        CLC
        ROR IMP             ;PONE SEGUNDO CERO

$03:    DEY

        BFL $01             ;AUN NO SON TRES BIT

        ROR IMP             ;CONCLUYE LA ROTACION
        ROR IMP

        LDA #0BBMS          ;LEVANTA 2BMS
        ORA IMP
        STA IMP             ;LO SALVA

        .ENDM

;*****************************************************************
;
* MACRO PARA FINALIZAR LA IMPRESION DE LA GRAFICA

        .MACRO FIN

        LDA INDFIN      ;INDICE PARA ITERACIONES
```

```
        STA TEMP3

$01:    ROTA            ;OBTIENE TRES BITS
        ROTA
        ROTA

        LDA ESCALA
        CMP #05         ;OBTIENE TRES BITS MAS EN CASO DE SER NECESARIO
        ROTA
        ROTA
        ROTA

$05:    LDA #00
        STA TEMP1
        LDA #00
        STA TEMP2       ;INDICE PARA IMPRESION DE LINEA

$02:    CARACTER TEMP1  ;IMPRIME CARACTER DOS VECES
        JSR SUBCAR

        LDA ESCALA
        BEQ $03
        LDA IMP
        JSR SUBCAR

$03:    INC TEMP1       ;OTRO CARACTER
        LDA TEMP2
        CMP TEMP1

        BNE $02         ;CONTINUA

        LDA CARDIA      ;IMPRIME LINEA
        JSR SUBCAR

        DEC TEMP3       ;PREGUNTA POR TERMINACION

        BMI $04         ;SIGUE SI NO SON CINCO
        JMP $01


$04:    NOP

        .ENDM

;*****************************************************************

;ZONA DE LOCALIDADES FIJAS EN MEMORIA

RETORNO .EQU 00000              ;DIRECCION DE RETORNO A PASCAL
INDIF   .EQU 00002              ;AUXILIAR PARA DIRECCIONAMIENTO INDIRECTO

SUBCAR  .EQU 0C100              ;DIRECCION DE LA SUBRUTINA EN ROM

;*****************************************************************

*PRINCIPIO SUBRUTINA

INIC:   GUARDA RETORNO          ;SALVA DIRECCION DE REGRESO A PASCAL
        GUARDA ESCALA           ;SALVA FACTOR DE ESCALA
        LDA ESCALA              ;APUNTADOR PARA TRASLACION DE BITS
        BEQ BAND                ;SIGUE SI NO HAY DUPLICACION DE ESCALA
        LDA #02                 ;EN CASO CONTRARIO PONE DOS EN INDICA
        STA INDICA
        LDA #04                 ; Y CUATRO EN INDFIN
        STA INDFIN

BAND:   LECTURA ACUM,COL        ;LEE ULTIMA COLUMNA
        LECTURA BUFF,COL        ; "  PENULTIMA COLUMNA

        TIRA                    ;QUITA 7 BITS DE BUFFER

        ROTA                    ;PASA UN BIT DE BUFFER A ACUM

        INC ACARREO             ;ACTUALIZA INDICADOR DE ACARREOS

        LDA CARCR               ;ASEGURA QUE EL BUFFER DE LA IMPRESORA ESTE
        JSR SUBCAR              ;VACIO
        LDA CARLF
        JSR SUBCAR

        LDA CARESC              ;PONE EN MODO DE GRAFICACION
        JSR SUBCAR
        LDA CARG
        JSR SUBCAR

LOOP:   CARACTER RENGLON        ;OBTIENE CARACTER A IMPRIMIR

        LDA IMP                 ;CARACTER A ACUMULADOR

        JSR SUBCAR              ;A RUTINA DE IMPRESION

        LDA ESCALA              ;NO DUPLICA EN ESCALA UNITARIA
        BEQ NODUP

        LDA IMP
        JSR SUBCAR      ;IMPRIME NUEVAMENTE EL CARACTER

NODUP:  INC RENGLON     ;ACTUALIZA NO. DE RENGLON

        LDA #00         ;PREGUNTA SI TERMINO EL RENGLON DE IMPRESION

        CMP RENGLON

        BEQ SIGUE       ;CONTINUA SI TERMINA CON RENGLONES

        JMP LOOP        ;CONTINUA CON EL SIGUIENTE RENGLON

SIGUE:  LDA COL         ;PREGUNTA SI LEYO LA ULTIMA COLUMNA
        CMP #00

        BMI FINAL       ;CONTINUA SINO HA CONCLUIDO

        JMP CONT

FINAL:  LDA CARDIA      ;TERMINA LINEA DE GRAFICA
        JSR SUBCAR
        FIN             ;IMPRIME CINCO LINEAS RESTANTES

        LDA CARMEN      ;SALE DE MODO DE GRAFICACION
        JSR SUBCAR
```

```
        LDA #00         ;INICIALIZA RENGLON Y ACARREO
        STA RENGLON     ;ANTES DE REGRESAR A PASCAL
        STA ACARREO

        LDA #27         ;INICIALIZA COLUMNA ANTES DEL
        STA COL         ;REGRESO A PASCAL

        LDA #05         ;INICIALIZA INDICADOR PARA TRASLACION
        STA INDICA
        LDA #02         ;INICIALIZA INDICADOR PARA ULTIMAS LINEAS
        STA INDFIN

        PONE RETORNO    ;DIRECCION DE RETORNO A PASCAL

        RTS             ;REGRESA A PASCAL

CONT:   LDA CARDIA      ;TERMINA LINEA DE GRAFICA
        JSR SUBCAR

        ROTA3           ;TIRA LOS 3 BMS DE ACUM
        LDA ESCALA      ;TIRA OTROS 3 SI LA ESCALA ES UNITARIA
        DEC CONT1
        JMP CONT2

CONT1:  ROTA3
        .
CONT2:  LDA #00         ;ACTUALIZA NO. DE RENGLON
        STA RENGLON

        JMP LOOP        ;CONTINUA CON IMPRESION

;*****************************************************************

;ZONA DE DECLARACION DE VARIABLES Y CONSTANTES

ACUM:   .BLOCK 0C0,0    ;ARREGLO PARA GUARDAR LOS 3 PUNTOS QUE SIGUEN A
                        ;IMPRIMIR
BUFF:   .BLOCK 0C0,0    ;ARREGLO PARA LA LECTURA DE PUNTOS

BLOQUE  .WORD  2000,2080,2100,2180,2200,2280,2300,2380,2400,2480,2500
        .WORD  2580,2600,2680,2700,2780,2800,2880,2900,2980,3000,3080
        .WORD  3100,3180;DIRECCIONES DE LOS BLOQUES DE RENGLONES
                        ;(9 RENGLONES/BLOQUE)

INDICE: .WORD  0400,0C00,1410,1C10
                        ;OFFSET PARA RENGLONES EN CADA BLOQUE
RENGLON:.BYTE   000     ;INDICE PARA RENGLON CORRIENTE
COL:    .BYTE    27     ;INDICE PARA COLUMNAS
ACARREO:.BYTE    00     ;CONTADOR DE ACARREOS

ESCALA: .WORD  0001     ;ESCALA A IMPRIMI
                        ;POR DEFAULT LA ESCALA ES 1
INDICA: .BYTE    05     ;INDICADOR PARA EL TRASLADO DE BITS
INDFIN: .BYTE    02     ;INDICADOR PARA ULTIMAS LINEAS

IMP:    .BYTE    00     ;CARACTER A IMPRIMIR

DOCE:   .BYTE   0C0     ;MASCARA PARA LEVANTAR LOS 2 BMS
```

```
CARESC: .BYTE   01B     ;CARACTER ESC
CARG:   .BYTE   047     ;  "       G
CARDIA: .BYTE   02F     ;CAMBIO DE LINEA EN MODO DE GRAFICACION
CARMEN: .BYTE   02D     ;TERMINACION DE MODO DE GRAFICACION
CARCR:  .BYTE   00D     ;REGRESO DE CARRO
CARLF:  .BYTE   00A     ;ALIMENTACION DE LINEA

TEMP1:  .BYTE   000     ;AUXILIAR PARA GUARDADO TEMPORAL
TEMP2:  .BYTE   000     ;IDEM
TEMP3:  .BYTE   000     ;IDEM
TEMP4:  .BYTE   000     ;IDEM

;FIN DE LA ZONA DE CONSTANTES

;*****************************************************************

;FIN DE LA SUBRUTINA

        .END
```

ANEXO C

ACOPLAMIENTOS PARA LAS PRUEBAS EN MOTORES DE PASOS

Para las pruebas de motores de pasos se utilizaron cuatro acoplamientos especialmente construidos para el efecto: un tacómetro digital, un reloj para generar los tiempos de conmutación, un codificador para detector de posición y un emulador para este último. Además se utilizaron un detector de posición y un acoplador de potencia para motores de pasos comerciales. Este anexo describe los acoplamientos especiales y proporciona los datos más importantes de los comerciales.

TACOMETRO DIGITAL

INTRODUCCION

Las pruebas del comportamiento dinámico de motores de pasos exigen mediciones de posición y velocidad angulares. Estas mediciones se pueden realizar tanto digital como analógicamente.

Para las primeras se deben obtener, a partir de potenciómetros, tacogenerador y temporizadores, señales que se conectan directamente a un graficador X-Y. La principal limitación de esta alternativa estriba en que el graficador responde inadecuadamente cuando se presentan eventos muy rápidos o de carácter altamente oscilatorio.

Las mediciones digitales requieren usar detectores de posición y relojes. Basan su funcionamiento en el muestreo de los eventos a medir. Si se quiere eliminar la limitación de las mediciones analógicas, la frecuencia de muestreo debe variar según la naturaleza de los fenómenos que se quiere medir.

En el caso específico de los motores de pasos, el comportamiento de la velocidad es altamente oscilatorio, lo que hace recomendable efectuar digitalmente las mediciones de velocidad.

Con este propósito se diseñó y construyó un tacómetro digital controlado directamente como un acoplamiento de la microcomputadora Aplle II. A continuación se describe el funcionamiento del mismo.

TEORIA DE FUNCIONAMIENTO

Las mediciones discretas de velocidad requieren conocer los incrementos de desplazamiento ocurridos en incrementos de tiempo. Esto se puede realizar de dos formas:

i) Dado un intervalo de tiempo constante, se determina el

desplazamiento que ocurre en él.

ii) La inversa.

En el caso de mediciones controladas por computadora, la resolución de los relojes digitales que estas pueden manejar es normalmente mayor que la de los detectores de posición (1 MHz vs. $1x10^3$ eventos por ciclo). Esto significa que las mediciones de velocidad son más precisas si se utiliza la segunda alternativa de medición.

Para resolver el problema de las frecuencias de muestreo mencionado en la introducción, se eligen intervalos de desplazamiento Ds que van desde fracciones de un paso del motor, hasta un centenar o millar de ellos. Es posible con esta elección medir tanto fenómenos oscilatorios, como el comportamiento promedio de la velocidad.

Al detectar la posición en los motores de pasos se pueden presentar oscilaciones en la posición. Para eliminar su efecto se requiere que los detectores de posición sean insensibles a dichas oscilaciones, o bien, que sean capaces de detectarlas totalmente. En el tacómetro digital que se describe se optó por conectar los detectores de posición a contadores hacia arriba y hacia abajo (CAA) que permiten medir el número absoluto de desplazamientos en una dirección, y con ellos son insensibles a las oscilaciones.

La velocidad se mide haciendo funcionar medidores de tiempo en forma paralela a los detectores de posición. Estos medidores funcionan mientras no ha transcurrido el desplazamiento Ds prefijado.

Resumiendo lo anterior, el funcionamiento del tacómetro digital es como sigue:

i) Se establece el intervalo de desplazamiento Ds que se desea medir.

ii) Se alimenta este valor a los CAA y se toma como convención que movimientos en la dirección deseada deben decrecer el valor de dichos contadores.

iii) Se conectan simultáneamente las señales que alimentan a los CAA y a los relojes que miden el tiempo.

iv) Cuando el valor de los CAA llega a cero se paran los relojes.

v) La velocidad se obtiene como el cociente entre el Ds elegido y el incremento de tiempo Dt medido al parar los relojes.

vi) Se repiten los pasos (ii) a (v) tantas veces como sea necesario.

Según los desarrollos teóricos de los métodos de control de los motores de pasos, es previsible que en ocasiones ocurran movimientos en la dirección contraria a la deseada, lo que produciría que el valor de los CAA nunca se anulase. Este problema se resolvió colocando dos sistemas iguales al decrito. En uno de ellos el valor de los CAA decrece con la señal de detección de posición, mientras que en el otro crece para la misma condición de movimiento. De esta forma, aún cuando la dirección de movimiento sea diferente a la prevista uno de los dos CAA parará el reloj cuando ocurra el Ds prefijado.

DESCRIPCCION DEL SISTEMA

Los componentes del tacómetro digital son:

i) Los CAA.
ii) Los relojes.
iii) Los dispositivos de control y entrefaz con la computadora.

Contadores hacia abajo y hacia arriba (CAA)

Los CAA son dispositivos diseñados para modificar el valor contenido en ellos cada vez que reciben un pulso eléctrico. Si este pulso se recibe por la terminal de cuenta hacia arriba, el valor del CAA se incrementa. Lo contrario sucede cuando el pulso se recibe por la terminal de cuenta hacia abajo.

Los CAA tienen además señales para indicar cuenta máxima y cuenta mínima, respectivamente. Con esta última señal se controla el arranque y paro de los relojes que miden el tiempo.

Otra característica importante de los CAA, es que el valor que contienen puede ser prefijado a uno conocido. De esta forma, si se prefijan con el número de evnetos equivalente al Ds deseado, y se presenta una señal de

cuenta mínima, el desplazamiento ha ocurrido.

Relojes

Los relojes empleados son contadores hacia abajo cuyo valor decrece una unidad cada vez que ocurre un ciclo de un reloj alimentador, en este caso se utiliza el reloj interno de la Apple II que opera a 1 MHz. aproximadamente, por lo que las mediciones tienen una precisión máxima de +/- 1 microsegundo.

El valor contenido en los contadores de los relojes también se puede prefijar. Esto permite conocer el tiempo Dt transcurrido para el Ds como la diferencia entre el valor inicialmente prefijado y el que se lee cuando el reloj se detiene por acción de la señal de cuenta mínima de los CAA.

Dispositivos de control e interfase con la computadora

Estos dispositivos son: memorias de un bit, alimentadores de tres estados y decodificadores. Su función es la de permitir que la microcomputadora Apple II pueda realizar las siguientes tareas:
a) modificar el estado de los CAA; b) arrancar y parar los relojes; c) identificar el CAA que produjo la señal de cuenta mínima.

OPERACION

El tacómetro digital se diseñó como un periférico de la Apple II, por lo que su operación ocurre a través de instrucciones ordenadas por el microprocesador 6502.

La Apple II cuenta con 7 ranuras para colocar periféricos. El tacómetro se puede colocar en cualquiera de ellas que esté libre, aunque se recomienda utilizar la cuarta si se quieren emplear directamente los programas desarrollados para probar el funcionamiento del tacómetro digital. La dirección base que corresponde a cada ranura es:

$ C0 (8+N) 0

donde N=número de ranura.

Cada ranura tiene asociadas 16 localidades contadas a partir de la dirección base, que se alcanzan mediante la combinación de las líneas A0 a A3 del microprocesador. Se cuenta además con una señal de habilitación del periférico (DS) qu indica que la dirección contenida en el procesador corresponde a alguna de la 16 mencionadas. El uso apropiado de esta última señal permite inhibir el acceso al tacómetro cuando no se hace referencia explícita a él, e impide modificaciones indeseadas en su funcionamiento.

De las señales disponibles en la ranura se usan las siguientes:

A0 a A3 - líneas de dirección menos significativas.

D0 a D7 - bus de datos.

DS - selección de ranura.

RS - restablecimiento.

R/W - lectura o escritura.

IRQ - petición de atención por interrupción.

0 - señal de sincronía del procesador.

A continuación se explica la forma de acceso a los distintos dispositivos del tacómetro digital, tomando como base la líneas A0 a A3 y R/W.

Contadores de tiempo

Los contadores de tiempo se basan en el circuito integrado INTEL 8253. Exite uno de ellos acoplado a cada conjunto de CAA y se denominarán 8253-1 y 8253-2, respectivamente.

Para accesar al 8253-1 se debe tener:

R/W	A3	A2	A1	A0
X	0	0	X	X

X=0,1; indistintamente

y para el 8253-2 se debe tener:

R/W	A3	A2	A1	A0
X	0	1	X	X

X=0,1; indistintamente

Para cada 8253 se pueden realizar operaciones que tienen por objeto escribir o leer los tres contadores con que cuenta el 8253, así como programar los modos de funcionamiento. La operación específica que se realiza depende de los valores de A0, A1 y R/W, según se muestra a continuación:

R/W	A1	A0	OPERACION
0	0	0	carga contador 0
0	0	1	carga contador 1
0	1	0	carga contador 2
0	1	1	modo de funcionamiento
1	0	0	lee contador 0
1	0	1	lee contador 1
1	1	0	lee contador 2
1	1	1	ninguna-sin uso

De las operaciones anteriores, la de modo de funcionamiento es la más importante. Cuando se accesa a la dirección asociada a esta operación el bus de datos se traslada al registro de control de los relojes. El significado de cada uno de los ocho bits de este registro se basa en la

siguiente definición:

D7	D6	D5	D4	D3	D2	D1	D0
SC1	SC0	RL1	RL0	M2	M1	M0	BCD

Donde SC1 y SC0 son los bits de selección de contador e indican cual de los contadores será programado según el valor del resto de los bits (D0 a D5), su valor se interpreta de acuerdo con la siguiente tabla:

SC1	SC0	EFECTO
0	0	selecciona contador 0
0	1	selecciona contador 1
1	0	selecciona contador 2
1	1	NO VALIDO

RL1 y RL0 son los bits que indican operaciones de cargado y lectura en los contadores, según se muestra a continuación:

RL1	RL0	EFECTO
0	0	lectura instántanea sin paro
0	1	carga o lee sólo el BMS
1	0	carga o lee sólo el BPS
1	1	carga lo lee secuencia BPS-BMS

BMS-byte más significativo

BPS-byte menos significativo

M2, M1 y M0 seleccionan el modo de operación como sigue:

M2	M1	M0	MODO DE OPERACION
0	0	0	modo 0
0	0	1	modo 1
X	1	0	modo 2
X	1	1	modo 3
1	0	0	modo 4
1	0	1	modo 5

De estos modos se seleccionó el modo 2 para los contadores 0 y 1. Este modo permite el funcionamiento repetitivo de ambos contadores. El contador 2 no se emplea, por lo que está disponible para otra aplicación.

El último bit BCD señala el formato de los caracteres para contar:

BCD	FORMATO DE CUENTA
0	cuenta en binario
1	cuenta BCD

Cargado y lectura de los contadores

Estas operaciones se deben realizar siempre después de una operación de selección de modo de funcionamiento. El número de bytes que se lee o escribe debe coincidir con lo indicado con los bits RL0 y RL1.

Caben tres aclaraciones importantes:

i) Cuando se utiliza la opción RL0=RL1=0 el valor del contador en cuestión se traslada a un registro temporal de donde puede ser leido después en forma normal.

ii) Es posible dar más de un comando de modo de funcionamiento en forma consecutiva, es decir, sin que medien operaciones de lectura y escritura. Se debe dar después las órdenes de lectura y escritura que se señalaron en dichos modos, sin embargo debe existir total correspondencia entre el número de estas órdenes y la secuencia de modos de funcionamiento programada.

iii) Los relojes requieren un ciclo completo del reloj de alimentación para precargar los valores de los contadores a un valor que corresponda a la orden de escritura. Esto significa que en señales de sincronía para los relojes relativamente lentas se debe tomar en cuenta dicho ciclo. Además para arreglos en cascada de dos contadores, la restricción anterior impone limitaciones pues para cargar el último contador de la cascada se deben diseñar rutinas especiales para el efecto.

Contadores hacia arriba y hacia abajo (CAA)

Estos contadores se construyeron con tres circuitos integrados 74LS193 en cascada, lo que permite contar hasta 4095 eventos. Existe un contador acoplado a cada 8253. Las señales de alimentación de los CAA provienen de los detectores de posición, previa etapa de codificación. Bajo la misma convención que el caso anterior se ha denominado a los dos conjuntos de 74LS193, CAA-1 y CAA-2.

Las operaciones en los CAA sirven para prefijar su valor, como sólo se pueden realizar operaciones de escritura, R/W=0 en todos los casos.

Para accesar al CAA-1 se debe tener:

R/W	A3	A2	A1	A0
0	1	0	0	X

además si:

X=0 ;se tiene acceso a los 8 bits menos significativos del CAA1 (D0 a D7).

X=1 ;se tiene acceso a los 4 bits más significativos del CAA-2 . (D8-D11).

Para accesar al CAA-2 se debe tener:

R/W	A3	A2	A1	A0
0	1	1	0	X

donde X se interpreta igual que para el CAA-1.

Para un funcionamiento correcto de los CAA se deben emplear pulsos de cuenta con lógica invertida, es decir, con un nivel normalmente alto. Cuando ocurre un nivel bajo en ambas entradas los CAA cuentan de forma errática, por lo que deben evitarse pulsos simultáneos en las terminales de cuenta arriba y cuenta abajo de ellos.

Identificadores de fuente de señal mínima

Este dispositivo produce la señal IRQ al microprocesador cuando alguno de los CAA llega a su valor mínimo y permite verificar cual de ellos fue el que produjo dicha señal. Se implantaron con base en los circuitos integrados 74LS74 y 74LS125. El primero se compone de dos biestables tipo D, a los que se denomina 74-1 y 74-2. El último se compone de cuatro acopladores inversores con salida de tres estados, y se le denominó 125. Sólo se permiten operaciones de escritura en los 74 y de lectura en el 125.

Los 74-1 y 2 usan las señales de cuenta mínima de ambos CAA como reloj de cambio de estado. La salida de cada uno se conecta a la terminal de arranque y paro de un 8253 y a la entrada de una compuerta del 125. La entrada D a los 74 tiene un nivel normalmente alto, para cumplir con los requerimientos de la señal de paro del 8253. El señal invertida necesaria para el IRQ se consigue a través de la inversión del 125. Además se usan las señales de borrado y prefijado del 74 para arrancar y parar los relojes en cualquier momento.

Cuando el microprocesador detecta la señal IRQ la secuencia que siempre

se debe seguir es: lectura del 125 y escritura sobre los 74.

El acceso al 125 se consigue cuando se tiene:

R/W	A3	A2	A1	A0
1	1	0	0	0

en estas condiciones se trasladan los valores de las salidas Q de los 74-1 y 74-2 a D0 y D1, respectivamente. Estos últimos bits se deben analizar de la siguiente forma:

D1	D0	SIGNIFICADO
1	1	no hay señal de cuenta mínima
1	0	cuenta mínima en el CAA-1
0	1	cuenta mínima en el CAA-2
0	0	cuenta mínima en ambos CAA

Para lograr el acceso al 74-1 se debe escribir sobre cualquiera de las localidades que siguen:

R/W	A3	A2	A1	A0
0	1	X	1	0

si A2=0 la escritura de cualquier dato coloca la salida del biestable 74-1 quede en estado lógico alto; si A2=1 la escritura produce el efecto contrario en el biestable.

El acceso al 74-2 se consigue si:

R/W	A3	A2	A1	A0
1	1	X	1	1

el efecto de la escritura es igual que en el 74-1.

La siguiente tabla resume el papel de las señales de control.

DS	R/W	A3	A2	A1	A0	DISPOSITIVO ACCESADO
0	X	0	0	X	X	8253-1
0	X	0	1	X	X	8253-2
0	0	1	0	0	0	CAA-1 (BPS)
0	0	1	0	0	1	CAA-1 (BMS)
0	0	1	0	1	0	74-1 (precarga)
0	0	1	0	1	1	74-2 (precarga)
0	0	1	1	0	0	CAA-2 (BPS)
0	0	1	1	0	1	CAA-2 (BMS)
0	0	1	1	1	0	74-1 (borrado)
0	0	1	1	1	1	74-2 (borrado)
0	1	1	0	0	0	125

A continuación se muestran los diagramas lógico y de disposición de componentes del tacómetro digital.

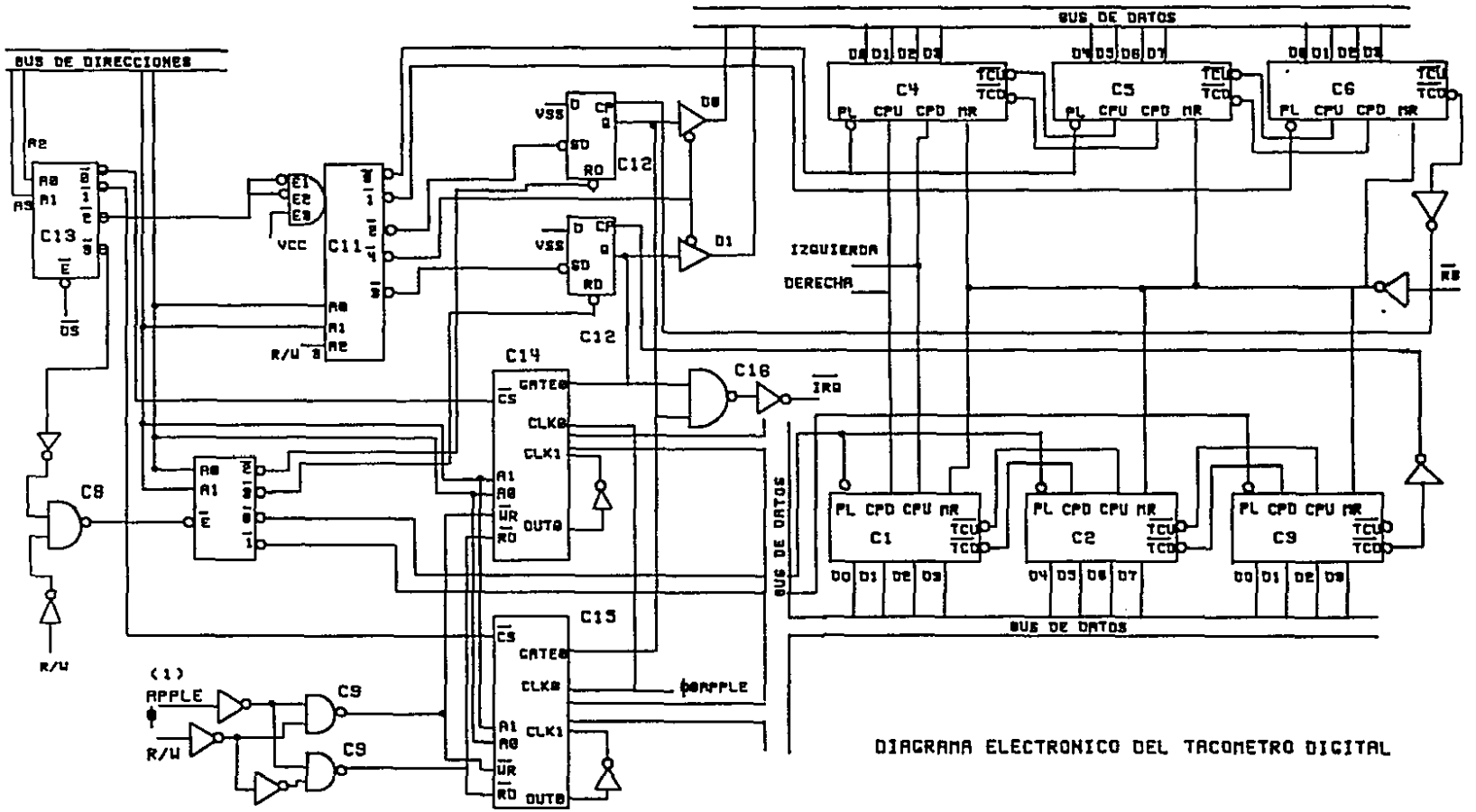
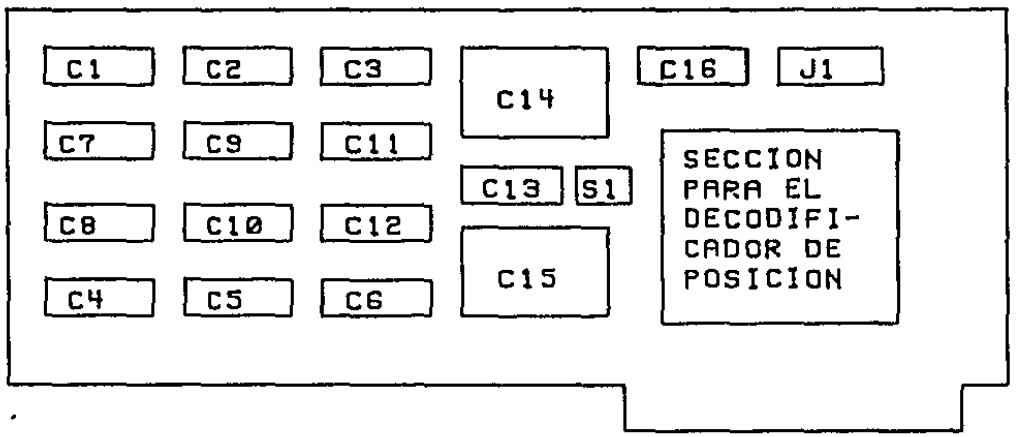


DIAGRAMA ELECTRONICO DEL TACOMETRO DIGITAL



C1 A C6: 74LS193
C7 Y C8: 74LS04
C9: 74LS00
C10: 74LS125
C11: 74LS138
C12: 74LS74
C13: 74LS139
C14 Y C15: 8253
C16: 74LS05
S1: MICROINTERRUPTOR
J1: CONECTOR AL DETECTOR DE POSICION

[1] ORIENTACION DE LOS CIRCUITOS

DIAGRAMA DE DISPOSICION DE COMPONENTES DEL TACOMETRO DIGITAL

RELOJ

DESCRIPCION GENERAL

El reloj que se emplea para enviar los pulsos a los motores, según la tabla de tiempos de conmutación, se basa en el circuito integrado MC-6840 de Motorola. Este dispositivo consta de tres contadores de 16 bits. Para la aplicación que se describe se colocaron dos de ellos en cascada. El primero genera a partir del reloj de la Apple II, una señal de 10, 100, 1,000 y 10,000 microsegundos de periodo. El segundo produce una señal de interrupción cuando se agota un valor prefijado en el contador. De esta forma la combinación de ambos contadores permite generar los pulsos con una precisión que es proporcional al tiempo entre pulsos.

El circuito se colocó en otra de las ranuras de la Apple II, y se ocuparon las ocho primeras localidades de memoria asignadas a ella. A continuación se describe el papel de cada una de las localidades según los valores de A0, A1, A2 y R/W:

R/W	A2	A1	A0	OPERACION
0	0	0	0	escribe registros control 1 y 3 (*)
0	0	0	1	escribe registro control 2
0	0	1	0	escribe BMS reloj 1
0	0	1	1	escribe BPS reloj 1
0	1	0	0	escribe BMS reloj 2
0	1	0	1	escribe BPS reloj 2
0	1	1	0	escribe BMS reloj 3
0	1	1	1	escribe BPS reloj 3

(*) el registro de control sobre el cual se escribe depende del bit cero del registro de control 2 (ver tabla más adelante)

R/W	A2	A1	A0	OPERACION
1	0	0	0	NINGUNA
1	0	0	1	lee registro de estado
1	0	1	0	lee BMS reloj 1
1	0	1	1	lee BPS reloj 1
1	1	0	0	lee BMS reloj 2
1	1	0	1	lee BPS reloj 2
1	1	1	0	lee BMS reloj 3
1	1	1	1	lee BPS reloj 3

REGISTROS DE CONTROL DE LOS RELOJES

Los relojes se manejan a través de un registro de control de 8 bits. Se llamará CRI-J al j-ésimo bit del i-ésimo registro de control. La interpretacion de estos bits es como sigue.

CR1-0	BIT DE RESET INTERNO
0	arranca todos los relojes
1	para todos los relojes

CR2-0	APUNTADOR A CR1 Y CR3
0	apunta a CR3
1	apunta a CR1

CR3-0	BIT DE PREESCALAMIENTO
0	reloj 3 no preescalado
1	reloj 3 preescalado por 8

CRX-1	FUENTE DE TIEMPO PARA LOS RELOJES
0	reloj externo (señal en CX)
1	reloj de la Apple II

CRX-2	MODO DE CUENTA
0	palabra de 16 bits (simétrica)
1	dos palabras de 8 bits (asimétrica)

CRX-6	CONTROL DE INTERRUPCIONES
0	no produce interrupciones
1	produce interrupción (contador nulo)

CRX-7	CONTROL DE SALIDA
0	inhibe salida OX de reloj X
1	permite salida OX de reloj X

Los bits CR3 a CR5 se interpretan según la siguiente tabla

CRX-3	CRX-4	CRX-5	MODO DE OPERACION
0	X	0	continuo
0	X	1	disparo simple
1	0	X	comparación de frecuencia
1	1	X	comparación ancho de pulso

REGISTRO DE ESTADO

El MC-6840 está dotado con un registro de estado (RE) de ocho bits, de los cuales sólo se pueden interpretar cuatro: RS-0,RS-1,RS-2 y RS-7. Su significado es el siguiente:

RS-I	BIT INTERRUPCION RELOJ I
0	no interrupción reloj I
1	interrupción reloj I

I=1,2,3

RS-7	BIT INTERRUPCION GENRAL
0	ningún reloj interrupme
1	algún reloj interrumpe

Cuando los relojes producen una señal IRQ al microprocesador, este puede eliminarla de las siguientes formas:

- con una señal de restablecimiento (reset)
- con CR1-0=1
- por lectura de un contador de un reloj
- por escritura a un alimentador de un reloj (sólo RS-0 a RS-2)
- por inicalización de contadores (sólo RS-0 a RS-2)

INICIALIZACION DE LOS CONTADORES DE LOS RELOJES

Se deben escribir los dos bytes en las localidades indicadas en la tablas anteriores. La secuencia siempre debe ser: primero los bits más significativos y después los menos.

La señal de restablecimiento del sistema (reset) pone los contadores en máxima cuenta, mientras que el reset interno (CR1-0=1) no los altera.

Los relojes se cargan con el valor de sus alimentadores cuando:

- se borra la bandera individual de interrupción RS-0 a RS-2)
- se presenta un reset externo o interno
- cuando el valor de contador vale cero y hay transición negativa del reloj de cuenta.

A continuación se proporciona el diagrama de conexiones para el reloj.

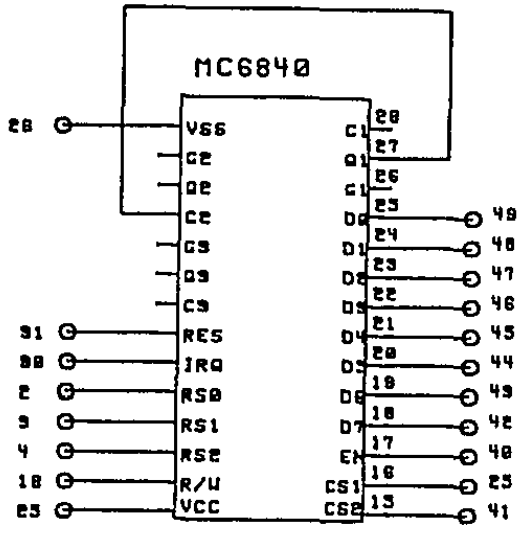


DIAGRAMA DE CONEXIONES DEL RELOJ

—o AL CONECTOR EN LAS RANURAS DE LA APPLE II

DETECTOR DE POSICION

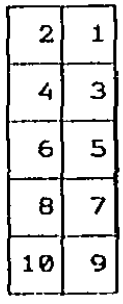
El detector de posición que se utilizó para las pruebas de los motores de pasos es el modelo HEDS-5000 producido por Hewlett Packard. Las características más importantes de este detector son:

- Detecta posición relativa.
- Dos canales de salida (defasados 90 grados uno respecto al otro).
- 500 ciclos por revolución.
- Alimentación TTL compatible (0-5 volts).
- Salidas LSTTL compatibles (0-5 volts, bajo consumo).

El detector se colocó en una base especial, sobre una flecha construida para el efecto. Esta última se acopló al motor a través de una flecha desmontable, sobre la que se coloca la inercia que debe mover el motor, que como ya se ha mencionado puede variar de un experimento a otro.

Información adicional sobre el detector se puede consultar en el catálogo del fabricante.

El detector de posición se conecta a través de un conector para cable plano estándar de 10 alfileres. La disposición de los mismos se muestra a continuación.



1 - Canal A
2 - 5 volts
3 - 0 volts
4 - s/c
5 - s/c
6 - 0 volts
7 - 5 volts
8 - Canal B
9 - 5 volts
10 - s/c

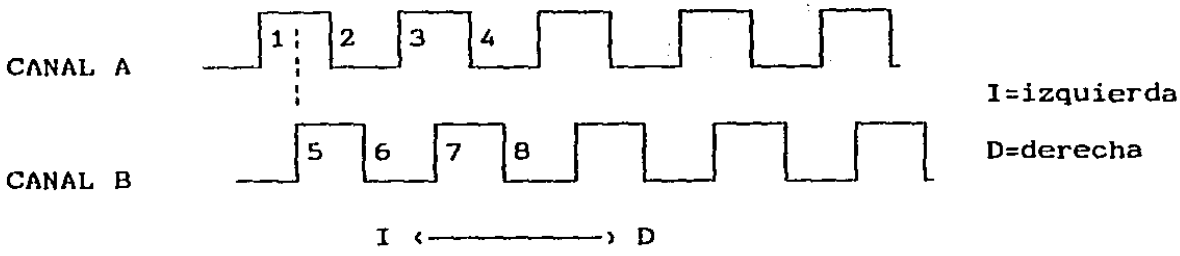
CONECTOR DEL HEDS-5000

s/c = sin conexión

CODIFICADOR DE POSICION

La salida del detector de posición se debe codificar para convertir las señales de entrada de los dos canales, en otro par de señales que indiquen ahora movimiento asociados con cada una de las direcciones de giro angular, respectivamente.

La forma de las señales de salida del detector de posición se muestra a continuación:



en estas señales se pueden detectar ocho transiciones de estado independientes. Cuatro de ellas corresponden al análisis de las transiciones del canal B, considerando fijo el canal A, el resto a la consideración inversa. Las transiciones que se pueden codificar según se muestra en la siguientes tablas:

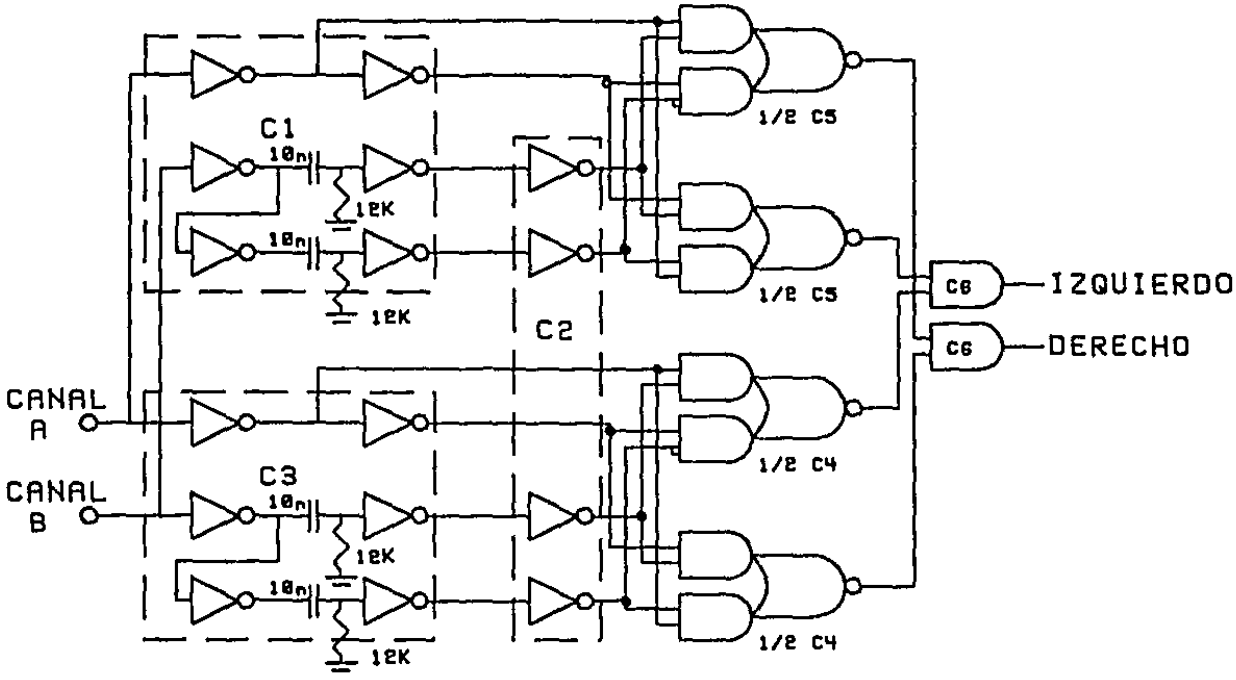
CANAL A(fijo)

CANAL B (transición)	Ø	1
Ø->1	D	I
1->Ø	I	D

CANAL B(fijo)

CANAL A (transición)	Ø	1
Ø->1	I	D
1->Ø	D	I

El codificador anterior se implantó a través del circuito que se muestra a continuación. En él se distinguen dos porciones simétrica; cada una corresponde. a la implantación de una de las tablas que se mostraron arriba. Cada sección consta de compuertas inversoras con banda de histéresis (schmitt trigger) y de compuertas lógicas Y e O negadas. Las señales de ambas secciones se mezclan a través de dos compuertas Y.



C1 A C3: MC14584
C4 Y C5: 74LS51
C6: 74LS08

DIAGRAMA ELECTRONICO DEL DECODIFICADOR

El diagrama de disposición del codificador se muestra a continuación. Físicamente el codificador se colocó en la misma tarjeta que el tacómetro digital.

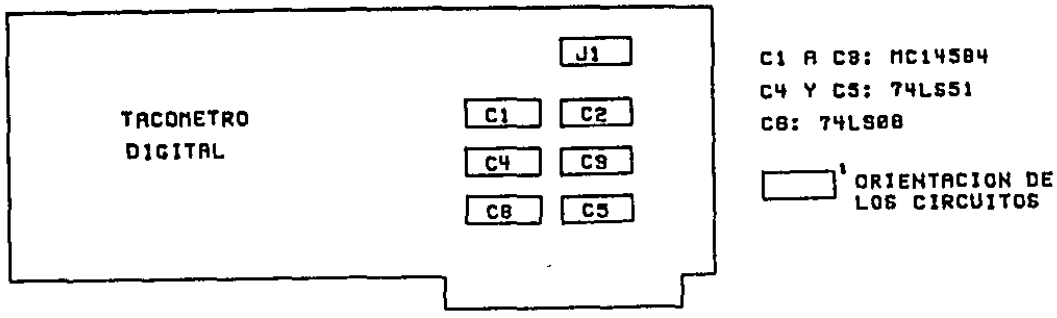


DIAGRAMA DE DISPOSICION DE COMPONENTES
DEL DECODIFICADOR DE POSICION

El codificador se conecta al detector de posición por medio de un cable plano, que tiene por un extremo un conector compatible con el del HEDS-5000, y por el otro un conector tipo base de circuito integrado de dieciseis alfileres en dos hileras (DIP 16). La disposición de las señales en este conector se muestra a continuación.

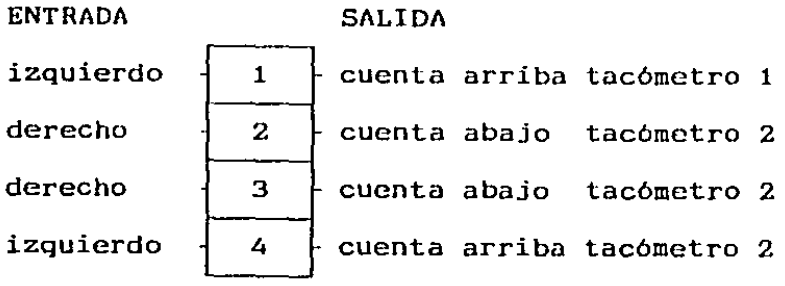
5 volts -	1	16	- canal A
s/c -	2	15	- 0 volts
0 volts -	3	14	- s/c
canal B -	4	13	- 5 volts
s/c -	5	12	- 5 volts
s/c -	6	11	- s/c
s/c -	7	10	- s/c
s/c -	8	9	- s/c

CONECTOR DEL
TACOMETRO DIGITAL

s/c = sin conexión

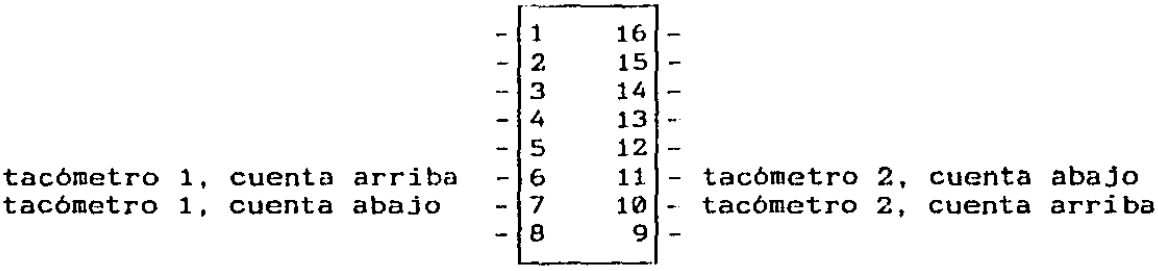
La conexión entre el tacómetro digital y el codificador de posición se realizó a través de cuatro interruptores de un polo un tiro. La intención de estos interruptores es la de desconectar la sección de codificación del tacómetro digital, para permitir el empleo de este último en experimentos en que la señal se obtenga de otra forma.

La disposición de estos interruptores se muestra a continuación:



INTERRUPTORES DE ACOPLAMIENTO EL CODIFICADOR DE POSICION Y EL TACOMETRO DIGITAL

Cuando se quiere eliminar el efecto del codificador, se debe abrir los cuatro interruptores mencionados. Se debe conectar la nueva señal que se desea medir en el mismo conector que acopla al detector con el codificador de posición, según la distribución que se muestra a continuación

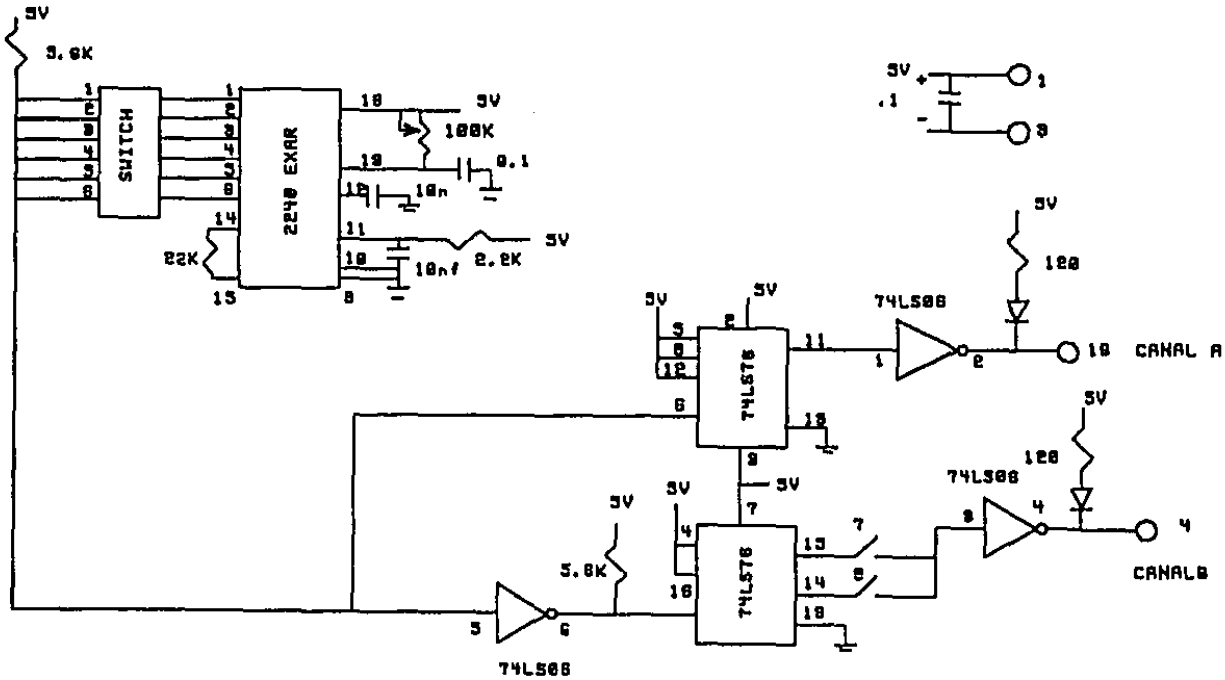


EMULADOR DEL DETECTOR DE POSICION

Con el fin de facilitar las pruebas del tacométro digital y del codificador de posición, se construyó un emulador del funcionamiento del detector de posición.

Este emulador se construyó con base en un oscilador EXAR-2240 alimentado por un circuito RC. Este circuito actua como un divisor de frecuencias, con seis salidas que corresponden a divisiones binarias de la frecuencia básica de alimentación. Se colocó un juego de ocho interruptores (DIP SWITCH-8). Seis de ellos permiten seleccionar una de las frecuencias de división, mientras que los otros dos sirven para simular el efecto de cambio de dirección. Se colocaron además dos biestables para filtrar las salidas del divisor.

El diagrama del emulador se muestra a continuación. Las conexiones al codificador de posición se realizan de la misma forma que con el detector real.



EMULADOR DEL DETECTOR DE POSICION

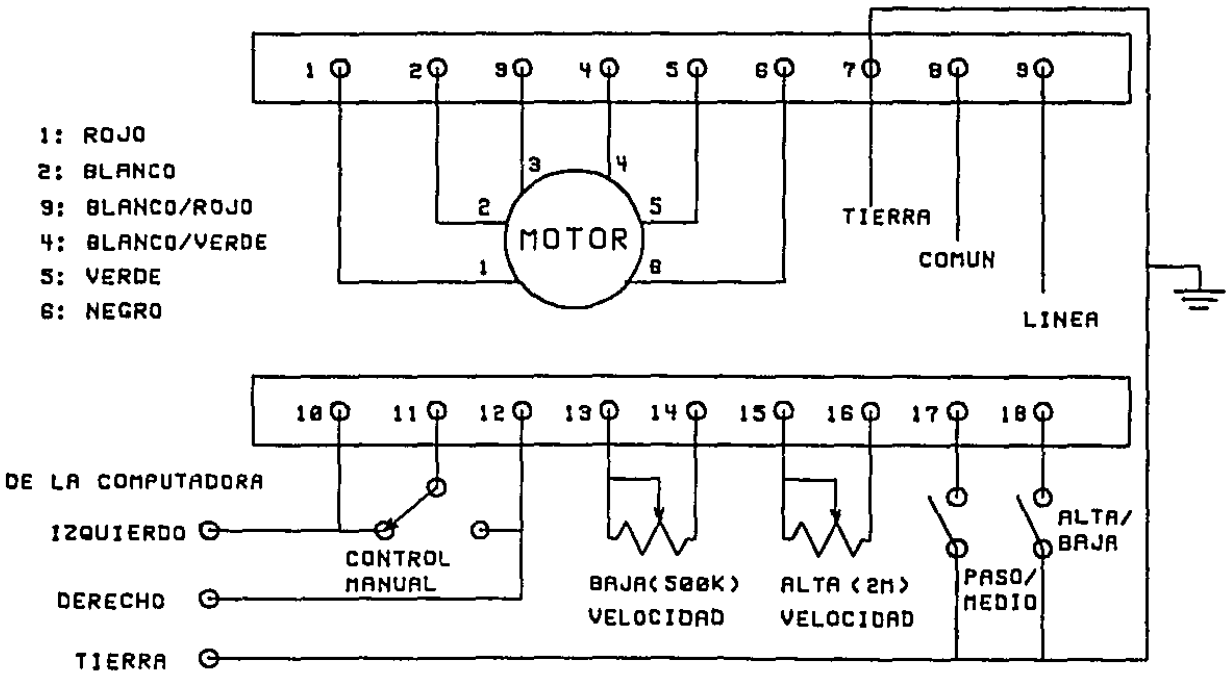
ACOPLAMIENTO DE POTENCIA PARA EL MOTOR DE PASOS

El acoplamiento de potencia que se empleó para manejar el motor de pasos bajo prueba es el modelo TBM105 que produce The Superior Electric Company.

Sus características más importantes son:

- Tipo voltaje dual
- Señales de alimentación TTL compatibles, lógica invertida.
- Velocidad máxima de operación 5000 pasos/segundo (pasos completos)
  10000 pasos/segundo (medios pasos)
- Corriente de salida por fase 5 amperes.

Las conexiones básicas se muestran a continuación.



ANEXO D

DATOS DE LOS EXPERIMENTOS

A continuación se muestran los datos para los experimentos que se mostraron el el capítulo 9. Se incluyen los parámetros y las constantes de las condiciones deseadas de movimiento para el motor y se presentan tabulados para facilitar su interpretación.

TABLA DE PARAMETROS PARA LOS EXPERIMENTOS CON MOTORES DE PASOS

		EXPERIMENTO No.				
VARIABLE	UNIDAD	1	2	3	4	5
No. Fases		4	4	4	4	4
Resolución	Rad.	0.0314159	0.0314159	0.0314159	0.0314159	0.0314159
Par máx. disponible	KDina cm	17675	17675	17675	17675	17675
Pérdida de par(1)	KDina cm s	168	81	81	81	81
Relación de reducción		1	1	1	1	1
Tolerancia (2)	Rad.	0.00314159	0.00314159	0.00314159	0.00314159	0.00314159
Constante posición (3)		2000	2000	2000	4000	4000
Constante velocidad (3)		200	400	400	300	300
Fricción seca	KDina cm	0	0	0	0	0
Fricción viscosa	KDina cm s	636	0	0	0	0
Radio inercia disco	cm	4.3	4.3	4.3	4.3	4.3
Masa inercia disco	Kg	0.355	0.355	0.355	0.355	0.355
Longitud brazo	cm	0	0	0	0	0
Masa del brazo	Kg	0	0	0	0	0
Radio masa concentrada	cm	0	0	0	0	0
Masa concentrada	Kg	0	0	0	0	0
Impresión (4)		5	5	5	5	5
Condiciones deseadas (5)		8	8	8	8	8
Conmutación (6)		7	7	7	7	7
Incremento tiempo (7)	s	0.0001	0.00005	0.00001	0.00004	0.00004
Incremento inercia	%	0	0	0	0	0

(1) Según acoplamiento de potencia usado
(2) Para detectar el cruce por cero en curvas par-posición
(3) Para el algoritmo de control
(4) No. de periodos de simulación para impresión de resultados
(5) No. d periodos de simulación para cálculo de condiciones deseadas
(6) No. de periodos de simulación para recalcular par requerido
(7) Periodo para la simulación

VARIABLE	UNIDAD	6	7.1	7.2	7.3	7.4
No. Fases		4	4	4	4	4
Resolución	Rad.	0.0314159	0.0314159	0.0314159	0.0314159	0.0314159
Par máx. disponible	KDina cm	17675	17675	17675	17675	17675
Pérdida de par(1)	KDina cm s	81	81	81	81	81
Relación de reducción		1	1	1	1	1
Tolerancia (2)	Rad.	0.00314159	0.00314159	0.00314159	0.00314159	0.00314159
Constante posición (3)		4000	4000	2000	2000	2000
Constante velocidad (3)		1800	1800	400	400	400
Fricción seca	KDina cm	0	0	0	0	0
Fricción viscosa	KDina cm s	0	0	0	0	0
Radio inercia disco	cm	4.3	4.3	4.3	4.3	4.3
Masa inercia disco	Kg	0.355	0.648	0.648	0.648	0.648
Longitud brazo	cm	0	0	0	0	0
Masa del brazo	Kg	0	0	0	0	0
Radio masa concentrada	cm	0	0	0	0	0
Masa concentrada	Kg	0	0	0	0	0
Impresión (4)		5	5	5	5	5
Condiciones deseadas (5)		8	8	8	8	8
Conmutación (6)		7	7	7	7	7
Incremento tiempo (7)	s	0.00003	0.00003	0.00003	0.00003	0.00003
Incremento inercia	%	0	0	16	31	-14

VARIABLE	UNIDAD	7.5	8
No. Fases		4	4
Resolución	Rad.	0.0314159	0.0314159
Par máx. disponible	KDina cm	17675	
Pérdida de par(1)	KDina cm s	81	
Relación de reducción		1	
Tolerancia (2)	Rad.	0.00314159	
Constante posición (3)		2000	
Constante velocidad (3)		400	
Fricción seca	KDina cm	0	
Fricción viscosa	KDina cm s	0	
Radio inercia disco	cm	4.3	
Masa inercia disco	Kg	0.648	
Longitud brazo	cm	0	
Masa del brazo	Kg	0	
Radio masa concentrada	cm	0	
Masa concentrada	Kg	0	
Impresión (4)		5	
Condiciones deseadas (5)		8	
Conmutación (6)		7	
Incremento tiempo (7)	s	0.00003	
Incremento inercia	%	-29	