UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

DESARROLLO E IMPLEMENTACION DE UN CONTROL DE POSICION/VELOCIDAD, PARA UN MANIPULADOR ARTICULADO

T E S I S

QUE PARA OBTENER EL TITULO DE

INGENIERO MECANICO ELECTRICISTA

P R E S E N T A :

GUSTAVO MONTEMAYOR GARCIA

D I R E C T O R D E T E S I S :

ING. ROBERTO MANDUJANO WILD

Ciudad Universitaria, México D.F. 1997

A mis padres

A mi hermana

Agradezco al Ing. Roberto Mandujano Wild, mi asesor, por la paciencia y la compresión mostradas durante el proceso de elaboración de esta tesis.

Agradezco al Ing. Francisco Rodríguez Ramírez por sus comentarios y anotaciones para la parte de control.

Agradezco al "Laboratorio de Control" de la División de Estudios de Posgrado de la Facultad de Ingeniería de la UNAM por las facilidades prestadas para la realización de este proyecto.

INDICE

R	Unión de revolución.
P	Unión prismática.
GDL	Grado de libertad.
D_H	Representación Denavit_Hartenberg.
Rot_{K,θ_i}	Rotación alrededor del eje "K", θ_i grados.
$Tras_{K,d_i}$	Traslación en dirección del eje "K", una distancia d_i.
T_0^n	Transformación homogénea del marco de referencia "n" al marco "0".
A_i	Transformación homogénea del marco de referencia "i" al marco "i-1".
T_i^j	Transformación entre el marco de ref. "j" al marco de ref. "i".
d_j^i	Vector de desplazamiento, entre el origen del marco de referencia "i" al origen del marco de referencia "j".
R_j^i	Matriz de rotación, entre la orientación del marco de referencia "i" y la orientación del marco de referencia "j".
$(R_j^i)^T$	Transpuesta de la matriz de rotación.
a_i, α_i, d_i y θ_i	Parámetros de la representación D_H.
θ, ϕ, ψ	Angulos de Euler.
X	Posición (y orientación) del actuador final.
q	Coordenadas generalizadas.

$\mathbf{J}$	Jacobiano del manipulador.
$\mathbf{J}_i$	Jacobiana del elemento "i"
$\mathbf{J}_v$	Parte de la jacobiana (velocidad lineal).
$\mathbf{J}_w$	Parte de la jacobiana (velocidad angular).
$\mathbf{I}$	Tensor de inercia.
ω	Velocidad angular.
α	Aceleración angular.
$\mathbf{V}_c$	Velocidad lineal del centroide.
$\mathbf{D}$	Matriz de inercia.
$\mathbf{OC}_i$	Coordenadas del centroide del elemento "i", de acuerdo al marco de referencia del elemento.
τ_k, f_k	Par y fuerza en la unión "k".
τ_{comp}	compensación anticipativa
R_a	Resistencia de armadura (motor de corriente directa).
i_a	Corriente de armadura.
K_b	Constante de la fuerza contra-electromotriz.
K_a	Constante de par/corriente.
K_R	Constante de proporcionalidad par/voltaje.
J_{eff}	Inercia efectiva de la unión.
B_{eff}	Amortiguamiento relativo en la unión.

Abreviaturas y notación

θ_m	Movimiento angular en el eje del motor.
$\theta_c = q$	Movimiento angular en el lado de la carga.
n	Relación de transformación de la transmisión de potencia.
K_1	Ganancia del controlador (integrador).
K_2	Ganancia del controlador (retroalimentación del estado).
K_e	Ganancia del controlador (observador).

$C_\theta = \cos(\theta)$ $\quad C_{\theta\phi} = \cos(\theta + \phi)$

$S_\theta = sen(\theta)$ $\quad S_{\theta\phi} = sen(\theta + \phi)$

CAPÍTULO 1

INTRODUCCIÓN

El objetivo de esta tesis se centra en el desarrollo e implementación del control de posición de un manipulador articulado tipo codo, de seis grados de libertad, en el que la información de los puntos del movimiento esté en coordenadas cartesianas.

Un manipulador o robot -como usualmente se le denomina-, es un sistema muy utilizado en la automatización de muchas industrias, entre ellas la automotriz, la electrónica, la metal_mecánica; y su uso es un fenómeno propiciado por las presiones sobre los sistemas de manufactura, mediante los cuales se tiene que producir bienes de alta calidad, a tiempo, y a costos razonables, en razón de una diversidad de productos que continuamente están cambiando.

Entre las ventajas que resultan del uso de robots podemos destacar las siguientes: el incremento de la producción, una reducción de la mano de obra directa, mejoras en el control de los procesos involucrados, como también que aleja a los operadores de ambientes peligrosos. Tales ventajas, sin embargo, no son inmediatas, ya que los manipuladores son sistemas muy complejos que requieren de un laborioso trabajo alrededor de ellos, como la adaptación del proceso, su calibración, su programación, la estructuración de su ambiente, etc.

Para lograr la versatilidad requerida para un robot se necesita de un sistema digital que establezca los movimientos y los cambios de acción ante las diversas circunstancias que surjan en el desempeño de las tareas además de algún esquema de control que haga posible seguir una trayectoria con la rapidez y precisión requeridas ante las perturbaciones e incertidumbres que se presenten en el sistema.

Un manipulador es un sistema compuesto de varias partes, cuyo comportamiento, entre más complejo sea, requiere de subsistemas más sofisticados, como sistemas digitales más potentes, sensores que supervisen el medio ambiente alrededor del manipulador, algoritmos de control más complejos, etc., además se requiere de una gran integración para que todo funcione eficientemente. Lo anterior implica que, para cualquier desarrollo en el campo de la robótica, se requiere de diversas habilidades, desde el diseño mecánico hasta posiblemente el procesamiento digital de señales, pasando por teoría de control, sistemas digitales y muchas otras disciplinas. Por ello, el campo de la tesis se limitará al desarrollo e implementación del control de un manipulador, dejando a un lado el diseño mecánico, el uso de sensores distintos de los exclusivos al lazo de control, etc.

Se centrará asimismo en los manipuladores articulados, es decir, en aquellos que cuentan con uniones giratorias cuya estructura cinemática semeja un brazo humano, pues cuentan con hombro, codo y muñeca. Estos manipuladores son de los más maniobrables y pueden, por tanto, realizar muchas tareas, aunque por otro lado son de los más difíciles de controlar, pues sus movimientos están acoplados dinámicamente. De los manipuladores articulados sólo se considerará a los denominados "de tipo codo", y, dentro de éstos, a los actuados por motores de corriente directa de imán permanente y que utilicen únicamente sensores de posición para su control.

El trabajo realizado se dividió en siete capítulos principales, el primero de los cuales es una introducción en la que a grosso modo se enuncia el tema del presente trabajo. En el capítulo dos, se sitúa a los manipuladores articulados en un contexto general, para lo cual se investigan las diferentes características de los manipuladores a nivel de los diferentes subsistemas que los componen, de los parámetros importantes en el funcionamiento de éstos y de sus diferentes formas de clasificación. En el capítulo tres se desarrollan los análisis cinemáticos y dinámicos para un manipulador articulado, de seis grados de libertad, tipo codo, con un offset en la segunda unión.

En el capítulo cuatro se desarrolla el algoritmo de control, proceso en el que primero es necesario obtener el modelo de un motor de CD acoplado a una transmisión de potencia, cuya carga está dada por el comportamiento dinámico del manipulador, y después establecer la controlabilidad y observabilidad del sistema para posteriormente desarrollar el sistema de control.

Dentro del capítulo cinco se especifican las necesidades con base en las cuales se va a desarrollar la arquitectura, se dan las características generales de ésta y se describe el hardware necesario para soportar estas características. Por último se muestra la implementación del controlador. El capítulo seis comprende el diseño de las rutinas necesarias para el funcionamiento del controlador, tanto a nivel del manejo de todos los dispositivos del hardware, como de las rutinas relacionadas con la operación del manipulador.

En el capítulo siete se listan las pruebas realizadas. Primero se prueba la arquitectura por medio de un programa de prueba que contiene las rutinas especificas del manejo del hardware, después se validan las rutinas propias del manejo del manipulador al implementarlas en "C" y realizar una

simulación de su operación; por último, se prueba el algoritmo de control por medio del manipulador de dos GDL del "Laboratorio de Control" de la DEPFI.
Al realizar las pruebas por separado se tienen resultados de cada parte, que permiten sacar conclusiones sobre comportamiento.

El manipulador de dos GDL, si bien no es una estructura tan compleja como uno de seis GDL, permite una validación bastante buena del control, pues se trata de un sistema ya implementado que cumple con tener uniones de giro, actuadas por motores de corriente directa, y cuenta con todo lo necesario para documentar las pruebas que se realicen sobre él. Además existen varios controles ya implementados para el manipulador, por lo que se puede establecer una comparación entre algoritmos.

Finalmente, en el capítulo ocho se exponen las conclusiones a las que se llegó en el trabajo realizado, y se enuncian además los posibles desarrollos a futuro.

CAPÍTULO 2

MANIPULADORES. CARACTERÍSTICAS PRINCIPALES

Un manipulador es un sistema mecánico formado por una secuencia de cuerpos rígidos[1], llamados elementos, conectados usualmente en serie por medio de uniones. Los elementos se conectan a los más con otros dos, de forma que no se formen lazos cerrados y al conectarse en serie forman una cadena cinemática abierta; ya que en un extremo de la cadena un elemento se encuentra fijo a una base o soporte, mientras que el otro extremo se encuentra libre y es en este extremo donde se encuentra la herramienta o actuador final, que produce las tareas que realiza el manipulador.

Las cadenas cinemáticas se dividen en dos tipos: las activas y las pasivas; siendo los manipuladores cadenas cinemáticas activas, en donde cada elemento tiene su fuente individual de alimentación (actuador); en contraparte de las cadenas pasivas, en donde todos los elementos reciben el movimiento de un sólo actuador.

---

[1] En forma general, los elementos y las uniones de un manipulador son en cierto grado flexibles; pero se va a tomar el caso ideal de elementos y uniones perfectamente rígidos.

En gran parte de la bibliografía, los términos manipulador y robot son muy afines y si se utiliza la definición del "Robot Institute of America"

"...un robot es un manipulador multifuncional, reprogramable, diseñado para mover materiales, piezas, herramientas o dispositivos especializados a través de movimientos variables, programados para la realización de una variedad de tareas...". Entonces se tiene que el término robot es mucho más amplio y de hecho un manipulador es un subsistema de un robot.

De forma funcional, un robot está compuesto por varias partes:

-Sistema mecánico, (elementos mecánicos y uniones con capacidad de movimientos).

-Sistema de potencia, (asociado con cada unión se tiene un actuador/transmisión de potencia, que es el causante del movimiento de ésta).

-Sistema de control, (comúnmente se utilizan controladores digitales, que van de configuraciones generales, a diseños específicos o arreglos de proceso en paralelo. Además se requiere de sensores internos, los asociados a cada actuador, para obtener la posición y posiblemente la velocidad a nivel de cada unión).

En relación al término robot, un manipulador comprende al sistema mecánico, al sistema de potencia y a los sensores internos de cada actuador. Y se restringe únicamente a las máquinas antropomorfas, que asemejan el brazo y mano humanas, en contraparte de los robots que pueden moverse dentro de su ambiente. Se debe de distinguir de los manipuladores industriales, aquellos en los que hay una interfase hombre-máquina para el control, como en los utilizados para manejar sustancias radioactivas; a esos manipuladores se les llama manipuladores teleoperados o telerobots y aunque se podrían mover autónomamente, no están diseñados para esto.

2.1 Parámetros básicos de un manipulador.

Algunos de los parámetros representativos de un manipulador son:

- Movilidad
- Configuración cinemática
- Espacio de trabajo
- Exactitud y repetibilidad de posicionamiento
- Agilidad (velocidad efectiva de ejecución de movimientos prescritos)
- Velocidades de las uniones
- Rango de movimiento de las uniones
- Tipo de actuadores y sensores
- Requerimientos de consumo de potencia
- Capacidad de carga
- Peso
- Coeficientes de amortiguamiento y frecuencias naturales.
- Rigidez estructural
- Economía (costo, confiabilidad, etc)

Muchos de los parámetros anteriores están interrelacionados; por ejemplo la carga que puede aceptar un manipulador depende del momento de inercia de ésta, que tan lejos de la base se realice el movimiento, del tipo de actuadores utilizados, de la velocidad y aceleración, etc.

-Movilidad:

La movilidad de un manipulador, está determinada por el número de grados de libertad (GDL)

del sistema mecánico, aunque algunas veces el número de uniones es utilizado en lugar de los grados de libertad.

Los grados de libertad, es un término que se utiliza para hacer referencia al número de variables que se tiene que utilizar para describir un sistema mecánico, las variables forman un conjunto de coordenadas y cuando estas son suficientes para describir al sistema, y son independientes entre si, se les llama coordenadas generalizadas. Por lo general el número de coordenadas generalizadas es igual al número de grados de libertad. En un manipulador, cada par unión_elemento constituye un GDL[2] y en la mayoría de los manipuladores industriales se tienen de cuatro a seis GDL.

En los manipuladores de cadena cinemática abierta, es usual tener dos tipos de uniones[3]: las de revolución(R) y las prismáticas(P).

Las uniones de revolución, son aquellas que permiten el movimiento angular entre los dos elementos que comparten la unión.

Las uniones prismáticas, son aquellas que permiten un movimiento de traslación rectilíneo entre los elementos.

Cada unión al tener su actuador, realiza un movimiento que no depende del movimiento de otras uniones; con lo que los elementos se mueven de forma independiente.

---

[2] En el caso de manipuladores de cadena cerrada, como los de estructura de cinco barras, se tienen por lo general más uniones para un cierto número de GDL que los de cadena abierta, así como un espacio de trabajo más reducido.

[3] En forma general existen seis tipos de uniones:
planar, de revolución, cilíndrica, prismática, esférica y de tornillo.

Los movimientos de un manipulador se pueden dividir en:

-Globales[4]: involucran movimientos a distancias mayores que las dimensiones del manipulador.

-Regionales: involucran movimientos para posicionar el actuador final (último elemento de la cadena), en un punto del espacio de trabajo.

-Locales: involucran movimientos del actuador final (orientación).

De acuerdo a esto, el manipulador se divide en un sistema mayor, que produce los movimientos regionales y un sistema menor que produce los movimientos locales, se les puede llamar brazo y muñeca respectivamente.

-Configuración cinemática:

El brazo para poder moverse en un espacio tridimensional, debe de tener por lo menos 3 uniones ya sean tipo R o P, lo que implica que pueden darse varias combinaciones de uniones en un brazo en específico, y aun dentro de una misma combinación se pueden tener diversas implementaciones, debido a diferentes posiciones relativas de las uniones. Son estos arreglos específicos, lo que constituye la configuración cinemática de un manipulador y las configuraciones más utilizadas, se verán en la parte de clasificación de manipuladores (sección 2.4).

Aunque la configuración cinemática está especificada por el arreglo particular de las uniones del brazo, en la muñeca también existen varias opciones cinemáticas. Los movimientos de la muñeca

[4] Los movimientos globales son los producidos por algún sistema de transporte, ya sea el movimiento en un riel, por ruedas, etc. Este tipo de movimiento está fuera de los alcances de la tesis.

producen la orientación del actuador final y normalmente sólo se utilizan uniones de revolución, las cuales pueden ser de dos tipos: la unión de giro y la de plegado. La unión de giro es la que une a dos elementos colineales y cuyo eje de rotación, también es colineal; por medio de este tipo de unión se tiene un movimiento sin restricciones. Una unión de plegado es la que tiene su eje de rotación, perpendicular al eje de los elementos que une y si no hay un defasamiento entre elementos, se tiene un movimiento restringido por los mismos elementos.

En las muñecas simples, (en las cuales todos los ejes de las uniones que la forman, o bien son perpendiculares entre si o son paralelos) se tienen configuraciones de uno a tres GDL y para el caso de tres GDL (con el que se puede lograr una orientación arbitraria), generalmente hay cuatro opciones diferentes (figura 2.1), de acuerdo a los diferentes tipos de uniones de revolución que hay y dado que en la última unión es común encontrar una unión de giro, para proporcionar un movimiento angular sin restricciones al actuador final o herramienta.

De la figura 2.1: la opción (PPG) "plegado/plegado/giro", requiere que los actuadores estén conectados casi en forma directa a los elementos, por lo que es utilizada generalmente con actuadores hidráulicos. La opción (GPG), tiene la posibilidad de tener a los actuadores en el antebrazo y puede ser bastante compacta.

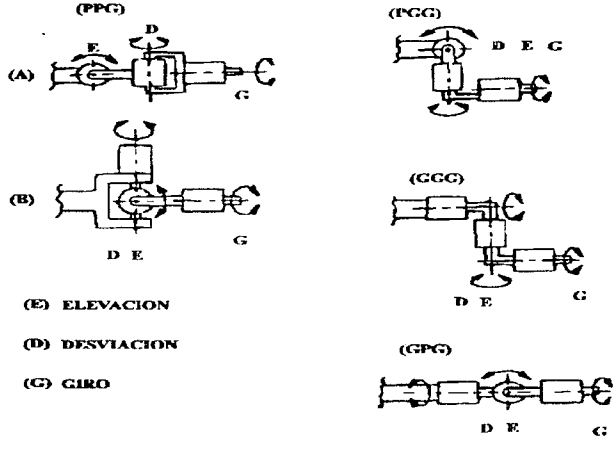


Fig. 2.1 Arreglos de una muñeca[5] de 3 GDL.

La opción (GGG), se le llama de triple giro y es la opción de muñeca simple más versátil, da un sistema muy compacto y extremadamente maniobrable. Hace que aumente la capacidad de carga del manipulador, la velocidad de movimientos y la accesibilidad a áreas restringidas. Por lo general el eje de la segunda unión es oblicuo a los de las otras dos uniones, para minimizar el espacio requerido por los mecanismo.

[5] Los términos elevación, desviación y giro, se refieren a los diferentes movimientos de orientación que puede realizar una muñeca de tres GDL. Posiblemente en donde se aprecia mejor a que movimientos de orientación se está refiriendo cada término, es en la opción "PPG".

-Espacio de trabajo:

Se llama espacio de trabajo a los puntos que pueden ser alcanzados por el manipulador, ya sea por la muñeca o por el punto final del brazo, mientras que los puntos que no se les puede alcanzar se les llama espacio muerto. La superficie de contorno del espacio de trabajo, está formada por los movimientos máximos y mínimos del manipulador.

El espacio de trabajo se divide en:

-El espacio alcanzable, que es el conjunto de todos los puntos alcanzables por el manipulador, al realizar el total de los movimientos posibles.

-El espacio "hábil", es el que tiene a todos los puntos alcanzables por el manipulador con orientación arbitraria; siendo un subconjunto del espacio alcanzable, ya que muchas veces la interferencia entre componentes de la muñeca, hace que no para todos los puntos del espacio de trabajo se tenga una orientación arbitraria.

Algunas veces se reduce el espacio de trabajo para excluir zonas en donde los parámetros básicos del manipulador son diferentes del comportamiento general estimado.

-Exactitud y repetibilidad:

Repetibilidad es que tan cerca, el manipulador, puede volver a una posición dada al repetirse un mismo conjunto de desplazamientos en sus uniones. La repetibilidad es afectada por la resolución del controlador, tomando esto como el cambio de posición más pequeño que el controlador puede detectar.

Exactitud se puede definir como la distancia entre las coordenadas programadas y el promedio de las coordenadas reales, en donde se posiciona el manipulador. Actualmente los robots son

altamente repetitivos, pero no muy exactos; esto por que en lo general no se sensa la posición del actuador final, si no el desplazamiento en cada unión y por un modelo de la geometría del manipulador se llega a la posición del actuador final; por lo que, la exactitud, depende de varios factores que pueden ser clasificados en:

-Ambientales, (tales como temperatura, humedad y ruido eléctrico).

-Paramétricos.

-De medición.

-Computacionales.

-De aplicación.

Cada categoría de inexactitudes puede no ser independiente de las otras, por ejemplo la temperatura puede afectar los parámetros cinemáticos del manipulador, así como la fricción y el comportamiento de los dispositivos electrónicos utilizados en el controlador.

Factores paramétricos:

-Parámetros cinemáticos: Los parámetros de esta clase más notables son las longitudes de los elementos; pueden variar por factores ambientales, por errores en el ensamblaje del manipulador y por las tolerancias en los maquinados. Otros parámetros de esta clase son:

-La posición de cero para las uniones.

-La transmisión de potencia, incluyendo a las no uniformidades de bandas, cadenas, engranes y levas en su rango de movimiento.

Mientras que todos los parámetros cinemáticos, contribuyen a un error en la exactitud, generalmente no afecta a la repetibilidad.

-Parámetros dinámicos: Afectan la exactitud en la trayectoria del manipulador. Siendo en cierto grado flexibles, los manipuladores se flexionan bajo la acción de la carga y bajo la acción de su propio peso; por lo que el actuador final se desviará de su posición esperada una distancia proporcional a la suma de los factores estáticos y dinámicos. Hay varios componentes que contribuyen a esta flexibilidad, como son los rodamientos, los componentes de la transmisión y la estructura de los propios elementos. Cuando un manipulador está en operación bajo movimientos previamente programados, al posicionarse en los diferentes puntos programados, sigue un comportamiento como el ilustrado en la figura 2.2

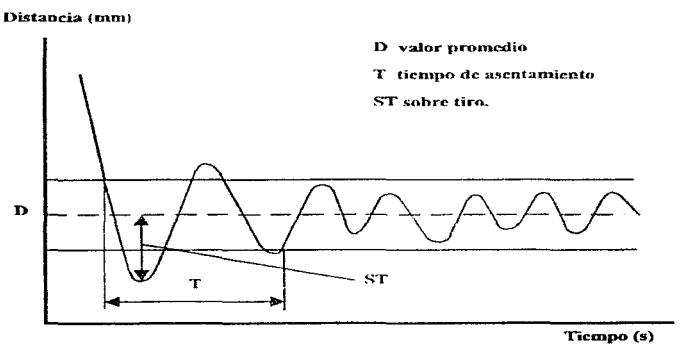


Fig. 2.2 Comportamiento de un manipulador.

Factores de medición:

Los encodificadores montados directamente en el eje del motor, hacen fácil el control del motor, ya que se mide exactamente el movimiento de éste; sin embargo la flexibilidad, los deslizamientos y otros factores paramétricos que afectan la exactitud del manipulador, no son retroalimentados

al control y por tanto la posición del actuador final no es controlada de forma exacta. A parte si hubiera un pequeño error en la medición del encodificador, este se amplifica a través de la longitud de cada elemento.

Factores de computación:

El cálculo de la trayectoria del manipulador contiene varios tipos de errores, como la representación de números reales por medio de un número finito de bits (redondeo) y que muchas veces en ciertas zonas se compromete la exactitud, para obtener movimientos estables.

Factores de aplicación:

Los factores de este tipo incluyen a los errores de instalación, de las piezas sobre las que el manipulador tiene que trabajar y las inexactitudes en el actuador final.

Cuando se programa al robot en línea, es decir manualmente, al llevar al manipulador hasta el punto de trabajo y grabar las posiciones necesarias, la única exigencia sobre el manipulador es que mantenga una repetibilidad elevada, en cada uno de los puntos programados. Cuando se programa fuera de línea, (por medio de una simulación) se utiliza un modelo del manipulador, que se aleja del comportamiento real por errores en los parámetros geométricos y en las variables de unión. En consecuencia los movimientos programados fuera de línea y simulados en una computadora, son ideales en cuanto a que los produjo un modelo que no tomó en cuenta muchos aspectos del manipulador real y difieren de los producidos por el manipulador, bajo el programa generado. En la programación fuera de línea se tienen requerimientos altos para la exactitud y la repetibilidad.

También hay problemas de exactitud cuando en una misma posición se tienen diferentes orientaciones programadas, ya que al producirse diferentes desplazamientos en las uniones, los defectos de fabricación de los elementos (desviaciones de las longitudes nominales) hace que se tengan errores diferentes para el mismo punto en el espacio de trabajo, según sea la orientación requerida. Por esto si se requiere cambiar la orientación para un punto y esto se hace fuera de línea lo más probable es que se necesite reprogramar el punto.

-Velocidad efectiva:

En la mayoría de los manipuladores, se especifica la velocidad máxima de las uniones, pero en realidad se llega muy pocas veces a alcanzar esta velocidad, por lo que es más indicativo la velocidad efectiva: que está dada por $Vef = L/(t_m + t_s)$;donde: L es la distancia recorrida, t_m es el tiempo nominal de recorrido (tiempo de aceleración + tiempo de desaceleración + tiempo de velocidad continua) y t_s es el tiempo de asentamiento de oscilaciones, al final del viaje.
La velocidad efectiva es mucho menor que la velocidad máxima especificada para las uniones; ya que se tienen períodos de aceleración y desaceleración. Además la velocidad efectiva no es constante dentro del espacio de trabajo, ya que la capacidad de aceleración y desaceleración varia en diversas zonas de éste.

-Tipo de actuadores y sensores:

-Actuadores:

Cada unión para poder realizar sus movimientos tiene un actuador/transmisión de potencia, que normalmente puede ser de tres tipos: actuadores eléctricos, hidráulicos o neumáticos. En la

selección de los actuadores, el parámetro crítico es la relación fuerza/peso, para las uniones prismáticas y para las uniones de revolución es el de par/peso. Por lo que, al escoger un actuador para una unión, se debe de determinar los requerimientos tanto de par(fuerza) máximo, como los de velocidad.

En el caso de actuadores hidráulicos, se tiene la relación par (fuerza)/peso más elevada por lo que se puede utilizar directamente a nivel de las uniones, con el único problema de la transmisión del fluido a presión, que a nivel de las uniones de la muñeca puede ser bastante complejo. Los actuadores hidráulicos fueron la opción principal para los primeros manipuladores y aún lo son para los de uso pesado; pero tienden a implicar costos mayores que los otros tipos de actuadores.

Los actuadores neumáticos son relativamente la opción de menor costo, pero su relación fuerza/peso es baja y por lo general implican una rigidez muy baja en el funcionamiento de la unión. Se aplican en manipuladores simples con movimientos controlados por topes mecánicos y en los sistemas de prensión de los actuadores finales; aunque se pueden tener ciertos arreglos para disminuir los efectos de la compresibilidad del aire y para poder controlar su posición de manera continua.

Los actuadores eléctricos, (es decir motores eléctricos) predominan sobre los otros tipos de actuadores en la generalidad de los casos, porque no requieren de mucho equipo adicional, son "limpios" (no hay problemas de fugas, como en el caso hidráulico o neumático), requieren de un mantenimiento moderado, son pequeños, etc.

En los manipuladores con actuadores eléctricos se tienen demandas grandes sobre los motores, como por ejemplo la necesidad de maximizar la razón potencia/peso y el requerimiento que el motor, en especial para aplicaciones en el actuador final, sea lo más pequeño posible. También se debe de conservar cierta exactitud/repetibilidad, dentro del espacio de trabajo, mientras se logran velocidades altas con una capacidad de carga aceptable. Se utilizan varias clases de motores eléctricos: motores de CD de imán permanente, motores de CA sin escobillas, motores de pasos, etc.

Un tipo de motor de CD de imán permanente, que tiene varias ventajas es el que tienen un imán de alto rendimiento; con este tipo de motor se puede tener un alto par, con un peso bajo ya que el campo magnético más intenso no implica un tamaño mayor del motor e inclusive se puede reducir el número de devanados en el rotor por lo que se requiere de una corriente menor durante su operación. Una mayor relación par/peso, implica que se puede operar el motor a velocidades menores con una relación de transmisión de potencia más baja.

Otro motor utilizado como actuador es el servomotor de CA sin escobillas. En construcción es muy similar a un motor síncrono trifásico, con los polos magnéticos en el rotor y en el estator la armadura con los devanados arreglados en tres fases. Se requiere de retroalimentar la posición para poder operar a velocidad variable; de lo contrario sólo se puede operar a la velocidad síncrona. Típicamente un motor de CA sin escobillas, es más pequeño que uno de CD de imán permanente y tiene una mayor disipación térmica por tener los devanados en el estator. Otras ventajas son que no hay escobillas que se desgasten, no hay chispas en el conmutador y hay menos fricción; pero requieren de más dispositivos para su uso, ya que se debe de controlar un voltaje trifásico.

Los motores de pasos tienen ciertas características que limitan su uso:

En ciertos instantes se tiene una velocidad negativa con respecto al promedio de un movimiento dado, por lo que hay una gran vibración.

Conforme aumenta la frecuencia de la señal de entrada, se llega a un momento en que el siguiente pulso llega antes que se de la inversión de la dirección momentánea y se tienen errores en el posicionamiento, ya que al detenerse los pulsos, se puede tener suficiente inercia para pasarse hasta el siguiente estado estable.

Por otro lado el poder controlarlos en lazo abierto, los hace atractivos pero, aun así son más comunes los manipuladores que utilizan motores de CD o motores de CA sin escobillas que los que utilizan motores de pasos.

En la mayoría de los manipuladores, los actuadores eléctricos están acoplados a una transmisión de potencia; pero hay algunos con los actuadores conectados en forma directa a los elementos (*direct drive*), de esta manera se eliminan los problemas creados por la transmisión de potencia (fricción, deslizamiento, etc) y mejora la respuesta dinámica al no tener una inercia reflejada al lado del motor, por parte de la transmisión de potencia. El *direct drive* se ha aplicado en manipuladores de alta velocidad, para ensamblado y manejo de materiales como el tipo SCARA (ver clasificación en la sección 2.4) ya que sus uniones no tienen que soportar cargas muy elevadas.

La ausencia de una transmisión de potencia, trae otros efectos asociados como por ejemplo, una rigidez no muy elevada a nivel de las uniones y el acoplo dinámico entre los elementos, que impone al control un problema no lineal y que se tenga que considerar a todos los elementos, en el control individual de las uniones.

-Sensores:

Se pueden dividir en internos, los que miden las variables de las uniones (posición, velocidad, par, etc) y externos, los que interactúan con el ambiente.

-Sensores internos:

-Posición/velocidad:

Una forma común de medir la posición angular de las uniones de revolución, es por medio de potenciómetros, de forma que el desplazamiento angular es proporcional a un voltaje. La velocidad puede ser determinada ya sea por diferenciación de la señal del potenciómetro o por medio de un tacómetro.

Otra forma de medir la posición es por medio de un encodificador óptico de barra. En este transductor se convierte una posición angular en un código binario por medios ópticos, hay dos tipos básicos: los absolutos (en este tipo se codifica la posición de la barra, utilizando un cierto número de bits, por lo que cada posición tiene asignada un código absoluto, que puede ser binario o Gray) y los incrementales (estos sensores requieren de un poco más de dispositivos de interfase y de algo de software para poder obtener la posición, en construcción son similares a los absolutos pero, en lugar de asignar un código binario a cada posición, da un pulso por cada cierto desplazamiento angular, por lo que se tiene una cuenta de pulsos por una vuelta completa, que depende de la resolución del sensor).

Para obtener la velocidad, partiendo de encodificadores ópticos se puede contar cuantos cambios de posición ocurren en un determinado tiempo o medir cuanto tiempo dura un desplazamiento; las dos técnicas tienen ventajas y desventajas.

-Fuerza/par:

Otros sensores internos, son los que miden la fuerza/par ya sea a nivel de las uniones o del actuador final; son necesarios para las aplicaciones en las que hay un contacto entre el ambiente y el manipulador. Uno de los sensores utilizado, son las galgas extensiométricas, las cuales cambian su resistencia eléctrica, ante un esfuerzo de tipo mecánico; este tipo de sensores se puede utilizar para sensar el par, en cada unión o en un arreglo de varios de estos sensores, para detectar la fuerza y par sobre la muñeca del manipulador.

-Sensores externos:

Son sistemas específicos a la tarea que se va a realizar pero, generalmente son sistemas de visión y/o sensores de proximidad.

Los sistemas de visión permiten al manipulador responder a cambios en su ambiente, de una forma flexible y se le considera uno de los sistemas de percepción más potentes. Aunque en muchos casos los sistemas de visión no pueden dar toda la información necesaria, ya que no tienen mucha exactitud, mientras mantienen un campo de visión constante y por que a veces hay partes ocultas en la imagen a interpretar. Para esto se utilizan sensores a nivel del actuador final que dan una percepción de la textura y forma de los objetos con los que se está en contacto; así como información sobre las fuerzas y momentos aplicados.

Por medio de sensores de proximidad se puede saber la presencia o ausencia de ciertos objetos, con lo que se puede evitar o conseguir el contacto con ellos; hay varios tipos, como los detectores ópticos de proximidad y los sensores ultrasónicos. Los sistemas de sensores externos, son muy importantes para la planeación de trayectorias y de tareas e influyen en el control en una forma supervisora; no están directamente en el lazo de control a nivel de las uniones.

2.2 Degeneración.

La degeneración es el fenómeno que se presenta en varios puntos del espacio de trabajo, en los que se tiene valores indeterminados en el desplazamiento de algunas uniones y por tanto no afectan a la posición del actuador final, con lo que se dice que se pierden grados de libertad.

Por ejemplo en las muñecas esféricas ocurre una degeneración, cuando los tres ejes de las uniones son paralelos a un plano y para todas las configuraciones de muñecas simples, la degeneración está asociada con la segunda unión de la muñeca (quinta unión en un manipulador de 6 GDL). El que se tenga una degeneración, no implica que no se pueda lograr una determinada orientación, si no que a veces no se puede llegar a ella por el camino más directo y se tiene que programar otras secuencias de movimiento. Cerca de las zonas de degeneración se tiene ciertos efectos en la velocidad del actuador final y se debe de aumentar la velocidad de la primera y tercera unión de la muñeca para lograr la misma velocidad de orientación, que en zonas fuera de los puntos de degeneración.

Lo importante de las zonas singulares o de degeneración es que:

1) En esas configuraciones, ciertas direcciones del movimiento son imposibles.

2) En puntos de configuraciones singulares, para una velocidad acotada del actuador final, puede corresponder a una velocidad no acotada de algunas uniones.

3) Para fuerzas y pares acotados en el actuador final, puede corresponder a fuerzas y pares no acotados en las uniones.

4) Usualmente corresponde a puntos en la frontera del espacio de trabajo.

En el análisis de un manipulador articulado (capítulo 3), se va a tratar un poco más las degeneraciones propias de esa clase de manipuladores y como influyen en su comportamiento.

2.3 Actuadores finales.

En el extremo del manipulador que permanece libre, después de la muñeca, es donde se encuentra el actuador final. Los actuadores finales que sirven para manejar objetos (i.e. para levantar objetos, tomarlos con seguridad mientras se les mueve y ponerlos en algún sitio en la posición correcta) se les llama "*grippers*" o pinzas; y los más simples son las pinzas que cierran o abren únicamente. Hay actuadores finales más complejos, como los que simulan una mano o herramientas que realizan cierta operación: como pintado, soldado, remoción de material, etc.

Las pinzas diseñadas para uso general son útiles, si la aplicación es simple pero, si se necesita de algún manejo complicado es inevitable el uso de pinzas de diseño específico; siempre cuidando de realizar todas la funciones encomendadas con la pinza de peso mínimo, ya que el menor peso en la pinza implica una capacidad de carga mayor.

De acuerdo a las demandas de las aplicaciones en específico, se han desarrollado una gran cantidad de pinzas, como:

-Pinzas de dos dedos, moviéndose en paralelo o en torno a un eje común.

-Ventosas, los cuales utilizan equipo neumático para succionar la pieza y poderla manipular; sobre todo con piezas relativamente ligeras con superficies grandes y suaves.

-Pinzas magnéticas.

-Pinzas de múltiples dedos, se utilizan en aplicaciones en donde se requiere de una manipulación compleja.

-etc.

Normalmente se utilizan actuadores neumáticos para las pinzas, aunque los actuadores eléctricos tienen ventajas cuando los elementos de sujeción deben de realizar sus movimientos con gran coordinación y exactitud de posicionamiento, sin necesidad de mucho equipo adicional. Si se requiere una potencia de prensión grande se necesita de actuadores hidráulicos, pero la conducción del flujo de presión desde el manipulador hasta el actuador final es bastante costoso y complicado.

Con base a requerimientos cada vez mayores, es que los actuadores finales han evolucionado, hasta llegar a ser dispositivos tan o más complejos que el propio manipulador, como en el caso de las manos mecánicas:

La mano mecánica Utah/MIT (figura 2.3), cuenta con cuatro dedos, cada uno de ellos con cuatro uniones; lo que hace a la mano un dispositivo de 16 GDL. Cada unión es controlada antagonisticamente con dos actuadores electroneumáticos por medio de un tendón de alta resistencia. La posición de cada unión de los dedos, es medida por sensores de efecto Hall.

La mano originalmente llamada Stanford/JPL (figura 2.4) es un dispositivo de tres dedos, cada uno con 3 GDL. Los dedos se mueven por medio de cables flexibles de acero, recubiertos de teflón dentro de conductos flexibles; la tensión de cada cable está dada por un motor de CD. Los conductos flexibles permiten el paso de los cables a través de la muñeca, por lo que los actuadores (doce motores) están en el antebrazo.

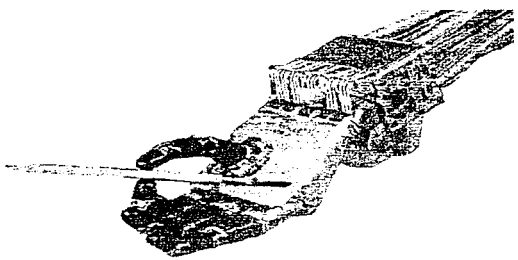


Fig. 2.3 Mano Utah/MIT

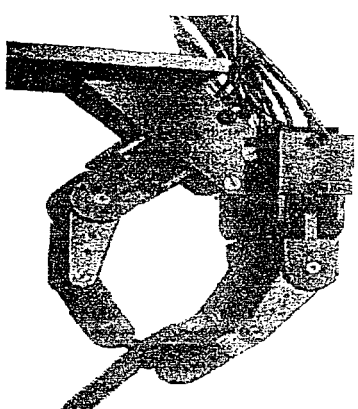


Fig. 2.4 Mano Stanford/JPL (Salsbury).

A pesar de la disponibilidad de sofisticadas manos mecánicas, estas no han sido aplicadas afuera de laboratorios. Una razón es que la sofisticación mecánica requiere de una igual sofisticación en el control, sensores y componentes de detección pero, pueden constituir una opción ante las pinzas de diseño en específico, ya que las manos mecánicas tan sólo se tienen que reconfigurar en software, en contraparte de las pinzas especiales que se tienen que reemplazar ante modificaciones de las piezas que se manipulan.

2.4 Clasificación.

Aunque un manipulador es un mecanismo de propósito general, se diseña pensando en cierta área de aplicación, y ésta dicta muchas de las características cinemáticas y dinámicas del manipulador; es por esto que hay un gran número de manipuladores con características diferentes y con ciertas ventajas y desventajas.

Los manipuladores se pueden clasificar de acuerdo a muchos criterios, como su geometría, el tipo de aplicación para la que fueron diseñados, la manera en la que son controlados, el tipo de actuadores que utiliza, etc. Normalmente los manipuladores industriales tienen seis o menos grados de libertad y se clasifican de acuerdo a las tres primeras uniones, es decir sin tomar en cuenta a las uniones de orientación.

En la figura 2.5, se muestran algunas de las posibles combinaciones de uniones P y R. Aunque cada combinación tiene un espacio de trabajo típico, este se puede modificar al cambiar las proporciones de los elementos e introduciendo offsets en la localización de las uniones.

Del total de configuraciones cinemáticas, por lo general en los manipuladores industriales sólo se utilizan cuatro o cinco configuraciones distintas.

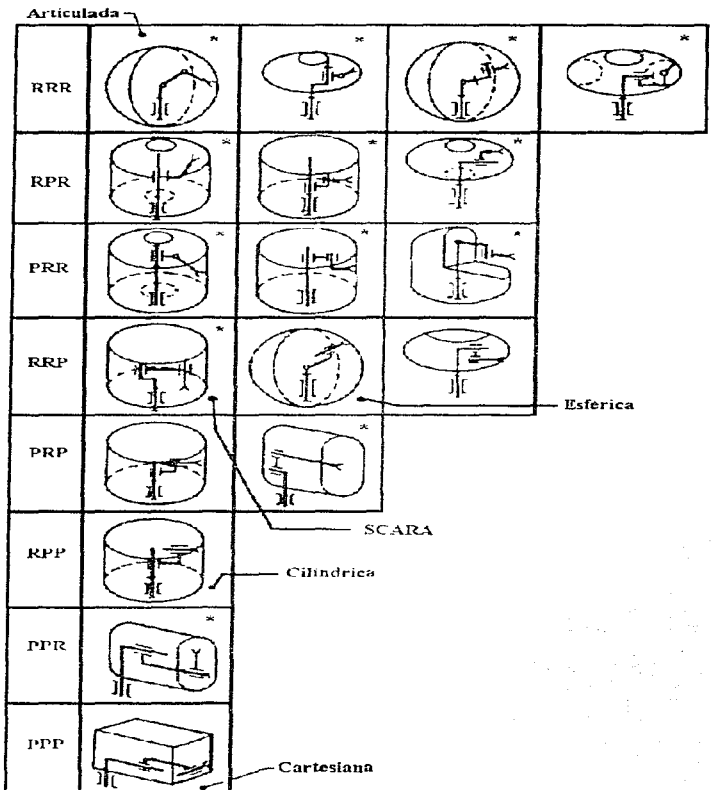


Fig. 2.5 Configuraciones cinemáticas.

Características principales de las estructuras más utilizadas:

-Cartesianos (PPP) o rectangulares.

Las tres uniones de posición son prismáticas, los movimientos pueden denominarse de viaje, elevación y alcance (x,y,z). Tienen la descripción cinemática más sencilla, ya que los movimientos de sus uniones son totalmente independientes entre si; lo que hace que sean los más fáciles de controlar. Sirve para ensamblajes en superficies planas (mesa), para transferencias de material y carga. Por lo general son demasiado grandes ya que necesitan de una estructura para las guías, lo que limita el espacio de trabajo y tienen una alta inercia. La exactitud es buena, aunque si el recorrido es grande, esta se puede ver afectada por deflexiones en las guías.

-Cilíndricos (RPP).

El espacio de trabajo es la intersección de dos cilindros con ejes comunes. Se utiliza para tareas que requieren accesos a áreas pequeñas, los movimientos son: alcance, rotación y elevación (r, theta, z). La mayoría utiliza hasta dos grados de libertad en la muñeca, ya que no necesita corregir la elevación, debido a que la orientación vertical no cambia; aunque se podría utilizar una muñeca esférica, para que el movimiento de elevación, lograra que el actuador final alcanzara puntos por debajo de su base.

-Esféricos (RRP).

El espacio de trabajo es una sección de una esfera y las variables de posición de las tres uniones corresponden directamente a las coordenadas esféricas del actuador final (r,theta,gama).Estos manipuladores son complicados, requieren de un control complejo y son grandes con respecto a su espacio de trabajo.

-**SCARA** (*Selective Compliant Articulated Robot, for Assembly*).

Aunque es un RRP difiere mucho del manipulador esférico. Los ejes de las tres primeras uniones son paralelos y verticales y se tiene un rango limitado de movimientos. Se desarrolló en Japón, para ensamblaje de circuitos impresos; básicamente tiene una cobertura en dos dimensiones, si no se utiliza el tercer GDL, que es una unión prismática.

-**Articulados** (RRR) o de revolución.

El que todas las uniones sean de revolución, trae varias ventajas como que los rodamientos rotatorios son más fáciles de sellar, tienen poca inercia y pocas pérdidas por fricción; esto en comparación con los rodamientos requeridos en los uniones prismáticas. Por su misma estructura, estos manipuladores, son pequeños en comparación con el espacio de trabajo que cubren, tienen una gran capacidad de realizar tareas en áreas pequeñas, pueden maniobrar entre obstáculos mejor y tienen un rango más amplio de aplicaciones.

Pese a las ventajas que tienen, existen algunas desventajas:

-La degeneración del comportamiento, cerca de las fronteras del espacio de trabajo.

-Aunque la configuración tipo codo, logra un espacio de trabajo grande, el precio es una rigidez baja, para prácticamente cualquier dirección del actuador final.

-Se tienen problemas por el acoplamiento dinámico y estático de los elementos, creando problemas de control más complejos.

-El tener los motores cerca de las uniones, hace que la dinámica de la estructura se vea modificada por el peso de los motores y de la transmisión de potencia.

En esta configuración cinemática, hay dos tipos principales de implementación: los manipuladores articulados tipo codo, figura 2.6 y figura 2.8. En el cual los ejes de las uniones dos y tres son paralelos y perpendiculares al eje de la unión número uno, esta clase de manipuladores tiene una gran libertad de movimiento en un pequeño espacio. En la figura 2.7 se muestra el espacio de trabajo.

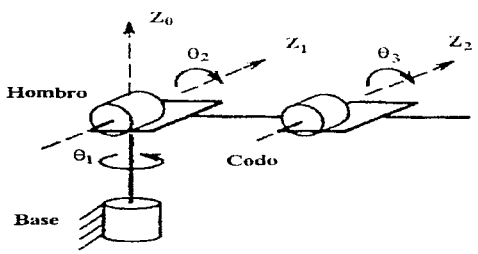


Fig. 2.6 Estructura de un manipulador tipo codo.

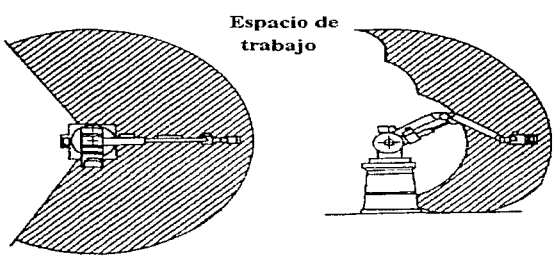


Fig. 2.7 Espacio de trabajo de un manipulador tipo codo.

Aunque comparten una misma configuración cinemática, los manipuladores articulados tipo codo pueden ser muy diferentes, como en las figuras 2.8, 2.9 y 2.10

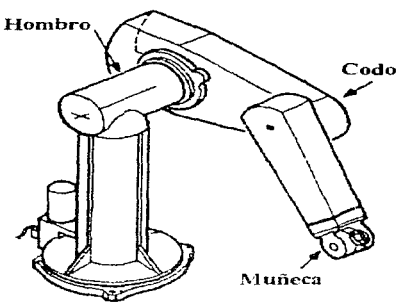


Fig. 2.8 Manipulador PUMA, Unimate.

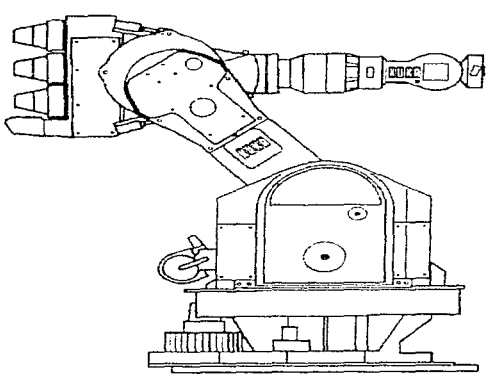


Fig. 2.9 Manipulador IR160/60 (KUKA).

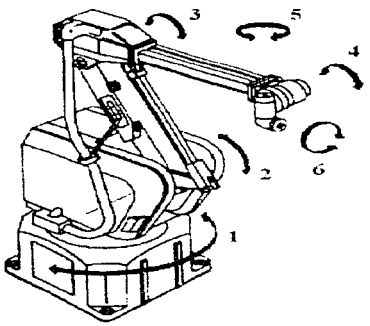


Fig. 2.10 Manipulador P-100 (ABB)

El segundo tipo es el denominado "de paralelogramo" (figura 2.11), aunque menos hábil que el de tipo codo, tiene muchas ventajas que lo hacen atractivo: una de ellas es que el actuador de la unión tres se encuentra a nivel de la unión uno, por lo que las uniones dos y tres pueden hacerse más ligeras y con motores menos poderosos. A parte la dinámica es más sencilla que en el tipo codo, por lo que es más fácil de controlar.

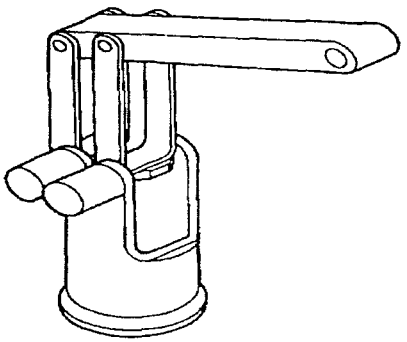


Fig. 2.11 Estructura de paralelogramo.

-Estructuras redundantes:

Un manipulador redundante es el que tiene más grados de libertad que los requeridos para una tarea dada, por ejemplo para realizar movimientos en dos dimensiones, se requiere de por lo menos dos grados de libertad, lo que hace a un manipulador con tres GDL redundante para esa tarea.

El que un manipulador sea redundante implica que pueden haber diferentes desplazamientos, a nivel de las uniones, para alcanzar el mismo punto; esto puede ayudar en la operación ya que ofrece muchas más opciones para alcanzar un punto dado y ayuda a resolver la degeneración pero, trae consigo el incremento en la complejidad del control y del sistema mecánico, aparte que reduce la rigidez de la estructura. En muchos casos se tienen estructuras comunes con algunos grados de libertad aumentados, como por ejemplo el que una estructura cartesiana se le agregue una primera unión rotativa (RPPP), o que se le añada una base de traslación.

Un caso especial es cuando se tienen siete GDL para el posicionamiento y orientación, esta configuración ofrece muchas ventajas y si todas las uniones son de revolución, se denomina configuración antropomórfica como en la figura 2.12

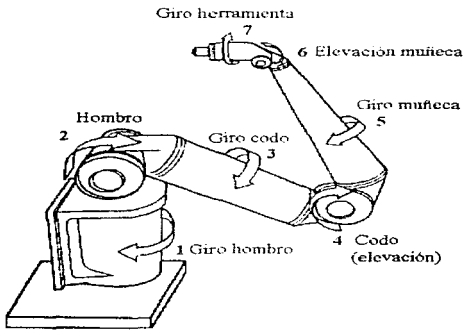


Fig. 2.12 Brazo antropomórfico.

Aparte del caso de tener más GDL que los necesarios, algunos manipuladores tienen cierta redundancia; en específico las configuraciones con "*", en la figura 2.5. En los referente a los manipuladores articulados se tienen varias redundancias, las cuales se verán en el análisis cinemático (capítulo 3).

CAPÍTULO 3

ANÁLISIS DE UN MANIPULADOR ARTICULADO

Antes de analizar una estructura cinemática (un manipulador), se debe de adoptar una convención en cuanto a nomenclatura y marcos de referencia; para esto se utiliza una representación de la geometría del manipulador, que por un lado asigne un marco de referencia a cada elemento, para determinar el estado global de la estructura cinemática, partiendo del comportamiento de cada elemento. Y que por el otro lado contenga una notación sistemática, que simplifique el planteamiento de las ecuaciones que representan el comportamiento de un manipulador de seis GDL; dado que son muchas las variables involucradas.

Hay varias representaciones que buscan hacer sistemática la descripción geométrica de un manipulador pero, posiblemente la representación más usada es la de Denavit_Hartenberg que es un método matricial para establecer de forma sistemática, un marco de referencia para cada elemento de una cadena articulada. En esta representación, un manipulador es un conjunto de elementos rígidos conectados por medio de uniones simples, de un grado de libertad, ya sean de revolución o prismáticas. Las uniones simples, al tener un sólo grado de libertad se representan

por una sola variable (q_i), por ejemplo una unión de revolución se representa por su desplazamiento angular.

El marco de referencia asociado a cada elemento, permite que un punto cualquiera en un elemento puede ser representado por medio de unas coordenadas constantes del marco de referencia, del elemento al que pertenezca, no importando si los elementos se están moviendo o si están estáticos. Y utilizando transformaciones homogéneas, (ver apéndice A), se pueden expresar las coordenadas de un punto en cierto elemento, según el sistema de referencia de otro elemento.

De esta forma, cada elemento se relaciona con el elemento precedente por medio de una transformación homogénea A_i y mediante transformaciones secuenciales, el actuador final, expresado en el marco de referencia del último elemento; se puede representar en coordenadas del marco de referencia fijo (inercial) de la base. Como A_i está formada por una matriz de rotación R_{i-1}^{i} y un vector de desplazamiento d_{i-1}^{i} : se tiene que A_i es función de seis variables, tres de rotación y tres de desplazamiento.

La representación Denavit_Hartenberg (D_H) simplifica las operaciones, ya que gran parte de ella se dedica a encontrar los marcos de referencia, que hagan posible que en lugar de seis variables sean cuatro las que representen a cada transformación homogénea, esas variables son[6]:

a_i Longitud

d_i Offset o distancia.

α_i Giro o ángulo de torsión.

θ_i Angulo.

---

[6] Se cumple que se pueda representar A_i por medio de cuatro variables si:
Z_i es el eje de la unión i + 1.
X_{i+1} es perpendicular a Z_i y además lo intersecta.

Pero al ser las uniones simples, la relación entre los marcos de referencia de elementos adyacentes, es la variable que describe el comportamiento de la unión que está entre ellos. Por esto de los cuatro parámetros, tres son constantes y sólo uno es una variable; que dependiendo del tipo de unión se tiene:

d_i es la variable de una unión prismática.

θ_i es la variable de una unión de revolución.

Al ser los elementos rígidos, se mantiene una relación constante entre los marcos de referencia en sus extremos, que se puede caracterizar con los parámetros constantes a_i, α_i y d_i o θ_i dependiendo del tipo unión en el extremo del elemento; por lo que son las características geométricas de cada unión/elemento, las que determinan el valor de los parámetros constantes. Por eso en algunas referencias al conjunto de los cuatro parámetros, se les llama parámetros cinemáticos estructurales o simplemente parámetros de los elementos. Para obtener la representación D_H, primero se numeran los elementos del manipulador, siendo el elemento cero la base. A continuación se numeran las uniones a partir de la unión que está en la base; como se ve en la figura 3.1

Después se especifican los marcos de referencia para cada elemento, de acuerdo a los siguientes pasos:

1) Localizar los ejes de las uniones. Donde Z_i es el eje de la unión $i+1$. Si la unión es de revolución, Z_i es el eje de rotación; si es prismática entonces es la dirección de traslación.

2) Establecer el marco base $(X_oY_oZ_o)$. El origen puede estar en cualquier lugar del eje Z_o.

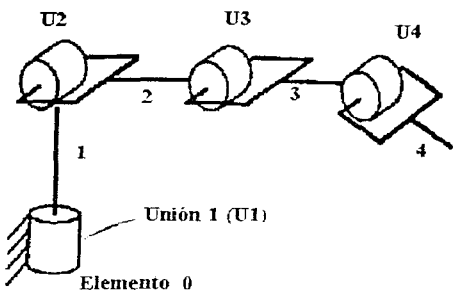


Fig. 3.1 Numeración en la representación D_H.

Para encontrar los marcos de referencia i=1,2,..n-1 se siguen los pasos 3 al 4, para cada marco de referencia:

3) Localizar el origen O_i, donde la normal común a Z_i y a Z_{i-1} intersecte a Z_i. Si Z_i intersecta a Z_{i-1}, localizar O_i en la intersección. Si Z_i y Z_{i-1} son paralelas, poner O_i en la unión i+1.

4) Establecer X_i a lo largo de la normal común entre Z_i y Z_{i-1}, a través de O_i o en la dirección normal al plano formado por Z_i y Z_{i-1}, si es que Z_i y Z_{i-1} se intersectan. Ubicar Y_i, de forma que se complete un marco de referencia positivo.

Después se establece el marco de referencia del actuador final ($O_nX_nY_nZ_n$). Asumiendo que la n-sima unión es de revolución, se ubica Z_n, en la dirección de Z_{n-1} y se fija el origen O_n, a la mitad de la pinza o en la punta de la herramienta. Se pone Y_n en la dirección de cerrado de la pinza. Si la herramienta no es una pinza, se coloca X_n y Y_n de forma que se tenga un marco de referencia positivo.

Por último se crea una tabla de parámetros $(a_i, d_i, \theta_i \text{ y } \alpha_i)$,donde:

-a_i es la distancia entre Z_{i-1} y Z_i, medida sobre el eje X_i.

-d_i es la distancia entre O_{i-1} y la intersección de X_i y Z_{i-1}, medida sobre el eje Z_{i-1}.

-α_i es el ángulo desde la parte positiva de Z_{i-1} a la parte positiva de Z_i, medido sobre el plano perpendicular a X_i.

-θ_i es el ángulo desde la parte positiva de X_{i-1} a la parte positiva de X_i, medido sobre el plano perpendicular a Z_{i-1}.

La forma en que se determinan los parámetros se muestra en la figura 3.2

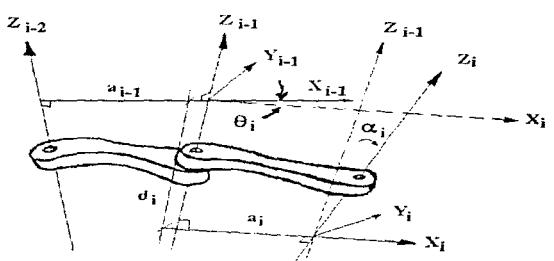


Fig. 3.2 Parámetros, en una configuración general.

En la representación de D_H, los marcos de referencia obtenidos no son únicos y fácilmente se pueden encontrar otros, ya que el sentido de los ejes de rotación puede escogerse libremente, tan

sólo respetando el signo de las rotaciones; también hay cierta libertad al escoger el origen del marco de referencia i (O_i), cuando Z_i y Z_{i-1} son paralelos. Por lo que variarán un poco las matrices (A_i) obtenidas pero, la transformación homogénea del actuador final a la base (T_0^n), obtiene los mismos resultados no importando las diferencias en las matrices A_i.

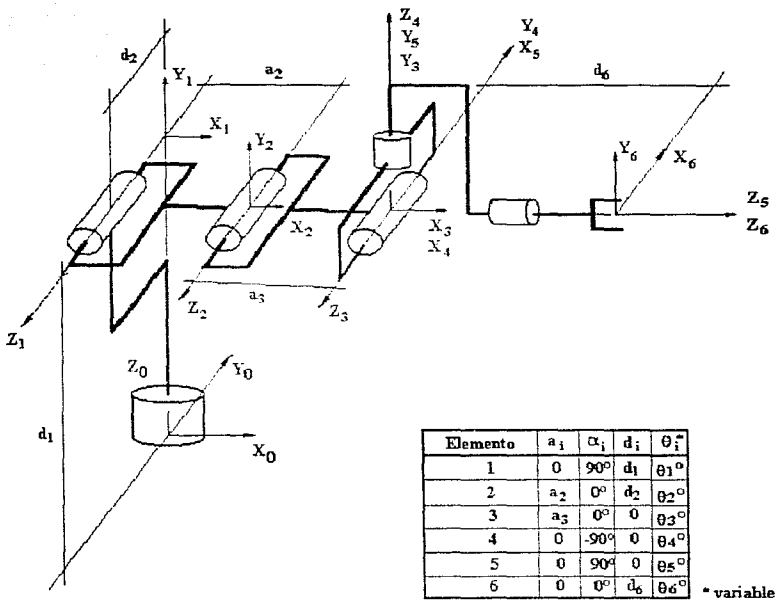
A la transformación T_0^n, se le pueden incluir transformaciones extras, como una transformación de la base (marco de referencia cero) a una referencia externa al manipulador, para tener una referencia global con otros equipos y con los objetos que van a ser manipulados. Otra transformación extra, podría ser una de la herramienta al marco de referencia "n", (último definido por la representación D_H), de forma que se sustituye la transformación adecuada, para la herramienta en uso.

$$T_{REF}^{HERR} = T_{REF}^{0}\ T_0^n\ T_n^{HERR} \qquad (3.1)$$

Donde T_{REF}^{HERR} es la transformación de la herramienta hasta una referencia fijada en cualquier lugar, no necesariamente en la base del manipulador.

Por simplicidad, en el análisis se considerará a una pinza, como única herramienta y no se tomará en cuenta el caso de un cambio de herramienta. En cuanto a la otra transformación, el manipulador que se está analizando no tiene un movimiento de la base y por tanto una transformación T_{REF}^{0}, tan sólo es un cambio de coordenadas que se puede efectuar al final del análisis y por lo que no se incluirá en este.

Un manipulador articulado con una muñeca esférica, tipo PPG, y una geometría con offset en la segunda unión, tiene una representación D_H como se muestra en la figura 3.3.



Elemento	a_i	α_i	d_i	θ_i^*
1	0	90°	d_1	θ_1°
2	a_2	0°	d_2	θ_2°
3	a_3	0°	0	θ_3°
4	0	-90°	0	θ_4°
5	0	90°	0	θ_5°
6	0	0°	d_6	θ_6°

* variable

Fig. 3.3 Representación D_H, para un manipulador con offset.

El mismo manipulador pero, con una muñeca esférica tipo GPG tiene una representación D_H con ciertas diferencias[7], ya que varían los marcos de referencia 3 y 4.

[7] K.S. Fu et al, "*Robótica: control, detección, visión e inteligencia*", México, Mc GrawHill, 1a. ed., trad. de la 1a. ed. en inglés, 1989, pp39.

3.1 Análisis cinemático.

El análisis cinemático de un manipulador, trata la geometría del movimiento de éste; sin cosiderar las fuerzas y momentos que lo originan y tratándolo, tanto a nivel de las variables en las uniones (q_i); como en la posición, orientación, velocidad y aceleraciones del actuador final.

Actualmente no hay un método simple para medir la posición, velocidad y aceleración del actuador final, ya que no es práctico acoplar transductores directamente porue agregan peso e impiden el movimiento. Por esto la mayoría de los robots calculan su posición, velocidad y aceleración por medio de un modelo cinemático y realizando las mediciones a nivel de cada una de las uniones.

El análisis cinemático se puede descomponer en tres partes principales:

a) La posición del actuador final, que a su vez se subdivide en dos:

-El problema directo, en el cual se parte de las variables de las uniones y se transforman en las variables de la posición del actuador final, por lo que se puede saber para un conjunto de variables de unión, en que posición se encuentra el actuador final.

-El problema inverso o hacia atrás en donde, dada una posición y orientación del actuador final, se obtienen las variables de las uniones que provocarían esa configuración.

b) La velocidad del actuador final, que también se divide en dos:

-La obtención de la matriz jacobiana, en el que partiendo de las velocidades de las uniones, se calcula la velocidad del actuador final.

-La velocidad inversa, en el cual dada una velocidad deseada para el actuador final, se encuentran las velocidades en las uniones.

c) La aceleración del actuador final:

-Problema directo, partiendo de las aceleraciones y velocidades de las uniones, se obtiene la aceleración del actuador final.

-Problema inverso; dada una aceleración deseada en el actuador final y conociendo la velocidad de las uniones, se calcula la aceleración necesaria en cada una de las uniones.

3.1.1 Posición del actuador final, problema directo

Para un manipulador de "n" elementos, al realizar la transformación del marco de referencia del actuador final ($O_nX_nY_nZ_n$), al marco de referencia fijo ($O_oX_oY_oZ_o$) se puede (dando las variables de las uniones y conociendo de antemano las características geométricas del manipulador) saber la posición, en coordenadas cartesianas, del marco de referencia del actuador final y su orientación en alguna forma estándar de representación, como en ángulos de Euler. De la forma en que simplifica la representación de D_H, se sabe que las matrices de transformación A_i, entre marcos de referencia adyacentes, se obtienen como[8]:

$$A_i = Rot_{z,\theta_i}\ Tras_{z,d_i}\ Tras_{x,a_i}\ Rot_{x,\alpha_i} \qquad (3.2)$$

[8] Mark W Spong y M. Vidyasagar, *Robot dynamics and control*, Singapur, John Wyley & Sons (Wyley international edition), 1a ed., 1989, pp. 65-70

Realizando las operaciones, se tiene que:

$$A_i = \begin{bmatrix} C_{\theta_i} & -S_{\theta_i}C_{\alpha_i} & S_{\theta_i}S_{\alpha_i} & a_iC_{\theta_i} \\ S_{\theta_i} & C_{\theta_i}C_{\alpha_i} & -C_{\theta_i}S_{\alpha_i} & a_iS_{\theta_i} \\ 0 & S_{\alpha_i} & C_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.3)$$

Por lo que el problema queda resuelto al sustituir los datos obtenidos en la representación de D_H, en la ecuación general (3.3) y se obtiene; para el caso de un manipulador de seis GDL:

$$T_0^6 = A_1\ A_2\ A_3\ A_4\ A_5\ A_6 \quad (3.4)$$

Los parámetros de la representación D_H para un manipulador articulado, con offset en la segunda unión, se muestran en la tabla 3.1

Elemento	a_i	α_i	d_i	θ_i
1	0	90°	d_1	$*\theta_1$
2	a_2	0°	d_2	$*\theta_2$
3	a_3	0°	0	$*\theta_3$
4	0	-90°	0	$*\theta_4$
5	0	90°	0	$*\theta_5$
6	0	0°	d_6	$*\theta_6$

Tabla 3.1 Parámetros D_H. (* variable)

Sustituyendo los datos de la tabla, en la ecuación (3.3), se tiene[9]:

$$A_1=\begin{bmatrix} C_1 & 0 & S_1 & 0 \\ S_1 & 0 & -C_1 & 0 \\ 0 & 1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad A_2=\begin{bmatrix} C_2 & -S_2 & 0 & a_2C_2 \\ S_2 & C_2 & 0 & a_2S_2 \\ 0 & 0 & 1 & d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad A_3=\begin{bmatrix} C_3 & -S_3 & 0 & a_3C_3 \\ S_3 & C_3 & 0 & a_3S_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_4=\begin{bmatrix} C_4 & 0 & -S_4 & 0 \\ S_4 & 0 & C_4 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad A_5=\begin{bmatrix} C_5 & 0 & S_5 & 0 \\ S_5 & 0 & -C_5 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad A_6=\begin{bmatrix} C_6 & -S_6 & 0 & 0 \\ S_6 & C_6 & 0 & 0 \\ 0 & 0 & 1 & d_6 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (3.5)$$

Para obtener T_0^6 se multiplican las matrices A_i (véase apéndice B) y se obtiene:

$$T_0^6=\begin{bmatrix} n_x & s_x & a_x & d_x \\ n_y & s_y & a_y & d_y \\ n_z & s_z & a_z & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (3.6)$$

Donde:

$$n_x=C_1(C_{234}C_5C_6-S_{234}S_6)-S_1S_5C_6$$
$$n_y=S_1(C_{234}C_5C_6-S_{234}S_6)+C_1S_5C_6$$
$$n_z=S_{234}C_5C_6+C_{234}S_6$$

$$s_x=-C_1(C_{234}C_5S_6+S_{234}C_6)+S_1S_5S_6$$
$$s_y=-S_1(C_{234}C_5S_6+S_{234}C_6)-C_1S_5S_6$$
$$s_z=C_{234}C_6-S_{234}C_5S_6$$

---

[9] $C_i=\cos(\theta_i)$ $\quad C_{ij}=\cos(\theta_i+\theta_j)$ $\quad C_{ijk}=\cos(\theta_i+\theta_j+\theta_k)$

$S_i=sen(\theta_i)$ $\quad S_{ij}=sen(\theta_i+\theta_j)$ $\quad S_{ijk}=sen(\theta_i+\theta_j+\theta_k)$

$$a_x = C_1C_{234}S_5 + S_1C_5$$
$$a_y = S_1C_{234}S_5 - C_1C_5$$
$$a_z = S_{234}S_5$$

$$d_x = C_1(C_{234}S_5d_6 + a_3C_{23} + a_2C_2) + S_1(C_5d_6 + d_2)$$
$$d_y = S_1(C_{234}S_5d_6 + a_3C_{23} + a_2C_2) - C_1(C_5d_6 + d_2)$$
$$d_z = S_{234}S_5d_6 + a_3S_{23} + a_2S_2 + d_1$$

Donde: n representa a $\mathbf{i}_6$,**s** representa a $\mathbf{j}_6$,**a** representa a $\mathbf{k}_6$, expresados en el marco de referencia 0. **d** es el vector del origen del marco de referencia 0 al origen del marco 6.

De la matriz T_0^6 , al sustituir q_1, q_2, ... q_6 (θ_1,θ_2,.....,θ_6), se encuentra una matriz de rotación y un vector de desplazamiento; la posición de la herramienta o el punto medio de la pinza está dado por el vector de desplazamiento y la orientación se da en la matriz de rotación.

Para un manipulador sin offset, la única diferencia, es que el elemento 2 no cuenta con d_2, por lo que se modifica la matriz A_2

$$A_2 = \begin{bmatrix} C_2 & -S_2 & 0 & a_2C_2 \\ S_2 & C_2 & 0 & a_2S_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \textbf{(3.7)}$$

Pero se obtiene una transformación T_0^6 muy parecida al resultado anterior, tan sólo con $d_2=0$.

Con los resultados obtenidos, se puede encontrar la posición y orientación del actuador final, para un conjunto cualquiera de posiciones en las uniones.

3.1.2 Posición del actuador final, problema inverso

El problema inverso tiene como entradas a la posición y orientación del actuador final; es decir para un manipulador de seis GDL se conoce la matriz de transformación T_0^6 y se quiere obtener las variables de las uniones, que producen dicha configuración del manipulador. Partiendo del supuesto que la posición dada está dentro del espacio de trabajo.

Del problema directo, se sabe que T_0^6 depende de las variables de las uniones:

$$T_0^6(q_1,q_2,q_3,\cdots,q_6) = A_1 \cdot A_2 \cdot A_3 \cdots A_6 = \begin{bmatrix} R & d \\ 0 & 1 \end{bmatrix} \tag{3.8}$$

T_0^6, es una matriz de 4x4, que es función de seis variables (q_i), pero el último renglón de la matriz es constante, por lo que genera doce ecuaciones para seis incógnitas y como hay más ecuaciones que incógnitas, se tienen múltiples soluciones para el problema inverso. Si un manipulador tiene uniones prismáticas para sus movimientos regionales, entonces sólo hay una forma de alcanzar cada punto en el espacio de trabajo pero, si tiene al menos una unión de revolución entonces para algunas posiciones, puede haber más de una forma de alcanzarlas.

No existe una solución cerrada al problema inverso de un manipulador de seis GDL, lo que hay son varios métodos de solución para configuraciones cinemáticas específicas; como por ejemplo los métodos: algebraicos, iterativos, geométricos, etc. Se va a utilizar el método geométrico ya que aplicado a manipuladores simples (cuya geometría de las primeras tres uniones, tiene un par

de uniones de revolución o un par de uniones prismáticas y la muñeca es esférica) tiene ventajas sobre los otros métodos, en cuanto a la rapidez y simplicidad en la obtención de una solución cerrada y que es más explícita la elección de la solución correcta, para una configuración del brazo dada.

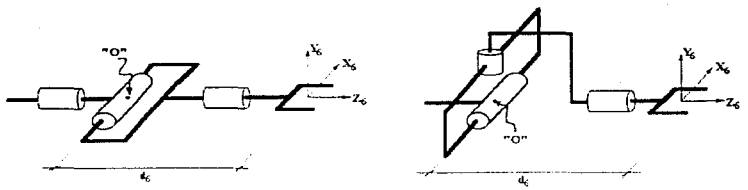
Para facilitar el análisis, a nivel del actuador final se desacopla el posicionamiento de la orientación; esto se puede hacer si la muñeca es esférica (las últimas tres uniones intersectan sus ejes en el centro de la muñeca) y con esto, el centro de la muñeca no cambia de posición por efecto del movimiento de las últimas tres uniones; es decir solamente intervienen en su posición las tres primeras uniones.

El método geométrico implica una serie de pasos:

1) Partiendo de T_0^6, se conoce R_0^6 y **d** (orientación y posición del actuador final). Se realiza el desacople, al encontrar la posición del centro de la muñeca "P", en función de **"d"** y R_0^6.

2) Por medio de la geometría del manipulador, al conocer el centro de la muñeca, se encuentran los valores para las primeras tres uniones.

3) Con los valores de las tres primeras uniones, se encuentra R_0^3 y al conocer R_0^6 se puede encontrar R_3^6, que da los valores de las últimas tres uniones, con lo que queda resuelto el problema inverso.



Ilustr. 3.4 Muñeca esférica.

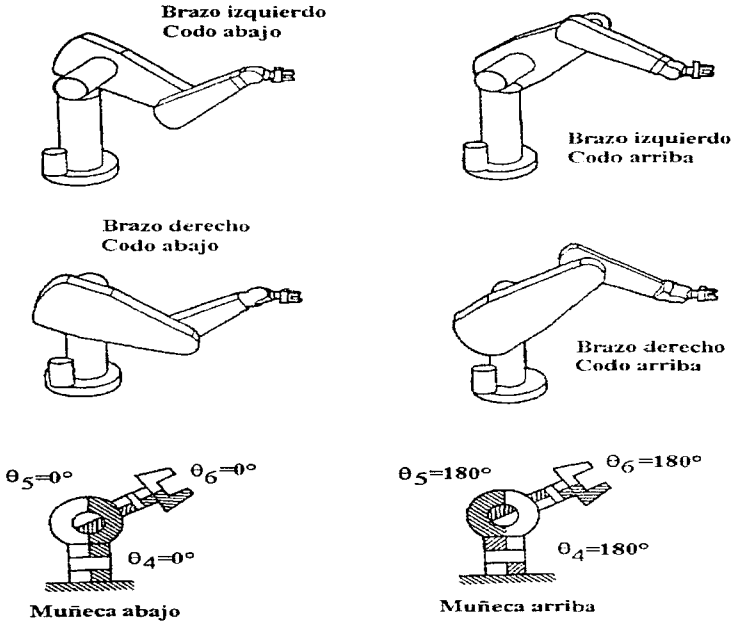
Las dos muñecas de la figura 3.4, pese a que tienen una configuración cinemática distinta, son análogas en cuanto a que tienen la misma transformación T_3^6 y como se puede apreciar en la misma figura, del centro de la muñeca "O" al origen del sistema de referencia del actuador final ($X_6Y_6Z_6$), sólo se tiene un desplazamiento d_6. Por lo que para saber la posición del centro de la muñeca, partiendo de la posición del actuador final "d" se tiene:

$$P = d - (d_6 \cdot R_0^6 \cdot Z_6) \qquad (3.9)$$

La posición del centro de la muñeca "P", es función sólo de θ_1, θ_2 y θ_3 [10].

En un manipulador articulado tipo codo, se tiene cierta redundancia y se puede llegar a tener múltiples soluciones al problema inverso. De forma general se tiene que para un manipulador de este tipo, con offset en la segunda unión, hay ocho posibles soluciones según muestra la figura 3.5, si no tuviera offset se tendrían sólo cuatro posibles soluciones.

[10] Pese a que las dos muñecas esféricas de la figura 3.4 son análogas, la representación D_H del manipulador completo, cambia de acuerdo a cual muñeca se utilice. En todo el análisis se supone que el manipulador tiene una muñeca tipo "PPG".



Ilustr. 3.5 Diferentes configuraciones, de un manipulador articulado tipo codo.

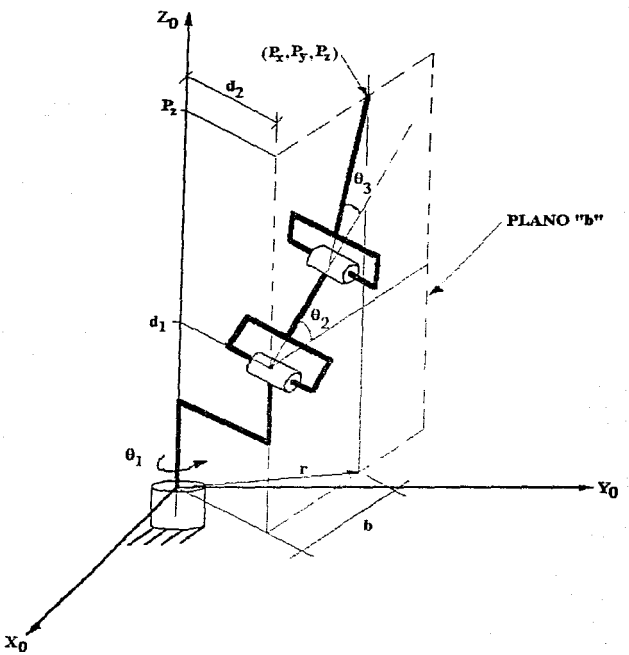
Las características de cada solución se verán conforme se presenten en el análisis. Para poder escoger una solución en particular se definen tres indicadores o variables:

BRAZO= +1 (brazo derecho) -1 (brazo izquierdo)

CODO= +1 (codo arriba) -1 (codo abajo)

MUÑECA= +1 (muñeca abajo) -1 (muñeca arriba).

Posición inversa: en la figura 3.6 se puede ver la relación entre las primeras tres uniones y el punto "P" (centro de la muñeca).



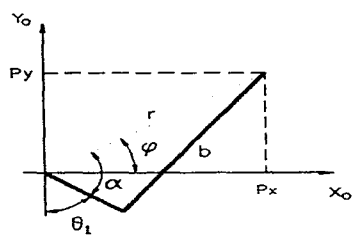
Ilustr. 3.6 Posición inversa.

-Primera variable (θ_1):

Para esta unión, se tienen ciertas características dependiendo del offset en la unión dos (d_2). Si no hay un offset, en el caso de que Px=Py=0 el centro de la muñeca está intersectando a Z_0, y entonces cualquier valor de θ_1 sirve; es decir que el manipulador está en una configuración singular o degenerada. En una zona no singular, simplemente la primera unión es $\theta_1 = tang^{-1}\frac{Py}{Px}$.

Si la unión dos tiene un offset, entonces no se da la configuración singular, pero se tiene que hay dos soluciones para θ_1, la llamada brazo derecho y brazo izquierdo (figuras 3.7 y 3.8).

Se puede identificar a la configuración de brazo derecho, por medio de la unión dos (θ_2), ya que al moverse en el sentido positivo, mueve a la muñeca en el sentido positivo de Z_0; mientras que en la configuración de brazo izquierdo, el mismo movimiento de la unión dos, mueve a la muñeca en el sentido negativo de Z_0.



$$\theta_1 = \varphi - \alpha + 90^\circ$$

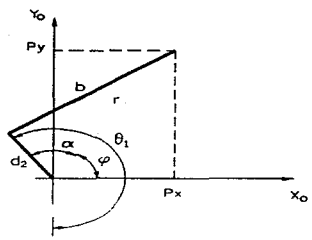
$$\theta_1 = \tan^{-1}\frac{Py}{Px} - \tan^{-1}\frac{b}{d_2} + 90^\circ$$

Con:

$$b = \sqrt{r^2 - (d_2)^2}$$

$$b = \sqrt{(Px)^2 + (Py)^2 - (d_2)^2}$$

Ilustr. 3.7 Brazo derecho.



$$\theta_1 = \varphi + \alpha + 90°$$

$$\theta_1 = \tan^{-1}\frac{Py}{Px} + \tan^{-1}\frac{b}{d_2} + 90°$$

Con:

$$b = \sqrt{r^2 - (d_2)^2}$$

$$b = \sqrt{(Px)^2 + (Py)^2 - (d_2)^2}$$

Ilustr. 3.8 Brazo izquierdo.

Uniendo los dos casos se tiene:

$$\theta_1 = 90° + tang^{-1}\left(\frac{P_y}{P_x}\right) - (BRAZO)\,tang^{-1}\left(\frac{\sqrt{(P_x)^2 + (P_y)^2 - (d_2)^2}}{d_2}\right) \qquad \textbf{(3.10)}$$

La misma ecuación se puede escribir de la forma:

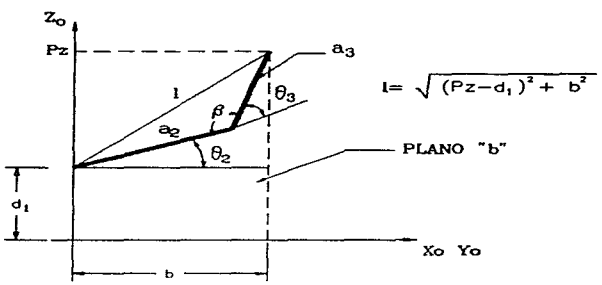
$$\theta_1 = 90° + tang^{-1}\left[\frac{P_y(d_2) - (BRAZO)\,P_x(b)}{P_x(d_2) + (BRAZO)\,P_y(b)}\right] \quad ; \; b = \sqrt{(P_x)^2 + (P_y)^2 - (d_2)^2} \qquad \textbf{(3.11)}$$

Donde:

$$tang^{-1}\left[\frac{A}{B}\right] = \begin{cases} Si\ A>0,\ B>0 & \theta_1 = tang^{-1}\left[\frac{A}{B}\right] \\ Si\ A>0,\ B<0 & \theta_1 = 180° + tang^{-1}\left[\frac{A}{B}\right] \\ Si\ A<0,\ B<0 & \theta_1 = -180° + tang^{-1}\left[\frac{A}{B}\right] \\ Si\ A<0,\ B>0 & \theta_1 = tang^{-1}\left[\frac{A}{B}\right] \end{cases}$$

-Segunda y tercer variable (θ_2 y θ_3):

El movimiento causado por las uniones dos y tres, se encuentra dentro del plano "b", y se muestra en la figura 3.9



Ilustr. 3.9 Relaciones de la tercera unión.

De la ley de cosenos para un triángulo oblicuángulo, se tiene que:

$$l^2 = (a_2)^2 + (a_3)^2 - 2(a_2)(a_3)\cos(\beta) \qquad \textbf{(3.12)}$$

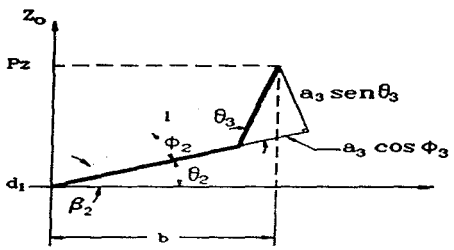
Y tras un desarrollo:

$$\theta_3 = tang^{-1}\left[\frac{(-CODO)\sqrt{1-A^2}}{A}\right] \quad ; A = \frac{b^2 + (P_z - d_1)^2 - (a_2)^2 - (a_3)^2}{2(a_2)(a_3)} \qquad \textbf{(3.13)}$$

b depende si hay offset o no:

$$\text{sin } \textit{offset}: \; b = r = \sqrt{P_x^2 + P_y^2}$$
$$\textit{con offset}: \; b = \sqrt{r^2 - (d_2)^2} = \sqrt{P_x^2 + P_y^2 - (d_2)^2} \qquad \textbf{(3.14)}$$

Para θ_2 se tiene:



Ilustr. 3.10 Relación entre la unión 2 y 3.

$$\theta_2 = \beta_2 - \Phi_2 \quad ;donde: \beta_2 = \tan^{-1}\left[\frac{P_z - d_1}{b}\right]$$

$$\Phi_2 = \tan^{-1}\left[\frac{a_3 sen\theta_3}{a_2 + a_3 \cos\theta_3}\right] \qquad \textbf{(3.15)}$$

b, se obtiene de la misma forma que para θ_3

-Orientación inversa: θ_1, θ_2 *y* θ_3 se sustituyen en R_0^3 y como:

$$R_3^6 = (R_O^3)^{-1} \cdot R_o^6 = (R_O^3)^T \cdot R_O^6 \qquad \textbf{(3.16)}$$

Se obtiene la orientación cinemática inversa (R_3^v) la que se iguala a una matriz de rotación, representada en ángulos de Euler (A_4;A_5;A_6).

$$\begin{bmatrix} C_\Phi C_\theta C_\Psi - S_\Phi S_\Psi & -C_\Phi C_\theta S_\Psi - S_\Phi C_\Psi & C_\Phi S_\theta \\ S_\Phi C_\theta C_\Psi + C_\Phi S_\Psi & -S_\Phi C_\theta S_\Psi + C_\Phi C_\Psi & S_\Phi S_\theta \\ -S_\theta C_\Psi & S_\theta S_\Psi & C_\theta \end{bmatrix} = \begin{bmatrix} u_{11} & u_{12} & u_{13} \\ u_{21} & u_{22} & u_{23} \\ u_{31} & u_{32} & u_{33} \end{bmatrix} = U \qquad \textbf{(3.17)}$$

Matriz de rotación, en ángulos de Euler.

Por lo que conociendo u_{ij}, se obtiene θ, Φ *y* Ψ.

Normalmente ($u_{13}\neq 0$ y $u_{23}\neq 0$) $S_\theta \neq 0$ y se tienen dos soluciones posibles:

$$u_{33} = C_\theta \quad ; S_\theta = \pm\sqrt{1-(u_{33})^2}$$

$$\theta = \tan^{-1}\left[\frac{\pm\sqrt{1-(u_{33})^2}}{u_{33}}\right] = \tan^{-1}\left[\frac{(MUÑECA)\sqrt{1-(u_{33})^2}}{u_{33}}\right]$$

Dependiendo del valor de MUÑECA se tiene dos soluciones: **(3.18)**

$MUÑECA = +1$	$MUÑECA = -1$
$\phi = \tan^{-1}(\frac{u_{23}}{u_{13}})$	$\phi = \tan^{-1}(\frac{-u_{23}}{-u_{13}})$
$\psi = \tan^{-1}(\frac{u_{32}}{-u_{31}})$	$\psi = \tan^{-1}(\frac{-u_{32}}{u_{31}})$

Pero si $u_{13}=0$ y $u_{23}=0$, entonces $S_\theta = 0$, por lo que C_θ puede ser ± 1.

El que $S_\theta = 0$, implica que Z_3 y Z_5 son colineales por lo que la muñeca está en una configuración singular y se tiene que resolver el problema inverso de una forma diferente de como se hizo anteriormente; con lo que se obtiene que la solución a la orientación inversa, tiene dos posibles soluciones cuando no se está en una configuración singular y a parte se tiene que manejar el caso especial de la singularidad.

Caso singular:

-Si $u_{33}=1$ entonces $C_\theta = 1$, por lo que $\theta = 0$ y la matriz queda:

$$\begin{bmatrix} C_\phi C_\psi - S_\phi S_\psi & -C_\phi S_\psi - S_\phi C_\psi & 0 \\ S_\phi C_\psi + C_\phi S_\psi & -S_\phi S_\psi + C_\phi C_\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} C_{\phi\psi} & -S_{\phi\psi} & 0 \\ S_{\phi\psi} & C_{\phi\psi} & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \textbf{(3.19)}$$

Unicamente se puede obtener la suma de ϕ y ψ.

$$\phi + \psi = \tan^{-1}\left(\frac{u_{21}}{u_{11}}\right) = \tan^{-1}\left(\frac{-u_{12}}{u_{11}}\right) = \tan^{-1}\left(\frac{u_{21}}{u_{22}}\right) \qquad \textbf{(3.20)}$$

Se puede tomar la convención que $\phi = 0$, y se obtiene ψ.

-Si $u_{33} = -1$ entonces $C_\theta = -1$ por lo que $\theta = \pi$ [*rad*]. En este caso la matriz de rotación queda:

$$\begin{bmatrix} -C_\phi C_\psi - S_\phi S_\psi & C_\phi S_\psi - S_\phi C_\psi & 0 \\ -S_\phi C_\psi + C_\phi S_\psi & S_\phi S_\psi + C_\phi C_\psi & 0 \\ 0 & 0 & -1 \end{bmatrix} = \begin{bmatrix} -C_{(\phi-\psi)} & S_{(\phi-\psi)} & 0 \\ S_{(\phi-\psi)} & C_{(\phi-\psi)} & 0 \\ 0 & 0 & -1 \end{bmatrix} \qquad \textbf{(3.21)}$$

Y únicamente se puede obtener la resta de ϕ y ψ.

$$\phi - \psi = \tan^{-1}\left(\frac{u_{12}}{-u_{11}}\right) = \tan^{-1}\left(\frac{u_{21}}{u_{22}}\right) \qquad \textbf{(3.22)}$$

Se puede seguir la misma convención ($\phi = 0$) y de esta forma se obtiene ψ.

Por último se tiene que ϕ *es* θ_4, θ *es* θ_5, ψ *es* θ_6. Con lo que el problema inverso queda resuelto.

3.1.3 La velocidad del actuador final, obtención de la jacobiana

Las relaciones de velocidad con la forma del problema directo, es decir al conocer las velocidades de las uniones se obtiene la velocidad lineal y angular del actuador final; con respecto al marco de referencia 0 son de la forma:

$$V_6 = J_v \dot{q}$$
$$\omega_6 = J_\omega \dot{q} \qquad (3.23)$$

Donde: $\dot{q}$ es la velocidad angular de las uniones.

v_6 es la velocidad lineal del actuador final.

ω_6 es la velocidad angular del actuador final.

La matriz jacobiana es la unión de las dos relaciones, por lo que se tiene que es una matriz de 6 X n, donde n es el número de uniones; es decir que el jacobiano es una matriz de transformación, entre el vector de dimensión n (para un manipulador de n grados de libertad o uniones) y el vector de dimensión 6 que representa las velocidades lineal y angular del actuador final.

$$\dot{X} = \begin{bmatrix} v_6 \\ \omega_6 \end{bmatrix} = J\ \dot{q} \quad ; J = \begin{bmatrix} J_v \\ J_\omega \end{bmatrix} \qquad (3.24)$$

Donde J es la jacobiana del manipulador.

Hay varios métodos para encontrar la jacobiana de un manipulador; y se podría decir que son equivalentes, aunque algunos métodos la obtienen de forma analítica y otros sólo de forma numérica.

Algunos de los métodos son:

-Producto vectorial.

-Traslación y rotación diferencial.

-Partiendo de las ecuaciones de Newton_Euler.

-Por diferenciación directa del problema directo.

La jacobiana se obtendrá por medio del método del producto vectorial[11], ya que es fácil de aplicar (mucho del trabajo se realizó en la parte del problema de posicionamiento directo) y se obtiene de forma analítica.

La jacobiana, para un manipulador articulado está formado por:

$$J = [J_1 \; J_2 \; \cdots \; J_n] \quad ; \; Con \; J_i = \begin{bmatrix} Z_{i-1} \times (d_0^n - d_0^{i-1}) \\ Z_{i-1} \end{bmatrix} \tag{3.25}$$

$$Donde \; Z_{i-1} = R_0^{i-1} k_{i-1}$$

Con la forma general de la jacobiana, se tiene que su desarrollo para un manipulador articulado tipo codo es: (3.26)

$$J = \begin{bmatrix} Z_0 \times (d_0^6 - d_0^0) & Z_1 \times (d_0^6 - d_0^1) & Z_2 \times (d_0^6 - d_0^2) & Z_3 \times (d_0^6 - d_0^3) & Z_4 \times (d_0^6 - d_0^4) & Z_5 \times (d_0^6 - d_0^5) \\ Z_0 & Z_1 & Z_2 & Z_3 & Z_4 & Z_5 \end{bmatrix}$$

Donde Z_i es la tercera columna en la matriz de rotación R_0^i, de la transformación T_0^i. De esa misma transformación se obtiene d_0^i y por ser una muñeca esférica d_0^3, d_0^4 y d_0^5, son iguales.

[11] K.S. Fu et al, *op. cit.*, pp 564-567

En el problema directo, se obtuvo T_0^6, y de esa transformación:

$$d_0^6 = \begin{bmatrix} C_1(C_{234}S_5d_6 + a_3C_{23} + a_2C_2) + S_1(C_5d_6 + d_2) \\ S_1(C_{234}S_5d_6 + a_3C_{23} + a_2C_2) - C_1(C_5d_6 + d_2 \\ S_{234}S_5d_6 + a_3S_{23} + a_2S_2 + d_1 \end{bmatrix} \tag{3.27}$$

En el desarrollo de T_0^6, (Apéndice B) se obtuvo $T_0^1, T_0^2, T_0^3, T_0^4$ y T_0^5 y de esos resultados parciales, se tiene:

$$Z_5 = \begin{bmatrix} C_1C_{234}S_5 + S_1C_5 \\ S_1C_{234}S_5 - C_1C_5 \\ S_{234}S_5 \end{bmatrix} \quad Z_4 = \begin{bmatrix} -C_1S_{234} \\ -S_1S_{234} \\ C_{234} \end{bmatrix} \quad Z_3 = \begin{bmatrix} S_1 \\ -C_1 \\ 0 \end{bmatrix}$$

$$Z_2 = \begin{bmatrix} S_1 \\ -C_1 \\ 0 \end{bmatrix} \quad Z_1 = \begin{bmatrix} S_1 \\ -C_1 \\ 0 \end{bmatrix} \quad Z_0 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{3.28}$$

$$d_0^3 = \begin{bmatrix} a_3C_1C_{23} + a_2C_1C_2 + d_2S_1 \\ a_3S_1C_{23} + a_2S_1C_2 - d_2C_1 \\ a_3S_{23} + a_2S_2 + d_1 \end{bmatrix} = d_0^4 = d_0^5$$

$$d_0^2 = \begin{bmatrix} a_2C_1C_2 + d_2S_1 \\ a_2S_1C_2 - d_2C_1 \\ a_2S_2 + d_1 \end{bmatrix} \quad d_0^1 = \begin{bmatrix} 0 \\ 0 \\ d_1 \end{bmatrix} \quad d_0^0 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{3.29}$$

Lo que resta para la obtención del jacobiano, es realizar las operaciones del tipo $Z_i \times (d_0^6 - d_0^i)$.

Desarrollando la matriz del jacobiano se tiene:

$$\begin{bmatrix} J_{11} & J_{12} & J_{13} & J_{14} & J_{15} & J_{16} \\ J_{21} & J_{22} & J_{23} & J_{24} & J_{25} & J_{26} \\ J_{31} & J_{32} & J_{33} & J_{34} & J_{35} & J_{36} \\ J_{41} & J_{42} & J_{43} & J_{44} & J_{45} & J_{46} \\ J_{51} & J_{52} & J_{53} & J_{54} & J_{55} & J_{56} \\ J_{61} & J_{62} & J_{63} & J_{64} & J_{65} & J_{66} \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \\ \dot{\theta}_4 \\ \dot{\theta}_5 \\ \dot{\theta}_6 \end{bmatrix} = \begin{bmatrix} \vec{V}_{6x} \\ \vec{V}_{6y} \\ \vec{V}_{6z} \\ \vec{\omega}_{6x} \\ \vec{\omega}_{6y} \\ \vec{\omega}_{6z} \end{bmatrix} \tag{3.30}$$

Donde:

$$J_{11} = C_1(C_5 d_6 + d_2) - S_1(C_{234} S_5 d_6 + a_3 C_{23} + a_2 C_2)$$

$$J_{21} = C_1(C_{234} S_5 d_6 + a_3 C_{23} + a_2 C_2) + S_1(C_5 d_6 + d_2)$$

$$J_{31} = 0$$

$$J_{41} = 0$$

$$J_{51} = 0$$

$$J_{61} = 1$$

$$J_{12} = -C_1(S_{234} S_5 d_6 + a_3 S_{23} + a_2 S_2)$$

$$J_{22} = -S_1(S_{234} S_5 d_6 + a_3 S_{23} + a_2 S_2)$$

$$J_{32} = C_{234} S_5 d_6 + a_3 C_{23} + a_2 C_2$$

$$J_{42} = S_1$$

$$J_{52} = -C_1$$

$$J_{62} = 0$$

$$J_{13} = -C_1(S_{234} S_5 d_6 + a_3 S_{23})$$

$$J_{23} = -S_1(S_{234} S_5 d_6 + a_3 S_{23})$$

$$J_{33} = C_{234} S_5 d_6 + a_3 C_{23}$$

$$J_{43} = S_1$$

$$J_{53} = -C_1$$

$$J_{63} = 0$$

$$J_{14} = -C_1 S_{234} S_5 d_6$$

$$J_{24} = -S_1 S_{234} S_5 d_6$$

$$J_{34} = C_{234} S_5 d_6$$

$$J_{44} = S_1$$

$$J_{54} = -C_1$$

$$J_{64} = 0$$

$$J_{15} = C_1 C_{234} C_5 d_6 - S_1 S_5 d_6$$

$$J_{25} = C_1 S_5 d_6 + S_1 C_{234} C_5 d_6$$

$$J_{35} = S_{234} C_5 d_6$$

$$J_{45} = -C_1 S_{234}$$

$$J_{55} = -S_1 S_{234}$$

$$J_{65} = C_{234}$$

$$J_{16} = 0$$

$$J_{26} = 0$$

$$J_{36} = 0$$

$$J_{46} = C_1 C_{234} S_5 + S_1 C_5$$

$$J_{56} = S_1 C_{234} S_5 - C_1 C_5$$

$$J_{66} = S_{234} S_5$$

La Jacobiana, del manipulador sin offset, es el mismo tan sólo con $d_2 = 0$.

El desarrollo de esta matriz indica que depende de los diferentes valores de **q**, que en su conjunto, dan una configuración geométrica específica del manipulador que se refleja en la jacobiana. Los valores de **q** en los que el rango de la matriz (número de columnas linealmente independientes) disminuye, corresponde con las configuraciones singulares o degeneradas.

Hay dos configuraciones singulares en un manipulador articulado tipo codo: en el codo cuando

la unión 3 está totalmente extendida ($\theta_3 = 0$) y cuando está totalmente retraída ($\theta_3 = 180°$).

Y en la intersección del centro de la muñeca con Z_0, que no se da si hay un offset en la unión dos ($d_2 \neq 0$).

3.1.4 La velocidad del actuador final, problema inverso

Conociendo la velocidad que se requiere en el actuador final, se encuentran las velocidades en las uniones que se necesitan para esto.

$$\begin{aligned} &Si && J\dot{q} = \dot{X} \\ &Entonces && \dot{q} = J^{-1}\dot{X} \end{aligned} \qquad (3.31)$$

Y se obtiene a partir de la inversa de la jacobiana y de la velocidad deseada para el actuador final ($\dot{X}$), el valor de las velocidades en las uniones ($\dot{q}$).

3.1.5 Aceleración del actuador final

En el problema directo se deriva la ecuación que relaciona la jacobiana, con la velocidad del actuador final.

$$\begin{aligned} &\frac{d}{dt}(J\dot{q}) = \frac{d}{dt}(\dot{X}) \\ &(\frac{d}{dt}J)\dot{q} + J(\ddot{q}) = \ddot{X} \end{aligned} \qquad (3.32)$$

Con lo que se obtiene la aceleración del actuador final ($\ddot{X}$), en función de las velocidades y aceleraciones del las uniones.

En el problema inverso, se despeja la aceleración de las uniones ($\ddot{q}$), de la ecuación anterior. Dados $\ddot{X}, \dot{q}$, se obtiene $\ddot{q}$.

3.2 Análisis Dinámico.

Las fuerzas y pares producidos por los actuadores en las uniones, producen aceleraciones, velocidades y cambios de posición que dependen en gran medida de las características dinámicas de los elementos del manipulador; así como de los actuadores y de los sistemas de transmisión de potencia. Debido a que un manipulador es un sistema mecánico complejo, el problema de obtener las ecuaciones que relacione a los pares y fuerzas de los actuadores, con las variables de movimiento de cada uno de los elementos tiende a complicarse en forma proporcional al número de elementos (# de GDL) y para un manipulador de seis grados de libertad, la complejidad de las ecuaciones es bastante grande; por eso hay varias aproximaciones en la obtención de las ecuaciones dependiendo del uso final de éstas, por ejemplo para el control de un manipulador se tiene que se debe de calcular la dinámica de éste en tiempo real, por lo que se necesitan ecuaciones con un número mínimo de operaciones. Por el contrario si se quiere analizar un manipulador y poder discernir las diferentes aportaciones al comportamiento general; se requieren ecuaciones con una estructura muy definida pero, por lo general se tiene una solución demasiado extensa, muy difícil de calcular en tiempo real y que involucra muchos parámetros de los elementos.

En la mayoría de la bibliografía se tratan dos aproximaciones, las ecuaciones de Lagrange-Euler y las ecuaciones de Newton-Euler pero, hay más métodos de obtener las ecuaciones dinámicas aunque muchos de ellos, son sólo simplificaciones u optimizaciones de los métodos generales.

Ecuaciones de Lagrange-Euler:

En este tipo de ecuaciones, se trata al manipulador como un todo y por la obtención del

Lagrangiano (diferencia entre la energía cinética y la energía potencial) del sistema se logra el planteamiento de las ecuaciones y se tienen tantas ecuaciones como elementos tiene el manipulador. Las ecuaciones son de tipo cerrado, es decir bien especificadas para todos los valores de las coordenadas generalizadas. Este tipo de ecuaciones son muy útiles ya que el comportamiento en el tiempo de las fuerzas generalizadas, está totalmente descrito al tener la evolución en el tiempo de las coordenadas generalizadas.

Ecuaciones Newton-Euler:

Estas ecuaciones en lugar de tratar al manipulador como un todo, se concentra en cada elemento a la vez, de forma que se toman en cuenta los acoplamientos dinámicos entre elementos adyacentes. Por medio de un método iterativo hacia adelante y después hacia atrás, se llega a conocer para un determinado conjunto de coordenadas generalizadas y sus dos primeras derivadas, el conjunto de fuerzas y pares que se deben de ejercer, para que se de el comportamiento deseado del manipulador.

Por lo que se obtiene la relación de fuerzas, para una cierta trayectoria (representada por **q** y sus derivadas), pero no se obtiene ecuaciones que directamente describa el comportamiento del manipulador en el tiempo.

Por su naturaleza recursiva, se utilizan ampliamente en el cálculo en tiempo real del comportamiento dinámico del sistema, aunque su estructura no permite que se puedan aplicar en técnicas avanzadas de control.

3.2.1 Ecuaciones de Lagrange-Euler

Las ecuaciones de Lagrange-Euler son ecuaciones diferenciales que describen la evolución en el tiempo de sistemas mecánicos, bajo restricciones "holonómicas" y cuando las fuerzas de restricción satisfacen el principio del trabajo virtual. Es decir un sistema mecánico que puede ser descrito por ecuaciones de Lagrange_Euler, tiene que cumplir varios aspectos:

1) Restricciones holonómicas.

Las restricciones son un concepto usado para describir el movimiento de un sistema mecánico, y sirven para representar las fuerzas dentro del sistema, que no se pueden conocer antes de que el movimiento del sistema sea entendido completamente. Las restricciones se clasifican en dos tipos: holonómicas y no holonómicas[12].

Las ecuaciones de restricción (no diferenciales) finitas y las ecuaciones diferenciales que se pueden expresar en una forma finita (integrada), siempre son restricciones holonómicas y tienen el efecto de reducir los grados de libertad del sistema; por lo que un sistema formado por "k" partículas, si no tuviera restricciones se necesitarían "k" coordenadas, todas independientes entre si, para describir el estado del sistema. Con restricciones holonómicas, los GDL del sistema disminuyen y se tiene que ya no son necesarias las "k" coordenadas, si no un número menor, las cuales son las coordenadas generalizadas del sistema.

---

[12] Las restricciones no holonómicas, son aquellas que se representan por desigualdades finitas (fronteras entre ecuaciones/etapas) y las ecuaciones diferenciales que no se pueden expresar en una forma integrada. Las restricciones no holonómicas, por lo general provocan que haya más coordenadas generalizadas que grados de libertad, para poder describir totalmente al sistema. Normalmente se dan en problemas de rodamientos ideales en tres dimensiones.

Por ejemplo el número de partículas que forman un cuerpo rígido, es casi infinito pero al estar contenidas en un cuerpo rígido, las restricciones del sistema hace que con seis coordenadas generalizadas se pueda describir a cualquier partícula que pertenezca al cuerpo, (tres coordenadas de posición y tres de orientación, del cuerpo rígido).

2) El que las fuerzas de restricción, satisfagan el principio del trabajo virtual.

El principio del trabajo virtual es una alternativa mucho más poderosa, en los problemas de estática con geometría no constante, que la primera ley de Newton; ya que no involucra a las fuerzas internas y de restricción y por tanto el número de ecuaciones es reducido. De acuerdo a este principio, una forma de analizar las restricciones, es por sus efectos por lo que se analizan los desplazamientos posibles de los elementos en su conjunto, sin violar las restricciones impuestas por el sistema en si. Estos desplazamientos se llaman desplazamientos virtuales[13] y un desplazamiento virtual de un sistema, es cualquier combinación de desplazamientos virtuales de sus puntos.

El trabajo virtual, se define como el incremento de trabajo, realizado por una fuerza aplicada a una partícula, cuando esta se desplaza virtualmente. Las restricciones que se analizan más frecuentemente son ideales, es decir que el trabajo virtual producido por las reacciones del sistema, en un desplazamiento virtual cualquiera, es cero. Por lo que el principio del trabajo virtual se aplica, cuando la única restricción es que el movimiento sea con las características de

[13] Tienen las características que:
-Son infinitesimales, de lo contrario cambiaría la configuración del sistema, así como sus condiciones iniciales.
-Son consistentes con las restricciones del sistema, si no la estructura del sistema no se respetaría.
-Son imaginarios, en cuanto a que son proposiciones y no un desplazamiento asociado a un movimiento real.
-Ocurren en forma instantánea, en un tiempo cero; es decir que el tiempo es constante durante el desplazamiento virtual.

un cuerpo rígido (restricciones holonómicas e ideales). El supuesto de cuerpo rígido, previene de la absorción de cualquier trabajo, por parte del sistema, ya sea por deformación elástica o por fricción; lo que impide que se pueda tomar en cuenta a las fuerzas de fricción en los puntos de contacto entre cuerpos, es decir es un sistema ideal (sin absorción de energía en las uniones).

El desarrollo detallado de las ecuaciones de Lagrange_Euler, se trata en muchos libros, por ejemplo véase[14] [15] y tras un desarrollo, la segunda forma general de la ecuación de Lagrange:

$$\tau_j = \frac{d}{dt}\left[\frac{\partial L}{\partial \dot{q}_j}\right] - \frac{\partial L}{\partial q_j} \qquad j=1,2,\ldots n \tag{3.33}$$

Donde: L es el Lagrangiano del sistema (L=K-V).

V es la energía potencial del sistema.

K es la energía cinética del sistema.

$\tau_j = \psi_j + \frac{\partial V}{\partial q_j}$. ψ_j es la fuerza generalizada j-esima, aplicada en la unión j.

No sirven de mucho las ecuaciones de Lagrange-Euler, si no se puede encontrar el Lagrangiano del sistema, por lo que se desarrollan las ecuaciones para tenerlas en función de los parámetros del manipulador.

La expresión general de la energía cinética, de un cuerpo rígido es de la forma:

$$K = \frac{1}{2} m v_c^T v_c + \frac{1}{2} \omega^T I \omega \tag{3.34}$$

Donde: V_c es la velocidad lineal del centroide del cuerpo.

---

[14] Samuel Doughty, *Mechanics of machines*, Estados Unidos, John Wiley & Sons, 1a. ed., 1988, pp. 411-414

[15] Mark W. Spong y M. Vidyasagar, op. cit., pp. 129-135

ω es la velocidad angular del cuerpo.

I es el tensor de inercia del cuerpo y tiene la forma:

$$I = \begin{bmatrix} \int(y^2+z^2)dm & -\int xydm & -\int xzdm \\ -\int xydm & \int(x^2+z^2)dm & -\int yzdm \\ -\int xzdm & -\int yzdm & \int(x^2+y^2)dm \end{bmatrix} \qquad (3.35)$$

En la ecuación (3.34), ω y I están referidos a un marco de referencia, con origen en el centroide del elemento y se mueve con éste, ya que si se hubieran referido a un marco de referencia fijo, fuera del cuerpo, la matriz de inercia (I), se modificaría según los movimientos del elemento, siendo un poco más complejo el análisis.

Las velocidades del centroide, tanto lineal como angular se pueden calcular por medio de un Jacobiano específico para cada elemento.

De forma que las velocidades del elemento son de la forma[16]:

$$\begin{aligned} v_{c_i} &= (J_{v_{c_i}})(\dot{q}) \\ \omega_{i'} &= (R_0^i)^T (J_{\omega_i})(\dot{q}) \end{aligned} \qquad (3.36)$$

Donde $\omega_{i'}$, es la velocidad angular del elemento "i", con respecto al marco de referencia "i'".

La velocidad lineal (v_{c_i}) no requiere de una transformación ya que en la energía cinética, sólo

[16] En el análisis dinámico se requiere de marcos de referencia con orígenes en los centroides de los elementos, de forma que difieren de los utilizados en el análisis cinemático y por tanto se les denomina "i'". Para no complicar el análisis, los dos marcos de referencia ("i'" e "i") tienen una orientación idéntica ($R_0^i = R_0^{i'}$) y tan sólo están desplazados dentro del elemento "i".

se requiere de su magnitud, por lo que no importa en que marco de referencia está expresada. Como la energía cinética total, es la suma de la energía de cada elemento, se tiene que para un manipulador de "n" elementos:

$$K = \frac{1}{2}(\dot{q})^T D(q) \tag{3.37}$$

Donde:

$$D = \sum_{i=1}^{n} [m_i (J_{v_{c_i}})^T (J_{v_{c_i}}) + (J_{\omega_i})^T (R_0^i)(I_i)(R_0^i)^T (J_{\omega_i})$$

D es una matriz (nxn), simétrica y se le llama matriz de inercia.

La energía potencial, se da por:

$$V_i = g^T r_{c_i} m_i \quad , V = \sum_{i=1}^{n} g^T r_{c_i} m_i \tag{3.38}$$

Donde: $g^T = [0,0,-9.81]$

r_{c_i} es la posición del centroide en el elemento "i", expresado en el marco de referencia 0.

El lagrangiano del sistema se puede escribir de la forma:

$$L = K - V = \frac{1}{2}(\dot{q})^T D(q) - V \tag{3.39}$$

Y tras un desarrollo, la ecuación de lagrange queda:

$$\sum_{j=1}^{n} d_{kj}\ddot{q}_j + \sum_{j=1}^{n}\sum_{i=1}^{n} c_{ijk}\dot{q}_i\dot{q}_j + \Phi_k = \tau_k \quad ;k = 1,2,...n \tag{3.40}$$

Donde: d_{kj} es el elemento (k,j) de la matriz D.

$$c_{ijk} = \frac{1}{2}\left[\frac{\partial d_{kj}}{\partial q_i} + \frac{\partial d_{ki}}{\partial q_j} - \frac{\partial d_{ij}}{\partial q_k}\right]$$

$$\Phi_k = \frac{\partial V}{\partial q_k}$$

Que es la ecuación de Lagrange-Euler que se utiliza normalmente; de la ecuación (3.40), se puede ver que es necesario conocer a la matriz D, antes de realizar las operaciones. Para la obtención de la matriz de inercia, véase el apéndice C.

3.2.2 Ecuaciones de Newton-Euler

Con base en la sumatoria de fuerzas y momentos en el centroide de cada elemento y tras un desarrollo, se obtiene un método[17] recursivo que relaciona a las coordenadas generalizadas y sus dos primeras derivadas ($q,\dot{q},\ddot{q}$), con las fuerzas y momentos ejercidos en cada unión (f_i y τ_i).

Primero se realiza una iteración hacia adelante, con las condiciones iniciales de $\omega_0 = 0$, $\alpha_0 = 0$, $a_{e,0} = 0$

Para i=1,2,3...n **(3.41)**

$$(I) \quad \omega_i = (R_{i-1}^{i})^T \omega_{i-1} + (R_0^i)^T Z_{i-1}\dot{q}_i$$

$$(II) \quad \alpha_i = (R_{i-1}^{i})^T \alpha_{i-1} + (R_0^i)^T Z_{i-1}\ddot{q}_i + \omega_i \times [(R_0^i)^T Z_{i-1}\dot{q}_i]$$

$$(III) \quad a_{e,i} = \dot{\omega}_i \times r_{i,i+1} + \omega_i \times (\omega_i \times r_{i,i+1}) + (R_{i-1}^{i})^T a_{e,i-1}$$

$$(IV) \quad a_{c,i} = \dot{\omega}_i \times r_{i,c_i} + \omega_i \times (\omega_i \times r_{i,c_i}) + (R_{i-1}^{i})^T a_{e,i-1}$$

Después se realiza una iteración hacia atrás, con las condiciones finales de $f_{n+1} = 0$, $\tau_{n+1} = 0$.

Para i=n, n-1, n-2,...,1 **(3.42)**

$$(V) \quad f_i = R_i^{i+1} f_{i+1} + m_i a_{c,i} - m_i g_i$$

$$(VI) \quad \tau_i = R_i^{i+1}\tau_{i+1} - f_i \times r_{i,c_i} - (R_i^{i+1} f_{i+1}) \times r_{i+1,c_i} + I_i \alpha_i + \omega_i \times (I_i \omega_i)$$

Donde: $a_{e,i}$ es la aceleración del extremo del elemento "i".

$a_{c,i}$ es la aceleración del centroide del elemento "i".

m_i es la masa del elemento "i".

g_i vector de la fuerza de gravedad.

ω_i, α_i velocidad y aceleración angulares del elemento "i".

---

[17] Mark W. Spong y M. Vidyasagar, op. cit., pp. 153-162.

Para poder aplicar el método, se requiere conocer a:

R_i^{i+1} , $(R_{i-1}^i)^T$, Z_{i-1} , $(R_0^i)^T$; que se obtienen de los resultados parciales del análisis cinemático.

$r_{i,i+1}$, r_{i,c_i} , r_{i+1,c_i} , I_i , m_i , g_i; que están relacionados con la geometría en particular del manipulador.

En realidad, los marcos de referencia para cada elemento, difieren de los utilizados en el análisis cinemático, ya que el origen de los que se está utilizando es el centroide del elemento. Pese a esto, se tiene marcos de referencia con orientaciones idénticas, en los dos análisis y por tanto las matrices de rotación obtenidas en los problemas cinemáticos, sirven para el análisis dinámico.

R_i^{i+1} *para* $i = 1,2,3,\cdots,6 \rightarrow R_1^2,\ R_2^3,\ R_3^4,\ R_4^5,\ R_5^6$

$(R_{i-1}^i)^T$;*para* $i = 1,2,3,\cdots,6 \rightarrow (R_0^1)^T$, $(R_1^2)^T$, $(R_2^3)^T$, $(R_3^4)^T$, $(R_4^5)^T$, $(R_5^6)^T$

Estas matrices se obtienen de las transformaciones A_i (i=1,2,....,6) en la sección 3.1.1

Z_{i-1} ;*para* $i = 1,2,3,\cdots,6 \rightarrow Z_0,\ Z_1,\ Z_2,\ Z_3,\ Z_4,\ Z_5$ son los vectores k_i situados en la unión "i", de acuerdo a la asignación efectuada en el análisis cinemático. Z_{i-1}, se obtuvo en la sección 3.1.5

$(R_0^i)^T$;*para* $i = 1,2,\cdots,6 \rightarrow (R_0^1)^T$, $(R_0^2)^T$, $(R_0^3)^T$, $(R_0^4)^T$, $(R_0^5)^T$, $(R_0^6)^T$ se obtienen de la transformación homogénea (T_0^i ; i=1,2..6) en el apéndice B.

$r_{i,i+1}$ el vector de la unión "i" a la unión "i+1", expresado según el marco de referencia "i", se obtiene de los parámetros cinemáticos.

$$r_{1,2} = [0, d_1, d_2]^T \quad r_{2,3} = [-a_2, 0, 0]^T \quad r_{3,4} = [-a_3, 0, 0]^T$$

$$r_{4,5} = [0, 0, b]^T \quad r_{5,6} = [0, -c, e]^T \quad con\ b,c,e < 1$$

r_{i+1,c_i} el vector de la unión "i+1" al centroide del elemento "i", expresado de acuerdo al marco de referencia "i", está dado en parte por la localización de los centroides (de acuerdo a los marcos de referencia del análisis cinemático) y los parámetros cinemáticos:

$$r_{2,c_1} = OC_1 - d_2(k_1) \quad r_{3,c_2} = OC_2 \quad r_{4,c_3} = OC_3$$

$$r_{5,c_4} = [0, 0, 0]^T \quad r_{6,c_5} = [0, 0, OC_{5Z}]^T$$

r_{i,c_i} el vector de la unión "i" al centroide del elemento "i", se puede expresar como:

$$r_{i,c_i} = r_{i+1,c_i} + r_{i,i+1} \tag{3.44}$$

g_i la aceleración de la gravedad, expresada según el marco de referencia "i", se tiene que:

$$g_i = (R_0^i)^T\, |9.81|(-\hat{k}_0) \tag{3.45}$$

Con lo que se tiene:

$$g_1 = \begin{bmatrix} 0 \\ -9.81 \\ 0 \end{bmatrix} \qquad g_2 = \begin{bmatrix} -9.81\,S_2 \\ -9.81\,C_2 \\ 0 \end{bmatrix} \qquad g_3 = \begin{bmatrix} -9.81\,S_{23} \\ -9.81\,C_{23} \\ 0 \end{bmatrix}$$

$$g_4 = \begin{bmatrix} -9.81\,S_{234} \\ 0 \\ -9.81\,C_{234} \end{bmatrix} \qquad g_5 = \begin{bmatrix} -9.81\,S_{234}C_5 \\ -9.81\,C_{234} \\ -9.81\,S_{234}S_5 \end{bmatrix} \qquad g_6 = \begin{bmatrix} -9.81(S_{234}C_5C_6 + C_{234}S_6) \\ -9.81(C_{234}C_6 - S_{234}C_5S_6) \\ -9.81(S_{234}S_5) \end{bmatrix}$$

I_i y m_i, dependen específicamente de las características del manipulador, y en el apéndice D se encuentran los datos utilizados en la simulación del capítulo 7 (Pruebas).

CAPÍTULO 4

CONTROL

Las tareas que se pueden realizar por medio de un manipulador, implican el movimiento de éste por determinados puntos en el espacio de trabajo y esto sólo es posible por el movimiento controlado de todas las uniones. De forma que en el caso de un manipulador de seis GDL con un motor de CD (como actuador) en cada unión, se tiene que generar seis señales de control (voltaje); no importando si el movimiento global que ejecuta el manipulador es de punto a punto, punto a punto interpolando alguna trayectoria o de trayectoria continua.

Existe una gran cantidad de estrategias[18] para el control de un manipulador, cuya utilización está determinada por el propio manipulador, por el desempeño esperado (velocidad/aceleración con una cierta exactitud, repetibilidad) y por la forma de especificación de las tareas (punto a punto, trayectoria continua, etc).

En el caso específico de la tesis, se optó por un control de posición independiente en cada unión, con compensación anticipativa. Es decir se está tomando al manipulador como un sistema no

[18] Para un panorama general de los diversos tipos de control, véase M. W. Spong y M. Vidyasagar, op. cit.

lineal y altamente acoplado, y por medio de la compensación anticipativa de la dinámica del manipulador, se linealiza y se desacopla al sistema en torno a la trayectoria deseada; de forma que cada unión se maneja por un control lineal e independiente. Tomando como perturbaciones al sistema de seguimiento, los efectos del movimiento de las otras uniones, que no son cancelados por la compensación anticipativa; que en el caso ideal de un seguimiento perfecto, con un modelo dinámico exacto, resulta en perturbaciones constantes y de un valor pequeño, causadas por los efectos no contemplados en las ecuaciones de Lagrange_Euler (fricción, flexibilidades, alinealidades en el actuador/transmisión de potencia, etc). Como la compensación anticipativa se realiza sobre la trayectoria deseada, se puede hacer parte de los cálculos con anterioridad y no estrictamente en tiempo real.

El control con realimentación lineal se utiliza frecuentemente, sobre todo el control independiente en cada unión tipo PD o PID, ya que es común encontrar manipuladores con una alta relación en la transmisión de potencia en cada unión, operando a baja velocidad y con un sobrediseño importante (elementos pesados y rígidos, para la capacidad de carga especificada). Por lo que la aproximación a un sistema lineal es bastante buena y el desempeño de un control PD es muy aceptable. Al tener relaciones de transmisión de potencia bajas o velocidades elevadas, el manipulador se comporta como un sistema no lineal y acoplado, y por tanto el control independiente en cada unión recibe perturbaciones no lineales pero, se sabe que el control lineal es en cierto grado robusto[19] y ante perturbaciones no lineales acotadas, se tiene un error de seguimiento acotado; siempre que se tengan ganancias suficientemente grandes.

[19] H. Berghuis y H. Nijmeijer "Robust control of robots via linear estimated state feedback", en *IEEE Trans. on Automatic Control*, Núm 10, Vol 39, octubre de 1994, pág 2159.

Las ganancias del control están en muchos aspectos limitadas, por un lado prácticamente no es recomendable operar con ganancias extremadamente altas y por el otro, en ciertos casos las anancias están restringidas por requerimientos de operación sobreamortiguada o críticamente amortiguada, por la frecuencia de resonancia del propio manipulador, por la saturación de los actuadores y por el ruido en la realimentación de la velocidad, que es causado por el tacómetro si se utiliza uno o por el método de obtención de la velocidad a partir de la posición. Al tener un límite superior en las ganancias, se tiene un cierto error de seguimiento que es imposible de disminuir y es por eso que se plantean técnicas como la compensación anticipativa; como una forma de disminuir los errores de seguimiento, sin necesariamente recurrir a ganancias altas pero, conservando un esquema de control lineal.

En cuanto al ruido en la señal de velocidad, en muchas referencias[20,21] se opta por no realizar la medición de la velocidad y realimentarla por medio de un observador; evitando los problemas de ruido y el peso (volumen) extra en el manipulador que implican los sensores de velocidad, así como el costo a nivel de sensores y de la adquisición de esas señales.

Por lo que se decidió realizar el control en variables de estado, que tiene un mejor control en la localización de los polos que un control PD, y con un observador lineal de orden mínimo; en el supuesto que la compensación anticipativa linealiza y desacopla efectivamente al sistema.

---

[20] Idem.

[21] C. Canudas de Wit y N Fixot "Adaptive control of robot manipulators via velocity estimated feedback", en *IEEE Trans. on Automatic Control*, Núm 8, Vol 37, agosto de 1992, pág 1234.

Posiblemente un control lineal con compensación anticipativa es el esquema de control, basado en el modelo dinámico, más sencillo que se puede aplicar pero, con ganancias lo suficientemente grandes se tiene un sistema exponencialmente estable[22] y comparable[23,24] en desempeño con el método del par calculado, (uno de los esquemas de control más citados en las referencias bibliográficas) que utiliza una realimentación no lineal, basada en la dinámica del manipulador, para linealizar de forma exacta y desacoplar al sistema.

Hay varias dificultades asociadas a los tipos de control basados en la dinámica del manipulador, principalmente relacionadas con la exactitud del modelo, como el obtener todos los parámetros necesarios para el modelo dinámico y aunado a esto, para los manipuladores con transmisiones de potencia, hay muchos efectos no modelados por las ecuaciones de Lagrange-Euler (capítulo 3) como fricción, inercia, etc de forma que hay efectos no lineales que no son compensados y que introducen perturbaciones a la realimentación. Otro problema se da por la cantidad de cálculos requeridos en la dinámica del manipulador, que impide períodos de muestreo rápidos y por tanto reduce el nivel de las ganancias que se pueden utilizar, con el consecuente error de seguimiento. Ante este problema se trata de minimizar el tiempo utilizado en los cálculos y en ciertas situaciones se simplifica la dinámica, sobre todo en manipuladores con una transmisión de potencia en las uniones, en los que a velocidades no muy elevadas los términos centrífugos y de coriolis son despreciables; pero si se opera a altas velocidades o en un manipulador sin

---

[22] R. Kelly y R. Salgado "PD control with computed feedforward of robot manipulators: A design procedure", en *IEEE Trans. on Robotics and Automation*, Núm. 4, Vol 10, agosto de 1994, pp 566-571.

[23] T. J. Tarn et al. "Performance comparison of four manipulator servo shemes", en *IEEE Control Systems Magazine*, Núm 1, Vol 13, febrero de 1993, pp. 22-29.

[24] C.H. An et al. "Experimental evaluation of feedforward and computed torque control", en *IEEE Trans. on Robotics and Automation*, Núm. 3, Vol. 5, junio de 1989, pp. 368-373.

transmisión de potencia, las simplificaciones en la dinámica sólo producen perturbaciones no lineales al control.

Una simplificación que se hace comúnmente, es el despreciar la inductancia en el motor de CD, con lo que se deja a un lado la dinámica del actuador y se supone que los motores son capaces de variar el par generado de forma instantánea. La simplificación permite manejar un modelo de la unión de segundo grado, en contraparte del modelo de tercer grado que resultaría de tomar en cuenta a la inductancia y que necesitaría de la medición de la aceleración para el control, con el consecuente aumento en la complejidad de los algoritmos de control pero, que ofrecería un mejor comportamiento dando un error de seguimiento menor.

4.1. Desarrollo del control

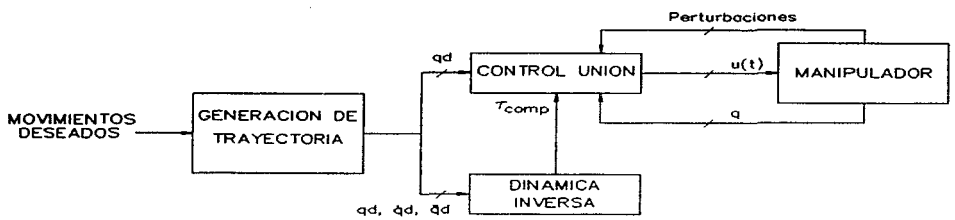


Fig. 4.1 Diagrama general del control.

La figura 4.1 muestra de forma general, el flujo de datos entre las partes que forman el sistema de control. Primeramente el generador de trayectoria, en base a una trayectoria ya definida, realiza todos los cálculos necesarios para generar un comportamiento en el tiempo de cada una de las uniones del manipulador, de forma que se obtienen q_d, $\dot{q}_d$ y $\ddot{q}_d$ (posición, velocidad y aceleración deseadas a nivel de unión). El bloque de generación de trayectoria se describe en el capítulo 6 (programación) pero, en forma general toma una trayectoria definida (una secuencia ordenada en el tiempo de puntos en el espacio de trabajo), de forma que se tiene la posición para cada unión en instantes de tiempo conocidos. Se interpola un polinomio de tercer grado entre cada posición conocida, para que haya continuidad de posición, velocidad y aceleración a lo largo del movimiento.

q_d, $\dot{q}_d$ y $\ddot{q}_d$ forman la trayectoria deseada y es la entrada al bloque de la dinámica inversa, en donde se calcula, por las ecuaciones de Newton_Euler (capítulo 3) el par requerido en cada unión

como compensación anticipativa (τ_{comp}).

El bloque de control de unión, es el control en variables de estado con un observador de orden mínimo y que acepta el par de compensación generado en el bloque de la dinámica inversa, con lo que genera la entrada de control (voltaje) al manipulador; por lo que se tiene seis controladores, uno para cada unión.

El manipulador es de acuerdo a los objetivos de la tesis, un manipulador articulado tipo codo, con motores de CD como actuadores y potenciómetros para realimentar la posición en cada unión. Los parámetros del manipulador, cinemáticos y dinámicos, se listan en el apéndice D.

4.1.1. Desarrollo del modelo de la unión.

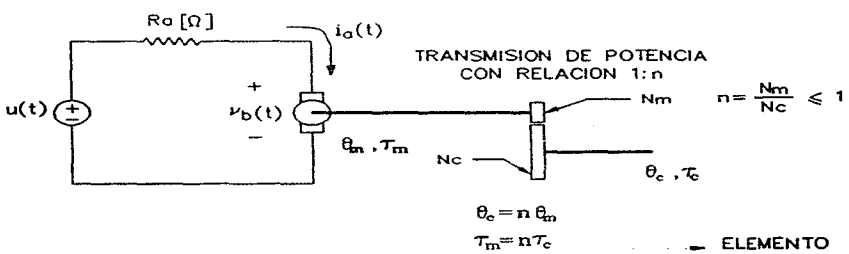


Fig. 4.2 Modelo de la unión.

De acuerdo a la figura 4.2 y despreciando la inductancia del motor:

$$u(t) = R_a\, i_a(t) + v_b(t) \qquad ; v_b(t) = k_b\, \dot{\theta}_m(t) \tag{4.1}$$

Donde: $u(t)$ señal de control (voltaje del motor).

R_a resistencia de la armadura.

$i_a(t)$ corriente de armadura.

$v_b(t)$ voltaje de la fuerza contra-electromotriz.

k_b constante de la fuerza contra-electromotriz.

$\dot{\theta}_m(t)$ velocidad angular del motor.

En el rango lineal de funcionamiento, el motor genera un par proporcional a la corriente de armadura, la cual se determina por la ec. (4.1)

$$\tau = k_a\, i_a(t) = \frac{k_a}{R_a}\left[u(t) - k_b\, \dot{\theta}_m(t)\right] \qquad (4.2)$$

Donde: τ par generado por el motor.

k_a constante par/corriente.

En un sistema rotacional los efectos de inercia y de amortiguamiento son iguales a la sumatoria de par sobre el sistema y realizando el análisis sobre la transmisión de potencia, se tiene:

$$\tau = (J_a + J_m)\,\ddot{\theta}_m(t) + B_m\,\dot{\theta}_m(t) = J_{eff}\ddot{\theta}_m(t) + B_{eff}\dot{\theta}_m(t) \qquad (4.3)$$

Donde: J_a inercia rotacional del motor.

J_m inercia rotacional de la transmisión de potencia, referida al lado del motor.

B_m coeficiente de amortiguamiento de la transmisión, referido al lado del motor.

J_{eff} inercia efectiva.

B_{eff} coeficiente de amortiguamiento relativo.

$\theta_m(t)$ movimiento angular del eje del motor.

Sólo se está considerando los efectos de inercia y de amortiguamiento, del motor y de la transmisión de potencia; en cuanto a los elementos-uniones posteriores aunque también son una carga, su inercia varia de acuerdo a los movimientos del manipulador y si se tomara en cuenta se tendría una ecuación diferencial con coeficientes variantes en el tiempo, por lo que se toma como una perturbación del lado de la carga (elemento). A la ec. (4.2) se agrega un término extra al voltaje de entrada, que está destinado a disminuir los efectos de la perturbación. Igualando (4.2) y (4.3) con los nuevos términos se tiene:

$$\tau = \frac{k_a}{R_a}\left[u(t) + k_R \tau_{comp}(t) - k_b \dot{\theta}_m(t) \right] = J_{eff}\ddot{\theta}_m(t) + B_{eff}\dot{\theta}_m(t) + nP(t) \qquad (4.4)$$

Donde: $k_R \tau_{comp}(t)$ es el término extra al voltaje de entrada y es proporcional por medio de k_R, al par de la compensación anticipativa.

$n\,P(t)$ par producido por los efectos de las demás uniones (perturbación), referido al lado del motor.

$n = (N_m / N_c)$ relación de transformación de la transmisión de potencia. N_m es el número de dientes en el engrane del lado del motor. N_c el número de dientes en el engrane del lado de la carga (elemento).

Para expresar el modelo de la unión en función de la variable de unión (q), se tiene que utilizar a θ_c (figura 4.2). Se sabe que:

$$\theta_c = n\,\theta_m \quad , \quad \dot{\theta}_c = n\,\dot{\theta}_m \quad y \quad \ddot{\theta}_c = n\,\ddot{\theta}_m$$

Sustituyendo en (4.4):

$$J_{eff}\ddot{\theta}_c(t) + (B_{eff} + \frac{k_a k_b}{R_a})\dot{\theta}_c(t) + n^2 P(t) - \frac{n k_a}{R_a}\left[u(t) + k_R \tau_{comp}(t)\right] = 0 \qquad (4.5)$$

Despejando a $\ddot{\theta}_c(t)$:

$$\ddot{\theta}_c(t) = \frac{n k_a}{R_a J_{eff}} u(t) + \frac{n k_a k_R}{R_a J_{eff}} \tau_{comp}(t) - \frac{n^2 P(t)}{J_{eff}} - (\frac{B_{eff}}{J_{eff}} + \frac{k_a k_b}{R_a J_{eff}})\dot{\theta}_c(t) \qquad (4.6)$$

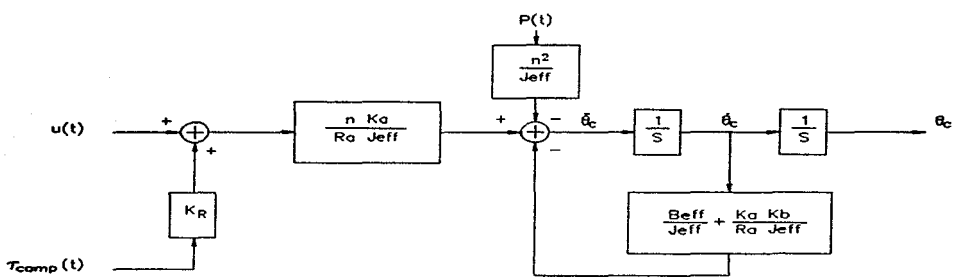


Fig. 4.3 Representación del modelo de la unión.

Como se puede ver en la figura 4.3, al ser la relación de la transmisión de potencia menor que uno, la transmisión de potencia produce una disminución en las perturbaciones.

Expresado el sistema en variables de estado:

$$\dot{X} = \begin{bmatrix} 0 & 1 \\ 0 & -(\frac{B_{eff}}{J_{eff}} + \frac{K_a K_b}{R_a J_{eff}}) \end{bmatrix} X + \begin{bmatrix} 0 \\ \frac{nK_a}{R_a J_{eff}} \end{bmatrix} u(t) - \begin{bmatrix} 0 \\ \frac{n^2}{J_{eff}} \end{bmatrix} P(t) + \begin{bmatrix} 0 \\ \frac{nK_a K_R}{R_a J_{eff}} \end{bmatrix} T_{comp}(t) \qquad (4.7)$$

$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} X \quad ; X = \begin{bmatrix} \theta_c \\ \dot{\theta}_c \end{bmatrix}$$

Si la compensación anticipativa tiene la forma de:

$$T_{comp}(t) = \tau_{dinamico} \quad ;con \quad k_R = \frac{nR_a}{K_a} \left[\frac{V}{N \cdot m}\right] \qquad (4.8)$$

Se puede compensar gran parte de las perturbaciones, que en su mayoría son debidas al comportamiento dinámico del manipulador. Las ecuación en variables de estado entonces, quedan de la forma:

$$\dot{X} = \begin{bmatrix} 0 & 1 \\ 0 & -(\frac{B_{eff}}{J_{eff}} + \frac{K_a K_b}{R_a J_{eff}}) \end{bmatrix} X + \begin{bmatrix} 0 \\ \frac{nK_a}{R_a J_{eff}} \end{bmatrix} u(t) - \begin{bmatrix} 0 \\ \frac{n^2}{J_{eff}} \end{bmatrix} w(t) \qquad (4.9)$$

Donde w(t), son las perturbaciones que no se pueden cancelar por la compensación anticipativa y que para un seguimiento perfecto se suponen constantes. El sistema en variables de estado se expresa, de manera general, como:

$$\dot{X}(t) = A\,X(t) + B\,u(t) + B_p\,w(t)$$
$$y(t) = C\,X(t) \qquad (4.10)$$

4.1.2. Discretización del sistema expresado en variables de estado.

Utilizando el equivalente discreto de retén de orden cero, u(t) constante durante el período de muestreo (T). El sistema tiene la forma de:

$$X(k+1) = G(T)X(k) + H(T)u(k) + H_p(T)w(k)$$
$$y(k) = C\ X(k) \qquad ; C = constante \tag{4.11}$$

(por sencillez en la notación, se omite que es función de kT o de (k+1)T y se simplifica a "k" y "k+1").

Donde[25]: $G(T) = e^{A\,T} = \mathcal{L}^{-1}\{(sI - A)^{-1}\}$

$$H(T) = (\int_0^T e^{A\,\lambda}\, d\lambda)B$$

$$H_p(T) = (\int_0^T e^{A\,\lambda}\, d\lambda)B_p$$;ya que w(t) se considera constante.

Desarrollando[26] las matrices , la ecuación queda:

$$e^{A\,T} = \mathcal{L}^{-1}\{(sI - A)^{-1}\} = \begin{bmatrix} 1 & \frac{1}{\alpha}(1 - e^{-\alpha T}) \\ 0 & e^{-\alpha T} \end{bmatrix} \quad ; \ \alpha = \frac{B_{eff}}{J_{eff}} + \frac{k_a k_b}{R_a J_{eff}} \tag{4.12}$$

---

[25] K. Ogata, *Discrete-time control systems*, USA, Prentice_Hall, 1987, pp. 538-540.

[26] Ibid pp. 601-603.

$$\mathbf{H} = (\int_0^T e^{\mathbf{A}\lambda} d\lambda)\mathbf{B} = \int_0^T \begin{bmatrix} 1 & \frac{1}{\alpha}(1 - e^{-\alpha\lambda}) \\ 0 & e^{-\alpha\lambda} \end{bmatrix} \begin{bmatrix} 0 \\ \beta \end{bmatrix} \quad ; \ \beta = \frac{n\, k_a}{J_{eff} R_a}$$

$$\mathbf{H} = \begin{bmatrix} \frac{\beta}{\alpha}(T + \frac{e^{-\alpha T} - 1}{\alpha}) \\ \frac{\beta}{\alpha}(1 - e^{-\alpha T}) \end{bmatrix} \tag{4.13}$$

$$\mathbf{H}_p = (\int_0^T e^{\mathbf{A}\lambda}\, d\lambda)\mathbf{B}_p = \begin{bmatrix} \frac{-n^2}{\alpha J_{eff}}(T + \frac{e^{-\alpha T} - 1}{\alpha}) \\ \frac{-n^2}{\alpha J_{eff}}(1 - e^{-\alpha T}) \end{bmatrix} \tag{4.14}$$

De forma que la ecuación en variable de estado es:

$$\mathbf{X}(k+1) = \begin{bmatrix} 1 & \frac{1}{\alpha}(1-e^{-\alpha T}) \\ 0 & e^{-\alpha T} \end{bmatrix} \mathbf{X}(k) + \begin{bmatrix} \frac{\beta}{\alpha}(T+\frac{e^{-\alpha T}-1}{\alpha}) \\ \frac{\beta}{\alpha}(1-e^{\alpha T}) \end{bmatrix} \mathbf{u}(k) - \begin{bmatrix} \frac{n^2}{\alpha J_{eff}}(T+\frac{e^{-\alpha T}-1}{\alpha}) \\ \frac{n^2}{\alpha J_{eff}}(1-e^{\alpha T}) \end{bmatrix} \mathbf{w}(k)$$

$$\mathbf{y}(k) = \begin{bmatrix} 1 & 0 \end{bmatrix} \mathbf{X}(k) \tag{4.15}$$

Controlabilidad:

Si el rango de [H | GH]=2, entonces el sistema es controlable, o por medio de la obtención

de los valores propios y vectores propios asociados, se puede confirmar la controlabilidad del sistema.

Los valores propios de G son:

$$\det[G - \lambda I] = \det\begin{bmatrix} 1-\lambda & \frac{1}{\alpha}(1-e^{-\alpha T}) \\ 0 & e^{-\alpha T} - \lambda \end{bmatrix} = 0 \qquad (4.16)$$

$$\lambda_1 = 1 \qquad \lambda_2 = e^{-\alpha T}$$

Los vectores propios asociados a cada valor propio se obtienen de:

$$(G - \lambda_i I)x_i = 0 \quad ; \; x_i - vector\,propio\,asociado\,a\;\lambda_i$$

Para $\lambda_1 = 1$

$$(G - (1)I)x_1 = \begin{bmatrix} 0 & \frac{1}{\alpha}(1 - e^{-\alpha T}) \\ 0 & e^{-\alpha T} - 1 \end{bmatrix} x_1 = 0 \quad ; \; x_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \qquad (4.17)$$

Para $\lambda_2 = e^{-\alpha T}$

$$(G - (e^{-\alpha T})I)x_2 = \begin{bmatrix} 1 - e^{-\alpha T} & \frac{1}{\alpha}(1 - e^{-\alpha T}) \\ 0 & 0 \end{bmatrix} x_2 = 0 \quad ; \; x_2 = \begin{bmatrix} -\frac{1}{\alpha} \\ 1 \end{bmatrix} \qquad (4.18)$$

Definiendo a $P=[x_1 | x_2]$, se tiene que:

$$P = \begin{bmatrix} 1 & -\frac{1}{\alpha} \\ 0 & 1 \end{bmatrix} \quad y \quad P^{-1} = \begin{bmatrix} 1 & \frac{1}{\alpha} \\ 0 & 1 \end{bmatrix}$$

Y se cumple que: $P\,G\,P^{-1} = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$

Si el sistema es controlable, se debe de cumplir que $P^{-1}H$ no tenga renglones con todos los elementos cero:

$$P^{-1}H = \begin{bmatrix} 1 & \frac{1}{\alpha} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{\beta}{\alpha}(T + \frac{e^{-\alpha T} - 1}{\alpha}) \\ \frac{\beta}{\alpha}(1 - e^{-\alpha T}) \end{bmatrix} = \begin{bmatrix} \frac{\beta}{\alpha}(T) \\ \frac{\beta}{\alpha}(1 - e^{-\alpha T}) \end{bmatrix} \qquad (4.19)$$

Como α , β y T son diferentes de cero, se cumple que no hay renglones con todos los elementos cero; de forma que el sistema es controlable.

Observabilidad:

Por medio del rango de la matriz de observabilidad:

$$\begin{bmatrix} C^T \mid G^T C^T \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & \frac{1}{\alpha}(1 - e^{-\alpha T}) \end{bmatrix} \quad ; \textit{que es de rango } 2. \qquad (4.20)$$

O por medio de los valores y vectores propios, CP no debe de tener columnas con todos los elementos cero:

$$CP = [1 \quad 0] \begin{bmatrix} 1 & -\frac{1}{\alpha} \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & -\frac{1}{\alpha} \end{bmatrix} \qquad (4.21)$$

Por lo que se tiene que el sistema es observable.

4.1.3 Control y observador

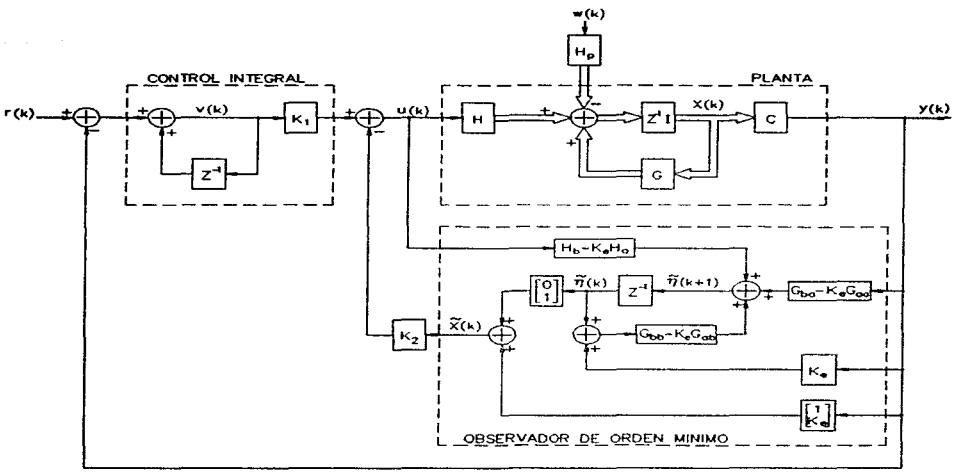


Fig. 4.4 Sistema de control.

Se requiere que la salida y(k) siga a una referencia , por lo que se establece el sistema de seguimiento de la figura 4.4; en donde se aumenta un integrador a la señal de error, para disminuir el error en estado estable. En la misma figura aparece la estructura del observador de orden mínimo.

El valor de las ganancias del integrador (K_1), de la realimentación del estado (K_2) y del observador (K_e); están especificadas por el comportamiento deseado del sistema. De un inicio se descarta un comportamiento sobreamortiguado, ya que se busca el sistema responda los más

rápido posible; involucrando un comportamiento críticamente amortiguado o subamortiguado, a depender del error de seguimiento que se permita.

El sistema sin realimentar tiene la ecuación característica:

$$|ZI - G| = \begin{vmatrix} Z-1 & -\frac{1}{\alpha}(1-e^{-\alpha T}) \\ 0 & Z-e^{-\alpha T} \end{vmatrix} = 0 \qquad (4.22)$$

$$(Z-1)(Z-e^{-\alpha T}) = Z^2 - (1+e^{-\alpha T})Z + e^{-\alpha T} = 0$$

Y los polos del sistema no realimentado, son $Z = 1$ y $Z = e^{-\alpha T}$.

El comportamiento deseado se expresa por el coeficiente de amortiguamiento relativo (ζ) y por la frecuencia natural del sistema (ω_n). Como se quiere un comportamiento de críticamente amortiguado a subamortiguado, se tienen que $\zeta \leq 1$ y el sistema de segundo orden, en el dominio de la frecuencia "s", se expresa como:

$$s^2 + 2\zeta\omega_n s + \omega_n^2 = 0 \qquad (4.23)$$

Con los polos dados por:

$$s = -\zeta\omega_n \pm j\omega_n\sqrt{1-\zeta^2} \quad ; \ \zeta \leq 1 \qquad (4.24)$$

En Z, se tiene que los polos están dados por $Z = e^{sT}$, es decir:

$$Z = e^{-\zeta\omega_n T}\, e^{\pm j\omega_n T\sqrt{1-\zeta^2}} = r\,e^{\pm j\theta} \quad ; \ r = e^{-\zeta\omega_n T}, \quad \theta = \omega_n T\sqrt{1-\zeta^2} \qquad (4.25)$$

La frecuencia natural (ω_n) tiene varias restricciones:

- ω_s (frecuencia de muestreo en rad/s) debe de ser por lo menos dos a cuatro veces mayor

que ω_n, para que no ocurran problemas de muestreo y algunas veces se recomienda hasta quince a cincuenta veces mayor.

- ω_n no puede ser mayor que la mitad de la frecuencia de resonancia (ω_r) de la estructura mecánica, para no crear oscilaciones en movimientos rápidos.

El control integral hace que se aumente un estado, v(k), por lo que se tiene que escoger un polo extra de forma que no altere la dinámica deseada para el sistema; lo que implica que el polo del control integral debe de ser más rápido que los polos dominantes, es decir debe de estar a la izquierda de los polos del sistema, en el semiplano izquierdo en "S" o en "Z" más cercano al origen del círculo unitario. Se toma al polo del integrador como γ veces más rápido que los polos dominantes, de forma que tiene una frecuencia natural de $\gamma\,\omega_n$.

Para el observador, se requiere la localización de un polo que debe de ser aun más rápido que el del integrador. Por el principio de separación se puede diseñar el sistema y después el observador, por lo que se verá la localización del polo del observador después.

Se quiere una ecuación característica de la forma:

$$(Z - re^{j\theta})(Z - e^{-j\theta})(Z - e^{-\gamma\zeta\omega_n T}) = Z^3 + \alpha_1 Z^2 + \alpha_2 Z + \alpha_3 \qquad \textbf{(4.26)}$$

Donde:

$$\alpha_1 = -(e^{-\gamma\zeta\omega_n T} + 2e^{-\zeta\omega_n T}\cos(\omega_n T\sqrt{1-\zeta^2}))$$

$$\alpha_2 = e^{-2\zeta\omega_n T} + 2e^{-\zeta\omega_n T(1+\gamma)}\cos(\omega_n T\sqrt{1-\zeta^2})$$

$$\alpha_3 = -e^{-\zeta\omega_n T(2+\gamma)}$$

Para encontrar[27] las ganancias (K_1 y K_2) el sistema formado por el estado del sistema, x(k), y por el estado del integrador, v(k), se expresan como un vector de estados formado por x(k) y u(k), para que la ecuación característica no dependa de r(k) (referencia al servosistema). Para aplicar directamente el método de localización de polos, en lugar del vector de estados x(k), u(k); se utiliza el vector de error con respecto al estado estable, con lo que se elimina el término r(k) de las ecuaciones del sistema, quedando de la forma:

$$\xi(k+1) = \hat{G}\,\xi(k) + \hat{H}\,W(k) \qquad ;\xi(k) = \begin{bmatrix} X_e(k) \\ u_e(k) \end{bmatrix} \quad ; \quad \begin{matrix} X_e(k) = X(k) - X(\infty) \\ u_e(k) = u(k) - u(\infty) \end{matrix} \tag{4.27}$$

Donde:

$$\hat{G} = \begin{bmatrix} G & H \\ 0 & 0 \end{bmatrix} \qquad \hat{H} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

$$W(k) = -K\,\xi(k)$$

$$= -[K_3 - K_2 G - K_1 CG \mid 1 - K_2 H - K_1 CH]\,\xi(k)$$

$X_e(k)$ y $u_e(k)$ son los vectores de error con respecto al estado estable.

El sistema expresado de esta forma, sin realimentar, tiene una ecuación característica:

$$\left| ZI - \hat{G} \right| = z^3 - (1 - e^{-\alpha T})z^2 + e^{-\alpha T} z = 0 \tag{4.28}$$

$$= z^3 + a_1 z^2 + a_2 z + a_3$$

Donde:

$$a_1 = -(1 + e^{-\alpha T})$$

$$a_2 = e^{-\alpha T}$$

$$a_3 = 0$$

[27] Ibid pp. 728-733.

Y se quiere llegar, por medio de la realimentación, a la ecuación característica dada en la ecuación **(4.26)**:

$$Z^3 + \alpha_1 Z^2 + \alpha_2 Z + \alpha_3 = 0$$

Con:

$$\alpha_1 = -(e^{-\gamma\zeta\omega_n T} + 2e^{-\zeta\omega_n T}\cos(\omega_n T\sqrt{1-\zeta^2}))$$

$$\alpha_2 = e^{-2\zeta\omega_n T} + 2e^{-\zeta\omega_n T(1+\gamma)}\cos(\omega_n T\sqrt{1-\zeta^2})$$

$$\alpha_3 = -e^{-\zeta\omega_n T(2+\gamma)}$$

De la técnica de colocación de polos, por la transformación a la forma canónica controlable se tiene:

$$\hat{K} = [\alpha_3 - a_3 | \alpha_2 - a_2 | \alpha_1 - a_1]\,T^{-1} \qquad \textbf{(4.29)}$$

Donde[28]:

$$T = M\,W \quad ; \; M = [\hat{H}|\hat{G}\hat{H}|\hat{G}^2\hat{H}]$$

$$; \; W = \begin{bmatrix} a_2 & a_1 & 1 \\ a_1 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

Pero como en realidad se quiere obtener a K_1 y K_2, se necesita:

$$[K_2|K_1] = \{\hat{K} + [0\;0\;1]\} \begin{bmatrix} G - I_2 & H \\ CG & CH \end{bmatrix}^{-1} \qquad \textbf{(4.30)}$$

---

[28] T es una transformación tal que:

$$T^{-1}\hat{G}\,T = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -a_3 & -a_2 & -a_1 \end{bmatrix}$$

Sustituyendo valores y realizando operaciones:

$$\mathbf{M} = \begin{bmatrix} 0 & \frac{\beta}{\alpha}(T + \frac{e^{-\alpha T}-1}{\alpha}) & \frac{\beta}{\alpha}(T - (\frac{1-e^{-\alpha T}}{\alpha}) + (\frac{1-e^{-\alpha T}}{\alpha})^2) \\ 0 & \frac{\beta}{\alpha}(1-e^{-\alpha T}) & \frac{\beta}{\alpha}e^{-\alpha T}(1-e^{-\alpha T}) \\ 1 & 0 & 0 \end{bmatrix} \tag{4.31}$$

$$\mathbf{W} = \begin{bmatrix} e^{-\alpha T} & -(1+e^{-\alpha T}) & 1 \\ -(1+e^{-\alpha T}) & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \tag{4.32}$$

$$\mathbf{T} = \begin{bmatrix} \frac{\beta}{\alpha}(-e^{-\alpha T}T + (\frac{1-e^{-\alpha T}}{\alpha})) & \frac{\beta}{\alpha}(T + (\frac{e^{-\alpha T}-1}{\alpha}) & 0 \\ \frac{\beta}{\alpha}(e^{-\alpha T}-1) & \frac{\beta}{\alpha}(1-e^{-\alpha T}) & 0 \\ e^{-\alpha T} & -(1+e^{-\alpha T}) & 1 \end{bmatrix} \tag{4.33}$$

$$\mathbf{T}^{-1} = \frac{1}{\beta T(1-e^{-\alpha T})^2} \begin{bmatrix} \alpha(1-e^{-\alpha T}) & 1-e^{-\alpha T}-\alpha T & 0 \\ \alpha(1-e^{-\alpha T}) & 1-e^{-\alpha T}-(\alpha T)e^{-\alpha T} & 0 \\ \alpha(1-e^{-\alpha T}) & 1-e^{\alpha T}-(\alpha T)e^{-2\alpha T} & \beta T(1-e^{-\alpha T})^2 \end{bmatrix} \tag{4.34}$$

$$\hat{\mathbf{k}} = \begin{bmatrix} e^{-\zeta\omega_n T(2+\gamma)} \\ e^{-2\zeta\omega_n T} - e^{-\alpha T} + 2e^{-\zeta\omega_n T(1+\gamma)}\cos(\omega_n T\sqrt{1-\zeta^2}) \\ 1 + e^{-\alpha T} - e^{-\gamma\zeta\omega_n T} - 2e^{-\zeta\omega_n T}\cos(\omega_n T\sqrt{1-\zeta^2}) \end{bmatrix}^T \mathbf{T}^{-1} \tag{4.35}$$

Al tener $\hat{K}$, se utiliza la ec. (4.30) y se obtiene K_1 y K_2.

Donde:

$$\begin{bmatrix} G - I_2 & H \\ CG & CH \end{bmatrix}^{-1} = \begin{bmatrix} -1 & 0 & 1 \\ \frac{1}{T} & \left(\frac{1}{\alpha T} - \frac{1}{1-e^{-\alpha T}}\right) & 0 \\ \frac{\alpha}{\beta T} & \frac{1}{\beta T} & 0 \end{bmatrix} \qquad (4.36)$$

Al realizar la sustitución de las matrices y sus valores, (ver el apéndice D, para los parámetros requeridos) y tras realizar las operaciones se obtiene para un ζ, ω_n, γ y T el valor de las ganancias K_1 y K_2. En el capítulo 7 (pruebas), se establecen valores y se obtienen las ganancias correspondientes, que se utilizan en la simulación del sistema.

Observador de orden mínimo:

Como se puede medir la salida y(k) y esta es un estado del sistema, sólo se requiere observar un estado (X_2). Por el principio de separación, se diseñó de forma independiente el observador del controlador y tan sólo se toman en cuenta los efectos de los dos.

Se tiene que el estado del sistema está dado por:

$$X(k) = \begin{bmatrix} \theta_c(k) \\ \dot{\theta}_c(k) \end{bmatrix} = \begin{bmatrix} X_a(k) \\ \hline X_b(k) \end{bmatrix} \quad ; \; \theta_c(k) = y(k) \qquad (4.37)$$

De forma que el estado del sistema se divide en dos:

-$X_a(k)$ es la parte que puede ser medida directamente.

-$X_b(k)$ es la parte que no puede ser medida directamente.

Al dividir el estado, también se dividen las ecuaciones del sistema realimentado:

$$\left[\begin{array}{c} X_a(k+1) \\ \hline X_b(k+1) \end{array}\right] = \left[\begin{array}{c|c} G_{aa} & G_{ab} \\ \hline G_{ba} & G_{bb} \end{array}\right] \left[\begin{array}{c} X_a(k) \\ \hline X_b(k) \end{array}\right] + \left[\begin{array}{c} H_a \\ \hline H_b \end{array}\right] u(k) \qquad (4.38)$$

$$y(k) = [1 \mid 0] \left[\begin{array}{c} X_a(k) \\ \hline X_b(k) \end{array}\right] \quad ; \; y(k) = X_a(k)$$

De las ecuaciones del sistema, se tiene:

-La ecuación de estado del observador.

$$X_b(k+1) = G_{ba} X_a(k) + G_{bb} X_b(k) + H_b u(k) \qquad (4.39)$$

-La ecuación de salida.

$$X_a(k+1) - G_{aa} X_a(k) - H_a u(k) = G_{ab} X_b(k) \qquad (4.40)$$

Utilizando la ecuación del observador del estado completo[29]:

$$\tilde{X}(k+1) = (G - K_e C)\tilde{X}(k) + H u(k) + K_e y(k) \qquad (4.41)$$

Donde: $\tilde{X}(k+1)$ es el estado estimado (observado).

[29] K. Ogata, op. cit., pág. 719.

Se llega a:

$$\tilde{X}_b(k+1) = (G_{bb} - K_e G_{ab})\tilde{X}_b(k) + G_{ba}X_a(k) + H_b u(k) + K_e [X_a(k+1) - G_{aa} X_a(k) - H_a u(k)] \quad (4.42)$$

Restando (4.42) de (4.39) y tras un desarrollo:

$$X_b(k+1) - \tilde{X}_b(K+1) = (G_{bb} - K_e G_{ab})[X_b(k) - \tilde{X}_b(k)]$$
$$e(k+1) = (G_{bb} - K_e G_{ab})e(k) \quad (4.43)$$

La ecuación de error resultante tiene como ecuación característica:

$$ZI - G_{bb} + K_e G_{ab} = Z - G_{bb} + K_e G_{ab} = 0 \quad (4.44)$$

En (4.42), como $\tilde{X}_b(k+1)$ depende entre otras cosas de $x_a(k+1)$ que es $y(k+1)$, se realiza una transformación:

$$\tilde{X}_b(k) = \tilde{\eta}(k) + k_e X_a(k) \quad (4.45)$$

Se sustituye en (4.42) y también se sustituye $y(k) = X_a(k)$:

$$\tilde{\eta}(k+1) = (G_{bb} - K_e G_{ab})\tilde{\eta}(k) + [(G_{bb} - K_e G_{ab})K_e + G_{ba} - K_e G_{aa}]y(k) + (H_b - K_e H_a)u(k) \quad (4.46)$$

Que depende únicamente del estado anterior del observador, de la salida de la planta, y(k), y de la señal de control u(k). De acuerdo a esto la realimentación del sistema tiene la forma:

$$-K_2 \tilde{X}(k) = -K_2 \begin{bmatrix} X_a(k) \\ \tilde{X}_b(k) \end{bmatrix} = -K_2 \begin{bmatrix} K_a(k) \\ \tilde{\eta}(k) + K_e X_a(k) \end{bmatrix} \quad (4.47)$$

La ecuación característica del observador está dada en **(4.44)** y sustituyendo:

$$G_{aa} = 1 \qquad G_{ba} = 0 \qquad H_a = \frac{\beta}{\alpha}(T + \frac{e^{-\alpha T} - 1}{\alpha})$$

$$G_{ab} = \frac{1}{\alpha}(1 - e^{-\alpha T}) \qquad G_{bb} = e^{-\alpha T} \qquad H_b = \frac{\beta}{\alpha}(1 - e^{-\alpha T}) \tag{4.48}$$

La ganancia del observador se obtiene de:

$$K_e = \frac{\alpha(e^{-\alpha T} - Z_{ob})}{(1 - e^{-\alpha T})} \tag{4.49}$$

En donde Z_{ob} es la posición deseada del polo del observador.

4.1.4) Implementación de las ecuaciones de control.

Juntando el comportamiento dinámico de la planta, del control integral y del observador se tiene una dinámica dada por:

$$\begin{bmatrix} X(k+1) \\ v(k+1) \\ e(k+1) \end{bmatrix} = \begin{bmatrix} G - HK_2 & HK_1 & HK_2\Gamma \\ CHK_2 - CG & 1 - CHK_1 & -CHK_2\Gamma \\ 0 & 0 & G_{bb} - K_e G_{ab} \end{bmatrix} \begin{bmatrix} X(k) \\ v(k) \\ e(k) \end{bmatrix} + \begin{bmatrix} H_p \\ -CH_p \\ 0 \end{bmatrix} w(k) + \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} r(k+1) \tag{4.50}$$

Donde : $\Gamma = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$

Y la ecuación característica de todo el sistema es:

$$\left| ZI_3 - \begin{bmatrix} G - HK_2 & HK_1 \\ CHK_2 - CG & 1 - CHK_1 \end{bmatrix} \right| (Z - G_{bb} + K_e G_{ab}) = 0 \tag{4.51}$$

La ec. **(4.50)** es utilizada en la simulación del capítulo 7 (pruebas). Las ecuaciones de control, que se implementan en la arquitectura son una fracción de las ecuaciones utilizadas en la simulación y se muestran en la figura 4.5. Y tiene la siguiente estructura:

(Entradas: y(k), r(k), k)

Inicio: Si k==0 entonces: $\bar{\eta}(k) = 0.5$; v(k-1)=0;

si no: v(k-1)=temporal1;
$\bar{\eta}(k)$ = temporal2;

$$\bar{X}(k) = \begin{bmatrix} y(k) \\ \bar{\eta}(k) + K_e\, y(k) \end{bmatrix};$$

v(k)= r(k) - y(k) + v(k-1);

$u(k) = K_1\, v(k) - K_2\, \bar{X}(k) + K_R\, r_{comp}$; (salida de control)

temporal1= v(k);
temporal2= $\bar{\eta}(k)$ = ecuación **(4.46)**;
k= k+1;

Ir a inicio;

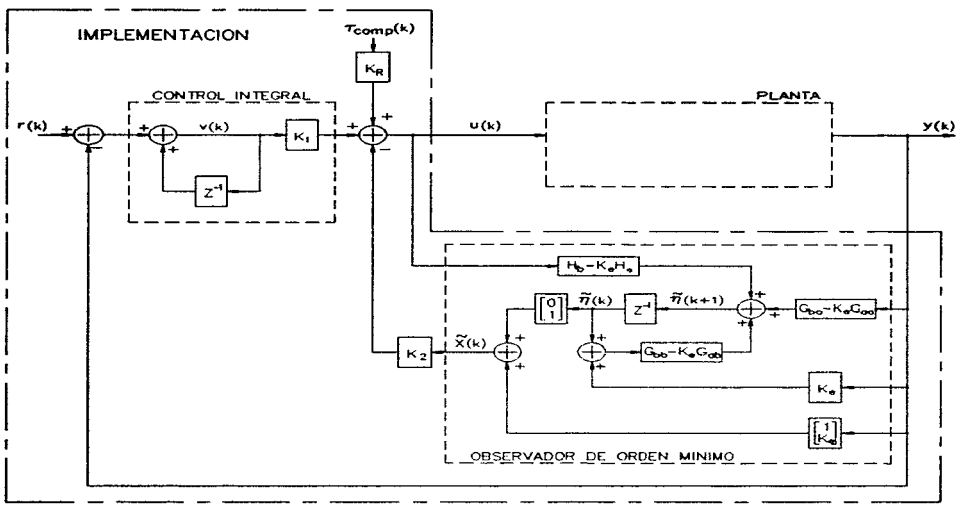


Fig. 4.5 Implementación del sistema de control.

CAPÍTULO 5

ARQUITECTURA

Los controladores utilizados en los manipuladores van desde el caso más sencillo, en el que se utiliza un control de parámetros fijos, sin tomar en cuenta a la dinámica y por tanto no se exige mucho sobre la arquitectura del propio controlador; hasta los casos más complejos con requerimientos altos sobre la arquitectura, en donde se tiene una estrategia de control sofisticada, una movilidad de alto desempeño, un robot de diseño avanzado con muchos GDL o un sistema complejo de sensores. El diseño de la arquitectura parte de especificaciones muy concretas a la arquitectura mínima, con la que se puede realizar el control de un manipulador: tomando en cuenta el tipo de manipulador, el equipo adicional (si es que existe), el tipo de control, la forma de especificar las tareas, los diferentes tipo de sensores, etc. Si se toma al controlador como un sistema, las especificaciones a la arquitectura mínima se pueden centrar en entradas, salidas y un proceso interno; de forma que se tienen especificaciones para la comunicación con el usuario, la interfase con el manipulador y la capacidad de procesamiento (número de operaciones aritméticas/lógicas por tiempo).

De acuerdo a que tan complejo sea el funcionamiento del manipulador, se van a tener ciertos requerimientos sobre la capacidad de procesamiento y sobre todo el tener que realizar varias funciones en tiempo real. También la cantidad de equipo externo que tenga que funcionar junto con el manipulador, va a crear necesidades de coordinación, de fusión de sensores (hardware de interconexión, software de manejo), etc. La interfase con el usuario, va a estar determinada en gran medida por la operación del manipulador, lo que hace que pueda ser sumamente sencilla o todo un sistema sofisticado (que necesita de un procesamiento elevado) en el que se pueda realizar una diversidad de acciones y funciones.

Capacidad de proceso

Los requerimientos de procesamiento están dados por los bloques operativos del sistema, como la interfase con el usuario, el algoritmo de control, la generación de trayectoria, el cálculo de la dinámica, etc: sobre todo los que tiene un tiempo de ejecución determinado. Una vez determinados los requerimientos de procesamiento, se puede elegir una configuración en específico de la arquitectura: por ejemplo, en el caso de tener un manipulador redundante (7 GDL) el cálculo de la cinemática inversa se vuelve un problema complejo y más aun considerando que el jacobiano del manipulador deja de ser una matriz cuadrada. Una posible implementación[30] de un controlador que soporte estos cálculos, tiene una arquitectura basada en cuatro procesadores digitales de señales (DSP), trabajando en paralelo, con capacidad de intercomunicación entre ellos y todos conectados a una estación de trabajo, a forma de base de todo el sistema. En conjunto dan un período de muestreo de 3 ms para las ecuaciones cinemáticas

[30] A. A. Maciejewski y J. M. Reagin "A parallel algorithm and architecture for the control of kinematically redundant manipulators" en *IEEE Trans. on Robotics and Automation*, núm. 4, vol. 10, agosto de 1989, pp. 405-414.

inversas (escoger la θ_i y $\dot{\theta}_i$ más adecuadas para una posición y velocidad deseadas del actuador final).

Muchas veces los objetivos referentes al comportamiento del manipulador, exigen una arquitectura más dedicada, como la requerida para resolver los problemas causados por las posiciones singulares; como se vio en el capítulo 2, en esas posiciones el jacobiano se vuelve singular y por tanto no se puede invertir, lo que implica un desajuste grave en el movimiento del manipulador.
El propio controlador[31] del manipulador no puede realizar los cálculos y se requiere de un aumento en la capacidad de procesamiento, que se da por un procesador extra que se comunica por medio de una memoria compartida con el controlador del manipulador, con un período de muestreo de 12 ms. Para aumentar el poder del conjunto, el procesador extra se conecta a su vez a una estación de trabajo en donde se realiza la inicialización de todo el sistema, la carga de todos los programas para la cinemática inversa y el análisis de las señales generadas por las demás partes del sistema.

En otra aproximación[32] al control de un manipulador de seis GDL, se utilizan varias parejas de microprocesadores/DSP unidas a un bus común. En cada pareja de μp/DSP se realiza una función especifica, como el cálculo de la dinámica inversa, el control de las uniones, la administración del sistema, etc. Se tiene una computadora personal como interfase al usuario y para el cálculo fuera de línea de la trayectoria; el sistema está construido de forma que pueda crecer, al aumentar

[31] S. Chiaverini et al. "Review of the damped least squares inverse kinematics with experiments on an industrial robot manipulator", en *IEEE Trans. on Control System Technology*, núm 2, vol. 2, junio de 1994, pp. 123-134.

[32] W. Leonard, "Trajectory control of a multi-axes robot with electrical servodrives", en *IEEE Control Systems Magazine*, núm 6, vol. 10, octubre de 1990, pp. 3-9.

componentes al bus común. Para una implementación que da un control PI de parámetros fijos a cada unión y un cálculo de la dinámica inversa simplificado a sólo las tres primeras uniones, se logra un período de muestreo de 1.25 ms.

Interfase manipulador

La adquisición y el acondicionamiento de las señales del manipulador, va a depender de que tantas variables se midan por unión (posición, velocidad, par, etc) y del tipo de sensores que se utilicen. De igual forma la generación de la señal de control a cada unión, va a estar determinada por el tipo de actuador utilizado. Si se tiene sensores externos o equipo adicional al manipulador, se puede tener una gran complejidad de procesos, señales y hardware de interconexión, que puede ser tan sencillo como entradas y salidas digitales por medio de un puerto de control, a la necesidad de variables analógicas e inclusive el pensar en un bus de interconexión entre equipos; de forma que si es mucha la información que se comunica entre partes ésta se haga de la forma más rápida, más confiable y con bajos requerimientos en hardware.

Comunicación con el usuario

Por lo general se requiere de hardware específico, ya sea a nivel de un control manual *(teach pendant)*, de un panel de control diseñado en especial o de un monitor capaz de representar lo necesario para operar al manipulador. Hay varias formas en que el usuario puede indicar la tarea ha realizar con el manipulador:

1) Enseñanza manual de los movimientos, en la que se graban las posiciones importantes por medio del propio manipulador; y en la ejecución del movimiento, el manipulador se mueve por los puntos grabados ya sea interpolando trayectorias lineales, circulares o de un polinomio

de mayor grado, para asegurar movimientos suaves.

2) Por medio de un lenguaje de alto nivel, que puede ser:

-Explícito, en el que se tiene que dar todos los movimientos deseados, en forma de un programa que al ser ejecutado, hace que se desarrollen los movimientos prescritos.

-Implícito, en el que se da la tarea ha realizar y el lenguaje la traduce a los movimientos requeridos.

Dentro del lenguaje de alto nivel, está descrita la comunicación con los sensores externos, de forma que se tiene la flexibilidad de alterar los movimientos, al tener en el programa decisiones sobre el estado de los sensores.

En la mayoría de los casos la comunicación con el usuario, es muy específica al manipulador/controlador y a la tarea ha realizar; si se quiere una interfase mucho más general, que no sea específica a un manipulador o a unos sensores y que permita la modificación de los algoritmos del controlador, se llega a una comunicación con el usuario que implica un sistema[33] muy complejo, con una interfase gráfica y una capacidad de comunicación con otros equipos muy grande. Todo se maneja por bloques operativos, ya sean estos el equipo en sí, un algoritmo de control, un algoritmo de simulación, el protocolo de comunicación con un controlador, etc. Por lo que se puede recuperar los algoritmos utilizados por cierto equipo y sin necesidad de escribir código extra, reutilizarlos en un manipulador diferente, o tan sólo cambiar el algoritmo de control, sin estar obligado al desarrollo en específico del código.

---

[33] M.W. Gertz et al., "A human-machine interfase for distributed virtual laboratories", en *IEEE Robotics & Automation Magazine*, núm 4, vol. 1, diciembre de 1994, pp. 5-13.

El que las especificaciones a la arquitectura estén dadas, por el tipo de manipulador, el tipo de control, la forma de operación, etc; implica que existe toda una gama de posibles arquitecturas, dada la gran cantidad de manipuladores que hay, de los varios tipos de control utilizados, de las diversas tareas que realizan los manipuladores (que implican formas muy diferentes de operación), etc. En el caso específico de la tesis el diseño de la arquitectura parte de los objetivos y límites de ésta, que dan características muy definidas al manipulador ha controlar y su forma de operación; y que junto con la estrategia de control desarrollada en el capítulo 4, determinan las especificaciones generales de la arquitectura. Posiblemente uno de los objetivos más relevantes al diseño de la arquitectura, es el referente a tener que implementar el controlador: ya que deja a un lado configuraciones que se podrían haber considerado, de ser un ejercicio de diseño totalmente teórico, y hace que se deban de tomar en cuenta restricciones como costos, disponibilidad, forma de implementación, tiempo requerido desde el diseño a la implementación, etc.

Especificaciones generales a la arquitectura

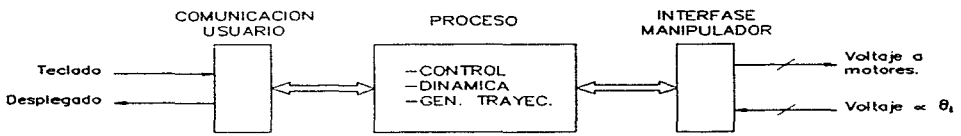


Fig. 5.1 Especificaciones generales a la arquitectura.

Por medio de la comunicación con el usuario se puede indicar secuencias simples de movimiento, formadas de puntos nodo por los cuales pasa el manipulador en un instante determinado. Los

puntos nodo se dan via teclado, ya sea en coordenadas cartesianas o en variables de unión y opcionalmente se puede utilizar el propio manipulador para grabar puntos en "línea" (en variables de unión). También se soporta el cambio de parámetros en los algoritmos de control, dinámica y generación de trayectoria, a parte de dar información sobre el comportamiento del sistema.
Las especificaciones a la capacidad de proceso, se centran en las rutinas que tienen un tiempo de ejecución determinado:

-Cálculo de la dinámica del manipulador (ecuaciones de N-E).

-Generación de la trayectoria (interpolación de la trayectoria entre puntos nodo).

-Cálculo de los seis servosistemas de control a nivel de cada unión (capítulo 4).

Un período de muestreo de alrededor de 5 ms (200Hz) en los servosistemas y en la generación de trayectoria, es el máximo esperado del sistema. Entendiendo que el período de muestreo es un compromiso entre el desempeño esperado y el costo del sistema e influye en la capacidad de rechazo a perturbaciones, la suavidad de la respuesta y en la sensibilidad a la variación de los parámetros involucrados. Si el sistema mecánico por lo general tiene una frecuencia estructural de alrededor[34] de 15 a 25 Hz (a mayor frecuencia, se puede tener movimientos más rápidos sin oscilaciones), en la mayoría de las referencias bibliográficas se establece un período de muestreo máximo de 16.667 ms (60 Hz), siendo preferible un período de muestreo menor que eso.
En la interfase con el manipulador se manejan dos tipos de señales, ambas de voltaje: 1) Se genera la señal de control a cada motor de corriente directa (CD).

2) Se adquiere la posición angular de cada unión, dada por un potenciómetro.

---

[34] Eugene I. Rivin "Mechanical Design of robots", Edit Mac Graw Hill 1988, USA, TJ211 R58, pág. 181

Las especificaciones generales a la arquitectura, establecen las características fundamentales del sistema: de inicio se requiere de un sistema digital para implementar la comunicación con el usuario y para los cálculos de la dinámica y generación de trayectoria. El tipo de control desarrollado en el capítulo 4, (un servosistema en variables de estado, con un observador de orden mínimo y con una compensación anticipativa) debe de implementarse en un sistema digital también, en contra parte de estrategias de control más sencillas, en las que existe la posibilidad de componentes analógicos en el servosistema de control.

Por lo que el controlador tiene la arquitectura de un sistema digital, basado en uno o varios microprocesadores o microcontroladores, cuyas características como su estructura general, tipo de procesador, cantidad de memoria, etc se tratan a continuación.

5.1 Descripción de la arquitectura.

La arquitectura puede ser descrita en base a los diferentes "ambientes" (unión de hardware/software que soporta una tarea) que se dan en un controlador para un manipulador. De las muy diversas configuraciones que pueden tenerse en la arquitectura de un controlador se distinguen, en cualquier sistema, tres "ambientes"[35]:

1) Desarrollo de la programación, el cual consiste del ambiente para la generación de movimientos (programa que da la secuencia de movimientos al manipulador) y el de desarrollo del programa en tiempo real (algoritmo de control, dinámica, trayectoria, etc).

---

[35] J. Ish-Shalom y P. Kazanzides, "SPARTA: Multiple signal processors for high-performance robot control", en *IEEE Trans. on Robotic and Automation*, núm 5, vol. 5, octubre de 1989, pp. 628-640.

2) Soporte de ejecución, es el ambiente por el cual el usuario opera al sistema, cambia el comportamiento de los algoritmos en tiempo real (al alterar sus parámetros), llama a las rutinas de movimiento creadas en el ambiente de desarrollo de la programación, y se las da al ambiente de tiempo real para su ejecución, etc.

3) Tiempo real, es el ambiente en donde se ejecuta el algoritmo de control, se calcula la dinámica, se genera la trayectoria y se realiza el acceso de entrada/salida con el manipulador.

De una forma u otra, en todos los controladores de manipuladores están los tres ambientes pero, su forma de implementación varia mucho; por ejemplo se puede implementar cada uno de forma independiente o por medio de un único sistema dar cabida a varios de ellos; y aunque no es indispensable, es preferible que los ambientes estén implementados en base a sistemas compatibles, para evitar problemas con formatos diferentes, códigos no compatibles, etc. La extensión de cada ambiente también puede variar, por ejemplo en un manipulador industrial el desarrollo de la programación no está implementado completamente ya que por lo común no se puede alterar los algoritmos en tiempo real.

Por lo general se busca como base del diseño[36], el no duplicar capacidades (tanto en hardware como en software) que se puedan encontrar en sistemas no dedicados a tareas en tiempo real; es por eso que una forma, bastante utilizada, de implementar la arquitectura es dividiéndola en dos grandes partes; una se encarga del ambiente en tiempo real y la otra de los otros dos ambientes que no tienen presiones sobre el tiempo de ejecución. Como se vio al inicio del capítulo, en los varios ejemplos de arquitecturas, esta división permite tener una arquitectura especializada para

[36] S. Narasimhan, "CONDOR: An architecture for controlling the Utah-MIT dexterous hand", en *IEEE Trans. on Robotic and Automation*, núm. 5, vol. 5, octubre de 1989, pp. 616-627.

cálculos en tiempo real y otra; generalmente basada en equipos convencionales como computadoras personales, para soportar los ambientes de desarrollo de programación y soporte de ejecución.

Debido a la cantidad de cálculos, la mayoría de las aproximaciones a la arquitectura de tiempo real, tienen alguna forma de proceso en paralelo; ya sea a nivel de circuitos integrados diseñados en específico o una arquitectura con varias unidades de proceso. Lo más utilizado es un multiproceso dado por varias unidades, basado en microprocesadores o en DSP y llegando hasta un esquema de procesamiento en paralelo "grueso" (a nivel de dividir funciones específicas a diferentes unidades de procesamiento); ya que la magnitud de la comunicación necesaria para organizar las operaciones y sincronizar datos entre procesadores en un procesamiento en paralelo "fino" es enorme, y a demás se requiere de algoritmos que exploten el paralelismo a nivel de operaciones básicas en las diferentes rutinas que se dan en el controlador, lo que no es trivial; por lo que muchas veces el procesamiento en paralelo "fino" se realiza a nivel de operaciones con vectores y matrices.

Por lo que pese a la carga de cálculos, se logra[37] cumplir con la tarea en tiempo real, con un esquema de procesamiento en paralelo "grueso" con características como: utilizar líneas de interrupción para indicar mensajes entre procesadores, tener una memoria de doble puerto para compartir estructuras y mensajes, contar con una unidad de punto flotante, etc.

En cuanto al ambiente de desarrollo de programación, se implementó únicamente la generación de movimientos que junto con el ambiente de soporte de ejecución, constituyen todo el proceso fuera de línea. Por simplicidad y para mantener la arquitectura en un tamaño razonable, se

[37] *Idem.*

decidió que la misma arquitectura que soporta al ambiente de tiempo real, soportara también a los otros dos ambientes; lo que provoca que se añadan elementos a la arquitectura para implementar una comunicación con el usuario, que por tratarse básicamente de una arquitectura para tiempo real, es muy sencilla.

De la figura 5.2 se puede observar que se tiene a dos procesadores trabajando en paralelo, cuya interconexión se logra a través de un puerto de control formado por dos puertos de ocho bits en cada procesador, uno de entrada y otro de salida; de forma que en cualquier momento un procesador puede escribir o leer un byte sin necesidad de sincronizarse con el uso que esté recibiendo el puerto de control, por el otro procesador. También se tiene una memoria compartida, es decir una memoria que está mapeada por los dos procesadores y que cada uno puede accesar; la memoria está aislada de ambos procesadores y tiene la lógica para evitar que en un acceso simultaneo, se de una contención en el bus de direcciones o en el de datos.

Cada procesador cuenta con su propia memoria, lo que permite un funcionamiento independiente de cada uno y sólo se establece la comunicación entre procesadores para sincronizar el inicio de operación y el período de muestreo, a parte del intercambio de comandos, información, etc. La conexión de los dos procesadores no permite fácilmente el crecimiento del sistema pero, logra la conexión de una forma efectiva y con un mínimo de dispositivos; lo que resulta en una arquitectura de tamaño reducido y en cierto grado sencilla, que ayuda a la implementación.
La línea de interrupción no enmascarable (NMI), se utiliza para indicar una condición de "ALTO" al manipulador, lo que constituye el paro de emergencia para el sistema.La línea de NMI es común a los dos procesadores, al igual que la línea de RESET.

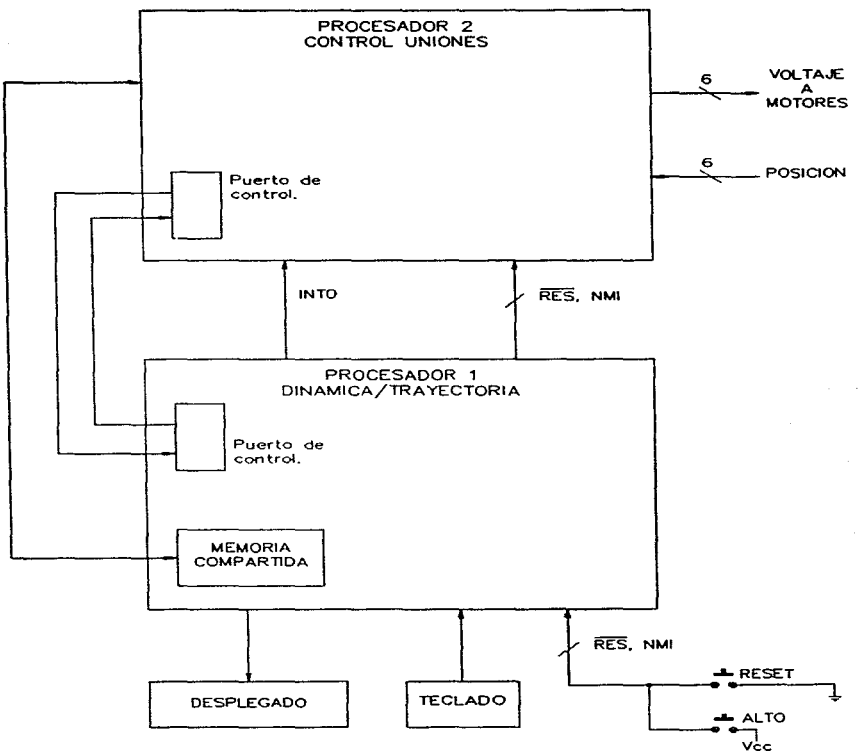


Fig. 5.2 Diagrama General

Al tener los tres ambientes implementados bajo una misma arquitectura, se tienen diferentes actividades de acuerdo a que ambiente esté en ejecución; como se puede ver en la figura 5.2 el procesador 1 es el que da la atención al teclado y al desplegado, es decir es el que maneja la comunicación con el usuario y por tanto implementa el ambiente de desarrollo de programación (generación de movimientos) y el de soporte de ejecución, por el cual se maneja todo el sistema; lo que hace al procesador 1 el administrador de el sistema y es prácticamente el único encargado de estos dos ambientes, ya que el procesador 2 se encuentra únicamente en comunicación con el procesador 1, con la capacidad de mandar y recibir mensajes, por lo que reacciona ante cambios de parámetros que conciernen a los algoritmos de control, pruebas de operación, comandos específicos, etc.

Por medio del ambiente de soporte de ejecución, se lleva al sistema al funcionamiento en tiempo real, en este ambiente cambia la actividad de los procesadores y se tiene un funcionamiento simultaneo de ambos:

-El procesador 1 se encarga del cálculo de la dinámica y de la generación de trayectoria.

-El procesador 2 ejecuta los seis servosistemas de control de unión y como se ve en la figura 5.2, es el encargado de generar la señal de control (voltaje) a los motores de CD en el manipulador y de realizar la adquisición de las señales analógicas (voltajes) que son proporcionales al desplazamiento angular de las uniones.

En tiempo real la comunicación entre procesadores consiste básicamente, en la posición deseada para cada unión de acuerdo a la trayectoria que se está ejecutando y la compensación anticipativa, que es el resultado de la dinámica inversa; en ambos casos es el procesador 1 el que manda los datos al procesador 2 por medio de la memoria compartida, sincronizando las transferencias por

medio de una línea de interrupción, para agilizar la comunicación. A parte se puede dar mensajes y comandos entre los dos procesadores, para propagar información, condiciones de operación,etc. Cada unidad de procesamiento está implementada en base a un sistema con un 80C186XL, de forma que se logra una implementación compacta, de poca disipación de potencia, teniendo el desempeño de un sistema de 16 bits; y hace que se pueda utilizar cualquier computadora personal, para el desarrollo de las rutinas.

5.1.1 Procesador 1

El procesador 1, al igual que el procesador 2, son básicamente un microprocesador unido a memoria, puertos y a una unidad de punto flotante; para simplificar el diseño se trató de que ambos sistemas (procesador 1 y 2) fueran lo más parecidos posibles.

Como se puede ver en la figura 5.3, el procesador 1 está compuesto por cinco bloques (en el apéndice E, se encuentran los esquemáticos de cada bloque).

1) 80C186XL/80C187[38]

Este bloque está compuesto por el microprocesador, su unidad de punto flotante y circuitos para el demultiplexaje de los "buses" de direcciones y datos; por tratarse del 80C186, un microprocesador de 16 bits de alta integración, se logra un ahorro en espacio y componentes en la implementación; ya que contiene varios de los dispositivos más usados en los sistemas basados en un microprocesador, como el generador de reloj, el controlador de interrupciones, timers, generador de habilitaciones, generador de estados de espera, etc; todos los periféricos

[38] Para mayor información sobre el microprocesador ver los datos técnicos 80C186XL (Intel Literature 2724B1-001) y el manual del 80C186XL en "Embedded microprocessors" (Intel Literature 272396). Para mayor información sobre el coprocesador matemático ver 80C187 (Intel Literature 270640-003).

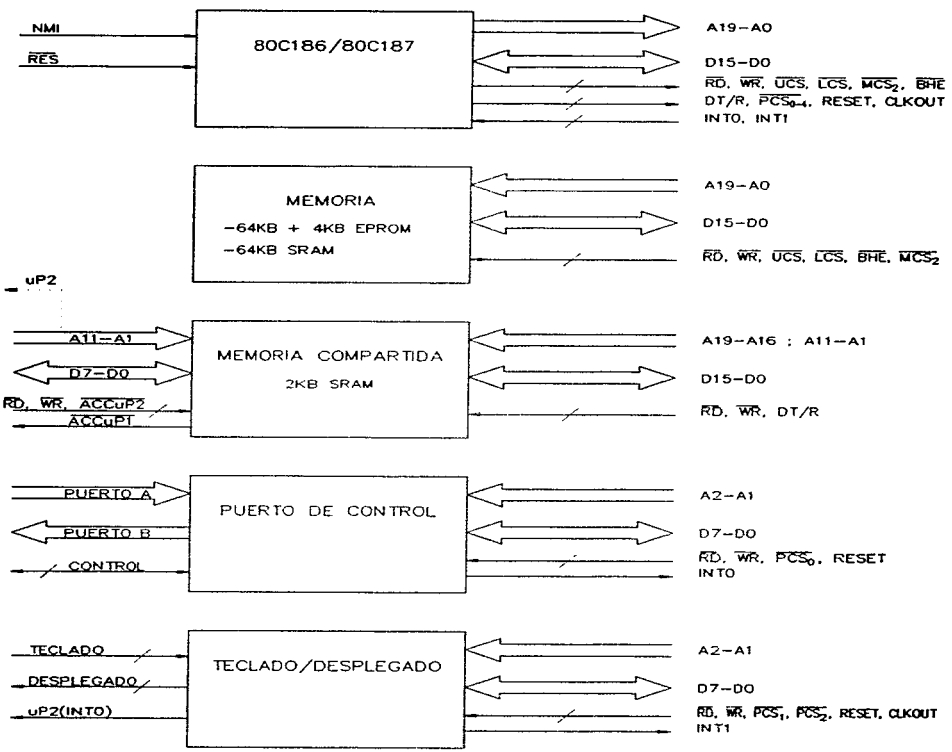


Fig. 5.3 Procesador I

integrados dentro del 80C186XL se controlan por medio de registros de 16 bits contenidos en un bloque de control de 256 bytes.

El generador de habilitaciones puede generar seis señales en el espacio de memoria y siete en el de entrada/salida, pudiéndose programar su localización, estados de espera, el tamaño del espacio de memoria habilitado, etc. El controlador de interrupciones se configura en el modo totalmente anidado y puede manejar cinco fuentes externas y dos fuentes internas (DMA y timers) de interrupción, siendo las externas:

-NMI Paro de emergencia.

-INT0 Puerto de control.

-INT1 Teclado.

-INT2 No utilizada.

-INT3 No utilizada.

El 80C186XL tiene ciertas diferencias con respecto al 80186 original, siendo posiblemente una de las más relevantes la del coprocesador matemático, ya que no soporta el uso del 8087, si no que utiliza el 80C187 que es compatible en código con el 80387DX. El 80C186XL al estar conectado con el 80C187, entra en operación en modo extendido lo que modifica el uso de tres señales de habilitación de la zona de memoria; dado que se utilizan en la interfase con el 80C187, obteniendose una mejor sincronización de datos, instrucciones y excepciones entre el coprocesador y el procesador, que en los sistemas que utilizan un 8087.

Como se puede ver en la figura 5.3, el bloque 80C186/80C187 genera el "bus" de direcciones de 20 líneas, el "bus" de datos de 16 bits y el "bus" de control en donde las líneas denominadas $\overline{UCS}$, $\overline{MCS_2}$, $\overline{LCS}$

son las señales dadas por el generador de habilitaciones a la zona de memoria y son las habilitaciones a zona alta, intermedia y baja respectivamente. La líneas $\overline{PCS_0}$ a $\overline{PCS_4}$ son las líneas de habilitación a dispositivos, siendo $\overline{PCS_0}$ la del puerto de control, $\overline{PCS_2}$ la del puerto del desplegado y $\overline{PCS_4}$ la del controlador del teclado (8279-5). No se utiliza a $\overline{PCS_1}$.

El 80C186XL opera a una frecuencia de 12 MHz, lo que implica los siguientes tiempos:

-Tiempos de lectura:

Tiempo desde dirección válida a datos ya en el bus.

$$(3+N)t_{CLCL} - t_{CLAV} - t_{DVCL} - t_{INOV} = 181 + N(83.33)\ ns$$

Tiempo desde $\overline{CS}$ (chip select).

$$(3+N)t_{CLCL} - t_{CLCSV} - t_{DVCL} = 202 + N(83.33)\ ns$$

Tiempo desde $\overline{RD}$.

$$(2+N)t_{CLCL} - t_{CLRL} - t_{DVCL} = 114.67 + N(83.33)\ ns$$

Duración de $\overline{RD}$.

$$(2+N)t_{CLCL} - 25 = 141.67 + N(83.33)\ ns$$

Donde: t_{CLCL} (período de reloj) = 83.33 ns.

t_{CLAV} (retraso en direcciones) = 36 ns máximo.

t_{DVCL} (setup en datos) = 15 ns máximo.

t_{INOV} (retraso en latch) = 18 ns máximo.

t_{CLCSV} (retraso en la habilitación) = 33 ns máximo.

t_{CLRL} (retraso en $\overline{RD}$) = 37 ns máximo.

N (estados de espera).

-Tiempos de escritura:

Tiempo desde dirección válida, hasta final de $\overline{WR}$.

$$(3+N)t_{CLCL} - t_{CLAV} - t_{IVOV} + t_{CVCTX} = 199 + N(83.33)\ ns$$

Tiempo de $\overline{CS}$ (chip select) a $\overline{WR}$ desactivado.

$$(3+N)t_{CLCL} - t_{CLCSV} + t_{CVCTX} + t_{CXCSX} = 244.667 + N(83.33)\ ns$$

Duración de $\overline{WR}$

$$(2+N)t_{CLCL} - 25 = 141.67 + N(83.33)\ ns$$

Donde: t_{CVCTX} (retraso en desactivar $\overline{WR}$) = 3 ns máximo.

t_{CXCSX} (retraso en desactivar chip select después de $\overline{WR}$) = t_{CLCH} - 10 mínimo.

t_{CLCH} (tiempo bajo en CLK) = 0.5 t_{CLCL} - 7 ns = 34.667 ns.

Para las EPROM`s que se quieran utilizar, se deben de tener las siguientes características:

$$t_{ACC} < 181 + N(83.33) \text{ ns}$$

$$t_{CE} < 202 + N(83.33) \text{ ns}$$

$$t_{OE} < 114.67 + N(83.33) \text{ ns}$$

Donde: t_{ACC} es el retraso de dirección válida a salida de los datos.

t_{CE} es el retraso de chipselect válido a salida de datos.

t_{OE} es el retraso de $\overline{OE}(\overline{RD})$ válido a salida de datos.

Por lo que si no se quiere insertar estados de espera, se requiere de EPROM`s con tiempos de acceso menores a 181 ns. Como el 80C186XL puede insertar un máximo de tres estados de espera, sin circuitos adicionales, en caso extremo las memorias deben de tener un tiempo de acceso menor a 431 ns. Para las memorias RAM, es exactamente el mismo caso, tiempos de acceso menores a 181 ns, para operar sin estados de espera.

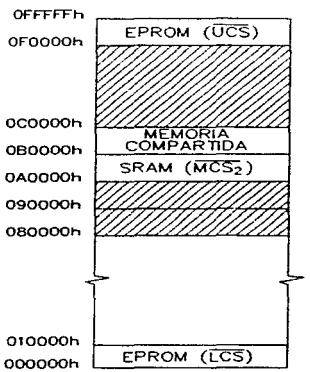


Fig. 5.4 Mapa memoria μP1.

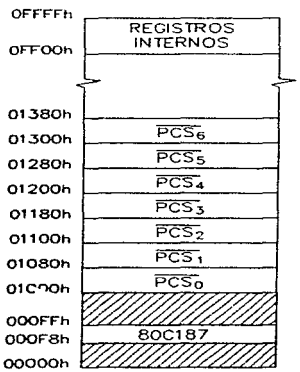


Fig. 5.5 Mapa entrada/salida.

Del mapa de memoria (figura 5.4) se tiene que el procesador cuenta con 64KB tanto de EPROM para el ejecutable con un segmento de código en 0F000h, como de SRAM (RAM estática) con el segmento de datos en 0A000h.

No se requiere de estados de estados de espera en la mayoría del espacio de memoria, tan sólo en la zona inferior (vectores de interrupción), en donde se requiere de tres estados de espera.

Se tienen las siguientes direcciones válidas en memoria:

0F0000h a 0FFFFFh (64KB EPROM); ejecutable y parámetros.

0B0000h a 0B0FFEh (2KB SRAM); direcciones pares, (mem. comp.).

0A0000h a 0AFFFFh (64KB SRAM); segmento de datos.

000000h a 000FFFh (4KB EPROM); vectores de interrupción.

El mapa de entrada/salida (figura 5.5), tiene la estructura dada por el modo en que el 80C186 genera las habilitaciones, dando a cada señal de habilitación una cobertura de 128 bytes. El puerto de control ($\overline{PCS_0}$) y el desplegado ($\overline{PCS_2}$) consisten de un puerto (82C55) que requiere de un estado de espera, el teclado ($\overline{PCS_4}$) requiere de tres estados de espera.

Se tienen las siguientes direcciones válidas en el espacio de entradasalida:

-Puerto de control (82C55A-2, U1_25) $\overline{PCS_0}$

01000h	Puerto A	(Entrada).							
01002h	Puerto B	(Salida).							
01004h	Puerto C	(Control+ 2 Led`s + INT0).							
			PC_7	PC_6	PC_5	PC_4	PC_3	PC_2	PC_1	PC_0
			$\left(\frac{LED}{BLANCO}\right)$	$\left(\frac{LED}{ROJO}\right)$	IBF_A	STB_A	$INT0$	ACK_B	OBF_B	x
01006h	Puerto de control del 82C55.									

-Desplegado (82C55A-2, U1_28) $\overline{PCS_2}$

01100h	Puerto A	(Datos a conector CON4, desplegado izquierdo).
01102h	Puerto B	(Datos a conector CON5, desplegado derecho).
01104h	Puerto C	(Líneas de control + INT a procesador 2).

PC_7	PC_6	PC_5	PC_4	PC_3	PC_2	PC_1	PC_0
$\left(\frac{INT}{\mu P_2}\right)$	E	R/W	RS	x	E	R/W	RS
		CON5				*CON4*	

01106h Puerto de control del 82C55.

-Teclado (8279_5, U1_27) $\overline{PCS_4}$

01200h	Datos
01202h	Control

-Coprocesador (80C187, U1_2)

000F8h a 000FFh Reservadas para la comunicación con el 80C186.

2) Memoria

Se tiene en la zona superior dos EPROM`s 27C256Q150, para dar los 64KB asignados al ejecutable, los 4KB de la zona inferior están dados por dos 27C16BQ200. El segmento de datos está formado por 64KB de SRAM (dos 62256LP-12) y tiene un respaldo por batería de forma que se conservan los datos del sistema (figura 5.6) y se utiliza un circuito de alimentación a las memorias 62256LP-12, independiente del resto del sistema y cuyo voltaje (+5.7v) contrarresta la caída en el diodo D1.

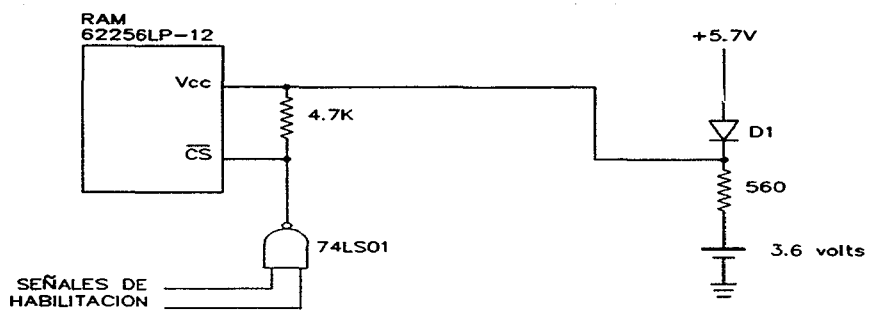


Fig. 5.6 Respaldo por batería.

En operación la fuente alimenta a las memorias y a través de la resistencia se carga la batería, la cual es de Nickel-Cadmio de 3.6 volts; durante la carga el voltaje de la batería aumenta a 3.9 V y recibe una corriente de 1.8 mA, con la que se puede estar cargando continuamente sin problemas.

Al apagarse la fuente, el diodo impide que la batería se descargue en los dispositivos de la fuente y sólo alimenta a las memorias por Vcc y las mantiene desactivadas al sostener en $\overline{CS}$ un "1", ya que están desactivadas las compuertas que activan a las SRAM. Las memorias consumen una potencia muy baja cuando no están activadas, del orden de 2 a 100 μA con un $2.0 \leq V_{CC} \leq 5.5$ volts y un $\overline{CS} = V_{CC} - 0.2$ volts. Durante el respaldo por batería, se midió una corriente total de 8 μA y considerando que la batería da un servicio de 280 mAh a 3.0 volts, se tiene asegurado un respaldo a los datos por un tiempo muy grande de inactividad del sistema.

3) Memoria compartida

Para poder comunicar a los dos procesadores y que esta comunicación sea lo más rápida posible, es que se implementó la memoria compartida (figura 5.7), simplemente es una memoria que los dos procesadores pueden accesar pero, que está aislada con "transivers" y "buffer" en cada procesador y al darse un acceso, la señal de habilitación permite que los circuitos de aislamiento se activen en el procesador adecuado y por tanto se puede escribir y leer sobre la memoria compartida, sin perturbar al bus de direcciones y datos del otro procesador.

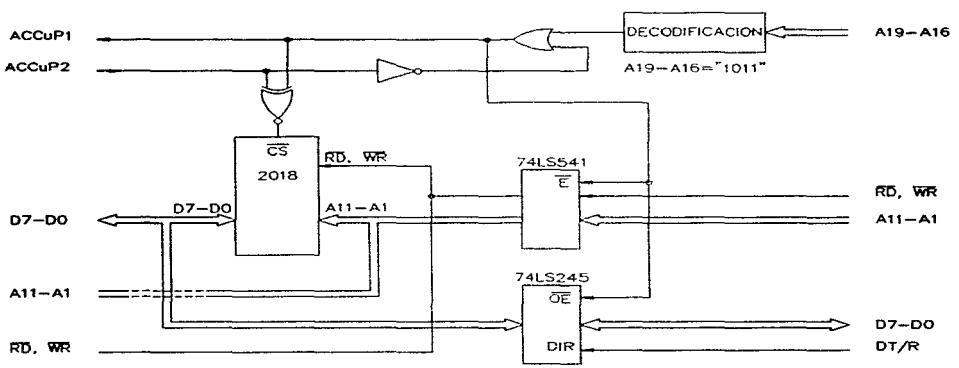


Fig. 5.7 Memoria compartida.

Por seguridad, se incluyó cierta lógica que impide el acceso de un procesador a la memoria compartida, si ésta ya está siendo usada por el otro procesador, simplemente los "buffers" y "transivers" no se activarán. En caso de un acceso simultaneo, en el que la lógica no alcance a desactivar uno de los accesos, la memoria no se activará, ya que ésta entra en operación con la

XNOR de ambas señales de habilitación. De todas formas el uso de la memoria compartida, es exclusivo para comunicar datos entre los procesadores y se da dentro de un protocolo que utiliza también al puerto de control, para sincronizar los accesos y que una escritura por parte de uno de los procesadores implique por la misma rutina de uso, que se le indique al otro procesador que existe cierto dato en memoria compartida y de igual forma una lectura a memoria compartida implica una indicación al otro procesador que los datos han sido recibidos, esta rutina se explica en el capítulo 6.

Por simplicidad, se decidió implementar sólo 2KB de la memoria compartida con un sólo circuito integrado, por lo que se tienen direcciones pares únicamente y es otra cosa que la rutina de comunicación por medio de memoria compartida, debe de tener en cuenta; el hacer los movimientos de datos, de bytes consecutivos a direcciones pares únicamente y viceversa según sea el acceso a memoria compartida.

La lógica de decodificación, así como la anulación de un acceso simultaneo junto con los circuitos de aislamiento de la parte del procesador 2, es encuentran dentro del sistema del procesador 2.

4) **Puerto de control**

El puerto de control (figura 5.8) da una comunicación entre procesadores, sin necesidad de sincronización y que permite mandar comandos, códigos de operación, información, etc a nivel de un byte, de forma que ayuda a coordinar la comunicación y la operación de todo el sistema. Consiste de un 82C55A-2[39], con los puertos A y B en modo 1 y configurados como entrada y salida respectivamente; en ese modo las transferencias utilizan dos líneas de control por puerto.

[39] Para mayor información sobre el 80C55A-2 ver la hoja de datos: 82C55A (Intel Literature 231256-004).

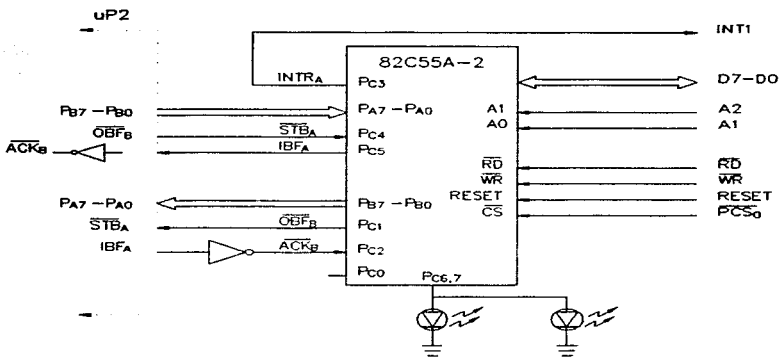


Fig. 5.8 Puerto de control.

En el puerto A (entrada), se cargan los datos al darse $\overline{STB_A}$, una vez cargados se activa IBF_A para indicar a la fuente de los datos, que estos ya han sido recibidos; al mismo tiempo se activa $INTR_A$ (INT1), indicando al procesador que hay datos en el puerto de entrada. Al leer el puerto A, IBF_A se desactiva.

En el puerto B (salida), al escribir sobre él se activa $\overline{OBF_B}$, indicando al receptor de los datos que estos ya están disponibles; el puerto espera a que se active $\overline{ACK_B}$ a forma de confirmación de la transferencia, una vez activado $\overline{ACK_B}$, se desactiva $\overline{OBF_B}$ y acaba la transferencia.

En el procesador 2 existe un puerto de control idéntico, de forma que se realiza una transferencia a la vez, de modo idéntico en ambos procesadores, como se muestra en la figura 5.8 .La rutina

de comunicación por el puerto de control requiere de revisar las señales para asegurar la realización de la transferencia; y para que ésta se de sin errores se tienen diseñadas diferentes formas de comunicación, las cuales se describen en el capítulo 6 (programación).

Dos de los bits que no son utilizados para las señales de control, se utilizan para manejar dos led`s a forma de indicadores del funcionamiento del sistema, pudiendo señalar externamente la ocurrencia de ciertos eventos (errores, modo de operación, fallas, etc).

5) Teclado / Desplegado

Este bloque realiza toda la comunicación con el usuario, tratando de trabajar con lo mínimo que cumpla con lo especificado; se escogió un teclado de doce teclas por que se puede trabajar datos numéricos, teniendo dos teclas extras para validar la entrada, aparte de contar con doce opciones para funciones no numéricas. Puede considerarse como el mínimo para manejar con cierta comodidad las especificaciones deseadas, con la ventaja que puede ser un teclado telefónico; lo que implica que se cuenta con un teclado de uso pesado, compacto, de buen diseño, etc. Se decidió controlarlo por medio de un 8279-5, evitando los circuitos extras para quitar el rebote del teclado y hacer la adquisición de la tecla presionada; y para tener la posibilidad de aumentar el número de teclas sin una modificación extensa del bloque.

Por tener un número de teclas reducido y por que desde un inicio se pensó que el manejo del sistema fuera por medio de menús, con selección de opciones, se optó por un desplegado formado por un display de cristal líquido; con esto se hizo posible el marcar claramente en que estado (menú) se encuentra el sistema, que se necesita como entrada, cuales son las opciones disponibles, etc; de igual forma se puede indicar los diferentes usos de las teclas, conforme van cambiando durante la operación del sistema.

Para facilitar la operación del sistema, se prefirió contar con cuatro renglones de diez y seis caracteres, repartidos en dos grupos; cada grupo está compuesto por un desplegado AND491 (ver el apéndice F, para más información sobre el AND491), que es un módulo compacto de desplegado de cristal líquido, el cual contiene el controlador, los circuitos de manejo y la pantalla de cristal líquido; el controlador es capaz de generar 168 caracteres diferentes, formados por una matriz de 5x7 puntos. La interfase del controlador con el procesador es por medio de un bus de datos de 8 bits y tres líneas de control ("RS" para seleccionar datos o comandos, "E" habilitación del desplegado y "R/W"). El controlador del desplegado es muy lento y requiere de mas de tres estados de espera, lo que impide que se pueda acoplar directamente al 80C186; por lo que se decidió controlar a los dos desplegados por medio de un puerto (82C55), en vez de generar estados de espera independientes de los internos del procesador. En el apéndice E, se muestra el esquemático de este bloque y los dos potenciómetros que aparecen en el diagrama son para controlar el contraste de cada pantalla de cristal líquido.

5.1.2 Procesador 2

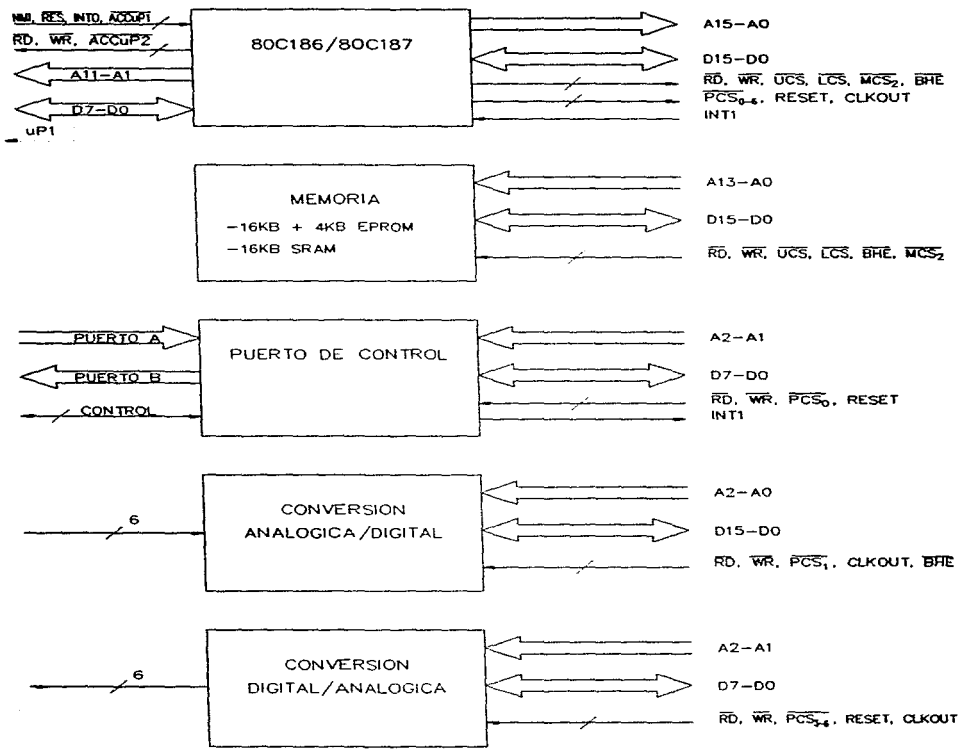


Fig. 5.9 Procesador 2.

El procesador 2, muy parecido al procesador 1, es el encargado de la comunicación con el manipulador; de la figura 5.9 se tiene que está compuesto por cinco módulos (ver el apéndice E, para mayor información de cada bloque):

1) **80C186/80C187**

Este bloque es casi idéntico al del procesador 1 pero, contiene a los circuitos de aislamiento ("buffers y transivers"), la lógica de anulación de acceso simultaneo y la decodificación de la dirección de la memoria compartida; que en si cumple con las mismas características explicadas en el procesador 1, en el bloque de memoria compartida. Otra diferencia es que no hay circuitos de generación de RESET y NMI, estas señales se producen en el procesador 1 y llegan al procesador 2 por medio del conector CON10. Por fuera de las diferencias, el bloque 80C186/80C187 funciona de la forma explicada para el procesador 1.

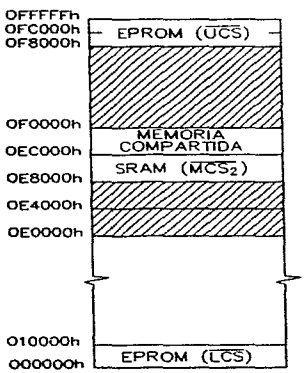


Fig. 5.10 Mapa memoria μP2.

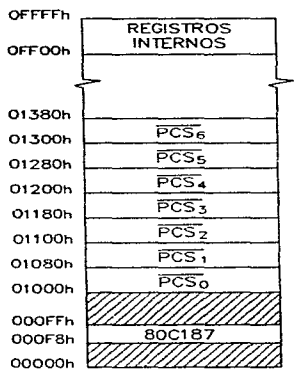


Fig. 5.11 Mapa entrada/salida.

De la figura 5.10 se tiene que el 80C186XL cuanta con un espacio reservado de 32KB de EPROM para el ejecutable pero, sólo se implementó la mitad; situando al segmento de código en 0F000h. El segmento de datos está situado en 0E000h implementado por 16KB de SRAM y los 2KB en direcciones pares de la memoria compartida. La zona de los vectores de interrupción se implementó con 4KB de EPROM, que al igual que en el procesador 1, requieren de tres estados de espera.

En la zona de memoria, se tiene las siguientes direcciones válidas:

0FC000h a 0FFFFFh (16KB EPROM): ejecutable y parámetros.

0F8000h a 0F8FFFh (16KB): reservado para el segmento de código.

0EC000h a 0ECFFEh (2KB SRAM): memoria compartida (dir. pares).

0E8000h a 0EBFFFh (16KB SRAM): segmento de datos.

000000h a 000FFFh (4KB EPROM): vectores de interrupción.

El mapa de entrada/salida tiene la misma estructura que el del procesador 1; el puerto de control ($\overline{PCS_0}$) y el bloque de conversión digital/analógica: motor 1 al 6 ($\overline{PCS_3}$,$\overline{PCS_4}$,$\overline{PCS_5}$) están formados por puertos (82C55) que requieren de un estado de espera. Los convertidores analógico/digital ($\overline{PCS_1}$) también requieren de un estado de espera, así como la conversión digital/analógica: dirección ($\overline{PCS_6}$).

Se tienen las siguientes direcciones válidas en el espacio de entradasalida:

-Puerto de control (82C55A-2, U2_26) $\overline{PCS_0}$

01000h Puerto A (Entrada).

01002h Puerto B (Salida).

01004h Puerto C (Control+ 2 Led`s + INT1).

PC_7	PC_6	PC_5	PC_4	PC_3	PC_2	PC_1	PC_0
$\left(\frac{LED}{BLANCO}\right)$	$\left(\frac{LED}{ROJO}\right)$	IBF_A	STB_A	$INT0$	ACK_B	OBF_B	x

01006h Puerto de control del 82C55.

-Convertidor A/D (ADC7802, U2_63 y U2_64) $\overline{PCS_1}$

01080h Escritura (16 bits): inicio de la conversión en los dos convertidores. D_9D_8 - D_1D_0 codifican que canal se convierte.

01080h Lectura (16 bits) obtención byte bajo de la conversión.

01082h Lectura (16 bits) obtención nibble alto de conversión.

01084h Reservado

01086h Acceso (escritura/lectura) al SFR (registro de funciones especiales).

-Conversión D/A motor 1 y 2 (82C55A-2, U2_21) $\overline{PCS_3}$

01180h Puerto A (Motor 2, byte bajo).

01182h Puerto B (Motor 1, byte bajo).

01184h Puerto C (Motor 1 y 2, nibble alto). P_{C7}-P_{C4} motor 1.
P_{C3}-P_{C0} motor 2.

01186h Puerto de control del 82C55.

-Conversión D/A motor 3 y 4 (82C55A-2, U2_22) $\overline{PCS_4}$

01200h Puerto A (Motor 4, byte bajo).

01202h Puerto B (Motor 3, byte bajo).

01204h Puerto C (Motor 3 y 4, nibble alto). P_{C7}-P_{C4} motor 3.
P_{C3}-P_{C0} motor 4.

01206h Puerto de control del 82C55.

-Conversión D/A motor 5 y 6 (82C55A-2, U2_23) $\overline{PCS_5}$

01281h Puerto A (Motor 6, byte bajo).

01283h Puerto B (Motor 5, byte bajo).

01285h Puerto C (Motor 5 y 6, nibble alto). P_{C7}-P_{C4} motor 5.

P_{C3}-P_{C0} motor 6.

01287h Puerto de control del 82C55.

-Conversión D/A dirección (82C55A-2, U2_24) $\overline{PCS_6}$

01301h |1/0|x|M_6|M_5|M_4|M_3|M_2|M_1|; |M_i| =dirección motor "i"

=(1=giro antihorario)

|1/0| = iluminación "ALTO".

-Coprocesador (80C187, U2_2)

000F8h a 000FFh Reservadas para la comunicación con el 80C186.

Las fuentes de interrupción externas están dadas por:

-NMI Paro de emergencia.

-INT0 Interrupción directa del procesador 1.

-INT1 Puerto de control.

-INT2 No utilizada.

-INT3 No utilizada.

2) **Memoria**

Se tiene en la zona superior dos EPROM's 27C64Q-150, para dar los 16KB del ejecutable pero, sin muchos problemas se pueden sustituir por dos 27C128 para dar los 32KB reservados. El segmento de datos está formado por dos 6264LP-10, que dan los 16KB de RAM y en la zona de los vectores de interrupción, dos 27C16BQ200 dan los 4KB de la zona baja.

3) **Puerto de control**

Este bloque es idéntico en funciones y características, al que se tiene en el procesador 1.

4) **Conversión analógica/digital**

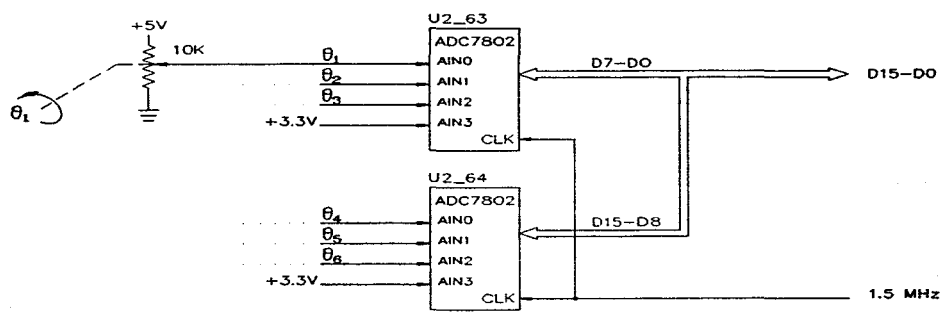


Fig. 5.12 Conversión A/D.

El bloque de conversión analógica/digital (A/D) realiza la adquisición de seis voltajes, que son proporcionales a la posición de cada unión; y conforme a los objetivos y límites de la tesis expuestos en el capítulo 1, se tiene que la posición de cada unión está dada por un potenciómetro

lineal de 10K Ω , acoplado directamente al movimiento de ésta. Por simplicidad se considera que no hay problemas de linealidad con los potenciómetros y estos son de una vuelta, de forma que el rango de 0 a 5 volts de las señales de voltaje, corresponde a los 360° de una vuelta completa.

Como se puede ver en la figura 5.12, se utilizan dos convertidores A/D de forma que se realiza la adquisición simultanea de dos señales, θ_1 con θ_4, θ_2 con θ_5 y θ_3 con θ_6. El objetivo de utilizar dos convertidores, es el de disminuir en lo posible el tiempo de adquisición de las seis señales, con lo que se puede trabajar con períodos de muestreo más pequeños y con menos presión sobre los tiempos de ejecución en el algoritmo de control.

El convertidor que se escogió es el ADC7802 de la Burr Brown (ver el apéndice F, para las características completas), que es un convertidor CMOS de aproximaciones sucesivas, de 12 bits, que cuenta con un multiplexor analógico de cuatro canales. Se optó por el ADC7802 por varias razones: -El que sea de 12 bits facilita la implementación, en comparación de las dificultades de ruido y sensibilidad a cambios de alimentación que se da en los convertidores de 16 bits.

-Tiene una interfase con el microprocesador, que hace fácil la conexión con el 80C186. Pese a que su bus de datos es de 8 bits, lo que implica dos lecturas para obtener una conversión completa, no hay problemas de desempeño por el uso de los dos ADC7802.

-Es rápido, con un tiempo de conversión de 11.33 μs operando a 1.5 MHz.

-Es fácil de diseñar con él, ya que todos los circuitos de compensación son internos al ADC7802, por medio de un ciclo de autocalibración que deja a $\pm$½ LSB (bit menos significativo) máximo al error ajustado total; garantizando que no hay pérdida de códigos. También al tener un muestreador/retén interno, así como el multiplexor de cuatro canales, hace

que se ahorre espacio al tener menos componentes y se tiene un buen desempeño, dando una razón de cambio máxima a la señal de entrada de 8 mV/μs, resultado del muestreador interno, con un tiempo de apertura de 76.29 ns y que implica una frecuencia máxima de 500 Hz, si la señal de entrada es sinusoidal, con un voltaje pico de 2.5 volts.

Al tener una interfase con el procesador de 8 bits, los ADC están colocados en direcciones adyacentes, en el espacio de entrada/salida, por lo que al hacer una escritura o lectura de 16 bits se está accesando a los dos ADC7802 al mismo tiempo. Como se puede ver en el apéndice E, el ADC7802 (U2_63) es el que maneja la adquisición de θ_1, θ_2 y θ_3 y se encuentra colocado en la parte baja del bus de datos (D_7-D_0), mientras que el ADC7802 (U2_64) se encuentra en la parte alta del bus de datos (D_{15}-D_8) y maneja a θ_4, θ_5 y θ_6.

El ADC7802 tiene varias entradas de control como es "HBE" y "SFR", las cuales controlan el acceso al byte bajo o nibble alto y al acceso al registro de funciones especiales respectivamente. Las entradas A_0 y A_1, en una escritura determinan en que canal se da la conversión. Como se puede ver en el esquemático (apéndice E), A_0 y A_1 están conectadas al bus de datos y "HBE", "SFR" al bus de direcciones; de forma que como se indicó en el mapa de entrada/salida una escritura a la dirección 01080h (HBE="0", SFR="0") inicia la conversión y de acuerdo a la palabra que se escriba se selecciona los canales a conviertir (00000h canales 0, 00101h canales 1, 00202h canales 2 y 00303h canales 3).

La salida del ADC7802 "BUSY" se dejó libre por lo que la única forma de saber si ya se terminó la conversión es accesando al SFR (registro de funciones especiales) y verificar el bit D_7 y D_{15}. El SFR se accesa en la dirección 01086h (HBE="1", SFR="1") y aparte de la información de "BUSY", también se puede accesar las banderas de error en la alimentación, error en la

calibración y el modo de operación; al escribir sobre el SFR se puede iniciar una calibración, desactivar alguna bandera de error o cambiar el modo de funcionamiento.

Para leer el resultado de la conversión, se tiene que hacer dos lecturas, una a la dirección 01080h con la que se recupera el byte bajo de las conversiones iniciadas; otra lectura a la dirección 01082h y se obtiene el nibble alto de las conversiones y como ya se vio, en la parte alta del bus de datos (D_{15}-D_8) se tiene a θ_4, θ_5 y θ_6 y en la parte baja (D_7-D_0) a θ_1, θ_2 y θ_3. Por lo que la rutina de uso de la conversión A/D, tiene que realizar la separación de los datos de cada ADC y ordenarlos ya en una palabra con 12 bits válidos; a parte de verificar el estado de "BUSY" y banderas de error en el SFR.

El modo de operación del ADC7802 que se prefirió, fue el de salida en "latch" en el que se guarda el resultado de la conversión aun cuando se haya iniciado otra, de esta forma al desactivarse "BUSY" de inmediato se inicia la conversión del canal siguiente y después se lee la conversión ya terminada, mientras el ADC está trabajando.

El reloj de los ADC7802 tiene una frecuencia de 1.5 MHz (tras dividir el reloj del sistema de 12 MHz por ocho) y como los ADC requieren de 17 ciclos para realizar la conversión se tiene un tiempo de conversión de 11.33 μs. Al traslapar la conversión con las lecturas se puede convertir los seis canales en un tiempo de alrededor de los 34 a 35 μs.

En total se tienen ocho canales, cuatro por ADC7802, de los cuales están utilizados seis, tres por convertidor, los dos restantes se usan como prueba del bloque. El canal 3 de cada ADC7802 está conectado a un diodo zener de 3.3 volts y durante la fase de prueba del sistema, se realizan varias lecturas a ese canal y su promedio se verifica con un rango ya especificado para cada convertidor,

de forma que se puede detectar fallas a nivel del bloque como ruido, falla en la alimentación, falla en un convertidor, etc.

Se va a manejar que los 12 bits de los convertidores codifiquen los 360° de movimiento de una unión, aunque haya códigos que nunca se den por los rangos restringidos de movimientos pero, que ayuda a que sea simple la adquisición de la posición y que sea general en cuanto a que no se está prespecificando algún rango a las uniones. Los 12 bits dan una resolución de $\pm$½ LSB es decir un $\pm 0.0122\%$ de la escala total, que si es 360° implica una resolución de 0.0439°. La máxima razón de cambio de la señal de entrada que soporta el ADC7802 es de 8 mV/μs y si 360° se da en un rango de 0 a 5 volts, implica que se puede adquirir la posición de una unión que se esté moviendo a una velocidad máxima de 1600 rev/s;lo que representa una velocidad pico increíblemente alta y sólo indica que no hay restricciones en la velocidad máxima de las uniones, por parte de la conversión analógica/digital.

5) Conversión digital/analógica

El bloque tiene la finalidad de generar seis señales de voltaje, para controlar igual número de motores de CD, de forma que las señales deben de tener una amplitud variable, así como poder invertir el sentido de giro del motor. Como no existen características exactas de los motores ha controlar, ya que la implementación se centra en el controlador y no en el manipulador; se decidió manejar un voltaje máximo en el rango de los 12 volts (aunque la fuente de los motores es capaz de dar un voltaje entre 1.25 y 21 volts) y una corriente de alrededor de 300 mA por motor, es decir se va a poder manejar motores de CD cercanos a los 3 watt de potencia; esto para manejar costos bajos al tener dispositivos de no mucha potencia, que simplifican la implementa-

ción al no requerir disipadores, más espacio interno en el controlador, protecciones por la inductancia de los motores, etc.

En aplicaciones de robótica[40] que involucran el control de motores de CD, es muy común el tener un convertidor PWM (modulación de ancho de pulso) con un puente tipo H, utilizando una fuente de directa; de forma que aumenta la eficiencia y se puede lograr un ancho de banda mayor que en los sistemas que utilizan amplificadores lineales, a demás la respuesta del motor es rápida y el rizo en el voltaje de armadura es pequeño, dando un menor calentamiento por armónicas y por pulsación en el par producido. El convertidor PWM utilizado es uno de CD a CD de bajada ("buck"), que al modular el ancho de pulso de una señal cuadrada; cambia el ciclo de trabajo de los semiconductores de potencia, variando el voltaje promedio de la carga.

Al tener un control digital, es preferible[41] que toda la etapa de manejo de los dispositivos de potencia también sea digital, pese a que el ancho de banda puede ser menor que en implementaciones analógicas. Por simplicidad la señal de reloj del propio microprocesador maneja la conversión y ésta tiene una resolución de 12 bits, a forma de un compromiso entre una buena resolución y una frecuencia de funcionamiento no muy baja.

Se implementó de forma discreta el bloque, para que tuviera las características exactas que se querían[42] y pese a que es una implementación algo extensa, el diseño es bastante simple y con la ventaja de un funcionamiento independiente del microprocesador.

---

[40] "The Electrical Engineering Handbook", editor R.C. Dorf, editorial CRC PRESS, 1993, 993, pp 728-730.

[41] "Concise International Encyclopedia of Robotics", editor R.C. Dorf, editorial John Wiley & Sons Inc., 1990, pág 14.

[42] Aunque existen controladores de motores de CD, como el HCTL_1100 de la Hewlett-Packard o el LM629 de la National S., que producen una señal de control PWM pero, ya con esquemas de control internos.

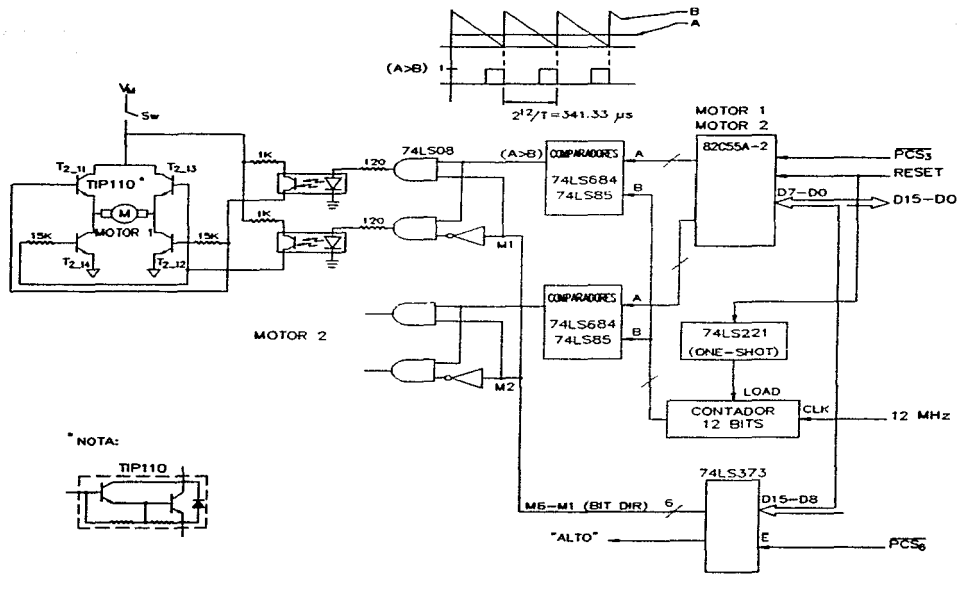


Fig. 5.13 Conversión D/A (motor 1 y 2).

La interfase del bloque de conversión D/A con el microprocesador es muy simple, se reduce a puertos de salida (tres 82C55A-2 y un 74LS373); en donde cada 82C55 controla dos motores y el 74LS373 maneja el sentido de giro de los seis motores. El tener una resolución de 12 bits en el voltaje promedio de los motores, utilizando los 82C55, implica compartir uno de los tres puertos (el puerto C) para dar los 12 bits a cada motor; como se puede ver en la figura 5.13 en

donde el puerto A y el puerto B manejan los bytes bajos y el puerto C los nibble altos de las palabras de control en cada 82C55; y es a nivel de la rutina de manejo de la conversión D/A, que se da el cambio de las dos palabras de control de 12 bits a los tres bytes que se necesitan en el 82C55. Aunque en la figura 5.13 sólo se muestre los dispositivos para los motores 1 y 2, para los demás motores es idéntico tanto el manejo como los dispositivos utilizados (ver apéndice E para el esquemático completo del bloque).

El manejo del sentido de giro está dado por el 74LS373, es decir un byte en la dirección 01301h, en el que cada sentido de giro de los seis motores está especificado por un bit (como se indica en el mapa de entrada/salida), de forma que un "1" da un sentido de giro antihorario. De los dos bits que sobran, uno no se utiliza y el otro sirve para controlar la iluminación del interruptor de paro de emergencia ("ALTO").

El funcionamiento interno de la conversión D/A se muestra en la figura 5.13 y como ya se dijo, por medio de tres 82C55 se proporcionan seis palabras de control de 12 bits cada una; a parte se tiene un contador también de 12 bits, corriendo continuamente una cuenta regresiva, con un reloj de 12 MHz, lo que resulta en una señal en forma de diente de sierra de los valores en binario de la cuenta y con una frecuencia fija de 2.92 KHz. El valor del contador se compara constantemente con cada una de las palabras de control, en bloques formados por un 74LS684 y un 74LS85 (comparadores de 8 y 4 bits respectivamente); de forma que de cada uno de los seis bloques de comparación, se obtiene una salida en lógica positiva, que se activa cuando la palabra de control es mayor que el valor del contador, lo que genera un tren de pulsos cuyo ciclo de trabajo es proporcional al valor de la palabra de control, como se ve en la figura. De esta forma se obtienen seis señales (PWM) con el ancho de pulso modulado por igual número de palabras

de control, que al ser de 12 bits implica un control sobre el ciclo de trabajo del 0% al 99.98% en incrementos de 0.0244%.

Cada señal PWM va a controlar un motor específico y como se muestra en la figura 5.13 para el motor 1, la señal PWM de ese motor se combina con el bit de dirección que le corresponde en dos compuertas AND, con el objetivo de activar a uno de los dos optoacopladores (4N25) y con esto controlar el sentido de giro del motor. Los dos optoacopladores sirven tanto para aislar los voltajes a nivel TTL de los voltajes en los darlingtons, como para aislar la tierra digital del neutro utilizado en el voltaje de los motores; y aparte activan a los darlington del puente tipo H, como se muestra en la misma figura, de forma que al activarse el fotodiodo de uno de los 4N25, éste satura al fototransistor del mismo 4N25 que a su vez activa a la pareja de darlingtons conectados a él, con un ciclo de trabajo dado por la palabra de control. Si se da un cambio en el bit de dirección se provoca que se utilice al otro optoacoplador y por tanto a la otra pareja de darlingtons, con el consecuente cambio en el sentido de giro del motor. Esta es la característica por la cual se eligió la configuración de puente tipo H para los darlingtons; es decir se adapta bien al control por medio de compuertas lógicas y no requiere de una fuente negativa para invertir el sentido de giro del motor.

Al manejar un motor de CD, se tiene el problema de manejar una carga inductiva; el puente tipo H utiliza darlingtons TIP110[43] que traen incorporado un diodo entre colector y emisor, que sirve de diodo volante para evitar problemas con la inductancia de la armadura del motor, de forma que como se muestra en la figura 5.13 mientras $T_{2\text{-}11}$ y $T_{2\text{-}12}$ están encendidos, al darse la

[43] Ver la hoja de datos: TIP110 en "Power Products Data Book", Texas Instrument, 1990.

conmutación por la modulación del ancho de pulso, la corriente creada por la inductancia circula por el diodo interno de $T_{2\text{-}13}$ y por $T_{2\text{-}11}$. De igual forma al encenderse $T_{2\text{-}13}$ y $T_{2\text{-}14}$, se utiliza el diodo interno de $T_{2\text{-}11}$ y a $T_{2\text{-}13}$.

La configuración mostrada en la figura 5.13 tiene los problemas asociados al uso de darlingtons en puentes tipo H: es decir se tienen voltajes relativamente altos en V_{CE} por lo que existen pérdidas de potencia en los darlingtons y el voltaje útil en la carga es menor que el nominal de la fuente y tiende a disminuir conforme aumenta la temperatura de operación. Se tiene calculada una pérdida de alrededor de los 3 volts en el voltaje útil en la carga con respecto a V_M y aunque se saturan ambos darlingtons durante su funcionamiento, no se saturan al mismo grado; de forma que $T_{2\text{-}11}$ o $T_{2\text{-}13}$ tienen un $V_{CE}(sat)$ mayor que el de $T_{2\text{-}12}$ o $T_{2\text{-}14}$ por lo que disipan una mayor potencia. Dado que el TIP110 (encapsulado TO-220AB) sin un disipador de potencia logra dar 2 Watt y se está restringiendo a cada motor a un máximo de 3 Watt de potencia, es decir una corriente alrededor de los 300 mA, que produciría una disipación en $T_{2\text{-}11}$ o $T_{2\text{-}13}$ de más o menos 0.8 Watt; garantizando el funcionamiento continuo, a potencia máxima, de los seis motores.

El utilizar los 82C55 para proporcionar las palabras de control puede causar problemas, ya que saliendo de RESET el 82C55 se configura como puertos de entrada, lo que da a los bloques de comparación una palabra de control máxima (0FFFh), activando los motores a su máximo, hasta que el procesador reconfigure al 82C55 como puertos de salida y de una palabra de control de apagado (0000h). Para lidiar con este problema, se utilizó un interruptor en la fuente de los motores, de forma que al apagar el controlador se da la indicación que se apague el interruptor de los motores; para que al encender el sistema el interruptor esté abierto y no haya problemas.

También se utilizó un 74LS221, (one_shot) como se ve en la figura 5.13, el circuito se dispara al inicio de RESET y durante tres a cuatro segundos mantiene una cuenta de 0FFFh en el contador, con lo que los bloques de comparación tienen una salida desactivada durante ese período, no importando que valor se de en las palabras de control; con lo que se tiene un tiempo más que de sobra para realizar la reconfiguración de los 82C55. Pese a que los 82C55 dan ciertas dificultades, logran un ahorro de espacio y conexiones con respecto a la opción de utilizar "latch" individuales de 8 bits y sus circuitos de decodificación.

Con respecto a otros controladores de motores de CD, como el HCTL-1100 que opera cerca de los 8KHz, se tiene una frecuencia de operación (2.92 KHz) algo lenta y el que esté en el rango audible hace que los motores operen con algo de ruido audible pero, se tiene un período de funcionamiento mucho menor que la constante de tiempo de la mayoría de los motores de CD, que usualmente está en el rango de los milisegundos, lo que sigue asegurando un comportamiento suave y continuo.

5.2 Implementación.

En la implementación de la arquitectura se buscó un buen funcionamiento, una confiabilidad adecuada y una buena presentación; tratando de mantener costos bajos y un tiempo de implementación no muy grande. Un aspecto importante es que desde un inicio se decidió implementar el prototipo del controlador, a fin de comprobar el diseño original y poder operar el control; dejando a un futuro todos los cambios y rediseños que se vayan planteando, una vez con la arquitectura funcionando. Es decir la implementación llega hasta el momento en que el diseño de la arquitectura está funcionando y se entiende que al ser un primer diseño, éste se puede mejorar, modificar y extender de acuerdo a las experiencias con el prototipo.
De las formas en que se pudo implementar la arquitectura, se optó por el "wire wrap" (alambrado por medio de bases especiales y alambre del 30AWG) para dar una implementación compacta, de buena presentación, con una confiabilidad buena y un funcionamiento según lo diseñado; los tiempos de implementación posiblemente no son los mínimos pero se cumplen los objetivos de implementación.

Se pensó en separar la fuente del resto del sistema, para concentrar el peso y la disipación de potencia; por lo que físicamente la implementación se reparte en dos chasises. Para la parte del sistema digital se tiene dos cajas de plástico unidas, cuyas dimensiones permiten tarjetas de 22.4 x 15.5 cm; en base a estas dimensiones se dividió la arquitectura en tres tarjetas, de forma que se minimizara el número de líneas de conexión entre tarjetas y tratando de agrupar por funciones a la implementación. La arquitectura quedó dividida en procesador 1, procesador 2 y conversión D/A; siendo la tarjeta del procesador 1, el bloque descrito en la sección 5.1.1, la tarjeta del

procesador 2, el bloque descrito en la sección 5.1.2, menos la conversión D/A que por ser extensa la implementación, se requirió de toda una tarjeta para ella.

Para evitar problemas causados por ruido, se cuidaron las líneas de RESET y del reloj (CLK) y en varios tramos se entrelazaron con tierra, para evitar el ruido externo y que fueran a inducir ruido a otras señales. También se utilizaron capacitores de desacoplo en cada circuito integrado (lo más cercano a la entrada de Vcc), con valor de 0.01μF para compuertas lógicas, 0.1μF para memorias, puertos y demás circuitos de mediana a alta integración y para los circuitos de conversión A/D se utilizó un arreglo especial de desacoplo y una fuente de +5V exclusiva de los convertidores ADC7802, como se muestra en los apéndices E y F. Para evitar que el ruido producido por los motores entrara a Vcc o a tierra lógica, se proporcionó el voltaje a los motores con otro transformador; dando un neutro diferente al de la tierra lógica.

La distribución de la alimentación a cada tarjeta, trató de que a cada circuito integrado llegaran los +5V que entran en el conector de la alimentación, esto por medio de varias líneas de Vcc y tierra lógica, con interconexiones entre ellas y conectando a cada línea un capacitor de 10μF para evitar la variación del voltaje en Vcc, por la operación de los circuitos lógicos y de las memorias. En la alimentación a cada tarjeta se utilizan varias líneas de tierra lógica, para tener una inductancia baja en el regreso de la corriente y con esto mantenerla lo más cercana a 0 volts.

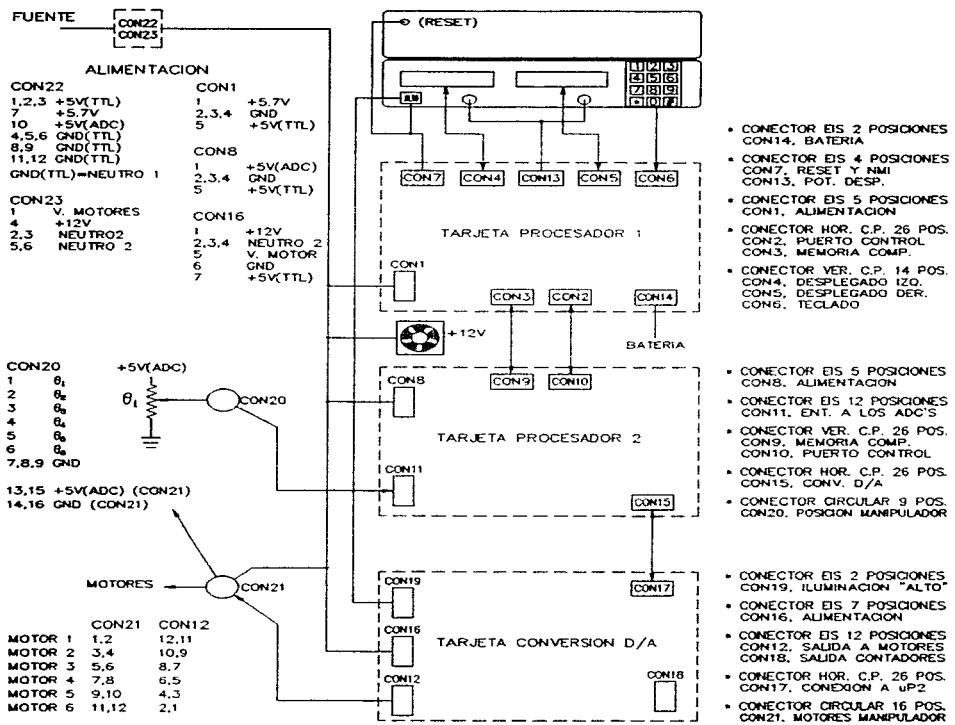


Fig. 5.14 Diagrama de conexiones de la arquitectura.

Pese a que en el chasis del controlador no hay dispositivos que sean de mucha potencia, el conjunto si representa una potencia grande; para el voltaje +5V(TTL), se consume alrededor de 3 A máximos, para el voltaje de los motores (V_m) se tiene unos 1.8 A máximos, para el voltaje +5.7V (memoria respaldada por baterías) se da unos 140 mA máximos y por último el +5V(ADC) requiere de unos 4.5 mA. Es por esto que se instaló un ventilador pequeño, para agilizar el movimiento del aire dentro del chasis del controlador y con esto disminuir los problemas asociados al calentamiento de los dispositivos, por la disipación de potencia del sistema en general.

Como se puede ver en la figura 5.14 todas las señales que van o vienen de una tarjeta lo hacen a través de un conector, al igual que la alimentación general del controlador y la particular de cada tarjeta; esto para que si en algún momento se requiere, se pueda separar a cada parte del sistema sin la necesidad de cortar cables o desoldar conexiones; ya sea a nivel de separar las tarjetas, la fuente del controlador o el manipulador del controlador.

La distribución de los componentes en las diferentes tarjetas se muestra en las figuras 5.15, 5.16 y 5.17; en el apéndice E se tienen los esquemáticos que comprenden a todo el sistema.

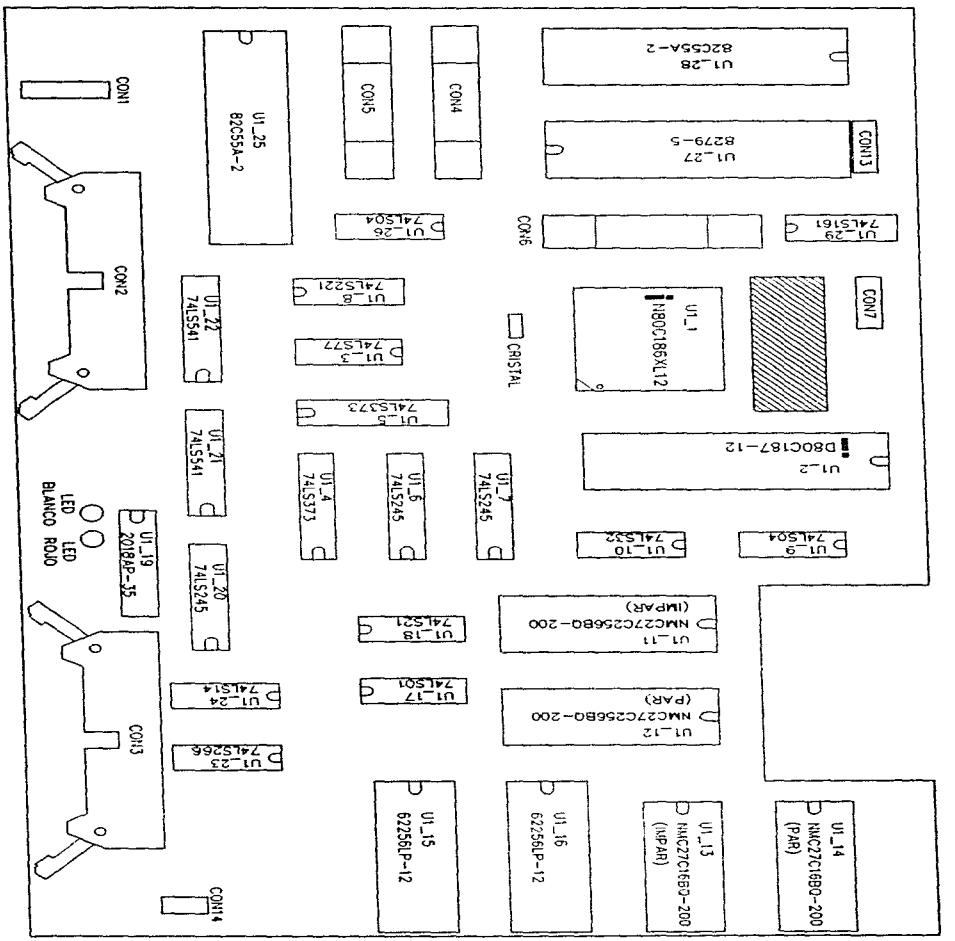


Fig. 5.15 Tarjeta procesador 1.

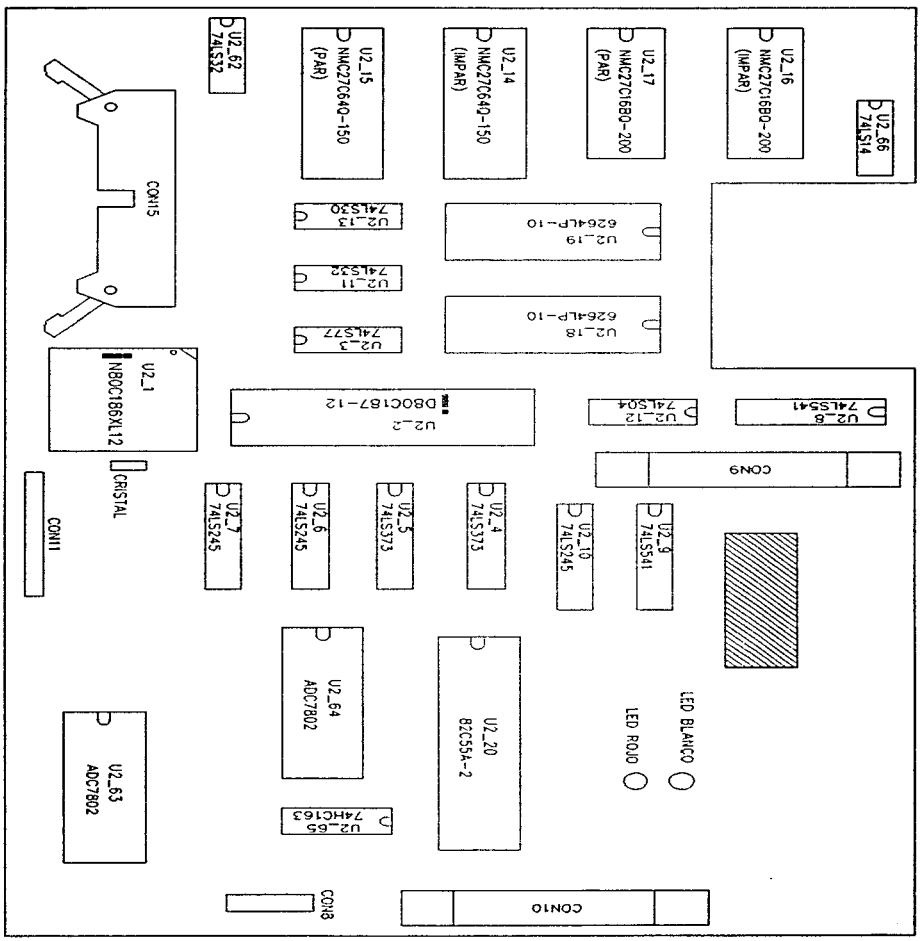


Fig. 5.16 Tarjeta procesador 2.

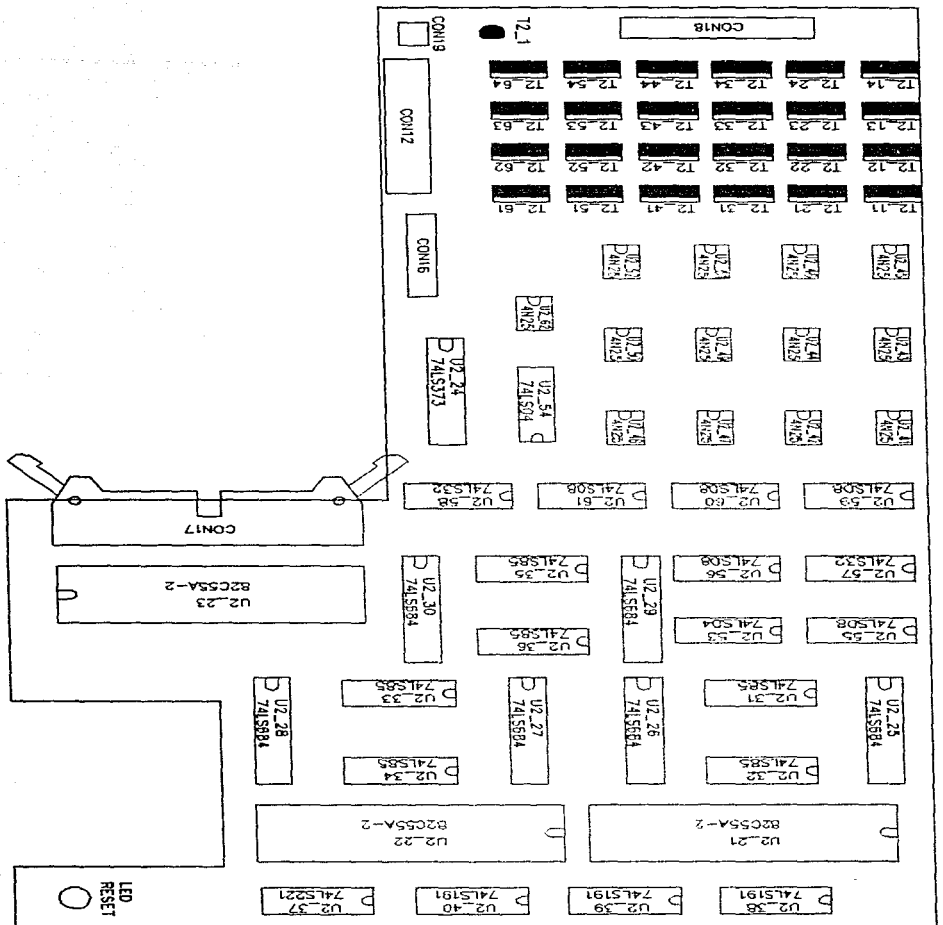


Fig. 5.17 Tarjeta conversión digital/analógica.

Conforme a toda la descripción de la arquitectura y su implementación, se sabe que hay una necesidad de varios voltajes y cada uno con ciertas especificaciones de potencia. Se diseño la fuente en base a esta necesidades y separando los voltajes que alimentan circuitos lógicos de los que alimentan a los motores y al ventilador; esto por medio de la utilización de dos transformadores, uno para cada grupo de voltajes. (ver el apéndice E, para el diseño detallado de la fuente). El voltaje de los motores se da por un regulador variable, de forma que se puede ajustar desde 1.25 a 21 volts; para dar cierta libertad al rango de motores de CD que se puede manejar y dando una corriente de hasta 2 A, repartida en los seis motores. El voltaje de +12V alimenta al ventilador, a los indicadores luminosos de la fuente y al indicador luminoso del interruptor de paro de emergencia del controlador.

Como parte de la implementación se requirió de un programa de prueba, para comprobar el funcionamiento de la totalidad de la arquitectura; desde memorias, coprocesador matemático, puertos, etc; así como de la comunicación entre procesadores, las conversiones D/A y A/D, tras hacer evidentes y ser un medio para solucionar los problemas de diseño, de implementación y de manejo de dispositivos; el programa de prueba sirvió de un inicio al programa definitivo, para asegurar la integridad de la arquitectura, cada vez que se arranque el sistema. En el capítulo 6 (programación), se describe las rutinas de prueba que se implementaron.

La forma en que se implementó físicamente la arquitectura se muestra en las figuras 5.18, 5.19, 5.20 y 5.21

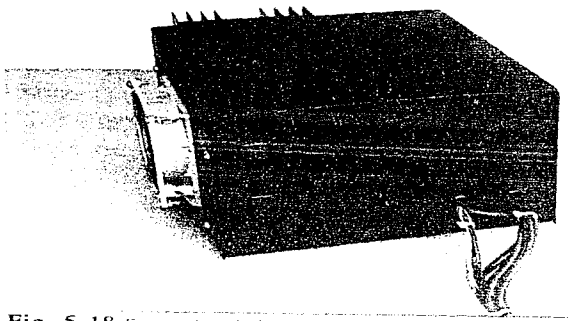


Fig. 5.18 Fuente de voltaje.

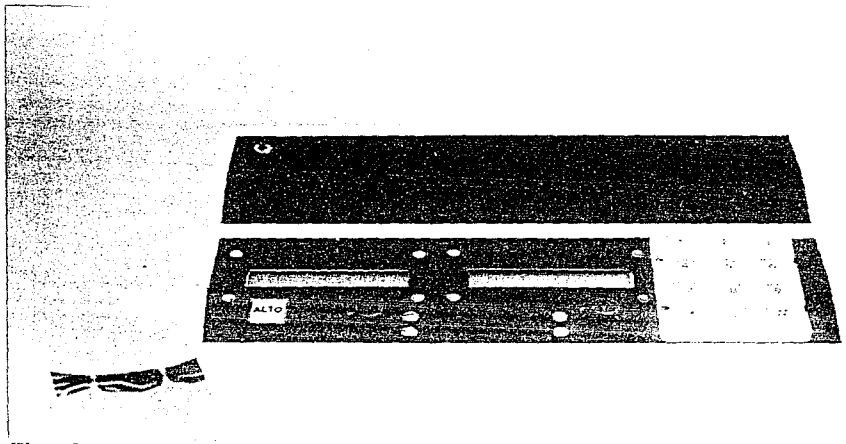


Fig. 5.19 Panel frontal del controlador.

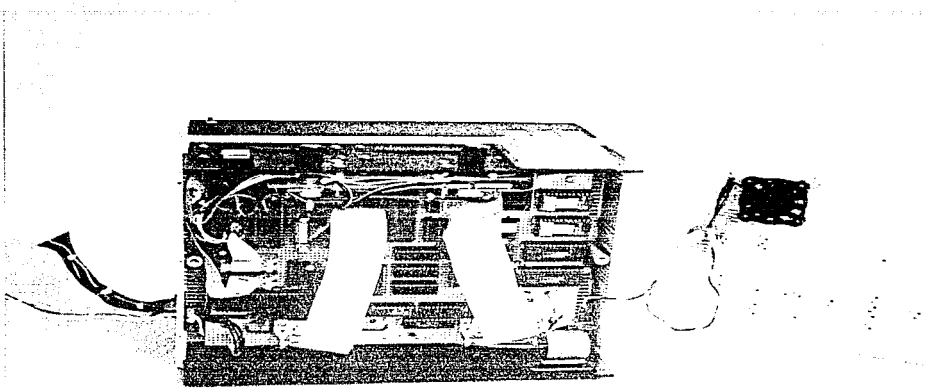


Fig. 5.20 Vista interna de [illegible]

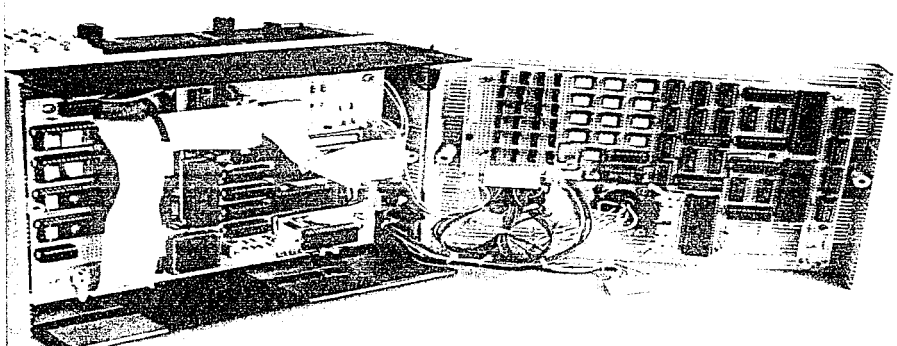


Fig. 5.21 Vista interna [illegible]

CAPITULO 6

DISEÑO DE LA PROGRAMACIÓN

El diseño de la programación para el controlador está definido por las características de la arquitectura, y por las tareas a realizarse. De inicio, por tener dos procesadores, se divide toda la programación en dos; de forma que se tienen dos programas corriendo en paralelo y que se sincronizan para lograr un objetivo: el control de un manipulador. Para simplificar la referencia a los programas de cada procesador, se denomina "Programa fuera de línea" al programa del procesador 1, ya que gran parte de la acción de este procesador es precisamente fuera de línea, es decir no en tiempo real. A la programación del procesador 2 se le denomina "Programa de control" ya que es la función más importante de este procesador.

Pese a que las tareas encomendadas a cada procesador son diferentes, la estructura de su programación es muy parecida y pueden caer ambas bajo un esquema general, como se muestra en la figura 6.1

Desde el inicio del diseño se pensó en la programación como todo un sistema de rutinas en ensamblador porque se debía tener el mejor uso posible de los recursos; mas el tamaño del sistema plantea muchos problemas que exigen una buena organización y estructura en los pro

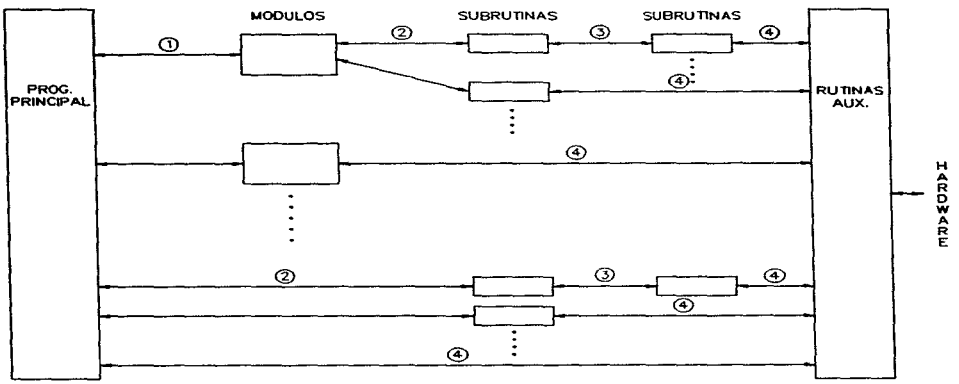


Fig. 6.1 Diagrama general de la programación.

gramas, para que conforme se avance en la programación no haya retrasos y replanteamientos de todo el sistema.

En la figura 6.1 se muestra, de una forma muy general, la estructura de la programación y se ven los diferentes tipos de rutinas y sus formas de comunicación entre ellas:

-Programa principal.

Esta rutina se encuentra en un archivo independiente y es el programa que organiza a todas las otras rutinas, además es en donde se declaran todas las variables globales que se utilizan. La idea tras el programa principal es el trabajar lo más estructurado posible, de forma que todo sea llamado por éste programa y no haya ligas o llamadas entre rutinas que no sean hechas dentro de éste. Junto con él se tiene un número de subrutinas locales, que sólo son llamadas por éste.

-Módulos.

Son rutinas extensas muy definidas en su tarea, y cada una de ellas forma un archivo independiente. Por ser muy específica su función sólo se utilizan subrutinas locales a estos procedimientos y no hay llamadas entre módulos, toda la comunicación entre éstos se da a nivel del programa principal.

-Subrutinas.

Son rutinas muy definidas que pueden ser llamadas desde el programa principal, un módulo u otra subrutina. El programa que llama a la subrutina es el encargado del manejo de la memoria para las variables locales a la subrutina, de forma que si se requiere de un espacio grande la rutina que llama a la subrutina debe de contar con la información de la ocupación de la memoria RAM y cuanta memoria puede asignar a la subrutina. Al regresar el control desde la subrutina, pierde significado todo el espacio de memoria temporalmente asignado a élla.

-Rutinas auxiliares.

Son programas reducidos que manejan los diferentes dispositivos de la arquitectura, y que permiten al resto de la programación utilizarlos fácilmente. Son accesibles desde cualquier programa y cada uno de ellos forma un archivo independiente.

Se implementan ciertos programas en archivos independientes por varias razones: por el tamaño del sistema es imposible tener un sólo archivo fuente y esto causaría muchos problemas, por lo que mejor es tener varios archivos fuente compilarlos por separado y después ligarlos para crear el ejecutable. Las rutinas auxiliares se ponen en archivos independientes porque se juntan todos en una biblioteca, para facilitar el compilar/ligar las diferentes partes del sistema.

En la figura 6.1 se muestran las diferentes formas en que los programas pasa/recibe datos[44]:

1) Comunicación programa principal con un módulo:

-Variables globales.

-Variables de comunicación a módulo.

-STACK, para mandar pocos bytes de información.

-Apuntador y un entero para localizar y dimensionar una zona de memoria que sirva para movilizar datos.

2) Comunicación con una subrutina:

-Variables en la memoria local del módulo que la llamó.

-Variables globales.

-STACK

-Apuntador y un entero.

3) Comunicación subrutina a subrutina:

-Variables globales.

-STACK

-Apuntador y un entero.

4) Comunicación con una rutina auxiliar:

-Variables de comunicación a rutina auxiliar.

-Registros.

-Apuntador y entero.

Para explicar en que consisten las diferentes formas en que se pasan datos entre las rutinas, es necesario definir la utilización de la memoria por parte de éstas.

---

[44] Los números de la figura 6.1 corresponde con los índices a continuación.

El código ejecutable de las rutinas se encuentra en EPROM y las rutinas que requieren de memoria RAM la van utilizando de forma ordenada, ascendente y sin huecos. Conforme se llama a las diferentes rutinas los datos que éstas requieran o vayan generando van llenando la RAM, y cuando se regresa el control al programa principal pierde significado la memoria utilizada por las diferentes rutinas; es decir que los datos que permanecen más tiempo se encuentran en una dirección más baja que los que son más volátiles. En la figura 6.2 se muestra la utilización de la memoria.

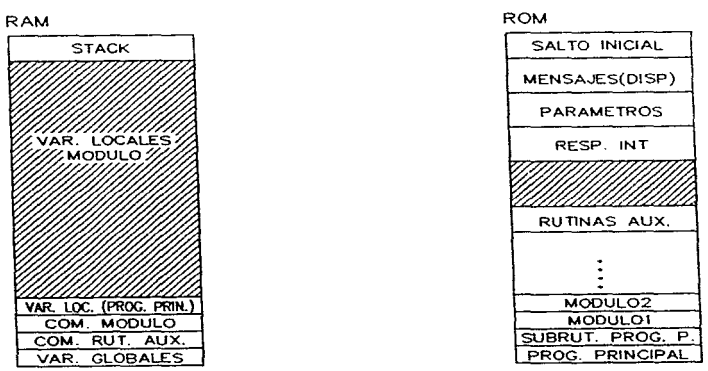


Fig. 6.2 Utilización de la memoria.

En la figura 6.2 se muestran los diferentes clases de variables utilizadas:

-Variables globales, las que tienen un significado y una dirección constante durante todo el funcionamiento de la programación, se utilizan para los parámetros del sistema, las variables de control, etc.

-Variables de comunicación con rutinas auxiliares, se define un conjunto de bytes y palabras para interactuar con las rutinas auxiliares y como se llama una rutina a la vez no hay una distinción entre los diferentes bytes de la comunicación.

-Variables de comunicación con los módulos, son variables para dar o recibir información adicional a los módulos: son al igual que las variables de comunicación con las rutinas auxiliares, un conjunto de bytes y palabras que no tienen un significado constante para todos los módulos si no que cada módulo los utiliza como mejor sea el caso.

-Variables locales al programa principal.

-Variables locales a los módulos. Como se puede observar en la figura 6.2 se deja la mayoría de la memoria RAM libre para su uso por parte de los diferentes módulos, y cada uno de ellos utiliza la memoria de una forma diferente con sólo ciertos aspectos en común: al ser programas autocontenidos se encargan del manejo de los datos considerando que toda la memoria libre es de ellos y al dejar de operar el módulo todas sus variables locales dejan de tener significado y sólo importan las variables destinadas a la salida de datos del módulo.

Las diferentes interconexiones entre rutinas (prog. principal, módulos, subrutinas y rutinas auxiliares) se dan principalmente a nivel de variables globales y variables específicas de comunicación, ya sea a módulos o a rutinas auxiliares; pero, se puede realizar una comunicación por medio del STACK si no son muchos los bytes involucrados o por medio de un apuntador y un entero, si es que se trata de muchos bytes de información, de forma que se apunta a una localidad de memoria en la cual están los datos o en la que se puede escribir los datos, y el entero sirve para dimensionar la zona de memoria utilizada.

Se estableció desde un inicio la forma de comunicación entre rutinas, tratando de abarcar todos los casos posibles, para asegurar que la estructura propuesta daba cabida a todo el diseño del sistema.

6.1 Programa fuera de línea (procesador 1)

Del capítulo 5 (Arquitectura), se sabe que el procesador 1 es el encargado de manejar el teclado, el desplegado, las funciones de comunicación con el usuario, la generación de trayectoria y el cálculo de la compensación anticipativa. De las tareas del procesador se considera de forma especial a la comunicación con el usuario, ya que esta determina el funcionamiento de la programación.

En la figura 6.3 se muestra el diagrama de flujo del programa principal del procesador 1 y éste refleja en gran medida la interfase con el usuario que se propone. Después de inicializar dispositivos y variables se realiza una prueba interna de casi todo el subsistema "Procesador 1" y dependiendo de los resultados de esta prueba se sigue adelante o si existe un problema grave, éste se comunica al usuario y se evita el funcionamiento del controlador.

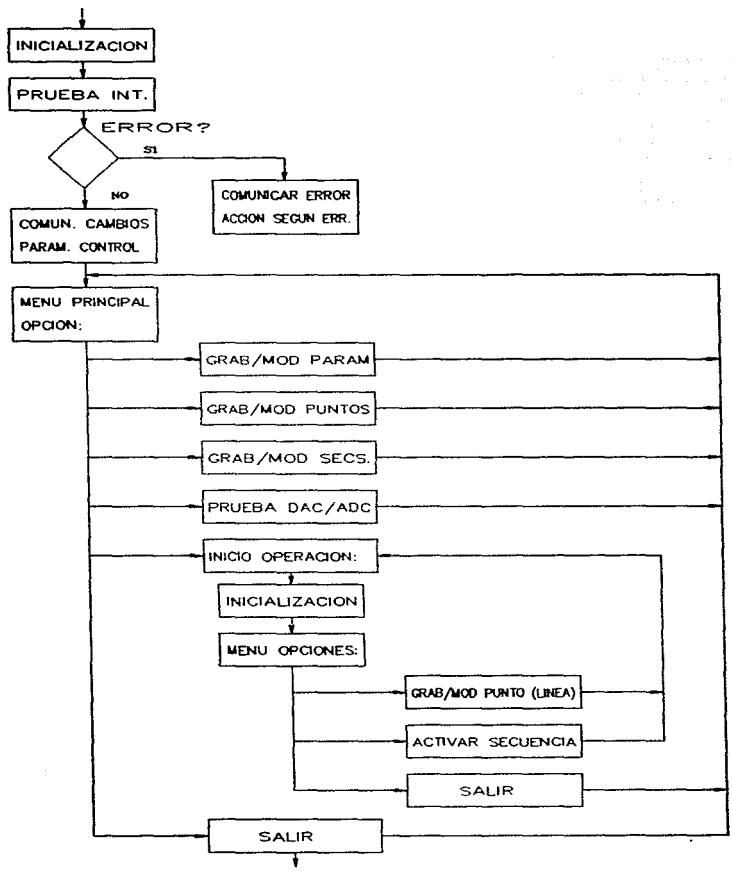


Fig. 6.3 Programa principal (programa fuera de línea).

Si todo está bien se inicia el funcionamiento del sistema, el cual se basa en menús o plantillas en las que se escoge opciones o se indica acciones a realizar. Se parte de un menú principal y se va recorriendo plantillas hasta llegar al inicio de operación, de forma que a través de los diferentes menús se puede grabar/modificar los parámetros del sistema, grabar/modificar puntos, grabar/modificar secuencias e iniciar operaciones.

Es decir que el manipulador va a poder ser operado al ejecutarse una secuencia de movimientos, la cual no es sino la especificación de ciertos puntos por los que se debe de pasar y el tiempo de recorrido que se debe de dar entre punto y punto. Con este funcionamiento se podría pensar que se tiene un control punto a punto pero, realmente se controla cada unión para seguir una trayectoria y es el propio algoritmo de generación de trayectoria el que para producir el comportamiento en el tiempo de las posiciones y velocidades a nivel de cada unión, sólo asegura el paso por ciertos puntos.

El procesador maneja todo el sistema de menús y al llegar al inicio de operaciones se cicla en ese bloque hasta que se acaba la secuencia, y es cuando vuelve a los menús para captar las futuras acciones que se deseen.

Descripción de los módulos utilizados:

-**Prueba interna.**

Esta rutina establece si todo está funcionando bien y se corre cada vez que inicia el programa. Por principio se prueban los displays, ya que toda la información de las pruebas se da a través de ellos y después se realizan pruebas sobre:

*La memoria RAM. Se leen 16 bytes de cierta zona de memoria y se compara con lo que tendría

que haber (ya que hay un respaldo de batería a la memoria RAM).

*Teclado. Se tiene que activar el teclado y se da un eco a los displays de forma que se pueden probar todas las teclas. Al llenar un renglón del display la prueba acaba.

*80C187. Se detecta la presencia del coprocesador.

*Puerto de control. Se realiza una transacción de dos bytes, en la que el código del mensaje indica que se está probando el puerto desde el procesador 1.

*Memoria ROM baja. Se lee una zona de memoria en la que se encuentra una cadena específica de bytes y se compara con lo que debe de ser.

*Memoria compartida. Se escribe en la memoria y después se lee lo que se escribió para probar el acceso a la memoria y la memoria en si.

*Se lee el resultado de la prueba del procesador 2 (RAM, 80C187, Memoria compartida, memoria ROM baja, ADC's, DAC's), por medio del puerto de control.

-**Módulo de Prueba DAC/ADC**. Este módulo sirve tanto para verificar el funcionamiento de los convertidores analógico/digital y digital/analógico, como para ajustar el funcionamiento del controlador con los motores y sensores de posición del manipulador.

Se escribe en cada canal de los DAC un valor en hexadecimal que corresponde al voltaje de salida y un bit para el sentido de giro, de forma que se puede controlar manualmente a los seis motores.

En cuanto a los ADC's se lee los diferentes canales y se despliegan los resultados en los displays, de forma que al moverse el manipulador se pueda tanto calibrar las posiciones cero de los potenciómetros, como su sentidos de giro; es decir se verifica tanto los ADC's como a los potenciómetros.

-Módulo grabación/modificación parámetros.

Por medio de esta rutina se puede hacer cambios en constantes o ganancias de los diferentes algoritmos utilizados, y es importante ya que la sintonización de los controladores es un proceso tardado que requiere de muchas modificaciones. También se puede modificar parámetros cinemáticos, dinámicos y de los relacionados con la generación de trayectoria. Todos los parámetros se guardan en su versión original en EPROM y se realiza una copia a RAM únicamente cuando falla el respaldo por batería a la memoria. Todos los cambios se van conservando en RAM y en el caso de los parámetros de control estos se mandan al procesador 2 cada inicio de operaciones y cuando hay modificaciones de parámetros.

-Módulo grabación/modificación puntos.

Antes de especificar un movimiento del manipulador, se graban los puntos por donde tiene que pasar. Los puntos se pueden grabar como coordenadas de unión, de forma que hay seis variables para cada punto en el espacio de trabajo.También se puede grabar en coordenadas cartesianas, implicando las siguientes variables: "x", "y", "z" y tres ángulos de orientación, junto con tres banderas de operación: brazo derecho/izquierdo, codo arriba/abajo y muñeca arriba/abajo.
Por simplicidad se limitó el número de puntos que se pueden grabar o definir a 10.

-Módulo grabación/modificación secuencia.

Una secuencia se crea al menos por dos puntos y se puso un límite máximo de 10, de forma que se especifica el orden en que el manipulador pasa por los puntos ya definidos y al llegar al último punto hay dos opciones: la primera es que el manipulador se detenga en el último punto y se de fin a la secuencia, y la segunda opción es que del último punto se regrese al inicio de la secuencia

y se establezca un ciclo de movimientos continuos hasta que se de un alto al manipulador. Como parte de la especificación de la secuencia, se da el tiempo de recorrido entre punto y punto, a manera de establecer la velocidad de movimiento deseada.

-Módulo grabación/modifiación punto (en línea).

Este módulo establece la comunicación con el procesador 2 (control), para realizar el movimiento del manipulador, de forma que se llama a las subrutinas de generación de trayectoria, dinámica y se establece el envio al procesador 2 de la posición deseada y del par de compensación para cada unión. Se mueve cada unión independientemente y al llegar a la posición deseada se activa la grabación de la posición de las seis uniones, por lo que un punto puede grabarse en línea, es decir, por medio del manipulador.

-Módulo activar secuencia.

El módulo establece la secuencia de movimiento establecida en la inicialización de "inicio de operaciones" y está formada por dos partes principales:

Primera parte, inicializaciones para la generación de trayectoria y compensación dinámica. Se realiza el proceso de cálculo de coeficientes de los polinomios para la generación de trayectoria y la inicialización de variables para el método de newton_Euler.

Segunda parte, se establece la comunicación con el procesador 2, de forma que se comienza un proceso en tiempo real, en el que se da la posición deseada y el par de compensación de acuerdo a la secuencia de movimiento escogida.

-Subrutina generación de trayectoria[45,46].

Para controlar por donde pasa el manipulador, a nivel de cada unión, se debe de convertir las trayectorias en coordenadas cartesianas en seis trayectorias de unión. Como las transformaciones que se obtuvieron en el capítulo tres (problema inverso, matriz jacobiana) son puntuales, es decir no se puede transformar la función de una trayectoria cartesiana; se utiliza un método de aproximación de curva con características:

a) Se debe de contar con suficientes puntos sobre la trayectoria a seguir, los puntos pueden estar grabados tanto en coordenadas cartesianas como en variables de unión.

b) Se transforma cada punto en coordenadas cartesianas en su equivalente en variables de unión, (problema inverso). En este momento se tiene una secuencia con "n-2" puntos en variables de unión y lo único que falta es aproximar una curva para que esta pase por todos los puntos definidos.

c) En el método dado por las referencias[45,46] se incorporan dos puntos adicionales, (un punto "2" y un punto "n-1") para que en total sean "n" puntos que se pueden unir por "n-1" polinomios cúbicos. Los polinomios van a tener coeficientes dados por un sistema de "n-2" ecuaciones lineales que se obtienen de la continuidad de posición, velocidad y aceleración en los puntos "nudo"[47] intermedios.

Se tiene "n-1" polinomios para cada unión, por lo que en total se tiene 6(n-1) polinomios, cada uno con cuatro coeficientes: es por eso que se limitó el número de puntos que definen una

---

[45] C.S. Lin, P.R. Chang y J.Y.S. Luh, "Formulation and Optimization of Cubic Polynomial Joint Trajectories for Industrial Robots",IEEE Trans. on Automatic Control, Vol 28, Núm. 12, diciembre 1983, pp 1066-1074

[46] J.Y.S. Luh y C.S. Lin, "Approximate Joint Trajectories for Control of Industrial Robots along Cartesian Paths", IEEE Trans. on Systems, Man & Cybernetics, Vol 14, Núm.3, mayo/junio 1984, pp. 444-450.

[47] Se denomina "nudo" o "nodo" a los puntos por donde se especifica que debe de pasar la trayectoria.

secuencia a 10.

Por las características propias del algoritmo, se puede dividir en dos su implementación:

1) Un proceso previo, en el que se calcula los coeficientes de los polinomios involucrados.

2) Un proceso en línea, en el que se genera la posición, velocidad y aceleración deseadas para cada una de las uniones.

-Subrutina dinámica.

En esta rutina se implementa el método de Newton_Euler para la obtención del comportamiento dinámico del manipulador.

Rutinas auxiliares utilizadas:

-DispDat.

Despliega datos a los displays.

Entradas[48]: [ra_w1] offset a localidad de la cadena a desplegar.

[ra_w2] segmento en donde está la cadena a desplegar.

[ra_w3] palabra de control, indica el display, el renglón, la posición y el número de caracteres a desplegar.

Salidas: No tiene salidas.

-DispCom.

Manda un comando de control a los display.

---

[48] [ra_w1] se refiere a la palabra 1 utilizada para la comunicación con una rutina auxiliar. [ra_b1] se refiere al byte 1 utilizado para la comunicación con una rutina auxiliar.

Entradas: BX, palabra de control en el que se indica el display y el comando[49] (8 bits) a mandar.

-ComPC.

Maneja la comunicación por puerto de control, la rutina soporta tres tipos de comunicación:

"2 bytes" es una transacción de dos bytes de ida y otros dos de vuelta, se utiliza para emitir un mensaje o un comando, y el valor de los cuatro bytes que se transfieren están dados en una tabla, es decir son fijos y determinan la validez de la transmisión.

"Info. salida" sirve para mandar un byte de información al procesador 2.

"Info. entrada" se le pide al procesador 2 que de un byte de información en específico.

Entradas: [ra_b1] código del tipo de comunicación.
[ra_b2] información (puede ser salida).

Salidas: [ra_b3] resultado de la transacción (bien/mal).
[ra_b4] a [ra_b8] códigos de error en los diferentes intentos de la transferencia.

La rutina realiza todo lo necesario para los tres tipos de transacción, sobre todo el trabajar con los errores y la recuperación de la transmisión (sincronización de los procesadores) si ocurre un error. El inicio, como se ve en la figura 6.4 es un "saludo" entre los procesadores y si por algo no ocurriera éste bien, se acaba la comunicación y se da como salida un error definitivo. Si se

[49] Ver apéndice F, para la lista de comandos de los displays.

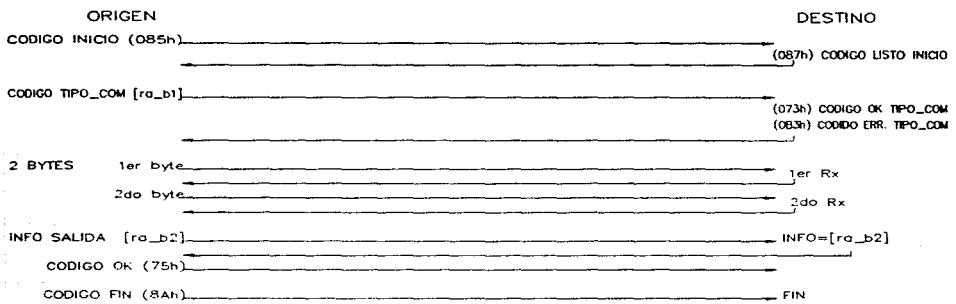


Fig. 6.4 Comunicación por puerto de control.

pasa del saludo inicial, la rutina permite cinco errores desde dar el tipo de comunicación, hasta llevar acabo la comunicación; y ante cada error se reinicia la rutina para sincronizar los procesadores y lograr la comunicación dentro de la misma llamada que comenzó la comunicación. Una vez iniciada la comunicación sólo hay dos códigos para acabarla, con lo que se evita que un procesador se quede dentro de la rutina de comunicación mientras que el otro esté ya fuera de la rutina.

"ComPC" por medio de [ra_b3] da el resultado de la transacción, y con esto el programa que provocó la comunicación sabe si ésta se logró o no y por medio de los códigos de error se obtiene los errores que impidieron la comunicación.

-ComMC.

La rutina da todo el soporte para escribir/leer una cadena de bytes a la memoria compartida.

Entradas: [ra_b10] tipo comunicación (escritura/lectura).

[ra_w1] segmento fuente de la cadena.

[ra_w2] índice a la cadena.

Salidas: [ra_b11] resultado de la transferencia (bien/mal).

[ra_b12] código de error (si hubo).

Se utiliza una variable global [uso_mc] en cada procesador a manera de semáforo y para prevenir el uso simultáneo de la memoria compartida. Se utiliza la rutina ComPC para leer el byte [uso_mc] del procesador 2 y si indicara que no está en uso la memoria compartida, se continua con la rutina de forma que se activa [uso_mc] para indicar que el procesador 1 está haciendo uso de ésta. Al finalizar la transferencia, se manda un mensaje por puerto de control, indicando al procesador 2 que:

Se escribió en la memoria compartida (MC) y por tanto que se requiere que lea la MC.

Si se leyó la MC entonces se le indica al procesador 2 que ya se leyó lo que ese procesador hubiera mandado por memoria compartida.

Por último se desactiva [uso_mc] para liberar la memoria compartida.

-ConvAscii.

Rutina que convierte a ASCII un byte binario.

-ConvFloatAscii.

Conversión de un número en precisión sencilla a ASCII.

Por la magnitud de los números con los que se va a trabajar, sólo se va a requerir cinco dígitos enteros y ocho decimales más signo y punto decimal. De los cuatro bytes de la precisión sencilla,

se pasa a 15 bytes en ASCII con representación de punto fijo, con el objetivo de desplegar un número de punto flotante en los displays.

-ConvAsciiFloat.

Rutina que convierte a punto flotante un número en ASCII de punto fijo.

La conversión se realiza al multiplicar cada número ASCII por su posición según el sistema decimal ($1x10^n$), empezando por 10,000 y disminuyendo hasta 10^{-8}, con lo que se obtiene un número en punto flotante. La rutina requiere del 80C187.

-Retardo.

Entrada: DL

Por medio del registro DI se obtiene un retardo de 39 ms por cada unidad en DL.

Respuestas a interrupción:

-ResPC (INT0)

La rutina da respuesta a la interrupción INT0, que se generada por un llamado por puerto de control. Se cumple con todo el protocolo de comunicación establecido para "ComPC", para que la interacción entre los procesadores se de la manera más fácil y sin interferencia con su funcionamiento.

Por medio de los códigos de comunicación, se sabe el tipo de comunicación que requiere el otro procesador y se actúa en concordancia es decir, si el procesador 2 pide un determinado byte de información se lee de memoria y se pone en el puerto de control; si se está recibiendo un byte información se escribe en la localidad correcta la información recibida, etc.

Como salidas se tiene a variables globales específicas a esta rutina, en las cuales el procesador puede saber si hubo errores en las transferencias requeridas por el procesador 2 y de que

naturaleza fueron estos errores.

-NMI (ALTO)

Al activarse el botón de "ALTO" se genera una interrupción NMI, la cual provoca la atención por esta rutina. Se confirma el alto del programa y se pregunta por la siguiente acción a realizar, la cual puede ser reanudar el programa en el punto exacto en donde se estaba antes de la interrupción o cancelar el movimiento y regresar al menú principal.

Durante el período de espera, el procesador 2 también está en la rutina a esta interrupción y mantiene al manipulador detenido.

-INT1 (TECLADO)

Ante esta interrupción se lee la cola del 8279 y se convierte el código de la tecla al código ASCII correspondiente, el resultado se escribe en la variable global [tecla] y se sale de la rutina.

6.2 Programa de control (procesador 2)

En la figura 6.5 se muestra el diagrama de flujo del programa principal del procesador 2. Después de inicializaciones y de una prueba interna el procesador entra a un ciclo de espera, hasta que el procesador 1 pide los resultados de la prueba interna; de los resultados de ésta y de la condición de operación de "Procesador 1" se decide si se evita la operación del manipulador y por tanto se detiene al procesador 2 o se sigue adelante con el programa principal.

El programa de control depende mucho de las acciones en el procesador 1 y la mayoría del tiempo se está a la espera de comandos provenientes de ese procesador; de forma que los diferentes comandos que se pueden recibir son:

-Salir, terminar con la operación del sistema.

-Prueba DAC/ADC en la que se llama al módulo "Prueba DAC/ADC".

-Cambios parámetros, en la que se afecta la copia de los parámetros relativos a los controladores de unión, que se encuentra en este procesador.

-Inicio de operaciones, en la que se entra al ciclo del algoritmo de control, (controlado por el período de muestreo):

*Leer del procesador 1: la trayectoria deseada y el par de compensación anticipativa para cada unión.

*Lectura de los convertidores A/D, (posición de cada unión).

*Cálculo de la señal de control para cada motor.

*Escribir en los puertos de los convertidores D/A.

*Verificar la comunicación con el procesador 1, por cualquier comando que afecte la secuencia de movimiento.

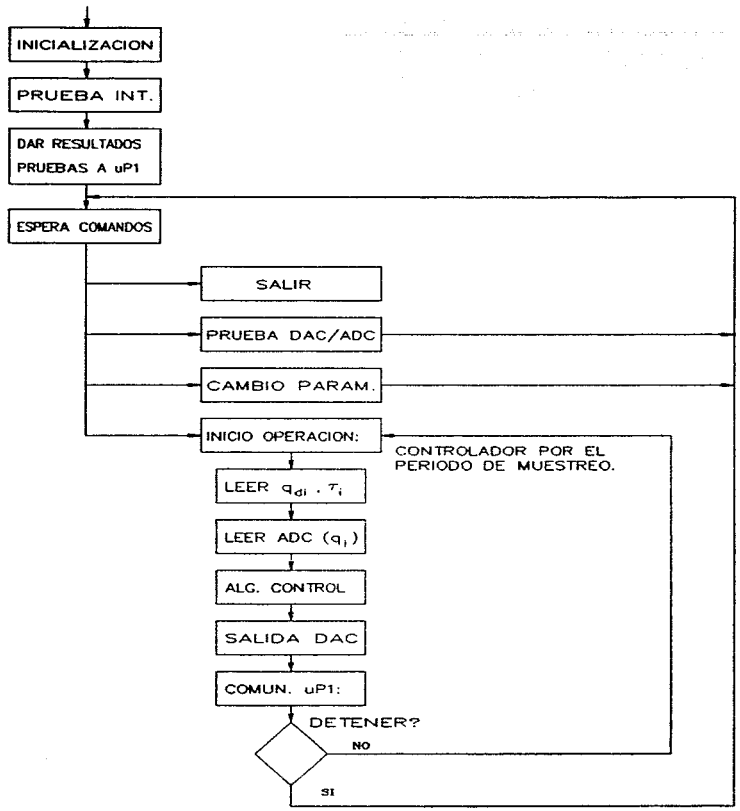


Fig. 6.5 Programa de control.

Como se ve en la figura 6.5 la tarea principal del procesador 2 es el control de cada unión y se ignora la trayectoria específica que está siguiendo el manipulador, tan sólo se reciben las posiciones, velocidades y aceleraciones deseadas para cada unión junto con los pares de compensación pero se ignora si se está moviendo el manipulador manualmente a un punto, si se esta grabando en línea un punto o si se está ejecutando una secuencia de movimiento ya establecida.

Módulos utilizados:

-Prueba interna.

Al igual que en el programa del procesador 1, se tiene diversas pruebas a los dispositivos dentro de la tarjeta del procesador 2:

*Prueba de la memoria RAM, en la que se escribe y después se lee una cadena de bytes para determinar el buen funcionamiento de la memoria.

*Prueba del 80C187, determina la presencia del coprocesador.

*Prueba de memoria EPROM baja, se lee cierta localidad de esa memoria y se compara con lo que debería tener.

*Prueba ADC's. Esta prueba es independiente del módulo de prueba DAC/ADC y tan sólo lee el cuarto canal de cada ADC7802, el cual está conectado a un diodo zener de 3.3 volts. Se realizan varias lecturas y se obtiene un promedio el cual, si está dentro de un rango ya predeterminado para cada ADC7802, se pasa la prueba.

*Prueba DAC's, esta prueba al igual que la de los ADC's es independiente del módulo del prueba DAC/ADC. Se prueban los puertos de los 8255, para comprobar que están programados correctamente y por tanto que se pueden leer/escribir y es estable su programación.

-Prueba DAC/ADC.

Este módulo se encarga de toda la comunicación con el procesador 1 de forma que se pueda llevar a cabo la prueba descrita en la programación del procesador 1. En esta prueba se manejan los DAC desde el teclado del sistema y se puede especificar un voltaje dado a cada motor, con lo que se maneja manualmente cada motor. En cuanto a los ADC's se leen continuamente los seis canales y la información se manda al procesador 1 por medio de la memoria compartida.

-Cambio de parámetros.

Al inicio del funcionamiento del sistema se hace una copia de los parámetros utilizados por el procesador 2, ya que el almacenamiento de todos los parámetros se encuentra centralizado en la memoria del procesador 1. Al momento en que se varía uno de los parámetros de control, se recibe el comando de llamar a este módulo y lo que se realiza es toda la comunicación para que quede actualizada la lista de parámetros situada en la memoria del procesador 2.

-Algoritmo de control.

El módulo implementa el algoritmo de control desarrollado en el capítulo 4 y como resultado del módulo se obtiene la salida de control, un voltaje, para cada unión.

Cuando se encuentra el programa controlando las uniones se llaman a varias subrutinas:

-Subrutina leer trayectoria deseada y par de compensación.

-Subrutina leer posiciones y escribir señal de control.

Rutinas auxiliares:

-ComPC

Rutina que maneja la comunicación por el puerto de control. Es la rutina que inicia la comunicación y la rutina RespPC es la rutina que responde al llamado inicial de ComPC.
Es idéntica a la rutina del procesador 1 con sólo diferentes códigos de comunicación, ya que al iniciarse la comunicación en este procesador existen otros comandos e informaciones que se pueden mandar y recibir.

-ComMC

La rutina maneja todos los aspectos del uso de la memoria compartida y es idéntica a la rutina del procesador 1, tan sólo con algunos cambios en los códigos de comunicación.

-LectAdc

Rutina de lectura de los ADC's.

Entrada: [ra_b1] número de canal.

Salidas: [ra_w1] Adc_alto.

[ra_w2] Adc_bajo

[ra_b2] resultado lectura ADC's.

[ra_b3] error (si hubo).

La rutina convierte los mismos canales de los dos ADC7802, por eso hay dos palabras de salida. El Adc_bajo da los canales 1,2,3 y el Adc_alto da los canales 4,5 y 6, en donde la numeración es del conector circular exterior.

-EscDac

Rutina para escribir a los registros del bloque de conversión digitalanalógica. Las entradas a esta rutina son las variables globales [motor1], [motor2], ..., [motor6] y no hay variables de salida.

-Retardo

Entrada: DL

La rutina produce un retardo de 39 ms por cada cuenta en DL.

Respuestas a interrupción:

-NMI

Al darse NMI se debe de provocar un paro total en el movimiento del manipulador. Por medio del puerto de control se recibe un código, para seguir el movimiento o ir al menú principal en espera de otro comando.

La lectura/escritura del puerto de control se realiza "manualmente" es decir sin la intervención de la rutina ComPC/RespPC.

-INT0

Esta interrupción es generada directamente por el procesador 1, cuando ya se escribió en memoria compartida la trayectoria deseada y el par de compensación. La rutina lo que hace es copiar de la memoria compartida a la memoria del procesador la información e informar a la subrutina que maneja la lectura de esos datos, que la información ya está disponible. Durante el funcionamiento del manipulador se tiene varias interrupciones de esta clase, ya que la cantidad de datos a transferir entre procesadores es bastante.

-INT1

Un llamado por puerto de control genera una interrupción INT1 y la respuesta a esta interrupción es la rutina RespPC, la cual es idéntica a la que se encuentra en el procesador 1, tan sólo se tienen diferentes códigos de comunicación ya que la comunicación se establece desde este procesador.

CAPÍTULO 7

PRUEBAS

La validación del trabajo realizado se divide en tres partes: pruebas a la arquitectura, al diseño de la programación y al algoritmo de control. Las pruebas a la arquitectura forman, lo que en el capítulo 6 (diseño de la programación), se llamó módulo de prueba interna y se compone de acciones muy concretas para determinar el estado de los componentes del sistema. El diseño de la programación se verificó por medio de la implementación de todas las rutinas descritas en el capítulo 6, menos las rutinas auxiliares, de forma que se pudo apreciar el funcionamiento de cada una de ellas y como se comporta el conjunto de toda la programación.

Para probar el algoritmo de control propuesto en el capítulo 4 (control), se utilizó el manipulador de dos GDL, que se encuentra en el "Laboratorio de Control" de la División de Estudios de Posgrado de la Facultad de Ingeniería. Este manipulador se controla desde una computadora y la obtención de datos experimentales está muy bien definida, lo que lo hace ideal para probar algoritmos de control.

7.1 Pruebas de la arquitectura.

Las diferentes pruebas contenidas en el programa se describen en el capítulo 6, y los resultados obtenidos son condiciones de operación o de no operación y se reflejan en el propio sistema. A lo largo de varias pruebas se aseguró el funcionamiento de todas las partes de la arquitectura y se logró una buena sincronización en la operación de los dos procesadores, controlando el funcionamiento de varios motores de corriente directa. La prueba sobre los motores se centró principalmente en la generación de voltajes con diferentes formas de onda para cada motor y unos resultados se muestran en las figuras 7.1, 7.2, 7.3 y 7.4 en donde se muestra una señal cuadrada y una señal constante.

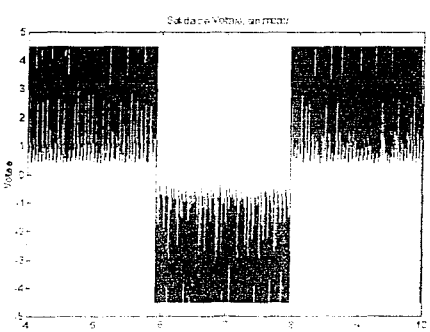


Fig. 7.1 Generación de una señal cuadrada.

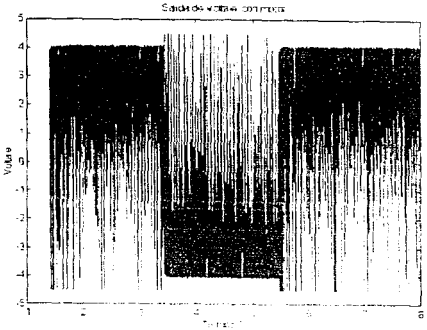


Fig. 7.2 Generación de señal cuadrada.

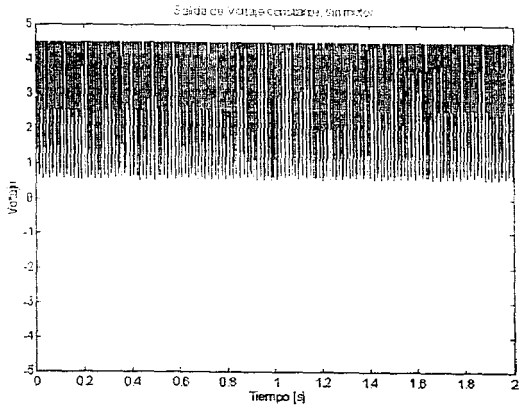


Fig. 7.3 Señal de voltaje constante.

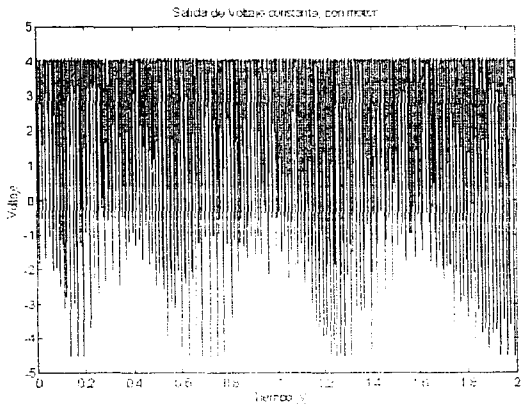


Fig. 7.4 Señal de voltaje constante.

Los convertidores A/D se probaron de la siguiente forma: utilizando el cuarto canal de cada convertidor ADC7802 para leer un voltaje fijo de 3.3 volts, se obtiene una lectura que es comparada contra un rango ya prestablecido para cada convertidor, y si esta se encuentra dentro del rango se pasa la prueba y de lo contrario se notifica del fallo de la prueba. Los rangos para cada convertidor son:

Convertidor 1 valor máximo 0AB8h = 3.3504 volts
valor mínimo 0A8Eh = 3.2992 volts

Convertidor 2 valor máximo 0AC5h = 3.3663 volts
valor mínimo 0A9Bh = 3.3150 volts

Estos valores se obtuvieron como resultado de un análisis estadístico, en el que se promediaron muchas lecturas del canal número 3 de cada convertidor: el rango está formado por $\bar{x} \pm 3 \cdot \sigma_n$,

es decir que el rango se centra en el promedio de las lecturas y se tiene una amplitud dada por seis veces la desviación estándar de la muestra. Suponiendo que las lecturas se comportan como una distribución normal, se tiene una probabilidad 0.99 de que una lectura esté dentro de este rango, y para asegurar el buen funcionamiento de la prueba, se toman 32 lecturas de cada canal, se promedian y es este valor el que se compara con los rangos dados anteriormente.

7.2 Prueba del diseño de la programación.

Se realizó un programa[50] en "C" con base en la estructura descrita en el capítulo de la programación, y se implementaron todos los módulos que no están directamente relacionados con el hardware de la arquitectura, es decir que no se implementaron[51] las rutinas auxiliares y el módulo de prueba interna.

El programa principal, el que llama a todos los módulos, también tiene la estructura descrita en el capítulo 6 y por tanto la interfase con el usuario es por medio de menús.

7.2.1 Resultados de las pruebas:

Se realizaron varias pruebas y para los parámetros listados en la tabla 7.1 se obtienen (véase el capítulo 4) las ganancias[52], dadas en la tabla 7.2, para el control de las seis uniones con un período de muestreo 10 ms.

---

[50] El listado de todas las rutinas del programa en "C" se encuentra en el apéndice G.

[51] La implementación de las rutinas relacionadas con el hardware de la arquitectura se realizó en la prueba a la arquitectura.

[52] En el apéndice D se listan otros parámetros importantes para la simulación del manipulador.

UNION	ζ	ω_n	γ	Z_{ob}
1	0.5	20	3	0.0001
2	0.5	20	3	0.001
3	0.5	20	3	0.001
4	1	5	3	0.001
5	1	5	3	0.001
6	0.6	25	3	0.001

Tabla 7.1 Parámetros de control.

UNION	K_1	$K_{2,1}$	$K_{2,2}$	K_e
1	23.2600	1.948E+02	1.013E+01	9.874E+01
2	35.1800	2.945E+02	1.470E+01	9.782E+01
3	8.81900	7.383E+01	3.628E+00	9.750E+01
4	0.08365	3.905E+00	4.359E-01	9.703E+01
5	0.08365	3.905E+00	4.359E-01	9.703E+01
6	4.91400	3.479E+01	1.362E+00	9.703E+01

Tabla 7.2 Ganancias del controlador de cada unión.

Con las ganancias anteriores y los parámetros dinámicos listados en la tabla 7.3 (véase el apéndice D para la descripción de cada uno de los parámetros listados en la tabla); donde los parámetros del elemento seis, implican un cubo de 5 cm de lado y masa 50 gr como carga.

Se realizó una simulación del comportamiento de las seis uniones y los resultados obtenidos se muestran en las figuras 7.5 a la 7.28

UNION	r [m]	h [m]	ρ [Kg/m^3]
1	0.05	0.1	1600
2	0.04	0.2	2500
3	0.05	0.25	1600
4	0	0	0
5	0.025	0.06	5000
6	a_n=0.05 b_n=0.05	l_n=0.05	400

Tabla 7.3 Parámetros dinámicos.

La secuencia utilizada en la simulación es:

SEC(3): P0 -P1 -P2 -P3 -P4 -(FIN)

Con tiempos: P0 -P1 1.0 seg

P1 -P2 1.5 seg

P2 -P3 2.0 seg

P3 -P4 2.5 seg

En donde los puntos nodo de la secuencia están dados en variables de unión y son:

P0= [0°,-45°, 0°, 0°,90°, 0°]

P1= [30°,10°,20°,10°,120°, 0°]

P2= [40°,80°,40°,40°,90°, 0°]

P3= [70°,85°,20°,50°,80°, 0°]

P4= [120°,90°,0°,10°,75°, 0°]

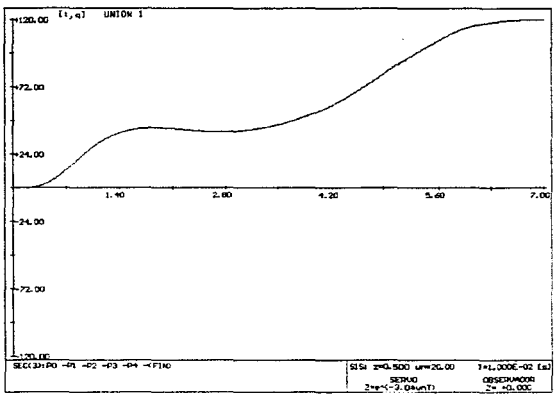


Fig. 7.5 Posición unión 1 [°]

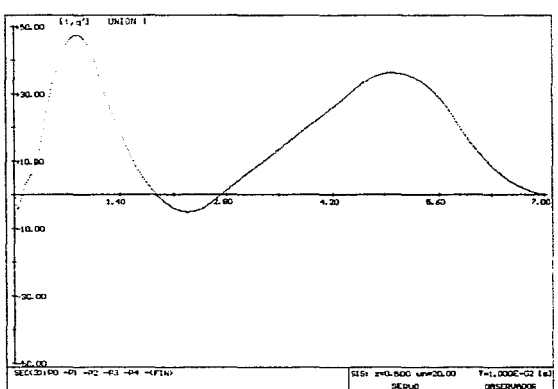


Fig. 7.6 Velocidad unión 1 [°/s]

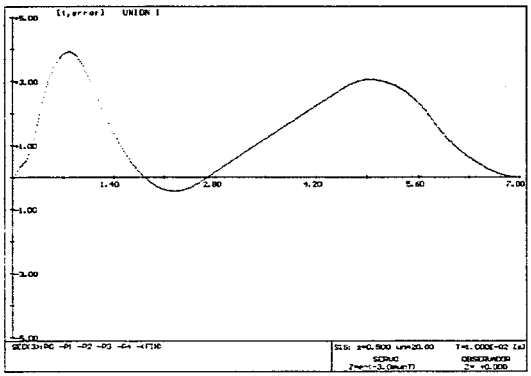


Fig. 7.7 Error de posición [°] unión 1

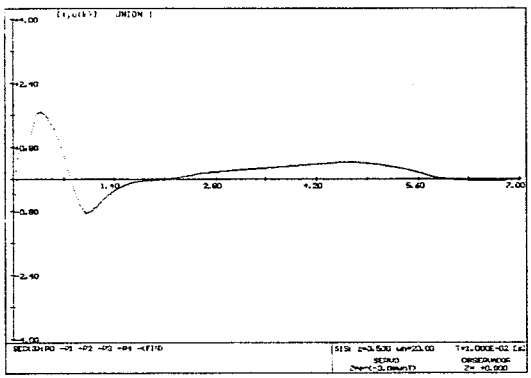


Fig. 7.8 Señal de control [Volt] unión 1

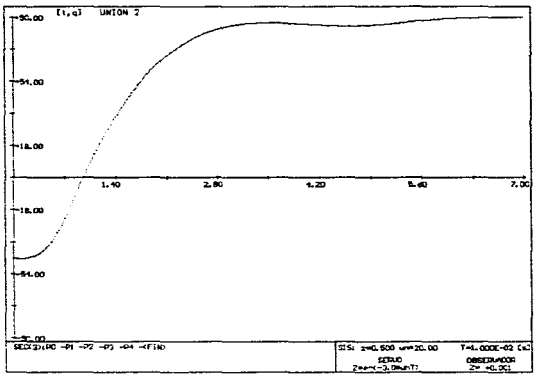


Fig. 7.9 Posición unión 2 [°]

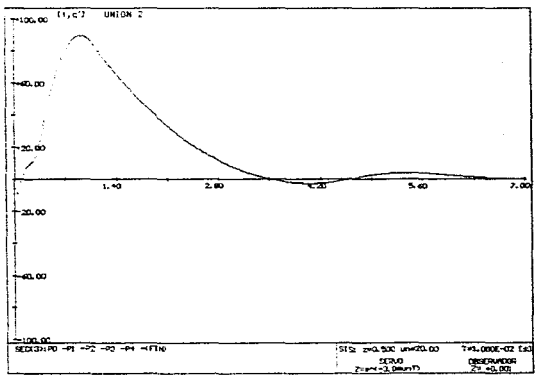


Fig. 7.10 Velocidad unión 2 [°/s]

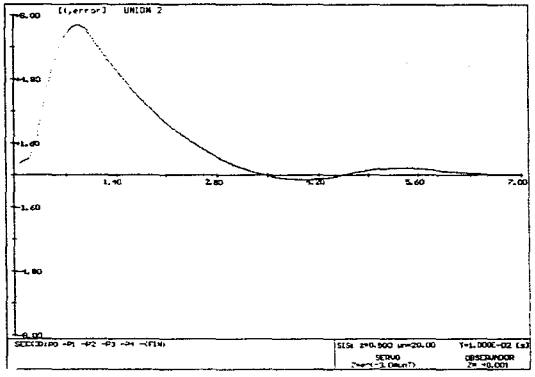


Fig. 7.11 Error [°] posición unión 2

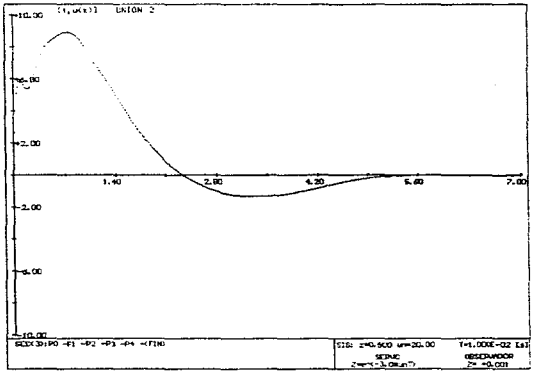


Fig. 7.12 Señal control [Volt] unión 2

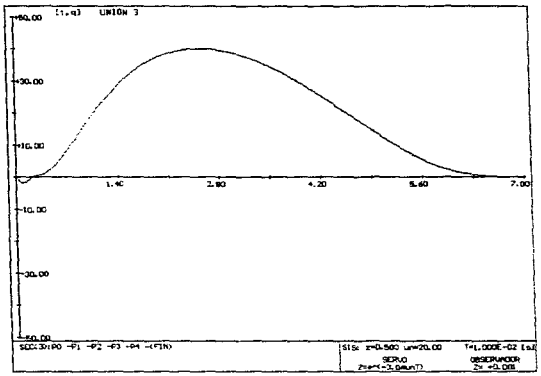


Fig. 7.13 Posición unión 3 [°]

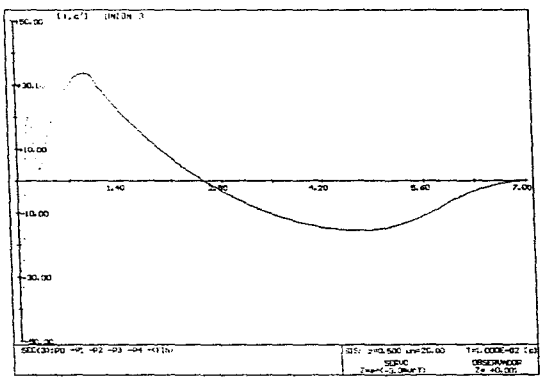


Fig. 7.14 Velocidad unión 3 [°/s]

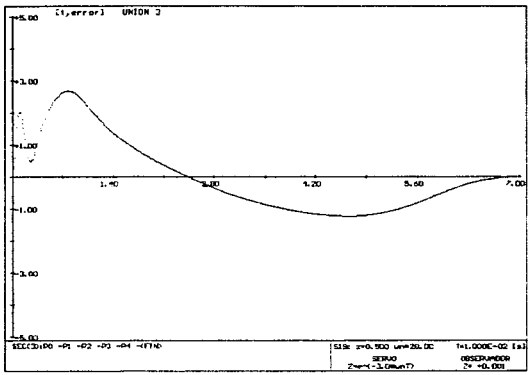


Fig. 7.15 Error posición [°] unión 3

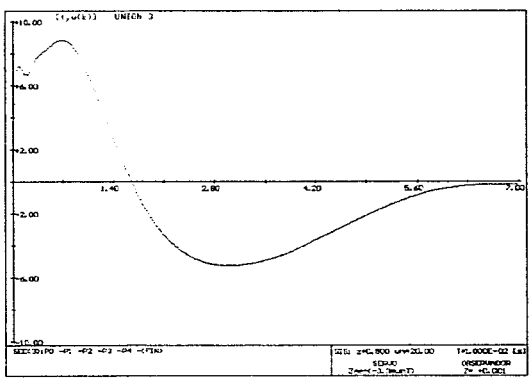


Fig. 7.16 Señal control [Volt] unión 3

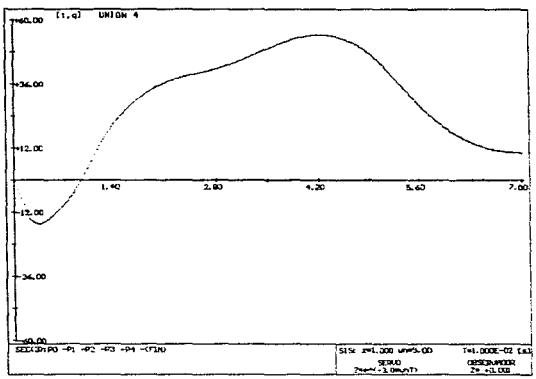


Fig. 7.17 Posición unión 4 [°]

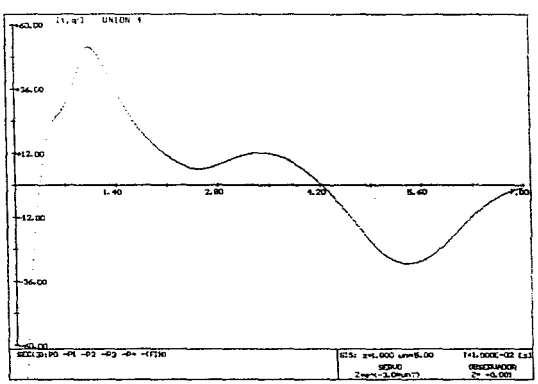


Fig. 7.18 Velocidad unión 4 [°/s]

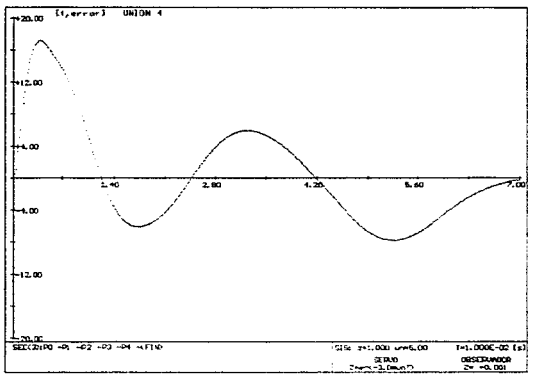


Fig. 7.19 Error de posición [°] unión 4

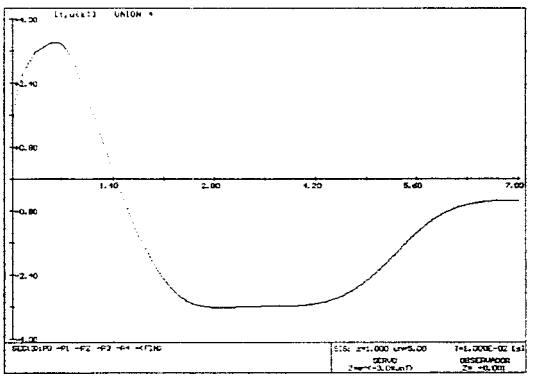


Fig. 7.20 Señal control [Volt] unión 4

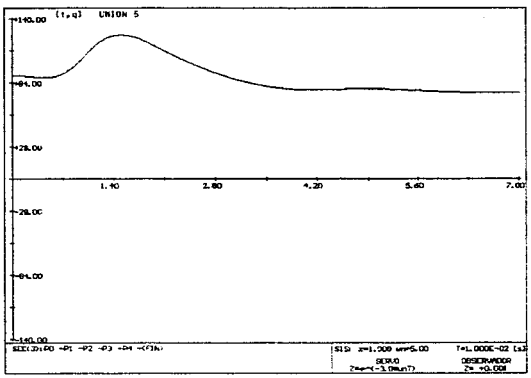


Fig. 7.21 Posición unión 5 [°]

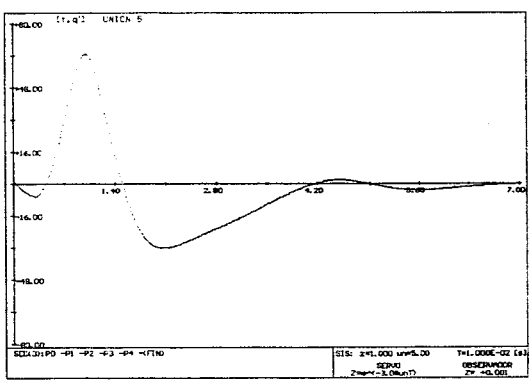


Fig. 7.22 Velocidad unión 5 [°/s]

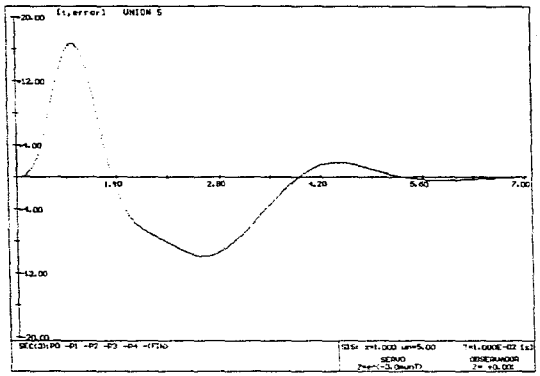


Fig. 7.23 Error [°] posición unión 5

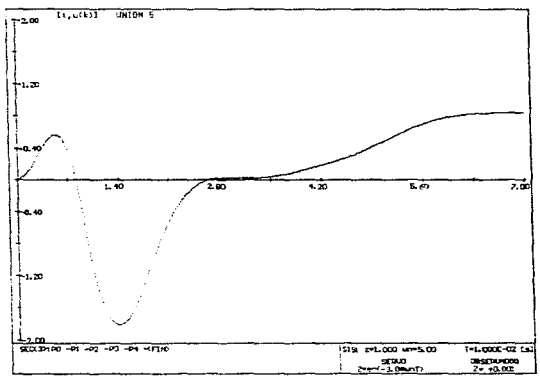


Fig. 7.24 Señal control [Volt] unión 5

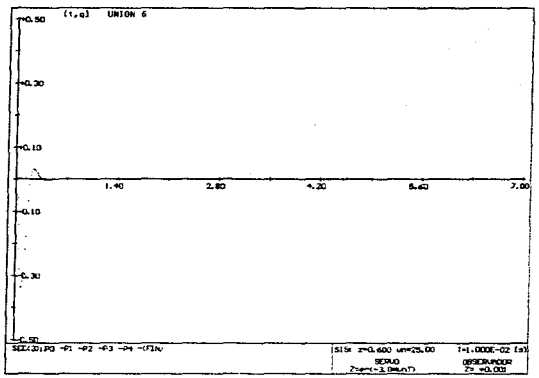


Fig. 7.25 Posición unión 6 [°]

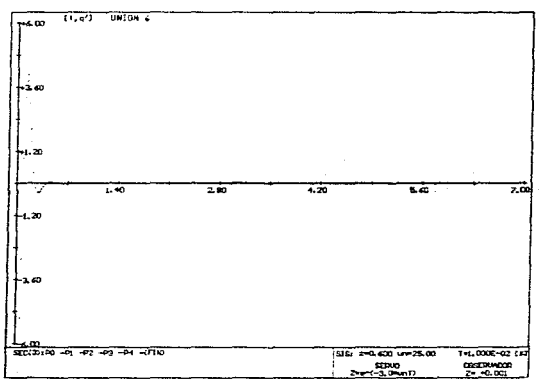


Fig. 7.26 Velocidad unión 6 [°/s]

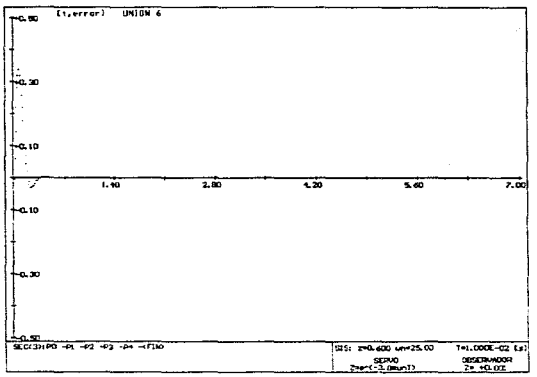


Fig. 7.27 Error posición [°] unión 6

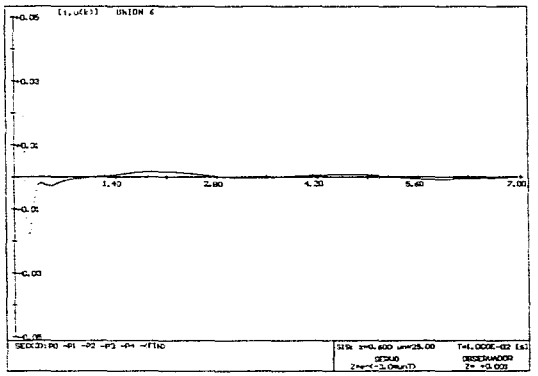


Fig. 7.28 Señal control [Volt] unión 6

7.3 Pruebas del algoritmo de control

Las pruebas se realizaron sobre el manipulador de dos GDL que se encuentra en el "Laboratorio de Control" (DEPFI) y el equipo utilizado consistió en:

-Manipulador de dos GDL (planar).

-Interfase de potencia.

-Fuente de voltaje "Power Module PAO103"

-Computadora PC486-33 con la tarjeta dSPACE (DS1102).

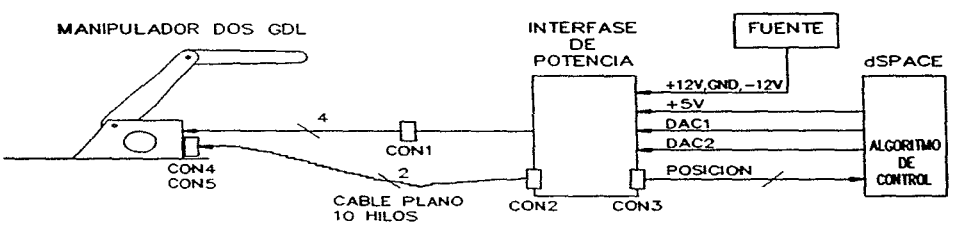


Fig. 7.29 Diagrama de interconexiones.

El manipulador utilizado tiene dos uniones de revolución y la disposición de éstas es tal que el manipulador únicamente puede realizar movimientos en un plano vertical (figura 7.29 y 7.30); y los movimientos están limitados por la base y por la interferencia entre los propios elementos. El manipulador está construido por perfiles de aluminio y cuenta con una transmisión de potencia para cada unión, con la característica especial que la segunda unión (codo) está actuada a distancia, es decir, el motor está situado a la altura de la primera unión y provoca que la orientación del segundo elemento no cambie por el movimiento de la primera unión, como se muestra en la figura 7.30, en donde se está moviendo únicamente la unión 1 (hombro).

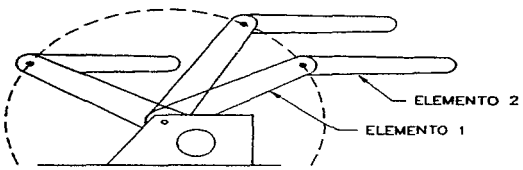


Fig. 7.30 Movimiento de la unión 1.

Por las longitudes de los elementos y por las interferencias entre ellos se tiene el espacio de trabajo mostrado en la figura 7.31

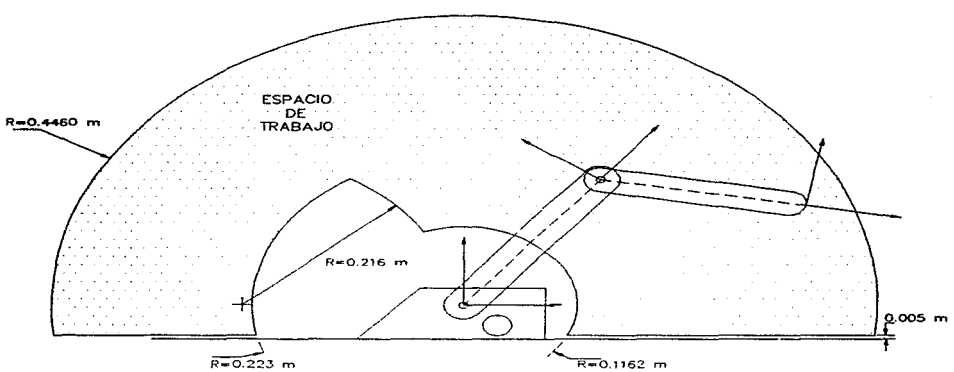


Fig. 7.31 Espacio de trabajo.

El manipulador cuenta con motores de CD de 20 volts controlados linealmente por medio de amplificadores operacionales de potencia, con la única limitante que se reduce el rango del voltaje a ± 10 volts; debido a los convertidores D/A que manejan a los amplificadores operacionales. Cada motor cuenta con un encodificador óptico y es por medio de este que se mide la posición en cada motor y al tomar en cuenta la transmisición de potencia se obtiene[53] la posición a nivel de cada unión.

El control del manipulador se realiza por medio de la tarjeta dSPACE, la cual está diseñada en base al DSP TMS320C30, y que es especialmente útil en la implementación de sistemas de control, ya que ofrece una plataforma de desarrollo muy potente que permite llegar rápidamente a pruebas de los algoritmos. La programación se realiza en "C" y se cuenta con las herramientas "Cockpit3" y "Trace3"; "Cockpit3" es una interfase gráfica para los programas desarrollados en "C" y básicamente es un enlace entre las variables del programa y diversos controles gráficos, de forma que se puede alterar u observar en tiempo real el contenido de las variables y por medio de esto controlar el flujo del programa y/o cambiar los parámetros de los algoritmos que se estén ejecutando. "Trace3" permite graficar el comportamiento en el tiempo de las variables con lo que se puede obtener resultados experimentales de las pruebas; para la documentación de los experimentos o para continuar el análisis de éstos, existe la posibilidad de guardar los resultados experimentales y llevarlos a MATLAB[54].

---

[53] En el apéndice H se cuenta con más información sobre la adquisición de la posición de cada unión.

[54] MATLAB es un programa en el que se manejan cálculos numéricos para el análisis y diseño en varias áreas, como control,procesamiento de señales, etc. Aparte del procesamiento numérico cuenta con una gran capacidad de representación gráfica.

Al ser un manipulador de revolución con ejes paralelos en sus dos primeras uniones, su comportamiento cinemático es una simplificación del análisis realizado en el capítulo 3 y el único problema es el acoplamiento entre las dos uniones, que no permite aplicar directamente la conveción D_H como se hizo para el análisis del manipulador de seis GDL pero, al ser sólo dos uniones el camino más simple es el realizar todos los cálculos directamente. De la misma forma para el análisis dinámico es más fácil obtener directamente las ecuaciones de Euler_Lagrange que aplicar el método de Newton_Euler; y la descripción de los análisis de este manipulador se encuentran en el apéndice H.

La prueba consistió en mover a las uniones del manipulador de acuerdo a una trayectoria senoidal con las siguientes características:

$$q_d = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix} sen(\omega_1) + \begin{bmatrix} 90° \\ 90° \end{bmatrix} \qquad (7.1)$$

Donde: $A_1 = 16°$

$A_2 = 16°$

$w_1 = 0.42$ Hz

Y en las figuras 7.32 y 7.33 se muestra la posición y la velocidad deseadas para cada unión.

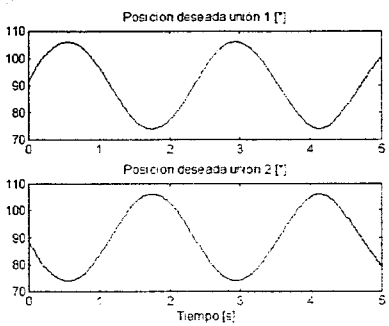


Fig. 7.32 Posición deseada.

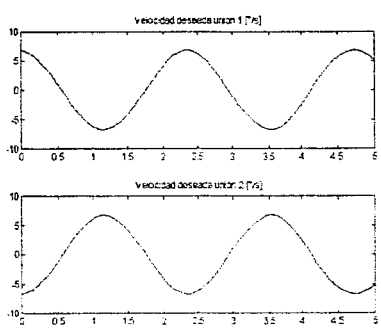


Fig. 7.33 Velocidad deseada.

Los resultados experimentales,utilizando el programa "Trace3" se muestran a continuación en las figuras 7.34, 7.35 y 7.36

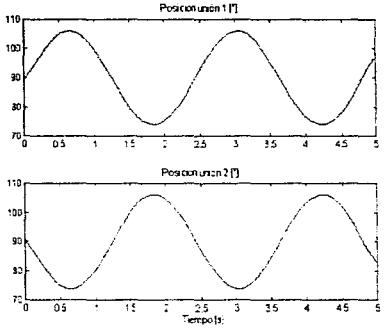


Fig. 7.34 Posición medida.

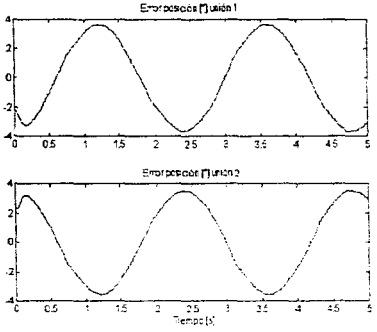


Fig. 7.35 Error de posición.

Por último se comparó el desempeño del algoritmo de control diseñado en el capítulo 4 contra otros algoritmos ya implementados para el manipulador de dos GDL, y los resultados obtenidos

se muestran[55] en las figuras 7.38 y 7.39

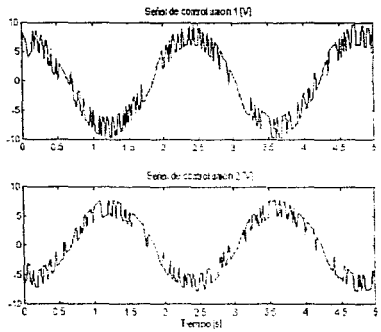


Fig. 7.36 Señal de control.

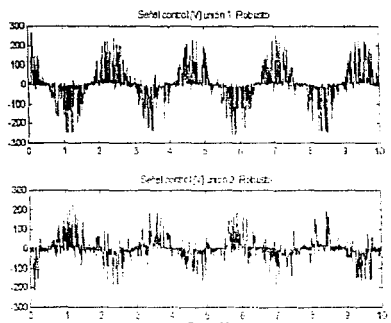
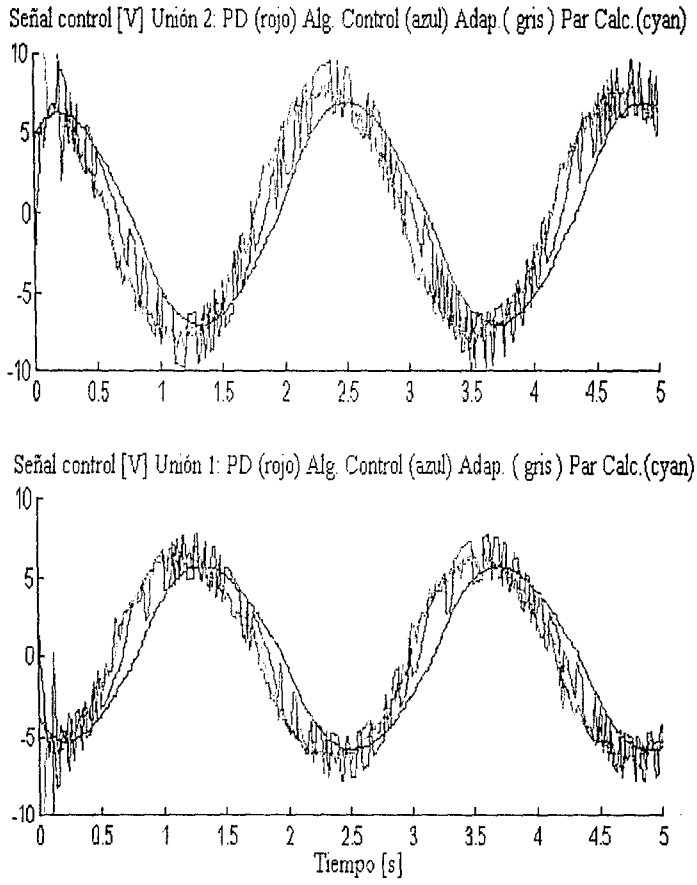


Fig. 7.37 "Control robusto".

[55] En la figura 7.38 falta la señal de control generada por el "control robusto" y esto se debe a su forma de onda, la cual no coincide en amplitud con las otras y es muy ruidosa, por lo que no dejaría ver las características de los otros controles. Por esto se muestra ésta señal en una figura separada, en la figura 7.37

Fig 7.38 Comparación entre algoritmos de control



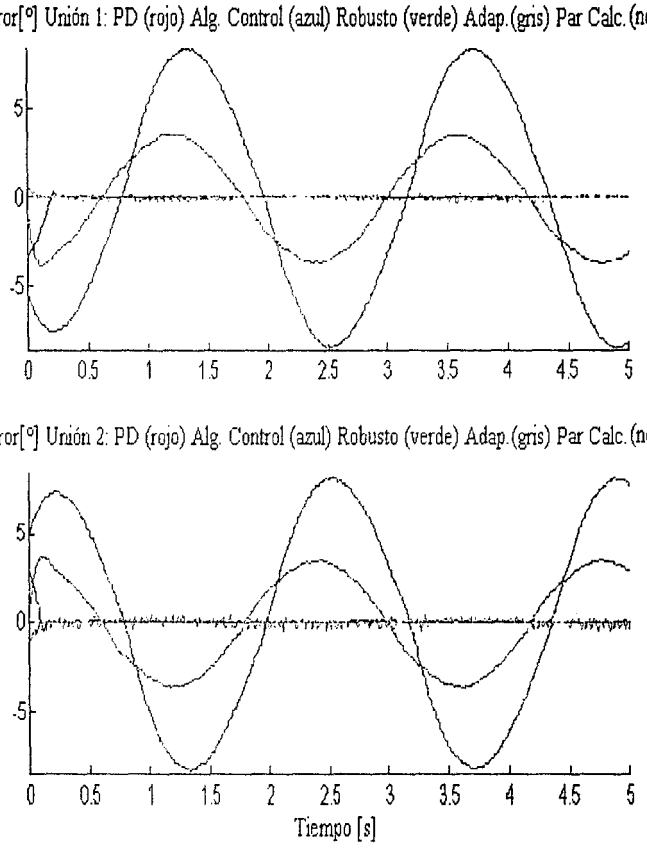


Fig 7.39 Comparación entre algortimos de control

Los algoritmos de control, cuya respuesta se muestra en las figuras 7.37, 7.38 y 7.39 tienen las siguientes características:

Algoritmo de control (diseñado en el capítulo 4):

$$\begin{aligned} &\textit{UNIÓN } 1: K_1=[3.756] \quad K_2=[325.4 \quad 2.758] \quad K_e=[959.4] \\ &\textit{UNIÓN } 2: K_1=[2.956] \quad K_2=[247.6 \quad 1.848] \quad K_e=[959.4] \end{aligned} \tag{7.2}$$

Control adaptable:

La señal de control se obtiene de acuerdo a las ecuaciones 7.3 y 7.4 en donde $\hat{\theta}$ es el vector de parámetros y "Y" representa el regresor, de forma que $Y\cdot\theta$ forman la dinámica del manipulador[56].

$$\begin{aligned} u = -Ks + Y\hat{\theta} \quad &; \frac{d}{dt}\hat{\theta} = \Gamma \cdot Y^T s \\ &; Y(q,\dot{q},\dot{q}_r,\ddot{q}_r) \\ &; \Gamma = diag > 0 \\ &; s = \dot{q} - \dot{q}_r \\ &; \dot{q}_r = \dot{q}_d - \lambda\tilde{q} \quad \lambda > 0 \\ &; \tilde{q} = q - q_d \end{aligned} \tag{7.3}$$

$$K = diag(45 \quad 65) \quad \Gamma = diag(20 \quad 20 \quad 20 \quad 20 \quad 20) \quad \lambda = 15 \tag{7.4}$$

[56] Véase el apéndice H para la descripción del regresor y del vector de parámetros utilizado.

Control PD más compensación gravitatoria:

La señal de control se genera de acuerdo a 7.5 y 7.6 en donde Kp y Kd son las ganancias proporcional y derivativas del control y "g" es la compensación gravitatoria y se obtiene de la ecuación h.22. en el apéndice H.

$$u = -K_p\tilde{q} - K_D\dot{q} + g \quad ; \quad \tilde{q}=q-q_d \tag{7.5}$$

$$K_P=diag(0.2 \quad 0.2) \qquad K_D=diag(0.002 \quad 0.0025) \tag{7.6}$$

Control robusto:

$$u = -Ks + Y(\hat{\theta} + v) \quad ; \quad v = \frac{-\rho Y^T s}{\|Y^T s\| + \varepsilon}$$
$$; \ \varepsilon>0, \ K>0, \ \lambda>0$$
$$; \ \|\hat{\theta} - \theta\| \le \rho \tag{7.7}$$
$$; \ s=\frac{d}{dt}\tilde{q} + \lambda\tilde{q}$$

$$K=diag(8 \quad 8) \qquad \varepsilon=0.001 \qquad \rho=0.045 \tag{7.8}$$

Par calculado:

$$u = -Ks + Y\theta \tag{7.9}$$

$$K=diag(40 \quad 40) \qquad \lambda=20 \tag{7.10}$$

CAPITULO 8

CONCLUSIONES

Después de realizar las tareas que implicó el objetivo de la tesis, es decir el desarrollo, implementación y pruebas del control para un manipulador de seis grados de libertad, se llegó a varias conclusiones las cuales se pueden agrupar en: programación, control, arquitectura y trabajo a futuro.

Programación:

En la prueba a la programación se validaron todos algoritmos de manejo del manipulador, como es el esquema de modificación de parámetros, la creación de secuencias de movimiento, la generación de trayectorias, cálculo de la dinámica del manipulador, etc.; de forma que se pueda manejar a un manipulador con estos algoritmos. Además la interfase del programa en "C" , en base a menús, demuestra que la interfase al usuario de la arquitectura tiene la capacidad para realizar las diferentes funciones que involucra el control de un manipulador.

El bloque de generación de trayectoria funcionó de acuerdo a las expectativas y resultó ser una buena opción para crear las trayectorias de las uniones en un control punto a punto (que consiste en pasar por una serie de puntos en el momento correcto).

En la simulación, el control del manipulador funciona como un control punto a punto debido al generador de trayectoria, que se basa en puntos nodo para crear el comportamiento en el tiempo de la posición en las uniones; para generar una trayectoria continua, bastaría con especificar suficientes puntos sobre la trayectoria deseada.

En los resultados obtenidos en la simulación, véase el capítulo 7, se tienen errores de seguimiento aceptables, la mayoría tiene un error pico menor a 5°. De la tabla 7.1 se ve que para las tres primeras uniones se utilizó un coeficiente de amortiguamiento muy bajo y esto produjo un error pequeño para la unión 1 y 3; la unión 2 (hombro) para los mismos parámetros en su control tiene un error mayor y esto se explica porque esta unión tiene que mover a la mayoría de los elementos y a la carga, por lo que las perturbaciones son mayores y aparte su trayectoria especificada es tal que se tiene una velocidad mayor que en las otras uniones, por lo que podría reducirse el error al especificarse un movimiento más lento para esta unión.

En cambio para las uniones 4 y 5 se tiene un error mucho mayor ya que se tiene un control más lento, con un coeficiente de amortiguamiento unitario y si se observa la señal de control producida por estos controles, es pequeña por lo que un aumento en la frecuencia natural o un coeficiente de amortiguamiento menor fácilmente pueden reducir el error.

La trayectoria de la unión 6 es fija en 0°, es por eso que tiene un error muy pequeño, ya que el control sólo tiene que ocuparse de las perturbaciones no canceladas, causadas por los otros elementos sobre la carga, un cubo de 5 cm de lado y 50 gr de masa.

Los coeficientes de amortiguamiento utilizados implican sobrepasos muy grandes, lo que no se puede permitir en el control de un manipulador pero, en este caso en particular se pudieron usar por el bloque de generación de trayectoria que crea una posición con respecto al tiempo que varia suavemente, y por tanto no hay problemas por sobrepasos debidos a cambios bruscos de posición.

La posibilidad de bajar el coeficiente de amortiguamiento permite especificar una respuesta más rápida con frecuencias naturales no tan elevadas, y esto es deseable ya que hay que recordar que se tiene límites en la frecuencia natural de los controladores, siendo el límite más estricto el impuesto[56] por la frecuencia resonante estructural del manipulador.

Control:

Si bien hubiera sido ideal probar sobre un manipulador real de seis GDL, las pruebas experimentales sobre el de dos GDL resultan adecuadas, ya que se cuenta con todo lo necesario para obtener los datos experimentales y comprobar el algoritmo de control: lo que sería muy difícil de lograr haciendo pruebas con la arquitectura desarrollada y un manipulador, debido a que se tendría que haber hecho un manipulador de seis GDL que pudiera ser manejado por la arquitectura, lo que involucra un diseño mecánico muy complejo y requerimientos en su construcción por fuera del alcance de la tesis.

Para el experimento con el manipulador de dos GDL, se generó una trayectoria senoidal especificada por la ecuación 7.1 y se obtuvo un error de seguimiento, fig 7.35, el cual es menor a 4° para una señal de control, fig 7.36, que no alcanza a saturar a los actuadores (es decir menor a ± 10 volts). Este comportamiento está dentro de los resultados esperados, ya que el error de seguimiento, -expresado como la diferencia entre el valor real menos el valor esperado en la posición de cada unión-, no presenta ruido en su respuesta, lo que indica un desempeño "suave" del manipulador al seguir la trayectoria deseada.

---

[56] K.S. Fu et al, op. cit., pág 220. Se tiene: $w_n <= 0.5 w_r$, donde w_r es la frecuencia resonante estructural del manipulador.

Al manejar el manipulador de dos GDL manualmente se observó que para desplazamientos grandes, alrededor de 40°, se presentaban los efectos de un comportamiento subamortiguado, ya que en el sistema de control de este manipulador no existe un generador de trayectoria como el desarrollado en la tesis; por lo que para operar al manipulador se necesita señales de referencia que causen errores pequeños y por tanto sobrepasos aceptables.

Al que incorporar un integrador a un sistema de control se produce un error cero en estado estable pero hay ciertos detalles en su funcionamiento que hay que considerar: el integrador afecta el comportamiento del sistema, ante errores grandes se tiene sobrepasos grandes, aun para sistemas sobreamortiguados y hace que la señal de control actúe por más tiempo.

Una de las ventajas de haber realizado las pruebas sobre el manipulador de dos GDL, fue la posibilidad de comparar el desempeño del algoritmo de control desarrollado, con otros ya implementados para este manipulador. Se esperaba un buen desempeño de los controladores robusto y adaptable, ya que tienen un buen comportamiento ante incertidumbres paramétricas de la planta, y de igual forma se esperaba que el comportamiento del algoritmo desarrollado tuviese un desempeño inferior (comparado con el control robusto o el adaptable), debido a que es sensible a estas incertidumbres. Por otro lado, al manejar una dinámica completa se esperaba que el algoritmo desarrollado se comportara mejor que un control PD con compensación gravitatoria y que fuera igual[57] de preciso que un control de par calculado, ya que se estaba usando la misma dinámica en ambos controladores.

Como resultados de la prueba, figuras 5.37, 5.38 y 5.39, se observa que el control adaptable

[57] C.H. An, et al, "Experimental evaluation of feedforward and computed torque control", en IEEE Trans. on Robotics & Automation, Vol 5 Núm. 3, junio 1989, pp. 368-373.

tiene un error aceptable, con una señal de control que no satura los actuadores mientras que el robusto logra un error menor, pero, con una señal de control muy ruidosa y saturando mucho a los actuadores. El algoritmo de par calculado tiene un error comparable con el control robusto lo que habla de un buen modelo dinámico. El algoritmo desarrollado, control por asignación de polos con observador de orden reducido, tiene un comportamiento mejor que el PD pero tiene un error muy grande en comparación con el control de par calculado y esto se debe a varias causas: en el par calculado se estima la velocidad[58] de forma diferente y aunque sea un esquema para control de posición se tiene buenos comportamientos mientras la velocidad no sea muy rápida. El observador de orden reducido utilizado en el algoritmo de control, depende del modelo de la unión, por lo que es sensible a la incertidumbre de los parámetros de los motores de CD. Otra causa es que el sistema es más lento, por el uso del integrador.

Pese a que los resultados experimentales sólo cumplieron parcialmente sus expectativas, es decir el algoritmo de control tuvo un mejor desempeño que un control PD, pero no llegó a tener un comportamiento similar al control de par calculado. El control por asignación de polos, con un observador de orden mínimo y con compensación anticipativa, es bastante atractivo ya que: es simple de diseñar y de implementar; el que muchos de los cálculos se puedan realizar con anterioridad es una ventaja en el caso de implementarse en un manipulador de seis GDL y se sabe que, con un buen modelo del manipulador, es comparable con el control de par calculado. Su limitante es que el método de diseño consiste básicamente en la localización de los polos e indirectamente modifica el comportamiento del sistema.

---

[58] H. Berghuis & Henk Nijmeijer, "Global regulation of robots with only positions measurements", en S&C Letter, Vol 21, 1993, pp 289-293.

Arquitectura:

Tras diseñar e implementar la arquitectura del controlador se llegó a varias conclusiones:
Implementar un sistema desde cero permite tener el sistema exacto que se busca, es decir un sistema compacto y muy enfocado a la aplicación que se trate, pero con las desventajas del tiempo de desarrollo y pruebas. El desarrollo con el 80C186 fue una buena elección, en cuanto a que se obtuvo un sistema pequeño en tamaño, bastante versátil y poderoso.
Se debe de diseñar pensando en el ruido, en la disipación de potencia, en cuidar las señales más sensibles a causar problemas como RESET y CLK. Hay que cuidar los niveles de Vcc y tierra cuando se divide el sistema en varias tarjetas y poner "drivers" y "receivers" para todas las señales que se van a mandar por un cable plano.
La programación en ensamblador se vuelve un trabajo muy difícil si no se cuenta con un mínimo de herramientas de diseño como un emulador del procesador, y es muy deseable un simulador de EPROMS para acelerar el proceso de diseño/pruebas. La programación de un sistema con dos procesadores es difícil, lograr el procesamiento en paralelo y la comunicación tardó bastante tiempo y es un proceso de muchos ajustes que necesita de un extenso trabajo.

La arquitectura implementada, pese a las dificultades en su implementación y pruebas, tiene un buen comportamiento, cumple con los requerimientos planteados en su diseño y es un sistema que puede ser la base para diseños posteriores de controladores de manipuladores.

Trabajo a futuro:

Debido a las limitantes de tiempo, costo y complejidad se tuvo que limitar los alcances de la tesis en muchos aspectos y por tanto se puede proponer varios desarrollos a futuro.

En una primera etapa, sin modificar el trabajo realizado en la tesis, se plantea el probar conjuntamente todas las partes, es decir probar la arquitectura corriendo toda la programación diseñada y con un manipulador más complejo.

En una segunda etapa se plantean modificaciones a ciertas partes de la tesis:

Se puede lograr un mejor modelo del manipulador y por tanto un mejor desempeño, si se toma en cuenta la dinámica de los actuadores, es decir, no despreciar la inductancia del motor de CD y trabajar con un sistema de tercer orden.

Mejorar el modelo del manipulador puede representar el tomar en cuenta las flexibilidades en uniones y elementos de éste, por lo que el sistema en variables de estado en lugar de tener dos estados tendría cuatro, aumentaría la complejidad pero sería más exacto el modelo.

Hay una infinidad de posibles ganancias para los controladores, lo que hace difícil el seleccionar un conjunto de ellas y considerar que sea la mejor opción. Una solución sería el encontrar K_1, K_2 y K_e para cada unión por un criterio de minimización.

Un factor clave en la implementación de un controlador para un manipulador son los parámetros del sistema, es decir los datos de los actuadores, de la transmisión de potencia, de los elementos mecánicos, etc y se debe de reconocer que muchos de ellos no se puede conocer exactamente y la mayoría de las veces sólo se tiene un mejor estimado de ellos, por lo que una opción a esquemas de control adaptable y robusto es trabajar en la identificación de los parámetros del manipulador.

Pensar en aumentar la complejidad del sistema implica el modificar ciertas partes de la arquitectura como es la interfase con el usuario, que en este momento es funcional pero no permite la especificación de trabajos complejos. La modificación de la interfase con el usuario implica la conexión a una computadora personal en donde se lleve cierto proceso fuera de línea, como es la comunicación con el usuario, la generación de trayectoria, la planeación de tareas, etc. La conexión con una computadora modificaría la tarjeta del procesador 1.

Para aplicaciones más complejas, en las que se requiera una trayectoria a seguir, el bloque de generación de trayectoria llegaría a su límite de funcionamiento ya que sería muy difícil establecer en base a puntos nodo toda una trayectoria, por lo que se requiere de otros algoritmos de generación que se conjunten con esquemas de planeación de tareas, de forma que de una tarea como puede ser pintar una pieza, se genere con base a la geometría de la pieza las trayectorias requeridas, tomando en cuenta la naturaleza de la tarea en cuestión.

Si se quiere tener una mayor libertad en cuanto al manipulador que se pueda controlar, se necesita modificar la conversión D/A para manejar una mayor potencia en los actuadores. También se necesitaría de incorporar todo lo necesario para trabajar con encodificadores ópticos ya sean absolutos o incrementales, ya que es común encontrar esta clase de sensores en los manipuladores comerciales.

CAPÍTULO 9

BIBLIOGRAFÍA

C.H. An et al.
"Experimental evaluation of feedforward and computed torque control",
en *IEEE Trans. on Robotics and Automation*,
núm. 3, vol. 5, junio de 1989, pp. 368-373.

H. Berghuis y H. Nijmeijer
"Robust contro of robots via linear estimated state feedback",
en *IEEE Trans. on Automatic Control*,
núm. 10, vol. 39, octubre de 1994, p. 2159.

R.C. Dorf (editor)
Concise International Encyclopedia of Robotics,
E.U., John Wiley & Sons Inc., 1a. ed., 1990.

S. Doughty
Mechanics of Machines
E.U., John Wiley & Sons, 1a. ed., 1988.

K.S. Fu, R.C. González y C.S.G. Lee
Robótica: Control, detección, visión e inteligencia
México, McGraw-Hill, trad de la 1a. ed. del inglés (1987), 1989.

J. Ish-Shalom y P. Kazanzides,
"SPARTA: Multiple signal processors for high-performance robot control",
en *IEEE Trans. on Robotic and Automation*,
núm. 5, vol. 5, octubre de 1989, pp.628-640.

C.S. Lin, P.R. Chang y J.Y.S. Luh,
"Formulation and Optimization of Cubic Polynomial Joint Trajectories for Industrial Robots",
en *IEEE Trans. on Automatic Control*,
núm. 12, vol. 28, diciembre de 1983, pp 1066-1074.

J.Y.S. Luh y C.S. Lin,
"Approximate Joint Trajectories for Control of Industrial Robots along Cartesian Paths",
en *IEEE Trans. on Systems, Man & Cybernetics*,
núm. 3, vol. 14, mayo/junio de 1984, pp. 444-450.

P. J. McKerrow
Introduction to robotics
Singapur, Addison-Wesley, 1a. ed., 1991.

Meriam y L.G. Kraige
Engineering mechanics, Vol 2 Dynamics
Prentice-Hall, 2a. ed., 1987

S. Narasimhan et al.
"CONDOR:An architecture for controlling the Utah-MIT dexterous hand",
en *IEEE Trans. on Robotic and Automation*,
núm. 5, vol. 5, octubre de 1989, pp. 616-627.

K. Ogata,
Discrete-time control systems
E.U., Prentice_Hall, 1a ed., 1987.

E.I. Rivin
Mechanical design of robots
E.U., McGraw-Hill, 1a. ed., 1988.

M.W. Spong y M. Vidyasagar
Robot Dynamics and Control
Singapur, John Wyley & Sons, 1a. ed., 1989.

T. J. Tarn et al.
"Performance comparison of four manipulator servo shemes",
en *IEEE Control Systems Magazine*,
núm. 1, vol. 13, febrero de 1993, pp. 22-29.

Hojas de datos técnicos:
80C186XL (Intel Literature 27248 1-001).
80C187 (Intel Literature 270640-003).
82C55A (Intel Literature 231256-004).
TIP110 ("Power Products Data Book", Texas Instrument, 1990).

Manual 80C186XL.
Embedded microprocessors (Intel Literature 272396).

APÉNDICES

APENDICE A

Una transformación homogénea, describe la relación entre dos marcos de referencia, uno fijo (inercial) y otro móvil; teniendo en cuenta las rotaciones y traslaciones de estos. Como el movimiento de un cuerpo rígido siempre se puede descomponer en una traslación y una rotación, el describir las posiciones de un cuerpo rígido conforme se mueve, por medio de una transformación homogénea es un método muy utilizado en el análisis de los manipuladores; en donde se tienen marcos de referencia fijos a los elemento de forma que la relación entre marcos de referencia de elementos adyacentes, está en función únicamente de la unión que está entre dichos elementos y se representa por una transformación homogénea.

En general se tiene que una transformación homogénea tiene la forma de:

$$T = \begin{bmatrix} R & \mathbf{d} \\ f & s \end{bmatrix} \qquad \text{(a.1)}$$

En donde R, establece la rotación entre los marcos de referencia, **d** el desplazamiento entre orígenes, "f" la perspectiva y "s" el factor de escala. Para el análisis de un manipulador, sólo se utiliza la parte de rotación y de desplazamiento; por lo que la parte de perspectiva es nula y el factor de escala es unitario, quedando la transformación:

$$T = \begin{bmatrix} R & \mathbf{d} \\ 0 & 1 \end{bmatrix} \qquad \text{(a.2)}$$

Las transformaciones se representan por una matriz de 4X4.

El que se pueda saber la relación entre dos marcos de referencia, implica que un punto que se encuentra fijo con relación a cierto marco de referencia (móvil), y por tanto con coordenadas constantes; se puede describir en coordenadas de otro marco de referencia (inercial), por medio

de una transformación homogénea, no importando el cambio de posición y/u orientación que sufra el marco de referencia móvil. Como las transformaciones son matrices de 4X4, se tienen que manejar coordenadas homogéneas, que tienen la forma:

$$\boldsymbol{p} = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} \qquad \textit{En coordenadas cartesianas.} \qquad \textbf{(a.3)}$$

$$\boldsymbol{p} = \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix} \qquad \textit{En coordenadas homogénas.} \qquad \textbf{(a.4)}$$

En forma general las coordenadas homogéneas utilizan el cuarto término para dar un factor de escala a la coordenada, pero en nuestro caso siempre se manejara un factor de escala unitario.

Como las transformaciones homogéneas manejan tanto rotaciones como desplazamientos, se tiene el caso de transformaciones básicas en las que sólo se efectúe un desplazamiento o sólo una rotación y por simplicidad se cambia de notación:

Rot_{k,θ_i} Es una rotación alrededor del eje k, θ_i grados.

$Tras_{k,d_i}$ Es un desplazamiento en dirección del eje k, una distancia d_i

La parte que describe las rotaciones dentro de la transformación, es una matriz de 3X3 y se llama matriz de rotación. Esta matriz es una matriz ortonormal, en la cual sus columnas son ortogonales entre si, al igual que sus renglones y forman vectores unitarios, debido a esta característica se tiene que la inversa de una matriz de rotación es su traspuesta y su determinante es +1. La matriz identidad de 3X3, implica una rotación nula: es decir una orientación idéntica entre dos marcos de referencia.

Tomando a los ejes que forman el marco de referencia, se tiene que se pueden efectuar tres rotaciones básicas y en base a estas se puede formar cualquier otra rotación:

$$Rot_{Z,\theta} = \begin{bmatrix} \cos\theta & -sen\theta & 0 \\ sen\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad Rot_{X,\theta} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -sen\theta \\ 0 & sen\theta & \cos\theta \end{bmatrix} \qquad (a.5)$$

$$Rot_{Y,\theta} = \begin{bmatrix} \cos\theta & 0 & sen\theta \\ 0 & 1 & 0 \\ -sen\theta & 0 & \cos\theta \end{bmatrix} \qquad (a.6)$$

Se tiene las siguientes propiedades en el manejo de las matrices de rotación:

$$\begin{aligned} &Rot_{K,0} = I \quad , I \text{ matriz identidad.} \\ &Rot_{K,\theta}\, Rot_{K,\phi} = Rot_{K,\theta+\phi} \\ &[Rot_{K,\theta}]^{-1} = Rot_{K,-\theta} \end{aligned} \qquad (a.7)$$

El que sean matrices de 3X3, hace que fuera de la transformación homogénea, se pueda utilizar las matrices de rotación a coordenadas no homogéneas. El premultiplicar las coordenadas de un punto, por una matriz de rotación, hace que se obtenga unas nuevas coordenadas y que representan al punto rotado de la forma descrita por la matriz de rotación; de igual forma se pudo haber explicado el resultado como la creación de un nuevo marco de referencia (móvil), rotado de la forma descrita en la matriz y el resultado son las coordenadas del punto expresado en este nuevo marco de referencia.

Una matriz de rotación arbitraria se puede expresar de forma simplificada de varias maneras[1]:

-como una matriz equivalente, rotando alrededor de un eje cierto número de grados.

-por medio de ángulos de Euler, en el que la rotación arbitraria se simplifica en tres

[1] Mark W. Spong y M. Vidyasagar, op. cit., pp 43-46

rotaciones consecutivas.

-por medio de giro/elevación/desviación, en el que la rotación se expresa como rotaciones básicas consecutivas.

Como la multiplicación de matrices no es conmutativa, en el caso de matrices de rotación el premultiplicar o posmultiplicar dos matrices afecta el orden en que se realiza la rotación.

La parte de desplazamiento de una transformación homogénea, es el vector entre el origen del marco de referencia fijo y el origen del marco de referencia móvil. La notación que se utiliza para denominar a una transformación dada, varia de libro en libro y la que se va a utilizar es: T_0^1 es la transformación del marco de ref. 1 al marco de ref. 0 y está compuesta por:

$$T_0^1 = \begin{bmatrix} R_0^1 & Q_0^1 \\ \mathbf{0} & 1 \end{bmatrix} \qquad \text{(a.8)}$$

Al igual que con las matrices de rotación, se puede premultiplicar o posmultiplicar las transformaciones homogéneas para que, en base a transformaciones básicas, crear una transformación más compleja; por ejemplo en el caso de un manipulador de seis ejes, se tienen seis marcos de referencia y por tanto se pueden crear muchas transformaciones en base a las relaciones entre marcos adyacentes, con lo que se puede conocer las coordenadas de un punto en un elemento cualquiera, con respecto al marco de referencia de la base, o con respecto al marco de referencia de algún elemento en particular. Por ejemplo la transformación homogénea del elemento "j", al elemento "i" sería de la forma:

$$T_i^j = A_{i+1}\, A_{i+2} \cdots A_{j-1}\, A_j \quad si\ i<j \qquad \text{(a.9)}$$

Donde A_i es la transformación homogénea (con una notación simplificada) del marco de referencia "i" al marco de referencia "i-1", y es función de la variable de la unión "i", es decir de q_i. Las diferentes matrices A (A_{i-1}, A_{i-2}, ·, A_{j-1}), son las transformaciones homogéneas de los elementos entre el elemento "j" e "i" y tienen la forma:

$$A_i(q_i) = \begin{bmatrix} R_{i-1}^{i} & d_{i-1}^{i} \\ 0 & 1 \end{bmatrix} \qquad \textbf{(a.10)}$$

Por lo que :

$$T_i^j = \begin{bmatrix} R_i^j & d_i^j \\ 0 & 1 \end{bmatrix} \qquad \textbf{(a.11)}$$

Donde: R_i^j se refiere a la matriz de rotación que toma en cuenta todas las diferencias en orientación entre el elemento "j" y el elemento "i".

d_i^j se refiere al vector que une el origen del marco de referencia del elemento "i", al marco de referencia del elemento "j". Como la transformación T_i^j está formada por varias trasformaciones, se tiene:

$$\begin{aligned} R_i^j &= R_i^{i+1} R_{i+1}^{i+2} \cdots\cdots R_{j-1}^{j} \\ d_i^j &= d_i^{j-1} + R_i^{j-1} d_{j-1}^j \qquad \textbf{(a.11)} \\ T_i^j &= I \qquad si\ j = i \\ T_i^j &= (T_j^i)^{-1} \qquad si\ i < j \end{aligned}$$

APENDICE B

Operaciones, para encontrar la transformación homogénea, entre el marco de referencia "seis" y el marco de referencia de la base ("cero").

$$T_0^2 = A_1 A_2 = \begin{bmatrix} C_1 & 0 & S_1 & 0 \\ S_1 & 0 & -C_1 & 0 \\ 0 & 1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C_2 & -S_2 & 0 & a_2C_2 \\ S_2 & C_2 & 0 & a_2S_2 \\ 0 & 0 & 1 & d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} C_1C_2 & -C_1S_2 & S_1 & a_2C_1C_2 + d_2S_1 \\ S_1C_2 & -S_1S_2 & -C_1 & a_2S_1C_2 - d_2C_1 \\ S_2 & C_2 & 0 & a_2S_2 + d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \textbf{(b.1)}$$

$$T_0^3 = T_0^2 A_3 = T_0^2 \begin{bmatrix} C_3 & -S_3 & 0 & a_3C_3 \\ S_3 & C_3 & 0 & a_3S_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} C_1C_{23} & -C_1S_{23} & S_1 & a_3C_1C_{23} + a_2C_1C_2 + d_2S_1 \\ S_1C_{23} & -S_1S_{23} & -C_1 & a_3S_1C_{23} + a_2S_1C_2 - d_2C_1 \\ S_{23} & C_{23} & 0 & a_3S_{23} + a_2S_2 + d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \textbf{(b.2)}$$

$$T_0^4 = T_0^3 A_4 = T_0^3 \begin{bmatrix} C_4 & 0 & -S_4 & 0 \\ S_4 & 0 & C_4 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} C_1C_{234} & -S_1 & -C_1S_{234} & a_3C_1C_{23} + a_2C_1C_2 + d_2S_1 \\ S_1C_{234} & C_1 & -S_1S_{234} & a_3S_1C_{23} + a_2S_1C_2 - d_2C_1 \\ S_{234} & 0 & C_{234} & a_3S_{23} + a_2S_2 + d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \textbf{(b.3)}$$

$$T_0^5 = T_0^4 A_5 = T_0^4 \begin{bmatrix} C_5 & 0 & S_5 & 0 \\ S_5 & 0 & -C_5 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \textbf{(b.4)}$$

$$T_0^5 = \begin{bmatrix} C_1C_{234}C_5 - S_1S_5 & -C_1S_{234} & C_1C_{234}S_5 + S_1C_5 & a_3C_1C_{23} + a_2C_1C_2 + d_2S_1 \\ S_1C_{234}C_5 + C_1S_5 & -S_1S_{234} & S_1C_{234}S_5 - C_1C_5 & a_3S_1C_{23} + a_2S_1C_2 - d_2C_1 \\ S_{234}C_5 & C_{234} & S_{234}S_5 & a_3S_{23} + a_2S_2 - d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_0^6 = T_0^5 A_6 = T_0^5 \begin{bmatrix} C_6 & -S_6 & 0 & 0 \\ S_6 & C_6 & 0 & 0 \\ 0 & 0 & 1 & d_6 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} n_x & s_x & a_x & d_x \\ n_y & s_y & a_y & d_y \\ n_z & s_z & a_z & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \textbf{(b.5)}$$

Donde:

$$n_x = C_1(C_{234}C_5C_6 - S_{234}S_6) - S_1S_5C_6$$

$$n_y = S_1(C_{234}C_5C_6 - S_{234}S_6) + C_1S_5C_6$$

$$n_z = S_{234}C_5C_6 + C_{234}S_6$$

$$s_x = -C_1(C_{234}C_5S_6 + S_{234}C_6) + S_1S_5S_6$$

$$s_y = -S_1(C_{234}C_5S_6 + S_{234}C_6) - C_1S_5S_6$$

$$s_z = C_{234}C_6 - S_{234}C_5S_6$$

$$a_x = C_1C_{234}S_5 + S_1C_5$$

$$a_y = S_1C_{234}S_5 - C_1C_5$$

$$a_z = S_{234}S_5$$

$$d_x = C_1(C_{234}S_5d_6 + a_3C_{23} + a_2C_2) + S_1(C_5d_6 + d_2)$$

$$d_y = S_1(C_{234}S_5d_6 + a_3C_{23} + a_2C_2) - C_1(C_5d_6 + d_2)$$

$$d_z = S_{234}S_5d_6 + a_3S_{23} + a_2S_2 + d_1$$

APENDICE C

Desarrollo de la matriz de inercia, en las ecuaciones de lagrange_Euler.

La matriz D, se obtiene por medio de su definición (ecuación 3.37):

$$D = \sum_{i=1}^{n} [m_i (J_{v_{c_i}})^T (J_{v_{c_i}}) + (J_{\omega_i})^T (R_0^i)(I_i)(R_0^i)^T (J_{\omega_i})] \qquad \textbf{(c.1)}$$

De esa ecuación se tiene que se requiere calcular los jacobianos específicos a cada elementos y las matrices de rotación (R_0^i). También se tiene que considerar las características físicas de los elementos como la masa y el tensor de inercia.

-Velocidad lineal del centroide (v_{c_i}).

Se tiene una relación de la forma:

$$v_{c_i} = J_{v_{ci}} \dot{q} \qquad \textbf{(c.2)}$$

Donde: v_{c_i} velocidad lineal del centroide del elemento "i", con respecto al marco de referencia cero.

$\dot{q}$ velocidades de las uniones.

$J_{v_{ci}}$ parte del jacobiano del elemento "i".

$J_{v_{ci}}$, es de la forma:

$$J_{v_{ci}} = [J_1 \; J_2 \; J_3 \; J_4 \; J_5 \; J_6] \quad ; \; J_k = Z_{k-1} \times (d_0^{ci} - d_0^{k-1}) \quad para \; k \leq i \qquad \textbf{(c.3)}$$

Donde: Z_{k-1} es el eje de la unión "k".

d_0^{ci} vector del origen del marco de referencia cero, al centroide del elemento "i".

d_0^{k-1} es el vector del origen del marco de referencia cero, al marco de referencia "k-1" (unión "k"). Todos los vectores expresados en el marco de referencia cero.

De las ecuaciones 3.28 y 3.29 se obtiene Z_{k-1} y d_0^{k-1}, para k=1,2,..6

Si se conoce la posición del centroide del elemento "i", según el marco de referencia del mismo elemento, por medio de una transformación homogénea se obtiene:

$$d_0^{ci} = T_0^i(OC_i) \qquad \textbf{(c.4)}$$

Donde OC_i es la posición del centroide del elemento "i", según el marco de referencia "i". De acuerdo a las características del manipulador a controlar, las coordenadas de los centroides expresados en los marcos de referencia utilizados en el análisis cinemáticos tienen las forma:

(c.5)

$$OC_1 = \begin{bmatrix} 0 \\ OC_{1y} \\ OC_{1z} \end{bmatrix} \quad OC_2 = \begin{bmatrix} OC_{2x} \\ 0 \\ 0 \end{bmatrix} \quad OC_3 = \begin{bmatrix} OC_{3x} \\ 0 \\ 0 \end{bmatrix} \quad OC_4 = \begin{bmatrix} 0 \\ 0 \\ OC_{4z} \end{bmatrix} \quad OC_5 = \begin{bmatrix} 0 \\ 0 \\ OC_{5z} \end{bmatrix} \quad OC_6 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

T_0^i es la transformación del marco de referencia "i", al marco cero y se obtiene del desarrollo del apéndice B.

Tras la realización de las operaciones se obtiene:

$$d_0^{c1} = \begin{bmatrix} S_1(OC_{1z}) \\ -C_1(OC_{1z}) \\ OC_{1y} + d_1 \end{bmatrix} \quad \textbf{(c.6)}$$

$$d_0^{c2} = \begin{bmatrix} C_1C_2(OC_{2x}) + a_2C_1C_2 + d_2S_1 \\ S_1C_2(OC_{2x}) + a_2S_1C_2 - d_2C_1 \\ S_2(OC_{2x}) + a_2S_2 + d_1 \end{bmatrix} \quad \textbf{(c.7)}$$

$$d_0^{c3} = \begin{bmatrix} C_1C_{23}(OC_{3x}) + a_3C_1C_{23} + a_2C_1C_2 + d_2S_1 \\ S_1C_{23}(OC_{3x}) + a_3S_1C_{23} + a_2S_1C_2 - d_2C_1 \\ S_{23}(OC_{3x}) + a_3S_{23} + a_2S_2 + d_1 \end{bmatrix} \quad \textbf{(c.8)}$$

$$d_0^{c4} = \begin{bmatrix} -C_1S_{234}(OC_{4z}) + a_3C_1C_{23} + a_2C_1C_2 + d_2S_1 \\ -S_1S_{234}(OC_{4z}) + a_3S_1C_{23} + a_2S_1C_2 - d_2C_1 \\ C_{234}(OC_{4z}) + a_3S_{23} + a_2S_2 + d_1 \end{bmatrix} \quad \textbf{(c.9)}$$

$$d_0^{c5} = \begin{bmatrix} (C_1C_{234}S_5 + S_1C_5)(OC_{5z}) + a_3C_1C_{23} + a_2C_1C_2 + d_2S_1 \\ (S_1C_{234}S_5 - C_1C_5)(OC_{5z}) + a_3S_1C_{23} + a_2S_1C_2 - d_2C_1 \\ S_{234}S_5(OC_{5z}) + a_3S_{23} + a_2S_2 + d_1 \end{bmatrix} \quad \textbf{(c.10)}$$

$$d_0^{c6} = \begin{bmatrix} C_1(C_{234}S_5d_6 + a_3C_{23} + a_2C_2) + S_1(C_5d_6 + d_2) \\ S_1(C_{234}S_5d_6 + a_3C_{23} + a_2C_2) - C_1(C_5d_6 + d_2) \\ S_{234}S_5d_6 + a_3S_{23} + a_2S_2 + d_1 \end{bmatrix} \quad \textbf{(c.11)}$$

De forma que ya se tiene todo para resolver la ecuación (c.3) y tras operaciones se obtiene que[1]:

$$J_{v_{c_1}} = \begin{bmatrix} C_1(OC_{1z}) & 0 & 0 & 0 & 0 & 0 \\ S_1(OC_{1z}) & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad \textbf{(c.12)}$$

$$J_{v_{c_2}} = \begin{bmatrix} -S_1C_2(OC_{2x}) + a_2S_1C_2 - d_2C_1 & -C_1[S_2(OC_{2x}) + a_2S_2] & 0 & 0 & 0 & 0 \\ C_1C_2(OC_{2x}) + a_2C_1C_2 + d_2S_1 & -S_1[S_2(OC_{2x}) + a_2S_2] & 0 & 0 & 0 & 0 \\ 0 & C_2(OC_{2x}) + a_2S_2 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad \textbf{(c.13)}$$

$$J_{v_{c_3}} = \begin{bmatrix} a & b & c & 0 & 0 & 0 \\ d & e & f & 0 & 0 & 0 \\ 0 & g & h & 0 & 0 & 0 \end{bmatrix}$$

Donde:

$$\begin{aligned} a &= -S_1C_{23}(OC_{3x}) - a_3S_1C_{23} - a_2S_1C_2 + d_2C_1 \\ b &= -C_1[S_{23}(OC_{3x}) + a_3S_{23} + a_2S_2] \\ c &= -C_1[S_{23}(OC_{3x}) + a_3S_{23}] \\ d &= C_1C_{23}(OC_{3x}) + a_3C_1C_{23} + a_2C_1C_2 + d_2S_1 \\ e &= -S_1[S_{23}(OC_{3x}) + a_3S_{23} + a_2S_2] \\ f &= -S_1[S_{23}(OC_{3x}) + a_3S_{23}] \\ g &= C_{23}(OC_{3x}) + a_3C_{23} + a_2C_2 \\ h &= C_{23}(OC_{3x}) + a_3C_{23} \end{aligned} \qquad \textbf{(c.14)}$$

---

[1] En una configuración sin offset en la segunda unión ($d_2=0$), se tiene que $OC_{1z}=0$ y por tanto $J_{v_{c_1}}=0$ (el centroide del primer elemento no tiene velocidad lineal, por estar sobre el eje de rotación de la primera unión).

$$J_{v_{c_4}} = \begin{bmatrix} a & b & c & d & 0 & 0 \\ e & f & g & h & 0 & 0 \\ 0 & i & j & k & 0 & 0 \end{bmatrix}$$

$$a = S_1 S_{234}(OC_{4z}) + a_3 S_1 C_{23} + a_2 S_1 C_2 - d_2 C_1$$

$$b = -C_1[C_{234}(OC_{4z}) + a_3 S_{23} + a_2 S_2]$$

$$c = -C_1[C_{234}(OC_{4z}) + a_3 S_{23}]$$

$$d = -C_1 C_{234}(0C_{4z})$$

$$e = -C_1 S_{234}(OC_{4z}) + a_3 C_1 C_{23} + a_2 C_1 C_2 + d_2 S_1$$

$$f = -S_1[C_{234}(OC_{4z}) + a_3 S_{23} + a_2 S_2]$$

$$g = -S_1[C_{234}(OC_{4z}) + a_3 S_{23}]$$

$$h = -S_1 C_{234}(OC_{4z})$$

$$i = -S_{234}(OC_{4z}) + a_3 C_{23} + a_2 C_2$$

$$j = -S_{234}(OC_{4z}) + a_3 C_{23}$$

$$k = -S_{234}(OC_{4z})$$

(c.15)

$$J_{v_{c_5}} = \begin{bmatrix} a & b & c & d & e & 0 \\ f & g & h & i & j & 0 \\ 0 & k & l & m & n & 0 \end{bmatrix}$$

$$a = -(S_1 C_{234} S_5 + C_1 C_5)(OC_{5z}) - a_3 S_1 C_{23} - a_2 S_1 C_2 + d_2 C_1$$

$$b = -C_1[S_{234} S_5(OC_{5z}) + a_3 S_{23} + a_2 S_2]$$

$$c = -C_1[S_{234} S_5(OC_{5z}) + a_3 S_{23}]$$

$$d = -C_1 S_{234} S_5(0C_{5z})$$

$$e = (C_1 C_{234} C_5 + S_1 S_5)(OC_{5z})$$

(c.16)

$$
\begin{aligned}
f &= (C_1C_{234}S_5 + S_1C_5)(OC_{5z}) + a_3C_1C_{23} + a_2C_1C_2 + d_2S_1 \\
g &= -S_1[S_{234}S_5(OC_{5z}) + a_3S_{23} + a_2S_2] \\
h &= -S_1[S_{234}S_5(OC_{5z}) + a_3S_{23}] \\
i &= -S_1S_{234}S_5(OC_{5z}) \\
j &= (C_1S_5 + S_1C_{234}C_5)(OC_{5z})
\end{aligned}
\qquad \textbf{(c.17)}
$$

$$
\begin{aligned}
k &= C_{234}S_5(OC_{5z}) + a_3C_{23} + a_2C_2 \\
l &= C_{234}S_5(OC_{5z}) + a_3C_{23} \\
m &= C_{234}S_5(OC_{5z}) \\
n &= C_5S_{234}(OC_{5z})
\end{aligned}
\qquad \textbf{(c.18)}
$$

$J_{v_{c6}}$ es igual a los tres primeros renglones del jacobiano obtenido en 3.1.3 (ecuación 3.30).

-Velocidad angular del elemento "i" (ω_i):

$$\omega_i = J_{\omega_i}\dot{q} \qquad \textbf{(c.19)}$$

Donde: ω_i velocidad angular del elemento "i" referida al marco cero.

J_{ω_i} parte del jacobiano del elemento "i".

$\dot{q}$ velocidad de las uniones.

$$J_{\omega_i} = [Z_0 \; Z_1 \; Z_2 \; Z_{i-1} \; 0 \; 0] \quad ; para \; i = 1,2,\cdots,6 \qquad \textbf{(c.20)}$$

Donde Z_{i-1} es el eje de la unión "i", de acuerdo al marco de referencia cero y ya es conocido

de la sección 3.1.3 (Obtención del jacobiano), por lo que se tiene:

$$J_{\omega_1} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad \text{(c.21)}$$

$$J_{\omega_2} = \begin{bmatrix} 0 & S_1 & 0 & 0 & 0 & 0 \\ 0 & -C_1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad \text{(c.22)}$$

$$J_{\omega_3} = \begin{bmatrix} 0 & S_1 & S_1 & 0 & 0 & 0 \\ 0 & -C_1 & -C_1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad \text{(c.23)}$$

$$J_{\omega_4} = \begin{bmatrix} 0 & S_1 & S_1 & S_1 & 0 & 0 \\ 0 & -C_1 & -C_1 & -C_1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad \text{(c.24)}$$

$$J_{\omega_5} = \begin{bmatrix} 0 & S_1 & S_1 & S_1 & -C_1S_{234} & 0 \\ 0 & -C_1 & -C_1 & -C_1 & -S_1S_{234} & 0 \\ 1 & 0 & 0 & 0 & C_{234} & 0 \end{bmatrix} \qquad \text{(c.25)}$$

$$J_{\omega_5} = \begin{bmatrix} 0 & S_1 & S_1 & S_1 & -C_1S_{234} & C_1C_{234}S_5 + S_1C_5 \\ 0 & -C_1 & -C_1 & -C_1 & -S_1S_{234} & S_1C_{234}S_5 - C_1C_5 \\ 1 & 0 & 0 & 0 & C_{234} & S_{234}S_5 \end{bmatrix} \quad \textbf{(c.26)}$$

En cuanto a las matrices de rotación R_0^i se obtiene de las transformaciones T_0^i del apéndice B.

m_i e I_i, se dan en el apéndice D, características del manipulador a controlar.

Faltando únicamente realizar las operaciones de la definición (c.1), para poder obtener la matriz de inercia "D". Prácticamente es muy difícil el desarrollar de forma analítica la matriz de inercia, ya que son muchos los cálculos involucrados, por lo que se dejará a la matriz en la forma de su definición.

APENDICE D

Parámetros utilizados en la simulación.

El objetivo de la tesis es controlar un manipulador articulado tipo codo de seis GDL, es decir un manipulador con todas las uniones de revolución y cuya configuración cinemática de las primeras tres uniones es tal que el eje de las uniones dos y tres son paralelos y perpendiculares al eje de la unión número uno. En lo relativo al offset de la segunda unión se va a considerar la existencia de este, ya que al ser el caso más general, se abarca las dos posibilidades; y en cuanto a las uniones de orientación, se está tomando una muñeca esférica de tres GDL tipo "PPG". En cada unión se tiene un motor de corriente directa de imán permanente como actuador, y este se conecta al elemento por medio de un tren de engranes.

La obtención de todos los parámetros concernientes al análisis y el control es una parte muy importante; y es un procedimiento difícil ya que muchas veces no se cuenta con toda la información[1] tanto a nivel geométrico, como de los actuadores y de la transmisión de potencia. Los parámetros utilizados en la simulación son una mezcla entre datos característicos de motores de corriente directa, y las características que tendría un manipulador con actuadores de baja potencia, es decir, un manipulador que pudiera ser manejado por la arquitectura descrita en el capítulo 5.

---

[1] Se han desarrollado métodos automáticos para la obtención de los parámetros necesarios para el control, ya que aún si se conocieran los parámetros por datos de los fabricantes, éstos serían teóricos y se apartarían de los reales, por los factores que afectan a la exactitud. Estos métodos se denominan métodos de calibración y por lo general requieren de equipo extra para medir las posiciones, velocidades y aceleraciones del manipulador.

Los parámetros del manipulador, relevantes para el control, se pueden dividir en:

-Parámetros de los elementos:

-Cinemáticos:

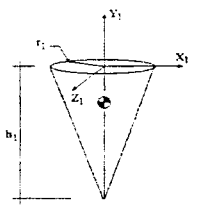
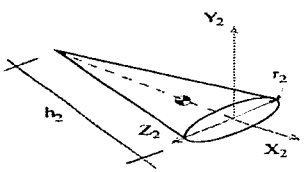
Estos parámetros básicamente son los parámetros estructurales del manipulador y se listan en la tabla d.1

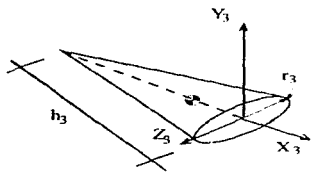
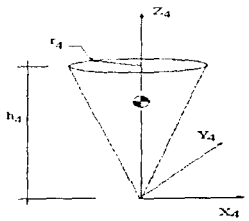
Elemento	a_i [m]	d_i [m]	α_i	θ_i
1	0	0.300	90°	$-40° < \theta_1 < 220°$
2	0.130	0.080	0°	$-90° < \theta_2 < 270°$
3	0.150	0	0°	$-130° < \theta_3 < 130°$
4	0	0	-90°	$-90° < \theta_4 < 90°$
5	0	0	90°	$-90° < \theta_5 < 90°$
6	0	0.050	0°	$-90° < \theta_6 < 90°$

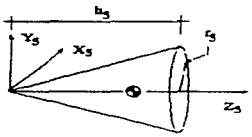
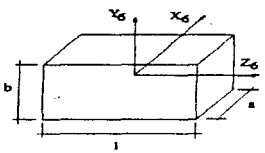
Tabla d.1 Parámetros estructurales.

-Dinámicos: El modelo que se utilizó para cada elemento, así como los parámetros resultantes se dan en la tabla d.2

Tabla d.2

Elemento	1	2
Modelo		
Centroide	$OC_1 = \begin{bmatrix} 0 \\ -\frac{1}{3}h_1 \\ 0 \end{bmatrix}$	$OC_2 = \begin{bmatrix} -\frac{1}{3}h_2 \\ 0 \\ 0 \end{bmatrix}$
Tensor de inercia.	$I_{YX} = I_{XZ} = I_{YZ} = 0$ $I_{YY} = \frac{1}{2}m_1(r_1)^2$ $I_{XX} = I_{ZZ} =$ $= \frac{1}{4}m_1(r_1)^2 + \frac{1}{18}m_1(h_1)^2$	$I_{XY} = I_{YZ} = I_{ZX} = 0$ $I_{XX} = \frac{1}{2}m_2(r_2)^2$ $I_{YY} = I_{ZZ} =$ $= \frac{1}{4}m_2(r_2)^2 + \frac{1}{18}m_2(h_2)^2$
Masa	$m_1 = \frac{\rho \pi r_1^2}{20}(\sqrt{r_1^2 + h_1^2})$	$m_2 = \frac{\rho \pi r_2^2}{20}\sqrt{r_2^2 + h_2^2}$

Elemento	3	4
Modelo		
Centroide	$OC_3 = \begin{bmatrix} -\frac{1}{3}h_3 \\ 0 \\ 0 \end{bmatrix}$	$OC_4 = \begin{bmatrix} 0 \\ 0 \\ \frac{2}{3}h_4 \end{bmatrix}$
Tensor I.	$I_{XY} = I_{YZ} = I_{ZX} = 0$ $I_{XX} = \frac{1}{2} m_3 (r_3)^2$ $I_{YY} = I_{ZZ} =$ $= \frac{1}{4} m_3 (r_3)^2 + \frac{1}{18} m_3 (h_3)^2$	$I_{XY} = I_{YZ} = I_{ZX} = 0$ $I_{XX} = I_{YY} =$ $= \frac{1}{4} m_4 (r_4)^2 + \frac{1}{18} m_4 (h_4)^2$ $I_{ZZ} = \frac{1}{2} m_4 (r_4)^2$
Masa	$m_3 = \frac{\rho \pi r_3^2}{20} \sqrt{r_3^2 + h_3^2}$	$m_4 = \frac{\rho \pi (r_4)^2}{20} \sqrt{r_4^2 + h_4^2}$

Elemento	5	6
Modelo		
Centroide	$OC_5 = \begin{bmatrix} 0 \\ 0 \\ \frac{2}{3} h_5 \end{bmatrix}$	$OC_6 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$
Tensor I.	$I_{XY} = I_{YZ} = I_{ZX} = 0$ $I_{XX} = I_{YY} =$ $= \frac{1}{4} m_5 (r_5)^2 + \frac{1}{18} m_5 (h_5)^2$ $I_{ZZ} = \frac{1}{2} m_5 (r_5)^2$	$I_{XX} = \frac{1}{12} m_6 (y^2 + l^2)$ $I_{YY} = \frac{1}{12} m_6 (a^2 + l^2)$ $I_{ZZ} = \frac{1}{12} m_6 (a^2 + b^2)$
Masa	$m_5 = \frac{\rho \pi (r_5)^2}{20} \sqrt{r_5^2 + h_5^2}$	$m_6 = (\rho_c \cdot a \cdot l \cdot b)$

Se tomó el modelo dinámico de los elementos, de la manera mostrada en la tabla d.2 para simplificar el problema de obtener los parámetros necesarios para el control y por que de quererse utilizar un modelo más real, se tendrían que involucrar la geometría de los elementos, la de los actuadores y la de la transmisión de potencia, para poder determinar de forma exacta el tensor de inercia para cada elemento y la posición su centroide.

El modelar los elementos como conos sólidos o huecos, ayuda a tomar en cuenta a los actuadores y transmisión de potencia dentro de los elementos y que son los que mueven al elemento contiguo. El modelo del elemento seis tiene que tomar en cuenta de alguna forma a la carga que se encuentra en la pinza, por lo que se utiliza un cubo con centroide en el centro de la pinza; de forma que se está despreciando la geometría del último elemento, en comparación con la geometría y masa de la carga.

-Parámetros de los actuadores/transmisión de potencia:

Todos los actuadores son motores de corriente directa, tipo imán permanente; la transmisión de potencia se lleva a cabo por medio de un tren de engranes. Su comportamiento en conjunto se representa por:

$$J_{ef}\,\ddot{\Theta}_e(t) + (B_{ef} + \frac{K_b K_a}{R_a})\,\dot{\Theta}_e(t) + n\,P(t) - \frac{n K_a}{R_a}\,v(t) - \frac{n k_a k_R}{R_a}\,T_{comp}(t) = 0 \qquad \textbf{(d.1)}$$

Donde:

J_{ef} es la inercia efectiva del lado del actuador.

B_{ef} es el coeficiente de amortiguamiento del lado del actuador.

K_a es la constante de par del motor.

K_b es la constante de la fuerza contra-electromotriz.

R_a es la resistencia del estator.

n es la relación de transformación de la transmisión.

v(t) es el voltaje del estator.

P(t) es el par generado por los otros elementos (perturbación).

θ_e es el desplazamiento angular del elemento.

$T_{comp}(t)$ es el par compensatorio de las perturbaciones.

k_R es una constante de proporcionalidad (voltaje/par).

En base a la ecuación (d.1), los parámetros utilizados para cada actuador/transmisión de potencia se dan en la tabla d.3

UNIÓN	J_{ef} $Kg \cdot m^2$	B_{ef} $\frac{N \cdot m}{rad/s}$	K_a $\frac{N \cdot m}{A}$	K_b $V \cdot s$	R_a Ω	n
1	50×10^{-6}	40×10^{-6}	106×10^{-3}	21.01×10^{-3}	26	1/20
2	30×10^{-6}	40×10^{-6}	106×10^{-3}	21.01×10^{-3}	26	1/50
3	25×10^{-6}	35×10^{-6}	106×10^{-3}	21.01×10^{-3}	26	1/15
4	20×10^{-6}	30×10^{-6}	106×10^{-3}	21.01×10^{-3}	26	1/5
5	20×10^{-6}	30×10^{-6}	106×10^{-3}	21.01×10^{-3}	26	1/5
6	20×10^{-6}	30×10^{-6}	106×10^{-3}	21.01×10^{-3}	26	1/5

Tabla d.3

APENDICE E

Esquemáticos:

*Procesador 1 (tarjeta μP1):

-Procesador.

-Memoria.

-Memoria compartida.

-Puerto de control.

-Teclado/desplegado.

*Procesador 2 (tarjeta μP2):

-Procesador.

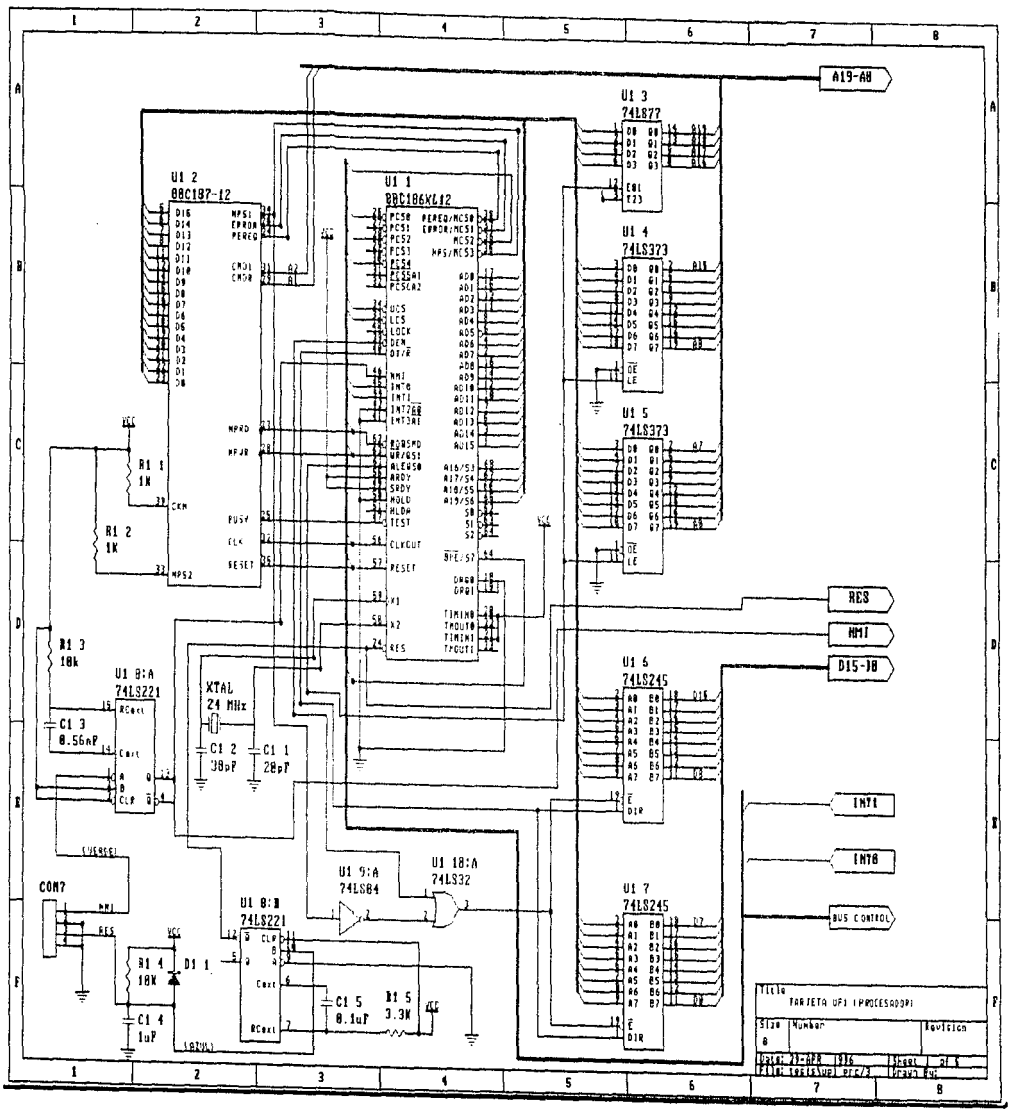
-Memoria.

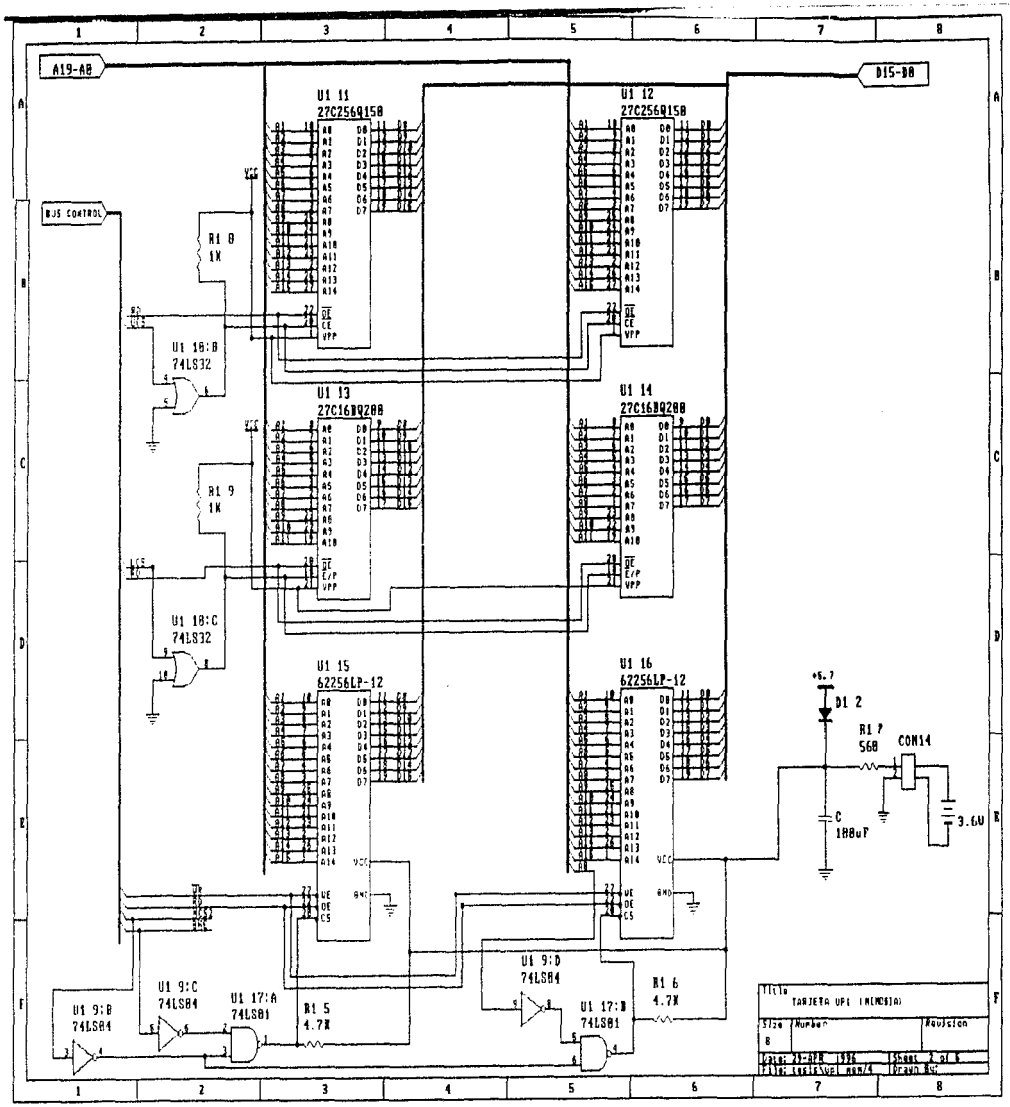
-Puerto de control.

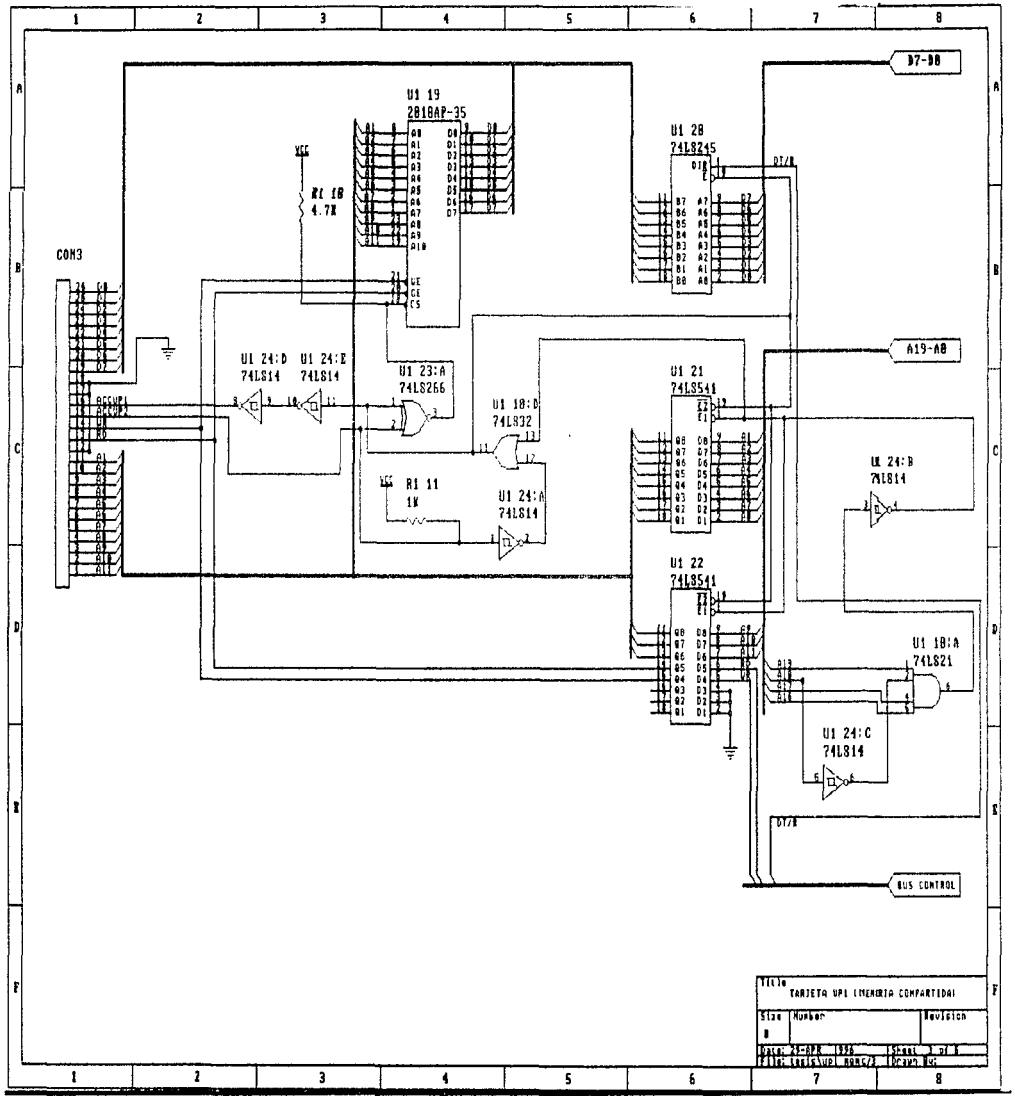
-Conversión A/D.

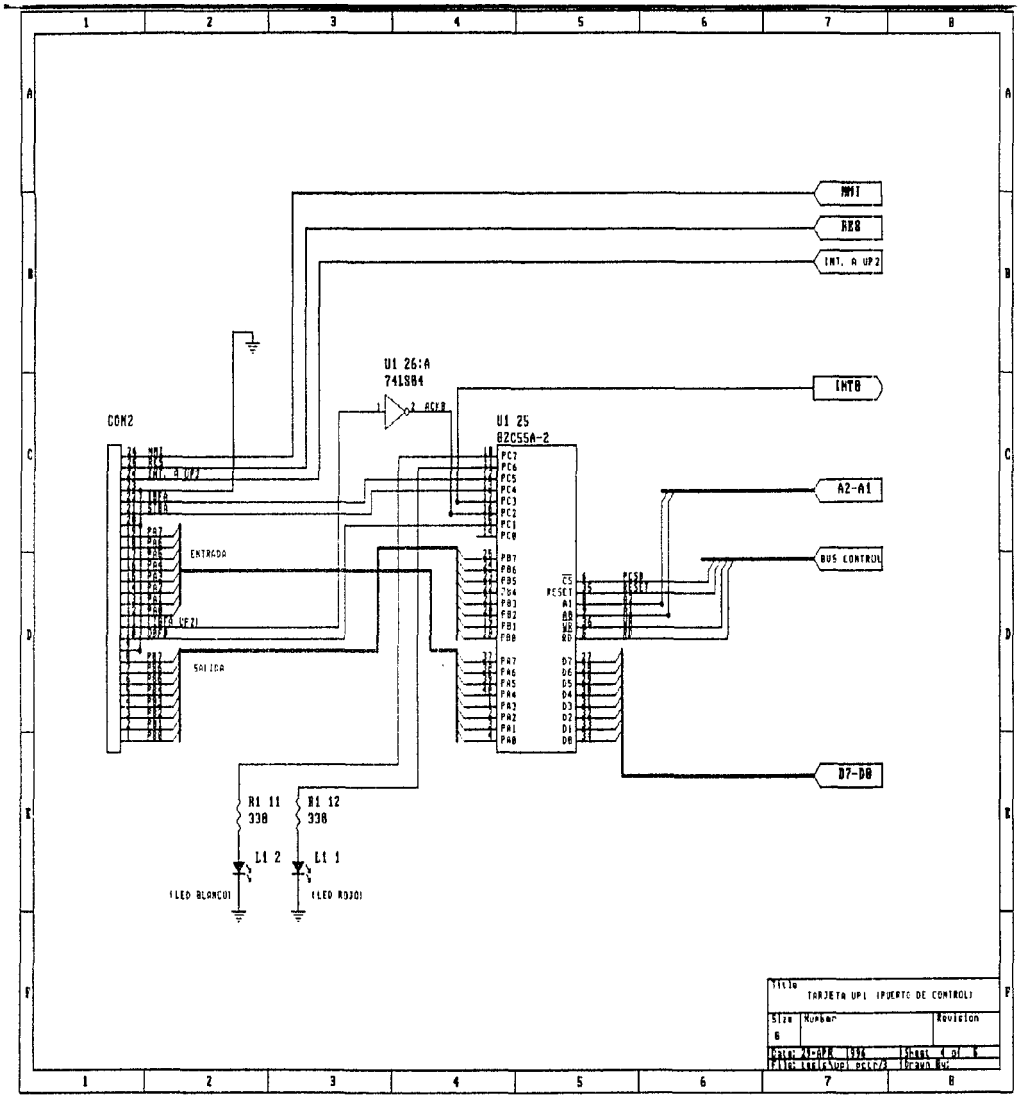
-Conversión D/A.

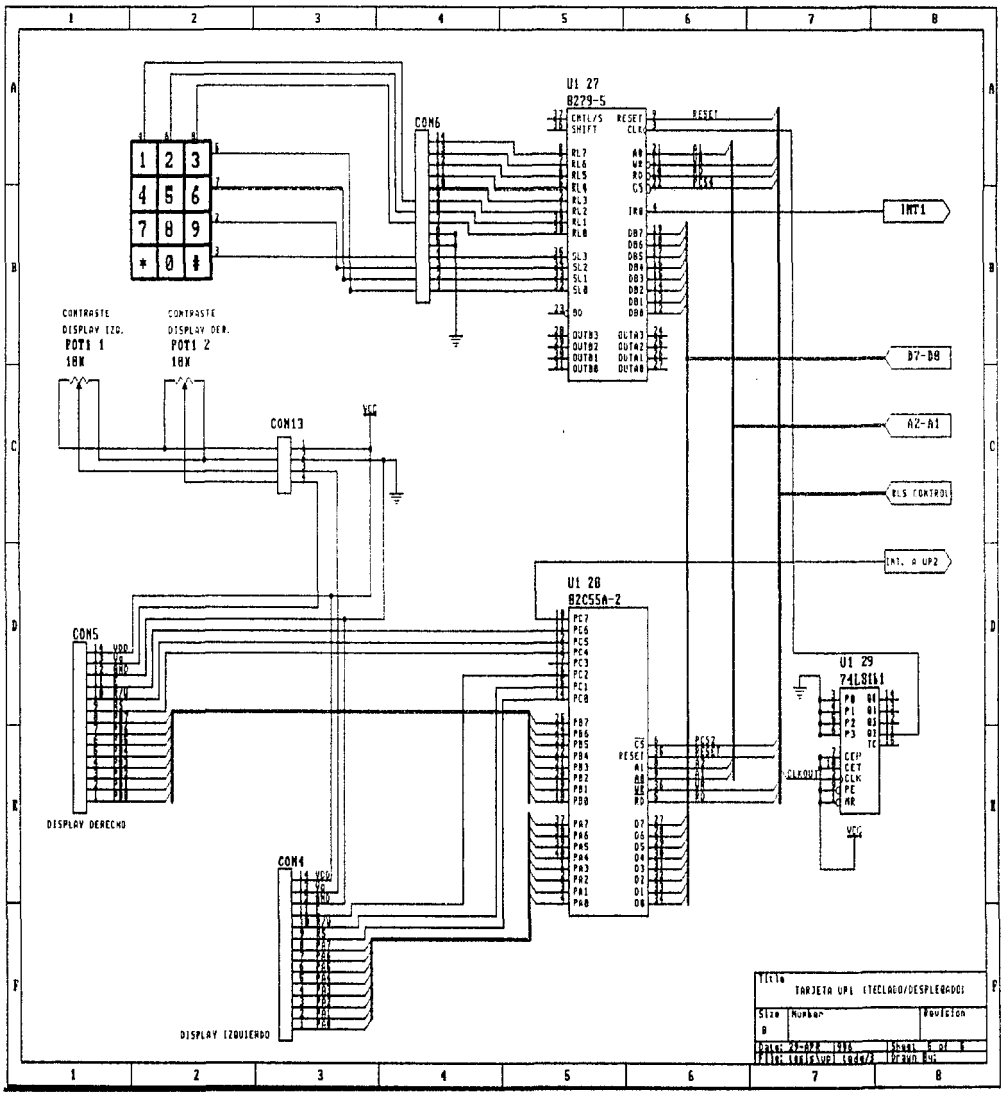
* Fuente.

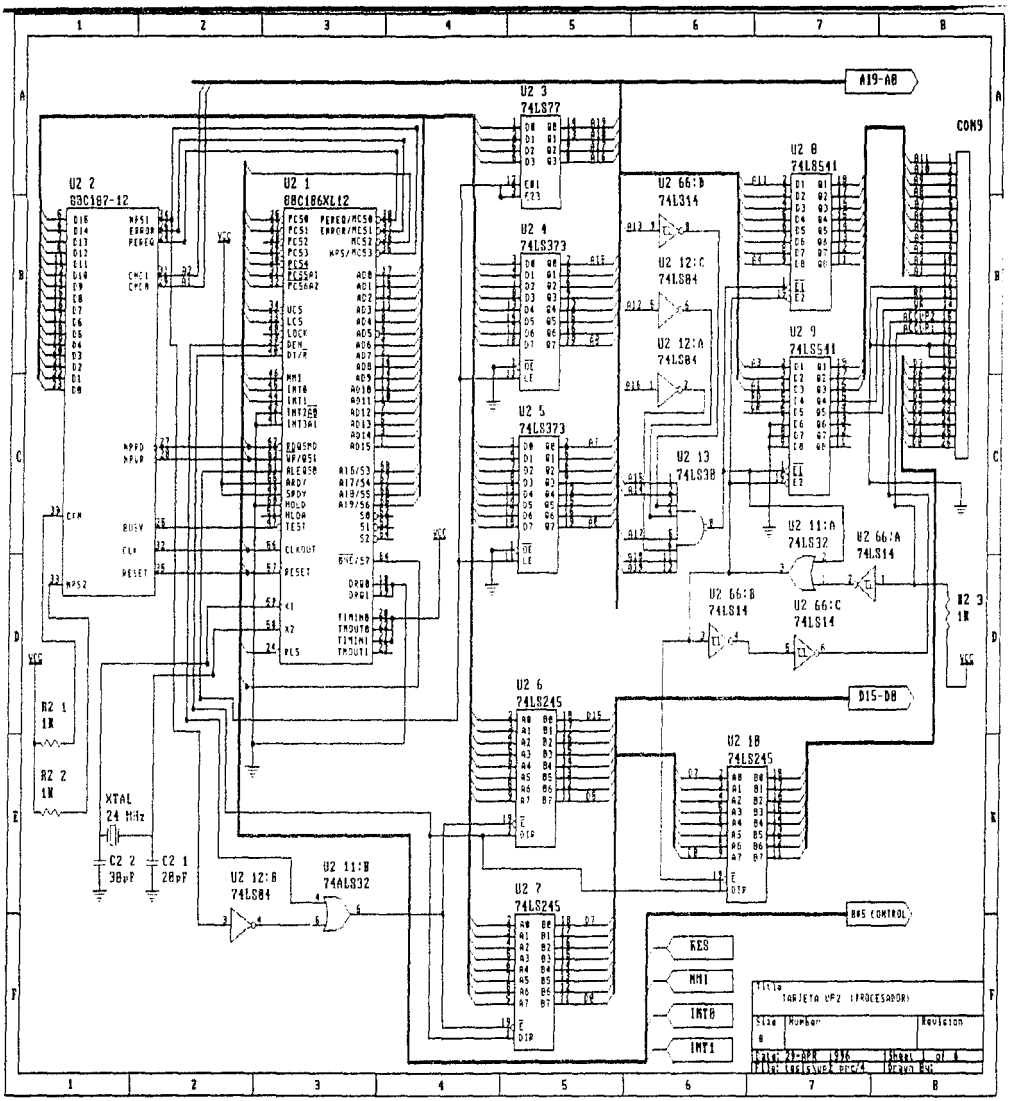

A19-A8
U1 3
74LS877
U1 2
80C187-12
U1 1
80C186XL12
U1 4
74LS373
U1 5
74LS373
R1 1
1K
R1 2
1K
R1 3
10k
U1 8:A
74LS221
XTAL
24 MHz
C1 3
0.56nF
C1 2
30pF
C1 1
20pF
U1 6
74LS245
RES
NMI
D15-D8
INT1
INT0
BUS CONTROL
COM7
U1 9:A
74LS04
U1 10:A
74LS32
U1 8:B
74LS221
U1 7
74LS245
R1 4
10K
D1 1
C1 4
1uF
C1 5
0.1uF
R1 5
3.3K
Title
TARJETA UP1 (PROCESADOR)
Size
B
Number
Revision
Date: 29-APR-96
Sheet 2 of 5
File: tarjeta\up1.prj/3
Drawn By:

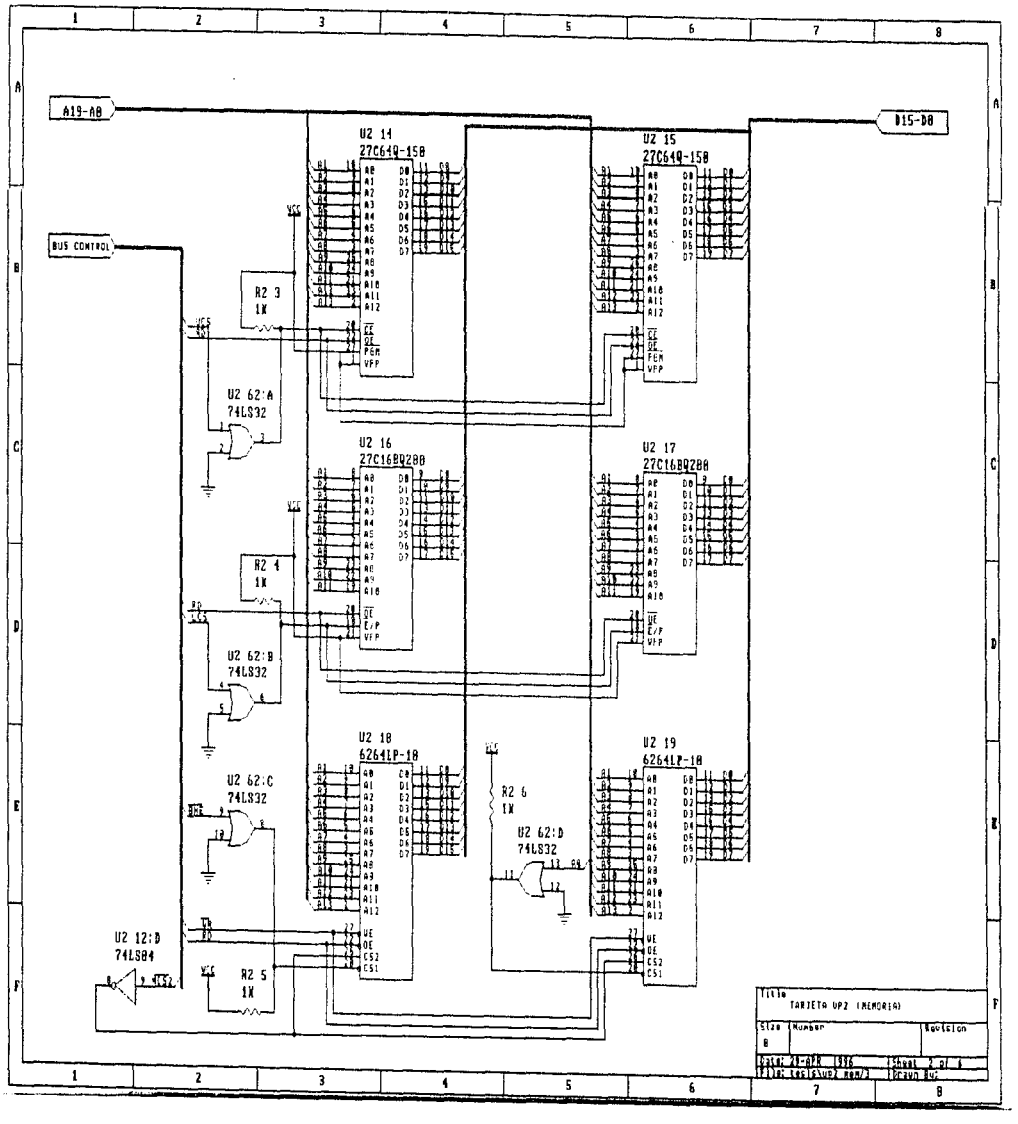

A19-A8
D15-D8
BUS CONTROL
U1 11
27C256Q15B
U1 12
27C256Q15B
U1 13
27C16BQ20B
U1 14
27C16BQ20B
U1 15
62256LP-12
U1 16
62256LP-12
R1 8
1K
R1 9
1K
U1 18:B
74LS32
U1 18:C
74LS32
U1 9:B
74LS04
U1 9:C
74LS04
U1 9:D
74LS04
U1 17:A
74LS81
U1 17:B
74LS81
R1 5
4.7K
R1 6
4.7K
+5.7
D1 2
R1 7
56Ω
CON14
C
100uF
3.6V
Title
TARJETA UP1 (MEMORIA)
Size
B
Number
Revision
Date: 25-APR-1995
Sheet 2 of 5
File: tarjeta\up1.mem/4
Drawn By:

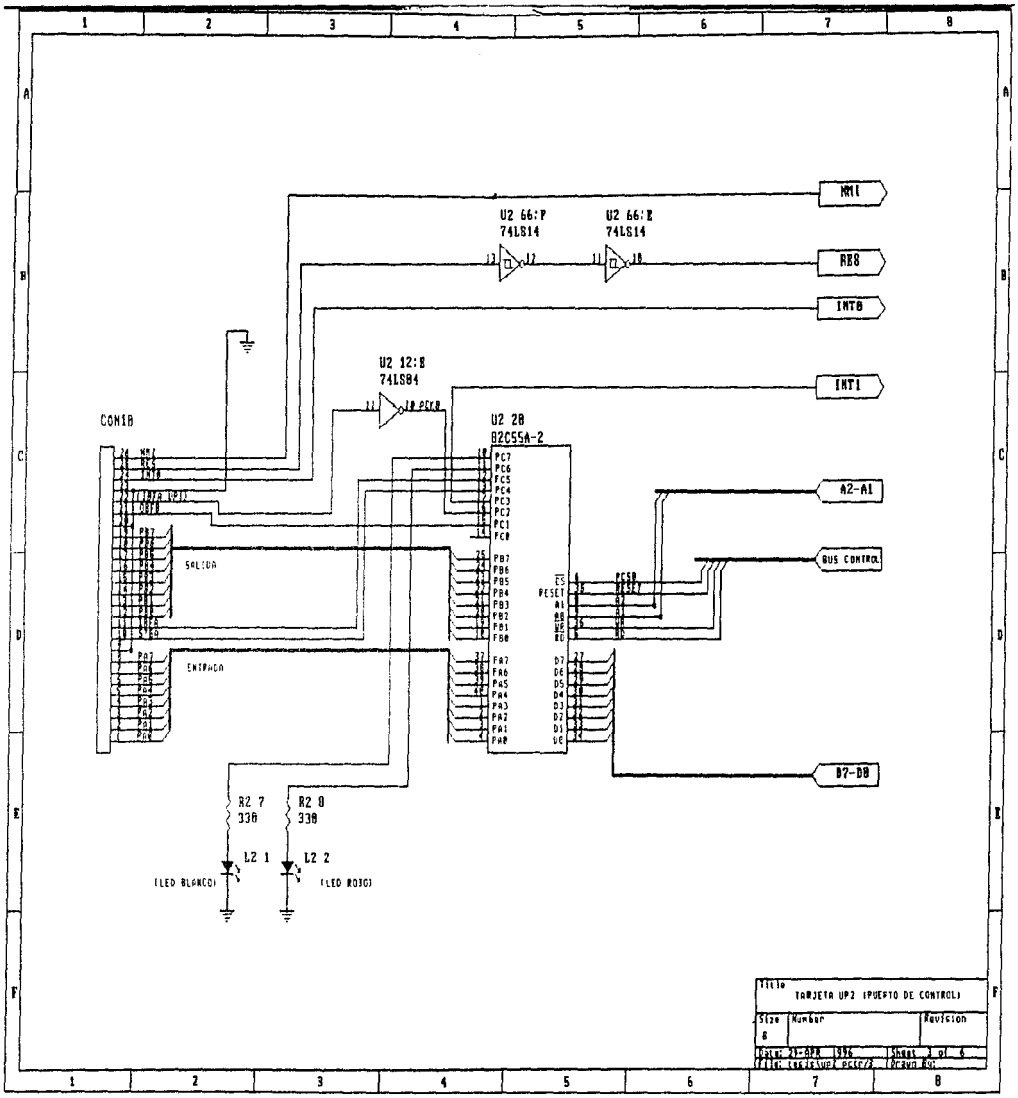

D7-D0
U1 19
2816AP-35
VCC
R1 10
4.7K
CON3
U1 28
74LS245
DIR
U1 24:D
74LS14
U1 24:E
74LS14
U1 23:A
74LS266
U1 18:D
74LS32
A19-A0
U1 21
74LS541
U1 24:B
74LS14
R1 11
1K
U1 24:A
74LS14
U1 22
74LS541
U1 18:A
74LS21
U1 24:C
74LS14
DIR
BUS CONTROL
Title
TARJETA UP1 (MEMORIA COMPARTIDA)
Size
B
Number
Revision
Date: 25-APR-1990
Sheet 3 of 8
File: LAB1&\UP1_RAM.SCH
Drawn by:

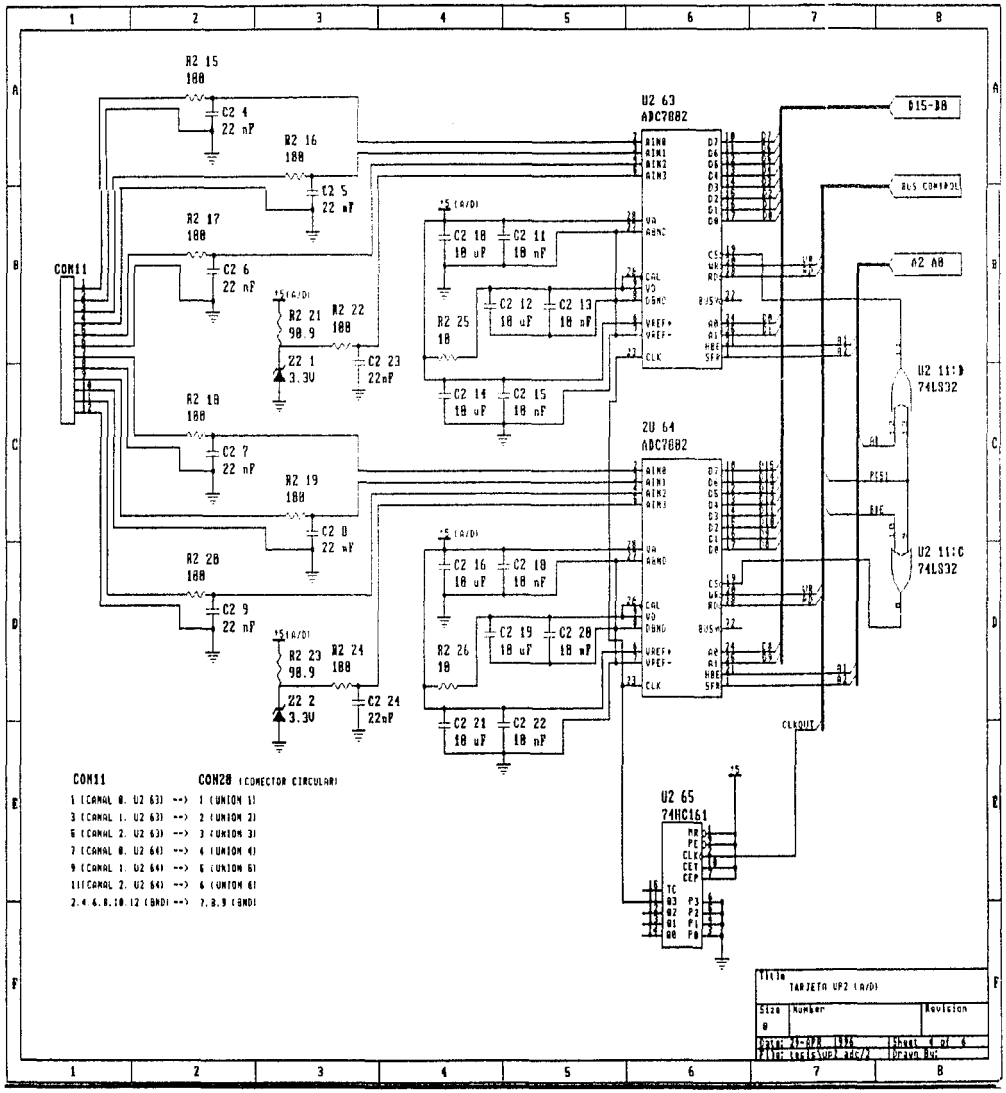

MMI
RES
INT. A UP2
INT8
A2-A1
BUS CONTROL
D7-D0
U1 26:A
74LS04
U1 25
82C55A-2
CON2
ENTRADA
SALIDA
R1 11
330
R1 12
330
L1 2
L1 1
(LED BLANCO)
(LED ROJO)
Title
TARJETA UP1 (PUERTO DE CONTROL)
Size
B
Number
Revision
Date: 21-APR-1996
Sheet 4 of 8
File: C:\ELE\SUP1 PCB\3
Drawn By:

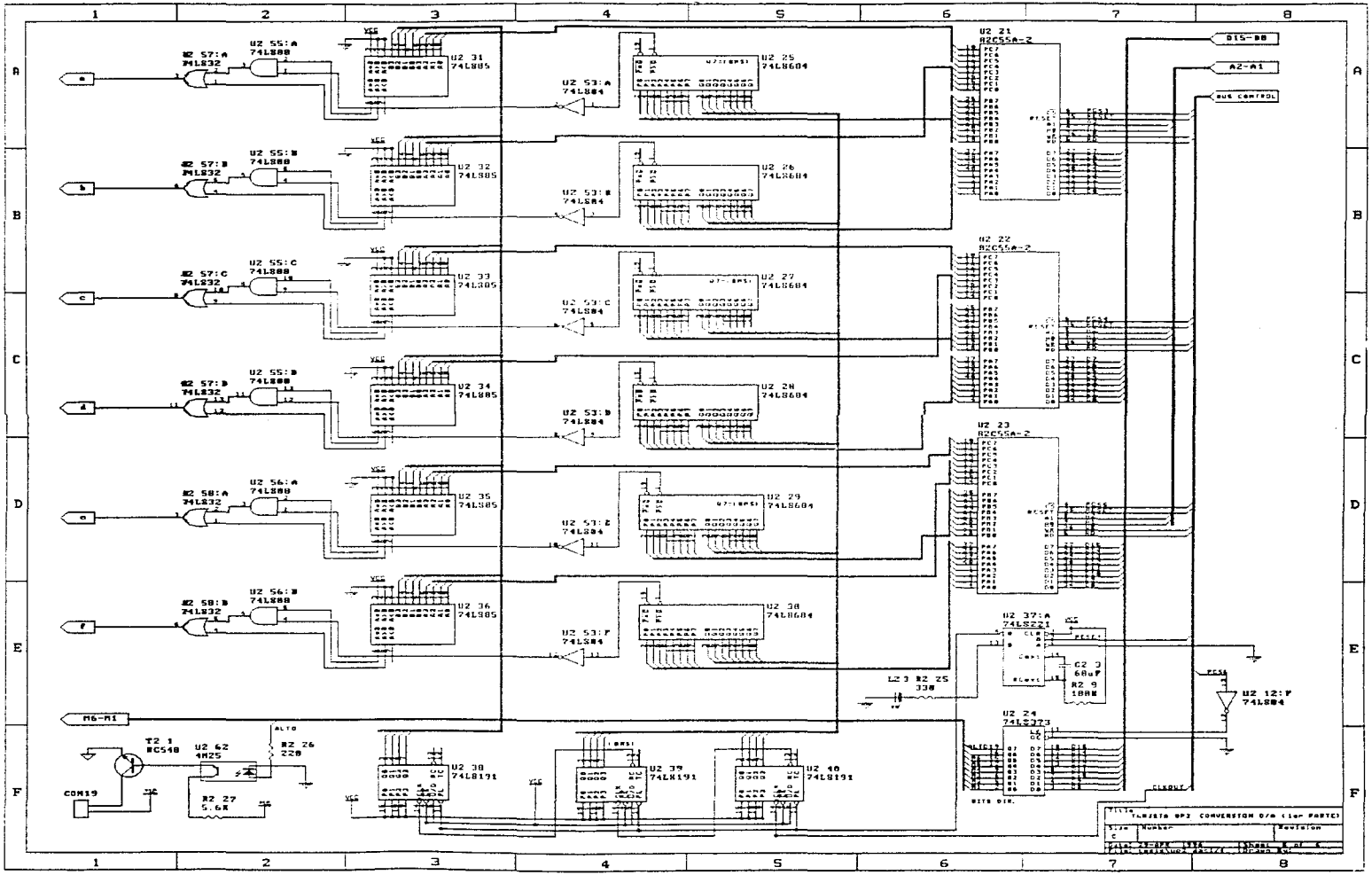

U1 27
8279-5
CNTL/S SHIFT
RESET
CLK
RESET
RL7
RL6
RL5
RL4
RL3
RL2
RL1
RL0
SL3
SL2
SL1
SL0
BD
OUTB3
OUTB2
OUTB1
OUTB0
A0
WR
RD
CS
IRQ
DB7
DB6
DB5
DB4
DB3
DB2
DB1
DB0
OUTA3
OUTA2
OUTA1
OUTA0
CON6
INT1
D7-D0
A2-A1
RLS CONTROL
INT. A UP2
1 2 3
4 5 6
7 8 9
* 0 #
CONTRASTE
DISPLAY IZQ.
POT1 1
10K
CONTRASTE
DISPLAY DER.
POT1 2
10K
CON13
VCC
U1 28
82C55A-2
PC7
PC6
PC5
PC4
PC3
PC2
PC1
PC0
PB7
PB6
PB5
PB4
PB3
PB2
PB1
PB0
PA7
PA6
PA5
PA4
PA3
PA2
PA1
PA0
CS
RESET
A1
A0
WR
RD
D7
D6
D5
D4
D3
D2
D1
D0
PCS2
CON5
VDD
VSS
DISPLAY DERECHO
U1 29
74LS161
P0
P1
P2
P3
Q0
Q1
Q2
Q3
TC
CEP
CET
CLK
PE
MR
CLKOUT
VCC
CON4
DISPLAY IZQUIERDO
Title
TARJETA UP1 (TECLADO/DESPLEGADO)
Size
B
Number
Revision
Date: 25-APR-1996
Sheet 5 of 8
File: LOGIS\UP1.SCH
Drawn By:

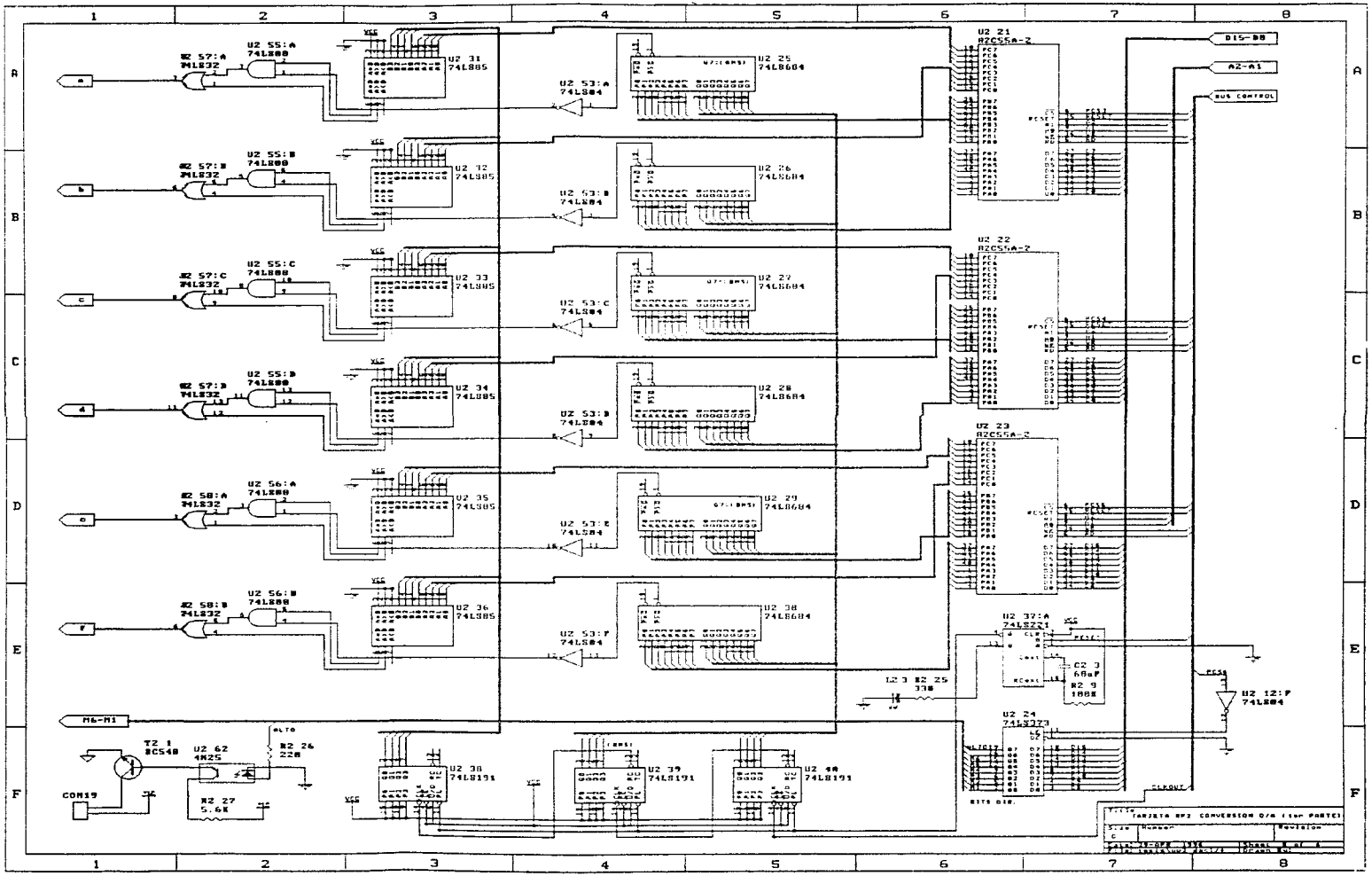

A19-A8
D15-D8
BUS CONTROL
CON9
U2 1
80C186XL12
U2 2
83C187-12
U2 3
74LS377
U2 4
74LS373
U2 5
74LS373
U2 6
74LS245
U2 7
74LS245
U2 8
74LS541
U2 9
74LS541
U2 10
74LS245
U2 11:A
74LS32
U2 11:B
74LS32
U2 12:A
74LS04
U2 12:B
74LS04
U2 12:C
74LS04
U2 13
74LS38
U2 66:A
74LS14
U2 66:B
74LS14
U2 66:C
74LS14
R2 1
1K
R2 2
1K
R2 3
1K
XTAL
24 MHz
C2 2
30pF
C2 1
20pF
VCC
RES
NMI
INT0
INT1
Title
TARJETA UP2 (PROCESADOR)
Size
B
Number
Revision
Date: 25-APR-96
Sheet of
File: LOGISUP2.PRC/4
Drawn By:

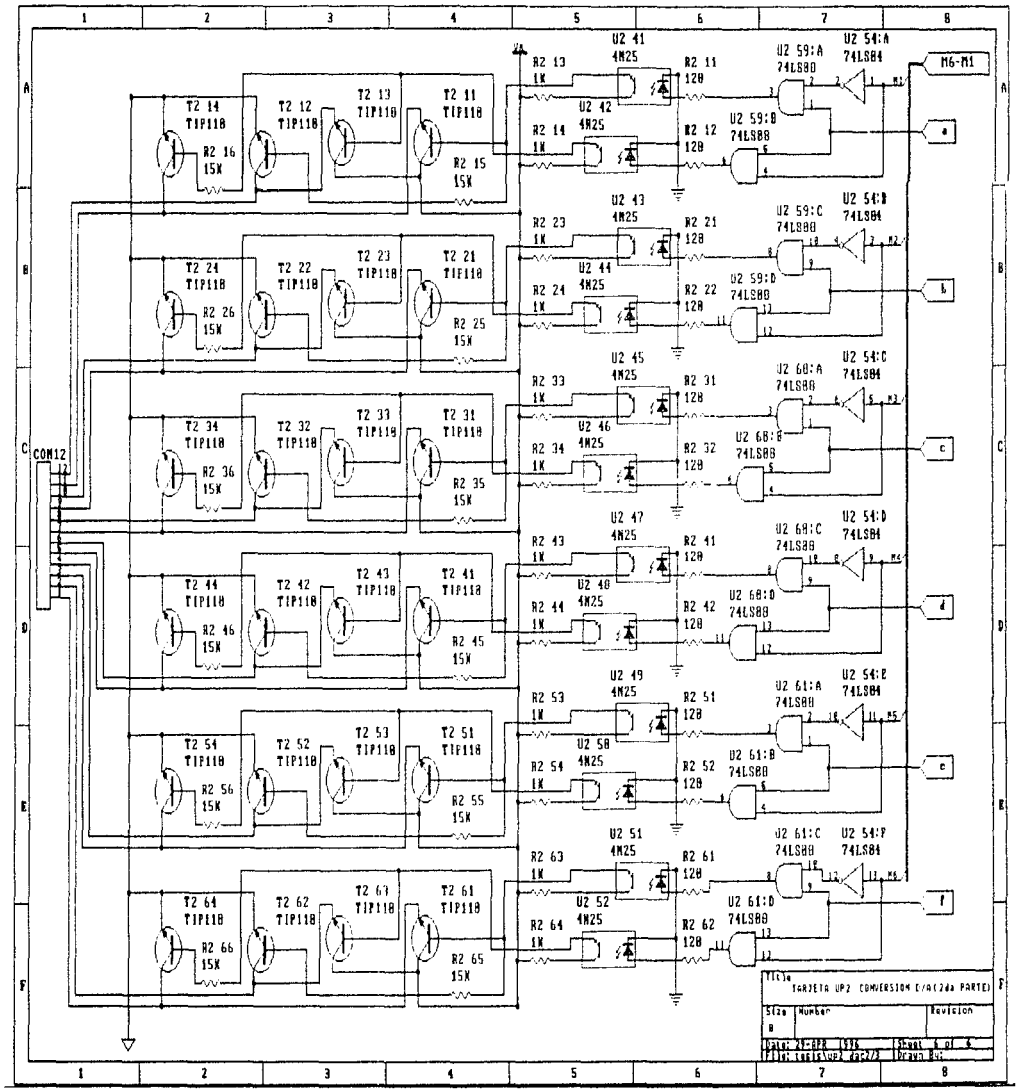

A19-A8
D15-D8
BUS CONTROL
U2 14
27C64Q-15B
U2 15
27C64Q-15B
U2 16
27C16BQZ8B
U2 17
27C16BQZ8B
U2 18
62641P-10
U2 19
62641P-10
R2 3
1K
R2 4
1K
R2 5
1K
R2 6
1K
U2 62:A
74LS32
U2 62:B
74LS32
U2 62:C
74LS32
U2 62:D
74LS32
U2 12:D
74LS04
VCC
A0
A1
A2
A3
A4
A5
A6
A7
A8
A9
A10
A11
A12
A13
A14
D0
D1
D2
D3
D4
D5
D6
D7
CE
OE
PGM
VPP
RE
E/P
VE
CS2
CS1
Title
TARJETA UP2 (MEMORIA)
Size
B
Number
Revision
Date: 28-APR-1996
Sheet 2 of 6
File: C:\EAGLE\UP2.MEM\2
Drawn By:

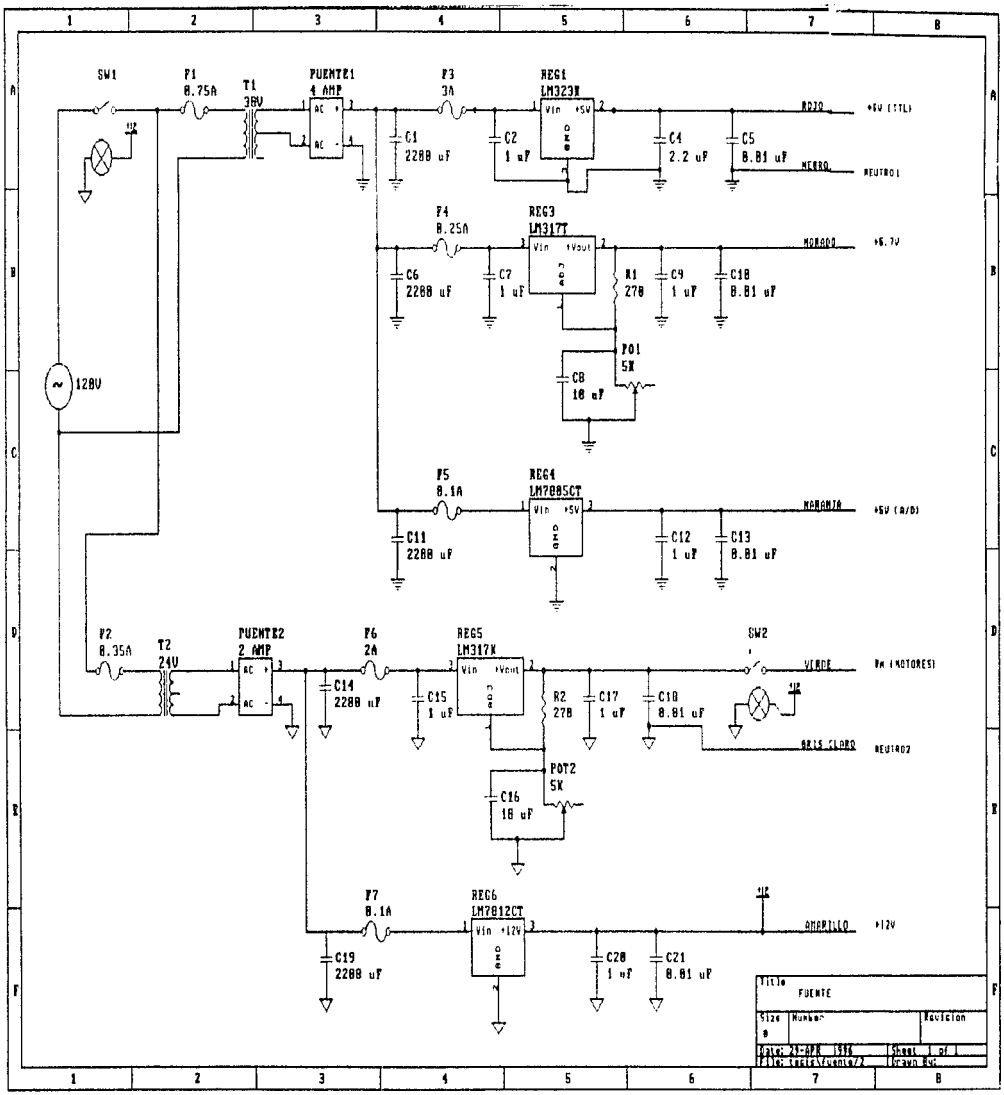

NMI
RES
INT8
INT1
A2-A1
BUS CONTROL
D7-D0
U2 66:F
74LS14
U2 66:E
74LS14
U2 12:E
74LS84
U2 28
82C55A-2
COM1B
SALIDA
ENTRADA
R2 7
330
R2 0
330
L2 1
(LED BLANCO)
L2 2
(LED ROJO)
TARJETA UP2 (PUERTO DE CONTROL)


R2 15
100
C2 4
22 nF
R2 16
100
C2 5
22 nF
R2 17
100
C2 6
22 nF
R2 18
100
C2 7
22 nF
R2 19
100
C2 8
22 nF
R2 20
100
C2 9
22 nF
CON11
R2 21
98.9
R2 22
100
Z2 1
3.3V
C2 23
22nF
R2 23
98.9
R2 24
100
Z2 2
3.3V
C2 24
22nF
U2 63
ADC7882
U2 64
ADC7882
C2 18
10 uF
C2 11
10 nF
C2 12
10 uF
C2 13
10 nF
R2 25
10
C2 14
10 uF
C2 15
10 nF
C2 16
10 uF
C2 10
10 nF
C2 19
10 uF
C2 20
10 nF
R2 26
10
C2 21
10 uF
C2 22
10 nF
D15-D8
BUS CONTROL
A2 A0
U2 11:B
74LS32
U2 11:C
74LS32
CLKOUT
U2 65
74HC161
CON11
1 (CANAL 0. U2 63) --> 1 (UNION 1)
3 (CANAL 1. U2 63) --> 2 (UNION 2)
5 (CANAL 2. U2 63) --> 3 (UNION 3)
7 (CANAL 0. U2 64) --> 4 (UNION 4)
9 (CANAL 1. U2 64) --> 5 (UNION 5)
11(CANAL 2. U2 64) --> 6 (UNION 6)
2.4.6.8.10.12 (GND) --> 7.8.9 (GND)
CON28 (CONECTOR CIRCULAR)
Title
TARJETA UP2 (A/D)
Size
B
Number
Revision
Date: 23-FEB-1996
Sheet 4 of 6
File: C:\PCB\UP2_ADC\2
Drawn By:


U2 55:A 74LS08
U2 31 74LS85
U2 53:A 74LS04
U2 25 74LS604
U2 21 82C55A-2
D15-D8
A2-A1
BUS CONTROL
U2 55:B 74LS08
U2 32 74LS85
U2 53:B 74LS04
U2 26 74LS604
U2 55:C 74LS08
U2 33 74LS85
U2 53:C 74LS04
U2 27 74LS604
U2 22 82C55A-2
U2 55:D 74LS08
U2 34 74LS85
U2 53:D 74LS04
U2 28 74LS604
U2 23 82C55A-2
U2 56:A 74LS08
U2 35 74LS85
U2 53:E 74LS04
U2 29 74LS604
U2 56:B 74LS08
U2 36 74LS85
U2 53:F 74LS04
U2 30 74LS604
U2 37:A 74LS221
C2 3 68uF
R2 9 10KR
U2 12:F 74LS04
U2 24 74LS273
R2 25 330
M6-M1
T2 1 BC548
U2 62 4N25
R2 26 220
R2 27 5.6K
COM19
U2 38 74LS191
U2 39 74LS191
U2 40 74LS191
CLKOUT
TARJETA GP2 CONVERSION D/A (1er PARTE)


U2 55:A 74LS08
U2 55:B 74LS08
U2 55:C 74LS08
U2 55:D 74LS08
U2 56:A 74LS08
U2 56:B 74LS08
U2 57:A 74LS32
U2 57:B 74LS32
U2 57:C 74LS32
U2 57:D 74LS32
U2 58:A 74LS32
U2 58:B 74LS32
U2 31 74LS05
U2 32 74LS05
U2 33 74LS05
U2 34 74LS05
U2 35 74LS05
U2 36 74LS05
U2 53:A 74LS04
U2 53:B 74LS04
U2 53:C 74LS04
U2 53:D 74LS04
U2 53:E 74LS04
U2 53:F 74LS04
U2 25 74LS684
U2 26 74LS684
U2 27 74LS684
U2 28 74LS684
U2 29 74LS684
U2 38 74LS684
U2 Z1 82C55A-2
U2 Z2 82C55A-2
U2 Z3 82C55A-2
U2 Z4 74LS373
U2 37:A 74LS221
U2 12:F 74LS04
U2 30 74LS191
U2 39 74LS191
U2 40 74LS191
T2 1 BC548
U2 62 4N25
R2 24 220
R2 27 5.6K
R2 25 330
C2 3 68nF
R2 9 100K
D15-D8
A2-A1
BUS CONTROL
M6-M1
COM19
CLKOUT
TARJETA RF2 CONVERSION D/A (1ª PARTE)


COM12
T2 14 TIP118
T2 12 TIP118
T2 13 TIP118
T2 11 TIP118
R2 16 15K
R2 15 15K
T2 24 TIP118
T2 22 TIP118
T2 23 TIP118
T2 21 TIP118
R2 26 15K
R2 25 15K
T2 34 TIP118
T2 32 TIP118
T2 33 TIP118
T2 31 TIP118
R2 36 15K
R2 35 15K
T2 44 TIP118
T2 42 TIP118
T2 43 TIP118
T2 41 TIP118
R2 46 15K
R2 45 15K
T2 54 TIP118
T2 52 TIP118
T2 53 TIP118
T2 51 TIP118
R2 56 15K
R2 55 15K
T2 64 TIP118
T2 62 TIP118
T2 63 TIP118
T2 61 TIP118
R2 66 15K
R2 65 15K
+5
R2 13 1K
R2 14 1K
R2 23 1K
R2 24 1K
R2 33 1K
R2 34 1K
R2 43 1K
R2 44 1K
R2 53 1K
R2 54 1K
R2 63 1K
R2 64 1K
U2 41 4N25
U2 42 4N25
U2 43 4N25
U2 44 4N25
U2 45 4N25
U2 46 4N25
U2 47 4N25
U2 48 4N25
U2 49 4N25
U2 50 4N25
U2 51 4N25
U2 52 4N25
R2 11 120
R2 12 120
R2 21 120
R2 22 120
R2 31 120
R2 32 120
R2 41 120
R2 42 120
R2 51 120
R2 52 120
R2 61 120
R2 62 120
U2 59:A 74LS08
U2 59:B 74LS08
U2 59:C 74LS08
U2 59:D 74LS08
U2 60:A 74LS08
U2 60:B 74LS08
U2 60:C 74LS08
U2 60:D 74LS08
U2 61:A 74LS08
U2 61:B 74LS08
U2 61:C 74LS08
U2 61:D 74LS08
U2 54:A 74LS04
U2 54:B 74LS04
U2 54:C 74LS04
U2 54:D 74LS04
U2 54:E 74LS04
U2 54:F 74LS04
M6-M1
a
b
c
d
e
f
Title
TARJETA UP2 CONVERSION D/A(2da PARTE)
Size
B
Number
Revision
Date: 24-FEB-1996
Sheet 6 of 6
File: \DATOS\UP2_4DA.SCH
Drawn By:


SW1
F1
0.75A
T1
36V
PUENTE1
4 AMP
F3
3A
REG1
LM323K
C1
2200 uF
C2
1 uF
C4
2.2 uF
C5
0.01 uF
ROJO
+5V (TTL)
NEGRO
NEUTRO1
F4
0.25A
REG3
LM317T
C6
2200 uF
C7
1 uF
R1
270
C9
1 uF
C10
0.01 uF
NARANJA
+5.7V
POT1
5K
C8
10 uF
120V
F5
0.1A
REG4
LM7805CT
C11
2200 uF
C12
1 uF
C13
0.01 uF
NARANJA
+5V (A/D)
F2
0.35A
T2
24V
PUENTE2
2 AMP
F6
2A
REG5
LM317K
C14
2200 uF
C15
1 uF
R2
270
C17
1 uF
C18
0.01 uF
SW2
VERDE
0V (MOTORES)
GRIS CLARO
NEUTRO2
POT2
5K
C16
10 uF
F7
0.1A
REG6
LM7812CT
C19
2200 uF
C20
1 uF
C21
0.01 uF
AMARILLO
+12V
Title
FUENTE
Size
B
Number
Revision
Date: 21-APR-1994
Sheet 1 of 1
File: tesis\fuente2
Drawn By:

APENDICE F

Hojas de datos:

-80C186XL

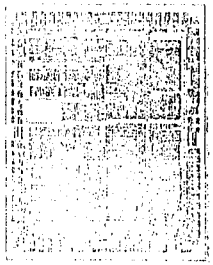
-AND491

-ADC7802

80C186XL/80C188XL
16-BIT HIGH-INTEGRATION EMBEDDED PROCESSORS

- Low Power, Fully Static Versions of 80C186/80C188
- Operation Modes:
  - Enhanced Mode
    - DRAM Refresh Control Unit
    - Power-Save Mode
    - Direct Interface to 80C187 (80C186XL Only)
  - Compatible Mode
    - HMOS 80186/80188 Pin-for-Pin Replacement for Non-Numerics Applications
- Integrated Feature Set
  - Static, Modular CPU
  - Clock Generator
  - 2 Independent DMA Channels
  - Programmable Interrupt Controller
  - 3 Programmable 16 Bit Timers
  - Dynamic RAM Refresh Control Unit
  - Programmable Memory and Peripheral Chip Select Logic
  - Programmable Wait State Generator
  - Local Bus Controller
  - Power-Save Mode
  - System-Level Testing Support (High Impedance Test Mode)
- Completely Object Code Compatible with Existing 8086/8088 Software and Has 10 Additional Instructions over 8086/8088
- Speed Versions Available
  - 20 MHz (80C186XL20/80C188XL20)
  - 12 MHz (80C186XL12/80C188XL12)
- Direct Addressing Capability to 1 Mbyte Memory and 64 Kbyte I/O
- Complete System Development Support
  - ASM 86 Assembler, PL/M-86, iC-86 and System Utilities
  - In-Circuit Emulator
- Available in 68-Pin
  - Plastic Leaded Chip Carrier (PLCC)
  - Ceramic Pin Grid Array (PGA)
  - Ceramic Leadless Chip Carrier (JEDEC A Package)
- Available in 80-Pin:
  - Quad Flat Pack (EIAJ)
  - Shrink Quad Flat Pack (SQFP)
- Available in Extended Temperature Range (−40°C to +85°C)

The Intel 80C186XL is a Modified Core implementation of the 80C186 Microprocessor. It offers higher speed and lower power consumption than the standard 80C186, but maintains 100% clock-for-clock functional compatibility. Packaging and pinout are also identical.



272189-1

CONTENTS PAGE

CONTENTS PAGE

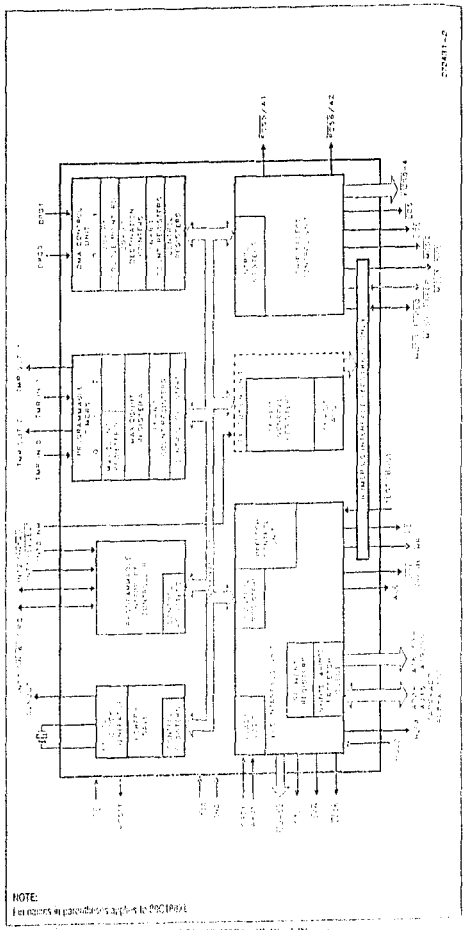


Figure 1. 80C186XL/80C188XL Block Diagram

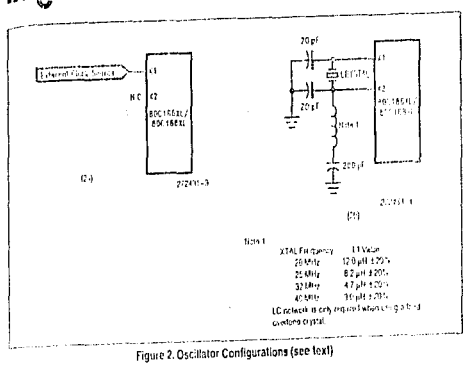


Figure 2. Oscillator Configurations (see text)

INTRODUCTION

Unless specifically noted, all references to the 80C186XL apply to the 80C188XL. References to pins that differ between the 80C186XL and the 80C188XL are given in parentheses.

The following Functional Description describes the base architecture of the 80C186XL. The 80C186XL is a very high integration 16-bit microprocessor. It combines 15–20 of the most common microprocessor system components onto one chip. The 80C186XL is object code compatible with the 80C86/80C88 microprocessors and adds 10 new instruction types to the 8086/8088 instruction set.

The 80C186XL has two major modes of operation, Compatible and Enhanced. In Compatible Mode (the default) the 80C186XL is completely compatible with NMOS 80186, with the exception of 8087 support. The Enhanced mode adds three new features to the system design. These are Power-Save control, Dynamic RAM refresh, and an asynchronous Numerics Coprocessor interface (80C186XL only).

80C186XL CORE ARCHITECTURE

80C186XL Clock Generator

The 80C186XL provides an on-chip clock generator for both internal and external clock generation. The clock generator features a crystal oscillator, a divide-by-two counter, synchronous and asynchronous ready inputs, and reset circuitry.

The 80C186XL oscillator circuit is designed to be used either with a parallel resonant fundamental or third-overtone mode crystal, depending upon the frequency range of the application. This is used as the time base for the 80C186XL.

The output of the oscillator is not directly available outside the 80C186XL. The recommended crystal configuration is shown in Figure 2b. When used in third-overtone mode, the tank circuit is recommended for stable operation. Alternately, the oscillator may be driven from an external source as shown in Figure 2a.

The crystal or clock frequency chosen must be twice the required processor operating frequency due to the internal divide by two counter. This counter is used to drive all internal phase clocks and the external CLKOUT signal. CLKOUT is a 50% duty cycle processor clock and can be used to drive other system components. All AC Timings are referenced to CLKOUT.

Intel recommends the following values for crystal selection parameters:

Temperature Range	Application Specific
ESR (Equivalent Series Resistance)	60Ω max
C_0 (Shunt Capacitance of Crystal)	7.0 pF max
C_1 (Load Capacitance)	20 pF ±2 pF
Drive Level	2 mW max

Bus Interface Unit

The 80C186XL provides a local bus controller to generate the local bus control signals. In addition, it employs a HOLD/HLDA protocol for relinquishing the local bus to other bus masters. It also provides outputs that can be used to enable external bufiers and to direct the flow of data on and off the local bus.

The bus controller is responsible for generating the set of address, read and write strobes, bus cycle status information and data (for write operations) information. It is also responsible for reading data from the local bus during a read operation. Synchronous and asynchronous ready input pins are provided to extend a bus cycle beyond the minimum four states (clocks).

The 80C186XL bus controller also generates bus control signals (DEN and DT/R) when interfacing to external transceiver chips. This capability allows the addition of transceivers for simple buffering of the multiplexed address/data bus.

During RESET the local bus controller will perform the following action:

- Drive ALE, RD and WR HIGH for one clock cycle, then float them
- Drive $\overline{SO}$–$\overline{S2}$ to the inactive state (all HIGH) and then float
- Drive LOCK HIGH and then float
- Float AD0 - 15 (AD7 - 0), A16 - 19 (A9 - A12), BHE (RFSH), DT/R
- Drive A1, LOW
- Drive HLDA LOW.

RD/QSMD, UCS, LCS, MCS0/PEREQ, MCS1/ERROR and R S1/BUST pins have internal pullup devices which are active when RES is applied. Consequently, floating or grounding certain of these pins causes the appropriate mode to enter an alternative mode of operation:

- RD/QSMD low enables Queue Status Mode
- UCS and LCS low enable ONCE Mode
- RESET/80C186 low (and high later) selects 8 bit Internal Mode

80C186XL PERIPHERAL ARCHITECTURE

All the 80C186XL integrated peripherals are controlled by 16-bit registers contained within an internal 256 byte control block. The control block may be mapped into either memory or I/O space. Internal logic will recognize control block addresses and respond to bus cycles. An offset map of the 256-byte control register block is shown in Figure 3.

Chip-Select/Ready Generation Logic

The 80C186XL contains logic which provides programmable chip-select generation for both memories and peripherals. In addition, it can be programmed to provide READY (or WAIT state) generation. It can also provide latched address bits A1 and A2. The chip-select lines are active for all memory and I/O cycles in their programmed areas, whether they be generated by the CPU or by the integrated DMA unit.

The 80C186XL provides 6 memory chip select outputs for 3 address areas, upper memory, lower memory, and intermediate memory. Optionally provided for each of memory and lower memory, while four are provided for intermediate memory.

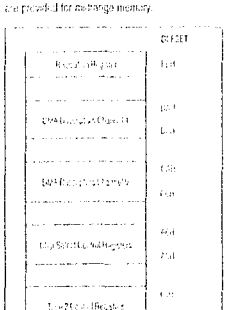


Figure 3. Internal Register Map

The 80C186XL provides a chip select, called UCS for the top of memory. The top of memory is usually used as the system memory because after reset the 80C186XL begins executing at memory location FFFF0H.

The 80C186XL provides a chip select for low memory called LCS. The bottom of memory contains the interrupt vector table, starting at location 00000H.

The 80C186XL provides four MCS lines which are active within a user-locatable memory block. This block can be located within the 80C186XL 1 Mbyte memory or I/O address space exclusive of the areas defined by UCS and LCS. Both the base address and size of this memory block are programmable.

The 80C186XL can generate chip selects for up to seven peripheral devices. These chip selects are active for seven contiguous blocks of 128 bytes above a programmable base address. The base address may be located in either memory or I/O space.

The 80C186XL can generate a READY signal internally for each of the memory or peripheral CS lines. The number of WAIT states to be inserted for each peripheral or memory is programmable to provide 0–3 wait states for all accesses to the area for which the chip select is active. In addition, the 80C186XL may be programmed to either ignore external READY for each chip select range or factor external READY with the internally generated READY.

Upon RESET, the Chip Select/Ready Logic will perform the following actions:

- All chip select outputs will be driven HIGH.
- Upon leaving RESET, the UCS line will be programmed to provide chip selects to a 1K block with the accompanying READY control bits set at 011 to insert 3 wait states in conjunction with external READY (i.e., UMCS resets to FFFBH).
- No other chip select or READY control registers have any predefined values after RESET. They will not become active until the CPU accesses their control registers.

DMA Unit

The 80C186XL DMA controller provides two independent high-speed DMA channels. Data transfers can occur between memory and I/O spaces (e.g., Memory to I/O) or within the same space (e.g., Memory to Memory or I/O to I/O). Data can be transferred either in bytes (8 bits) or in words (16 bits) or from even or odd addresses.

NOTE:
Only byte transfers are possible on the 80C186XL.

Each DMA channel maintains both a 20-bit source and destination pointer which can be optionally incremented or decremented after each data transfer. Byte or word transfers can be requested. Each data transfer consumes 2 bus cycles (a minimum of 8 clocks), one cycle to fetch data and the other to store data.

Timer/Counter Unit

The 80C186XL provides three internal 16-bit programmable timers. Two of these are highly flexible and are connected to four external pins (2 per timer). They can be used to count external events, time external events, generate nonrepetitive waveforms, etc. The third timer is not connected to any external pins, and is useful for real time coding and time delay applications. In addition, the third timer can be used as a prescaler to the other two, or as a DMA request source.

Interrupt Control Unit

The 80C186XL can receive interrupts from a number of sources, both internal and external. The 80C186XL has 5 external and 2 internal interrupt sources (Timer/Counters and DMA). The internal interrupt controller serves to merge these requests on a priority basis, for individual service by the CPU.

Enhanced Mode Operation

In Compatible Mode the 80C186XL operates with all the features of the NMOS 80186, with the exception of 8087 support (i.e. no math coprocessing is possible in Compatible Mode). Queue-Status information is still available for design purposes other than 8087 support.

All the Enhanced Mode features are completely inhibited when in Compatible Mode. A write to any of the Enhanced Mode registers will have no effect, while a read will not return any valid data.

In Enhanced Mode, the 80C186XL will operate with Power-Save, DRAM refresh, and numerics coprocessor support (80C186EM only) in addition to all the Compatible Mode features.

If connected to a math coprocessor (80C187/87 only), this mode will be invoked automatically. Without an NPX, this mode can be entered by tying the RESET output signal from the 80C186XL to the TEST/BUSY input.

Queue-Status Mode

The queue-status mode is entered by strapping the RD pin low. RD is sampled at the end of RESET. If LOW, the 80C186XL will reconfigure pin 48 (ALE/QS0) to be QS0 and QS1 respectively. This mode is available on the 68, 84-pin, and PGA Compatible and Enhanced Modes.

DRAM Refresh Control Unit

The Refresh Control Unit (RCU) automatically generates DRAM refresh bus cycles. The RCU operates only in Enhanced Mode. After a programmable period of time, the RCU generates a memory read request to the BIU. If the address generated during a refresh bus cycle is within the range of a properly programmed chip select, that chip select will be activated when the BIU executes the refresh bus cycle.

Power-Save Control

The 80C186XL, when in Enhanced Mode, can enter a power saving state by internally dividing the processor clock frequency by a programmable factor. This divided frequency is also available at the CLKOUT pin.

All internal logic, including the Refresh Control Unit and the timers, have their clocks slowed down by the division factor. To maintain a real-time counter or a valid DRAM refresh rate, these peripherals must be reprogrammed when entering and leaving the power save mode.

Interface for 80C187 Math Coprocessor (80C186XL Only)

In Enhanced Mode, many of the mid-range memory chip selects are redefined according to Table 1 for use with the 80C187. The fourth chip select, MCS2 functions as in compatible mode, and may be programmed for activity with ready logic and wait states accordingly. As in Compatible Mode, MCS2 will function for one-fourth a programmed block size.

Table 1. MCS Assignments

Compatible Mode	Enhanced Mode
MCS0	PEREQ Processor Extension Request
MCS1	ERROR NPX Error
MCS2	MCS2 Mid-Range Chip Select
MCS3	NPS Numeric Processor Select

ONCE Test Mode

To facilitate testing and debugging of designs on a board into a target system, the 80C186XL has a test mode available which allows all pins to be placed in a high impedance state. ONCE stands for "ON Circuit Emulation". When placed in this mode, the 80C186XL will put all pins in the high impedance state until RESET.

The ONCE mode is selected by tying the UCS and the LCS LOW during RESET. These pins are sampled on the rising high transition of the RES pin. The UCS and the LCS pins have weak internal pull-up resistors similar to the RD and TEST/BUSY pins to guarantee ONCE Mode is not entered inadvertently during normal operation. LCS and UCS must be held low at least one clock after RES goes high to guarantee entrance into ONCE Mode.

PACKAGE INFORMATION

This section describes the pin functions, pinout and thermal characteristics for the 80C186XL in the Quad Flat Pack (QFP), Plastic Leaded Chip Carrier (PLCC), Leadless Chip Carrier (LCC) and the Shrink Quad Flat Pack (SQFP). For complete package specifications and information, see the Intel Packaging Outlines and Dimensions Guide (Order Number 231369).

Pin Descriptions

Each pin or logical set of pins is described in Table 3. There are four columns for each entry in the Pin Description Table. The following sections describe each column.

Column 1: Pin Name

In this column is a mnemonic that describes the pin function. Negation of the signal name (i.e., RES=0) implies that the signal is active-low.

Column 2: Pin Type

A pin may be either power (P), ground (G), input only (I), output only (O) or input/output (I/O). Please note that some pins have more than one function.

Column 3: Input Type (for I and I/O types only)

There are two different types of input pins on the 80C186XL: asynchronous and synchronous. **Asynchronous** pins require that setup and hold times be met only to *guarantee recognition*. **Synchronous** input pins require that the setup and hold times be met to *guarantee proper operation*. Stated simply, missing a setup or hold on an asynchronous pin will result in something minor (i.e., a timer count will be missed) whereas missing a setup or hold on a synchronous pin result in system failure (the system will "lock up").

An input pin may also be edge or level sensitive.

Column 4: Output States (for O and I/O types only)

The state of an output or I/O pin is dependent on the operating mode of the device. There are four modes of operation that are different from normal active mode: Bus Hold, Reset, Idle Mode, Powerdown Mode. This column describes the output pin state in each of these modes.

The legend for interpreting the information in the Pin Descriptions is shown in Table 2.

As an example, please refer to the table entry for AD7:0. The "I/O" signifies that the pins are bidirectional (i.e., have both an input and output function). The "S" indicates that as an input the signal must be synchronized to CLKOUT for proper operation. The "HiZ" indicates that these pins will float while the processor is in the Hold Acknowledge state. R(Z) indicates that these pins will float while RESIN is low.

All pins float while the processor is in the ONCE Mode (with the exception of X2).

Table 2. Pin Description Nomenclature

Symbol	Description
P	Power Pin (apply + V_{CC} voltage)
G	Ground (connect to V_{SS})
I	Input only pin
O	Output only pin
I/O	Input/Output pin
S(E)	Synchronous, edge sensitive
S(L)	Synchronous, level sensitive
A(E)	Asynchronous, edge sensitive
A(L)	Asynchronous, level sensitive
H(1)	Output driven to V_{CC} during bus hold
H(0)	Output driven to V_{SS} during bus hold
H(Z)	Output floats during bus hold
H(Q)	Output remains active during bus hold
H(X)	Output retains current state during bus hold
R(WH)	Output weakly held at V_{CC} during reset
R(1)	Output driven to V_{CC} during reset
R(0)	Output driven to V_{SS} during reset
R(Z)	Output floats during reset
R(Q)	Output remains active during reset
R(X)	Output retains current state during reset

Table 3. Pin Descriptions

Pin Name	Pin Type	Input Type	Output States	Pin Description
V_{CC}	P			System Power. +5 volt power supply.
V_{SS}	G			System ground.
RESET	O		H(0) R(1)	RESET Output indicates that the CPU is being reset, and can be used as a system reset. It is active HIGH, synchronized with the processor clock, and lasts an integer number of clock periods corresponding to the length of the RES signal. Hexadecimal shows that two 2 clockout periods after RES goes inactive. When tied to the TEST/BUSY pin, RESET forces the processor into enhanced mode. RESET is not floated during bus hold.
X1	I	A(L)		Crystal Input. X1 and X2 provide external connections for a fundamental mode crystal used with these parallel resonant crystal for the internal oscillator. X1 can connect to an external clock instead of a crystal. In this case, minimize the capacitance on X2. The input or oscillator frequency is internally divided by two to generate the clock signal (CLKOUT).
X2	O		H(Q) R(Q)	
CLKOUT	O		H(Q) R(Q)	Clock Output provides the system with a 50% duty cycle waveform. All device pin timings are specified relative to CLKOUT. CLKOUT is active during reset and bus hold.
RES	I	A(E)		An active RES causes the processor to immediately terminate its present bus activity, clear the internal logic, and enter a dormant state. This signal may be asynchronous to the clock. The processor begins fetching instructions approximately 6½ clock cycles after RES is returned HIGH. For proper initialization, V_{CC} must be within specifications and the clock signal must be stable for more than 4 clocks with RES held LOW. RES is internally synchronized. This input is provided with a Schmitt-trigger to facilitate power-on RES generation via an RC network.
TEST/BUSY (TEST)	I	A(E)		The TEST pin is sampled during and after reset to determine whether the processor is to enter Compatible or Enhanced Mode. Enhanced Mode requires TEST to be HIGH on the rising edge of RES and LOW four CLKOUT cycles later. Any other combination will place the processor in Compatible Mode. During power up, active RES is required to configure TEST/BUSY as an input. A weak internal pullup ensures a HIGH state when the input is not externally driven. TEST—In Compatible Mode this pin is configured to operate as TEST. This pin is examined by the WAIT instruction. If the TEST input is HIGH when WAIT execution begins, instruction execution will suspend. TEST will be resampled every five clocks until it goes LOW, at which time execution will resume. If interrupts are enabled while the processor is waiting for TEST, interrupts will be serviced. BUSY (80C186XL Only)—In Enhanced Mode, this pin is configured to operate as BUSY. The BUSY input is used to notify the 80C186XL of Math Coprocessor activity. Floating point instructions executing in the 80C186XL sample the BUSY pin to determine when the Math Coprocessor is ready to accept a new command. BUSY is active HIGH.

NOTE:
Pin names in parentheses apply to the 80C188XL.

Table 3. Pin Descriptions (Continued)

Pin Name	Pin Type	Input Type	Output States	Pin Description
TMR IN 0 TMR IN 1	I	A(L) A(E)		Timer inputs are used either as clock or control signals, depending upon the programmed timer mode. These inputs are active HIGH (or LOW to HIGH transitions are counted) and internally synchronized. Timer inputs must be tied HIGH when not being used as clock or retrigger inputs.
TMR OUT 0 TMR OUT 1	O		H(Q) R(1)	Timer outputs are used to provide single pulse or continuous waveform generation, depending upon the timer mode selected. These outputs are not floated during a bus hold.
DRQ0 DRQ1	I	A(L)		DMA Request is asserted HIGH by an external device when it is ready for DMA Channel 0 or 1 to perform a transfer. These signals are level-triggered and internally synchronized.
NMI	I	A(E)		The Non-Maskable Interrupt input causes a Type 2 interrupt. An NMI transition from LOW to HIGH is latched and synchronized internally, and initiates the interrupt at the next instruction boundary. NMI must be asserted for at least one CLKOUT period. The Non-Maskable Interrupt cannot be avoided by programming.
INT0 INTA1/SELECT	I	A(E) A(L)		Maskable Interrupt Requests can be requested by activating one of these pins. When configured as inputs, these pins are active HIGH. Interrupt Requests are synchronized internally. INT2 and INT3 may be configured to provide active-LOW interrupt-acknowledge output signals. All interrupt inputs may be configured to be either edge- or level-triggered. To ensure recognition, all interrupt requests must remain active until the interrupt is acknowledged. When Slave Mode is selected, the function of these pins changes (see Interrupt Controller section of this data sheet).
INT2/INTA0 INT3/INTA1/IRQ	I/O	A(E) A(L)	H(1) R(Z)	
A19/S6 A18/S5 A17/S4 A16/S3 (A9 - A15)	O		H(Q) R(Z)	Address Bus Outputs and Bus Cycle Status (3 - 6) indicate the four most significant address bits during T_1. These signals are active HIGH. During T_2, T_3, T_W and T_4, the S6 pin is LOW to indicate a CPU-initiated bus cycle or HIGH to indicate a DMA-initiated or refresh bus cycle. During the same T-states, S3, S4 and S5 are always LOW. On the 80C188XL, A15 - A8 provide valid address information for the entire bus cycle.
AD0 - AD15 (AD0 - AD7)	I/O	S(L)	H(Z) R(Z)	Address/Data Bus signals constitute the time multiplexed memory or I/O address (T_1) and data (T_2, T_3, T_W and T_4) bus. The bus is active HIGH. For the 80C186XL, A_0 is analogous to $\overline{BHE}$ for the lower byte of the data bus, pins D_7 through D_0. It is LOW during T_1 when a byte is to be transferred onto the lower portion of the bus in memory or I/O operations.

NOTE:
Pin names in parentheses apply only to the 80C188XL.

Table 3. Pin Descriptions (Continued)

Pin Name	Pin Type	Input Type	Output States	Pin Description
BHE (RFSH)	O		H(Z) R(Z)	The $\overline{BHE}$ (Bus High Enable) signal is analogous to A0 in that it is used to enable data on to the most significant half of the data bus, pins D15 - D8. $\overline{BHE}$ will be LOW during T_1 when the upper byte is transferred and will remain LOW through T_2 and T_W. $\overline{BHE}$ does not need to be latched. On the 80C188XL, RFSH is asserted LOW to indicate a refresh bus cycle. In Enhanced Mode, $\overline{BHE}$ (RFSH) will also be used to signify DRAM refresh cycles. A refresh cycle is indicated by both $\overline{BHE}$ (RFSH) and A0 being HIGH.
ALE/QS0	O		H(Q) R(Z)	Address Latch Enable/Queue Status 0 is provided by the processor to latch the address. ALE is active HIGH, with addresses guaranteed valid on the trailing edge.
WR/QS1	O		H(Z) R(Z)	Write Strobe/Queue Status 1 indicates that the data on the bus is to be written into a memory or an I/O device. It is active LOW. When the processor is in Queue Status Mode, the ALE/QS0 and $\overline{WR}$/QS1 pins provide information about processor/instruction queue interaction.
RD/QSMD	O		H(Z) R(1)	Read Strobe is an active LOW signal which indicates that the processor is performing a memory or I/O read cycle. It is guaranteed not to go LOW before the A/D bus is floated. An internal pull up ensures that RD/QSMD is HIGH during RESET. Following RESET the pin is sampled to determine whether the processor is to provide ALE, RD, and $\overline{WR}$, or queue status information. To enable Queue Status Mode, RD must be connected to GND.
ARDY	I	A(L) S(L)		An asynchronous Ready informs the processor that the addressed memory space or I/O device will complete a data transfer. The ARDY pin accepts a rising edge that is asynchronous to CLKOUT and is active HIGH. The falling edge of ARDY must be synchronized to the processor clock. Connecting ARDY HIGH will always assert the ready condition to the CPU. If this line is unused, it should be tied LOW to yield control to the SRDY pin.

QS1	QS0	Queue Operation
0	0	No queue operation
0	1	First opcode byte fetched from the queue
1	1	Subsequent byte fetched from the queue
1	0	Empty the queue

80C186XL BHE and A0 Encodings

BHE Value	A0 Value	Function
0	0	Word Transfer
0	1	Byte Transfer on upper half of data bus (D15 - D8)
1	0	Byte Transfer on lower half of data bus (D_7 - D_0)
1	1	Refresh

NOTE:
Pin names in parentheses apply only to the 80C188XL.

Table 3. Pin Descriptions (Continued)

Pin Name	Pin Type	Input Type	Output States	Pin Description
SRDY	I	S(L)	--	Synchronous Ready informs the processor that the addressed memory space or I/O device will complete a data transfer. The SRDY pin accepts an active-HIGH input synchronized to CLKOUT. The use of SRDY allows a relaxed system timing over ARDY. This is accomplished by elimination of the one-half clock cycle required to internally synchronize the ARDY input signal. Connecting SRDY high will always assert the ready condition to the CPU. If this feature is unused, it should be tied LOW to yield control to the ARDY pin.
LOCK	O	--	H(Z) R(Z)	LOCK output indicates that other system bus masters are not to gain control of the system bus. LOCK is active LOW. The LOCK signal is requested by the LOCK prefix instruction and is activated at the beginning of the first data cycle associated with the instruction immediately following the LOCK prefix. It remains active until the completion of that instruction. No instruction prefetching will occur while LOCK is asserted.
S0 S1 S2	O	--	H(Z) R(1)	Bus cycle status S0 - S2 are encoded to provide bus-transaction information. **Bus Cycle Status Information** S2 may be used as a logical M/IO indicator, and S1 as a DT/R indicator.

S2	S1	S0	Bus Cycle Initiated
0	0	0	Interrupt Acknowledge
0	0	1	Read I/O
0	1	0	Write I/O
0	1	1	Halt
1	0	0	Instruction Fetch
1	0	1	Read Data from Memory
1	1	0	Write Data to Memory
1	1	1	Passive (no bus cycle)

HOLD	I	A(L)	--	HOLD indicates that another bus master is requesting the local bus. The HOLD input is active HIGH. The processor generates HLDA (HIGH) in response to a HOLD request. Simultaneous with the issuance of HLDA, the processor will float the local bus and control lines. After HOLD is detected as being LOW, the processor will lower HLDA. When the processor needs to run another bus cycle, it will again drive the local bus and control lines. In Enhanced Mode, HLDA will go low after the DRAM refresh cycle in progress (if any) and before the next refresh cycle is initiated. This allows the bus to be relinquished to the refresh controller logic by first lowering HOLD so that the processor will regain control after the refresh cycle.
HLDA	O	--	H(1) R(0)	

NOTE:
Pin names in parentheses apply to the 80C188XL.

Table 3. Pin Descriptions (Continued)

Pin Name	Pin Type	Input Type	Output States	Pin Description
UCS	I/O	A(L)	H(1) R(WH)	Upper Memory Chip Select is an active LOW output whenever a memory reference is made to the defined upper portion (1K–256K block) of memory. The address range activating UCS is software programmable. UCS and LCS are sampled upon the rising edge of RES. If both pins are held low, the processor will enter ONCE Mode. In ONCE Mode all pins assume a high impedance state and remain so until a subsequent RESET. UCS has a weak internal pullup that is active during RESET to ensure that the processor does not enter ONCE Mode inadvertently.
LCS	I/O	A(L)	H(1) R(WH)	Lower Memory Chip Select is active LOW whenever a memory reference is made to the defined lower portion (1K–256K) of memory. The address range activating LCS is software programmable. UCS and LCS are sampled upon the rising edge of RES. If both pins are held low, the processor will enter ONCE Mode. In ONCE Mode all pins assume a high impedance state and remain so until a subsequent RESET. LCS has a weak internal pullup that is active only during RESET to ensure that the processor does not enter ONCE mode inadvertently.
MCS0/PEREQ MCS1/ERROR	I/O	A(L)	H(1) R(WH)	Mid-range Memory Chip Select signals are active LOW when a memory reference is made to the defined mid-range portion of memory (8K–512K). The address ranges activating MCS0–3 are software programmable. On the 80C186XL, in Enhanced Mode, MCS0 becomes a PEREQ input (Processor Extension Request). When connected to the Math Coprocessor, this input is used to signal the 80C186XL when to make numeric data transfers to and from the coprocessor. MCS1 becomes ERROR in Enhanced Mode and is used to signal numerics coprocessor errors.
MCS2 MCS3/NPS	O		H(1) R(1)	
PCS0 PCS1 PCS2 PCS3 PCS4	O		H(1) R(1)	Peripheral Chip Select signals 0–4 are active LOW when a reference is made to the defined peripheral area (64 Kbyte I/O or 1 MByte memory space). The address ranges activating PCS0–4 are software programmable.
PCS5/A1	O		H(1)/R(A) R(1)	Peripheral Chip Select 5 or Latched A1 may be programmed to provide a sixth peripheral chip select, or to provide an internally latched A1 signal. The address range activating PCS5 is software programmable. PCS5/A1 does not float during bus HOLD. When programmed to provide latched A1, this pin will retain the previously latched value during HOLD.

NOTE:
Pin names in parentheses apply to the 80C188XL.

Table 3. Pin Descriptions (Continued)

Pin Name	Pin Type	Input Type	Output States	Pin Description
PCS6/A2	O	–	H(1)/H(0) R(1)	Peripheral Chip Select 6 or Latched A2 may be programmed to provide a seventh peripheral chip select, or to provide an internally latched A2 signal. The address range activating PCS6 is software programmable. PCS6/A2 does not float during bus HOLD. When programmed to provide latched A2, this pin will retain the previously latched value during HOLD.
DT/R	O	–	H(Z) R(Z)	Data Transmit/Receive controls the direction of data flow through an external data bus transceiver. When LOW, data is transferred to the processor. When HIGH the processor places write data on the data bus.
DEN	O	–	H(Z) R(1)	Data Enable is provided as a data bus transceiver output enable. DEN is active LOW during each memory and I/O access (including 80C187 accesses). DEN is HIGH whenever DT/R changes state. During RESET, DEN is driven HIGH for one clock, then floated.
N.C.	–	–	–	Not connected. To maintain compatibility with future products, do not connect to these pins.

NOTE:
Pin names in parentheses apply to the 80C188XL.

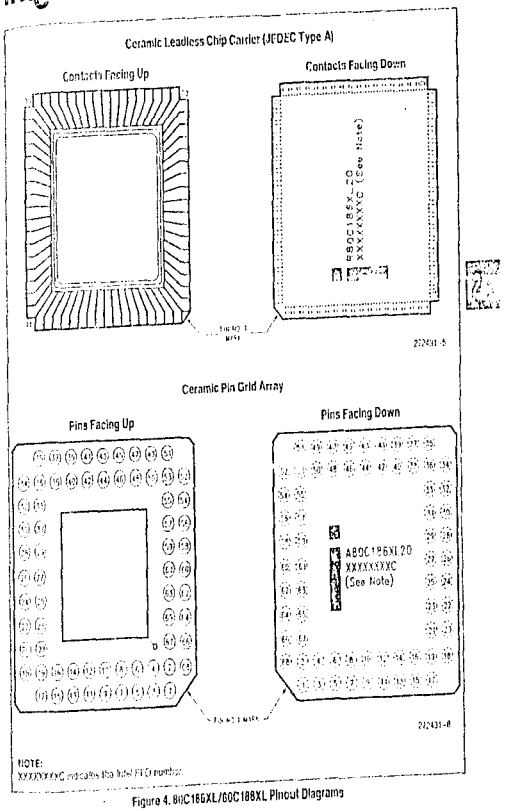


Figure 4. 80C186XL/80C188XL Pinout Diagrams

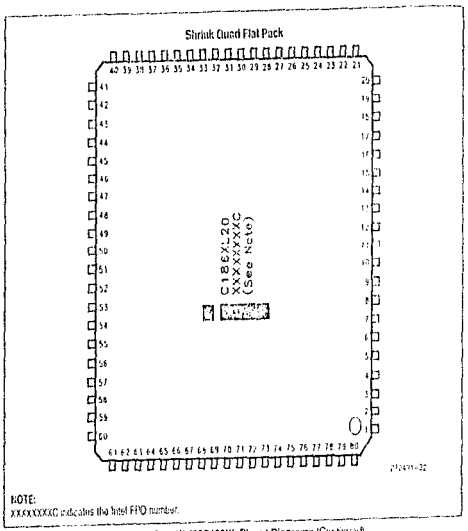


Figure 4. 80C186XL/80C188XL Pinout Diagrams (Continued)

80C186XL/80C188XL

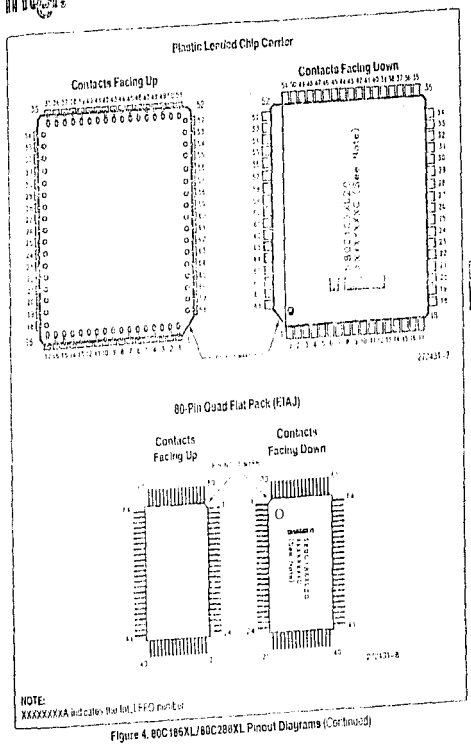


Figure 4. 80C186XL/80C188XL Pinout Diagrams (Continued)

Table 4. LCC/PLCC Pin Functions with Location

AD Bus		Bus Control		Processor Control		I/O	
AD0	17	ALE/QS0	61	RES	24	UCS	34
AD1	15	BHE (RFSH)	64	RESET	57	LCS	35
AD2	13	S0	52	X1	59		
AD3	11	S1	53	X2	58	MCS0/PEREQ	39
AD4	8	S2	54	CLKOUT	56	MCS1/ERROR	37
AD5	6	RD/QSMD	62	TEST/BUSY	47	MCS2	36
AD6	4	WR/QS1	63	NMI	46	MCS3/INPS	35
AD7	2	ARDY	55	INT0	45		
AD8 (A8)	16	SRDY	49	INT1/SELECT	44		
AD9 (A9)	14	DEN	50	INT2/INTA0	42	PCS0	25
AD10 (A10)	12	DT/R	48	INT3/INTA1	41	PCS1	27
AD11 (A11)	10	LOCK	60			PCS2	28
AD12 (A12)	7	HOLD	65	Power and Ground		PCS3	29
AD13 (A13)	5	HLDA	66			PCS4	30
AD14 (A14)	3			V_{CC}	9	PCS5/A1	31
AD15 (A15)	1			V_{CC}	43	PCS6/A2	32
A16/S3	68			V_{CC}	26		
A17/S4	67			V_{SS}	69	TMR IN 0	20
A18/S5	66					TMR IN 1	21
A19/S6	65					TMR OUT 0	22
						TMR OUT 1	23
						DRQ0	18
						DRQ1	19

NOTE:
Pin names in parentheses apply to the 80C188XL.

Table 5. LCC/PGA/PLCC Pin Locations with Pin Names

1	AD15 (A15)	18	DRQ0	35	MCS3/INPS	52	S0
2	AD7	19	DRQ1	36	MCS2	53	S1
3	AD14 (A14)	20	TMR IN 0	37	MCS1/ERROR	54	S2
4	AD6	21	TMR IN 1	38	MCS0/PEREQ	55	ARDY
5	AD13 (A13)	22	TMR OUT 0	39	DEN	56	CLKOUT
6	AD5	23	TMR OUT 1	40	DT/R	57	RESET
7	AD12 (A12)	24	RES	41	INT3/INTA1	58	X2
8	AD4	25	PCS0	42	INT2/INTA0	59	X1
9	V_{CC}	26	V_{CC}	43	V_{CC}	60	V_{SS}
10	AD11 (A11)	27	PCS1	44	INT1/SELECT	61	ALE/QS0
11	AD3	28	PCS2	45	INT0	62	RD/QSMD
12	AD10 (A10)	29	PCS3	46	NMI	63	WR/QS1
13	AD2	30	PCS4	47	TEST/BUSY	64	BHE (RFSH)
14	AD9 (A9)	31	PCS5/A1	48	LOCK	65	A19/S6
15	AD1	32	PCS6/A2	49	SRDY	66	A18/S5
16	AD8 (A8)	33	V_{SS}	50	HOLD	67	A17/S4
17	AD0	34	UCS	51	HLDA	68	A16/S3

NOTE:
Pin names in parentheses apply to the 80C188XL.

Table 6. QFP Pin Functions with Location

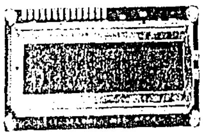
AD Bus		Bus Control		Processor Control		I/O	
AD0	64	ALE/QS0	10	RES	55	UCS	45
AD1	66	BHE (RFSH)	7	RESET	14	LCS	46
AD2	68	S0	73	X1	16		
AD3	70	S1	72	X2	17	MCS0/PEREQ	39
AD4	74	S2	71	CLKOUT	19	MCS1/ERROR	40
AD5	76	RD/QSMD	9	TEST/BUSY	31	MCS2	41
AD6	78	WR/QS1	8	NMI	30	MCS3/INPS	42
AD7	80	ARDY	24	INT0	34		
AD8 (A8)	65	SRDY	27	INT1/SELECT	32		
AD9 (A9)	67	DEN	28	INT2/INTA0	33	PCS0	51
AD10 (A10)	69	DT/R	37	INT3/INTA1	35	PCS1	52
AD11 (A11)	71	LOCK	29			PCS2	53
AD12 (A12)	75	HOLD	26	Power and Ground		PCS3	54
AD13 (A13)	77	HLDA	25			PCS4	56
AD14 (A14)	79			V_{CC}	11	PCS5/A1	57
AD15 (A15)	1			V_{CC}	36	PCS6/A2	58
A16/S3	3			V_{CC}	48		
A17/S4	4			V_{CC}	63	TMR IN 0	59
A18/S5	5			V_{SS}	12	TMR IN 1	60
A19/S6	6			V_{SS}	13	TMR OUT 0	61
				V_{SS}	62	TMR OUT 1	62
		No Connection					
		N.C.	2			DRQ0	63
		N.C.	21			DRQ1	64
		N.C.	44				
		N.C.	45				
		N.C.	46				
		N.C.	61				

NOTE:
Pin names in parentheses apply to the 80C188XL.

Table 7. QFP Pin Locations with Pin Names

1	AD15 (A15)	21	S2	41	MCS2	61	DRQ0
2	N.C.	22	S1	42	MCS3/INPS	62	N.C.
3	A16/S3	23	S0	43	N.C.	63	N.C.
4	A17/S4	24	N.C.	44	N.C.	64	AD0
5	A18/S5	25	HLDA	45	UCS	65	AD8 (A8)
6	A19/S6	26	HOLD	46	LCS	66	AD1
7	BHE (RFSH)	27	SRDY	47	PCS6/A2	67	AD9 (A9)
8	WR/QS1	28	LOCK	48	PCS5/A1	68	AD2
9	RD/QSMD	29	TEST/BUSY	49	PCS4	69	AD10 (A10)
10	ALE/QS0	30	NMI	50	PCS3	70	AD3
11	N.C.	31	INT0	51	PCS2	71	AD11 (A11)
12	V_{SS}	32	INT1/SELECT	52	PCS1	72	V_{SS}
13	V_{SS}	33	V_{CC}	53	V_{CC}	73	V_{CC}
14	N.C.	34	V_{SS}	54	PCS0	74	AD4
15	N.C.	35	INT2/INTA0	55	RES	75	AD12 (A12)
16	X1	36	INT3/INTA1	56	TMR OUT 1	76	AD5
17	X2	37	DT/R	57	TMR OUT 0	77	AD13 (A13)
18	RESET	38	DEN	58	TMR IN 1	78	AD6
19	CLKOUT	39	MCS0/PEREQ	59	TMR IN 0	79	AD14 (A14)
20	V_{CC}	40	MCS1/ERROR	60	DRQ1	80	AD7

NOTE:
For details of the pin functions, refer to the 80C186XL

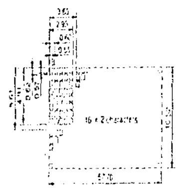


[illegible]

FEATURES

- Compact [illegible] display module
- High contrast [illegible] display with large characters
- Low voltage +5V single power supply
- Wide operating temperature range (0°C to +50°C)
- 5 x 7 dot character format and cursor line
- Built-in control LSI with display RAM and character generator ROM
- Direct interface to 4 or 8 bit CPU
- 11 commands for control
- Transflective module is available as an option

Dot Matrix Dimensions (in millimeters)



Mechanical Characteristics

Item	Specification	Unit
Outline Dimension	67W x 36H x 12.0	mm
Character Size	2.95W x 4.2H	mm
Number of Characters	16 x 2 (32) Characters (5 x 7 dots & cursor)	—
Viewing Area	64.5W x 15H	mm
Bezel Opening	64.5W x 15H	mm
Dot Size	0.55W x 0.57H	mm
Dot Pitch	0.60W x 0.62H	mm
Weight	approx. 30	gram

Absolute Maximum Ratings

Item	Symbol	Rating	Unit
Supply voltage	V_{DD}	7	V
Input voltage	V_{IN}	[illegible]	V
Operating Temperature	T_{OP}	[illegible]	°C
Storage Temperature	T_{STG}	[illegible]	°C

Electrical Characteristics (T_a = 25°C)

Item	Symbol	Min	Typ	Max	Unit
Supply voltage	V_{DD}	4.75	5.0	5.25	V
	GND	—	0	—	V
[illegible] (V_{DD} = 5.0V)	V_{IH}	2.2	—	—	V
[illegible] (I_{OL} = 0.2mA)	V_{IL}	—	—	0.6	V
[illegible] (I_{OL} = 1.2mA)	V_{OL}	—	—	0.4	V
Power consumption	P_D	—	10	—	mW

Optical Characteristics (T_a = 25°C, [illegible])

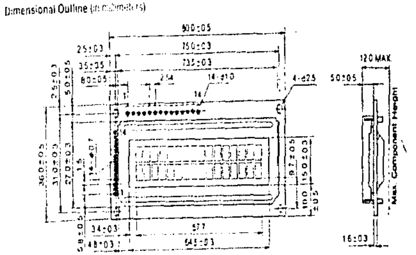
Item	Symbol	Min.	Typ.	Max	Unit
Viewing Angle	ϕ	10	25	40	degree
Contrast	K	—	3.0	—	—
Turn On Time	t_{on}	—	200	400	ms
Turn Off Time	t_{off}	—	150	400	ms

Note: [illegible]

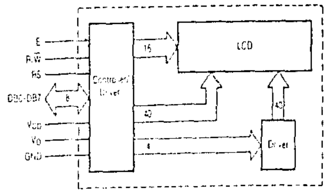
Connector Pin Assignment

Pin No.	Signal	Function
1	GND	0V — Power Supply
2	V_{DD}	5V — Power Supply
3	V_O	LCD Drive Voltage [illegible]
4	RS	H: Data Input; L: Command Input
5	R/W	H: Data Read; L: Data Write
6	E	Enable Signal
7	DB0	Data Bus (8 bit)
8	DB1	
9	DB2	
10	DB3	
11	DB4	Data Bus (4 bit)
12	DB5	
13	DB6	
14	DB7	

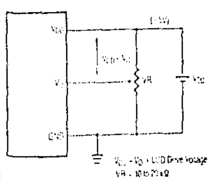
Dimensional Outline (in millimeters)



Block Diagram



Power Supply



Recommended Power Supply for LCD Drive (V_O)

LCD Panel is driven by the voltage V_{DD} − V_O, so adjustable V_O is required for contrast control and temperature compensation.

Temperature	V_O
0°C	[illegible]
+25°C	[illegible]
+50°C	[illegible]

Relation between Character Position and Character Address

AND241, AND391, AND501

Character Position	[illegible]
DD RAM (HEX) Address	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F 20 21 22 23 24 25 26 2
Module Type	AND241 / AND391 / AND591

AND671

Character Position	[illegible]
DD RAM (HEX) Address	00 01 02 03 04 05 06 07 40 41 42 43 44 45 46 47

AND491

Character Position	[illegible]
DD RAM (HEX) Address	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
Character Position	[illegible]
DD RAM (HEX) Address	40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F

AND501

Character Position	[illegible]
DD RAM (HEX) Address	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 11 12 13
Character Position	[illegible]
DD RAM (HEX) Address	40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F 50 51 52 53

AND591

Character Position	[illegible]
DD RAM (HEX) Address	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F 20 21 22 23 24 25 26 27
Character Position	[illegible]
DD RAM (HEX) Address	40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F 50 51 52 53 54 55 56 57 58 59 5A 5B 5C 5D 5E 5F 60 61 62 63 64 65 66 67

Relation between Character Position and Character Address (Continued)

AND771

Character Position	[illegible]
DD RAM (HEX) Address	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 16 17
Character Position	[illegible]
DD RAM (HEX) Address	40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F 50 51 52 53 54 55 56 57

AND731

Character Position	[illegible]
DD RAM (HEX) Address	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
Character Position	[illegible]
DD RAM (HEX) Address	40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F
Character Position	[illegible]
DD RAM (HEX) Address	10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F
Character Position	[illegible]
DD RAM (HEX) Address	50 51 52 53 54 55 56 57 58 59 5A 5B 5C 5D 5E 5F

AND721

Character Position	[illegible]
DD RAM (HEX) Address	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 11 12 13
Character Position	[illegible]
DD RAM (HEX) Address	40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F 50 51 52 53
Character Position	[illegible]
DD RAM (HEX) Address	14 15 16 17 18 19 1A 1B 1C 1D 1E 1F 20 21 22 23 24 25 26 27
Character Position	[illegible]
DD RAM (HEX) Address	54 55 56 57 58 59 5A 5B 5C 5D 5E 5F 60 61 62 63 64 65 66 67

Timing Characteristics (T_a = 25°C)

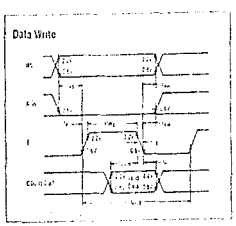
Data Write

Item	Symbol	Value		Unit
		Min	Max	
Enable Cycle Time	t_{cycE}	1000		
Enable Pulse Width	t_{PW}	450		
Enable Rise/Fall Time	$t_{r,}$ t_f		25	ns
Set Up Time	t_{su}	140		
Address Hold Time	t_{HD}	10		
Data Set Up Time	t_{DSW}	195		
Data Hold Time	t_H	10		

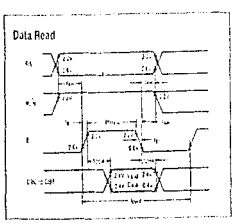
Data Read

Item	Symbol	Value		Unit
		Min	Max	
Enable Cycle Time	t_{cycE}	1000		
Enable Pulse Width	t_{PW}	450		
Enable Rise/Fall Time	$t_{r,}$ t_f		25	ns
Set Up Time	t_{su}	140		
Address Hold Time	t_{HD}	10		
Data Delay Time	t_{DDR}		320	
Data Hold Time	t_H	20		

Data Write



Data Read

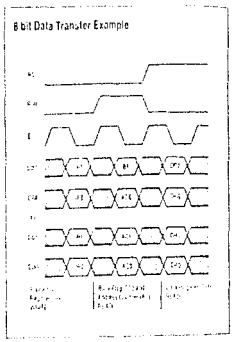


Data Transfer Example

The data can be sent in the form of two 2-cycles or 4-bit data or 1 cycle of 8-bit data so that it can be connected to both 4- and 8-bit CPUs.

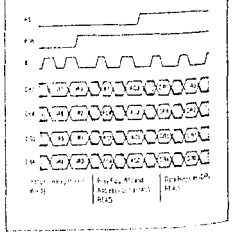
(1) When data is 8-bit long, the data is transferred using only the 8 lines of DB0 to DB7.

8 bit Data Transfer Example



(2) When data is 4-bit long, it is transferred by using only 4 lines of DB4 to DB7, DB3 to DB0 are not used. Data transfer between the MPU and LCM is completed when 4-bit data is transferred twice. The order of data transfer is 4 high order bits (bit 7 to bit 4) first, then 4 low order bits (bit 3 to bit 0).

4 bit Data Transfer Example



Function of Registers

1. Instruction Register and Data Register

The LCD module is built-in connected has two 8-bit registers, an instruction Register (IR) and a Data Register (DR). IR stores instruction codes, such as display clear and cursor shift, and address information for display data RAM (DD RAM) and character generator RAM (CG RAM).

Shuttle between by a CPU to DR or DR to IR.

The DR temporarily stores data to be written into or read from DD RAM or the CG RAM. Data written into the DR is automatically sent to the DD RAM or the CG RAM as an internal operation. The DR is also used for data storage when reading data from the DD RAM or the CG RAM. When RS=H is written into the IR, data is transferred to the DR from the DD RAM or the CG RAM as an internal operation. When the CPU reads the DR, data in the DD RAM or the CG RAM addressed by IR is sent to DR for the next reading.

Register Selector (RS) signals affect these functions.

Table 1 Register Selection

RS	R/W	Operation
0	0	IR write as an internal operation (display clear, etc.)
0	1	Read busy flag (DB7) and address counter (DB0 to DB6)
1	0	DR write as an internal operation (DR → DD or CG RAM)
1	1	DR read as an internal operation (DD or CG RAM → DR)

2. Busy Flag (BF)

When the Busy Flag is "1", the LCD module is in the internal operation mode, and the next instruction is not accepted. It at the time. As shown in Table 1, the Busy Flag is shown in DB7 when RS = 0 and R/W = 1. The next instruction must be written after checking that the Busy Flag is "0".

3. Address Counter (AC)

The address counter (AC) assigns DD and CG RAM address. When an instruction for address setting is written in IR, the address information is sent from IR to AC.

Selection of either the DD or CG RAM is also determined by an instruction. After writing into (reading from) DD or CG RAM display data, AC is automatically incremented by +1 (or decremented by −1). Data in address counters (AC) are in DB0 to DB6 when RS = 0 and R/W = 1, as shown in Table 1.

4. Display Data RAM (DD RAM)

The display data RAM (DD RAM) stores display data represented in 8-bit character codes.

Relationship between the DD RAM address and display position on LCD Display is shown on page 210 and 211.

Commands

The command code refers to the signal that, when the LCD module is accessed through the CPU, the LCD module begins operation upon receipt of the code input.

As the internal processing operation of the LCD module is started with a timing that does not affect the LCD display, the busy status continues longer than the CPU cycle time.

When the Busy status (when the busy flag is set to "1"), the LCD module does not receive any commands other than the busy flag read.

For this reason, the CPU has to verify that the busy flag is set to "0" prior to the input of the command code.

Table 2 shows the commands and the execution times for the commands. The commands can be divided into the following 4 types.

(a) Commands that designate the module functions such as display format, data length, etc.
(b) Commands that give internal RAM addresses
(c) Commands that perform data transfer with internal RAM
(d) Other commands

Table 2 Command List

Command	Command Code RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	Description	Execution Time (max) (note 1)	Execution Time (max) (note 2)
Clear Display	0	0	0	0	0	0	0	0	0	1	Clears entire display and sets DD RAM address 0 in address counter.	1.64 ms	40 ms
Return Home	0	0	0	0	0	0	0	0	1	*	Sets DD RAM address 0 in address counter. Also returns display from being shifted to original position. DD RAM contents remain unchanged.	1.64 ms	40 ms
Entry Mode Set	0	0	0	0	0	0	0	1	I/D	S	Sets cursor move direction and specifies display shift. These operations are performed during data write and read.	40 µs	1.6 ms
Display ON/OFF Control	0	0	0	0	0	0	1	D	C	B	Sets entire display (D) on/off, cursor on/off (C), and blinking of cursor position character (B).	40 µs	1.6 ms
Cursor and Display Shift	0	0	0	0	0	1	S/C	R/L	*	*	Moves cursor and shifts the display without changing DD RAM contents.	40 µs	1.6 ms
Function Set	0	0	0	0	1	DL	N	F	*	*	Sets interface data length (DL), number of display lines (N), and character font (F).	40 µs	1.6 ms
Set CG RAM Address	0	0	0	1	ACG						Sets CG RAM address in address counter.	40 µs	1.6 ms
Set DD RAM Address	0	0	1	ADD							Sets DD RAM address in address counter.	40 µs	1.6 ms
Read Busy Flag & Address	0	1	BF	AC							Reads Busy flag (BF) indicating internal operation is being performed and reads address counter contents.	0 µs	1.6 ms
Write Data to CG or DD RAM	1	0	Write Data								Writes data into DD RAM or CG RAM.	40 µs	1.6 ms
Read Data from CG or DD RAM	1	1	Read Data								Reads data from DD RAM or CG RAM.	40 µs	1.6 ms

I/D = 1: Increment
I/D = 0: Decrement
S = 1: Accompanies display shift
S/C = 1: Display shift
S/C = 0: Cursor move
R/L = 1: Shift to the right
R/L = 0: Shift to the left
DL = 1: 8 bits
DL = 0: 4 bits
N = 1: 2 lines
N = 0: 1 line
F = 1: 5 x 10 dots
F = 0: 5 x 7 dots
BF = 1: Internally operating
BF = 0: Can accept instruction

X = Don't care

Notes:
1 Applies to AMC(X)1, AMC(X)3, AMC(X)5, AMC(X)6, AMC(X)7
2 Applies to AMC(X)2, AMC(X)21, AMC(X)31, AMC(X)71
3 Instruction Clause and Data Set time varies by the AMC(X)21
4 The typical time per execution (Clear Display) is calculated at 270 kHz (typ.) (1.5 µs x 1 clock) and Max. time (fosc = 250 kHz).

1. Clear Display

	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Code	0	0	0	0	0	0	0	0	0	1

Write to clear code "20" (blank character) into the DD RAM addresses. The cursor returns to address 0 (DD RAM Address "00H") and display, if it has been shifted, returns to its original position. In other words, display disappears and the cursor goes to the left side of the display.

2. Return Home

	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Code	0	0	0	0	0	0	0	0	1	*

Returns the cursor to character position 1 (DD RAM Address = 00H) and returns the display to the original position if it has been shifted. (S of the entry mode register is 1). The DD RAM contents remain unchanged.

3. Entry Mode Set

	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Code	0	0	0	0	0	0	0	1	I/D	S

I/D: Increment (I/D = 1) or decrement (I/D = 0) the DD RAM address by one when writing a character code into the DD RAM or reading a character code from the DD RAM. The cursor moves to the right when I/D = 1, and to the left when I/D = 0.

S: Shifts the entire display to the right (when I/D = 0) or left (when I/D = 1). S = 0 when writing to the DD RAM. Even when the cursor looks as if it does not shift, display shift operation does occur. This is not shift when reading from the DD RAM. Display is not shifted when S = 0.

4. Display ON/OFF Control

	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Code	0	0	0	0	0	0	1	D	C	B

D: Display is turned ON when D = 1 and OFF when D = 0. When display is turned OFF, the display data remains in the DD RAM and may can be displayed immediately by setting D = 1.

C: The cursor is displayed when C = 1 and not displayed when C = 0. Even if the cursor disappears, function of I/D, etc. does not change during display data write. The cursor is displayed at the 8th line when the 5 x 7 dots character font is selected.

B: The character at the cursor position blinks when B = 1. The blink is done by switching between all black dots and display characters at 0.4 second interval. The cursor and the blink can be set concurrently.

X = Don't care

5. Cursor or Display Shift

	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Code	0	0	0	0	0	1	S/C	R/L	X	X

Shift the cursor position or display position to the right or the left without writing or reading the display data. This function can be used for correction or search of display.

S/C	R/L	Function
0	0	Shift the cursor position to the left (AC is decremented by one)
0	1	Shift the cursor position to the right (AC is incremented by one)
1	0	Shift the entire display to the left. The cursor follows the display shift.
1	1	Shift the entire display to the right. The cursor follows the display shift.

6. Function Set

	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Code	0	0	0	0	1	DL	N	F	X	X

DL: Sets the interface data length. Data is sent or received in 8-bit length (DB7 to DB0) when DL = 1 and 4-bit length (DB7 to DB4) when DL = 0. When 4-bit length is selected, data must be sent or received twice.

N: Sets number of display lines.

F: Sets character font. The 5 x 7 dots character font is selected when F = 0, while 5 x 10 dots character font is selected when F = 1 and N = 0.

N	F	No. of Display Lines	Character Font	Duty Ratio	Module Type No.
0	0	1	5x7 Dots	1/8	AMC(X)01, AMC(X)1, AMC(X)11
0	1	1	5x10 Dots	1/11	AMC(X)21, AMC(X)31, AMC(X)91
1	0	2	5x7 Dots	1/16	AMC(X)71, AMC(X)81, AMC(X)91
1	0	4	5x7 Dots	1/16	AMC(X)51, AMC(X)52

7. Set CG RAM Address

	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Code	0	0	0	1	A	A	A	A	A	A

Sets the CG RAM address to a binary number of AAAAAA in the address counter. After execution of this instruction, all the data from MPU is written into the CG RAM and all the data is read from CG RAM.

8. Set DD RAM Address

	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Code	0	0	1	A	A	A	A	A	A	A

Set the DD RAM address (a binary number of AAAAAAA in the address counter (An - 0 to the left one, An - 1 to the left second). The execution of this instruction, after the data from MPU is written into the DD RAM and sets the data sheet from DD RAM.

Read Busy Flag and Address

	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Code	0	1	BF	A	A	A	A	A	A	A

Read Busy Flag (BF) and the value of the address counter (AAAAAAA). BF = 1 indicates that an internal operation is going on and next command is not accepted until BF becomes 0. Therefore, necessary to check the BF status before the next write operation. The address counter is used for the CG or DD RAM address.

10. Write Data to CG RAM or DD RAM

	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Code	1	0	D	D	D	D	D	D	D	D

Write binary 8-bit data DDDDDDDD to the CG RAM or the DD RAM. Whether the CG RAM or the DD RAM is to be written is determined by the previous designation (CG RAM address setting or DD RAM address setting). After writing, the address is automatically incremented or decremented by one according to entry mode. Display shift also follows the entry mode.

11. Read Data from CG RAM or DD RAM

	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Code	1	1	D	D	D	D	D	D	D	D

Read binary 8-bit data DDDDDDDD from the CG RAM or the DD RAM. Whether the CG RAM or the DD RAM is to be read is determined by the previous designation. Prior to inputting this read command, either the CG RAM address set command or the DD RAM address set command must be executed. If it is not done, the first data read is invalid, and the second data read of the next address can be read normally. After reading, the address is automatically incremented or decremented by one according to the entry mode. However, display shift is not performed regardless of entry mode.

Character Patterns and Character Codes

1. Character Generator ROM (CG ROM)

The character generator ROM generates 5 x 7 dot or 5 x 10 dot character patterns from an 8-bit character code signal.

When the 8-bit character code signal is applied to the CG ROM, as mentioned previously, the character pattern of the CG ROM corresponding to the code is displayed on the LCD display. Refer to correspondence of the CG ROM address. Table 3 shows the relation between character patterns and character codes.

Note

ANDEN, AND101, AND102, AND103, AND104, AND105 and AND201 currently... 5 x 7 dot character patterns.

2. Character Generator RAM (CG RAM)

The character generator RAM is used for designing character patterns other than for the CG ROM. The CG RAM has the capacity of 64 bytes. 8/16 bytes are for 8 types of character patterns with 5 x 7 dots and 4 types with 5 x 10 dots. When displaying character patterns stored in the CG RAM, write 8-bit character codes (000 to 007 or 00H to 06H) on the left side as shown in Table 3.

Table 4 shows the relation between CG RAM addresses and data and display patterns for 5 x 7 dots.

Table 5 shows the relation between CG RAM addresses and data and display patterns for 5 x 10 dots.

Table 3. Character Pattern and Character Code

Upper 4 bit / Lower 4 bit	0000	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
XXXX0000	(1)														
XXXX0001	(2)														
XXXX0010	(3)														
XXXX0011	(4)														
XXXX0100	(5)														
XXXX0101	(6)														
XXXX0110	(7)														
XXXX0111	(8)														
XXXX1000	(1)														
XXXX1001	(2)														
XXXX1010	(3)														
XXXX1011	(4)														
XXXX1100	(5)														
XXXX1101	(6)														
XXXX1110	(7)														
XXXX1111	(8)														

BURR-BROWN®

ADC7802

Autocalibrating, 4-Channel, 12-Bit ANALOG-TO-DIGITAL CONVERTER

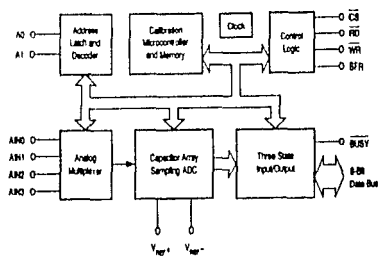
FEATURES

- TOTAL UNADJUSTED ERROR ≤1/2LSB OVER FULL TEMPERATURE RANGE
- FOUR-CHANNEL INPUT MULTIPLEXER
- LOW POWER: 10mW plus Power Down Mode
- SINGLE SUPPLY: +5V
- FAST CONVERSION TIME: 8.5μs Including Acquisition
- AUTOCAL: No Offset or Gain Adjust Required
- UNIPOLAR INPUTS: 0V to 5V
- MICROPROCESSOR-COMPATIBLE INTERFACE
- INTERNAL SAMPLE/HOLD

DESCRIPTION

The ADC7802 is a monolithic CMOS 12-bit A/D converter with internal sample/hold and four-channel multiplexer. An autocalibration cycle, occurring automatically at power on, guarantees a total unadjusted error within ±1/2LSB over the specified temperature range, eliminating the need for offset or gain adjustment. The 5V single supply requirements and standard $\overline{CS}$, $\overline{RD}$, and $\overline{WR}$ control signals make the part very easy to use in microprocessor applications. Conversion results are available in two bytes through an 8-bit three-state output bus.

The ADC7802 is available in a 28-pin plastic DIP and 28-lead PLCC, fully specified for operation over the industrial –40°C to +85°C temperature range.



International Airport Industrial Park • Mailing Address: PO Box 11400 • Tucson, AZ 85734 • Street Address: 6730 S. Tucson Blvd. • Tucson, AZ 85706
Tel: (602) 746-1111 • Twx: 910-952-1111 • Cable: BBRCORP • Telex: 066-6491 • FAX: (602) 889-1510 • Immediate Product Info: (800) 548-6132

PDS-1070A

SPECIFICATIONS

ELECTRICAL

$V_S = V_A = V_{REF+} = 5V$ ±5%; V_B ± $V_{REF-} = V_{REF-} = AGND = DGND = 0V$, CLK = 2MHz internal with 50% duty cycle, $T_A = -40°C$ to +85°C, after calibration cycle at any temperature, unless otherwise specified.

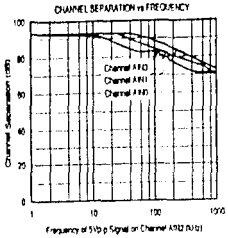
PARAMETER	CONDITIONS	ADC7802BP/ADC7802BH MIN	TYP	MAX	UNITS
RESOLUTION				12	Bits
ANALOG INPUT					
Voltage Input Range	$V_{IN+} = 5V$, $V_{IN-} = 0V$	0		5	V
Input Capacitance			50		pF
On State Bias Current			100		nA
Off State Bias Current	$T_A = 25°C$			10	nA
	$T_A = -40°C$ to +85°C			100	nA
On Resistance Multiplexer			8		kΩ
Off Resistance Multiplexer			10		MΩ
Channel Separation	500Hz		92		dB
REFERENCE INPUT					
For Specified Performance V_{REF+}	$V_{REF} = 5V$		5		V
V_{REF-}			0		V
For Derated Performance V_{REF+}		2.5		V_A	V
V_{REF-}		0		1	V
Input Reference Current	$V_{REF+} = 5V$, $V_{REF-} = 0V$		10	100	µA
THROUGHPUT TIMING					
Conversion Time With External Clock (including Multiplexer Settling Time and Acquisition Time)	CLK = 2MHz, 50% Duty Cycle			8.5	µs
	CLK = 1MHz, 50% Duty Cycle			17	µs
	CLK = 500kHz, 50% Duty Cycle			34	µs
With Internal Clock Using Recommended Clock Components	$T_A = +25°C$			10	µs
	$T_A = -40°C$ to +85°C		10		µs
Analog Signal Bandwidth [4]			500		kHz
Slew Rate [4]		8			mV/µs
Multiplexer Settling Time to 0.01%			400		ns
Multiplexer Access Time				20	ns
ACCURACY					
Total Adjusted Error [5] All Channels				±1/2	LSB
Differential Nonlinearity				±1/2	LSB
No Missing Codes			Guaranteed		
Gain Error	All Channels			±1/4	LSB
Gain Error Drift	Between Calibration Cycles		±0.2		ppm/°C
Offset Error	All Channels			±1/4	LSB
Offset Error Drift	Between Calibration Cycles		±0.8		ppm/°C
Channel to Channel Mismatch				±1/4	LSB
Power Supply Sensitivity	$V_S = V_A = 4.75V$ to 5.25V			±1/8	LSB
DIGITAL INPUTS					
All Pins Other Than CLK V_{IL}				0.8	V
V_{IH}		2.4			V
Input Current	$T_A = +25°C$, $V_{IN} = 0$ to V_S			1	µA
	$T_A = -40°C$ to +85°C, $V_{IN} = 0$ to V_S			10	µA
CLK Input V_{IL}				0.8	V
V_{IH}		3.5			V
I_{IL}				10	µA
I_{IH}				1.6	mA
I_{IH}	Power Down Mode (D3 in SFR is HIGH)			100	nA
DIGITAL OUTPUTS					
V_{OL}	$I_{SINK} = 1.6mA$			0.4	V
V_{OH}	$I_{SOURCE} = 200µA$	4			V
Leakage Current	High Z State, $V_{OUT} = 0V$ to V_S			±1	µA
Output Capacitance	High Z State			15	pF
POWER SUPPLIES					
Supply Voltage for Specified Performance V_S		4.75	5	5.25	V
V_A	$V_A \geq V_S$	4.75	5	5.25	V
Supply Current I_S			1	2.5	mA
I_A	Logic Input Pins HIGH or LOW		1	2	mA
Power Dissipation	$\overline{WR} = \overline{RD} = \overline{CS} = \overline{BUSY} = HIGH$		10		mW
Power Down Mode	See Table III Page 9		50		µW
TEMPERATURE RANGE					
Specification		-40		+85	°C
Storage		-65		+150	°C

NOTES: (1) For $(V_{REF+}) - (V_{REF-})$ as low as 2.5V, the total error will typically not exceed 1LSB. (2) Faster signals can be accurately converted by using an external sample/hold in front of the ADC7802. (3) After calibration cycle, without external adjustment. Includes gain (full scale) error, offset error, integral nonlinearity, differential nonlinearity, and drift.

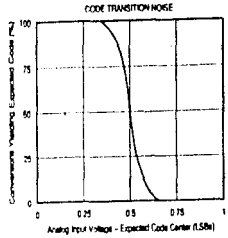
TYPICAL PERFORMANCE CURVES

$V_S = V_A = V_{REF+} = 5V$, $V_{REF-} = AGND = 0V$, $T_A = +25°C$ unless otherwise specified.

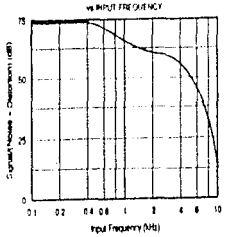
CHANNEL SEPARATION vs FREQUENCY



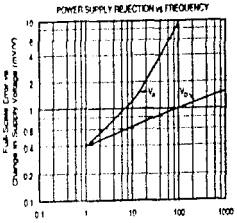
CODE TRANSITION NOISE



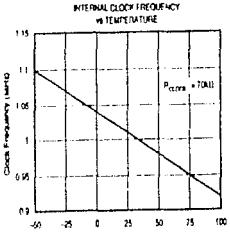
(SIGNAL/NOISE + DISTORTION) vs INPUT FREQUENCY



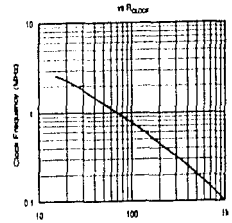
POWER SUPPLY REJECTION vs FREQUENCY



INTERNAL CLOCK FREQUENCY vs TEMPERATURE



INTERNAL CLOCK FREQUENCY vs R_{CLOCK}

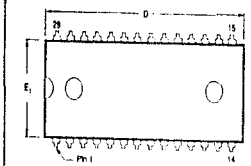


ORDERING INFORMATION

MODEL	MAXIMUM TOTAL ERROR, LSB	SPECIFICATION TEMPERATURE RANGE, °C	PACKAGE
ADC7802BH	±1/2	-40 to +85	PLCC
ADC7802BP	±1/2	-40 to +85	Plastic DIP

MECHANICAL

P Package — 28-Pin Plastic DIP

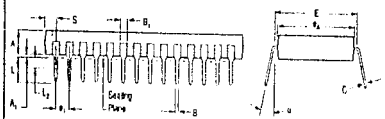


DIM	INCHES MIN	INCHES MAX	MILLIMETERS MIN	MILLIMETERS MAX
A	.125	.200	3.20	5.08
A_1	.015	.070	0.38	1.78
B	.015	.022	0.38	0.51
B_1	.015	.060	0.38	1.40
C	.008	.012	0.20	0.30
D	1.380	1.445	35.05	36.68
E	.600	.625	15.24	15.88
E_1	.485	.550	12.32	13.97
e	.100 BASIC		2.54 BASIC	
e_A	.600 BASIC		15.24 BASIC	

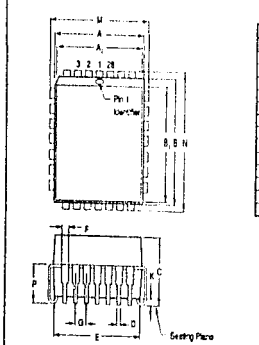
DIM	INCHES MIN	INCHES MAX	MILLIMETERS MIN	MILLIMETERS MAX
L	.100	.200	2.54	5.08
L_1	.000	.020	0.00	0.76
α	0°	15°	0°	15°
Q	.060	.070	0.81	1.78
S_1	.040	.080	1.02	2.27

(1) Per JEDEC Standard

NOTE: Leads in true position within 0.01" (0.25mm) R at MMC at seating plane. Pin numbers shown for reference only. Numbers may not be marked on package.



N Package — 28-Pin Plastic LCC



DIM	INCHES MIN	INCHES MAX	MILLIMETERS MIN	MILLIMETERS MAX
A	0.450	0.495	11.43	11.68
A_1	0.450	0.480	11.43	11.68
B	0.450	0.480	11.43	11.68
B_1	0.450	0.480	11.43	11.68
C	0.165	0.180	4.19	4.57
D	0.013	0.023	0.33	0.58
E	0.390	0.430	9.91	10.92
F	0.009	0.032	0.86	0.81
G	0.50 BASIC		1.27 BASIC	
K	0.015	0.025	0.38	0.64
M	0.485	0.495	12.32	12.57
N	0.485	0.495	12.32	12.57
P	0.100	0.110	2.54	2.79

NOTE: Leads in true position within 0.01" (0.25mm) R at MMC at seating plane. Pin numbers shown for reference only. Numbers may not be marked on package.

ABSOLUTE MAXIMUM RATINGS

V_S to Analog Ground	8.5V
V_S to Digital Ground	8.5V
Pin V_L to Pin V_S	+0.3V
Analog Ground to Digital Ground	±1V
Control Inputs to Digital Ground	-0.3V to V_L +0.3V
Analog Input Voltage to Analog Ground	-0.3V to V_S +0.3V
Maximum Junction Temperature	+150°C
Internal Power Dissipation	875mW
Lead Temperature (soldering, 10s)	+300°C
Thermal Resistance, θ_{JA} Plastic DIP	75°C/W
PLCC	75°C/W

THEORY OF OPERATION

ADC7802 uses the advantages of advanced CMOS technology (logic density, stable capacitors, precision analog switches, and low power consumption) to provide a precise 12-bit analog-to-digital converter with on-chip sampling and four-channel analog-input multiplexer.

The input stage consists of an analog multiplexer with an address latch to select from four input channels.

The converter stage consists of an advanced successive approximation architecture using charge redistribution on a capacitor network to digitize the input signal. A temperature-stabilized differential auto-zeroing circuit is used to minimize offset errors in the comparator. This allows offset errors to be corrected during the acquisition phase of each conversion cycle.

Linearity errors in the binary weighted main capacitor network are corrected using a capacitor trim network and correction factors stored in on-chip memory. The correction terms are calculated by a microcontroller during a calibration cycle, initiated either by power up or by applying an external calibration signal at any time. During conversion, the correct trim capacitors are switched into the main capacitor array as needed to correct the conversion accuracy. This is faster than a complex digital error correction system, which could slow down the throughput rate. With all of the capacitors in both the main array and the trim array on the same chip, excellent stability is achieved, both over temperature and over time.

For flexibility, timing circuits include both an internal clock generator and an input for an external clock to synchronize with external systems. Standard control signals and three-state input/output registers simplify interfacing ADC7802 to most micro-controllers, microprocessors or digital storage systems.

Finally, this performance is matched with the low power advantages of CMOS structures to allow a typical power consumption of 30mW.

OPERATION

BASIC OPERATION

Figure 1 shows the simple circuit required to operate ADC7802 in the Transparent Mode, converting a single input channel. A convert command on pin 20 ($\overline{WR}$) starts a conversion. Pin 22 ($\overline{BUSY}$) will output a LOW during the

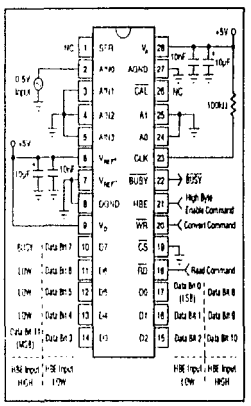
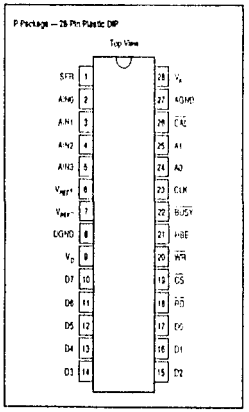


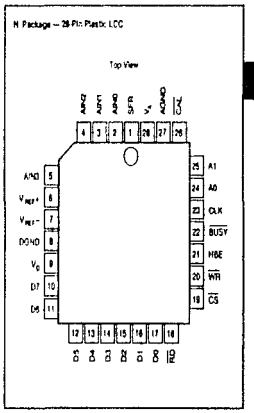
FIGURE 1. Basic Operation.

PIN CONFIGURATIONS

P Package — 28-Pin Plastic DIP



N Package — 28-Pin Plastic LCC



conversion process (including sample acquisition and conversion), and rises only after the conversion is completed. The two bytes of output data can then be read using pin 18 ($\overline{RD}$) and pin 21 (HBE).

STARTING A CONVERSION

A conversion is initiated on the rising edge of the $\overline{WR}$ input, with valid signals on A0, A1 and $\overline{CS}$. The selected input channel is sampled for five clock cycles, during which the comparator offset is also auto-zeroed to below 1/4LSB of error. The successive approximation conversion takes place during clock cycles 6 through 17.

Figures 2 and 3 show the full conversion sequence and the timing to initiate a conversion.

CALIBRATION

A calibration cycle is initiated automatically upon power up (or after a power failure). Calibration can also be initiated by the user at any time by the rising edge of a minimum 100ns-wide LOW pulse on the $\overline{CAL}$ pin (pin 26), or by setting D1 HIGH in the Special Function Register (see SFR section). A calibration command will initiate a calibration cycle, regardless of whether a conversion is in process. During a calibration cycle, convert commands are ignored.

Calibration takes 168 clock cycles, and a normal conversion (17 clock cycles) is added automatically. For maximum accuracy, the supplies and reference need to be stable during the calibration procedure. To ensure that supply voltages and reference voltages have settled and are stable, an internal timer provides a waiting period of 42,425 clock cycles between power up/power failure and the start of the calibration cycle.

READING DATA

Data from the ADC7802 is read in two 8 bit bytes, with the Low byte containing the 8 LSBs of data, and the High byte containing the 4 MSBs of data. The outputs are coded in straight binary (with 0V = 000 hex, 5V = FFF hex), and the data is presented in a right-justified format (with the LSB as the most right bit in the 16 bit word). Two read operations are required to transfer the High byte and Low byte, and the bytes are presented according to the input level on the High Byte Enable pin (HBE).

The bytes can be read in either order, depending on the status of the HBE input. If HBE changes while $\overline{CS}$ and $\overline{RD}$ are LOW, the output data will change to correspond to the HBE input. Figure 4 shows the timing for reading first the Low byte and then the High byte.

ADC7802 provides two modes for reading the conversion results. At power up, the converter is set in the Transparent Mode.

TRANSPARENT MODE

This is the default mode for ADC7802. In this mode, the conversion decisions from the successive approximation register are latched into the output register as they are made. Thus, the High byte (the 4 MSBs) can be read after the end of the ninth clock cycle (five clock cycles for the mux setting, sample acquisition and auto-zeroing of the comparator, followed by the four clock cycles for the 4MSB deci-

PIN ASSIGNMENTS

PIN #	NAME	DESCRIPTION
1	SFR	Special Function Register. When connected to a microprocessor address pin, allows access to special functions 8 through D0 to D7. See the section "By using the Special Function Register". If not used, connect to DGND. This pin has an internal pull down.
2 to 5	AIN0 to AIN3	Analog Inputs. Channel 0 to channel 3.
6	$V_{REF}+$	Positive voltage reference input. Normally +5V. Must be $\leq V_A$.
7	$V_{REF}-$	Negative voltage reference input. Normally 0V.
8	DGND	Digital ground. DGND = 0V.
9	V_A	Logic supply voltage. V_A = +5V. Must be $\leq V_A$ and applied after V_A.
10 to 17	D0 to D7	Data Bus Input/Output Pins. Normally used to read output data. See section on SFR (Special Function Register) for other uses. When SFR is LOW, these function as follows: 10 D7 Data Bit 7 if HBE is LOW. If HBE is HIGH, acts as converter status pin and is HIGH during conversion or calibration, goes LOW after the conversion is completed. (Acts as an inverted $\overline{BUSY}$.) 11 D6 Data Bit 6 if HBE is LOW. LOW if HBE is HIGH. 12 D5 Data Bit 5 if HBE is LOW. LOW if HBE is HIGH. 13 D4 Data Bit 4 if HBE is LOW. LOW if HBE is HIGH. 14 D3 Data Bit 3 if HBE is LOW. Data Bit 11 (MSB) if HBE is HIGH. 15 D2 Data Bit 2 if HBE is LOW. Data Bit 10 if HBE is HIGH. 16 D1 Data Bit 1 if HBE is LOW. Data Bit 9 if HBE is HIGH. 17 D0 Data Bit 0 (LSB) if HBE is LOW. Data Bit 8 if HBE is HIGH.
18	$\overline{RD}$	Read Input. Active LOW, used to read the data outputs in combination with $\overline{CS}$ and HBE.
19	$\overline{CS}$	Chip Select Input. Active LOW.
20	$\overline{WR}$	Write Input. Active LOW, used to start a new conversion and to select an analog channel via address inputs A0 and A1 in combination with $\overline{CS}$. The minimum $\overline{WR}$ pulse LOW is 100ns.
21	HBE	High Byte Enable. Used to select high or low data output byte in combination with $\overline{CS}$ and $\overline{RD}$, or to select SFR.
22	$\overline{BUSY}$	$\overline{BUSY}$ is LOW during conversion or calibration. $\overline{BUSY}$ goes HIGH after the conversion is completed.
23	CLK	Clock input. For internal-external clock operation: For external clock operation, connect pin 23 to a 74 HC compatible clock source. For internal clock operation, connect pin 23 per the clock operation description.
24 to 25	A0 to A1	Address inputs. Used to select one of four analog input channels in combination with $\overline{CS}$ and $\overline{WR}$. The address inputs are latched on the rising edge of $\overline{WR}$ or $\overline{CS}$. A1 A0 Selected Channel LOW LOW AIN0 LOW HIGH AIN1 HIGH LOW AIN2 HIGH HIGH AIN3
26	$\overline{CAL}$	Calibration Input. A calibration cycle is initiated when $\overline{CAL}$ is LOW. The minimum pulse width of $\overline{CAL}$ is 100ns. If not used connect to V_A. In this case calibration is only initiated at power on, or with SFR. This pin has an internal pull up.
27	AGND	Analog Ground. AGND = 0V.
28	V_A	Analog Supply. V_A = +5V. Must be $\geq V_A$ and $V_{REF}+$.

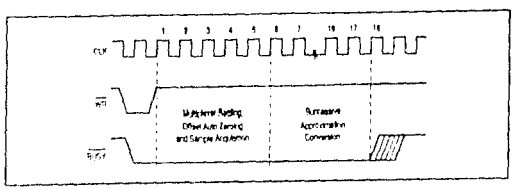


FIGURE 2. Converter Timing.

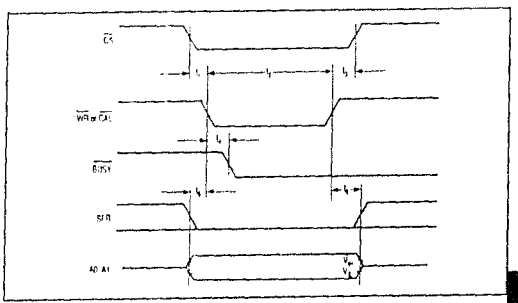


FIGURE 3. Write Cycle Timing (for initiating conversion or calibration).

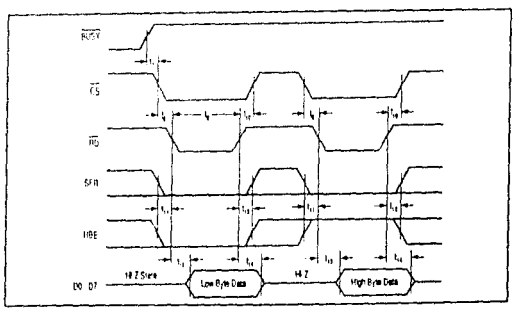


FIGURE 4. Read Cycle Timing.

sions.) The complete 12 bit data is available after $\overline{BUSY}$ has gone HIGH, or the internal status flag goes LOW (D7 when HBE is HIGH).

LATCHED OUTPUT MODE

This mode is activated by writing a HIGH to D0 and LOWs to D1 to D7 in the Special Function Register with $\overline{CS}$ and $\overline{WR}$ LOW and SFR and HBE HIGH. (See the discussion of the Special Function Register below.)

In this mode, the data from a conversion is latched into the output buffers only after a conversion is complete, and remains there until the next conversion is completed. The conversion result is valid during the next conversion. This allows the data to be read even after a new conversion is started, for faster system throughput.

TIMING CONSIDERATIONS

Table I and Figures 3 through 8 show the digital timing of ADC7802 under the various operating modes. All of the critical parameters are guaranteed over the full -40°C to +85°C operating range for ease of system design.

SPECIAL FUNCTION REGISTER (SFR)

An internal register is available, either to determine additional data concerning the ADC7802, or to write additional instructions to the converter. Access to the Special Function Register is made by driving SFR HIGH.

Table II shows the data in the Special Function Register that will be transferred to the output bus by driving HBE HIGH (with SFR HIGH) and initiating a read cycle (driving $\overline{RD}$ and $\overline{CS}$ LOW with $\overline{WR}$ HIGH as shown in Figure 4.) The Power Fail flag in the SFR is set when the power supply falls below about 3V. The flag also means that a new calibration has been started, and any data written to the SFR has been lost. Thus, the ADC7802 will again be in the Transparent Mode. Writing a LOW to D5 in the SFR resets the Power Fail flag. The Cal Error flag in the SFR is set when an overflow occurs during calibration, which may happen in very noisy systems. It is reset by starting a calibration, and remains low after a calibration without an overflow is completed.

SYMBOL	PARAMETER [1]	MIN	TYP	MAX	UNITS
t_1	$\overline{CS}$ to $\overline{WR}$ Setup Time [2]	0	0	0	ns
t_2	$\overline{WR}$ or $\overline{CAL}$ Pulse Width	100			ns
t_3	$\overline{CS}$ to $\overline{WR}$ Hold Time [2]	0	0	0	ns
t_4	$\overline{WR}$ to $\overline{BUSY}$ Propagation Delay	20	50	150	ns
t_5	A0, A1, HBE, SFR Valid to $\overline{WR}$ Setup Time	0			ns
t_6	A0, A1, HBE, SFR Valid to $\overline{WR}$ Hold Time	20			ns
t_7	$\overline{BUSY}$ to $\overline{CS}$ Setup Time	0			ns
t_8	$\overline{CS}$ to $\overline{RD}$ Setup Time [2]	0	0	0	ns
t_9	$\overline{RD}$ Pulse Width	100			ns
t_{10}	$\overline{CS}$ to $\overline{RD}$ Hold Time [2]	0	0	0	ns
t_{11}	HBE, SFR to $\overline{RD}$ Setup Time	50			ns
t_{12}	HBE, SFR to $\overline{RD}$ Hold Time	0			ns
t_{13}	$\overline{RD}$ to Valid Data (Bus Access Time) [2]		80	150	ns
t_{14}	$\overline{RD}$ to Hi Z Delay (Bus Release Time) [2]		90	180	ns
t_{15}	$\overline{RD}$ to Hi Z Delay for SFR [2]	20		60	ns
t_{16}	Data Valid to $\overline{WR}$ Setup Time	100			ns
t_{17}	Data Valid to $\overline{WR}$ Hold Time	20			ns

NOTES: (1) All input control signals are specified with $t_{rise} = t_{fall} = 20ns$ (10% to 90% of 5V) and timed from a voltage level of 1.6V. Data is timed from V_{IH} or V_{IL}. (2) The internal RD pulse is performed by a NOR writing of $\overline{CS}$ and $\overline{RD}$. The internal WR pulse is performed by a NOR writing of $\overline{CS}$ and $\overline{WR}$ (3) Figures 3 and 8 show the measurement circuits and pulse diagrams for testing transitions to and from Hi Z states.

TABLE I. Timing Specifications (CLK_{IN} = 1MHz external, T_A = -40°C to +85°C).

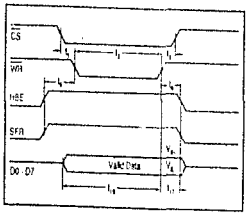


FIGURE 5. Writing to the SFR.

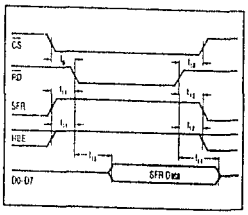


FIGURE 6. Reading the SFR.

Writing a HIGH to D3 in the FSR puts the ADC7802 in the Power Down Mode. Power consumption is reduced to 50µW and D3 remains HIGH. To exit Power Down Mode, either write a LOW to D3 in the SFR, or initiate a calibration by sending a LOW to the $\overline{CAL}$ pin or writing a HIGH to D1. During Power Down Mode, a pulse on $\overline{CS}$ and $\overline{WR}$ will initiate a single conversion, then the ADC7802 will revert to power down.

Table III shows how instructions can be transferred to the Special Function Register by driving HBE HIGH (with SFR HIGH) and initiating a write cycle (driving $\overline{WR}$ and $\overline{CS}$ LOW with $\overline{RD}$ HIGH.) The timing is shown in Figure 3. Note that writing to the SFR also initiates a new conversion.

CONTROL LINES

Table IV shows the functions of the various control lines on the ADC7802. The use of standard $\overline{CS}$, $\overline{RD}$ and $\overline{WR}$ control signals simplifies use with most microprocessors. At the same time, flexibility is assured by availability of status information and control functions, both through the SFR and directly on pins.

INSTALLATION

INPUT BANDWIDTH

From the typical performance curves it is clear that ADC7802 can accurately digitize signals up to 50kHz, but distortion will increase beyond this point. Input signals slewing faster than 8mV/µs can degrade accuracy. This is a result of the high-precision auto-zeroing circuit used during the acquisition phase. For applications requiring higher signal bandwidth, any good external sample/hold, like the SHC5320, can be used.

INPUT IMPEDANCE

ADC7802 has a very high input impedance (input bias current over temperature is 100nA max), and a low 50pF

PIN	FUNCTION	DESCRIPTION
D0	Mode Status	If LOW, Transparent Mode enabled for data latches. If HIGH, Latched Output Mode enabled.
D1	CAL Flag	If HIGH, calibration cycle in progress.
D2		Reserved for factory use.
D3	Power Down Status	If HIGH, in Power Down Mode.
D4		Reserved for factory use.
D5	POWER FAIL Flag	If HIGH, a power supply failure has occurred. (Supply fell below 3V.)
D6	CAL ERROR Flag	If HIGH, an overflow occured during calibration.
D7	BUSY Flag	If HIGH, conversion or calibration in progress.

NOTE: These data are transferred to the bus when a read cycle is initiated with SFR and HBE HIGH. Reading the SFR with SFR HIGH and HBE LOW is reserved for factory use at this time, and will yield unpredictable data.

TABLE II. Reading the Special Function Register.

	CS,WR	SFR/HBE	D0	D1	D3	D5	D7	D2/D4/D6
Enables Transparent Mode for Data Latches	LOW	HIGH	LOW	X	LOW	X	LOW	LOW
Enables Latched Output Mode for Data Latches	LOW	HIGH	HIGH	X	LOW	X	LOW	LOW
Initiates Calibration Cycle	LOW	HIGH	X	HIGH	LOW	X	LOW	LOW
Resets Power Fail flag	LOW	HIGH	X	X	LOW	LOW	LOW	LOW
Activates Power Down Mode	LOW	HIGH	X	X	HIGH	X	LOW	LOW

NOTES: (1) In Power Down Mode, a pulse on $\overline{CS}$ and $\overline{WR}$ will initiate a single conversion, then the ADC7802 will revert to power down. (2) X means it can be either HIGH or LOW without affecting the action. Writing HIGH to D2, D3, D4 or D6, or writing with SFR HIGH and HBE LOW, may result in unpredictable behavior. These modes are reserved for factory use at this time.

TABLE III. Writing to the Special Function Register.

$\overline{CS}$	$\overline{RD}$	$\overline{WR}$	SFR	HBE	$\overline{CAL}$	BUSY	OPERATION
X	X	X	X	X	0↑1	X	Initiates calibration cycle
X	X	X	X	X	X	0	Conversion or calibration in process. Inhibits new conversion from starting
1	X	X	X	X	1	X	None. Outputs in Hi Z State
0	1	0↑1	0	X	1	1	Initiates conversion
0	0	1	0	0	1	X	Low byte conversion results output on data bus
0	0	1	0	1	1	X	High byte conversion results output on data bus
0	1	0	1	1	1	1	Write to SFR and rising edge on $\overline{WR}$ initiates conversion
0	0	1	1	1	1	X	Contents of SFR output on data bus
0	1	0	1	0	1	X	Reserved for factory use.
0	0	1	1	0	1	X	Reserved for factory use. (Unpredictable data on data bus.)

TABLE IV. Control Line Functions.

input capacitance. To ensure a conversion accurate to 12 bits, the analog source must be able to charge the 50pF and settle within the first five clock cycles after a conversion is initiated. During this time, the input is also very sensitive to noise at the analog input, since it could be injected into the capacitor array.

In many applications, a simple passive low-pass filter as shown in Figure 9a can be used to improve signal quality. In this case, the source impedance needs to be less than 5kΩ to keep the induced offset errors below 1/2LSB, and to meet the acquisition time of five clock cycles. The values in Figure 9a meet these requirements, and will maintain the full power bandwidth of the system. For higher source impedances, a buffer like the one in Figure 9b should be used.

INPUT PROTECTION

The input signal range must not exceed $\pm V_{REF}$ or V_A by more than 0.3V.

The analog inputs are internally clamped to V_A. To prevent damage to the ADC7802, the current that can flow into the inputs must be limited to 20mA. One approach is to use an external resistor in series with the input filter resistor. For example, a 1kΩ input resistor allows an overvoltage to 20V without damage.

REFERENCE INPUTS

A 10µF tantalum capacitor is recommended between $V_{REF}+$ and $V_{REF}-$ to insure low source impedance. These capacitors should be located as close as possible to the ADC7802 to reduce dynamic error, since the reference provides packets of current as the successive approximation steps are carried out.

$V_{REF}+$ must not exceed V_A. Although the accuracy is specified with $V_{REF}+ = 5V$ and $V_{REF}- = 0V$, the converter can function with $V_{REF}+$ as low as 2.5V and $V_{REF}-$ as high as 1V. As long as there is at least a 2.5V difference between $V_{REF}+$ and $V_{REF}-$, the absolute value of errors does not change significantly, so that accuracy will typically be within ±1LSB. (1/2LSB for a 5V span is 610µV, which is 1LSB for a 2.5V span.)

The power supply to the reference source needs to be considered during system design to prevent $V_{REF}+$ from exceeding (or overshooting) V_A, particularly at power-on. Also, after power-on, if the reference is not stable within 42,425 clock cycles, an additional calibration cycle may be needed.

POWER SUPPLIES

The digital and analog power supply lines to the ADC7802 should be bypassed with 10µF tantalum capacitors as close to the part as possible. Although ADC7802 has excellent power supply rejection, even for higher frequencies, linear regulated power supplies are recommended.

Care should be taken to insure that V_D does not come up before V_A, or permanent damage to the part may occur. Figure 10 shows a good supply approach, powering both V_A and V_D from a clean linear supply, with the 10Ω resistor between V_A and V_D insuring that V_D comes up after V_A. This is also a good method to further isolate the ADC7802 from digital supplies in a system with significant switching currents that could degrade the accuracy of conversion.

GROUNDING

To maximize accuracy of the ADC7802, the analog and digital grounds are not connected internally. These points should have very low impedance to avoid digital noise feeding back into the analog ground. The $V_{REF}-$ pin is used as the reference point for input signals, so it should be connected directly to AGND to reduce potential noise problems.

EXTERNAL CLOCK OPERATION

The circuitry required to drive the ADC7802 clock from an external source is shown in Figure 11a. The external clock must provide a 0.8V max for LOW and a 3.5V min for HIGH, with rise and fall times that do not exceed 200ns. The minimum pulse width of the external clock must be 300ns. Synchronizing the conversion clock to an external system clock is recommended in microprocessor applications to prevent beat-frequency problems.

Note that the electrical specification tables are based on using an external 2MHz clock. Typically, the specified accuracy is maintained for clock frequencies between 0.5 and 2.4MHz.

INTERNAL CLOCK OPERATION

Figure 11b shows how to use the internal clock generating circuitry. The clock frequency depends only on the value of the resistor, as shown in "Internal Clock Frequency vs R_{CLOCK}" in the Typical Performance Curves section.

The clock generator can operate between 100kHz and 2MHz. With R = 100kΩ, the clock frequency will nominally be 800kHz. The internal clock oscillation may vary by up to 20% from device to device, and will vary with temperature, as shown in the typical performance curves. Therefore, use of an external clock source is preferred in many applications where control of the conversion timing is critical, or where multiple converters need to be synchronized.

APPLICATIONS

BIPOLAR INPUT RANGES

Figure 12 shows a circuit to accurately and simply convert a bipolar ±5V input signal into a unipolar 0 to 5V signal for conversion by the ADC7802, using a precision, low-cost complete difference amplifier, INA105.

Figure 13 shows a circuit to convert a bipolar ±10V input signal into a unipolar 0 to 5V signal for conversion by the ADC7802. The precision of this circuit will depend on the matching and tracking of the three resistors used.

To trim this circuit for full 12-bit precision, R2 and R3 need to be adjustable over appropriate ranges. To trim, first have the ADC7802 converting continually and apply +9.9927V (+10V - 1.5LSB) at the input. Adjust R3 until the ADC7802 output toggles between the codes FFE hex and FFF hex. This makes R3 extremely close to R1. Then, apply -9.9976V (-10V + 0.5LSB) at the input, and adjust R2 until the ADC7802

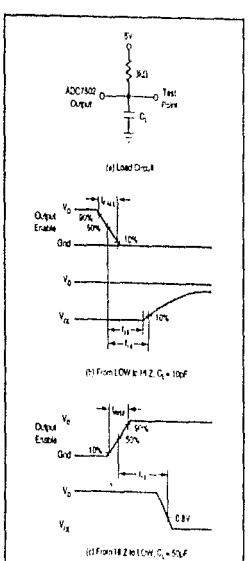


FIGURE 7. Measuring Active LOW to/from Hi-Z State.

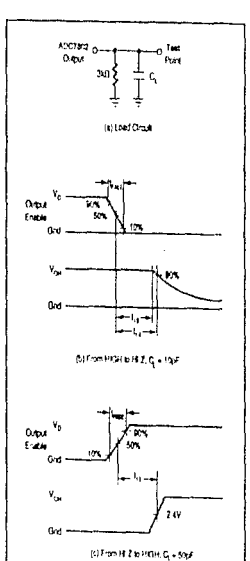


FIGURE 8. Measuring Active HIGH to/from Hi-Z State.

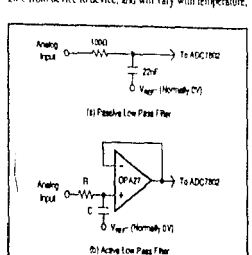


FIGURE 9. Input Signal Conditioning.

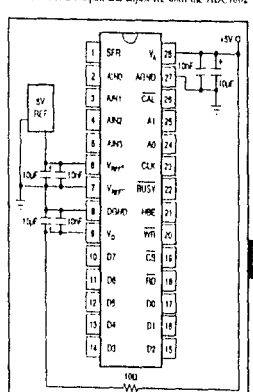


FIGURE 10. Power Supply and Reference Decoupling.

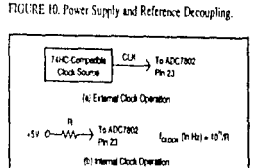


FIGURE 11. Internal Clock Operation.

output toggles between 000 hex and 001 hex. At each trim point, the current through the third resistor will be almost zero, so that one trim iteration will be enough in most cases. More iterations may be required if the op amp selected has large offset voltage or bias currents, or if the +5V reference is not precise.

This circuit can also be used to adjust gain and offset errors due to the components preceding the ADC7802, to match the performance of the self-calibration provided by the converter.

INTERFACING TO MOTOROLA MICROPROCESSORS

Figure 14 shows a typical interface to Motorola microprocessors, while Figure 15 shows how the result can be placed in register D0.

Conversion is initiated by a write instruction decoded by the address decoder logic, with the lower two bits of the address bus selecting an ADC input channel, as follows:

MOVE.W D0, ADC-ADDRESS

The result of the conversion is read from the data bus by a read instruction to ADC-ADDRESS as follows:

MOVEP.W $000 (ADC-ADDRESS), D0

This puts the 12-bit conversion result in the D0 register, as shown in Figure 15. The address decoder must pull down $\overline{\text{ADC\_CS}}$ at ADC-ADDRESS to access the Low byte and ADC-ADDRESS +2 to access the High byte.

INTERFACING TO INTEL MICROPROCESSORS

Figure 16 shows a typical interface to Intel.

A conversion is initiated by a write instruction to address $\overline{\text{ADC\_CS}}$. Data pins D00 and D01 select the analog input channel. The $\overline{\text{BUSY}}$ signal can be used to generate a microprocessor interrupt (INT) when the conversion is completed.

A read instruction from the $\overline{\text{ADC\_CS}}$ address fetches the Low byte, and a read instruction from the $\overline{\text{ADC\_CS}}$ address +2 fetches the High byte.

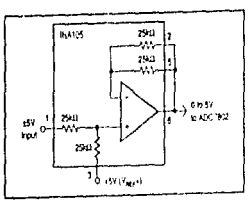


FIGURE 12. ±5V Input Range.

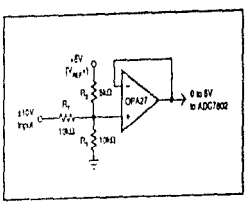


FIGURE 13. ±10V Input Range.

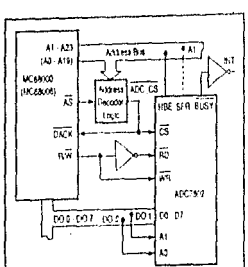


FIGURE 14. Interface to Motorola Microprocessors.

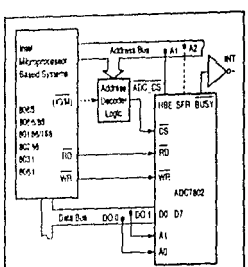


FIGURE 16. Interface to Intel Microprocessors.

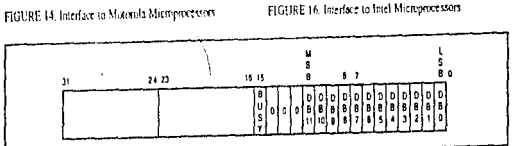


FIGURE 15. Conversion Results in Motorola Register D0.

APENDICE G

```
/* Programa del menu principal
   Simulacion del control de un manipulador de seis GDL
   Tercera version, terminada al 16 de agosto de 1995. Ya con la modificacion
   de las restricciones a nivel de las uniones.   */

#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <string.h>
#include <math.h>
#include <graphics.h>

/* Variables globales */

struct parametros {
      char   nombre[6];
      float  val;
};
struct parametros cinematicos[41];
struct parametros dinamicos[81];
struct parametros cont_union[81];
struct parametros gen_tray[41];
float puntos[10] [10], tiem_sec[10] [14];
int sec[10] [14], cambio=0;

main ()
{
   /* variables locales a main */
   char op[80], sal[80], err[80];
   int opcion, salir;
   /* Declaracion de funciones */
   void mod_param (void);
   void grab_mod_pun (void);
   void esp_sec_mov (void);
   void inic_op (void);
   void inicializa (void);
   void guardar (void);
   int prob_inv(int, float mat[]);

   /* Menu principal */
   inicializa ();
   salir=0;
   while (!salir) {
       opcion=0;
       while ((opcion>5)||(opcion<1)) {
           clrscr ();
           printf("                    MENU PRINCIPAL \n\n\n\n\n");
           printf("          1) MODIFICACION PARAMETROS \n");
           printf("          2) GRABACION/MODIFICACION DE PUNTOS \n");
           printf("          3) ESPECIFICACION SECUENCIA MOVIMIENTOS \n");
           printf("          4) INICIO OPERACION \n");
           printf("          5) SALIR \n\n\n");
           printf("                    OPCION:");
           gets (op);
           opcion = atoi(op);
           if ((opcion>5)||(opcion<1)) {
              printf("\n OPCION EQUIVOCADA !!!");
              printf("\n PRESIONE ENTER PARA CONTINUAR");
              gets(err);
           }
       }
       switch (opcion) {
           case 1:
              mod_param ();
              break;
           case 2:
              grab_mod_pun ();
              break;
```

```
            case 3:
                esp_sec_mov ();
                 break;
            case 4:
                inic_op ();
                break;
            case 5:
                printf("\n SALIR ? [S/N]");
                gets (sal);
                if ((sal[0] == 'S')||(sal[0] == 's')) salir = 1;
        }
    }
    guardar ();
}

/* Funcion modificacion parametros */
/* Primera version  25/enero/95 */
void mod_param (void)
{
    /* Declaracion de variables locales */
    int  inicpag, salir, opcion, opcion2, ultreng, numparam, modparam;
    int  pagactual, numpag, i;
    char op[80], op2[80], modif[80], val[80], err[80];
    float nuevalor;

    /* Menu modificacion parametros */
    salir=0;
    while (!salir) {
        opcion=0;
        while ((opcion > 5)||(opcion < 1)){
            clrscr ();
            printf("\n                MENU MODIFICACION PARAMETROS: \n\n\n\n\n");
            printf("          1) CINEMATICOS \n");
            printf("          2) DINAMICOS \n");
            printf("          3) CONTROL UNION \n");
            printf("          4) GENERACION TRAYECTORIA \n");
            printf("          5) SALIR \n");
            printf("                OPCION:");
            gets(op);
            opcion=atoi(op);
            if ((opcion > 5)||(opcion < 1)) {
                printf("\n OPCION EQUIVOCADA !!!");
                printf("\n PRESIONAR ENTER PARA CONTINUAR");
                gets(err);
            }
        }
        /* Trabajo de acuerdo a opcion */
        opcion2=0;
        switch (opcion) {
            case 1:  /* Parametros Cinematicos */
              /* Inicializacion paginas */
                pagactual=1;
                numpag=2;
                inicpag=1;
                ultreng=10;
                numparam=40;
              while (opcion2!=1){
                clrscr ();
                printf("          PARAMETROS CINEMATICOS \t\t\t pag(%-d) \n\n\n", pagactual);
                  for (i=inicpag; i<=ultreng; i=i+1) {
                        printf("%-d)\t%-s %-+.4f\t\t\t\t%-d)\t%-s %-+.4f\n",i, cinematicos[i].nombre,
                            cinematicos[i].val, i+10), cinematicos[i+10].nombre, cinematicos[i+10].val);
                  }
                  printf("\n\n\n\t\t\t\t  1) SALIR \n");
                  printf("\t\t\t\t  2) MODIFICAR \n");
                  printf("\t\t\t\t  3) ANTERIOR \n");
                  printf("\t\t\t\t  4) SIGUIENTE \n");
                  printf("\t\t\t\t             :");
                  opcion2=0;
                  while ((opcion2 > 4)||(opcion2 < 1)){
                      gets(op2);
                    opcion2=atoi(op2);
```

```
        if ((opcion2>4)||(opcion2<1)) printf("ERROR. LAS OPCIONES VALIDAS SON: 1,2,3 O 4. DAR OPCION: ");
      }
      switch (opcion2) {
          case 2:  /* Modificar */
            printf("\n DA EL NUMERO DEL PARAMETRO HA MODIFICAR: ");
              gets(modif);
            modparam=atoi(modif);
              if (modparam<=numparam) {
                  clrscr ();
                  cambio=1;
                printf("\n\n\n\n\n\n\n\n %-d) \t %-s %-.5E", modparam, cinematicos[modparam].nombre,cinematicos[modparam].val);
                printf("\n DA EL NUEVO VALOR O ENTER SI NO HAY CAMBIO: ");
                  gets(val);
                nuevalor=atof(val);
                if (val[0]!='\0') {
                      cinematicos[modparam].val=nuevalor;
                  }
              }
              break;
          case 3:  /* Anterior */
              if (pagactual!=1) {
                  pagactual=pagactual-1;
                  inicpag=inicpag-20;
                  ultreng=ultreng-20;
              }
              break;
          case 4:  /* Siguiente */
              if (pagactual<numpag) {
                  pagactual=pagactual+1;
                  inicpag=inicpag+20;
                  ultreng=ultreng+20;
              }
              break;
      }
    }
    break;

case 2:   /* Parametros dinamicos */
    /* Inicializacion paginas */
    pagactual=1;
  numpag=4;
    inicpag=1;
  ultreng=10;
    numparam=80;
  while (opcion2!=1){
      clrscr ();
      printf("          PARAMETROS DINAMICOS \t\t\t pag(%-d) \n\n\n", pagactual);
        for (i=inicpag; i<=ultreng; i=i+1) {
            printf("%-d)  %-s %-+.4f\t\t\t\t\t\t%-d)  %-s %-+.4f\n",i, dinamicos[i].nombre,
              dinamicos[i].val, (i+10), dinamicos[i+10].nombre, dinamicos[i+10].val);
        }
        printf("\n\n\n\n\t\t\t\t  1) SALIR \n");
        printf("\t\t\t\t  2) MODIFICAR \n");
        printf("\t\t\t\t  3) ANTERIOR \n");
        printf("\t\t\t\t  4) SIGUIENTE \n");
        printf("\t\t\t\t\t  :");
        opcion2=0;
        while ((opcion2>4)||(opcion2<1)){
            gets(op2);
          opcion2=atoi(op2);
          if((opcion2>4)||(opcion2<1)) printf("ERROR. LAS OPCIONES VALIDAS SON 1, 2, 3 O 4. DAR OPCIONES: ");
        }
        switch (opcion2) {
            case 2:  /* Modificar */
              printf("\n DA EL NUMERO DEL PARAMETRO HA MODIFICAR: ");
                gets(modif);
                modparam=atoi(modif);
                if (modparam<=numparam) {
                    cambio=1;
                    clrscr ();
                  printf("\n\n\n\n\n\n\n\n %-d) \t %-s %-+.5E", modparam, dinamicos[modparam].nombre,dinamicos[modparam].val);
                    printf("\n DA EL NUEVO VALOR O ENTER SI NO HAY CAMBIO: ");
```

```
                    gets(val);
                    nuevalor = atof(val);
                    if (val[0] != '\0') {
                        dinamicos[modparam].val = nuevalor;
                    }
                }
                break;
            case 3:  /* Anterior */
                if (pagactual != 1) {
                    pagactual = pagactual-1;
                  inicpag = inicpag-20;
                  ultreng = ultreng-20;
                }
                break;
            case 4:  /* Siguiente */
                if (pagactual < numpag) {
                    pagactual = pagactual+1;
                  inicpag = inicpag+20;
                  ultreng = ultreng+20;
                }
                break;
        }
    }
    break;

case 3:  /* Control de union  */
    /* Inicializacion paginas */
    pagactual = 1;
  numpag = 4;
    inicpag = 1;
  ultreng = 10;
    numparam = 80;
  while (opcion2 != 1){
      clrscr ();
      printf("          CONTROL DE UNION \t\t\t pag(%-d)\n\n\n",pagactual);
        for (i = inicpag; i <= ultreng; i = i+1) {
            printf("%-d)\t%-s %-+.4f\t\t\t\t%-d)\t%-s %-+.4f\n",i, cont_union[i].nombre, cont_union[i].val,
              (i+10), cont_union[i+10].nombre, cont_union[i+10].val);
        }
        printf("\n\n\n\t\t\t\t  1) SALIR \n");
        printf("\t\t\t\t  2) MODIFICAR \n");
        printf("\t\t\t\t  3) ANTERIOR \n");
        printf("\t\t\t\t  4) SIGUIENTE \n");
        printf("\t\t\t\t           :");
        opcion2 = 0;
        while ((opcion2 > 4)||(opcion2 < 1)){
            gets(op2);
          opcion2 = atoi(op2);
         if((opcion2 > 4)||(opcion2 < 1)) printf("ERROR, LAS OPCIONES VALIDAS SON 1, 2, 3 O 4. DAR OPCIONES: ");
        }
        switch (opcion2) {
            case 2:  /* Modificar */
             printf("\n DA EL NUMERO DEL PARAMETRO HA MODIFICAR: ");
              gets(modif);
              modparam = atoi(modif);
              if (modparam <= numparam) {
                  cambio = 1;
                  clrscr ();
                printf("\n\n\n\n\n\n\n %-d) \t %-s %-.5E", modparam, cont_union[modparam].nombre,cont_union[modparam].val);
                printf("\n DA EL NUEVO VALOR O ENTER SI NO HAY CAMBIO: ");
                  gets(val);
                nuevalor = atof(val);
                  if (val[0] != '\0') {
                      cont_union[modparam].val = nuevalor;
                  }
              }
              break;
            case 3:  /* Anterior */
                if (pagactual != 1) {
                    pagactual = pagactual-1;
                  inicpag = inicpag-20;
                  ultreng = ultreng-20;
```

```
                }
                break;
            case 4:  /* Siguiente */
                if (pagactual<numpag) {
                    pagactual=pagactual+1;
                  inicpag=inicpag+20;
                  ultreng=ultreng+20;
                }
                break;
        }
    }
    break;

case 4:  /* Generacion de trayectoria  */
    /* Inicializacion paginas */
    pagactual=1;
  numpag=2;
    inicpag=1;
  ultreng=10;
    numparam=40;
  while (opcion2!=1){
      clrscr ();
      printf("          PARAMETROS GENERACION TRAYECTORIA \t\t pag(%-d) \n\n\n", pagactual);
       for (i=inicpag; i<=ultreng; i=i+1) {
           printf("%-d)\t%-s %-+.4f\t\t\t\t%-d)\t%-s %-+.4f\n",i, gen_tray[i].nombre, gen_tray[i].val,
            (i+10), gen_tray[i+10].nombre, gen_tray[i+10].val);
       }
       printf("\n\n\n\t\t\t\t  1) SALIR \n");
       printf("\t\t\t\t  2) MODIFICAR \n");
       printf("\t\t\t\t  3) ANTERIOR \n");
       printf("\t\t\t\t  4) SIGUIENTE \n");
       printf("\t\t\t\t     :");
       opcion2=0;
       while ((opcion2>4)||(opcion2<1)){
          gets(op2);
         opcion2=atoi(op2);
        if((opcion2>4)||(opcion2<1)) printf("ERROR, LAS OPCIONES VALIDAS SON 1, 2, 3 O 4. DAR OPCIONES: ");
       }
       switch (opcion2) {
           case 2:  /* Modificar */
            printf("\n DA EL NUMERO DEL PARAMETRO HA MODIFICAR: ");
              gets(modif);
              modparam=atoi(modif);
              if (modparam<=numparam) {
                   cambio=1;
                   clrscr ();
                printf("\n\n\n\n\n\n %-d) \t %-s %-+.5E", modparam, gen_tray[modparam].nombre, gen_tray[modparam].val);
                printf("\n DA EL NUEVO VALOR O ENTER SI NO HAY CAMBIO: ");
                   gets(val);
                nuevalor=atof(val);
                   if (val[0] != '\0') {
                       gen_tray[modparam].val=nuevalor;
                   }
              }
              break;
           case 3:  /* Anterior */
              if (pagactual!=1) {
                  pagactual=pagactual-1;
                inicpag=inicpag-20;
                ultreng=ultreng-20;
              }
              break;
           case 4:  /* Siguiente */
              if (pagactual<numpag) {
                   pagactual=pagactual+1;
                inicpag=inicpag+20;
                ultreng=ultreng+20;
              }
              break;
       }
    }
    break;
```

```
            case 5:   /* Salir */
                salir=1;
                break;
        }
    }
    return;
}

/* Funcion grabacion/modificacion puntos */
void grab_mod_pun (void)
{
/* Declaracion de variables locales */
    #define PI 3.14159265
    struct simbolo {char etiqueta [4];};
    int numpun, salir, opcion, i, j, apuntador, error, resp_err;
    float temporal[6];
    char op[80], pun[80], op2[80], err[80], coord[80];
    struct simbolo ccart[7] = {{" "}, {"x="}, {"y="}, {"z="}, {"d="}, {"e="}, {"g="}};
    struct simbolo vuni[7] = {{" "}, {"q1="}, {"q2="}, {"q3="}, {"q4="}, {"q5="}, {"q6="}};
    salir=0;
    while (!salir){
        opcion=0;
        while ((opcion !=1) && (opcion !=2)) {
            /* Desplegado de la lista de puntos */
            clrscr ();
            for (i=0; i<=9; i=i++) {
                printf("P(%d): ", i);
                for (j=1; j<=6; j=j+1) {
                 if (puntos[i] [0] == 1) {
                    if (j<=3) printf("%s%+.4f ", ccart[j].etiqueta, puntos[i][j]);
                      else printf("%s%+.2f ", ccart[j].etiqueta, puntos[i][j]);
                   }
                  else printf("%s%+.2f ", vuni[j].etiqueta, puntos[i] [j]);
                }
                if(puntos[i][0] == 1) {
                    printf("\n      b=%+.0f   c=%+.0f   m=%+.0f",puntos[i][7], puntos[i][8], puntos[i][9]);
                }
                printf ("\n");
            }
            printf("\n                    MENU GRABACION/MODIFICACION PUNTOS");
            printf("\n          1) GRABACION/MODIFICACION DE PUNTOS");
            printf("\n          2) SALIR");
            printf("\n                    OPCION:");
            gets(op);
            opcion= atoi(op);
            if ((opcion!=1) && (opcion!=2)) {
                printf("\n OPCION EQUIVOCADA !!!");
                printf("\n PRESIONE ENTER PARA CONTINUAR");
                gets(err);
            }
        }
        if(opcion==1) {
            printf("\n DAR EL PUNTO A GRAB/MOD. [0-9]: ");
            gets(pun);
            numpun= atoi(pun);
            if ((numpun>9) || (numpun<0)) {
                printf("\n OPCION EQUIVOCADA !!!");
                printf("\n PRESIONE ENTER PARA CONTINUAR");
                gets(err);
            }
            else {
                clrscr();
                cambio=1;
                printf("\n\n las unidades usadas en las coordenadas de los puntos son:");
                printf("\n\t coord cartesianas: [m](posicion) y [grados](orientacion)");
                printf("\n\t var de union       [grados]");
                printf("\n\n\n COORDENADAS CARTESIANAS [S/N]? ");
                gets(op2);
              printf("\n\n    PUNTO (%d)   EN",numpun);
                if ((op2[0]=='S')||(op2[0]=='s')) {
                    puntos[numpun][0]=1;
                    error=1;
```

```
while(error) {
  clrscr();
   printf("\n     PUNTO (%d) EN COORDENADAS CARTESIANAS \n\n",numpun);
   printf("\t VALOR ACTUAL \t\t NUEVO VALOR (o ENTER) \n\n");
   for (i=1; i<=6; i=i+1) {
       printf("\t%s %+-.4f ", ecart[i].etiqueta, puntos[numpun] [i]);
       printf("\t\t %s ", ecart[i].etiqueta);
       gets (coord);
       if (coord[0]!='\0') puntos[numpun] [i] = atof(coord);
   }
   /* Asignacion de valores a brazo, codo y muñeca */
   printf("\n\n\t brazo=%+-.0f \t +1=derecho, -1=izquierdo", puntos[numpun][7]);
   printf("\n\t brazo= ");
   gets(coord);
   if(coord[0]!='\0') puntos[numpun][7]=atof(coord);
   if(fabs(puntos[numpun][7])!=1) puntos[numpun][7]=1;
   printf("\n\t codo=%+-.0f \t +1=arriba, -1=abajo", puntos[numpun][8]);
   printf("\n\t codo= ");
   gets(coord);
   if(coord[0]!='\0') puntos[numpun][8]=atof(coord);
   if(fabs(puntos[numpun][8])!=1) puntos[numpun][8]=1;
   printf("\n\t muñeca=%+-.0f \t +1=abajo, -1=arriba", puntos[numpun][9]);
   printf("\n\t muñeca= ");
   gets(coord);
   if(coord[0]!='\0') puntos[numpun][9]=atof(coord);
   if(fabs(puntos[numpun][9])!=1) puntos[numpun][9]=1;

   if((puntos[numpun][1]==0)&&(puntos[numpun][2]==0)) {
       printf("\n\n !!!ERROR, X=Y=0 NO ESTA DENTRO DEL ESPACIO DE TRABAJO");
       printf("\n PRESIONE ENTER PARA CONTINUAR");
       gets(err);
   }
   else {
       if((puntos[numpun][3]>(cinematicos[3].val+cinematicos[1].val+cinematicos[2].val))||
          (puntos[numpun][3]<(cinematicos[3].val-cinematicos[1].val-cinematicos[2].val))) {
           printf("\n\n !!!ERROR, LA 'Z' ESPECIFICADA ESTA FUERA DEL ESPACIO DE TRABAJO");
           printf("\n PRESIONE ENTER PARA CONTINUAR");
           gets(err);
       }
       else {
           resp_err=prob_inv(numpun, temporal);
           switch(resp_err) {
               case 0: /* Sin problemas */
                   error=0;
                   for (i=1; i<=6; i++) {
                       if((temporal[i-1]<=(PI/180)*cinematicos[4+(2*i)].val)&&(temporal[i-1]>=(PI/180)*cinematicos[5+(2*i)].val));
                       else {
                           error=1;
                           printf("\n\n !!!ERROR EN q%d= %+-.2f, VALOR FUERA DE RANGO",i,(180/PI)*temporal[i-1]);
                           printf("\n PRESIONE ENTER PARA CONTINUAR");
                           gets(err);
                       }
                   }
                   break;
                   case 1: /* Problema por singularidad */
                   printf("\n\n !!!ERROR, (X,Y) ESTA EN UNA ZONA INTERIOR QUE NO PERTENENCE");
                   printf("\n AL ESPACIO DE TRABAJO. SE TIENE QUE AUMENTAR CUALQUIERA DE LOS DOS");
                       printf("\n PRESIONE ENTER PARA CONTINUAR");
                       gets(err);
                       break;
                  case 2: /* (x,y) fuera del espacio de trabajo */
                       printf("\n\n !!!ERROR, (X,Y) ESTA POR FUERA DEL ESPACIO DE TRABAJO");
                       printf("\n PRESIONE ENTER PARA CONTINUAR");
                       gets(err);
                       break;
                  case 3: /* (x,y) bien pero z mal */
                       printf("\n\n !!!ERROR, (X,Y) ESTA BIEN, PERO 'Z' SE SALE DEL ESPACIO DE TRABAJO");
                       printf("\n APROXIMAR 'Z' A d1=%+-.4f",cinematicos[3].val);
                       printf("\n PRESIONE ENTER PARA CONTINUAR");
                       gets(err);
                       break;
           }
```

```
                        }
                    }
                }
            }
        else {
            puntos[numpun][0]=0;
            puntos[numpun][7]=0;
            puntos[numpun][8]=0;
            puntos[numpun][9]=0;
            clrscr();
            printf("\n\n    PUNTO (%d) EN VARIABLES DE UNION (en grados)\n\n\n",numpun);
            printf(" VALOR ACTUAL\t\t\t\t\tNUEVO VALOR (o ENTER)\n\n");
            for (i=1; i<=6; i=i+1) {
                error=1;
                while(error) {
                  printf("\n  %s %-+.2f\t", vuni[i].etiqueta, puntos[numpun][i]);
                    printf("\t\t%-+.2f\t <=q%d<= %-+.2f\t",cinematicos[5+(i*2)].val,i,cinematicos[4+(2*i)].val);
                    printf("\t\t%s ", vuni[i].etiqueta);
                    gets (coord);
                    if (coord[0]!='\0') {
                        if((atof(coord)<=cinematicos[4+(2*i)].val)&&(atof(coord)>=cinematicos[5+(2*i)].val)) {
                            puntos[numpun][i]=atof(coord);
                            error=0;
                        }
                        else {
                            printf("\n\n !!!ERROR VALOR FUERA DE RANGO");
                            printf("\n PRESIONAR ENTER PARA CONTINUAR");
                            gets(err);
                            clrscr();
                            printf("\n\n    PUNTO (%d) EN VARIABLES DE UNION (en grados)\n\n\n",numpun);
                            printf(" VALOR ACTUAL\t\t\t\t\tNUEVO VALOR (o ENTER)\n\n");
                        }
                    }
                    else error=0;
                }
            }
        }
    }
    else salir=1;
  }
  return;
}

/* Funcion especificacion secuencia movimientos */
void esp_sec_mov (void)
{
/* Declaracion de variables locales */
  int opcion, opcion2, salir, numsec, i, j, pun_inic, bien, pun_sig, punto;
  char op[80], op2[80], op3[80], err[80], entrada[80], entrada2[80];
  salir=0;
  while (!salir) {
    opcion=0;
    while ((opcion!=1)&&(opcion!=2)) {
      clrscr();
          printf("\n\n\t\t ESPECIFICACION DE LAS SECUENCIAS DE MOVIMIENTO\n");
      for (i=0; i<=9; i=i+1) {
        printf("\nSEC(%d):",i);
        for (j=0; j<=13; j=j+1) {
          if (sec[i] [j]!=30) {
            if (sec[i] [j]==10) printf("(FIN)");
            else {
              if (sec[i] [j]==20) printf("(CICLO)");
              else printf("P%d - ",sec[i] [j]);
            }
          }
        }
      }
      printf("\n\n\n\n\t\t 1) GRABAR/MODIFICAR SECUENCIA");
      printf("\n\t\t 2) SALIR");
      printf("\n\n\t\t\t OPCION:");
      gets(op);
```

```
    opcion = atoi(op);
    if ((opcion!=1)&&(opcion!=2)) {
      printf("\n OPCION EQUIVOCADA !!!");
      printf("\n PRESIONAR ENTER PARA CONTINUAR");
      gets(err);
    }

}
if  (opcion==2) salir=1;
else {
    /* opcion grabar/modificar secuencia */
    printf("\n DA EL NUMERO DE LA SECUENCIA A GRAB/MOD: ");
    gets(entrada);
    numsec= atoi(entrada);
    if (entrada[0] != '\0') {
      if ((numsec>=0) && (numsec<=9)) {
        /* entrada buena */
        cambio=1;
        clrscr();
        printf("\n\n\n\n\n SEC(%d):",numsec);
        for (j=0; j<=13; j=j+1) {
          if (sec[numsec] [j]!=30) {
            if (sec[numsec] [j]==10) printf("(FIN)");
            else {
              if (sec[numsec] [j]==20) printf("(CICLO)");
              else printf("P%d - ",sec[numsec] [j]);
            }
          }
        }
        printf("\n\n DA EL PUNTO INICIAL DE LA SECUENCIA: ");
        pun_inic=10;
        while ((pun_inic>9)||(pun_inic<0)){
          gets(entrada2);
          pun_inic= atoi(entrada2);
          if (entrada2[0]!='\0') {
            if ((pun_inic>9)||(pun_inic<0)) {
              printf("\n OPCION EQUIVOCADA. DA PUNTO INICIAL (0 a 9): ");
            }
            else sec[numsec][0]=pun_inic;
          }
          else {
            printf("\n SE TIENE QUE SELECCIONAR UN PUNTO INICIAL. DA PUNTO INICIAL (0 a 9): ");
            punto=10;
          }
        }
        opcion2=1;
        pun_sig=1;
        bien=0;
        while (opcion2==1) {
          clrscr();
          printf("\n\t\t\t GRABACION/MODIFICACION SECUENCIA");
          printf("\n\n SEC(%d):",numsec);
          for (j=0; j<=13; j=j+1) {
            if (sec[numsec] [j]!=30) {
              if (sec[numsec] [j]==10) printf("(FIN)");
              else {
                if (sec[numsec] [j]==20) printf("(CICLO)");
                else printf("P%d - ",sec[numsec] [j]);
              }
            }
          }
          printf("\n\n\n\t\t\t\t\t PASO(%d)", pun_sig);
          printf("\n            1) PUNTO SIGUIENTE");
          printf("\n            2) CICLO");
          printf("\n            3) FIN");
          printf("\n            4) SALIR");
          printf("\n\n                  OPCION: ");
          gets(op2);
          opcion2= atoi(op2);
          if ((opcion2>0)&&(opcion2<5)) {
            if ((opcion2!=1)&&(sec[numsec][1]==30)) {
              opcion2=1;
```

```
      printf("\n\n UNA SECUENCIA DEBE DE CONTAR CON AL MENOS 2 PUNTOS");
      printf("\n SE DEBE DE DAR EL SIGUIENTE PUNTO");
      printf("\n\n PRESIONAR ENTER PARA CONTINUAR");
      gets(err);
    }
    else {
      switch (opcion2) {
      case 1:  /* punto siguiente */
        if (pun_sig <= 12) {
          punto=10;
          printf("DA EL PUNTO SIGUIENTE (0 al 9): ");
          while ((punto<0)||(punto>9)) {
            gets(entrada);
            punto= atoi(entrada);
            if (entrada[0]!='\0') {
              if ((punto<0)||(punto>9)) {
                printf("\n OPCION EQUIVOCADA, DA PUNTO SIGUIENTE (0 al 9): ");
              }
              else {
                sec[numsec][pun_sig]=punto;
                pun_sig=pun_sig+1;
              }
            }
            else printf("\n SE TIENE QUE SELECCIONAR UN PUNTO (0 a 9): ");
          }
        }
        else {
          printf("\n EL MAXIMO DE PUNTOS EN UNA SECUENCIA ES DE 13");
          printf("\n SE TIENE QUE CERRAR ESTA SECUENCIA, YA SEA CON 'FIN' O 'CICLO'");
          printf("\n PRESIONAR ENTER PARA CONTINUAR");
          gets(err);
        }
        break;
      case 2: /* ciclo */
        if (pun_sig==1) {
          printf("\n UN CICLO POR LO MENOS DEBE DE CONTAR CON 2 PUNTOS");
          printf("\n PRESIONAR ENTER PARA CONTINUAR");
          gets(err);
          opcion2=1;
        }
        else {
          printf("\n SI EN LA SECUENCIA HAY PUNTOS DEFINIDOS, DESPUES DE DONDE SE");
          printf("\n PONDRA (CICLO), ESTOS SE VAN A A BORRAR. ESTAS SEGURO? [S/N]?");
          gets(op3);
          if ((op3[0]=='S')||(op3[0]=='s')) {
            sec[numsec][pun_sig]=20;
            pun_sig=pun_sig+1;
            for (j=pun_sig; j<=13; j=j+1) {
              sec[numsec][j]=30;
            }
          }
          else opcion2=1;
        }
        break;
      case 3: /* fin */
        printf("\n SI EN LA SECUENCIA HAY PUNTOS DEFINIDOS, DESPUES DE DONDE SE");
        printf("\n PONDRA (FIN), ESTOS SE VAN A A BORRAR. ESTAS SEGURO? [S/N]?");
        gets(op3);
        if ((op3[0]=='S')||(op3[0]=='s')) {
          sec[numsec][pun_sig]=10;
          pun_sig=pun_sig+1;
          for (j=pun_sig; j<=13; j=j+1) {
            sec[numsec][j]=30;
          }
        }
        else opcion2=1;
        break;
      case 4: /* salir */
        for (j=2; j<=13; j=j+1) {
          if ((sec[numsec][j]==10)||(sec[numsec][j]==20)) bien=1;
        }
        if (!bien) {
```

```
                    opcion2 = 1;
                    printf("\n ERROR !!!");
                    printf("\n UNA SECUENCIA SE DEBE DE TERMINAR CON (FIN) O CON (CICLO)");
                    printf("\n PRESIONAR ENTER PARA CONTINUAR");
                    gets(err);
                  }
                }
              }
            }
            else {
              opcion2 = 1;
              printf("\n OPCION EQUIVOCADA !!!");
              printf("\n PRESIONAR ENTRAR PARA CONTINUAR");
              gets(err);
            }
          }
          /* Especificacion de tiempos de recorrido */
          clrscr();
          printf("\n\n\n\n\n SEC(%-d): ",numsec);
          for (j=0; j<=13; j=j+1) {
            if (sec[numsec] [j]!=30) {
              if (sec[numsec] [j] == 10) printf("(FIN)");
              else {
                if (sec[numsec] [j] == 20) printf("(CICLO)");
                else printf("P%-d - ",sec[numsec] [j]);
              }
            }
          }
          printf("\n\n DA EL TIEMPO DE RECORRIDO, EN SEGUNDOS DEL:\n");
          for (j=0; j<=13; j=j+1) {
            if ((sec[numsec][j]<10)&&(sec[numsec][j+1]<10)) {
              printf("\n TRAYECTO(%-d): del P(%-d) al P(%-d), ",(j+1), sec[numsec][j], sec[numsec][j+1]);
              if (tiem_sec[numsec][j]!=0) {
                printf("\n EL TIEMPO DE RECORRIDO ES %-.2f ",tiem_sec[numsec][j]);
                printf("\n DA EL NUEVO VALOR O ENTER SI NO HAY CAMBIO: ");
              }
              else printf("\n DA EL TIEMPO DE RECORRIDO: ");
              gets(entrada2);
              if (entrada2[0]!='\0') tiem_sec[numsec][j]= atof(entrada2);
            }
            else {
              if (sec[numsec][j+1]==20) {
              printf("\n TRAYECTO(%-d): del P(%-d) al P(%-d), ",(j+1), sec[numsec][j], sec[numsec][0]);
                if (tiem_sec[numsec][j]!=0) {
                  printf("\n EL TIEMPO DE RECORRIDO ES %-.2f ",tiem_sec[numsec][j]);
                  printf("\n DA EL NUEVO VALOR O ENTER SI NO HAY CAMBIO: ");
                }
                else printf("\n DA EL TIEMPO DE RECORRIDO: ");
                gets(entrada2);
                if (entrada2[0]!='\0') tiem_sec[numsec][j]= atof(entrada2);
              }
              else tiem_sec[numsec][j] = 0.0;
            }
          }
          /* (posible) especificacion de puntos despegue/asentamiento */

        }
        else {
          printf("\n OPCION EQUIVOCADA !!!");
          printf("\n PRESIONAR ENTER PARA CONTINUAR");
          gets(err);
        }
      }
      else {
        printf("\n SE TIENE QUE ELEGIR ALGUNA SECUENCIA (0 a 9)");
        printf("\n PRESIONAR ENTER PARA CONTINUAR");
        gets(err);
      }
    }
  }
  return;
}
```

```
/* Funcion de inicializacion de las variables globales, por un archivo */
void inicializa (void)
{
   FILE *archivo;
   int i;
   char err[80];
   if ((archivo=fopen("DATOS.DAT","r"))==NULL) {
        printf("\n\t No se encontro el archivo de datos. Todas las variables se");
        printf("\n\t van a inicializar a cero. Presionar ENTER para continuar..");
        gets(err);
   }
   else {
       /* Inicializacion de parametros cinematicos */
       for (i=0; i<=36; i=i+4) {
              fscanf(archivo,"%5s %f %5s %f %5s %f %5s %f",
                cinematicos[i].nombre,   &cinematicos[i].val,
                cinematicos[i+1].nombre, &cinematicos[i+1].val,
                cinematicos[i+2].nombre, &cinematicos[i+2].val,
                cinematicos[i+3].nombre, &cinematicos[i+3].val);
       }
       fscanf(archivo,"%5s %f",cinematicos[40].nombre, &cinematicos[40].val);

       /* Inicializacion de parametros dinamicos */
       for (i=0; i<=76; i=i+4) {
              fscanf(archivo,"%5s %f %5s %f %5s %f %5s %f",
                dinamicos[i].nombre,   &dinamicos[i].val,
                dinamicos[i+1].nombre, &dinamicos[i+1].val,
                dinamicos[i+2].nombre, &dinamicos[i+2].val,
                dinamicos[i+3].nombre, &dinamicos[i+3].val);
       }
       fscanf(archivo,"%5s %f", dinamicos[80].nombre, &dinamicos[80].val);

       /* Inicializacion de parametros de control de union */
       for (i=0; i<=76; i=i+4) {
              fscanf(archivo,"%5s %f %5s %f %5s %f %5s %f",
                cont_union[i].nombre,   &cont_union[i].val,
                cont_union[i+1].nombre, &cont_union[i+1].val,
                cont_union[i+2].nombre, &cont_union[i+2].val,
                cont_union[i+3].nombre, &cont_union[i+3].val);
       }
       fscanf(archivo,"%5s %f", cont_union[80].nombre, &cont_union[80].val);

            /* Inicializacion de parametros de generacion de trayectoria */
       for (i=0; i<=36; i=i+4) {
              fscanf(archivo,"%5s %f %5s %f %5s %f %5s %f",
                gen_tray[i].nombre,   &gen_tray[i].val,
                gen_tray[i+1].nombre, &gen_tray[i+1].val,
                gen_tray[i+2].nombre, &gen_tray[i+2].val,
                gen_tray[i+3].nombre, &gen_tray[i+3].val);
       }
            fscanf(archivo,"%5s %f", gen_tray[40].nombre, &gen_tray[40].val);

       /* Inicializacion de los puntos */
       for (i=0; i<=9; i++) {
              fscanf(archivo, "%f %f %f %f %f %f %f",
                &puntos[i][0], &puntos[i][1], &puntos[i][2], &puntos[i][3],
               &puntos[i][4], &puntos[i][5], &puntos[i][6]);
                fscanf(archivo, "%f %f %f",
                &puntos[i][7], &puntos[i][8], &puntos[i][9]);
       }

       /* Inicializacion de las secuencias de movimientos */
       for (i=0; i<=9; i++) {
            fscanf(archivo, "%d %d %d %d %d %d %d %d %d %d %d %d %d %d %d",
            &sec[i][0],  &sec[i][1],  &sec[i][2],  &sec[i][3], &sec[i][4],
            &sec[i][5],  &sec[i][6],  &sec[i][7],  &sec[i][8], &sec[i][9],
            &sec[i][10], &sec[i][11], &sec[i][12], &sec[i][13] );
       }

       /* Inicializacion del tiempo involucrado en las secuencias */
       for (i=0; i<=9; i++) {
            fscanf(archivo, "%f %f %f %f %f %f %f %f %f %f %f %f %f %f %f",
```

```
            &tiem_sec[i][0], &tiem_sec[i][1], &tiem_sec[i][2], &tiem_sec[i][3],
            &tiem_sec[i][4], &tiem_sec[i][5], &tiem_sec[i][6], &tiem_sec[i][7],
            &tiem_sec[i][8], &tiem_sec[i][9], &tiem_sec[i][10], &tiem_sec[i][11],
            &tiem_sec[i][12], &tiem_sec[i][13] );
        }
    }
    fclose(archivo);
    return;
}

/* Funcion para guardar el contenido de las variables globales en un archivo */
void guardar (void)
{
    FILE *archivo;
    int i;
    char op[80];
    if (cambio == 1) {
        printf("\n\n\n\t\t Hay modificaciones en el valor(es) de los parametros");
        printf("\n\t\t Quieres que se guarden los cambios [s/n]:" );
        gets(op);
        if ((op[0] == 'S')||(op[0] == 's')) {
            archivo = fopen("datos.dat", "w");

            /* Almacenado de parametros cinematicos */
            for (i=0; i<=36; i=i+4) {
              fprintf(archivo,"%-5s %-f %-5s %-f %-5s %-f %-5s %-f \n",
                cinematicos[i].nombre,  cinematicos[i].val,
                cinematicos[i+1].nombre, cinematicos[i+1].val,
                cinematicos[i+2].nombre, cinematicos[i+2].val,
                cinematicos[i+3].nombre, cinematicos[i+3].val);
            }
            fprintf(archivo,"%-5s %-f \n", cinematicos[40].nombre, cinematicos[40].val);


            /* Almacenado de parametros dinamicos */
            for (i=0; i<=76; i=i+4) {
              fprintf(archivo,"%-5s %-f %-5s %-f %-5s %-f %-5s %-f \n",
                dinamicos[i].nombre,  dinamicos[i].val,
                dinamicos[i+1].nombre, dinamicos[i+1].val,
                dinamicos[i+2].nombre, dinamicos[i+2].val,
                dinamicos[i+3].nombre, dinamicos[i+3].val);
            }
            fprintf(archivo,"%-5s %-f \n", dinamicos[80].nombre, dinamicos[80].val);

            /* Almacenado de parametros de control de union */
            for (i=0; i<=76; i=i+4) {
              fprintf(archivo,"%-5s %-f %-5s %-f %-5s %-f %-5s %-f \n",
                cont_union[i].nombre,  cont_union[i].val,
                cont_union[i+1].nombre, cont_union[i+1].val,
                cont_union[i+2].nombre, cont_union[i+2].val,
                cont_union[i+3].nombre, cont_union[i+3].val);
            }
            fprintf(archivo,"%-5s %-f \n", cont_union[80].nombre, cont_union[80].val);

            /* Almacenado de parametros de generacion de trayectoria */
            for (i=0; i<=36; i=i+4) {
              fprintf(archivo,"%-5s %-f %-5s %-f %-5s %-f %-5s %-f \n",
                gen_tray[i].nombre,  gen_tray[i].val,
                gen_tray[i+1].nombre, gen_tray[i+1].val,
                gen_tray[i+2].nombre, gen_tray[i+2].val,
                gen_tray[i+3].nombre, gen_tray[i+3].val);
            }
            fprintf(archivo,"%-5s %-f \n", gen_tray[40].nombre, gen_tray[40].val);

            /* Almacenado de los puntos */
            for (i=0; i<=9; i++) {
              fprintf(archivo, "%-f %-f %-f %-f %-f %-f %-f \n",
                puntos[i][0], puntos[i][1], puntos[i][2], puntos[i][3],
                puntos[i][4], puntos[i][5], puntos[i][6]);
                fprintf(archivo, "%-f %-f %-f \n",
                puntos[i][7], puntos[i][8], puntos[i][9]);
            }
```

```
            /* Almacenado de las secuencias de movimientos */
            for (i=0; i<=9; i++) {
              fprintf(archivo, "%d %d %d %d %d %d %d %d %d %d %d %d %d %d \n",
                sec[i][0], sec[i][1], sec[i][2], sec[i][3], sec[i][4],
                sec[i][5], sec[i][6], sec[i][7], sec[i][8], sec[i][9],
                sec[i][10], sec[i][11], sec[i][12], sec[i][13] );
            }

            /* Almacenado del tiempo involucrado en las secuencias */
            for (i=0; i<=9; i++) {
              fprintf(archivo, "%f %f %f %f %f %f %f %f %f %f %f %f %f %f \n",
                tiem_sec[i][0], tiem_sec[i][1], tiem_sec[i][2], tiem_sec[i][3],
                tiem_sec[i][4], tiem_sec[i][5], tiem_sec[i][6], tiem_sec[i][7],
                tiem_sec[i][8], tiem_sec[i][9], tiem_sec[i][10], tiem_sec[i][11],
                tiem_sec[i][12], tiem_sec[i][13] );
            }
        }
    }
    fclose(archivo);
    return;
}
```

```
/* Programa con las rutinas comprendidas en el menu inicio operaciones.
   Cuarta version 27/sep/95 (hasta implementar control) */

#include <math.h>
#include <graphics.h>
#include <alloc.h>
#include <conio.h>
#define PI 3.14159265

/* Variables globales */
struct parametros {
      char nombre[6];
      float val;
};
extern struct parametros cinematicos[41];
extern struct parametros dinamicos[81];
extern struct parametros cont_union[81];
extern struct parametros gen_tray[41];
extern float puntos[10][10], tiem_sec[10][14];
extern int sec[10][14];
static float t[17], h[17], ddq[17][6], pos_d[6], vel_d[6], ace_d[6], q[17][6];
static int ciclo, coord_x, coord_y;

void inic_op (void)
{
   /* Variables locales */
   int salir, opcion, numsec, error, i, n, var, num_tiem, numvar, uniones,cambsec;
   int graphdriver = VGA;
   int graphmode=VGAHI;
   int g_errorcode, k, stop_graf=0;
   float tiem_tot, vmax_ejevert, valor, *ptr_tiem_tot, real, var_cont, par, k_gt;
   float *ptr_g1, *ptr_g2, *ptr_h1, *ptr_h2, *ptr_hp1, *ptr_hp2, *ptr_k2_1,
   *ptr_k2_2, *ptr_k1, *ptr_ke, g1, g2, h1, h2, hp1, hp2, k2_1, k2_2, k1, ke;
   char op1[80], op2[80], err[80], titulo[50], info[30];
   unsigned nada=0;
   struct nombre { char etiqueta[10]; };
   struct nombre variables[11]={
     {"q"}, {"q'"}, {"T"}, {"u(k)"}, {"qd"}, {"qd'"}, {"qd''"}, {"error"}, {"error_ob"}, {"q'_ob"}, {"int_u²"}
   };

   /* Declaracion de funciones */
   void grafica(int, float, float, float);
   int prev_gt(int);
   void g_trayec(int, int, float);
   void prob_direc(void);
   void jacobiano(void);
   void prev_cont(int, float *, float *, float *, float *, float *, float *,
                  float *, float *, float *, float *, float *);
   float control(int, float, float, float, float, float, float, float, float, float, float, float, float, int, int);
   float dinamica(int, float pos_d[], float vel_d[], float ace_d[]);

   /* Inicializaciones */
   error=0;
   var=100;
   salir=0;
   cambsec=1;
   while (!salir){
            if (cambsec){   /* cambsec es una bandera para controlar el cambio de secuencia */
               num_tiem=0;
               tiem_tot=0;
            /* Peticion del numero de secuencia a simular */
            opcion=10;
            cambsec=0;   /* Se cancela el cambio de secuencia hasta que se especifique */
            while ((opcion<0)||(opcion>9)) {
                clrscr();
                printf("\t\t\t  MENU INICIO OPERACION \n\n\n");
                printf("DA EL NUMERO DE SECUENCIA A SIMULAR: ");
                gets (op1);
                opcion=atoi(op1);
                if ((opcion<0)||(opcion>9)){
                    printf("\n ERROR !! LAS OPCIONES VALIDAS SON DEL 0 AL 9");
                    printf("\n PRESIONAR ENTER PARA CONTINUAR");
```

```
                gets (err);
            }
            else numsec=opcion;
        }
        /* Verificacion de los tiempos involucrados en la secuencia */
        for (i=1; i<=13; i++){
            if ((i==1)&&(sec[numsec][i]>=10)){
                printf("\n ERROR !! NO HAY SECUENCIA");
                printf("\n PRESIONAR ENTER PARA CONTINUAR");
                gets (err);
                error=1;
            }
            if ((sec[numsec][i]<=9)||(sec[numsec][i]==20)){
                num_tiem=num_tiem+1;
                if (tiem_sec[numsec][i-1]<=0) {
                    printf("\n ERROR !! EN EL TIEMPO DEL TRAYECTO(%d)", i);
                    printf("\n PRESIONAR ENTER PARA CONTINUAR");
                    gets(err);
                    error=1;
                }
                else tiem_tot=tiem_tot + tiem_sec[numsec][i-1];
            }
        }
    }
    if (error==1) opcion=4;
    else {
        opcion=5;
        while((opcion<1)||(opcion>4)){
            clrscr();
            printf("\t\t\t  MENU SIMULACION");
            printf("\n\n SECUENCIA A SIMULAR:");
            /* Despiegar la secuencia */
            printf("\n SEC(%d):", numsec);
            for (i=0; i<=13; i++) {
                if (sec[numsec][i]!=30) {
                    if (sec[numsec][i]==10) printf("(FIN)");
                    else {
                        if (sec[numsec][i]==20) printf("(CICLO)");
                        else printf("P%d - ",sec[numsec][i]);
                    }
                }
            }
            printf ("\n\n TIEMPOS INVOLUCRADOS:");
            for (i=0; i<=6; i++){
              if(i<=(num_tiem-1)){
                  printf("\n(%d) P(%d) a ", i, sec[numsec][i]);
                  if (sec[numsec][i+1]==20) printf("P(%d) = %-.4f", sec[numsec][0], tiem_sec[numsec][i]);
                  else printf("P(%d) = %-.4f", sec[numsec][i+1], tiem_sec[numsec][i]);
                  if ((i+7)<=(num_tiem-1)) {
                        printf("\t(%d) P(%d) a ",(i+7), sec[numsec][i+7]);
                        if (sec[numsec][i+8]==20) printf("P(%d) = %-.4f", sec[numsec][0], tiem_sec[numsec][i+7]);
                        else printf("P(%d) = %-.4f", sec[numsec][i+8], tiem_sec[numsec][i+7]);
                  }
              }
            }
            printf("\n\n TIEMPO TOTAL = %-.4f seg", tiem_tot);
            printf("\n\n VARIABLE A GRAFICAR: ");
            uniones=var/100;
            numvar=var-(uniones*100);
            printf("%s  UNION: %d", variables[numvar].etiqueta, uniones);
            printf("\n\n\n\t\t  1) INICIO SIMULACION");
            printf("\n\t\t  2) CAMBIO DE SECUENCIA A SIMULAR");
            printf("\n\t\t  3) CAMBIO DE VAR A GRAFICAR");
            printf("\n\t\t  4) SALIR");
            printf("\n\t\t\t OPCION: ");
            gets(op1);
            opcion=atoi(op1);
            if ((opcion<1)||(opcion>4)){
                printf("\n ERROR !! LAS OPCIONES VALIDAS SON DEL 1 AL 4");
                printf("\n PRESIONAR ENTER PARA CONTINUAR");
                gets(err);
            }
```

```
    }
}
switch(opcion){
    case 1: /* Simulacion */
      clrscr();
      /* Inicializacion de variables para la simulacion */
       vmax_ejevert=0;
       k_gt=0;
      cont_union[56].val=0;
       ptr_g1=&g1;
       ptr_g2=&g2;
       ptr_h1=&h1;
       ptr_h2=&h2;
       ptr_hp1=&hp1;
       ptr_hp2=&hp2;
       ptr_k2_1=&k2_1;
       ptr_k2_2=&k2_2;
       ptr_k1=&k1;
      ptr_ke=&ke;
      cont_union[53].val=0; /* Variable de la integracion de u² */

      /* Rutina previa a simulacion, en la generacion de trayectoria */
      n=prev_gt(numsec);

        /* Rutina previa a control */
        prev_cont(uniones, ptr_g1, ptr_g2, ptr_h1, ptr_h2, ptr_hp1, ptr_hp2,ptr_k2_1, ptr_k2_2, ptr_k1, ptr_ke);

      /* Pedir por el titulo */
      printf("\n\n\t DA EL TITULO DE LA GRAFICA, (max 49 carac):\n\t");
      gets(titulo);
      /* Pedir por el valor maximo en el eje vertical */
      while(vmax_ejevert<=0) {
            printf("\n\t DA EL VALOR MAXIMO EN EL EJE VERTICAL \n");
            printf("\t(el eje es simetrico con cero en medio): ");
            gets(op1);
            vmax_ejevert=atof(op1);
            if (vmax_ejevert<=0) printf("\n ¡¡¡ ERROR, EL VALOR MAXIMO DEBE SER MAYOR QUE CERO");
       }
       /* Se detiene la simulacion si la grafica se sale del espeacio */
       printf("\n\t SE DETIENE LA SIMULACION SI EL VALOR GRAFICADO SUPERA AL VALOR MAX?");
       printf("\n\t [S/N]:");
       gets(op1);
      if((op1[0]=='S')||(op1[0]=='s')) stop_graf=1;
      else stop_graf=0;

        /* Arreglo del modo grafico */
      initgraph(&graphdriver, &graphmode, "C:\\TC");
      g_errorcode=graphresult();
      if (g_errorcode!=grOk) {
            printf("\n\n Error al entrar al modo grafico: %s \n", grapherrormsg(g_errorcode));
            exit(1);
       }

       /* Inicializacion de font y tipo de linea */
       settextstyle(SMALL_FONT, HORIZ_DIR, 0);
        setusercharsize(12,10,12,10);
       setlinestyle(SOLID_LINE, nada, NORM_WIDTH);
        setcolor(EGA_LIGHTGRAY);

       /* Inicializacion de la pantalla (titulo, info, etc=  */
       rectangle(0,0, 639,479);
        line(0,439, 639,439);
        line(400,439, 400,679);

       uniones=var/100;
       numvar=var-(uniones*100);
       sprintf(info, "[t,%s]   UNION %d", variables[numvar].etiqueta, uniones);
       settextjustify(LEFT_TEXT, TOP_TEXT);
       outtextxy(65,2, info);
       settextjustify(CENTER_TEXT, TOP_TEXT);
       outtextxy(380,2, titulo);
       setviewport(0,439,400,479,1);
```

```
            moveto(10,10);
            setusercharsize(1,1,1,1);
            settextjustify(LEFT_TEXT, BOTTOM_TEXT);
            sprintf(op1,"SEC(%-d):",numsec);
            outtext(op1);
            for(i=0; i<=13; i++) {
                if (sec[numsec][i]!=30) {
                    if (sec[numsec][i]==10) outtext("(FIN)");
                    else {
                        if (sec[numsec][i]==20) outtext("(CICLO)");
                        else {
                            sprintf(op1,"P%-d -", sec[numsec][i]);
                            outtext(op1);
                        }
                    }
                }
            }
            setviewport(400,439,639,479,1);
            setusercharsize(1,1,1,1);
            settextjustify(LEFT_TEXT, TOP_TEXT);
            sprintf(op1,"SIS: z=%.3f w=%.2f",cont_union[2+(4*(uniones-1))].val,cont_union[3+(4*(uniones-1))].val);
            outtextxy(4,3,op1);
            settextjustify(RIGHT_TEXT, TOP_TEXT);
            sprintf(op1,"T=%.4E [s]", cont_union[1].val);
            outtextxy(239,3,op1);
            settextjustify(LEFT_TEXT, BOTTOM_TEXT);

            sprintf(op1,"      SERVO              OBSERVADOR");
            outtextxy(10,27,op1);
            sprintf(op1,"  Z=%f(-%.1f*wnT)          Z=  %+.3f",cont_union[4+(4*(uniones-1))].val,cont_union[5+(4*(uniones-1))].val);
            outtextxy(10,37,op1);

            /* Ejes */
            setviewport(0,0,639,479,1);
            line(10,13,10,435);
            line(7,224,632,224);
            for(i=0; i<=10; i++) {   /* Marcas eje vertical */
                line(7,(14+(i*42)),12, (14+(i*42)));
            }
            for(i=0; i<=9; i++) {   /* Marcas eje horizontal */
                line((72+(i*62)),222, (72+(i*62)),226);
            }
            /* Numeros en eje horizontal */
          setusercharsize(10,10,10,10);
            settextjustify(CENTER_TEXT,TOP_TEXT);
            for(i=1; i<=5; i++) {
                sprintf(op1, "%-.2f", ((ciclo+1)*tiem_tot*0.2*i));
                outtextxy((5+(i*124)),228, op1);
            }
            /* Numeros en eje vertical */
            settextjustify(LEFT_TEXT,CENTER_TEXT);
          for(i=-5; i<=5; i=i+2) {
                real=i;
                sprintf(op1, "%+.2f", (vmax_ejevert*(real/5)));
                outtextxy(13, (224-(i*42)), op1);
            }
/* Fin de la inicializacion de la pantalla */
```

```
/* Inicio ciclo de simulacion de la variables elegida */
            for (k=0; k<=((ciclo+1)*tiem_tot/cont_union[1].val); k++) {
                /* Generacion de trayectoria */
                if(((k*cont_union[1].val)/gen_tray[3].val)>=k_gt) {
                    k_gt=k_gt+1;
                    g_trayec(n,k,((ciclo+1)*tiem_tot));
                }
                if(numvar!=4 && numvar!=5 && numvar!=6) {
                    /* Dinamica */
                  par=dinamica(uniones, pos_d, vel_d, ace_d);
                    if(numvar!=2) {
                        /* Control */
                      var_cont=control(k, pos_d[uniones-1], (cont_union[26].val*par), g1, g2, h1, h2, hp1, hp2, k2_1, k2_2, k1, ke, numvar,uniones);
                    }
                }
                /* Graficar la variable escogida */
                switch(numvar) {
                    case 0: /* Posicion medida (q) */
                      valor=var_cont;
                        break;
                    case 1: /* Velocidad observada (q') */
                      valor=var_cont;
                        break;
                    case 2: /* Par (T) */
                        valor=par;
                        break;
                    case 3: /* Entrada a la planta (u(k)) */
                        valor=var_cont;
                        break;
                    case 4: /* Posicion deseada (qd)=pos_d[] */
                        valor=(180/PI)*pos_d[uniones-1];
                        break;
                    case 5: /* Velocidaa deseada (qd')=vel_d[] */
                      valor=(180/PI)*vel_d[uniones-1];
                        break;
                    case 6: /* Aceleracion deseada (qd'')=ace_d[] */
                        valor=(180/PI)*ace_d[uniones-1];
                        break;
                    default: /* Error (error), Error observador (error_ob) y vel. observada (q'_ob) */
                      valor=var_cont;
                }
                if((fabs(valor)>vmax_ejevert)&&(stop_graf)) break;
                grafica(k, valor, ((ciclo+1)*tiem_tot), vmax_ejevert);
                /* Deteccion de la activacion de alguna tecla */
                if(kbhit()) {
                    i=getch();
                    if(i==' ') {
                        /* Problema Directo */
                        prob_direc();

                        /* Jacobiano */
                        jacobiano();

                        /* Marcar tiempo (k) de X y dev(X) */
                        i=getch();
                        if(i=='M'||i=='m') {
                            setcolor(EGA_LIGHTCYAN);
                            setviewport(0,0,639,479,1);
                            circle(coord_x,coord_y,4);
                            circle(coord_x,coord_y,1);
                            /* coord_x y coord_y son var static, que se evaluan
                            en la funcion grafica */
                        }
                    }
                    else break;
                }
                    }
                    gets(err);    /* Detencion de la imagen grafica hasta que se presione ENTER */
                    closegraph(); /* Fin de modo grafico */
            break;
        case 2: /* Cambio de secuencia a simular */
            cambsec=1;
```

```
                break;
            case 3: /* Cambio de variable a graficar */
                uniones=7;
                while ((uniones<1)||(uniones>6)) {
                    clrscr();
                    printf("\n\t\t\t   CAMBIO DE VARIABLES A GRAFICAR");
                    printf("\n\n DA LA UNION DE LA VARIABLE: ");
                    gets(op2);
                    uniones=atoi(op2);
                    if ((uniones<1)||(uniones>6)){
                        printf("\n  ERROR !! LAS UNIONES SON DEL 1 AL 6");
                        printf("\n  PRESIONAR ENTER PARA CONTINUAR");
                        gets(err);
                    }
                }
                numvar=11;
                printf("\n\n\ DA LA VARIABLE A GRAFICAR");
                while((numvar<0)||(numvar>10)){
                    for (i=0; i<=10; i++) {
                        printf("\n %d) %s", i, variables[i].etiqueta);
                    }
                    printf("\n\n\tu OPCION: ");
                    gets (op2);
                    numvar=atoi(op2);
                    if((numvar<0)||(numvar>10)) {
                        printf("\n  ERROR!! LAS OPCIONES SON DEL 0 AL 10");
                        printf("\n  PRESIONAR ENTER PARA CONTINUAR");
                        gets(err);
                        clrscr();
                        printf("  UNION: %d \n\n", uniones);
                    }
                }
                var=(uniones*100)+numvar;
                break;
            case 4: /* Salir */
                salir=1;
        }
    }
    return;
}

/* Funcion para graficar los valores generados en la graficacion.
   Se calcula todo para un monitor VGA (640,480) y tras titulos y demas info
   se tiene un espacio para graficas de (10,14) a (630,432) con el cero en el
   eje vertical en y=224.   */

void grafica(int k, float valor, float tiem_tot, float vmax_ejevert)
{
    int x, y, i;
    float a,b,c;  /*variables temporales, para el redondeo en coord. x,y */
    char op1[100];
    setviewport(0,0,639,479,1);

    /* Coord. en X  */
    a=(cont_union[1].val/tiem_tot)*k*620;
    b=floor(a+0.5);
    x=10+b;

    /* Coord en Y  */
    a=210*(valor/vmax_ejevert);
    c=fabs(a);
    b=floor(c+0.5);
    if (a<0) b=-b;
    y=224-b;

    /* Encendido del pixel adecuado  */
    putpixel(x,y, EGA_RED);
    coord_x=x;
    coord_y=y;
    return;
}
```

```
/* Funcion para generar las aceleraciones y tiempos, apartir de la secuencia a
   simular y que son requeridos para la generacion de trayectoria. Como entrada
   solo tiene a la secuencia a simular, como salida se tiene a t[], h[], q[][]
   y ddq[][] que son variables globales a nivel de simulaxx.c  */

int prev_gt (int numsec)
{
    int n, nodo, i, j, r, s, k;
    float p[10][6], mat_a[15][15], mat_b[15][6], inv_a[15][15], temp, divisor, temporal[6];
    char  err[80];

    /* Revision del tipo de coordenadas de todos los puntos */
    for (i=0; i<=9; i++){
        if (puntos[i][0]==0) {
            for (j=0; j<=5; j++) {
                p[i][j]=(PI/180)*puntos[i][j+1];
            }
        }
        else { /* Problema inverso */
            j=prob_inv(i,temporal);
            for(j=0; j<=5; j++) p[i][j]=temporal[j];
        }
    }

    /* Analisis de la secuencia */
    nodo=sec[numsec][0];
    for (j=0; j<=5; j++) {
        q[1][j]=p[nodo][j];
    }
    t[1]=0;
    t[2]=gen_tray[1].val*tiem_sec[numsec][0];
    h[1]=t[2];
    t[3]=tiem_sec[numsec][0];
    h[2]=t[3]-t[2];

    for (i=1; i<=12; i++) {
        nodo=sec[numsec][i];
        if(sec[numsec][i+1]>9) {
            if(sec[numsec][i+1]==10) {
                for (j=0; j<=5; j++) {
                    q[i+3][j]=p[nodo][j];
                }
                if(i==1) {
                    t[4]=tiem_sec[numsec][0];
                    t[3]=(gen_tray[2].val*h[2])+t[2];
                    h[2]=t[3]-t[2];
                    h[3]=t[4]-t[3];
                }
                else {
                    h[i+1]=gen_tray[2].val*tiem_sec[numsec][i-1];
                    h[i+2]=tiem_sec[numsec][i-1]-h[i+1];
                    t[i+2]=h[i+1]+t[i+1];
                    t[i+3]=tiem_sec[numsec][i-1]+t[i+1];
                }
                n=i+3;
                ciclo=0;
            }
            else {
                if (sec[numsec][i+1]==20) {
                    for (j=0; j<=5; j++) {
                        q[i+2][j]=p[nodo][j];
                        q[i+4][j]=q[1][j];
                    }
                    h[i+2]=gen_tray[2].val*tiem_sec[numsec][i];
                    h[i+3]=(1-gen_tray[2].val)*tiem_sec[numsec][i];
                    t[i+3]=h[i+2]+t[i+2];
                    t[i+4]=h[i+3]+t[i+3];
                    n=i+4;
                    ciclo=1;
                }
            }
        }
```

```
        else {
            h[i+2]=tiem_sec[numsec][i];
            t[i+3]=h[i+2]+t[i+2];
            for (j=0; j<=5; j++) {
                q[i+2][j]=p[nodo][j];
            }
        }
    }
    /* Limpiar las matrices mat_a, mat_b, inv_a y ddq */
    for(i=1; i<=14; i++) {
        for(j=1; j<=14; j++) {
            mat_a[i][j]=0;
            inv_a[i][j]=0;
        }
    }
    for(i=1; i<=14; i++) {
        for(j=0; j<=5; j++) {
            ddq[i][j]=0;
            mat_b[i][j]=0;
        }
    }
    for(i=15; i<=16; i++) {
        for(j=0; j<=5; j++) {
            ddq[i][j]=0;
        }
    }


    /* Formar las matrices mat_a y mat_b */
    mat_a[1][1]=(3*h[1])+(2*h[2])+(pow(h[1],2)/h[2]);
    if(n==4) { /* Caso especial de trayectoria con 2 puntos definidos */
        mat_a[1][2]=h[2]-(pow(h[3],2)/h[2]);
        mat_a[2][1]=h[2]-(pow(h[1],2)/h[2]);
        mat_a[2][2]=(3*h[3])+(2*h[2])+(pow(h[3],2)/h[2]);
        for (i=0; i<=5; i++) {
            mat_b[1][i]=6*( (q[4][i]/h[2])+(q[1][i]/h[1])-((1/h[1]+1/h[2])*q[1][i]) );
            mat_b[2][i]=6*( (q[1][i]/h[2])+(q[4][i]/h[3])-((1/h[3]+1/h[2])*q[4][i]) );
        }
    }
    else {
        mat_a[1][2]=h[2];
        mat_a[2][1]=h[2]-(pow(h[1],2)/h[2]);
        mat_a[2][2]=2*(h[2]+h[3]);
        if(n==5) { /* Caso especial de trayectoria con 3 puntos (o ciclo de 2 puntos) */
            mat_a[2][3]=h[3]-(pow(h[4],2)/h[3]);
            mat_a[3][2]=h[3];
            mat_a[3][3]=(3*h[4])+(2*h[3])+(pow(h[4],2)/h[3]);
            for (i=0; i<=5; i++) {
                mat_b[1][i]=6*( (q[3][i]/h[2])+(q[1][i]/h[1])-((1/h[1]+1/h[2])*q[1][i]) );
                mat_b[2][i]=6*( (q[5][i]/h[3])-(q[3][i]*(1/h[3]+1/h[2]))+(q[1][i]/h[2]) );
                mat_b[3][i]=6*( (q[5][i]/h[4])+(q[3][i]/h[3])-((1/h[4]+1/h[3])*q[5][i]) );
            }
        }
        else { /* Caso general de trayectoria con 4 puntos (o ciclo de 3 puntos) */
            mat_a[2][3]=h[3];
            mat_a[n-3][n-4]=h[n-3];
            mat_a[n-3][n-3]=2*(h[n-2]+h[n-3]);
            mat_a[n-3][n-2]=h[n-2]-(pow(h[n-1],2)/h[n-2]);
            mat_a[n-2][n-3]=h[n-2];
            mat_a[n-2][n-2]=(3*h[n-1])+(2*h[n-2])+(pow(h[n-1],2)/h[n-2]);
            for(i=3; i<=n-4; i++) {
                mat_a[i][i-1]=h[i];
                mat_a[i][i]=2*(h[i]+h[i+1]);
                mat_a[i][i+1]=h[i+1];
            }
            for (i=0; i<=5; i++) {
                mat_b[1][i]=6*( (q[3][i]/h[2])+(q[1][i]/h[1])-((1/h[1]+1/h[2])*q[1][i]) );
                mat_b[2][i]=6*( (q[4][i]/h[3])-(q[3][i]*(1/h[3]+1/h[2]))+(q[1][i]/h[2]) );
                for(j=3; j<=n-4; j++) {
                    mat_b[j][i]=6*( ((q[j+2][i]-q[j+1][i])/h[j+1])-((q[j+1][i]-q[j][i])/h[j]) );
                }
                mat_b[n-3][i]=6*( (q[n][i]/h[n-2])-(q[n-2][i]*(1/h[n-3]+1/h[n-2]))+(q[n-3][i]/h[n-3]) );
```

```
            mat_b[n-2][i]=6*( (q[n-2][i]/h[n-2])+(q[n][i]/h[n-1])-((1/h[n-1]+1/h[n-2])*q[n][i]) );
        }
    }
}


/* Inversion de mat_a */
for(i=1; i<=n-2; i++) inv_a[i][i]=1;
for(i=1; i<=n-2; i++) {
    if(mat_a[i][i]!=0) divisor=mat_a[i][i];
    else { /* Cambio de renglo */
        for(j=i+1; j<=n-2; j++) {
            if(mat_a[j][i]!=0) {
                for(r=1; r<=n-2; r++) {
                    temp=mat_a[i][r];
                    mat_a[i][r]=mat_a[j][r];
                    mat_a[j][r]=temp;
                    temp=inv_a[i][r];
                    inv_a[i][r]=inv_a[j][r];
                    inv_a[j][r]=temp;
                }
                printf("\n CAMBIO DE RENGLON, DEL %d AL %d",i,j);
                printf("\n ENTER PARA CONTINUAR");
                gets(err);
                j=n;
            }
        }
        if(j==n) divisor=mat_a[i][i];
        else {
            printf("\n ERROR !!! mat_a  ES SINGULAR");
            printf("\n ENTER PARA CONTINUAR");
            gets(err);
        }
    }
    for(j=1; j<=n-2; j++) {
        mat_a[i][j]=mat_a[i][j]/divisor;
        inv_a[i][j]=inv_a[i][j]/divisor;
    }
    for(j=i+1; j<=n-2; j++) {
        if(mat_a[j][i]!=0) {
            divisor=mat_a[j][i];
            for(r=1; r<=n-2; r++) {
                mat_a[j][r]=mat_a[j][r]-(divisor*mat_a[i][r]);
                inv_a[j][r]=inv_a[j][r]-(divisor*inv_a[i][r]);
            }
        }
        else break;
    }
}
for(i=n-2; i>=2; i=i-1) {
    for(j=i-1; j>=1; j=j-1) {
        if(mat_a[j][i]!=0) {
            divisor=mat_a[j][i];
            for(r=1; r<=n-2; r++) {
                mat_a[j][r]=mat_a[j][r]-(divisor*mat_a[i][r]);
                inv_a[j][r]=inv_a[j][r]-(divisor*inv_a[i][r]);
            }
        }
        else break;
    }
}


/* Multiplicacion de (inv_a)*(mat_b)  */
for(i=1; i<=n-2; i++) {
   for(s=0; s<=5; s++) {
      for(j=1; j<=n-2; j++) {
         ddq[i+1][s]=(inv_a[i][j]*mat_b[j][s])+ddq[i+1][s];
      }
   }
}
for (j=0; j<=5; j++) {
```

```
        q[2][j] = q[1][j]+(ddq[2][j]*pow(h[1],2)/6);
        q[n-1][j] = q[n][j]+(ddq[n-1][j]*pow(h[n-1],2)/6);
    }
    /* Si se trata de un ciclo, en una version (simula04.c) se maneja el crear dos
    matrices a nivel de mat_b y ddq, para tomar el caso en el que se quiere que -
    despues del primer ciclo que tiene vel y ace cero, se pudiera tener una trayec-
    toria que una ciclo a ciclo con una velocidad y aceleracion distintas de cero
    pero se vio que los valores de liga entre ciclo y ciclo, son especificos a cada
    secuencia, por lo que cada secuencia tendria que tener almacenados los valores de
    liga a nivel de velocidad y aceleracion, para cada union, lo que hacia que fuera
    un aumento en la cantidad de datos a almacenar y con una mejora en el comporta-
    miento muy relativo, por lo que se elimino esa posibilidad y con solo generar una
    matriz ddq, con valores de aceleracion cero al inicio y fin, se tiene la capacidad
    de crear un ciclo, con requerimientos no tan altos de memoria y que aun sigue con
    un comportamiento bueno. */

    return(n);
}

/* Funcion para generar la trayectoria que sigue la secuencia a simular. La
trayectoria es la posicion, velocidad y aceleracion a nivel de cada union,
durante todo el tiempo que dure la trayectoria. Como entrada se tiene a k,
tiem_tot, n(numero puntos), ddq[][](aceleraciones en cada punto), t[], h[],
q[][], ciclo, etc. Como salida se tiene a pos_q[], vel_d[] y ace_d[]. */

void g_trayec(int n, int k, float tiem_tot)
{
    /* Declaraciones de var. locales */
    float tiempo,temp;
    int i, j;

    temp = k*cont_union[1].val;
    if(temp > t[n]) tiempo = temp-(tiem_tot/2);
    else tiempo = temp;
    for(i = 1; i <= n-1; i++) {
            if((t[i] <= tiempo)&&(tiempo <= t[i+1])) break;
    }
    if(i == n) i = i-1;
    for(j = 0; j <= 5; j++) {
        ace_d[j] = ((ddq[i][j]/h[i])*(t[i+1]-tiempo))+((ddq[i+1][j]/h[i])*(tiempo-t[i]));
        vel_d[j] = ((ddq[i+1][j]/(2*h[i]))*pow((tiempo-t[i]),2))-(pow((t[i+1]-tiempo),2)*ddq[i][j]/(2*h[i]))+((q[i+1][j]-q[i][j])/h[i])+
            ((ddq[i][j]-ddq[i+1][j])*(h[i]/6));
        pos_d[j] = (pow((t[i+1]-tiempo),3)*ddq[i][j]/(6*h[i]))+(pow((tiempo-t[i]),3)*ddq[i+1][j]/(6*h[i]))+((tiempo-t[i])*((q[i+1][j]/h[i])-
            (h[i]*ddq[i+1][j]/6)))+((t[i+1]-tiempo)*((q[i][j]/h[i])-(h[i]*ddq[i][j]/6)));
    }
    return;
}
```

```
/* Funcion para calcular el problema inverso de un punto dado. La funcion se
declara en menu principal, ya que tiene que tener acceso la funcion de grabar
modificar puntos (para comprobacion de la validez de un punto en coordenadas
cartesianas) y tambien se debe de accesar desde la funcion de rutinas previas
a generacion de trayectoria (prev_gt).
La entrada es el punto al que se le va a realizar el problema inverso (es decir
un entero de 0 a 9) y un vector de por lo menos dimension 6, para albergar el
resultado de los calculos. La salida, se tiene un codigo entero de confirmacion
que es:        -0 si no hubo problemas en la rutina.
                -1 si el centro de la muneca esta sobre el eje Z (x=y=0). Aunque
        es valido el tener puntos en esa zona, se crea una singularidad que se
        tendria que trabajar como caso aparte, por lo que se excluye el eje Z
        del espacio de trabajo. No causa gran problema y solo se tendria este
        caso, si d6 fuera comparable a d2, para que fisicamente se pudiera dar
        y ese no es el caso en el manipulador construido y en la generalidad de
        los manipuladores.
                -2 si (x,y) esta por fuera del espacio de trabajo.
                -3 si pese a que (x,y) esta en la zona de trabajo, la "z" especi-
        ficada se sale del espacio de trabajo, ya sea por estar o muy arriba o
        muy abajo. Se requiere que en cierta forma las "z's" esten cercanas al
        valor de d1, para asegurar en valores (x,y) cercanos a los limites del
        espacio de trabajo, que se pueda alcanzar el punto. */

int prob_inv (int i, float p[]);
{
    /* Declaracion de variables locales */
    int j, r, s, k, codo, brazo, muneca;
    float giro, elev, desv, px, py, pz, a, b, r30t[3][3], r60[3][3], r63[3][3];
    char err[80];

    giro=(PI/180)*puntos[i][6];
    elev=(PI/180)*puntos[i][5];
    desv=(PI/180)*puntos[i][4];
    brazo=puntos[i][7];
    codo=puntos[i][8];
    muneca=puntos[i][9];

    /* Desacoplo posicion/orientacion */
    px=puntos[i][1]-(cinematicos[5].val*((sin(giro)*sin(desv))+(cos(giro)*sin(elev)*cos(desv))));
    py=puntos[i][2]-(cinematicos[5].val*((sin(giro)*sin(elev)*cos(desv))-(cos(giro)*sin(desv))));
    pz=puntos[i][3]-(cinematicos[5].val*cos(elev)*cos(desv));
    if((pow(px,2)+pow(py,2))<pow(cinematicos[4].val,2)) return(1);

    /* Posicion inversa */
    b=sqrt(pow(px,2)+pow(py,2)-pow(cinematicos[4].val, 2));
    a=(pow(b,2)+pow((pz-cinematicos[3].val),2)-pow(cinematicos[1].val,2)-pow(cinematicos[2].val,2))/(2*cinematicos[1].val*cinematicos[2].val);
    if(b>(cinematicos[1].val+cinematicos[2].val)) return(2);
    if(a>1) return(3);
    p[0]=PI/2+atan2(py,px)-(brazo*atan2(b,cinematicos[4].val));
    p[2]=atan2((-codo*sqrt(1-pow(a,2))),a);
    p[1]=atan2((pz-cinematicos[3].val),b)-atan2((cinematicos[2].val*sin(p[2])),(cinematicos[1].val+(cinematicos[2].val*cos(p[2]))));

    /* Orientacion inversa */
    r30t[0][0]=cos(p[0])*cos((p[1]+p[2]));
    r30t[0][1]=sin(p[0])*cos((p[1]+p[2]));
    r30t[0][2]=sin((p[1]+p[2]));
    r30t[1][0]=-(cos(p[0])*sin((p[1]+p[2])));
    r30t[1][1]=-(sin(p[0])*sin((p[1]+p[2])));
    r30t[1][2]=cos((p[1]+p[2]));
    r30t[2][0]=sin(p[0]);
    r30t[2][1]=-cos(p[0]);
    r30t[2][2]=0;

    r60[0][0]=cos(giro)*cos(elev);
    r60[0][1]=(cos(giro)*sin(elev)*sin(desv))-(sin(giro)*cos(desv));
    r60[0][2]=(cos(giro)*sin(elev)*cos(desv))+(sin(giro)*sin(desv));
    r60[1][0]=sin(giro)*cos(elev);
    r60[1][1]=(sin(giro)*sin(elev)*sin(desv))+(cos(giro)*cos(desv));
    r60[1][2]=(sin(giro)*sin(elev)*cos(desv))-(cos(giro)*sin(desv));
    r60[2][0]=-sin(elev);
    r60[2][1]=cos(elev)*sin(desv);
    r60[2][2]=cos(elev)*cos(desv);
```

```
    for(j=0; j<=2; j++) {
        for(r=0; r<=2; r++) {
            r63[j][r]=0;
        }
    }
    /* Multiplicacion de r30t con r60 que da como resultado r63 */
    for (r=0; r<=2; r++) {
        for (s=0; s<=2; s++) {
            for (k=0; k<=2; k++) {
                r63[r][s]=r63[r][s]+(r30t[r][k]*r60[k][s]);
            }
        }
    }
    if (fabs(r63[0][2])<=0.000001 && fabs(r63[1][2])<=0.000001) {
        /* Caso singular */
        if (r63[2][2]>0) {
            p[5]=atan2(r63[1][0], r63[1][1]);
            p[4]=0;
        }
        else {
            p[5]=-atan2(r63[1][0], r63[1][1]);
            p[4]=PI;
        }
        p[3]=0;
    }
    else {
        /* Caso normal */
        p[4]=atan2((muneca*sqrt(1-pow(r63[2][2],2))), r63[2][2]);
        if (muneca==1) {
            p[3]=atan2(r63[1][2], r63[0][2]);
            p[5]=atan2(r63[2][1], -r63[2][0]);
        }
        else {
            p[3]=atan2(-r63[1][2], -r63[0][2]);
            p[5]=atan2(-r63[2][1], r63[2][0]);
        }
    }

    return(0);
}

/* Funcion que obtiene el problema directo, a partir de pos_d[]. La salida es
unicamente a pantalla. */
void prob_direc(void)
{
    /* Declaracion de variables locales */
    float rot[3][3], pos[3], pos_234, desv, elev, giro;
    char cadena[80];
    pos_234=pos_d[1]+pos_d[2]+pos_d[3]; /* suma de q2+q3+q4 */

    /* Obtencion de la posicion */
    pos[0]=(cos(pos_d[0])*((cos(pos_234)*sin(pos_d[4])*cinematicos[5].val)+(cinematicos[2].val*cos(pos_d[1]+pos_d[2]))+(cinematicos[1].val*
        cos(pos_d[1])))+(sin(pos_d[0])*((cos(pos_d[4])*cinematicos[5].val)+cinematicos[4].val));
    pos[1]=(sin(pos_d[0])*((cos(pos_234)*sin(pos_d[4])*cinematicos[5].val)+(cinematicos[2].val*cos(pos_d[1]+pos_d[2]))+(cinematicos[1].val*
        cos(pos_d[1])))-(cos(pos_d[0])*((cos(pos_d[4])*cinematicos[5].val)+cinematicos[4].val));
    pos[2]=(sin(pos_234)*sin(pos_d[4])*cinematicos[5].val)+(cinematicos[2].val*sin(pos_d[1]+pos_d[2]))+(cinematicos[1].val*sin(pos_d[1]))+cinematicos[3].val;

    /* Obtencion de la orientacion */
    rot[0][0]=(cos(pos_d[0])*((cos(pos_234)*cos(pos_d[4])*cos(pos_d[5]))-(sin(pos_234)*sin(pos_d[5]))))-(sin(pos_d[0])*sin(pos_d[4])*cos(pos_d[5]));
    rot[1][0]=(sin(pos_d[0])*((cos(pos_234)*cos(pos_d[4])*cos(pos_d[5]))-(sin(pos_234)*sin(pos_d[5]))))+(cos(pos_d[0])*sin(pos_d[4])*cos(pos_d[5]));
    rot[2][0]=(sin(pos_234)*cos(pos_d[4])*cos(pos_d[5]))+(cos(pos_234)*sin(pos_d[5]));

    rot[0][1]=-cos(pos_d[0])*((cos(pos_234)*cos(pos_d[4])*sin(pos_d[5]))+(sin(pos_234)*cos(pos_d[5])))+(sin(pos_d[0])*sin(pos_d[4])*sin(pos_d[5]));
    rot[1][1]=-sin(pos_d[0])*((cos(pos_234)*cos(pos_d[4])*sin(pos_d[5]))+(sin(pos_234)*cos(pos_d[5])))-(cos(pos_d[0])*sin(pos_d[4])*sin(pos_d[5]));
    rot[2][1]=(cos(pos_234)*cos(pos_d[5]))-(sin(pos_234)*cos(pos_d[4])*sin(pos_d[5]));

    rot[0][2]=(cos(pos_d[0])*cos(pos_234)*sin(pos_d[4]))+(sin(pos_d[0])*cos(pos_d[4]));
    rot[1][2]=(sin(pos_d[0])*cos(pos_234)*sin(pos_d[4]))-(cos(pos_d[0])*cos(pos_d[4]));
    rot[2][2]=sin(pos_234)*sin(pos_d[4]);

    /* Obtencion de la representacion desv, elev, giro */
    if (fabs(rot[0][0])<=0.000001 && fabs(rot[1][0])<=0.000001) {
```

```
        /* Caso singular */
        if (rot[2][0] > 0) {
            desv = (180/PI)*atan2(-rot[0][1], rot[1][1]);
            elev = -90;
        }
        else {
            desv = (180/PI)*atan2(rot[0][1], rot[1][1]);
            elev = 90;
        }
        giro = 0;
    }
    else {
        /* Caso normal */
            desv = (180/PI)*atan2(rot[2][1],rot[2][2]);
            giro = (180/PI)*atan2(rot[1][0],rot[0][0]);
        elev = (180/PI)*atan2(-rot[2][0],(sqrt(1-pow(rot[2][0],2))));
    }

    /* Mandar a pantalla */
    sprintf(cadena,"Xd = (%-+.3f, %-+.3f, %-+.3f, %-+.2f, %-+.2f, %-+.2f)",
    pos[0],pos[1],pos[2],desv,elev,giro);
    setviewport(2,454,399,464,1);
    clearviewport();
    setcolor(EGA_LIGHTCYAN);
    setusercharsize(1,1,1,1);
    settextjustify(LEFT_TEXT, BOTTOM_TEXT);
    outtextxy(8,9,cadena);

    return;
}

/* Funcion que obtiene el jacobiano del manipulador y tras la multiplicacion
con vel_d[], se tiene Xd'[]. La salida es unicamente a pantalla. */
void jacobiano(void)
{
    /* Declaracion de las variables locales */
    float jaco[6][6], pos_234, pos_23, dev_x[6] = {0};
    int i,j;
    char cadena[80];

    pos_234 = pos_d[1] + pos_d[2] + pos_d[3];
    pos_23 = pos_d[1] + pos_d[2];

    /* Llenado del jacobiano del manipulador */

    jaco[0][0] = (-sin(pos_d[0])*((cos(pos_234)*sin(pos_d[4])*cinematicos[5].val) + (cinematicos[2].val*
        cos(pos_23)) + (cinematicos[1].val*cos(pos_d[1])))) + (cos(pos_d[0])*((cos(pos_d[4])*cinematicos[5].val) + cinematicos[4].val));
    jaco[1][0] = (cos(pos_d[0])*((cos(pos_234)*sin(pos_d[4])*cinematicos[5].val) + (cinematicos[2].val*
        cos(pos_23)) + (cinematicos[1].val*cos(pos_d[1])))) + (sin(pos_d[0])*((cos(pos_d[4])*cinematicos[5].val) + cinematicos[4].val));
    jaco[2][0] = 0;
    jaco[3][0] = 0;
    jaco[4][0] = 0;
    jaco[5][0] = 1;

    jaco[0][1] = -cos(pos_d[0])*((sin(pos_234)*sin(pos_d[4])*cinematicos[5].val) + (cinematicos[2].val*sin(pos_23)) + (cinematicos[1].val*sin(pos_d[1])));
    jaco[1][1] = -sin(pos_d[0])*((sin(pos_234)*sin(pos_d[4])*cinematicos[5].val) + (cinematicos[2].val*sin(pos_23)) + (cinematicos[1].val*sin(pos_d[1])));
    jaco[2][1] = (cos(pos_234)*sin(pos_d[4])*cinematicos[5].val) + (cinematicos[2].val*cos(pos_23)) + (cinematicos[1].val*cos(pos_d[1]));
    jaco[3][1] = sin(pos_d[0]);
    jaco[4][1] = -cos(pos_d[0]);
    jaco[5][1] = 0;

    jaco[0][2] = -cos(pos_d[0])*((sin(pos_234)*sin(pos_d[4])*cinematicos[5].val) + (cinematicos[2].val*sin(pos_23)));
    jaco[1][2] = -sin(pos_d[0])*((sin(pos_234)*sin(pos_d[4])*cinematicos[5].val) + (cinematicos[2].val*sin(pos_23)));
    jaco[2][2] = (cos(pos_234)*sin(pos_d[4])*cinematicos[5].val) + (cinematicos[2].val*cos(pos_23));
    jaco[3][2] = sin(pos_d[0]);
    jaco[4][2] = -cos(pos_d[0]);
    jaco[5][2] = 0;

    jaco[0][3] = -cos(pos_d[0])*sin(pos_234)*sin(pos_d[4])*cinematicos[5].val;
    jaco[1][3] = -sin(pos_d[0])*sin(pos_234)*sin(pos_d[4])*cinematicos[5].val;
    jaco[2][3] = cos(pos_234)*sin(pos_d[4])*cinematicos[5].val;
    jaco[3][3] = sin(pos_d[0]);
```

```
    jaco[4][3] = -cos(pos_d[0]);
    jaco[5][3] = 0;

    jaco[0][4] = (cos(pos_d[0])*cos(pos_234)*cos(pos_d[4])*cinematicos[5].val)-(sin(pos_d[0])*sin(pos_d[4])*cinematicos[5].val);
    jaco[1][4] = (sin(pos_d[0])*cos(pos_234)*cos(pos_d[4])*cinematicos[5].val)+(cos(pos_d[0])*sin(pos_d[4])*cinematicos[5].val);
    jaco[2][4] = sin(pos_234)*cos(pos_d[4])*cinematicos[5].val;
    jaco[3][4] = -cos(pos_d[0])*sin(pos_234);
    jaco[4][4] = -sin(pos_d[0])*sin(pos_234);
    jaco[5][4] = cos(pos_234);

    jaco[0][5] = 0;
    jaco[1][5] = 0;
    jaco[2][5] = 0;
    jaco[3][5] = (cos(pos_d[0])*cos(pos_234)*sin(pos_d[4]))+(sin(pos_d[0])*cos(pos_d[4]));
    jaco[4][5] = (sin(pos_d[0])*cos(pos_234)*sin(pos_d[4]))-(cos(pos_d[0])*cos(pos_d[4]));
    jaco[5][5] = sin(pos_234)*sin(pos_d[4]);

    /* Multiplicado del jacobiano por vel_d */
    for (i=0;i<=5;i++) {
        for (j=0;j<=5;j++) {
            dev_x[i]=dev_x[i]+(jaco[i][j]*vel_d[j]);
        }
    }

    /* Mandar a pantalla */
    sprintf(cadena,"Xd' =(%-+.3f, %-+.3f, %-+.3f, %-+.2f, %-+.2f, %-+.2f)",dev_x[0],dev_x[1],dev_x[2],
        (180/PI)*dev_x[3],(180/PI)*dev_x[4],(180/PI)*dev_x[5]);
    setviewport(2,467,399,477,1);
    clearviewport();
    setcolor(EGA_LIGHTCYAN);
    setusercharsize(1,1,1,1);
    settextjustify(LEFT_TEXT, BOTTOM_TEXT);
    outextxy(8,9,cadena);

return;
}
```

```
/* Programa con las rutinas referentes a la dinamica del sistema.
   Se utiliza el metodo de Newton-Euler, con los parametros dados
   por el modelo dinamico de los elementos (simplificado). Para
   evitar el crear todas las variables que son constantes de una
   iteracion a otra y bien se podrian inicializar en una rutina
   previa a dinamica (prev_dina), se realiza la inicializacion den-
   tro de la rutina dinamica, pese a que cada vez que se corra, se
   realice los calculos, que den los mismo resultados una y otra vez.
   Pero se tiene, que son muchos los resultados que se tendrian que
   comunicar de rutina a rutina, a traves de inic_op y para simplificar
   la implementacion se elmina la rutina prev_dina y solo se tiene a
   la rutina "dinamica" que a su vez requiere de una rutina auxiliar
   para el calculo del producto cruz (prod_cr).
   Primera version 30/oct/95
   Ultima revision 11/nov/95  */

#include <math.h>
#define PI 3.14159265

/* Variables globales */
struct parametros {
       char nombre[6];
       float val;
};
extern struct parametros cinematicos[41];
extern struct parametros dinamicos[81];
extern struct parametros cont_union[81];
extern struct parametros gen_tray[41];
extern float puntos[10][10], tiem_sec[10][14];
extern int sec[10][14];

float dinamica(int uniones, float pos_d[], float vel_d[], float ace_d[])
{
    /* Declaracion de variables locales */
    float masa[6], iner[3][3][6]={0}, ri_ci[3][6], ri1_ci[3][6], ri_i1[3][6];
    float mri_0t[3][3][6], mri_i_1t[3][3][6], mri1_i[3][3][6], zi_1[3][6], gi[3][6];
    float w_i[3][7], alfa_i[3][7], a_ci[3][7], a_ci[3][7], f_i[3][8], tau_i[3][8];
    float temp1[3], temp2[3], temp3[3], temp4[3], temp5[3], c1, c2, c3;
    float *ptr_c1, *ptr_c2, *ptr_c3;
    float c234, s234, magnitud;
    int i, j, k;

    /* Declaracion de funciones aux */
    void prod_cr(float, float, float, float, float, float, float *, float *, float *);

    /* Inicialización de variables */
    for (i=0; i<=2; i++) {
        f_i[i][7]=0;
        tau_i[i][7]=0;
        w_i[i][0]=0;
        alfa_i[i][0]=0;
        a_ci[i][0]=0;
    }
    ptr_c1=&c1;
    ptr_c2=&c2;
    ptr_c3=&c3;

    /* Inicializacion de matrices constantes */

    /* Masa de cada elemento */
    masa[0]=(1./3)*PI*dinamicos[4].val*pow(dinamicos[2].val,2)*dinamicos[3].val;
    for (i=1;i<=4;i++) {
        masa[i]=sqrt(pow(dinamicos[2+(i*3)].val,2)+pow(dinamicos[3+(i*3)].val,2))*
                ((dinamicos[4+(i*3)].val*PI*pow(dinamicos[2+(i*3)].val,2))/20);
    }
    masa[5]=dinamicos[17].val*dinamicos[18].val*dinamicos[19].val*dinamicos[20].val;

    /* Tensor de inercia de cada elemento */
    iner[0][0][0]=(pow(dinamicos[2].val,2)+(pow(dinamicos[3].val,2)/4))*(masa[0]*3./20);
    iner[1][1][0]=0.3*pow(dinamicos[2].val,2)*masa[0];
    iner[2][2][0]=iner[0][0][0];
```

```
for(i=1;i<=2;i++) {
    iner[0][0][i]=(masa[i]*pow(dinamicos[2+(i*3)].val,2))/2;
    iner[1][1][i]=((pow(dinamicos[2+(i*3)].val,2)*masa[i])/4)+((pow(dinamicos[3+(i*3)].val,2)*masa[i])/18);
    iner[2][2][i]=iner[1][1][i];
}
for(i=3;i<=4;i++) {
    iner[0][0][i]=((pow(dinamicos[2+(i*3)].val,2)*masa[i])/4)+((pow(dinamicos[3+(i*3)].val,2)*masa[i])/18);
    iner[1][1][i]=iner[0][0][i];
    iner[2][2][i]=(masa[i]*pow(dinamicos[2+(i*3)].val,2))/2;
}
iner[0][0][5]=((pow(dinamicos[18].val,2)+pow(dinamicos[19].val,2))*masa[5])/12;
iner[1][1][5]=((pow(dinamicos[17].val,2)+pow(dinamicos[19].val,2))*masa[5])/12;
iner[2][2][5]=((pow(dinamicos[17].val,2)+pow(dinamicos[18].val,2))*masa[5])/12;

/* Vectores de la union i a la union i+1, expresados en el marco de ref i */
ri_i1[0][0]=0;
ri_i1[1][0]=cinematicos[18].val;
ri_i1[2][0]=cinematicos[4].val;

ri_i1[0][1]=cinematicos[1].val;
ri_i1[1][1]=0;
ri_i1[2][1]=cinematicos[19].val;

ri_i1[0][2]=cinematicos[2].val;
ri_i1[1][2]=0;
ri_i1[2][2]=-cinematicos[19].val;

ri_i1[0][3]=0;
ri_i1[1][3]=0;
ri_i1[2][3]=0;

ri_i1[0][4]=0;
ri_i1[1][4]=0;
ri_i1[2][4]=cinematicos[20].val;

ri_i1[0][5]=0;
ri_i1[1][5]=0;
ri_i1[2][5]=0;

/* Vectores de la union i al centroide del elemento i */
ri_ci[0][0]=0;
ri_ci[1][0]=(dinamicos[3].val/4)+cinematicos[18].val;
ri_ci[2][0]=0;

ri_ci[0][1]=cinematicos[1].val-(dinamicos[6].val/3);
ri_ci[1][1]=0;
ri_ci[2][1]=0;

ri_ci[0][2]=cinematicos[2].val-(dinamicos[9].val/3);
ri_ci[1][2]=0;
ri_ci[2][2]=-cinematicos[19].val;

ri_ci[0][3]=0;
ri_ci[1][3]=0;
ri_ci[2][3]=0;

ri_ci[0][4]=0;
ri_ci[1][4]=0;
ri_ci[2][4]=(dinamicos[15].val*2)/3;

ri_ci[0][5]=0;
ri_ci[1][5]=0;
ri_ci[2][5]=cinematicos[5].val-cinematicos[20].val;

/* Vectores de la union i+1 al centroide del elemento i */
ri1_ci[0][0]=0;
ri1_ci[1][0]=(dinamicos[3].val/4);
ri1_ci[2][0]=-cinematicos[4].val;

ri1_ci[0][1]=-(dinamicos[6].val/3);
ri1_ci[1][1]=0;
ri1_ci[2][1]=-cinematicos[19].val;
```

```
ril_ci[0][2]=-(dinamicos[9].val/3);
ril_ci[1][2]=0;
ril_ci[2][2]=0;

ril_ci[0][3]=0;
ril_ci[1][3]=0;
ril_ci[2][3]=0;

ril_ci[0][4]=0;
ril_ci[1][4]=0;
ril_ci[2][4]=((dinamicos[15].val*2)/3)-cinematicos[20].val;

ril_ci[0][5]=0;
ril_ci[1][5]=0;
ril_ci[2][5]=0;

/* Inicializacion de matrices que depende de pos_d[] */
/* Matriz de rotacion "i" a "i-1" transpuesta */
mri_i_1t[0][0][0]=cos(pos_d[0]);
mri_i_1t[0][1][0]=sin(pos_d[0]);
mri_i_1t[0][2][0]=0;
mri_i_1t[1][0][0]=0;
mri_i_1t[1][1][0]=0;
mri_i_1t[1][2][0]=1;
mri_i_1t[2][0][0]=sin(pos_d[0]);
mri_i_1t[2][1][0]=-cos(pos_d[0]);
mri_i_1t[2][2][0]=0;

mri_i_1t[0][0][1]=cos(pos_d[1]);
mri_i_1t[0][1][1]=sin(pos_d[1]);
mri_i_1t[0][2][1]=0;
mri_i_1t[1][0][1]=-sin(pos_d[1]);
mri_i_1t[1][1][1]=cos(pos_d[1]);
mri_i_1t[1][2][1]=0;
mri_i_1t[2][0][1]=0;
mri_i_1t[2][1][1]=0;
mri_i_1t[2][2][1]=1;

mri_i_1t[0][0][2]=cos(pos_d[2]);
mri_i_1t[0][1][2]=sin(pos_d[2]);
mri_i_1t[0][2][2]=0;
mri_i_1t[1][0][2]=-sin(pos_d[2]);
mri_i_1t[1][1][2]=cos(pos_d[2]);
mri_i_1t[1][2][2]=0;
mri_i_1t[2][0][2]=0;
mri_i_1t[2][1][2]=0;
mri_i_1t[2][2][2]=1;

mri_i_1t[0][0][3]=cos(pos_d[3]);
mri_i_1t[0][1][3]=sin(pos_d[3]);
mri_i_1t[0][2][3]=0;
mri_i_1t[1][0][3]=0;
mri_i_1t[1][1][3]=0;
mri_i_1t[1][2][3]=-1;
mri_i_1t[2][0][3]=-sin(pos_d[3]);
mri_i_1t[2][1][3]=cos(pos_d[3]);
mri_i_1t[2][2][3]=0;

mri_i_1t[0][0][4]=cos(pos_d[4]);
mri_i_1t[0][1][4]=sin(pos_d[4]);
mri_i_1t[0][2][4]=0;
mri_i_1t[1][0][4]=0;
mri_i_1t[1][1][4]=0;
mri_i_1t[1][2][4]=1;
mri_i_1t[2][0][4]=sin(pos_d[4]);
mri_i_1t[2][1][4]=-cos(pos_d[4]);
mri_i_1t[2][2][4]=0;

mri_i_1t[0][0][5]=cos(pos_d[5]);
mri_i_1t[0][1][5]=sin(pos_d[5]);
mri_i_1t[0][2][5]=0;
mri_i_1t[1][0][5]=-sin(pos_d[5]);
```

```
mri_i_1t[1][1][5]=cos(pos_d[5]);
mri_i_1t[1][2][5]=0;
mri_i_1t[2][0][5]=0;
mri_i_1t[2][1][5]=0;
mri_i_1t[2][2][5]=1.

/* Matriz de rotacion de "i" a "0" transpuesta */
c234=cos(pos_d[1]+pos_d[2]+pos_d[3]);
s234=sin(pos_d[1]+pos_d[2]+pos_d[3]);

mri_0t[0][0][0]=cos(pos_d[0]);
mri_0t[0][1][0]=sin(pos_d[0]);
mri_0t[0][2][0]=0;
mri_0t[1][0][0]=0;
mri_0t[1][1][0]=0;
mri_0t[1][2][0]=1;
mri_0t[2][0][0]=sin(pos_d[0]);
mri_0t[2][1][0]=-cos(pos_d[0]);
mri_0t[2][2][0]=0;

mri_0t[0][0][1]=cos(pos_d[0])*cos(pos_d[1]);
mri_0t[0][1][1]=sin(pos_d[0])*cos(pos_d[1]);
mri_0t[0][2][1]=sin(pos_d[1]);
mri_0t[1][0][1]=-cos(pos_d[0])*sin(pos_d[1]);
mri_0t[1][1][1]=-sin(pos_d[0])*sin(pos_d[1]);
mri_0t[1][2][1]=cos(pos_d[1]);
mri_0t[2][0][1]=sin(pos_d[0]);
mri_0t[2][1][1]=-cos(pos_d[0]);
mri_0t[2][2][1]=0;

mri_0t[0][0][2]=cos(pos_d[0])*cos(pos_d[1]+pos_d[2]);
mri_0t[0][1][2]=sin(pos_d[0])*cos(pos_d[1]+pos_d[2]);
mri_0t[0][2][2]=sin(pos_d[1]+pos_d[2]);
mri_0t[1][0][2]=-cos(pos_d[0])*sin(pos_d[1]+pos_d[2]);
mri_0t[1][1][2]=-sin(pos_d[0])*sin(pos_d[1]+pos_d[2]);
mri_0t[1][2][2]=cos(pos_d[1]+pos_d[2]);
mri_0t[2][0][2]=sin(pos_d[0]);
mri_0t[2][1][2]=-cos(pos_d[0]);
mri_0t[2][2][2]=0;

mri_0t[0][0][3]=cos(pos_d[0])*c234;
mri_0t[0][1][3]=sin(pos_d[0])*c234;
mri_0t[0][2][3]=s234;
mri_0t[1][0][3]=-sin(pos_d[0]);
mri_0t[1][1][3]=cos(pos_d[0]);
mri_0t[1][2][3]=0;
mri_0t[2][0][3]=-cos(pos_d[0])*s234;
mri_0t[2][1][3]=-sin(pos_d[0])*s234;
mri_0t[2][2][3]=c234;

mri_0t[0][0][4]=(cos(pos_d[0])*c234*cos(pos_d[4]))-(sin(pos_d[0])*sin(pos_d[4]));
mri_0t[0][1][4]=(sin(pos_d[0])*c234*cos(pos_d[4]))+(cos(pos_d[0])*sin(pos_d[4]));
mri_0t[0][2][4]=s234*cos(pos_d[4]);
mri_0t[1][0][4]=-cos(pos_d[0])*s234;
mri_0t[1][1][4]=-sin(pos_d[0])*s234;
mri_0t[1][2][4]=c234;
mri_0t[2][0][4]=(cos(pos_d[0])*c234*sin(pos_d[4]))+(sin(pos_d[0])*cos(pos_d[4]));
mri_0t[2][1][4]=(sin(pos_d[0])*c234*sin(pos_d[4]))-(cos(pos_d[0])*cos(pos_d[4]));
mri_0t[2][2][4]=s234*sin(pos_d[4]);

mri_0t[0][0][5]=(cos(pos_d[0])*((c234*cos(pos_d[4])*cos(pos_d[5]))-(s234*sin(pos_d[5]))))-(sin(pos_d[0])*sin(pos_d[4])*cos(pos_d[5]));
mri_0t[0][1][5]=(sin(pos_d[0])*((c234*cos(pos_d[4])*cos(pos_d[5]))-(s234*sin(pos_d[5]))))+(cos(pos_d[0])*sin(pos_d[4])*cos(pos_d[5]));
mri_0t[0][2][5]=(c234*sin(pos_d[5]))+(s234*cos(pos_d[4])*cos(pos_d[5]));
mri_0t[1][0][5]=-(cos(pos_d[0])*((c234*cos(pos_d[4])*sin(pos_d[5]))+(s234*cos(pos_d[5]))))+(sin(pos_d[0])*sin(pos_d[4])*sin(pos_d[5]));
mri_0t[1][1][5]=-(sin(pos_d[0])*((c234*cos(pos_d[4])*sin(pos_d[5]))+(s234*cos(pos_d[5]))))-(cos(pos_d[0])*sin(pos_d[4])*sin(pos_d[5]));
mri_0t[1][2][5]=(c234*cos(pos_d[5]))-(s234*cos(pos_d[4])*sin(pos_d[5]));
mri_0t[2][0][5]=(cos(pos_d[0])*c234*sin(pos_d[4]))+(sin(pos_d[0])*cos(pos_d[4]));
mri_0t[2][1][5]=(sin(pos_d[0])*c234*sin(pos_d[4]))-(cos(pos_d[0])*cos(pos_d[4]));
mri_0t[2][2][5]=s234*sin(pos_d[4]);

/* Matriz de rotacion de "i+1" a "i" */
mri1_i[0][0][0]=cos(pos_d[1]);
```

```
mri1_i[0][1][0]=-sin(pos_d[1]);
mri1_i[0][2][0]=0;
mri1_i[1][0][0]=sin(pos_d[1]);
mri1_i[1][1][0]=cos(pos_d[1]);
mri1_i[1][2][0]=0;
mri1_i[2][0][0]=0;
mri1_i[2][1][0]=0;
mri1_i[2][2][0]=1;

mri1_i[0][0][1]=cos(pos_d[2]);
mri1_i[0][1][1]=-sin(pos_d[2]);
mri1_i[0][2][1]=0;
mri1_i[1][0][1]=sin(pos_d[2]);
mri1_i[1][1][1]=cos(pos_d[2]);
mri1_i[1][2][1]=0;
mri1_i[2][0][1]=0;
mri1_i[2][1][1]=0;
mri1_i[2][2][1]=1;

mri1_i[0][0][2]=cos(pos_d[3]);
mri1_i[0][1][2]=0;
mri1_i[0][2][2]=-sin(pos_d[3]);
mri1_i[1][0][2]=sin(pos_d[3]);
mri1_i[1][1][2]=0;
mri1_i[1][2][2]=cos(pos_d[3]);
mri1_i[2][0][2]=0;
mri1_i[2][1][2]=-1;
mri1_i[2][2][2]=0;

mri1_i[0][0][3]=cos(pos_d[4]);
mri1_i[0][1][3]=0;
mri1_i[0][2][3]=sin(pos_d[4]);
mri1_i[1][0][3]=sin(pos_d[4]);
mri1_i[1][1][3]=0;
mri1_i[1][2][3]=-cos(pos_d[4]);
mri1_i[2][0][3]=0;
mri1_i[2][1][3]=1;
mri1_i[2][2][3]=0;

mri1_i[0][0][4]=cos(pos_d[5]);
mri1_i[0][1][4]=-sin(pos_d[5]);
mri1_i[0][2][4]=0;
mri1_i[1][0][4]=sin(pos_d[5]);
mri1_i[1][1][4]=cos(pos_d[5]);
mri1_i[1][2][4]=0;
mri1_i[2][0][4]=0;
mri1_i[2][1][4]=0;
mri1_i[2][2][4]=1;

mri1_i[0][0][5]=0;
mri1_i[0][1][5]=0;
mri1_i[0][2][5]=0;
mri1_i[1][0][5]=0;
mri1_i[1][1][5]=0;
mri1_i[1][2][5]=0;
mri1_i[2][0][5]=0;
mri1_i[2][1][5]=0;
mri1_i[2][2][5]=0;

/* Vectores Zi-1 */
zi_1[0][0]=0;
zi_1[1][0]=0;
zi_1[2][0]=1;

zi_1[0][1]=sin(pos_d[0]);
zi_1[1][1]=-cos(pos_d[0]);
zi_1[2][1]=0;

zi_1[0][2]=sin(pos_d[0]);
zi_1[1][2]=-cos(pos_d[0]);
zi_1[2][2]=0;
```

```
zi_1[0][3]=sin(pos_d[0]);
zi_1[1][3]=-cos(pos_d[0]);
zi_1[2][3]=0;

zi_1[0][4]=-cos(pos_d[0])*s234;
zi_1[1][4]=-sin(pos_d[0])*s234;
zi_1[2][4]=c234;

zi_1[0][5]=(cos(pos_d[0])*c234*sin(pos_d[4]))+(sin(pos_d[0])*cos(pos_d[4]));
zi_1[1][5]=(sin(pos_d[0])*c234*sin(pos_d[4]))-(cos(pos_d[0])*cos(pos_d[4]));
zi_1[2][5]=s234*sin(pos_d[4]);

/* Vectores de la aceleracion de la gravedad de cada elemento */
gi[0][0]=0;
gi[1][0]=-9.81;
gi[2][0]=0;

gi[0][1]=-9.81*sin(pos_d[1]);
gi[1][1]=-9.81*cos(pos_d[1]);
gi[2][1]=0;

gi[0][2]=-9.81*sin(pos_d[1]+pos_d[2]);
gi[1][2]=-9.81*cos(pos_d[1]+pos_d[2]);
gi[2][2]=0;

gi[0][3]=-9.81*s234;
gi[1][3]=0;
gi[2][3]=-9.81*c234;

gi[0][4]=-9.81*s234*cos(pos_d[4]);
gi[1][4]=-9.81*c234;
gi[2][4]=-9.81*s234*sin(pos_d[4]);

gi[0][5]=-9.81*((s234*cos(pos_d[4])*cos(pos_d[5]))+(c234*sin(pos_d[5])));
gi[1][5]=-9.81*((c234*cos(pos_d[5]))-(s234*cos(pos_d[4])*sin(pos_d[5])));
gi[2][5]=-9.81*s234*sin(pos_d[4]);

/* Recursion hacia adelante */
for (i=1;i<=6;i++) {
    /* Velocidad angular del centroide w_i */
    for (j=0;j<=2;j++) { /* Inicializar temp1 y temp2 */
        temp1[j]=0;
        temp2[j]=0;
    }
    for (j=0;j<=2;j++) {
        for (k=0;k<=2;k++) {
            temp1[j]=temp1[j]+(mri_i_1t[j][k]*w_i[k][i-1]);
            temp2[j]=temp2[j]+(mri_0t[j][k]*zi_1[k][i-1]);
        }
    }
    for (j=0;j<=2;j++) {
        w_i[j][i]=temp1[j]+(temp2[j]*vel_d[i-1]);
    }

    /* Aceleracion angular del centroide i */
    for (j=0;j<=2;j++) { /* Inicializar temp1 y temp3 */
        temp1[j]=0;
        temp3[j]=0;
    }
    for (j=0;j<=2;j++) {
        for (k=0;k<=2;k++) {
            temp1[j]=temp1[j]+(mri_i_1t[j][k]*alfa_i[k][i-1]);
        }
    }
    prod_cr(w_i[0][i], w_i[1][i], w_i[2][i], (temp2[0]*vel_d[i-1]), (temp2[1]*vel_d[i-1]), (temp2[2]*vel_d[i-1]), ptr_c1, ptr_c2, ptr_c3);
    temp3[0]=*ptr_c1;
    temp3[1]=*ptr_c2;
    temp3[2]=*ptr_c3;
    for (j=0;j<=2;j++) {
        alfa_i[j][i]=temp1[j]+(temp2[j]*ace_d[i-1])+temp3[j];
    }
```

```
    /* Aceleracion lineal del extremo del elemento i */
    for (j=0;j<=2;j++) { /* Inicializar temp1, temp2 y temp3 */
        temp1[j]=0;
        temp2[j]=0;
        temp3[j]=0;
    }
    prod_cr(alfa_i[0][i], alfa_i[1][i], alfa_i[2][i], ri_i1[0][i-1], ri_i1[1][i-1], ri_i1[2][i-1], ptr_c1, ptr_c2, ptr_c3);
    temp1[0]=*ptr_c1;
    temp1[1]=*ptr_c2;
    temp1[2]=*ptr_c3;
    prod_cr(w_i[0][i], w_i[1][i], w_i[2][i], ri_i1[0][i-1], ri_i1[1][i-1], ri_i1[2][i-1], ptr_c1, ptr_c2, ptr_c3);
    prod_cr(w_i[0][i], w_i[1][i], w_i[2][i], *ptr_c1, *ptr_c2, *ptr_c3, ptr_c1, ptr_c2, ptr_c3);
    temp2[0]=*ptr_c1;
    temp2[1]=*ptr_c2;
    temp2[2]=*ptr_c3;
    for (j=0;j<=2;j++) {
        for (k=0;k<=2;k++) {
            temp3[j]=temp3[j]+(mri_i_1t[j][k][i-1]*a_ei[k][i-1]);
        }
    }
    for (j=0;j<=2;j++) {
        a_ei[j][i]=temp1[j]+temp2[j]+temp3[j];
    }

    /* Aceleracion lineal del centroide del elemento i */
    for (j=0;j<=2;j++) { /* Inicializar temp1 y temp2 */
        temp1[j]=0;
        temp2[j]=0;
    }
    prod_cr(alfa_i[0][i], alfa_i[1][i], alfa_i[2][i], ri_ci[0][i-1], ri_ci[1][i-1], ri_ci[2][i-1], ptr_c1, ptr_c2, ptr_c3);
    temp1[0]=*ptr_c1;
    temp1[1]=*ptr_c2;
    temp1[2]=*ptr_c3;
    prod_cr(w_i[0][i], w_i[1][i], w_i[2][i], ri_ci[0][i-1], ri_ci[1][i-1], ri_ci[2][i-1], ptr_c1, ptr_c2, ptr_c3);
    prod_cr(w_i[0][i], w_i[1][i], w_i[2][i], *ptr_c1, *ptr_c2, *ptr_c3, ptr_c1, ptr_c2, ptr_c3);
    temp2[0]=*ptr_c1;
    temp2[1]=*ptr_c2;
    temp2[2]=*ptr_c3;
    for (j=0;j<=2;j++) {
        a_ci[j][i]=temp1[j]+temp2[j]+temp3[j];
    }
} /* Fin recursion hacia adelante */


/* Recursion hacia atras */
for (i=6; i>=1; i=i-1) {
    /* Fuerza aplicada en cada union */
    for (j=0;j<=2;j++) { /* Inicializar temp1, temp2 y temp3 */
        temp1[j]=0;
        temp2[j]=0;
        temp3[j]=0;
    }
    for (j=0;j<=2;j++) {
        for (k=0;k<=2;k++) {
            temp1[j]=temp1[j]+(mri1_i[j][k][i-1]*f_i[k][i+1]);
        }
    }
    for (j=0;j<=2;j++) {
        temp2[j]=masa[i-1]*a_ci[j][i];
        temp3[j]=masa[i-1]*gi[j][i-1];
    }
    for (j=0;j<=2;j++) {
        f_i[j][i]=temp1[j]+temp2[j]-temp3[j];
    }

    /* Par generado en cada union */
    for (j=0;j<=2;j++) { /* Inicializar temp1, temp2, temp3, temp4 y temp5 */
        temp1[j]=0;
        temp2[j]=0;
        temp3[j]=0;
        temp4[j]=0;
        temp5[j]=0;
```

```
        }
        for (j=0;j<=2;j++) {
            for (k=0;k<=2;k++) {
                temp1[j]=temp1[j]+(mri1_i[j][k][i-1]*tau_i[k][i+1]);
            }
        }

        prod_cr(f_i[0][i], f_i[1][i], f_i[2][i], ri_ci[0][i-1], ri_ci[1][i-1], ri_ci[2][i-1], ptr_c1, ptr_c2, ptr_c3);
        temp2[0]=*ptr_c1;
        temp2[1]=*ptr_c2;
        temp2[2]=*ptr_c3;

        for (j=0;j<=2;j++) {
            for (k=0;k<=2;k++) {
                temp3[j]=temp3[j]+(mri1_i[j][k][i-1]*f_i[k][i+1]);
            }
        }
        prod_cr(temp3[0], temp3[1], temp3[2], ri1_ci[0][i-1], ri1_ci[1][i-1], ri1_ci[2][i-1], ptr_c1, ptr_c2, ptr_c3);
        temp3[0]=*ptr_c1;
        temp3[1]=*ptr_c2;
        temp3[2]=*ptr_c3;

        for (j=0;j<=2;j++) {
            for (k=0;k<=2;k++) {
               temp4[j]=temp4[j]+(inerI[j][k][i-1]*alfa_i[k][i]);
                 temp5[j]=temp5[j]+(inerI[j][k][i-1]*w_i[k][i]);
            }
        }
        prod_cr(w_i[0][i], w_i[1][i], w_i[2][i], temp5[0], temp5[1], temp5[2], ptr_c1, ptr_c2, ptr_c3);
        temp5[0]=*ptr_c1;
        temp5[1]=*ptr_c2;
        temp5[2]=*ptr_c3;

        for (j=0;j<=2;j++) {
            tau_i[j][i]=temp1[j]-temp2[j]+temp3[j]+temp4[j]+temp5[j];
        }

    }


    /* Obtencion de la magnitud del par requerido */
    switch(uniones){
        case 1:
            magnitud=tau_i[1][1];
            break;
        case 2:
            magnitud=tau_i[2][2];
            break;
        case 3:
            magnitud=tau_i[2][3];
            break;
        case 4:
                magnitud=-tau_i[1][4];
            break;
        case 5:
            magnitud=tau_i[1][5];
            break;
        case 6:
            magnitud=tau_i[2][6];
    }

    return(magnitud);
}

/* Funcion para obtener el producto cruz de dos vectores. La entrada son seis
   valores y la salida se da en las localidades dadas por tres apuntadores */

void prod_cr(float a1, float a2, float a3, float b1, float b2, float b3, float *ptr_c1, float *ptr_c2, float *ptr_c3)
{
    *ptr_c1=(a2*b3)-(a3*b2);
    *ptr_c2=(a3*b1)-(a1*b3);
    *ptr_c3=(a1*b2)-(a2*b1);
```

```
    return;
}

/* Programa con las rutinas referentes al control.
   Primera version 27/sep/95  */

#include <math.h>
#define PI  3.14159265

/* Variables globales */
struct parametros {
       char nombre[6];
       float val;
};
extern struct parametros cinematicos[41];
extern struct parametros dinamicos[81];
extern struct parametros cont_union[81];
extern struct parametros gen_tray[41];
extern float puntos[10][10], tiem_sec[10][14];
extern int sec[10][14];

/* Funcion de rutinas previas al control. Se generan el contenido de las matrices
   en la ecuacion en diferencia, del algoritmo de control. Por facilidad, no
   se le da una estructura de matriz. */

void prev_cont(int uniones, float *ptr_g1, float *ptr_g2, float *ptr_h1, float *ptr_h2,
 float *ptr_hp1, float *ptr_hp2, float *ptr_k2_1, float *ptr_k2_2, float *ptr_k1, float *ptr_ke)
{
    /* Declaracion de variables locales */
    float beff, jeff, n, ka, kb, ra, alfa, beta, e_alfa, z, wn, gama, z_ob, divisor;
    float mat_1[3], inv_t[3][3], mat_2[3][3], mat_k[3]={0}, mat_3[3]={0};
    int i, j;
    char op[80];

    /* Obtencion de los parametros de control */
    ka=cont_union[28].val;
    kb=cont_union[29].val;
    ra=cont_union[30].val;
    n=cont_union[31+(3*(uniones-1))].val;
    jeff=cont_union[32+(3*(uniones-1))].val;
    beff=cont_union[33+(3*(uniones-1))].val;


    /* Obtencion de alfa, beta, e_alfa */
    alfa=(beff/jeff)+((ka*kb)/(ra*jeff));
    beta=(n*ka)/(ra*jeff);
    e_alfa=exp(-alfa*cont_union[1].val);

    /* Calculo de matrices control */
    *ptr_g1=(1-e_alfa)/alfa;
    *ptr_g2=e_alfa;
    *ptr_h1=(beta/alfa)*(cont_union[1].val+(-(1-e_alfa)/alfa));
    *ptr_h2=(beta/alfa)*(1-e_alfa);
    *ptr_hp1=(-pow(n,2)/(alfa*jeff))*((-(1-e_alfa)/alfa)+cont_union[1].val);
    *ptr_hp2=(-pow(n,2)/(alfa*jeff))*(1-e_alfa);

    /* Obtencion de polos control */
    z=cont_union[2+(4*(uniones-1))].val;
    wn=cont_union[3+(4*(uniones-1))].val;
    gama=cont_union[4+(4*(uniones-1))].val;
    z_ob=cont_union[5+(4*(uniones-1))].val;

    /* Calculo de las ganancias del sistema K2, K1 y Ke */
    mat_1[0]=1-exp(-z*wn*cont_union[1].val*(2+gama));
    mat_1[1]=(exp(-2*z*wn*cont_union[1].val)-e_alfa)+(2*exp(-z*wn*cont_union[1].val*(1+gama))*cos(wn*cont_union[1].val*sqrt(1-pow(z,2))));
    mat_1[2]=1+e_alfa-exp(-gama*z*wn*cont_union[1].val)-(2*exp(-z*wn*cont_union[1].val)*cos(wn*cont_union[1].val*sqrt(1-pow(z,2))));

    divisor=beta*cont_union[1].val*pow((1-e_alfa),2);
    inv_t[0][0]=(alfa*(1-e_alfa))/divisor;
    inv_t[1][0]=inv_t[0][0];
    inv_t[2][0]=inv_t[0][0];
```

```
inv_t[0][1]=(1-e_alfa-(alfa*cont_union[1].val))/divisor;
inv_t[1][1]=(1-e_alfa-(e_alfa*alfa*cont_union[1].val))/divisor;
inv_t[2][1]=(1-e_alfa-(alfa*cont_union[1].val*exp(-2*alfa*cont_union[1].val)))/divisor;
inv_t[0][2]=0;
inv_t[1][2]=0;
inv_t[2][2]=1;

mat_2[0][0]=-1;
mat_2[1][0]=1/cont_union[1].val;
mat_2[2][0]=alfa/(beta*cont_union[1].val);
mat_2[0][1]=0;
mat_2[1][1]=(1/(alfa*cont_union[1].val))-(1/(1-e_alfa));
mat_2[2][1]=1/(beta*cont_union[1].val);
mat_2[0][2]=1;
mat_2[1][2]=0;
mat_2[2][2]=0;

/* Operaciones de multiplicacion para obtener mat_k*/
for(i=0;i<=2;i++) {
    for(j=0;j<=2;j++) {
        mat_k[i]=mat_k[i]+(mat_1[j]*inv_t[j][i]);
    }
}

/* Operaciones de multiplicacion para obtener mat_3*/
mat_k[2]=mat_k[2]+1;
for(i=0;i<=2;i++) {
    for(j=0;j<=2;j++) {
        mat_3[i]=mat_3[i]+(mat_k[j]*mat_2[j][i]);
    }
}

/* Almacenado de los valores de las ganancias */
*ptr_k2_1=mat_3[0];
*ptr_k2_2=mat_3[1];
*ptr_k1=mat_3[2];
*ptr_ke=(alfa*(e_alfa-z_ob))/(1-e_alfa);

/* Codigo para probar resultados */
clrscr();
printf("Ra=%.4E\t Ka=%.4E\t Kb=%.4E\n",ra ,ka ,kb);
printf(" n=%.4E\tJeff=%.4E\tBeff=%.4E\tT=%.4E\n\n\n",n, jeff, beff,cont_union[1].val);
printf("\t ┌─\t      ─ \t ┌─\t   ─\t      ┌─\t      ─ \n");
printf("\t |  1  %.4E  |\t      |  %.4E  | \t    |  %.4E  | ", *ptr_g1, *ptr_h1, *ptr_hp1);
printf("\nx(k+1)=  | \t      |  x(k)  +  | \t    |  u(k)  +  | \t      |  w(k)\n");
printf("\t |  0  %.4E  |\t      |  %.4E  | \t    |  %.4E  | \n", *ptr_g2, *ptr_h2, *ptr_hp2);
printf("\t └─\t      ─┘ \t └─\t   ─┘\t      └─\t      ─┘ ");
printf("\n\ny(k)=[1  0] x(k)");
printf("\n\n\n\nu(k)= K1*v(k) - K2*x(k)");
printf("\n\nu(k)= [ %.4E ]*v(k) - [ %.4E  %.4E ]*x(k)", *ptr_k1, *ptr_k2_1, *ptr_k2_2);
printf("\n\n\nKe= [ %.4E ]", *ptr_ke);
gets(op);
clrscr();

if(op[0]=='1') {
    printf("\n[K2|K1]=([mat_k]+[0 0 1])*[mat_2]");
    printf("\n[mat_k]=[mat_1]*[T^-1]");
    printf("\n\nMatriz T^-1\n");
    for (i=0;i<=2;i++) {
        for(j=0;j<=2;j++) {
            printf("\t%.6f",inv_t[i][j]);
        }
        printf("\n");
    }
    printf("\nMatriz mat_1\n");
    for (i=0;i<=2;i++) {
        printf("\t%.6f",mat_1[i]);
    }
    printf("\n\nMatriz mat_k\n");
    for (i=0;i<=2;i++) {
        printf("\t%.6f",mat_k[i]);
    }
```

```
        printf("\n\nMatriz mat_2\n");
        for (i=0;i<=2;i++) {
            for(j=0;j<=2;j++) {
                printf("\t%.6f",mat_2[i][j]);
            }
            printf("\n");
        }
        printf("\nMatriz [K2|K1]=[ %f   %f   %f ]\n", *ptr_k2_1, *ptr_k2_2, *ptr_k1);
        gets();
        clrscr();
    }
    return;
}



/* Funcion de control. Se genera el algoritmo de control y por medio de las
   ganancias calculadas en prev_cont, se obtiene una serie de resultados que
   por medio de numvar, indica el valor regresado a inic_op */


float control (int k, float r, float w, float g1, float g2, float h1, float h2, float hp1, float hp2, float k2_1, float k2_2, float k1, float ke, int numvar, int uniones)
{
    /* Declaracion de variables */
    float x1, x2, y, v, e, u, error, error_ob;


    if (k==0) {
        x1=r;
        x2=0;
        y=r;
        v=(k2_1/k1)*r;
        e=0.1;
    }
    else {
        x1=cont_union[49].val;
        x2=cont_union[50].val;
        y=cont_union[49].val;
        v=cont_union[51].val+r;
        e=cont_union[52].val;
    }


    /* Operaciones para encontrar la compensacion anticipativa */
    /* cont_union[56].val lleva la cuenta de los periodos de activacion de la
    compensacion anticipativa, similar a k_din de la version anterior. Se ini-
    cializa en inic_op, antes del ciclo de simulacion  */

    if((k*cont_union[1].val)>=(cont_union[56].val*dinamicos[1].val)) {
        cont_union[56].val=cont_union[56].val-1;
        cont_union[55].val=(cont_union[30].val*w*cont_union[54].val*cont_union[28+(uniones*3)].val)/cont_union[28].val;
    }


    /* Operaciones para la señal de control (u) y los errores */
    u=(k1*v)-((k2_1*x1)+(k2_2*x2))+(k2_2*e)+cont_union[55].val;

    error=(r-y)*(180/PI);
    error_ob=e;
```

```
    /* Almacenado de valores para la proxima iteracion */
    cont_union[49].val=((1-(h1*k2_1))*x1)+((g1-(h1*k2_2))*x2)+(h1*k1*v)+(h1*k2_2*e)+(hp1*(w+cont_union[27].val));
    cont_union[50].val=(-h2*k2_1*x1)+((g2-(h2*k2_2))*x2)+(h2*k1*v)+(h2*k2_2*e)+(hp2*(w+cont_union[27].val));
    cont_union[51].val=(((h1*k2_1)-1)*x1)+(((h1*k2_2)-g1)*x2)+((1-(h1*k1))*v)-(h2*k2_2*e)-(hp1*(w+cont_union[27].val));
    cont_union[52].val=(g2-(ke*g1))*e;

    /* Seleccion del valor a ser regresado a inic_op */
    switch(numvar) {
        case 0: /* q */
            return((180/PI)*y);
        case 1: /* q' */
            return((180/PI)*x2);
        case 3: /* entrada u(k) */
            return(u);
        case 7: /* error */
            return(error);
        case 8: /* error del obervador */
            return(error_ob);
        case 9: /* velocidad observada */
            return((180/PI)*(x2-e));
        case 10: /* Energia dada en la entrada */
            if (k == 0) return(0);
                cont_union[53].val=cont_union[53].val+(pow(u,2)*cont_union[1].val);
                return(cont_union[53].val);
    }
}
```

Archivo de datos

```
xxxxx 0.000000 ___a2 0.130000 ___a3 0.150000 ___d1 0.300000
___d2 0.080000 ___d6 0.050000 li_s1 220.000000 li_i1 -40.000000
li_s2 270.000000 li_i2 -90.000000 li_s3 130.000000 li_i3 -130.000000
li_s4 60.000000 li_i4 -60.000000 li_s5 135.000000 li_i5 45.000000
li_s6 180.000000 li_i6 -180.000000 distA 0.070000 distB 0.050000
distC 0.015000 ____ = 0.000000 ____ = 0.000000 ____ = 0.000000
____= 0.000000 ____ = 0.000000 ____ = 0.000000 ____ = 0.000000
____= 0.000000 ____ = 0.000000 ____ = 0.000000 ____ = 0.000000
____= 0.000000 ____ = 0.000000 ____ = 0.000000 ____ = 0.000000
____= 0.000000 ____ = 0.000000 ____ = 0.000000 ____ = 0.000000
c40= 0.000000
xxxxx 0.000000 T_din 0.01000 ___r1 0.050000 ___h1 0.100000
__ro1 1600.000000 ___r2 0.040000 ___h2 0.200000 __ro2 2500.000000
___r3 0.050000 ___h3 0.250000 __ro3 1600.000000 ___r4 0.000000
___h4 0.000000 __ro4 0.000000 ___r5 0.025000 ___h5 0.050000
__ro5 5000.000000 ___a6 0.050000 ___b6 0.050000 ___l6 0.050000
__ro6 400.000000 ____ = 0.000000 ____ = 0.000000 ____= 0.000000
____= 0.000000 ____ = 0.000000 ____ = 0.000000 ____ = 0.000000
____= 0.000000 ____ = 0.000000 ____ = 0.000000 ____ = 0.000000
____= 0.000000 ____ = 0.000000 ____ = 0.000000 ____= 0.000000
____= 0.000000 ____ = 0.000000 ____ = 0.000000 ____ = 0.000000
____= 0.000000 ____ = 0.000000 ____ = 0.000000 ____ = 0.000000
____= 0.000000 ____ = 0.000000 ____ = 0.000000 ____= 0.000000
____= 0.000000 ____ = 0.000000 ____ = 0.000000 ____= 0.000000
____= 0.000000 ____ = 0.000000 ____ = 0.000000 ____= 0.000000
____= 0.000000 ____ = 0.000000 ____ = 0.000000 ____ = 0.000000
____= 0.000000 ____ = 0.000000 ____ = 0.000000 ____ = 0.000000
____= 0.000000 ____ = 0.000000 ____ = 0.000000 ____ = 0.000000
____= 0.000000 ____ = 0.000000 ____ = 0.000000 ____ = 0.000000
____= 0.000000 ____ = 0.000000 ____ = 0.000000 ____ = 0.000000
____= 0.000000 ____ = 0.000000 ____ = 0.000000 ____ = 0.000000
d80= 0.000000
xxxxx 0.000000 ___T= 0.01000 zeta1 0.500000 __wn1 20.000000
gama1 3.000000 _Zob1 0.00010 zeta2 0.500000 __wn2 20.000000
gama2 3.000000 _Zob2 0.001000 zeta3 0.500000 __wn3 20.000000
gama3 3.000000 _Zob3 0.001000 zeta4 1.000000 __wn4 5.000000
gama4 3.000000 _Zob4 0.001000 zeta5 1.000000 __wn5 5.000000
gama5 3.000000 _Zob5 0.001000 zeta6 0.600000 __wn6 25.000000
gama6 3.000000 _Zob6 0.001000 1/0_d 1.000000 p_cte 0.000000
___Ka 0.106000 ___Kb 0.021010 ___Ra 26.000000 ___n1 0.050000
Jeff1 0.000050 Beff1 0.000040 ___n2 0.020000 Jeff2 0.000030
Beff2 0.000040 ___n3 0.066700 Jeff3 0.000025 Beff3 0.000035
___n4 0.200000 Jeff4 0.000020 Beff4 0.000030 ___n5 0.200000
Jeff5 0.000020 Beff5 0.000030 ___n6 0.200000 Jeff6 0.000020
Beff6 0.000030 Parm1 2.094414 Parm2 0.000988 Parm3 15.442038
Parm4 0.000000 int_u 0.000000 1/0_c 1.000000 Tcomp -0.000000
k_din 701.000000 ____ = 0.000000 ____ = 0.000000 ____= 0.000000
____= 0.000000 ____ = 0.000000 ____ = 0.000000 ____ = 0.000000
____= 0.000000 ____ = 0.000000 ____ = 0.000000 ____ = 0.000000
____= 0.000000 ____ = 0.000000 ____ = 0.000000 ____ = 0.000000
____= 0.000000 ____ = 0.000000 ____ = 0.000000 ____ = 0.000000
____= 0.000000 ____ = 0.000000 ____ = 0.000000 ____ = 0.000000
____= 0.000000
____= 0.000000 __ini 0.300000 __fin 0.500000 T__gt 0.01000
____= 0.000000 ____ = 0.000000 ____ = 0.000000 ____ = 0.000000
____= 0.000000 ____ = 0.000000 ____ = 0.000000 ____ = 0.000000
____= 0.000000 ____ = 0.000000 ____ = 0.000000 ____ = 0.000000
____= 0.000000 ____ = 0.000000 ____ = 0.000000 ____ = 0.000000
____= 0.000000 ____ = 0.000000 ____ = 0.000000 ____ = 0.000000
____= 0.000000 ____ = 0.000000 ____ = 0.000000 ____ = 0.000000
____= 0.000000 ____ = 0.000000 ____ = 0.000000 ____ = 0.000000
____= 0.000000 ____ = 0.000000 ____ = 0.000000 ____ = 0.000000
____= 0.000000 ____ = 0.000000 ____ = 0.000000 ____ = 0.000000
____= 0.000000
0.000000 0.000000 -45.000000 0.000000 0.000000 90.000000 0.000000
0.000000 0.000000 0.000000
0.000000 30.000000 10.000000 20.000000 10.000000 120.000000 0.000000
0.000000 0.000000 0.000000
0.000000 40.000000 80.000000 40.000000 40.000000 90.000000 0.000000
0.000000 0.000000 0.000000
```

0.000000 70.000000 85.000000 20.000000 50.000000 80.000000 0.000000
0.000000 0.000000 0.000000
0.000000 120.000000 90.000000 0.000000 10.000000 75.000000 0.000000
0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 90.000000 0.000000
0.000000 0.000000 0.000000
0.000000 0.000000 150.000000 0.000000 0.000000 90.000000 0.000000
0.000000 0.000000 0.000000
0.000000 95.000000 98.000000 10.975000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000
0.000000 120.000000 100.000000 11.600000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000
0.000000 40.000000 -20.000000 20.000000 -5.000000 75.000000 70.000000
0.000000 0.000000 0.000000
5 6 7 8 10 30 30 30 30 30 30 30 30
5 6 10 30 30 30 30 30 30 30 30 30 30
1 2 3 4 20 30 30 30 30 30 30 30 30
0 1 2 3 4 10 30 30 30 30 30 30 30
0 6 8 10 30 30 30 30 30 30 30 30
5 5 5 10 30 30 30 30 30 30 30 30
1 2 9 4 10 30 30 30 30 30 30 30
1 2 9 4 20 30 30 30 30 30 30 30
1 2 9 4 20 30 30 30 30 30 30 30
1 2 3 4 1 2 3 4 1 10 30 30 30
1.250000 0.700000 1.250000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
2.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.500000 0.500000 0.500000 1.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
1.000000 1.500000 2.000000 2.500000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
2.000000 2.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
1.000000 1.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
1.000000 0.400000 1.100000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
1.250000 0.500000 1.373000 2.500000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
1.000000 0.400000 1.100000 2.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.500000 0.500000 0.500000 1.000000 0.500000 0.500000 0.500000 1.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000

APENDICE H

Análisis del manipulador de dos GDL utilizado en las pruebas del algoritmo de control.

Este manipulador planar tiene su segunda unión actuada a distancia, es decir desde la primera unión y por tanto la orientación del segundo elemento no varia ante un movimiento de la primera unión. Es por este hecho que no se puede trabajar directamente la convención D-H[1] para la obtención de los marcos de referencia de los elementos, como se hizo en el capitulo 3 (análisis), y se tiene que realizar el análisis directamente.

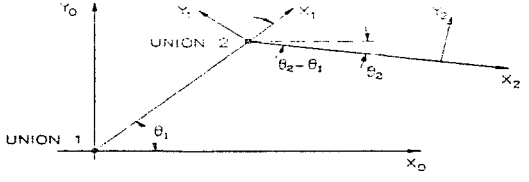


Figura h.1

En la figura h.1 se muestra los marcos de referencia utilizados en el análisis cinemático y es fácil obtener las transformaciones homogeneas entre los diferentes marcos:

$$A_1 = T_0^1 = \begin{bmatrix} \cos(\theta_1) & -\text{sen}(\theta_1) & 0 & l_1\cos(\theta_1) \\ \text{sen}(\theta_1) & \cos(\theta_1) & 0 & l_1\,\text{sen}(\theta_1) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \textbf{(h.1)}$$

[1] M.W. Spong y M. Vidyasagar, Robot Dynamics and Control, John Wilcy & Sons, 1989, pp 65-71

$$A_2 = T_0^2 = \begin{bmatrix} \cos(\theta_2 - \theta_1) & -\operatorname{sen}(\theta_2 - \theta_1) & 0 & l_2 \cos(\theta_2 - \theta_1) \\ \operatorname{sen}(\theta_2 - \theta_1) & \cos(\theta_2 - \theta_1) & 0 & l_2 \operatorname{sen}(\theta_2 - \theta_1) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \textbf{(h.2)}$$

$$T_0^2 = A_1 A_2 = \begin{bmatrix} \cos(\theta_2) & -\operatorname{sen}(\theta_2) & 0 & l_1 \cos(\theta_1) + l_2 \cos(\theta_2) \\ \operatorname{sen}(\theta_2) & \cos(\theta_2) & 0 & l_1 \operatorname{sen}(\theta_1) + l_2 \operatorname{sen}(\theta_2) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \textbf{(h.3)}$$

Donde: θ_1, θ_2 son la posición de la unión 1 y 2 respectivamente.

l_1, l_2 son la longitud del elemento 1 y del elemento 2 respectivamente.

De T_0^2 se puede observar que la orientación del actuador final sólo depende de la posición de la unión 2, por lo que la solución al problema directo tiene la forma:

$$f(\theta_1, \theta_2) = \begin{bmatrix} l_1 \cos(\theta_1) + l_2 \cos(\theta_2) \\ l_1 \operatorname{sen}(\theta_1) + l_2 \operatorname{sen}(\theta_2) \\ 0 \\ 0^\circ \\ 0^\circ \\ \theta_2 \end{bmatrix}$$; expresando la orientación en la forma "yaw-pitch-roll".

(h.4)

Problema cinemático inverso:

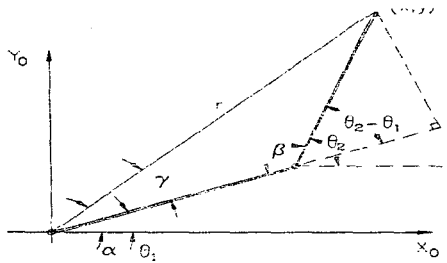


Figura h.2

De la figura h.2 se obtiene:

(h.5)

$$\alpha = \theta_1 + \gamma \Rightarrow \quad \theta_1 = \alpha - \gamma \quad ;\alpha = tang^{-1}\left(\frac{y}{x}\right)$$

$$\cos(\gamma) = \frac{l_1 + l_2\cos(\theta_2 - \theta_1)}{r} \quad ;r^2 = x^2 + y^2$$

$$\left\{\gamma = tang^{-1}\left(\frac{l_2\,\mathrm{sen}(\theta_2 - \theta_1)}{l_1 + l_2\cos(\theta_2 - \theta_1)}\right)\right.$$

$$\mathrm{sen}(\gamma) = \frac{l_2\,\mathrm{sen}(\theta_2 - \theta_1)}{r}$$

Por lo que se tiene que:

$$\theta_1 = tang^{-1}\left(\frac{y}{x}\right) - tang^{-1}\left(\frac{l_2\,\mathrm{sen}(\psi)}{l_1 + l_2\cos(\psi)}\right) \qquad \psi = \theta_2 - \theta_1 \qquad \textbf{(h.6)}$$

De la ley de los cosenos:

$$r^2 = l_1^2 + l_2^2 - 2l_1l_2\cos(\beta) \quad ;\beta = \pi - \psi \quad \Rightarrow r^2 = l_1^2 + l_2^2 + 2l_1l_2\cos(\psi) \qquad \textbf{(h.7)}$$

despejando se obtiene:

$$\cos(\psi) = \frac{r^2 - l_1^2 - l_2^2}{2l_1 l_2} = D \quad \text{y} \quad \text{sen}(\psi) = \pm\sqrt{1 - D^2} \tag{h.8}$$

$$\Rightarrow \psi = tang^{-1}\left(\frac{\text{sen}(\psi)}{\cos(\psi)}\right) = tang^{-1}\left(\frac{\pm\sqrt{1 - D^2}}{D}\right) \tag{h.9}$$

con (+) se tiene una configuración codo abajo y con (-) una de codo arriba.

Como se sabe que: $\psi = \theta_2 - \theta_1 \Rightarrow \theta_2 = \psi - \theta_2$

Con lo que se obtiene la posicion de las dos uniones dada la posición del actuador final (x,y) y se debe de cumplir que el actuador final esté dentro del espacio de trabajo, es decir que:

$$\begin{aligned} &y \geq -0.045 \\ &x^2 + y^2 \leq (l_1 + l_2)^2 \\ &x^2 + y^2 \geq (l_1^2 + l_2^2 + 2l_1 l_2 \cos(\theta_2 - \theta_1)_{min}) \end{aligned} \tag{h.10}$$

Y los resultados deben de estar en el rango de operación, para cualquiera de los dos soluciones (codo arriba / codo abajo):

$$0° \leq \theta_1 \leq 180°$$
$$-150° \leq (\theta_2 - \theta_1) \leq 120°$$

Si no hubiera alguna solución con $\theta_1 y \theta_2$ dentro del rango, entonces se está en las zonas no alcanzables por las limitaciones de movimiento de las uniones.

Obtención de la matriz jacobiana.

No se puede obtener por los métodos "estándar" , ya que no se utilizó la convención D-H.

Por lo que se tiene que hacer directamente[2]

$P_0 = R_0^2 p + d_0^2$;si P_0 es el origen del marco 2, p=0

$P_0 = d_0^2$

[2] M.W. Spong & M. Vidyasagar *op. cit.* pp148-150.

Para la velocidad del marco 2 se tiene

$$v_0^2 = \dot{P}_0 = \dot{d}_0^2 = \frac{d}{dt}\begin{bmatrix} a_1C_1 + a_2C_2 \\ a_1S_1 + a_2S_2 \\ 0 \end{bmatrix} = \begin{bmatrix} -a_1S_1\dot{\theta}_1 - a_2S_2\dot{\theta}_2 \\ a_1C_1\dot{\theta}_1 + a_2C_2\dot{\theta}_2 \\ 0 \end{bmatrix} \quad \textbf{(h.11)}$$

$$v_0^2 = \begin{bmatrix} -a_1S_1 & -a_2S_2 \\ a_1C_1 & a_2C_2 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix} \qquad \omega_0^2 = \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_2 \end{bmatrix} \quad \textbf{(h.12)}$$

que es la velocidad del actuador final en función de las velocidades de las uniones, es decir las partes de la matriz jacobiana.

Análisis dinámico.

Antes de realizar el análisis dinamico del manipulador se verá las caracteristicas de los actuadores/sensores, para despues juntar la dinamica de los actuadores con la del sistema mecanico y llegar a un modelo del manipulador.

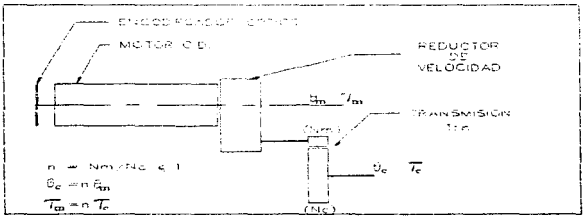


Figura h.3

Actuadores:

Cada elemento del manipulador se mueve por medio de un motor de CD acoplado a una transmisión de potencia como se muestra en la figura h.3 El motor de corriente directa, de imán permanente, tiene

un voltaje máximo de ±20 volts y no hay datos del fabricante pero, si una caracterización de sus parámetros y se da como dato:

B_m=1e-6	[N·m·s/rad]	;amortiguamiento en motor y transmisión.
J_m=32e-7	[Kg·m^2]	;inercia motor.
K_m=33e-3	[N·m/A]	;constante de par.
K_b=33.44e-3	[V·s]	;constante de la fuerza contraelectromotriz.
R_a=4.5	[Ω]	;resistencia de los devanados del motor.

Si se desprecia la inductancia del motor se llega a un modelo del par entregado a la carga en función del voltaje de alimentación del motor:

$$\tau_m = K_m \cdot i_a(t) = \frac{K_m}{R_a}\left[u(t) - K_b\dot{\theta}_m(t)\right] \quad \textbf{(h.13)}$$

Donde τ_m es el par generado por el motor.

Para la parte mecánica se tiene:

$$J_m\ddot{\theta}_m + B_m\dot{\theta}_m = \tau_m - r\tau_c \quad \textbf{(h.14)}$$

Donde: $r\tau_c$ es el par de carga reflejado al lado del motor.

θ_m es la posición de la unión del lado del motor.

Por lo que uniendo las dos partes se tiene:

$$J_m\ddot{\theta}_m + \left(B_m + \frac{K_b K_m}{R_a}\right)\dot{\theta}_m = \frac{K_m}{R_a}u(t) - r\tau_c \quad \textbf{(h.15)}$$

y el par de carga está dado por el modelo del sistema mecánico.

El término "r" es la relación de la transmisión de potencia y está dada por:

-Cada motor cuenta con un reductor de velocidad que tiene una relación[3] de:

1:0.01512 = 1:29/1918

[3] Una relación 1:n se toma como: $\theta_c = n\theta_m$ $\theta_c = posicion\_carga$
$\tau_m = n\tau_c$ $\theta_m = posicion\_motor$

-A parte cada motor está conectado a una transmisión de potencia por lo que se tiene que la relación total es:

Elemento 1 (hombro):

$$r_1 = \left(\frac{N_m}{N_c}\right)\left(\frac{29}{1918}\right) = \left(\frac{9}{30}\right)\left(\frac{29}{1918}\right) = \frac{37}{8157} = 0.00454 \qquad \textbf{(h.16)}$$

Elemento 2 (codo):

$$r_2 = \left(\frac{N_m}{N_c}\right)\left(\frac{29}{1918}\right) = \left(\frac{9}{25}\right)\left(\frac{29}{1918}\right) = \frac{60}{11023} = 0.00544 \qquad \textbf{(h.17)}$$

Sensores:

Aunque no se está tomando en cuenta la dinámica de los sensores se tiene que listar sus características para obtener los factores de conversión entre la lectura de los encodi-ficadores ópticos y la posición en radianes de cada unión.

El sensor incorporado al motor es un "encoder" incremental que genera dos señales "A" y "B", las cuales están defasadas 90° y se tiene una resolución de 6 pulsos por revolución del lado del motor. La tarjeta dSPACE realiza la adquisición[4] de las señales "A/B" para cada uno de los motores de forma que se genera una cuenta en un registro interno (de 24 bits) la cual está normalizada a ±1; por lo que para obtener la posición en radianes de cada unión (lado del elemento) se tiene que leer el registro y multiplicarlo por un factor:

$$\theta_1 = r_1\theta_{m1} \quad ; \theta_{m1} = (registro[cuenta])\left(\frac{2^{23}[pulsos]}{registro[cuenta]}\right)\left(\frac{1[rev]}{4 \cdot 6[pulsos]}\right)\left(\frac{2\pi[rad]}{1[rev]}\right) \qquad \textbf{(h.18)}$$

[4] Ver DS1102 User's Guide

y para la unión 2 se tiene algo similar por lo que los factores para cada unión son:

$$\theta_1 = (9961.615820)(registro)[rad]$$
$$\theta_2 = (11953.90967)(registro)[rad] \quad \textbf{(h.19)}$$

Modelo mecánico: Por medio de las ecuaciones de Euler-Lagrange, se obtiene el modelo del manipulador y por ser éste muy sencillo y por no poder aplicar la convención D-H, se tiene que hacer directamente[5] y se llega a un modelo del sistema mecánico que tiene la forma:

$$D(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = \tau$$

Donde:
$$D(q) = \begin{bmatrix} m_1 l_{c1}^2 + m_2 l_1^2 + I_1 & m_2 l_1 l_{c2} \cos(q_2 - q_1) \\ m_2 l_1 l_{c2} \cos(q_2 - q_1) & m_2 l_{c2}^2 + I_2 \end{bmatrix} \quad \textbf{(h.20)}$$

$$C(q,\dot{q}) = \begin{bmatrix} 0 & -m_2 l_1 l_{c2} \operatorname{sen}(q_2 - q_1)\dot{q}_2 \\ m_2 l_1 l_{c2} \operatorname{sen}(q_2 - q_1)\dot{q}_1 & 0 \end{bmatrix} \quad \textbf{(h.21)}$$

$$g(q) = \begin{bmatrix} (m_1 l_{c1} + m_2 l_1) g \cos(q_1) \\ m_2 l_{c2} g \cos(q_2) \end{bmatrix} \quad \textbf{(h.22)}$$

$$q = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} \quad \text{;dadas en la figura 3} \quad \textbf{(h.23)}$$

τ son los pares generalizados a nivel de cada unión.

l_{c1} y l_{c2} son la posición del centroide en el elemento 1 y 2 respectivamente.

En la matriz de inercia (D), se tiene los términos I_1 y I_2 que son el momento de inercia, con respecto al eje "Z", de los elementos 1 y 2 respectivamente. Considerando a cada elemento como

[5] M.W. Spong & M. Vidyasagar *ibid*

un prisma rectangular homogéneo se obtiene un momento de inercia simplificado que está dado por:

$$I_{zz1} = \frac{1}{12} m_1 (l_1^2 + (ancho)^2)$$
$$I_{zz2} = \frac{1}{12} m_2 (l_2^2 + (ancho)^2)$$

;ancho=0.04 m (ancho de los elementos) **(h.24)**

Para completar el modelo del manipulador se considera que la carga a los actuadores está dada por el modelo mecánico y juntando ecuaciones se llega[6] a:

$$(D+J)\ddot{\theta} + C\dot{\theta} + B\dot{\theta} + g = u \quad \textbf{(h.25)}$$

$$\text{Donde: } J = \begin{bmatrix} \frac{1}{r_1^2} J_m & 0 \\ 0 & \frac{1}{r_1^2} J_m \end{bmatrix}$$

$$B = \begin{bmatrix} \frac{1}{r_1^2}(B_m + \frac{K_b K_m}{R_a}) & 0 \\ 0 & \frac{1}{r_1^2}(B_m + \frac{K_b K_m}{R_a}) \end{bmatrix}$$

$$u = \begin{bmatrix} \frac{K_m}{r_1 R_a} v_1 \\ \frac{K_m}{r_2 R_a} v_2 \end{bmatrix} \quad ;donde \quad \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} \text{ son los voltajes aplicados.}$$

Y se llega al modelo dinamico del manipulador, que por simplicidad se expresa como:

$$M\ddot{\theta} + C\dot{\theta} + B\dot{\theta} + g = u \quad \textbf{(h.26)}$$

Donde M=D+J

[6] M. W. Spong & M. Vidyasagar *op. cit.* pp216-217

Otra forma de representar la dinámica del manipulador es por medio de un regresor (Y) y un vector de parámetros (θ), y como se puede escoger libremente el vector de parámetros se tiene que esta representación no es única.

Para un vector de parámetros dado, se tiene una representación:

$$\tau = Y \cdot \theta \qquad \textbf{(h.27)}$$

Donde:

$$\theta = \begin{bmatrix} m_1 l_{C1}^2 + m_2 l_1^2 + I_1 \\ m_2 l_1 l_{C2} \\ m_2 l_{C2}^2 + I_2 \\ m_2 l_{C2} \\ m_1 l_{C1} + m_2 l_1 \end{bmatrix}$$

$$Y = \begin{bmatrix} \ddot{q}_{d1} & \cos(q_{d2} - q_{d1})\ddot{q}_{d2} - \text{sen}(q_{d2} - q_{d1})\dot{q}_{d2}^2 & 0 & 0 & g\cos(q_{d1}) \\ 0 & \cos(q_{d2} - q_{d1})\ddot{q}_{d1} + \text{sen}(q_{d2} - q_{d1})\dot{q}_{d1}^2 & \ddot{q}_{d2} & g\cos(q_{d2}) & 0 \end{bmatrix}$$