Universidad Autónoma de Querétaro
Facultad de Ingeniería
INGENIERÍA EN INSTRUMENTACIÓN Y CONTROL DE PROCESOS

"Diseño de Software, tarjeta y controlador de motores a pasos para generación de trayectorias"

TESIS

Que para obtener el título de:

INGENIERO EN INSTRUMENTACIÓN Y CONTROL DE PROCESOS

Presenta:

HÉCTOR IVÁN CADENA QUINTERO

Dirigido por:

DR. GILBERTO HERRERA RUIZ

Santiago de Querétaro, Qro. Enero de 2004

ACUERDO **848/03**
21 DE OCTUBRE DE 2003

C. HÉCTOR IVÁN CADENA QUINTERO
Pasante de la Licenciatura en Instrumentación y control de Procesos
Presente.

Con relación a su solicitud relativa a la opción de titulación **Tesis Individual,** con título **"Diseño de Software, tarjeta y controlador de motores a pasos para generación de trayectorias"** me permito informarle que en sesión ordinaria del 21 de octubre del año en curso, fue aceptado su protocolo de tesis individual, bajo la dirección del **Dr. Gilberto Herrera Ruiz.**

El contenido aceptado por el H. Consejo Académico es el siguiente:

C O N T E N I D O

3. BUSES Y PUERTOS DE COMUNICACIÓN

3.1 Interfaces de E/S para la PC.
3.2 E/S aislada.
3.3 Descripción del puerto serial.
3.4 Descripción del puerto paralelo
3.5 E/S ubicada en memoria.
3.6 Descripción del bus ISA (Arquitectura Estándar del Industria)
3.7 Descripción del bus VESA
3.8 Descripción del bus PCI
3.9 Descripción del USB (Bus Serial Universal)
3.10 Descripción del AGP (Puerto Gráfico Acelerado)
3.11 Selección de la interfaz a trabajar
3.12 Descripción de las señales del bus isa.
3.13 Mapa de E/S de una computadora personal
3.14 Interfaces básicas de entrada y de salida
3.15 Interfaz de entrada.
3.16 Interfaz de salida.

4. DISEÑO DE LA TARJETA DE CONTROL DE MOVIMIENTO.

4.1 Decodificación de direcciones.
4.2 Selección del dispositivo para la decodificación de direcciones.
4.3 Programación del PLD para la decodificación de direcciones.
4.4 Conservación de la información transferida al bus de datos.
4.5 Generación de secuencias para motores a pasos usando un PLD.
4.6 Generación de secuencias para motores a pasos usando un PLD.
4.7 Generación de secuencias para un motor a pasos mediante el circuito L297.
4.8 Esquema descriptivo del L297.4.3.2.2. Usos comunes del L297.
4.8 Descripción del circuito L297.
4.8 Selección de circuito para la generación de secuencias.
4.9 Implementación de la tarjeta de control de movimiento.

5. ETAPA DE POTENCIA.

5.1 Generación de potencia por el ULN2803.
5.2 Modificaciones para mejorar el desempeño de los buffers de transistores.
5.3 Ventajas del uso del ULN2803.
5.4 Desventajas del uso del ULN2803.
5.5 Generación de potencia por el circuito L298.
5.6 El puente H.
5.7 Limitación de corriente máxima (Chopper).
5.8 Cálculo de la frecuencia de Chopper.
5.9 Selección de la etapa de potencia.
5.10 Implementación de la etapa de potencia.

6. SOFTWARE DE GENERACIÓN DE TRAYECTORIAS.

6.1 Tipos de Interpoladores.
6.2 El Integrador DDA.
6.3 Principio de funcionamiento del integrador DDA.
6.4 Aceleración y desaceleración exponencial.
6.5 Interpolación lineal.
6.6 Interpolación circular
6.7 Implementación del algoritmo

CONCLUSIONES

ANEXOS

BIBLIOGRAFÍA

1.- Robótica Industrial, Mikell P. Groover, Mitchel Weiss, Roger N. Nagel y Nicholas G. Odrey, Mc Graw Hill, Primera edición Febrero de 1989.
2.- Robot Dynamics and Control, Mark W. Spong, M. Vidyasagar, Impreso por Quinn-Woodbine, Inc., 1987.
3.- Electromechanical Systems, Electric Machines and Applied Mechatronics, Sergey E. Lyshevsky, Impreso por CRC Press, 2000.
4.- The Industrial Electronics handbook, Editor en jefe J. David Irwin, Impreso por CRC Press e IEEE Press, 1997.
5.- Mechanical Enginners Handbook, Editor Myer Kutz, Impreso por Wiley-Interscience, Segunda edición, 1998.
6.- Mechanical Design Handbook, Harold A. Rothbart, Editorial Mc Graw Hill, 1996.
7.- Manual del Ingeniero Mecánico, Baumeister y Marks, Editorial UTEHA, 1960.
8.- Autómatas programables, Joseph Balcells, José Luis Romeral, Editorial Alfaomega, 1998.
9.- Análisis de Circuitos en Ingeniería, William H. Hayt, Jr., Jack E. Kemmerly, Editorial Mc Graw Hill, Quinta edición 1993.
10.- Sistemas Electrónicos Digitales, Enrique Mandado, Editorial Alfaomega, Séptima edición 1998.
11.- Electrónica Industrial: Técnicas de potencia, J.A. Gualda, S. Martínez, P.M. Martínez, Editorial Alfaomega, Segunda Edición 1998.
12.- Electronics in Industry, George M. Chute, Robert D. Chute, Impreso por GLENCOE DIVISIÓN Macmillan/Mc Graw Hill, Quinta edición 2001.
13.- The Electronics Handbook, Editor en jefe Jerry C. Whitaker, Impreso por CRC Press e IEEE Press, 1996.
14.- The Electrical Engineering Handbook, Editor en jefe Richard C. Dorf, Impreso por CRC Press e IEEE Press, 1997.
15.- Eletrónica Industrial Moderna, Timothy J. Maloney, Editorial Prentice Hall, tercera edición 1997.
16.- VHDL Design Representation and Síntesis, James R. Armstrong, F. Gail Gray, Editorial Prentice Hall PTR, Segunda edición 2000.
17.- Fundamentos de los Microprocesadores, Roger L. Tokheim, Editorial Mc Graw Hill, Segunda edición 1985.
18.- The Mechatronics Handbook, Editor en Jefe Robert H. Bishop, impreso por CRC Press e ISA, 2002.
19.- Electrónica de Potencia, Circuitos, Dispositivos y Aplicaciones, Muhammad H. Rashid, Pearson Education, Prentice May, Segunda Edición 1993.
20.- Electrónica: Teoría de circuitos, Robert L. Boylestad, Louis Nashelsky, Pearson Education, Prentice May, Sexta Edición 1997.
21.- Inductrial Electronics and Robotics, Shuler, Mc Namee Editorial Mc Graw Hill, 1986.
22.- Analog and Computer electronics for Scientists, Publicada por Wiley-Interscience, John Wiley & Sons, Inc., Cuarta edición 1993.
23.- The designer's guide to VHDL, Peter J. Ashenden, Publicada por Morgan Kaufman Publishers Inc., 1996.

24.- Arquitectura de Computadoras y Procesamiento Paralelo, Kai Wwang/Fayé A. Briggs, Editorial Mc Graw Hill, 1988.
25.- Organización y Arquitectura de Computadoras, William Stallings, Editorial Prentice Hal, Quinta edición, 2000.
26.- The Circuits and Filtres Handbook, Wai-Kai Chen, impreso por CRC Press e IEEE Press, 1995.
27.- Fundamentals of Logic Design, Charles H. Roth, Jr. Publicado por PWS Publishing Company, Cuarta edición 1995.
28.- Digital Electronics with PLD Integration, Nigel P. Cook, Editorial Prentice Hall, 2001.
29.- A Guide to VHDL, Stanley Mazor, Patricia Langstrat, Publicado por Kluwer Academic Publishers, Segunda Edición 1993.
30.- Organización y Arquitectura de Computadoras, Jaime Martínez Garza, Editorial Prentice Hall, 2000.
31.- Ingeniería Computacional, Diseño de Hardware, M. Morris Mano, Editorial Prentice Hall Hispanoamericana, 1991.
32.- Programmable Logic Handbook, Geoff Bostock, Publicado por Butterworth Heinemann Ltd, Segunda edición, 1994.
33.- The Control Handbook, Editor William S. Levine, Impreso por CRC Press e IEEE Press, 1996.
34.- Ingeniería de Control Moderna, Katsuhiko Ogata, Editorial Pearson-Prentice Hall, Tercera edición 1998.
35.- Lenguaje Ensamblador para microcomputadoras IBM, J. Ferry Godfrey, Editorial Prentice Hall, 1991.
36.- Ensamblador básico, A. Rojas, Editorial Computec, 1995.
37.- Los microprocesadores Intel, Barry B. Brey, Editorial Prentice Hall, Quinta edición 2001.

También hago de su conocimiento las disposiciones de nuestra Facultad, en el sentido de que previo a su Examen Profesional, deberá cumplir con los requisitos de nuestra Legislación y deberá reimprimir el presente oficio en todos los ejemplares de su Tesis.

Sin más por el momento quedo de usted.

Atentamente,

"EL INGENIO PARA CREAR, NO PARA DESTRUIR"

M. en I. Gerardo René Serrano Gutiérrez
Director de la Facultad

C.c.p. Archivo.
GRSG/RRPV/img.

C.U., 10 de noviembre del 2003

H. CONSEJO ACADÉMICO
DE LA FACULTAD DE INGENIERÍA
P R E S E N T E:

Me permito comunicar a este Órgano Colegiado que, una vez revisada la tesis individual con titulo **"Diseño de Software, Tarjeta y Controlador de Motores a Pasos para Generación de Trayectoria"** del pasante de la Licenciatura en Instrumentación y Control de Procesos **HÉCTOR IVÁN CADENA QUINTERO,** de acuerdo al articulo 20 inciso H del Reglamento de Titulaciones Vigente, emito mi voto aprobatorio.

Atentamente,
"EL INGENIO PARA CREAR, NO PARA DESTRUIR"

Dr. Gilberto Herrera Ruíz
Director de Tesis

C.c.p. Archivo
*RRPV/lgo

C.U., 10 de noviembre del 2003

H. CONSEJO ACADÉMICO
DE LA FACULTAD DE INGENIERÍA
P R E S E N T E:

Me permito comunicar a este Órgano Colegiado que, una vez revisada la tesis individual con titulo **"Diseño de Software, Tarjeta y Controlador de Motores a Pasos para Generación de Trayectoria"** del pasante de la Licenciatura en Instrumentación y Control de Procesos **HÉCTOR IVÁN CADENA QUINTERO**, emito mi voto aprobatorio.

Atentamente,
"EL INGENIO PARA CREAR, NO PARA DESTRUIR"

M. en C. Pedro Alaníz Lumbreras
Sinodal

c.c. Archivo
*RRPV/img.

C.U., 10 de noviembre del 2003

H. CONSEJO ACADÉMICO
DE LA FACULTAD DE INGENIERÍA
P R E S E N T E:

Me permito comunicar a este Órgano Colegiado que, una vez revisada la tesis individual con titulo **"Diseño de Software, Tarjeta y Controlador de Motores a Pasos para Generación de Trayectoria"** del pasante de la Licenciatura en Instrumentación y Control de Procesos **HÉCTOR IVÁN CADENA QUINTERO**, emito mi voto aprobatorio.

Atentamente,
"EL INGENIO PARA CREAR, NO PARA DESTRUIR"

M. en I. Sergio Eduardo Cervantes Pérez
Sinodal

c.c. Archivo
*RRPV/img.

C.U., 10 de noviembre del 2003

H. CONSEJO ACADÉMICO
DE LA FACULTAD DE INGENIERÍA
P R E S E N T E:

Me permito comunicar a este Órgano Colegiado que, una vez revisada la tesis individual con titulo **"Diseño de Software, Tarjeta y Controlador de Motores a Pasos para Generación de Trayectoria"** del pasante de la Licenciatura en Instrumentación y Control de Procesos **HÉCTOR IVÁN CADENA QUINTERO**, emito mi voto aprobatorio.

Atentamente,
"EL INGENIO PARA CREAR, NO PARA DESTRUIR"

Ing. Mario Trejo Perea
Sinodal

c.c. Archivo
*RRPV/img.

Agradecimientos

A **Dios Todopoderoso**, que me ha dado tantos regalos en la vida. Por favor sígueme ayudando.

A **mi madre**, por todo el apoyo y la formación como persona que me ha dado desde que nací. Como puede ver mamá el "papelito" no se lo entrego siendo un anciano.

A **mi padre**, quien a pesar de no estar conmigo, me dio su cariño y apoyo en mi niñez y adolescencia. Yo lo entiendo papá.

A **mi hermana mayor Liliana**, por su ejemplo de tenacidad e investigación, fuiste tu quien en la niñez me sembró la semilla de la curiosidad por saber como funcionaban las cosas cuando entre los dos abríamos los aparatos eléctricos "Para saber que tenían".

A **mi hermano Hugo**, quien siempre me ha puesto los pies en la tierra cada vez que me subo demasiado. Nada más no seas tan brusco.

A **mi hermana Palmira**, tu cariño aunque a veces raro, me hace sentir muy bien. Aunque a veces no lo puedo demostrar, respeto tu forma de ser.

A **Alicia, mi crayolita**, por ese sentimiento que hace que todo parezca que se puede hacer. Perdona que no te pueda dedicar el tiempo que te mereces.

Al **M. en C. Sergio Cervantes,** quien le dio a la carrera un toque más profesional en base a mucho trabajo. Solamente le puedo decir que Don Quijote le dijo a Sancho Panza: "Avanzad, avanzad, dejad que ladren los perros; significa que vamos por buen camino".

Al **Dr. Gilberto Herrera**, por todo el apoyo en este trabajo y en la maestría. Esta tesis no se hubiera podido hacer sin su apoyo.

Prólogo

Cuando uno llega de primer ingreso a la carrera de Instrumentación y Control de Procesos y ve a los alumnos de últimos semestres haciendo robots y procesos automáticos uno siempre dice que hará lo mismo o cosas mejores. La mejor manera aparte de aprender fue para mi gusto el de implementar los dispositivos, de esta forma se fijaban los conocimientos de mejor forma y se descubrían varios aspectos que no se habían visto en clase. Para ello siempre tenía un arsenal de preguntas para todos aquellos que hacían un proyecto que yo no había hecho o que habían mejorado. Este trabajo trata de aportar por lo menos los conocimientos básicos para implementar un servomecanismo el cual en este caso es en este caso una mesa XY, así de esta manera si alguien me honra con leer esta tesis podré transmitir algo de lo que aprendí en la carrera y que considero tiene un gran valor.

Introducción y objetivo.

Debido a que en México se importa la mayoría del equipo de Control Numérico los cuales se basan principalmente en generadores de trayectorias, los costos de recuperación de la inversión en la compra de equipo son muy altos, por lo que es necesario que se implemente tecnología mexicana para que los costos de estos equipos se hagan más bajos y así poder competir a nivel mundial con los productos nacionales.

El objetivo de esta tesis es la construcción de un servomecanismo, en este caso de una mesa de trabajo XY la cual generará trayectorias circulares o lineales. No se busca por el momento, crear un dispositivo con fin industrial, sino que se persigue solamente aportar los conocimientos necesarios para poder entender una máquina de este tipo. Se aprovechó la estructura de los microprocesadores INTEL debido a que al ser tan comerciales una computadora de desecho (80486, Pentium I) corre perfectamente los programas necesarios para esta tarea y su costo es inferior a 1000 pesos MN.

El **Capítulo 1** Muestra una forma de seleccionar componentes para el diseño mecánico de una mesa XY con un eje adicional Z el cual sirve para subir o bajar una pluma.

El **Capítulo 2** Trata sobre los motores a pasos desde su descripción hasta su selección de tipo y el porque escogerlos sobre otros motores para este trabajo.

El **Capítulo 3** Describe los buses y puertos de comunicaciones los cuales son la interfase con el microprocesador, también se muestra como se hizo la selección del bus ISA sobre otras interfases así como una descripción más completa del bus ISA.

El **Capítulo 4** Muestra el diseño de la tarjeta de control de movimiento, la decodificación de las direcciones, programación de PLDs en lenguaje VHDL y un circuito comercial de generación de secuencias para un motor a pasos además de mostrar el diseño de la tarjeta.

El **Capítulo 5** Hace referencia a la etapa de potencia, la cual da el poder necesario a los motores a pasos para que éstos se muevan. Se hace referencia a dos tipos de circuitos (ULN2803 y el L298) para la implementación.

El **Capítulo 6** Trata sobre la programación de los algoritmos para la generación de trayectorias, el integrador DDA, la interpolación lineal, la interpolación circular y la aceleración.

Contenido

Índice de figuras.

Capítulo 1

Parte mecánica.

Esta sección muestra algunas consideraciones para el diseño de un sistema mecánico, desde el maquinado hasta el acoplamiento de los motores. El sistema mecánico consta de tres ejes, de los cuales, solamente dos el X y el Y mueven una plataforma a la cual le llamaremos mesa XY. El tercer eje sostendrá, bajará y subirá una pluma la cual podrá dibujar en la superficie en la mesa. Cabe mencionar que en el diseño mecánico intervienen muchos fenómenos y dispositivos, sin embargo, en este trabajo sólo se discuten los de más peso; varios de ellos son explicados, lo cual da una forma de selección lógica del dispositivo a utilizar.

1.1. Maquinado de piezas e implementación de ejes.

El maquinado de las piezas se realizó en una fresadora semiautomática FANUC de alta precisión. Para maquinar las piezas del eje X se usó nylamid blanco de espesor de 1cm, en el cual se maquinaron 4 cuadros para acoplar el tornillo sin fin de calidad estándar con cuerda de ¼" (pulgadas). Se usaron dos varillas de guía de ¼ " rectificadas para un mejor deslizamiento de los bujes, los cuales están en el rectángulo donde se ubica la tuerca para girar en un tornillo sinfín, creando así un tornillo de potencia. Para el eje Y se usó el mismo procedimiento que en el X.

1.2. Tornillo de potencia.

El acoplamiento del tornillo sinfín con tuerca que fue usado en este trabajo se le conoce comúnmente como "tornillo de potencia o fuerza", se utiliza para convertir movimiento angular a movimiento longitudinal y transmitir así, generalmente fuerza o potencia. Hay varios tipos de tornillos los cuales fluctúan en precio desde 50 pesos hasta varios miles de dólares. Las aplicaciones más comunes incluyen, válvulas, husillos, tornillos de avance de tornos, tornillos de banco, gatos y prensas. Este montaje permite un desplazamiento lineal muy exacto por la reducción tan grande de desplazamiento rotacional a longitudinal.

El ángulo de los dientes del tornillo sinfín es de gran importancia, la eficiencia depende de su amplitud, si es muy amplio baja la eficiencia. Por ende los dientes en V no son aptos para la transmisión de grandes cargas, sin embargo se usan donde la precisión no es tan importante o se puede compensar, donde se requiere una transmisión de fuerza de bajo costo y/o donde la demanda de fuerza es muy pequeña.

La forma de calcular la distancia lineal d_m que recorre el tornillo por cada revolución depende de la distancia entre crestas o valles p correspondiente a dos dientes consecutivos) [Harold A. Rothbart,1996], quedando definido para n número de vueltas como:

$$d_m = np$$

Siendo que el tornillo tiene una distancia entre crestas de 1 mm y definimos para una vuelta entonces la distancia que recorre linealmente la mesa por revolución es de solamente 1 mm.

El par requerido para mover la carga debido a que no hay un levantamiento de la mesa, no hay fuerzas verticales que contrarrestar y por lo mismo no se consideran. El elemento que sí podemos usar para calcular el par (sin considerar la inercia), es la fuerza a vencer ocasionada por la fricción (la cual dependerá del radio medio del tornillo y del paso o avance dado) [Harold A. Rothbart,1996]. Al considerar los elementos descritos anteriormente podemos ver que el par necesario para mover la mesa XY se define con la siguiente ecuación:

$$T = \frac{d_m}{2}\left(\frac{1+\pi\mu d_m}{\pi d_m - \mu l}\right)$$

donde d_m es el radio medio del tornillo, μ es la fricción de la rosca y l es el paso o avance dado.

Con un diámetro de 5 mm, y tomando el peor caso para la fricción en seco del acero que es de 0.25) [Harold A. Rothbart,1996] .

Entonces

$$d_m = d\text{-}p/2 = 0.8\text{-}1/8 = 0.675 \text{ plg}$$

$$T = \frac{0.675}{2}\left(\frac{1+3.141516*0.675}{3.141516*0.675-0.25*0.8}\right)$$

donde T = 1.11 lb*plg= 17.76 oz*plg

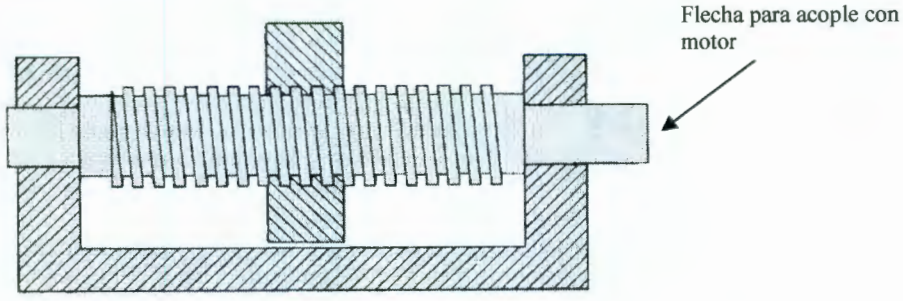


Figura 1.1. Ensamble del tornillo para el desplazamiento de la mesa.

1.2.1. Selección de uso de tornillo de potencia.

La selección del tornillo de potencia usado se baso en el tipo de carga, la cual es una mesa de peso ligero, por lo tanto, la selección de un tornillo de dientes en "V" es recomendable, ya que no tendría un error muy significativo en el desplazamiento y su costo es moderado.

Para poder acoplar el motor con el tornillo de potencia se maquinó un acoplamiento.

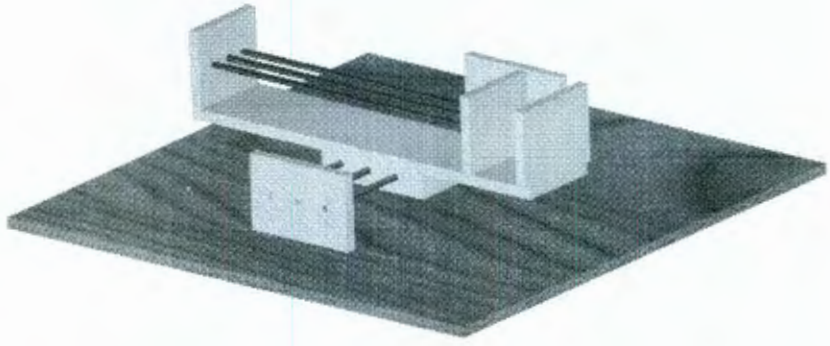


Figura1.2. Mesa XY con tornillo de potencia.

1.3. Acoplamientos.

Los acoplamientos son conexiones entre dos árboles o ejes, y permitir también ciertos desplazamientos de éstos, que pueden ocurrir al ser instalados o durante su funcionamiento.

Los desplazamientos de los ejes pueden ser de tres clases:

- Desplazamiento axial
- Desplazamiento radial
- Desplazamiento angular

Dos ejes sufren desplazamiento axial sólo cuando se encuentran alineados y se desplazan a lo largo de su eje geométrico. Mientras que su desplazamiento es radial si los ejes geométricos permanecen paralelos, aunque separados por una distancia r. Por otro lado, el desplazamiento es angular cuando los ejes forman entre sí un determinado ángulo α. Los tres tipos de desplazamiento se muestran e en la Figura 1.3.

La selección de acopladores se basa en las siguientes consideraciones:

- Carga: Carga máxima, cargas estables y vibratorias, transmisión de torque.
- Desalineamiento: Desalineación paralela y angular máximas, habilidades para compensar desplazamiento axial.
- Rigidez: Especialmente en el caso de acoplamientos flexibles, la deflexión máxima permisible del acoplamiento debido a las cargas estáticas y dinámicas.

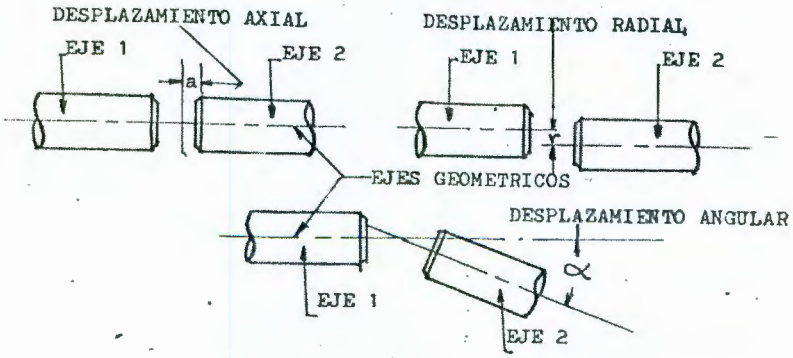


Figura 1.3. Desplazamiento de los ejes.

La mayoría de los acoplamientos todavía no han sido estandarizados, y la información específica de ellos es más fácil de obtener de las hojas de datos, lo cual es una práctica muy común, ya que se generan lotes de acoplamientos con características de acuerdo a las diferentes aplicaciones.

Pero cuando se diseña un acoplamiento, es importante considerar:

- Tamaño. Las dimensiones totales del acoplamiento.
- Peso. Debido a que es un factor importante en la dinámica y estática del sistema.
- Balance estático y dinámico. Debido a que puede introducir fuerzas adicionales a nuestro sistema.
- Instalación y mantenimiento. Por la accesibilidad al acoplamiento tratando de no desensamblaje del sistema.

1.3.1. Tipos de acoplamientos.

Los acoplamientos pueden ser:

- Acoplamientos rígidos
- Acoplamientos flexibles
- Acoplamientos deslizantes
- Acoplamientos de unión universal tipo Hooke
- Acoplamientos contra sobrecarga

De los cuales sólo describiré dos tipos ya que son los que más interesan en este trabajo.

1.3.1.1. Acoplamientos rígidos.

Los acoplamientos rígidos se utilizan cuando los ejes están perfectamente alineados y no van a sufrir ninguna clase de desplazamiento durante el funcionamiento. Tal alineación es difícil de conseguir en la práctica, y si se consigue es difícil mantenerla a causa de factores como la variación de la temperatura, el desgaste de los cojinetes, la deformación de los árboles sometidos a carga, etc., los cuales originan esfuerzos que pueden conducir a la ruptura del acoplamiento. Por eso los acoplamientos rígidos se usan más que todo en ejes relativamente flexibles y que trabajan a bajas velocidades.

Los acoplamientos rígidos se fabrican en tres formas diferentes:

Acoplamiento de manguito. El acoplamiento de manguito consta de un manguito o buje y dos pasadores cónicos que unen rígidamente los extremos de dos flechas (Figura 1.4).

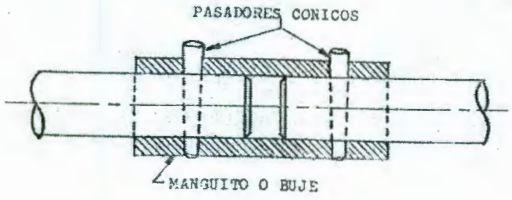


Figura 1.4. Acoplamiento de manguito.

Acoplamiento de manguito partido. El acoplamiento de manguito partido consta de dos mitades que se ajustan mediante pernos y de una chaveta larga que ayuda a la alineación de los dos ejes. Una de las ventajas que ofrece este tipo de acoplamiento es que puede ser montado y desmontado sin necesidad de mover los ejes (Figura 1.5).

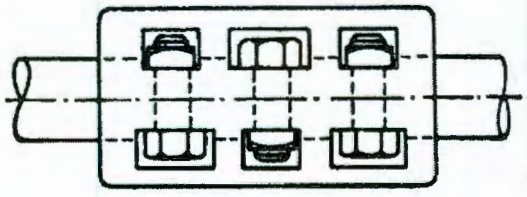


Figura1.5. Acoplamiento de manguito partido.

Acoplamiento de bridas. El acoplamiento de bridas (Figura 1.6) está compuesto de dos platos o bridas que se ajustan mediante pernos, y de una chaveta larga. Además de tener la misma ventaja de montarse y desmontarse sin mover los ejes, como el acoplamiento de manguito partido, tiene la ventaja de transmitir grandes potencias.

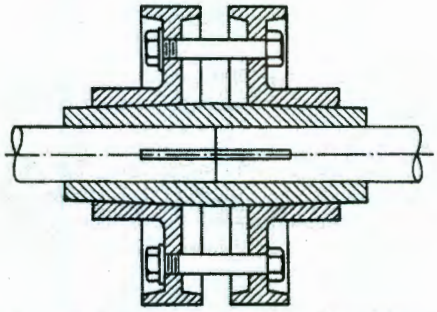


Figura 1.6. Acoplamiento de bridas.

1.3.1.2. Acoplamientos flexibles

Los acoplamientos flexibles constan básicamente de dos partes metálicas iguales, unidas a cada flecha mediante chavetas, y de un elemento intermedio flexible que puede ser de caucho o metal.

Los acoplamientos flexibles permiten pequeños desplazamientos de los ejes, tanto de forma axial como radial, y sirven también para absorber choques y vibraciones ocurridas durante el funcionamiento.

Los acoplamientos flexibles más utilizados son:

Acoplamiento tipo Oldham. El acoplamiento tipo Oldham (Figura 1.7) está compuesto de dos discos metálicos ranurados frontalmente y de una pieza intermedia de caucho que ensambla las ranuras, las cuales forman entre sí un ángulo de 90°. Este tipo de acoplamiento permite cierto desplazamiento axial y radial de los ejes.

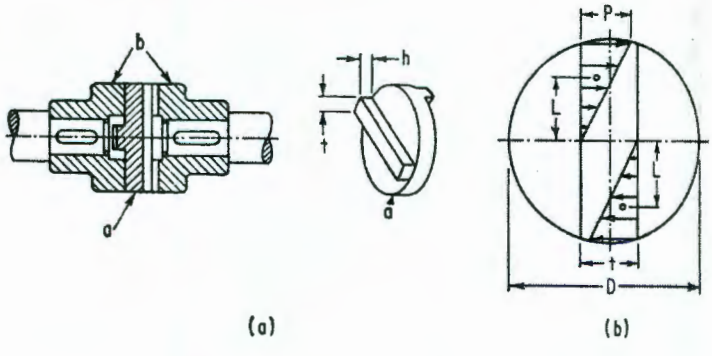


Figura 1.7. Acoplamiento tipo Oldham. a) Acoplamiento completo, b) Fuerzas actuando sobre el acoplamiento.

Acoplamiento de cadena. El acoplamiento de cadena (Figura 1.7) está compuesto por dos cubos con ruedas dentadas, sobre las que se adapta una cadena doble de rodillos (Ver Figura 1.8), de tal manera que la transmisión de movimiento se efectúa a través de la cadena. Este tipo de acoplamiento permite no sólo cierto desplazamiento axial y radial, sino también, un pequeño desplazamiento angular de los árboles acoplados.

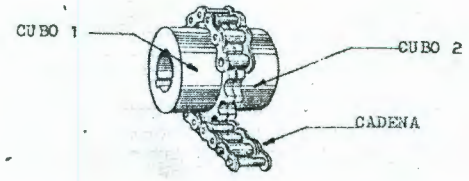


Figura 1.8. Acoplamiento de cadena.

Acoplamiento de cojín. El acoplamiento de cojín (Figura 1.9) consiste de dos bridas enchavetadas en los ejes y de una banda intermedia de caucho que actúa a manera de cojín. Los acoplamientos de cojín se caracterizan por soportar un alto grado de desplazamiento de los árboles, tanto axial como radial, así como angular.

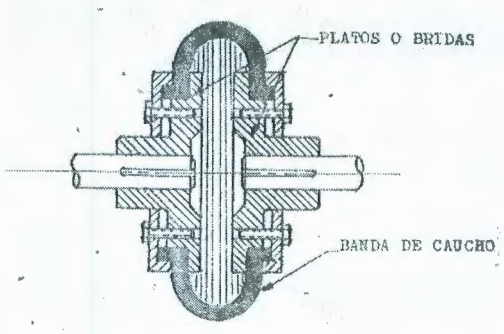


Figura 1.9. Acoplamiento de cojín.

1.3.2. Acoplamiento utilizado.

El acoplamiento de los motores a pasos se hizo con el acoplador de manguito, teniendo la precaución de que no tuviera un desalineamiento muy pronunciado en los motores, ya que de ser así, el motor se vería expuesto a esfuerzos muy grandes. En la Figura 1.10 se puede ver el montaje completo.

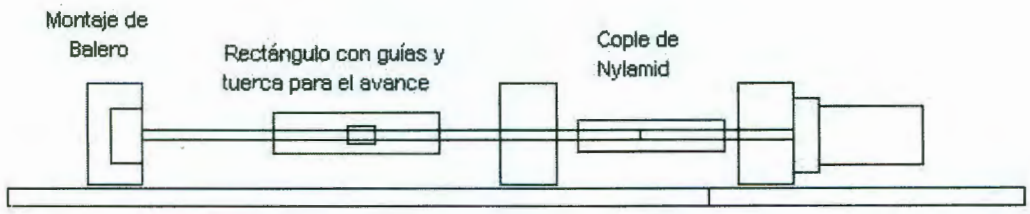


Figura 1.10.Montaje del eje X con el motor.

1.4. Implementación del eje Z y terminado del modelo

En la implementación del eje Z (Función de subir o bajar); lo único que se realizo fue el soldar un ángulo para sostener una pluma amortiguada, quedando la implementación completa del modelo como se ve en las Figuras 1.11 y 1.12.

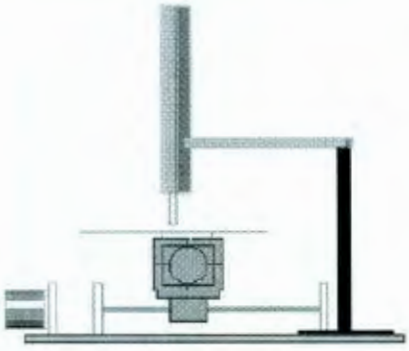


Figura 1.11 .Vista de la mesa con el eje Z implementado.

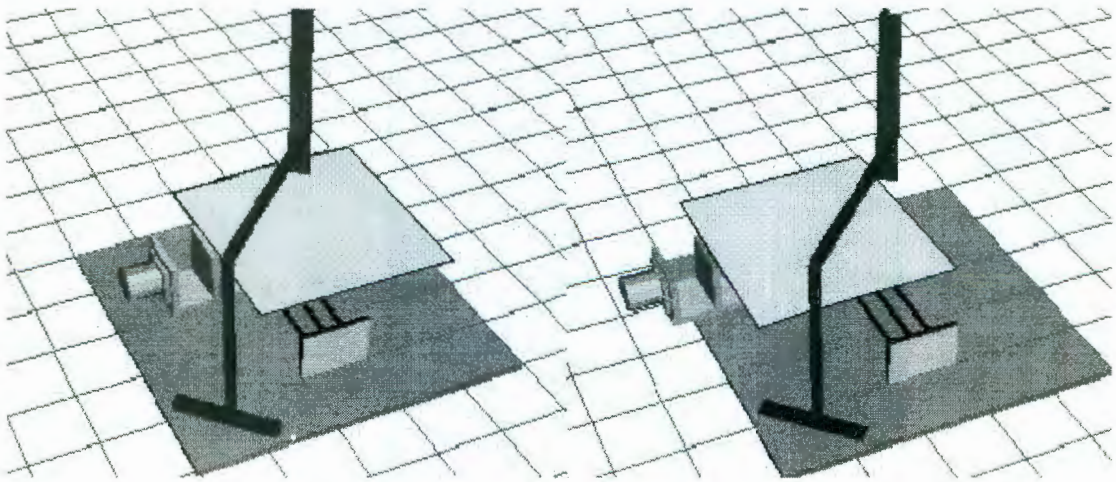


Figura 1.12. Vistas de la mesa con movimientos XY.

Ya teniendo la mesa de trabajo, lo siguiente es considerar el tipo de motores que se va a utilizar para el movimiento de la mesa, lo cual se hace en el capítulo 2.

Capítulo 2

Los motores a pasos.

En muchas aplicaciones de control automático, es necesario el accionamiento de válvulas o sistemas de engranes con una exactitud y precisión muy alta. En robótica, son indispensables estas características, donde las brazos mecánicos deben de ejecutar movimientos de gran precisión. Existen muchas otras ramas de la ingeniería como el de las máquinas-herramientas, aeronáutica, sector automotriz, etc. donde la utilización de dispositivos de posicionamiento mecánico es indispensable.

Este Capítulo describe que es y como un motor a pasos resuelve en gran medida este problema, ya que el principio de funcionamiento (el cual se muestra) le permite realizar pequeños movimientos (pasos), con gran exactitud y repetibilidad. Asimismo se hacen algunas comparaciones de este motor con otros tipos de motores, y la selección de motor a pasos de acuerdo a la aplicación.

2.1. Definición.

El motor a pasos es un motor eléctrico cuyo eje gira una cantidad específica por cada pulso de entrada que recibe, lo cual permite el control de posición, velocidad, y sentido (dirección).

La cantidad de rotación es directamente proporcional al número de pulsos y la velocidad de rotación es relativa a la frecuencia de dichos pulsos .Los motores a pasos son simples de operar en una configuración de lazo abierto y debido a su tamaño proporcionan un excelente torque a baja velocidad.

Los beneficios ofrecidos por estos motores incluyen :

- Un diseño efectivo y un bajo costo.
- Alta confiabilidad.
- Libres de mantenimiento debido a que no disponen de escobillas.
- Si se trabaja en condiciones nominales no requieren dispositivos de realimentación puesto que no se perderán pasos.
- Límite conocido al "error de posición dinámica " en condiciones nominales de operación.

2.2. Comparación entre un motor a pasos y un motor de corriente directa (DC).

Los motores DC de fácil rotación (llamado hobby motors o free-spinning DC motors), son dificiles de posicionar de forma precisa. Aun cuando se haya calculado el tiempo exacto para prender y apagar el motor, la armadura no para inmediatamente debido a que los motores DC tienen unas curvas de aceleración y desaceleración muy graduales, por lo tanto la estabilización es muy lenta. Agregando un juego de engranajes y amortiguadores podríamos reducir el problema, pero aún así la estabilización por parte del motor DC continua siendo lenta y afectaría características importantes como el torque que debe entregar el motor.

La única manera de usar un motor DC para un preciso posicionamiento sería usar un servomotor. Un servomotor consiste de un pequeño motor DC, un mecanismo de realimentación (usualmente es un potenciómetro conectado al rotor por medio de engranajes o encoder acoplado a la flecha del motor), y un circuito de control que compara la posición del motor con la posición deseada, y mueve el motor convenientemente. Esto puede ser en algunos casos muy complicado, costoso y ocupa demasiado espacio para muchas aplicaciones. Los motores a pasos, sin embargo, se comportan de una forma diferente que un motor DC. Primero, ellos no pueden correr libremente por su cuenta al ser desenergizados. Como su propio nombre lo dice, ellos se "paran" en un determinado tiempo. También difieren de los motores DC en su relación Torque-Velocidad. Los DC no son buenos para generar grandes torques a bajas velocidades, en cambio, los motores a pasos hacen lo contrario ya que producen grandes torques a bajas velocidades. Los motores a pasos tienen otra característica que no tiene un motor DC, el torque estático (holding torque), el cual permite que el rotor se mantenga en su posición cuando no esta girando. Esto es muy útil en aplicaciones donde el motor se prende y apaga continuamente, mientras la fuerza que actúa contra el motor aún esta presente. Esto elimina la necesidad de usar mecanismos de frenado.
Los motores a pasos no responden simplemente a una señal de reloj, ellos tienen muchos bobinados que tienen que ser activados en forma secuencial para hacer girar el rotor, ya sea en sentido horario o antihorario. Si las señales de control no son enviadas de forma correcta, el motor no girará, y si pudiera hacerlo, ocasionaría ruido y/o giraría de una manera tosca e inapropiada.
Un circuito capaz de convertir señales de reloj a señales de dirección y control para energizar secuencialmente las bobinas para generar movimiento rotacional en la flecha del motor se denomina como controlador (driver). Muchos motores a pasos incluyen un controlador seguido de una etapa de potencia para entregar la corriente necesaria a las bobinas del motor. La conexión de mores a pasos se ilustra en la Figura 2.1.

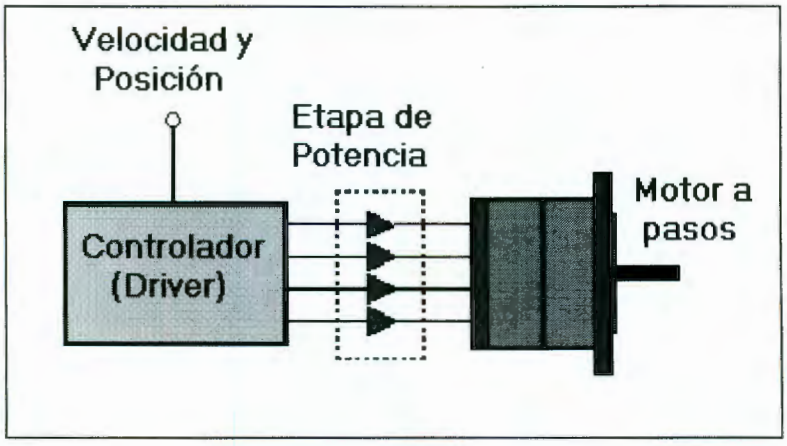


Figura 2.1. Conexión de motores a pasos.

Los motores a pasos se diferencian de los otros tipos de motores por las siguientes características:

- Convierten pulsos eléctricos en movimientos rotacionales discretos.
- No son muy rápidos en términos de RPM (revoluciones por minuto), en comparación con los demás tipos de motores, por ejemplo para un motor de 1000 pasos por segundo, tiene un RPM de 150 y 1.8° por paso.
- Siempre necesitan de un circuito especial externo para controlarlo (driver o controlador) debido a que no se le puede conectar directamente a una fuente de alimentación.
- Son ideales para el posicionamiento, ya que son de fácil manejo y normalmente no necesitan una constante de realimentación (lazo cerrado de control) o monitoreo. Lo único que se requiere es transmitir un número exacto de pasos para llevarlo a una posición exacta y repetible. El motor a pasos de lazo abierto es ideal para sistemas que operan a bajas aceleraciones y cargas estáticas, pero un sistema de lazo cerrado seria esencial para altas aceleraciones y cargas variables.
- Alcanzan una gran precisión y pueden moverse en incrementos muy pequeños, característica difícil de lograr en los motores DC pues aunque se desenergize el motor muy rápido, la inercia del rotor continuará girando el eje hasta una posición casual y si se trata de energizar al motor de DC con un voltaje muy pequeño, puede ser que debido a la zona muerta[1] del motor no haya movimiento rotacional.
- Debido a su bajo costo y pequeño tamaño en comparación con los demás tipos de motores, son empleados en disk-drivers, impresoras, plotters, etc.

[1] En teoría de control es un elemento que no tiene salida para entradas de determinada amplitud.

2.3. Características de los motores a Pasos.

Entre otras características comunes de los motores a pasos, tenemos los siguientes elementos:

2.3.1. Voltaje

Los motores a pasos usualmente tienen un rango de voltaje, que va indicado en el mismo motor o en las hojas de datos. Usualmente es necesario exceder el rango de voltaje para obtener el torque deseado de un motor dado, pero esto puede sobrecalentar y/o disminuir el tiempo de vida del motor debido a que la corriente excesiva en los devanados aumente la resistencia del cobre y se vuelva más quebradizo, además que con el excedente de corriente la temperatura aumentará, provocando que los rodamientos se dilaten ocasionando con ello mas fricción.

2.3.2. Resistencia

Una característica común es la resistencia por bobina. Esta resistencia determinará la corriente que pase por el motor, también como la curva de torque del motor y la máxima velocidad de operación.

2.3.3. Grados por paso

Este es el factor más importante al momento de escoger un motor para una determinada aplicación. Este factor especifica el número de grados que el rotor girará por cada paso. En la operación de medio paso del motor, el número de pasos por revolución es el doble y los grados por revolución se reducen a la mitad. Hay motores de 0.72°, 1.8°,3.6°, 7.5°, 15°, y hasta 90° por paso. Los grados por paso es comúnmente referido como la **resolución** del motor. Si un motor tiene sólo el número de pasos/revolución, basta dividir 360° por este número para obtener el valor de grados/pasos.

2.4. Tipos de motores a pasos

Existen dos tipos de motores a pasos los cuales son:los de imán permanente y los de reluctancia variable. Existen también los híbridos de imán permanente desde el punto de vista del controlador.

Los motores a pasos de imán permanente muestran resistencia cuando intentamos girar el eje con los dedos, mientras que los de reluctancia variable, casi siempre giran libremente o con menos dificultad. Sin embargo pueden mostrar cierta resistencia debido a una magnetización residual en el rotor.
Los motores a pasos de reluctancia variable usualmente tienen 3 o 4 devanados con un retorno común, mientras que los de imán permanente tienen 2 devanados independientes, sin tap central en motores bipolares, y con tap central en motores

unipolares. Existen una amplia gama de motores a paso. Hay desde motores con un paso de 90° hasta motores de alta resolución de imán permanente con un paso de 1.8° . Con un controlador adecuado un motor de imán permanente puede trabajar en fase partida, es decir a medios pasos. Algunos controladores pueden manejar micropasos.

2.4.1. Motor de reluctancia variable.

El tipo de motor de reluctancia variable (V.R.) (Figura. 2.2) consiste en un rotor y un estator cada uno con un número diferente de dientes . Ya que el rotor no dispone de un imán permanente el mismo gira libremente , o sea que no tiene torque de detención . A pesar de que la relación del torque a la inercia es buena , el torque dado para un tamaño de armazón dado es restringido , por lo tanto pequeños tamaños de armazones son generalmente usados y los mismos raramente varían para aplicaciones industriales .Éstos no son sensibles al cambio de polaridad de la corriente y requieren de otro tipo de manejo que los otros tipos de motores a pasos.

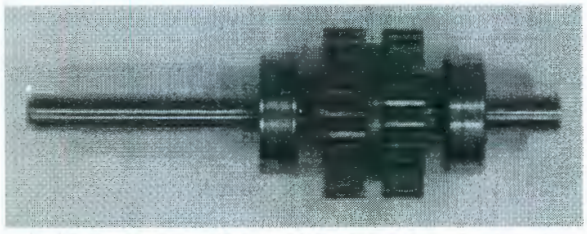


Figura 2.2. Vista de un rotor de un motor de reluctancia variable.

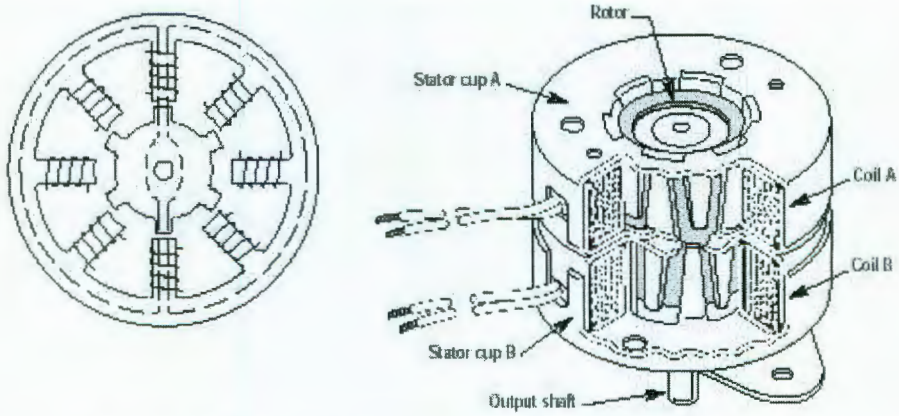


Figura 2.3. Vistas de un rotor y estator de un motor de reluctancia variable.

2.4.2. Motor de imán permanente.

El motor de imán permanente (PM) o tipo enlatado (Figura 2.4) es quizá el motor a pasos mas ampliamente usado para aplicaciones no industriales . En su forma mas simple , el motor consiste en un rotor imán permanentemente magnetizado radial y en un estator similar al motor de reluctancia variable . Debido a las técnicas de manufactura usadas en la construcción del estator, los mismos se conocen a veces como motores de "polo de uñas ".

Se usa en aplicaciones de bajo costo, bajo torque y baja velocidad. Parecidas a las que se usan en los periféricos de las computadoras. La construcción del motor resulta en ángulos de movimiento relativamente grandes, con una inercia relativamente baja, lo cual lo limita a aplicaciones como la de mover la cabeza de impresión de una impresora.

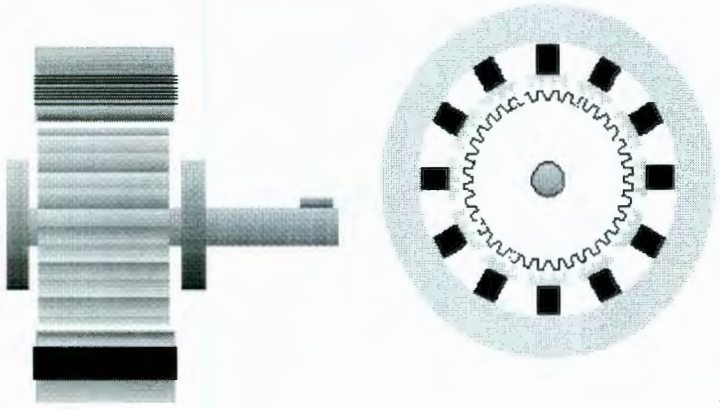


Figura 2.4. Vista en sección de un motor a pasos de imán permanente .

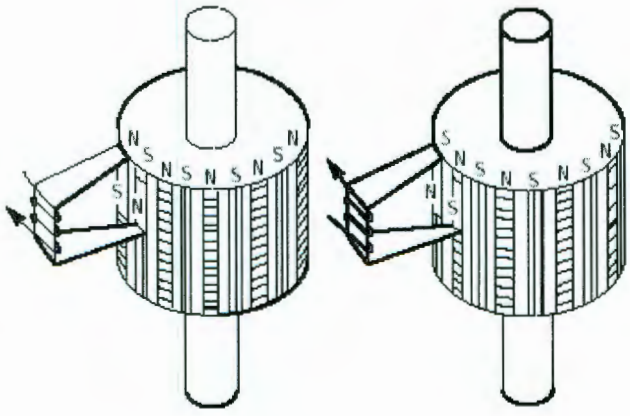


Figura 2.5 . Vista de un rotor de motor a pasos de imán permanente .

Dentro de los motores a pasos de imán permanente tenemos motores unipolares, motores bipolares y motores multifases los cuales identificaremos por la construcción de las bobinas del estator, podemos diferenciar entre motores "bipolares" y motores "unipolares" (Figura 2.6). Los primeros tienen las bobinas con un arrollamiento único, mientras que los segundos tienen las bobinas compuestas por dos arrollamientos cada una.

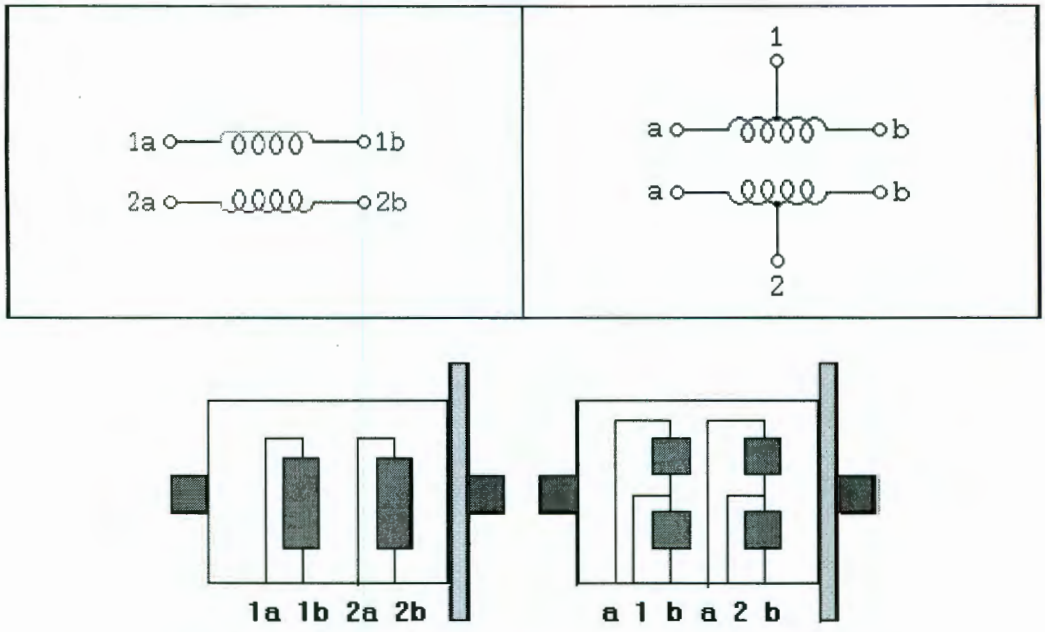


Figura 2.6. Devanados para motores de imán permanente unipolar y bipolar.

A igual número de espiras, el unipolar deberá tener menor sección en el hilo de las bobinas, aumentando por tanto su resistencia y disminuyendo su par a bajas velocidades.
Al número de grados que gira el rotor, cuando se efectúa un cambio de polaridad en las bobinas del estator, se le denomina "ángulo a paso" y puede ser muy variado en función de la aplicación y por tanto de la construcción del mismo.
Además, existe la posibilidad (con el control electrónico apropiado) de conseguir una rotación de medio paso. Los motores son fabricados para trabajar en un rango de frecuencias determinado por el fabricante y rebasado dicho rango, estaremos provocando una velocidad de giro del campo magnético creado por el estator muy elevada, no siendo el rotor capaz de alcanzar esa velocidad, provocando una pérdida de sincronización y quedando frenado en estado de vibración.

Su característica principal es tener un tap central, de manera que el esquema de cableado es tomar el (los) tap (s) centrales y conectarlos a la fuente de alimentación positiva. El circuito controlador se encargará de poner cada bobina a tierra para energizarla de manera secuencial. El número de fases es el doble al

número de bobinas, ya que cada bobina es dividida en dos por medio del tap central.

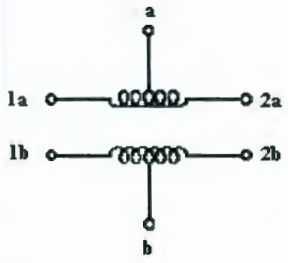
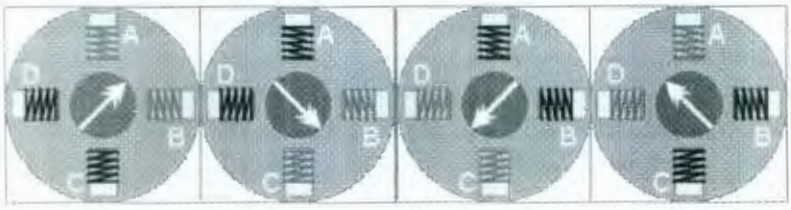
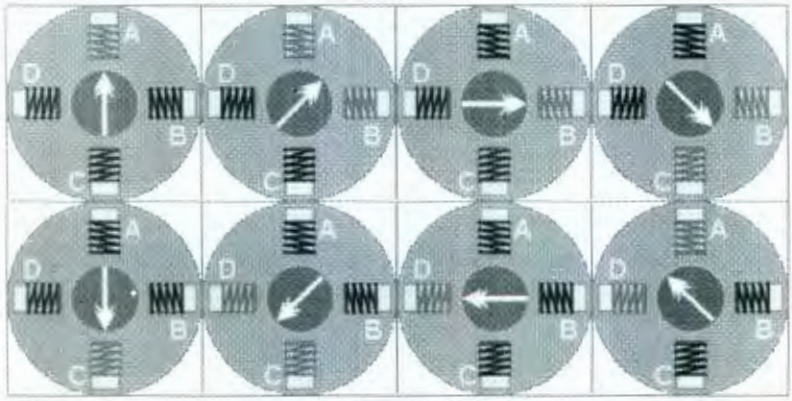


Figura 2.7. Conexión de cuatro fases.

En adición a la secuencia de manejo estándar, una secuencia de manejo a medio paso y otra de gran torque son posibles. En la secuencia de gran torque, 2 bobinados están activados al mismo tiempo para cada paso del motor. En este caso el torque es 1.5 veces mayor que el entregado en una secuencia estándar, pero maneja el doble de corriente. La secuencia de manejo a medio paso es la combinación de las 2 anteriores. Primero una bobina es activada, luego dos bobinas, luego una, etc. Esto hace que le número a pasos por revolución sea el doble y el ángulo por paso se reduzca a la mitad.



a)



b)

Figura 2.8.Secuencia de gran torque (Izq.) y secuencia de medio paso(Der.).

Los motores a pasos unipolar, ya sean de imán permanente o híbridos con 5 ó 6 cables, son cableados con un tap central en cada uno de los bobinados. Los tap centrales típicamente son conectados a al fuente de alimentación positiva, y los extremos de cada bobinado son alternativamente puestos a tierra para invertir la dirección del campo entregado por el bobinado.

2.4.3. Motor de tipo híbrido.

El motor de tipo Híbrido es probablemente el más usado de todos los motores por pasos. Originalmente desarrollado como un motor de imán permanente sincrónico de baja velocidad su construcción es una combinación de los diseños reluctancia variable e imán permanente. El motor híbrido consiste en un estator dentado y un rotor de tres partes (apilado simple). El rotor de apilado simple contiene dos piezas de polos separados por un imán permanente magnetizado, con los dientes opuestos desplazados en una mitad de un salto de diente (Figura 2.9) para permitir una alta resolución apasos .

Figura 2.9. Vista expandida ilustrativa del desplazamiento de dientes .

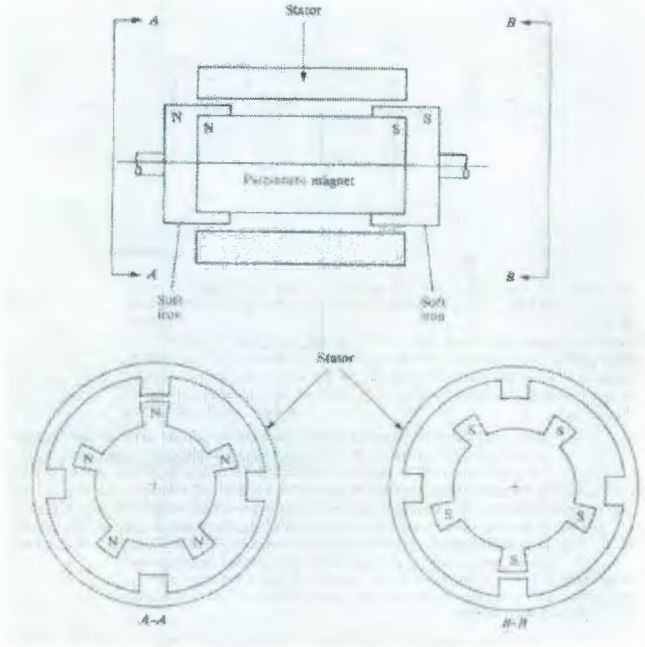


Figura 2.10. Vista de un rotor y un estator de un motor a pasos híbrido.

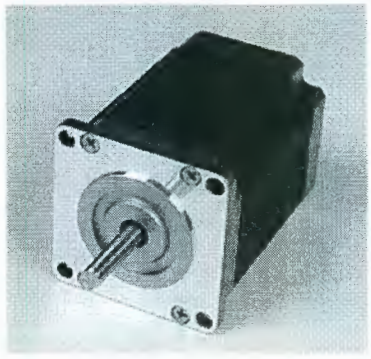


Figura 2.11. Vista de sección de un motor a pasos híbrido.

El motor híbrido (Figuras 2.10 y 2.11) se caracteriza por tener un gran torque de inercia al des-energizarse, una gran inercia del rotor y un gran torque al ser energizado.

2.5. CONCEPTOS BÁSICOS EN EL CONTROL DE MOTORES A PASOS

2.5.1. Pasos (steps)

La mayoría de estos motores tienen un paso entre 7.5° y 1.8°. Esto se traduce en 48 a 200 pasos por revolución respectivamente. Si se quiere una mejor resolución, el paso debe ser menor.

2.5.2. Pull-in y Pull-out rate

El Pull in rate es la máxima velocidad con la cual puede arrancar un motor con carga sin perder pasos. El Pull out rate es la máxima velocidad a la cual puede operar un motor con carga sin perder pasos. El Pull in rate es siempre menor al Pull out rate, ya que si se quiere que el motor rinda a su máxima velocidad sin perder pasos es necesario acelerar desde una velocidad menor.

2.5.3. Resonancia

Hacer resonar un motor significa que el motor sufre una pérdida de pasos. Esto ocurre a ciertas frecuencias que deben evitadas. Al operar un motor sin carga en un rango de frecuencias se detectarán frecuencias naturales de resonancia, estas se pueden detectar auditivamente o por medio de sensores.
Cuando se trabaja bajo estas condiciones o no se pueden evitar estas velocidades, se debe agregar un factor de amortiguamiento externo, mayor inercia o un driver o controlador adecuado.

Los motores de imán permanente son menos inestables que los de reluctancia variable, pues tienen mayor inercia en el rotor y un torque de arranque mas elevado.

2.5.4. Resolución de un motor a pasos y ángulo a pasos .

La resolución (número de pasos) y el ángulo de paso de un motor por pasos dependen de :

- El número de pares de polos del rotor(p) ,
- El número de fases del motor (m),
- El modo de impulsión (k = completa o medio paso)

La resolución puede ser calculada usando la fórmula:

$$z = p*m*k$$

El ángulo a pasos puede ser calculado dividiendo la rotación (360) por el número apasos.

2.6. Selección del tipo de motor

La selección del motor depende principalmente del torque que necesitemos para nuestra aplicación, así como la velocidad en la que podamos energizar y desenergizar las bobinas del motor a pasos. Si tenemos en cuenta que el modelo para energizar una bobina está dada por la relación:

$$I(s) = \frac{\frac{1}{R}}{\frac{L}{R}s+1} Eo(s)$$

donde I(s)=Corriente de la bobina, R=Resistencia de la bobina, L=Inductancia de la bobina, Eo(s)=Voltaje de alimentación.

Como podemos ver de la ecuación anterior este sistema es de primer orden viéndolo como un sistema de control en Laplace, por lo que su respuesta transitoria es como se ve en la Figura 2.12.

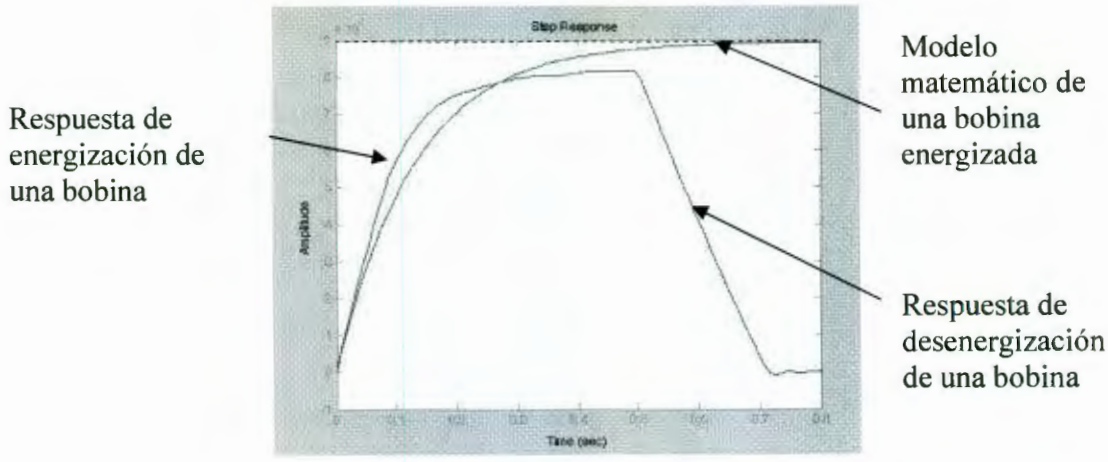


Figura 2.12. Respuesta de la energización de una bobina.

De esta manera el tiempo que carga la bobina está dado por la relación L/R, de lo cual podríamos pensar que teniendo una resistencia muy alta este tiempo disminuirá, sin embargo, la intensidad del campo magnético está principalmente limitado por el valor de R. Así que aunque tengamos una inductancia alta capaz de proporcionar un campo magnético importante, el valor de R lo limitará de tal manera que la cantidad de corriente que fluya por el devanado será baja y el campo magnético será insuficiente para mover el eje del motor. De lo anterior podemos ver que tenemos que seleccionar de acuerdo a un equilibrio de estos dos valores.

En el Capítulo 1 se definió que el par necesario para mover la mesa XY es de 17.76 oz*plg, por lo tanto ya tenemos tres parámetros importantes para seleccionar el motor: el par, el tiempo de energización y por supuesto el costo.

Desgraciadamente sólo tengo dos tipos de motores para implementar el trabajo, de los cuales hay que seleccionar el mejor para el sistema.

Tipo de motor	Motor de disco magnetizado[2]	Motor híbrido[3]
Resistencia de fase (Ω)	0.34	1.2
Corriente de carga (A)	5.2	2.9
Inductancia (mH)	0.7	2.9
Torque de detención (oz*plg)	150	33.4
Ángulo de paso (°)	1.8	3.6
Pasos por revolución	200	100

Tabla2.1. Especificaciones de los motores.

[2] La hoja de especificaciones se encuentra en el anexo 2.1
[3] La hoja de especificaciones se encuentra en el anexo 2.2

Es interesante ver que los dos cumplen con el par para mover la mesa XY, así que lo siguiente sería analizar la respuesta de los dos motores en el tiempo.
Para el motor de disco magnetizado la ecuación de la corriente quedaría como:

$$I(s) = \frac{2.94}{0.002s+1} Eo(s)$$

Siendo que para el motor híbrido es de:

$$I(s) = \frac{0.83}{.0024s+1} Eo(s)$$

Al parecer los dos tienen la misma constante de respuesta en el tiempo, ahora tendríamos que graficar para ver como se comporta la corriente.

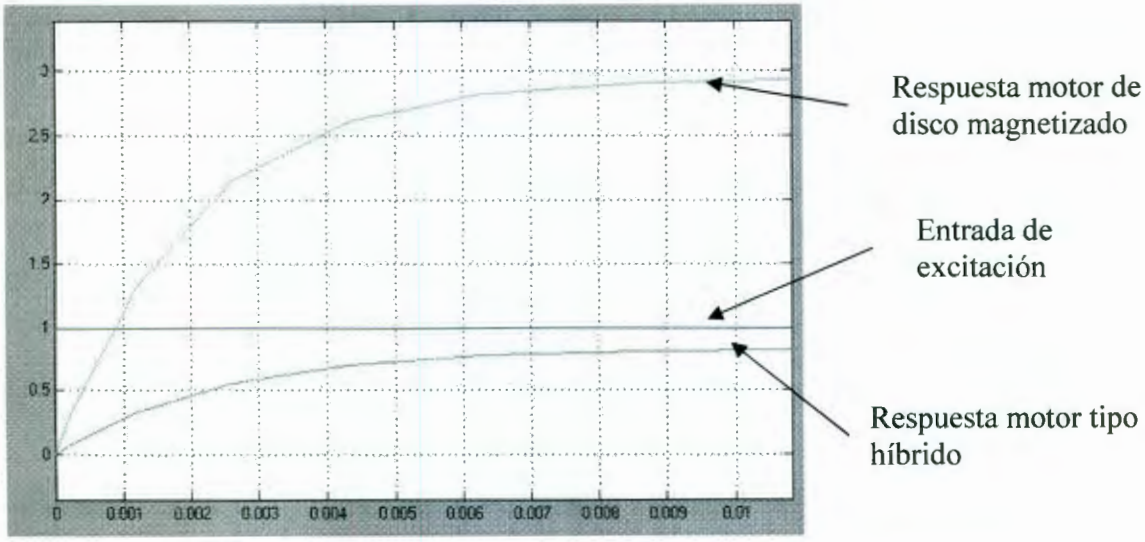


Figura 2.13. Respuesta de corriente de dos tipos de motores diferentes

De la Figura 2.13 se puede ver que a pesar de tener una constante de tiempo parecida el motor híbrido, éste se vuelve demasiado lento en el alcance de la corriente nominal. Si tuviéramos una entrada de voltaje de 3.4 V la corriente nominal la alcanzaría en 8 milésimas de segundo, claro que con un mayor par. El hecho de no alcanzar la corriente nominal de forma rápida, se puede compensar con el uso de un driver o controlador de motores a pasos especial (El cual se verá en el Capítulo 5 Diseño de la Etapa de Potencia), sin embargo, por el momento, se tomará el motor a pasos de disco magnetizado, no solo por el tiempo de respuesta sino por el costo, ya que el motor híbrido utilizado cuesta 400 dólares mientras que el motor de disco solamente 20 dólares.

Este Capítulo definió el tipo de motor a utilizar y con el sistema mecánico definido en el capítulo 1, solamente falta la comunicación entre la mesa XY con la PC. A dormir se obtendrá una forma de mover la mesa lo cual será a través de la PC lo cual se ve en el capítulo 3.

Capítulo 3

Buses y puertos de comunicación

Hoy en día el control de la mayoría de los sistemas está dado de forma digital, ya es cosa del pasado encontrar sistemas controlados analógicamente, ya sea por medio de la electricidad, electrónica o neumática. La mayoría de los procesos están siendo actualizados a sistemas digitales debido a tienen muchas bondades con los procesos (Mayor productividad, costo mínimo, mejor tiempo de respuesta, flexibilidad en los programas de control, etc.). Últimamente por el abaratamiento de los sistemas de cómputo y el desarrollo del tratamiento de las señales digitales, se ha introducido el control de los procesos en la industria por medio de la PC. En este capítulo se abordará la forma y los medios en que se comunica la computadora personal con arquitectura Intel, lo anterior, para poder tomar una decisión sobre la forma en que se dará la comunicación entre la mesa XY y la PC.

3.1. Interfaces de E/S para la PC.

La forma de comunicarse con el microprocesador se le llama interfase, pero no es un interfase hombre máquina como la conocemos sino la interfase máquina-máquina, en este caso la primer máquina es el microprocesador y la segunda es la mesa XY. Normalmente para optimizar los recursos de la PC se han manejado dos tipos de manejo de interfaces: entradas y salidas aisladas además de las entradas y salidas ubicadas en memoria las cuales ocupan memoria de forma diferente en el microprocesador. En la Figura 3.1 podemos ver algunas de estas interfaces, algunas de las cuales les llamamos ranuras de expansión.

Cuando hablamos de entradas y salidas aisladas nos referimos a circuitos que pueden almacenar, enviar o recibir el contenido completo de un byte o de un paquete de información en los registros propios sin afectar las localidades de memoria del microprocesador.

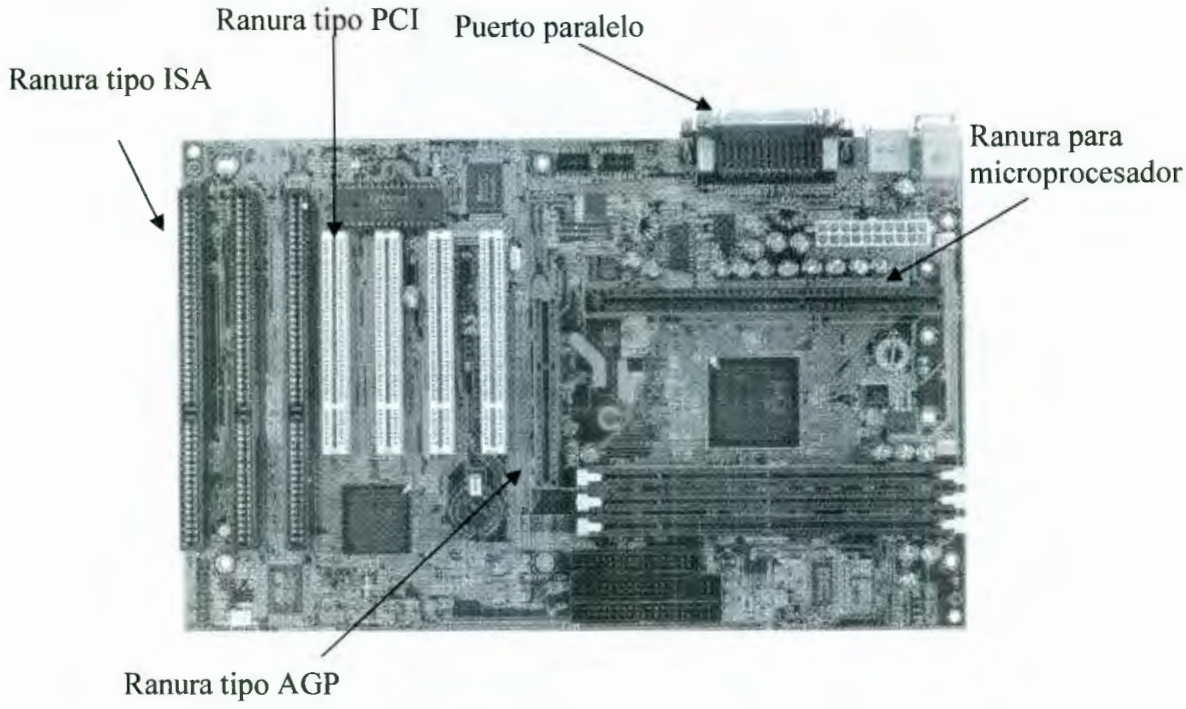


Figura 3.1. Vista de la tarjeta madre con interfaces de comunicación.

Cuando se trata de entradas y salidas ubicadas en memoria, los datos pueden accederse directamente de la memoria, claro que afectando las localidades de memoria del microprocesador (Figura 3.2). Este tipo interfaz es muy común y lo vemos en lo que generalmente se denomina como ranuras de expansión. Si el usuario quiere agregar una tarjeta de sonido, una tarjeta de vídeo, una tarjeta de red es normalmente que lo haga a través de este tipo de entradas. Mientras si quiere agregar un modem, una cámara, una impresora o un escáner lo hará a través de las entradas y salidas aisladas.

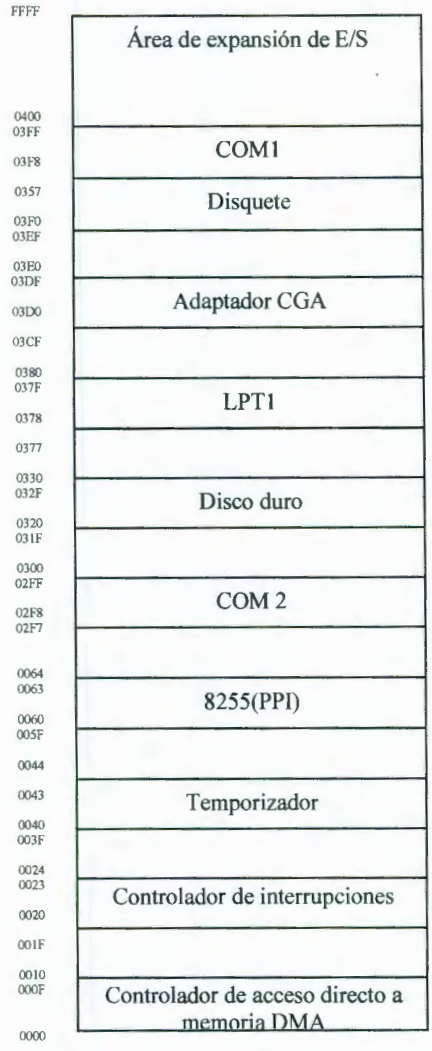


Figura 3.2. Mapa de memoria de E/S de una computadora personal.

3.1.1. E/S aislada.

La mayoría de las comunicaciones que se utilizan en los sistemas basadas en microprocesadores Intel es la de E/S aisladas. El decir aislada nos dice que las localidades de memoria de E/S están aisladas del sistema de memoria en un espacio de dirección de E/S separado. Estas direcciones son llamadas puertos. Debido a que los puertos están separados, el usuario puede expandir la memoria a su totalidad sin utilizar espacio de memoria para los dispositivos de E/S.

La forma de acceder a los datos de transferencia entre las E/S y el microprocesador se da a través de las instrucciones de ensamblador IN, INS, OUT y OUTS. Las señales de control separadas para el espacio de E/S son proporcionadas empleando M/IO (Memoria o entradas y salidas) y W/R (Escribir o

leer), las cuales indican una lectura (IORC) o una escritura de E/S (IOWC). Estas señales indican que una dirección de puerto de E/S, la cual aparece en el bus de direcciones, es utilizada para seleccionar un dispositivo de E/S. En la computadora personal, los puertos aislados de E/S son utilizados para controlar dispositivos periféricos. Una dirección de puerto de 8 bits se utiliza para acceder a dispositivos ubicados en la tarjeta de sistema, como el temporizador y la interfaz de teclado, mientras que un puerto de 16 bits se utiliza para acceder a los puertos serial y paralelo así como a los sistemas controladores de vídeo y de disco.

Dentro de este tipo de transferencia de datos podemos ver dos formas comunes en la paralela y la serial. En las transferencias de datos paralela, ocho o más líneas (conductores) son usadas para transferir datos algún dispositivo que esté a unos cuantos pies de distancia. Un ejemplo de transferencia paralela es el de las impresoras además de los discos duros los cuales tienen cables planos con muchos conductores. A pesar de que pueden transferir muchos datos, esto no puede ser en distancias grandes. Para transferir datos algún dispositivo que está a varios metros de distancia, se usa la comunicación serial. En la comunicación serial, los datos se mandan bit a bit, en contraste con la comunicación paralela, en la cual los datos se mandan byte a byte o en paquetes más grandes al momento.

3.1.1.1. Descripción del puerto serial.

Debido a que los cables largos atenúan e incluso distorsionan las señales, se ha creado una forma de comunicación la cual se llama serial. Esta forma de comunicación se utiliza para transferir datos entre dos sistemas localizados a cientos o miles de pies de distancia. El hecho que la comunicación serial se utilice un solo conductor el lugar de un conjunto de conductores ya sea de 8, 16, 32 o de la cantidad de datos que se quieran transferir, no solamente lo hace más barato sino que hace que dos computadoras se puedan comunicar a través de la línea telefónica. Para que la comunicación serial se de se necesita que la palabra o byte a transferir, hay que descomponerla en una señal serial de bits, esto sería través de un convertidor paralelo a serial, lo cual implica que el receptor debe tener un convertidor de serial a paralelo. La velocidad de transmisión modem va desde 110 hasta 56,000 bps, lo cual los indica que realmente es un puerto lento. Por lo que no es recomendable para usarlo en una aplicación donde la velocidad sea importante.

3.1.1.2. Descripción del puerto paralelo

La interfase por el puerto paralelo, contiene un conector (Figura 3.3) donde los pines están identificados desde el 1 al 36. La mayoría de los pines se usan para igualar las tierras, permitiendo que varias señales tengan su línea de tierra independiente, lo que reduce el ruido eléctrico. Agrupar los pines nos lleva a la siguiente clasificación:

1. Líneas de datos, los cuales llevan los datos de la PC a la impresora.
2. Las señales de el status, las cuales nos dicen cómo se encuentra la impresora.
3. Señales de control de impresión, son las señales que le dicen al impresora que hacer.
4. Señales de tierra, las cuales provén un regreso individual de tierra para cada línea de datos y para algunas líneas de control y status.

Respecto a la velocidad de transmisión el puerto paralelo puede hacerlo a la velocidad de 300 Kbs, lo cual implica que es más veloz que el puerto serial.

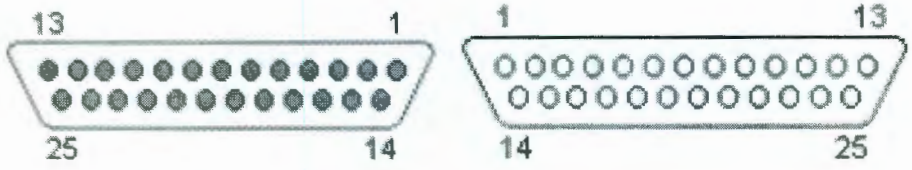


Figura 3.3. Configuración de los pines macho y hembra del puerto paralelo.

3.1.2. E/S ubicada en memoria.

A diferencia de la E/S aislada, la E/S ubicada en memoria no utiliza las instrucciones IN, INS, OUT o OUTS. En su lugar, usa cualquier instrucción que transfiera datos entre el microprocesador y la memoria. Un dispositivo de E/S ubicado en memoria es manejado como una localidad del mapa de memoria. La ventaja principal de la E/S ubicada en memoria es que puede utilizarse cualquier instrucción de transferencia de memoria para acceder al dispositivo de E/S. La desventaja principal es que una parte del sistema de memoria se utiliza como el mapa de E/S. Esto reduce la cantidad de memoria disponible para aplicaciones.

Otra ventaja es que las señales IORC e IOWC no tienen aplicación en un sistema de E/S ubicado en memoria, lo que puede reducir la cantidad de circuitos requeridos para la decodificación.

A continuación se describirá el bus ISA (Arquitectura Estándar de la Industria), el bus VESA , el bus PCI (Interfaz de Componentes Periféricos), el USB (Bus Serial Universal) y el AGP (Puerto Acelerado de Gráficos).

3.1.2.1. Descripción del bus ISA (Arquitectura Estándar del Industria)

Este tipo de puerto nos proporciona una comunicación directa con la tarjeta madre del sistema, de este modo podemos tener acceso a todas las líneas de comunicación del microprocesador (Datos, Direcciones y Control).

Presentado en un principio con un canal de datos de 8 bits, el ISA fue ampliado a un canal de 16 bits en 1984, cuando IBM lanzó al mercado el PC/AT. ISA se refiere generalmente a los propios zócalos de expansión, que se denominan zócalos (slots) de 8 bits o de 16 bits.

También es bueno hablar del bus ISA extendido (EISA) el cual es una modificación de 32 bits al bus ISA. A medida que las computadoras crecieron y tuvieron buses de datos de más bits (80386-Pentium II), fue necesario un nuevo bus que pudiera transferir datos de 32 bits. A pesar de que el bus EISA parece estar desapareciendo, representa un escalón en la evolución del bus de sistema de la computadora. La problemática del bus EISA fue que a pesar de que la cantidad de bits del bus de datos era de 32, la velocidad de reloj se mantuvo en 8 MHZ, razón por la cual este estándar de interfaz prácticamente ha desaparecido. Podemos ver en la actualidad que los buses VESA local y PCI operan actualmente a 33 MHZ. El bus EISA se utiliza principalmente como uno controlador de disco o adaptador de vídeo gráfico. Estas aplicaciones se ven beneficiadas por un bus de datos más ancho, porque las frecuencias de transferencias de datos para estos dispositivos son altas.

En las computadoras modernas se ve que las tarjetas madre ya no hay ranuras para conectar tarjetas tipo ISA. Esto podría decirnos que el bus ISA esta desapareciendo, pero no es así. El bus ISA es la forma más transparente de comunicación con el microprocesador, además que su velocidad de transmisión de datos es superior a la de los puertos paralelo y serial. En la industria todavía se piden computadoras industriales las cuales en sus especificaciones esta incluida la cantidad de ranuras para tarjetas tipo ISA. Por lo anterior a pesar de que en las computadoras de uso personal ya no se encuentre el bus ISA disponible, podemos encontrarlo en los procesos industriales.

3.1.2.2. Descripción del bus VESA

El bus VESA es una mejor opción para el establecimiento una interfaz de 32 bits. El bus ISA extendido opera solamente a 8 MHZ. Esto significa que las aplicaciones que requieren transferencias de datos a alta velocidad se benefician del bus VESA local. Esta característica del bus VESA se utiliza normalmente para interfaces entre vídeo y disco a la computadora personal. Como se puede denotar el bus VESA local también es una extensión del bus ISA. La diferencia radica en el que el bus VESA local no agrega nada en los conectores ISA de 16 bits; en su lugar, se añade un tercer conector (conector VESA) detrás del conector ISA de 16 bits.

3.1.2.3. Descripción del bus PCI

El bus PCI (Interfaz de Componentes Periféricos) es comúnmente el único bus que se encuentra en los sistemas más recientes basados en el Pentium II. En las computadoras nuevas de escritorio se puede ver que en las tarjetas madre

solamente se encuentran ranuras para insertar tarjetas tipo PCI y AGP. Podríamos preguntarnos el porqué de ello, la razón es porque el bus PCI tiene características "Plug and Play" lo cual permite instalar de forma más fácil el dispositivo que se conecte a este bus, además de que el bus de datos es más amplio el cual es de 64 bits. Una interfaz PCI contiene una serie de registros, alojados en un dispositivo de memoria en la interfaz PCI, que contiene información acerca de la tarjeta. La información en estos registros permite a la computadora configurar automáticamente la tarjeta PCI.

La estructura del sistema para el bus PCI es la siguiente, el bus del microprocesador está separado y es independiente del bus PCI. El microprocesador se conecta al bus PCI por medio de un circuito integrado llamado puente PCI. Esto significa que puede establecerse una interfaz desde prácticamente cualquier microprocesador al bus PCI, siempre y cuando se diseñe un puente o controlador PCI para el sistema, lo cual permite multiprocesamiento en tiempo real más eficiente que en el BUS ISA. En la Figura 3.4 se puede ver el sistema de comunicación entre el microprocesador y el bus PCI.

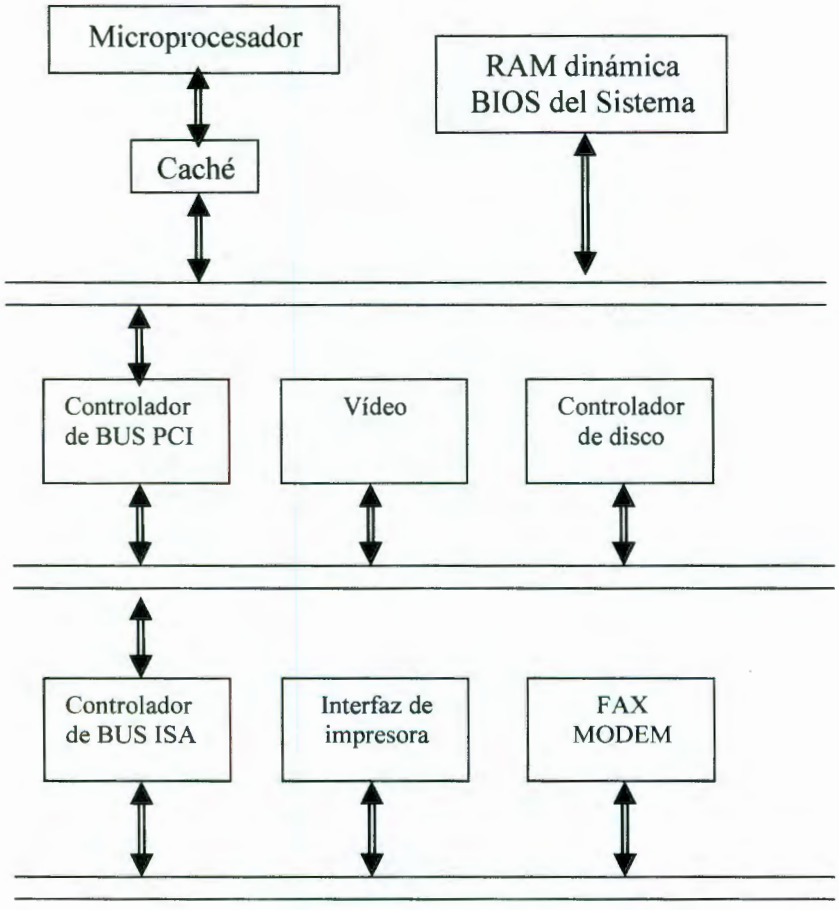


Figura 3.4. Diagrama de bloques para la computadora personal con BUS PCI.

El bus PCI contiene todas las señales de control del sistema. A diferencia de otros buses, el PCI funciona tanto como un bus de datos de 32 bits como uno de 64 bits, y con un bus de direcciones completo de 32 bits. Para reducir el tamaño del conector las tarjetas los buses de direcciones y datos están multiplexados. Las terminales multiplexadas están etiquetadas como AD0-AD63 en el conector. Si se trabaja en el bus PCI con 32 bits sólo se utilizarán las conexiones de la 1 a la 62, pero si se usan los 64 bits se usarán las 94 conexiones. Lo interesante de trabajar la dirección a 64 bits es que en el futuro puede ser que se utilice.

3.1.2.4. Descripción del USB (Bus Serial Universal)

El bus serial se puede considerar en velocidad a cuatro puertos seriales, pero la ventaja principalmente de este bus es que el USB permite que el dispositivo a conectar tenga su propia fuente de alimentación, eliminando el ruido asociado con la fuente de la PC permitiendo una comunicación sin distorsiones. Asimismo la conexión se hace más fácil debido a que los buses USB generalmente tienen terminales en la parte exterior de la PC. Una de las limitaciones del bus USB es que sólo se puede conectar 127 dispositivos a una PC. Este tipo interfaz se utiliza en teclados, ratones, tarjetas de sonido, escáners, cámaras y modems. Para el USB 1 las velocidades de transferencia de datos son de 12Mbps a velocidad completa y 1.5 Mbps para operación a baja velocidad. Las longitudes de cable están limitadas a cinco metros como máximo para la interfaz a velocidad completa y a tres metros para el interfaz a baja velocidad. El mismo bus lleva alimentación de voltaje a 5 V y tierra, la corriente máxima disponible es de es de 100 mA. Si la cantidad de corriente sobrepasa los 100 mA el sistema lo detectará y emitirá una señal de alarma. Para el USB 2 se tiene una velocidad mayor de hasta 10 veces más que el USB 1 sin embargo todavía los controladores y varios dispositivos no lo reconocen. Las señales de datos del USB son bifásicas. Aunque el USB parece una buena opción para la automatización, este bus no está estandarizado en la industria así que no es recomendable su uso en proyectos industriales de importancia debido a que solamente se usan en la industria dispositivos que se ha comprobado que funcionan plenamente.

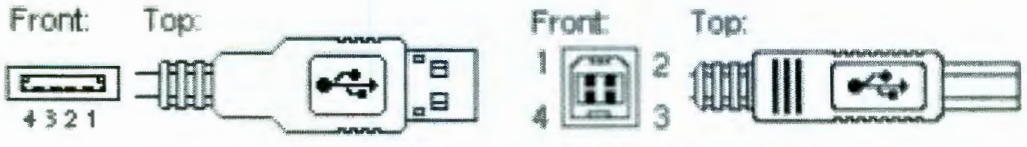


Figura 3. 5. Conectores USB a la PC y al dispositivo.

3.1.2.5. Descripción del AGP (Puerto Gráfico Acelerado)

Este bus funciona a la frecuencia de reloj del bus del microprocesador. Está diseñado de tal forma que una transferencia entre la tarjeta de vídeo y la memoria del sistema pueda realizarse a velocidad máxima. El AGP puede transferir datos a

una velocidad máxima de 528 en MB por segundo en el sistema 1 X, en el sistema 4 X la velocidad es mayor 1 GB, en comparación con el bus PCI que tiene una velocidad máxima de transferencia de 100 MB.

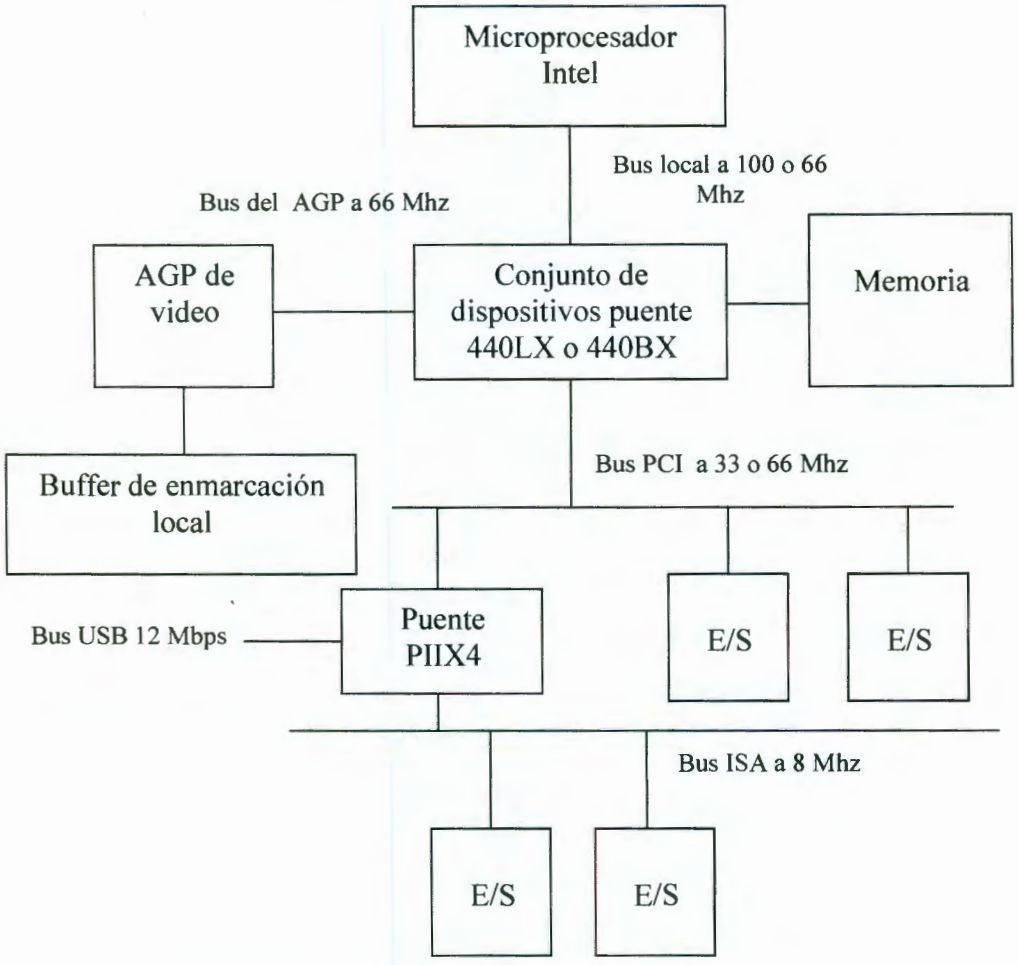


Figura 3.6. Estructura de una computadora moderna mostrando todos los buses.

3.2. Selección de la interfaz a trabajar

Se han descrito varias interfaces para poder comunicar al microprocesador con la PC, la selección será en base al protocolo y velocidad de comunicación, al costo de la implementación y capacidad de crecimiento.

En cuanto a velocidad del puerto serial podemos decir que la velocidad de transmisión de datos es muy baja, mientras que la velocidad puerto paralelo es mayor pero también es muy limitado para un proceso continuo. Del bus ISA podemos decir que la velocidad es mucho mayor pero se queda opacado por la velocidad de los demás buses. Sin embargo se puede ver que el puerto USB no se puede utilizar por no está estandarizado la industria, el bus PCI requiere de una

interfaz más elaborada pues requiere de un protocolo de petición y respuesta, por lo que su implementación es más elaborada. En cuanto al bus AGP, probablemente nunca se utilizará como un dispositivo diferente al procesamiento de video pero es bueno recordar que existe un bus que puede transferir datos a la misma velocidad del microprocesador. Por todo lo anterior, se seleccionará el bus ISA, no tomando en cuenta los buses EISA y VESA los cuales son solamente ampliaciones del anterior. Ya hecha la selección de la interfaz con el microprocesador sólo queda describirla para poder empezar a trabajar con ella.

3.3. Descripción de las señales del bus isa.

El bus ISA tiene 62 contactos de tipo de borde de tarjeta, 31 por cada cara (A/B), en pasos de 0.1 pulgadas. De las señales que pueden ser utilizadas en el trabajo son las siguientes :

- **A0-A19 (bus de direcciones) :** Son 20 líneas que determinan el máximo de memoria direccionable (1Mbyte). "A0" es el bit menos significativo y "A19" es el más significativo. Las señales de salida se generan por el microprocesador o por el controlador de DMA cuando este toma el control sobre el bus, se usan los contactos A12-A31 del bus.

- **DB0-DB7 (bus de datos) :** Por estas líneas se hacen las peticiones de entrada al sistema el cual es de 8 bits. Y sus terminales son de la 2 a 9.

Las líneas de control de lectura/escritura (R/W).-Las siguientes son para acceso a memoria:

- **OIR (salida):** indica a los periféricos la lectura de dato situado en el Bus. Puede ser Controlada por el procesador o por el controlador DMA; activa en nivel bajo.
- **OIW (salida):** indica a los periféricos la escritura de un dato situado en el bus. Puede ser controlada del mismo modo, y es activa también en nivel bajo.

Seis líneas para acceso directo a Memoria (DMA).-

- **AEN (salidas) :** cuando ésta es activa, el DMA controla el bus de direcciones, bus de datos y líneas de R/W.

Cuatro niveles de tensión de alimentación. **+5,-5, +12 y -12** volts de corriente continua.

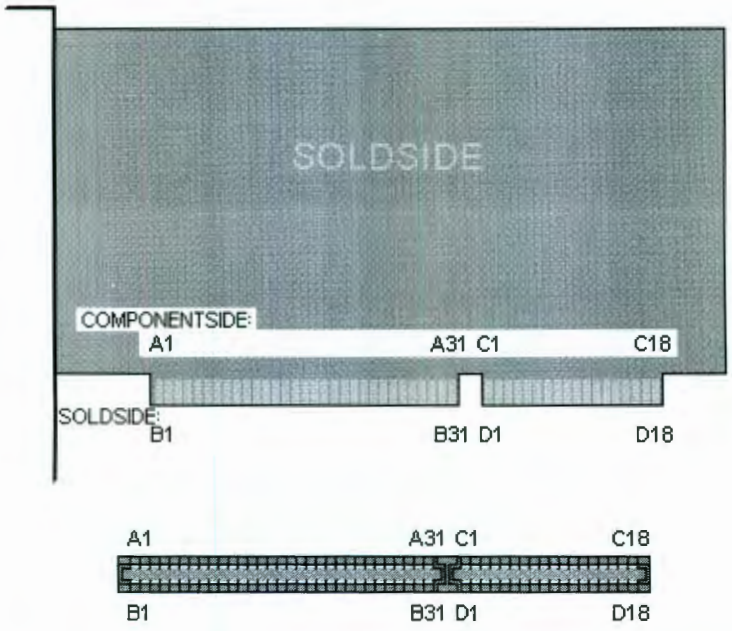


Figura 3.7. Vistas de una tarjeta ISA y conector de la computadora.

3.4. Mapa de E/S de una computadora personal

La computadora personal utiliza parte del mapa de E/S para funciones dedicadas.. Normalmente el espacio de E/S entre las direcciones OOOOH y 03FFH está normalmente reservado para el sistema de cómputo y el bus ISA. Los puertos de E/S ubicados en 0400H a FPFFH generalmente están disponibles para aplicaciones del usuario, funciones de la tarjeta principal y el bus PCI. Observe que el coprocesador aritmético 80287 emplea las direcciones de E/S OOF8H a OOFFH para comunicaciones. Por este motivo, Intel mantiene reservados los puertos de E/S OOFOH a OOFFH. Los microprocesadores 80386 al Pentium II utilizan los puertos de E/S 800000F8H a 800000FFH para comunicación con sus coprocesadores. A los puertos de E/S ubicados entre OOOOH y OOFFH se accede por medio de las instrucciones de E/S de puerto fijo; a los puertos ubicados por arriba de OOFFH se accede con las instrucciones de E/S de puerto variable[1]. Por lo general no importando el tipo de CPU, el rango de direcciones disponibles en memoria se acota de la 300H a la 31FH para las tarjetas prototipo conectadas a un puerto de expansión. Por lo que la dirección de memoria que debe tomar la tarjeta la definiré como la 300H.

[1] Barry B. Brey,2000.

3.5. Interfaces básicas de entrada y de salida

El dispositivo básico de entrada es un conjunto de buffers de tres estados. El dispositivo básico de salida es un conjunto de registros transparentes de datos. El término IN se refiere al movimiento de datos desde un dispositivo de E/S al microprocesador, y el término OUT al movimiento de datos desde el microprocesador al dispositivo de E/S.

3.5.1. Interfaz de entrada.

Los buffers de tres estados (alto, bajo y alta impedancia) son utilizados para construir el puerto de entrada. El buffer de tres estados permite al microprocesador leer el contenido de las señales que van al bus de datos, cuando se genere una señal de selección SEL. De esta forma, al ejecutar la instrucción IN, se genera una habilitación para leer el contenido del bus de datos y copiarlo al registro AL. La forma en que genera la señal SEL es la siguiente: Cuando se ejecuta una instrucción IN de algún programa, la dirección del puerto de E/S se decodifica a través de algún circuito lógico para generar la señal SEL lo cual habilita el buffer y puede "transferir" las señales de unos y ceros. Cuando al señal habilitadora SEL esta desactivada el buffer pone en alta impedancia sus salidas, protegiendo así el circuito y permitiendo que otros dispositivos utilicen el bus de datos. Aunque en este trabajo no se requiere usar entradas al microprocesador definí esta interfaz para mostrar lo fácil que es interactuar con la PC a través del bus ISA.

3.5.2. Interfaz de salida.

Esta interfaz toma valores del bus de datos y generalmente debe mantenerlos hasta que nuevamente se haga una petición de acceso. La petición de acceso se logra mediante un programa que pueda manejar los registros del microprocesador como por ejemplo el lenguaje ensamblador o el lenguaje C. Por ejemplo cuando el microprocesador ejecuta una instrucción OUT en el lenguaje ensamblador, los datos están presentes en el bus de datos únicamente por menos de 1 µs. Sin un biestable o retenedor como el circuito 74374, la salida del microprocesador no se mantendría el tiempo suficiente para ejercer alguna acción.

En este capítulo se definió la interfaz de salida a trabajar y cómo se va a hacer. En el capítulo 4 se mostrará la implementación de los circuitos necesarios para la transmisión de datos a través del bus ISA.

Capítulo 4

Diseño de la tarjeta de control de movimiento.

Este capítulo aborda los criterios que se utilizan para el diseño de la tarjeta control de movimiento, uno de ellos es el de no causar conflictos en el desempeño de la computadora ya que de repetir una dirección para poder transferir los datos, podríamos haber cambiado la información de algún otro dispositivo. Asimismo se definen los dispositivos para decodificar las direcciones, para mantener los datos hasta que sean cambiados nuevamente por el programa y el dispositivo que decodificara las instrucciones del bus de datos.

Una vez recibido el dato, el cual se recibe como tres señales digitales las cuales nos indican la frecuencia de pulso, la dirección del motor y si se dan pasos completos o medios pasos. Con estas tres señales se generara una secuencia de energización de las bobinas para que se dé la rotación de la flecha.

4.1. Decodificación de direcciones.

La decodificación de direcciones se utiliza para habilitar algún chip en este caso será un latch (retenedor), el cual mantiene a la salida un valor que tuvo en un determinado tiempo en la entrada, y el cual solamente cambiará con una señal de cambio o habilitación. La razón por la que se hace esto es muy simple: Debido a que el bus de direcciones cambia constantemente ya que el microprocesador direcciona[1] constantemente a los diferentes dispositivos, el bus de datos mantiene el valor solamente el tiempo que dura el direccionamiento Es así como podemos ver que si tenemos diferentes dispositivos, el valor del bus de datos cambiará respecto a los datos que se procesen por cada dispositivo, así que es necesario que se conserve el valor del bus de datos en nuestro dispositivo y que cambie solamente cuando se le indique hacerlo. Como se definió en el capítulo 3 la dirección que utilizaremos para la interfaz será la 300H para las señales de control de los motores X y Y, además de tomar una adicional la cual será la 301H para el manejo del eje Z.

La función lógica que realiza esta habilitación se puede escribir como mintérminos o maxtérminos. Si deseamos asignar el puerto 300H a las entradas digitales la función lógica seria:

[1] Llama por medio de una dirección.

En mintérminos

CS = A9 & A8 & !A7 & !A6 & !A5 & !A4 & !A3 & !A2 & !A1 &!A0 & !AEN & !IOW & IOR

CS2 = A9 & A8 & !A7 & !A6 & !A5 & !A4 & !A3 & !A2 & !A1 &A0 & !AEN & !IOW & IOR

Donde CS es el habilitador del chip (chip select) de los motores de los ejes X y Y, CS2 es el chip select del motor del eje Z, A9 hasta A0 nos indican el número de dirección del dispositivo, IOW e IOR nos indican si se está leyendo o escribiendo siendo la habilitación de leer o escribir negada, por último tenemos a AEN el cual nos ayudara a evitar conflictos cuando se realiza un ciclo DMA. La figura 4.1 nos describe la operación de la decodificación:

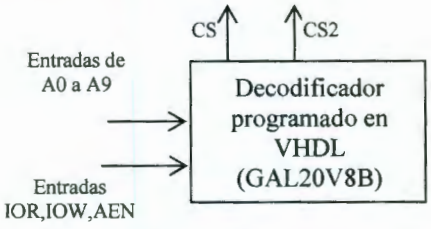


Figura 4.1. Diagrama esquemático del decodificador.

4.1.1. Selección del dispositivo para la decodificación de direcciones.

El circuito seleccionado para hacer la decodificación será un dispositivo lógico programable (PLD) debido a que se puede obtener la misma funcionalidad con un circuito integrado en vez de usar varios chips lógicos individuales. Esto significa menos espacio en la tarjeta, menores requerimientos de potencia, mayor confiabilidad, menor inventario, y menos costo global de manufactura. La forma de seleccionar un PLD es mediante la cantidad de macroceldas lógicas de salida las cuales generan los productos de los términos de entrada. En este caso se utiliza una GAL20V8B[2] debido a que:

- No se requiere una gran cantidad de macroceldas para el procesamiento de las entradas.
- Facilidad de encontrarla en el mercado
- Bajo costo (el cual es menor de 20 pesos)

[2] La hoja de especificaciones se encuentran en el anexo 4.1.

- Permite que se tengan 20 entradas y por lo menos ocho salidas (La cantidad de entradas y salidas de una GAL20V8B es configurable).

4.1.2. Programación del PLD para la decodificación de direcciones.

La información es enviada a las entradas de una GAL20V8B, la cual las procesará y habilitará un latch (retenedor) que mantiene el valor de los ejes X y Y o para el eje Z.

Aunque existen varios tipos de software para programar PLDs preferí usar el lenguaje VHDL[3]. Ya que otros tipos de software para programar como el CUPL no están desarrolladas lo suficiente como para ser amigable en la programación.

A continuación se muestra parte del programa en VHDL el cual hace las funciones lógicas de selección de dispositivo[4].

Siendo las entradas a procesar:

```
Entradas: in STD_LOGIC_VECTOR (13 downto 1);     --Declarada como vector
CS: out STD_LOGIC;                               --Declarada como pin
CS2: out STD_LOGIC
```

Y el programa:

```
process(Entradas)
begin
      case Entradas is
            when "1100000000010" => CS <= '1'; --Cuando se dé en la com-
                                               --binación en mintérminos
                                               --CS=1
            when others => CS <= '0'  ;        --En otros casos CS=0
      end case;
end process;
```

Cuando se programa en VHDL hay que obtener el archivo con extensión JED o también llamado mapa de fusibles para que un programador de PLDs pueda transferir el programa a la GAL. La forma de que se genere el archivo JED es mediante el programa Galaxy. Al momento de crear el archivo JED se obtiene un archivo adicional, el cual tiene la extensión de RPT. Este archivo es un reporte el cual nos indica como conectar los pines de la GAL. El software que se usó en este trabajo para poder grabar la GAL20V8B fue el WINLV. A continuación se muestra parte del archivo RPT[5] en donde podemos ver como quedaron las terminales del

[3] Lenguaje Visual Descriptivo de Hardware
[4] El código completo se muestra en el anexo 4.2.
[5] El contenido del archivo se muestra en el anexo 4.3.

circuito para su conexión y cuales fueron las ecuaciones para la salida de selección de los chips.

Las ecuaciones simplificadas para la programación generadas por el Galaxy son:

```
DESIGN EQUATIONS            (22:01:41)


    cs =
          entradas_13 * entradas_12 * /entradas_11 * /entradas_10 *
          /entradas_9 * /entradas_8 * /entradas_7 * /entradas_6 *
          /entradas_5 * /entradas_4 * /entradas_3 * entradas_2 *
          /entradas_1

    cs2 =
          entradas_13 * entradas_12 * /entradas_11 * /entradas_10 *
          /entradas_9 * /entradas_8 * /entradas_7 * /entradas_6 *
          /entradas_5 * entradas_4 * /entradas_3 * entradas_2 *
          /entradas_1
```

Y la configuración de pines es:

```
Completed Successfully

                                C20V8A
                    ____________________________________
     entradas_1 =| 1|                                |24|* not used
     entradas_2 =| 2|                                |23|* not used
     entradas_3 =| 3|                                |22|= cs2
     entradas_4 =| 4|                                |21|* not used
     entradas_5 =| 5|                                |20|* not used
     entradas_6 =| 6|                                |19|* not used
     entradas_7 =| 7|                                |18|* not used
     entradas_8 =| 8|                                |17|* not used
     entradas_9 =| 9|                                |16|* not used
    entradas_10 =|10|                                |15|= cs
    entradas_11 =|11|                                |14|= entradas_13
       not used *|12|                                |13|= entradas_12
                    ____________________________________
```

4.2. Conservación de la información transferida al bus de datos.

Al momento que el microprocesador ejecuta la instrucción y selecciona la dirección a la cual le va mandar los datos el dato transmitido solamente se mantiene por 1µs en el bus de datos. Así que al programar la GAL20V8B para que arroje una señal alta cuando las direcciones 300H o 301H fueran direccionadas sirve para activar un circuito de latch el cual actualizará la entrada del bus de datos de la mesa XY, y no importará que aunque el bus de datos de la computadora cambie constantemente debido a que también tiene que comunicarse con otros dispositivos, solamente se actualizará el valor que el programa arroje por la

dirección de la mesa XY o del eje Z. El circuito que se utilizará para este trabajo será el 74LS373[6].

4.3. Generación de secuencias para motores a pasos usando un PLD.

Una vez retenido el valor obtenido del bus de datos, éste debe ser procesado para poder generar secuencias para las bobinas del motor a pasos. Para ello serán necesarias las señales del sentido de rotación del motor, pasos completos o a medios pasos, la velocidad de rotación (La cual está dada por la frecuencia de estos pulsos) y si es necesario una señal de reset. En éste trabajo se utilizará la señal de reset para colocar un interruptor de límite el cual asegurará que la mesa no se salga de sus ejes. Para generar las secuencias de las bobinas hay dos formas de hacerlo: Programar un dispositivo programable y usar un circuito comercial.

4.3.1. Generación de secuencias para motores a pasos usando un PLD.

Estas tres señales de control como se hizo con la decodificación de datos fueron introducidas en un PLD que soporta un programa escrito en VHDL para las secuencias de bobinas descritas en el Capítulo 2 Motores a pasos. En la Figura 4.2 se observa lo que se espera que arroje la GAL22V10 al momento de obtener las señales de entrada.

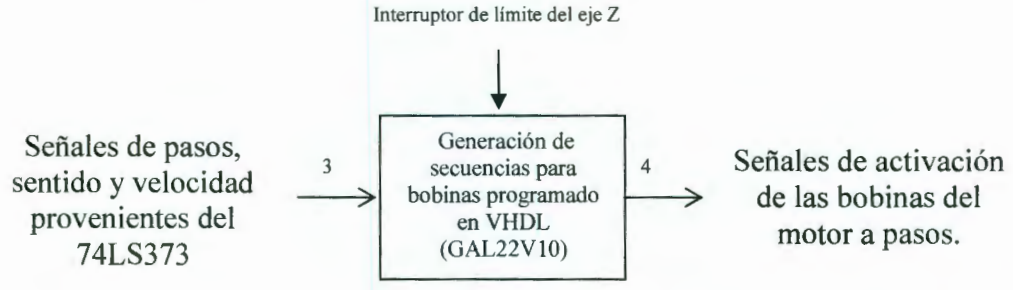


Figura 4.2. Diagrama esquemático de la función de la GAL22V10.

Parte del programa escrito en VHDL que genera una secuencia de bobinas de acuerdo a la señal de reset, sentido, medios o pasos completos, y frecuencia o velocidad del motor[7] es el siguiente:

---

[6] Hoja de especificaciones se encuentra en el anexo 4.4.
[7] El código completo se muestra en el anexo 4.6.

Las entradas físicas son las siguientes:

```
clk: in STD_LOGIC;
clr: in STD_LOGIC;
direccion: in STD_LOGIC;
paso_comp_medio: in STD_LOGIC;
y: out STD_LOGIC_vector(3 downto 0)
```

Las señales internas de procesamiento son las siguientes:

```
signal estado_presente,
       estado_proximo:std_logic_vector(3 downto 0);
```

Y parte del código de procesamiento es:

```
if (direccion='1'and paso_comp_medio='1')then
      case estado_presente is
            when "0000"=> estado_proximo <="0001";
            when "0001"=> estado_proximo <="0011";
            when "0011"=> estado_proximo <="0010";
            when "0010"=> estado_proximo <="0110";
            when "0110"=> estado_proximo <="0100";
            when "0100"=> estado_proximo <="1100";
            when "1100"=> estado_proximo <="1000";
            when "1000"=> estado_proximo <="1001";
            when "1001"=> estado_proximo <="0001";
            when others=> estado_proximo <="0000";
      end case;

y<=estado_presente;
```

La simulación del circuito programada en VHDL se muestra en la Figura 4.3.

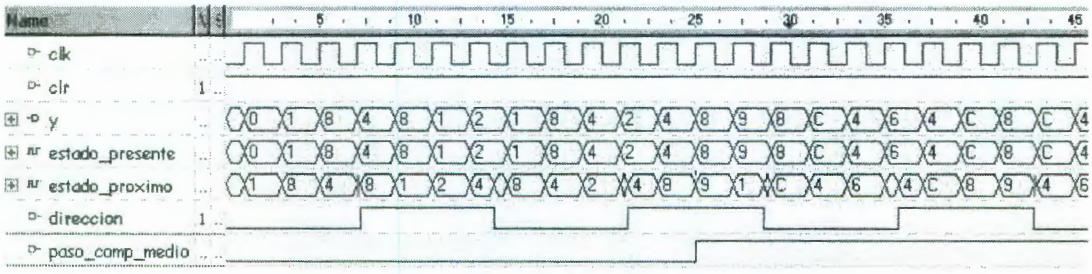


Figura 4.3. Simulación en VHDL de un generador de secuencias para energizar bobinas de motores a pasos.

Al tratar de generar el archivo JEDEC para cargarlo en la GAL20V8B pude ver que no podía soportar el programa ya que el compilador del Galaxy marcaba demasiados errores a pesar de solamente tener solamente cuatro señales a procesar. Este error es debido a que la GAL20V8B no tiene las suficientes macroceldas lógicas de salida para hacer las operaciones, así que utilicé un dispositivo diferente que tiene más macroceldas, la GAL22V10[8].

Parte del archivo RPT[9] para el generador de secuencias se muestra a continuación:

Ecuaciones generadas por el Galaxy para las salidas:

```
DESIGN EQUATIONS            (22:05:43)


    y_3.D =
          paso_comp_medio * y_3.Q * /y_2.Q * /y_1.Q * /y_0.Q
        + direccion * y_2.Q * /y_1.Q * /y_0.Q
        + /direccion * /y_2.Q * /y_1.Q * y_0.Q

    y_3.AR =
          /clr

    y_3.SP =
          GND

    y_3.C =
          clk
```

Y la configuración de entradas y salidas del circuito es:

```
                              C22V10

                 ____________________________________
          clk =| 1|                                |24|* not used
          clr =| 2|                                |23|= y_1
paso_comp_me.. =| 3|                                |22|= y_3
     direccion =| 4|                                |21|* not used
      not used *| 5|                                |20|* not used
      not used *| 6|                                |19|* not used
      not used *| 7|                                |18|* not used
      not used *| 8|                                |17|* not used
      not used *| 9|                                |16|* not used
      not used *|10|                                |15|= y_2
      not used *|11|                                |14|= y_0
      not used *|12|                                |13|* not used
                 ____________________________________
```

[8] Hoja de especificaciones se encuentra en el anexo 4.5.
[9] El contenido del archivo se muestra en el anexo 4.6.

4.3.2. Generación de secuencias para un motor a pasos mediante el circuito L297.

Otra opción para la generación de secuencias es mediante circuitos comerciales, en este apartado sólo se discutirá el L297[10] debido a que su arquitectura está muy aceptada en el ámbito de la ingeniería al momento de generar secuencias de bobinas en especial si se usa con su circuito complemento de potencia: el L298. En el mercado se encuentra un componente que contiene ya los dos elementos: el EDE 12400[11](Figura 4.4). En este trabajo se usará el L297 para poder escoger por separado la parte de potencia.

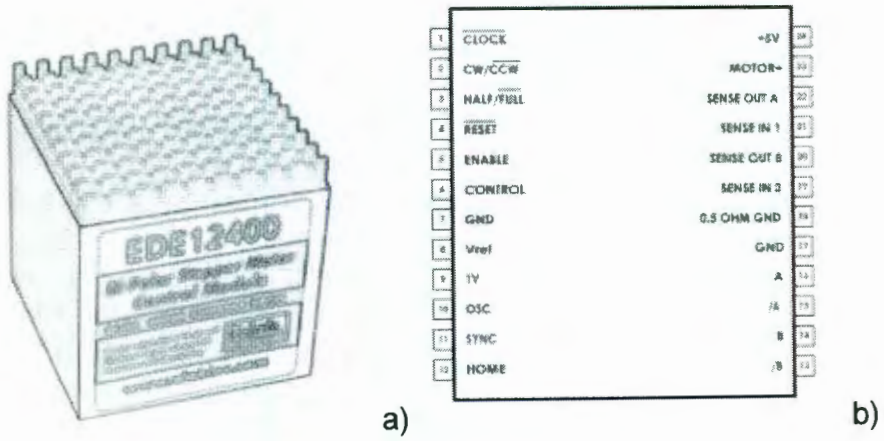


Figura 4.4. El componente EDE 12400. a) Encapsulado, b) Diagrama de pines

El circuito L297 (Figura 4.5) es capaz de generar 4 u 8 códigos, según el modo de funcionamiento que preseleccionemos (paso entero, con 1 o 2 fases activas, o medio paso). Además dispone de un circuito de PWM[12] de chopeado que realizará el control de la corriente que fluye por el motor al funcionar, cuya frecuencia de corte podrá ser definida por el usuario.

[10] Hoja de especificaciones se encuentra en el anexo 4.7.
[11] Hoja de especificaciones se encuentra en el anexo 4.8.
[12] Pulse Width Modulator. Se define en el Capítulo 5 Etapa de Potencia.

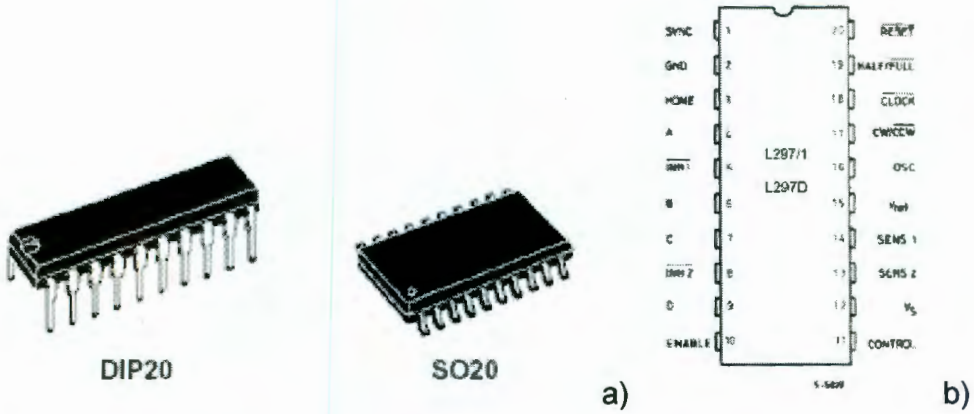


Figura 4.5. El L297. a)Tipos de encapsulado b)Diagrama de conexiones.

4.3.2.1. Esquema descriptivo del L297.

Internamente y de forma esquemática, este chip tendrá los siguientes elementos (Figura 4.6):

- Traductor (translator).
- Lógica de salida (Output Logic)
- Comparadores
- Flip-Flops
- Osciladores

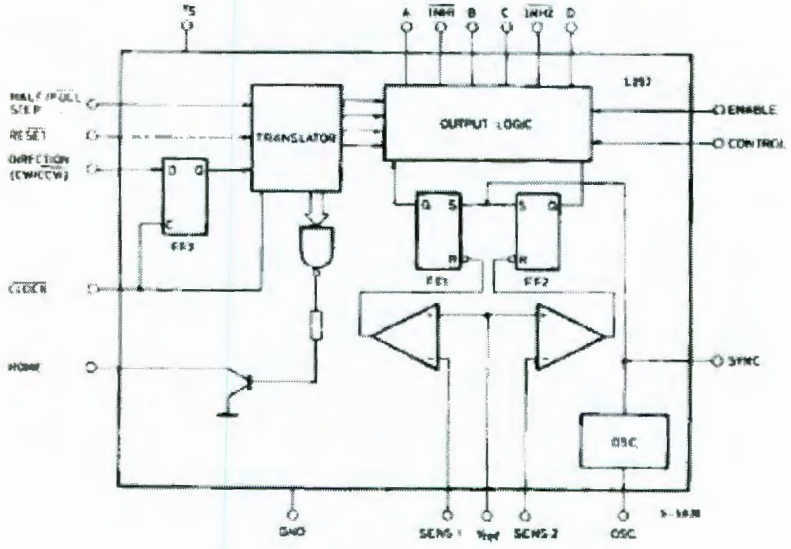


Figura 4.6. Diagrama esquemático del L297.

Debido a su arquitectura, este componente sólo necesita que le proporcionemos, además de la alimentación, una señal de reloj (con la que enviará los códigos al

circuito de potencia, y de ahí al motor), la dirección de giro y las señales de control diversas (inicialización, habilitación, etc).

4.3.2.2. Usos comunes del L297.

El L297 se usa normalmente con puentes H[13] como el L298 para las actividades de control en lazo abierto de motores por pasos. En realidad el L297 realiza una doble función, a saber:

- Generar los códigos de funcionamiento necesarios para los giros del motor.
- Realizar una regulación de la corriente por las bobinas del motor, mediante un circuito de chopeado que le permite al motor funcionar a mayor velocidad, independientemente del tiempo de respuesta del motor dado por t=L/R.

4.3.2.3. Descripción del circuito L297.

El pin SYNC es la salida del oscilador de chopeo (Figura 4.7). Esta salida permitirá sincronizar varios L297 en cascada eliminando el ruido por defasamiento y, de paso, ahorrándonos componentes en un sistema en el que usemos varios L297 en cascada, a saber:

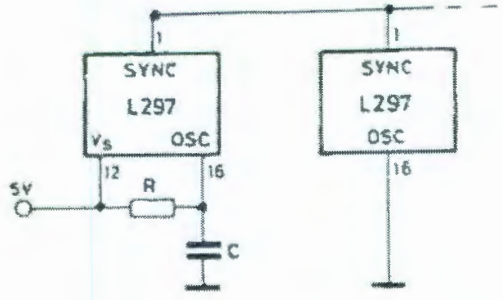


Figura 4.7. Salida del oscilador de chopper.

El pin HOME es también una salida (en colector abierto, ya que el transistor dará circuito abierto cuando el pin esté activo) que será activa cuando ABCD=0101.

Los pines INH1* e INH2* activas a nivel bajo inhiben el control de un bobinado (A y B para el primero y C y D para el segundo).

Cuando usamos el puente en H de forma bipolar, estos pines podrán usarse para asegurar una rápida recirculación de la corriente en los devanados cuando éstos se encuentran en un proceso de desenergetización. También pueden ser usados

[13] El puente H se define en el Capítulo 5 Etapa de potencia.

por el subcircuito de chopeado para regular la corriente por los devanados, siempre y cuando el pin CONTROL se encuentre a nivel bajo. Este pin (CONTROL) es una entrada que define la activación del chopeo. A nivel bajo, el chopeo actúa por INH1* e INH2* como se dijo y cuando el nivel es alto la acción de regulación de corriente por chopeado se hará por los pines ABCD.

Los pines SENS1 y SENS2 nos permitirán fijar la referencia que determinará la corriente de paso en los devanados mediante una resistencia de shunt[14].

El pin ENABLE a nivel "H" hará que INH1*,INH2*, A, B, C y D permanezcan a nivel bajo, o sea deshabilita el L297.

El pin OSC mediante una red RC determinará la frecuencia de chopeo mediante la relación:

$$fr=1/(0{,}69\ Ro\ C)$$

El pin CW/CCW es una entrada con la que determinaremos un sentido dextrógiro si se pone a nivel "H" o levógiro si lo ponemos a nivel "L".

El pin CLOCK* determinará la velocidad con la que daremos los pasos en función de la frecuencia de entrada de la señal que le introducimos.

El pin HALF/FULL determinará el modo de trabajo del motor; así si está a nivel "H" trabajará a medio paso, y si está a nivel "L", lo hará a paso completo, seleccionándose si es a 1 ó 2 fases activas en función de si estamos (en medio paso) en posición par o impar (en el cambio a paso completo). Por último, el pin RESET puesto a nivel "L" repondrá en los devanados la posición inicial ABCD=0101 (home).

En dos de las diferentes formas de trabajo con el motor (paso completo con una o dos fases activas y medio paso) se generan las señales INH1* e INH2* por el L297. Éstas se conectarán directamente al L298, obviamente a los correspondientes pines (6 y 11), con el fin de deshabilitar las correspondientes secciones del puente en H (dejarlos en alta impedancia) y con ello permitir una rápida circulación de la Im de las bobinas del motor cuando están desenergizadas.

El pin de CONTROL determinará que el subcircuito de chopeado actúe sobre las salidas A, B, C y D o las INH1* e INH2* en función de la forma de uso del motor, respecto de la recirculación de la corriente por el puente (de forma rápida o lenta). Los modos de trabajo, como ya se vio, son tres (aunque aquí sólo se han visto dos), donde el avance de las posiciones del motor (códigos emitidos) se hará en el paso del estado bajo al estado alto de la señal CLOCK.

[14] La resistencia de shunt o en derivación se usa para sensar o medir una corriente.

4.4. Selección de circuito para la generación de secuencias.

La ventaja del L297 sobre la GAL22V10 es la posibilidad de la limitación de corriente para las bobinas cuando se alimentan con un voltaje mayor. Por lo anterior la selección es de dos L297 para los ejes XY, y para el eje Z la generación con una GAL22V10.

4.5. Implementación de la tarjeta de control de movimiento.

Como se puede ver la selección del generador de secuencias usando el L297 es un buen complemento del L298 o etapa de potencia por lo que es natural que se vea en el Capítulo 5. La figura 4.8 muestra un diagrama esquemático tentativo de la configuración de la tarjeta de control de movimiento.

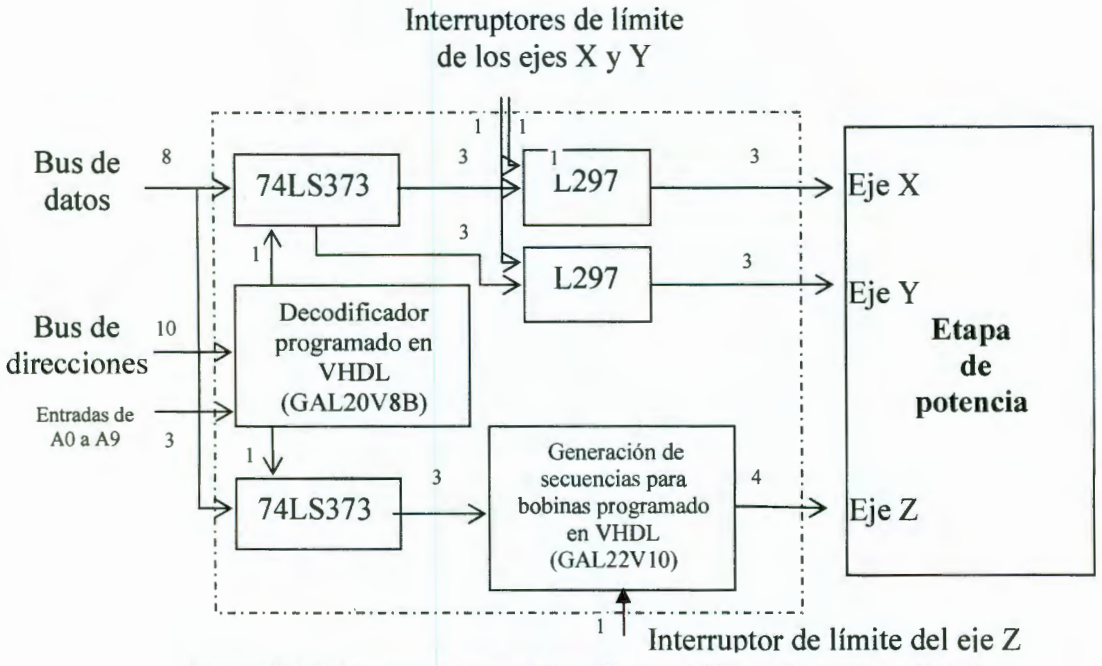


Figura 4.8. Diagrama esquemático de la configuración de la tarjeta de control de movimiento (Dentro de la línea punteada).

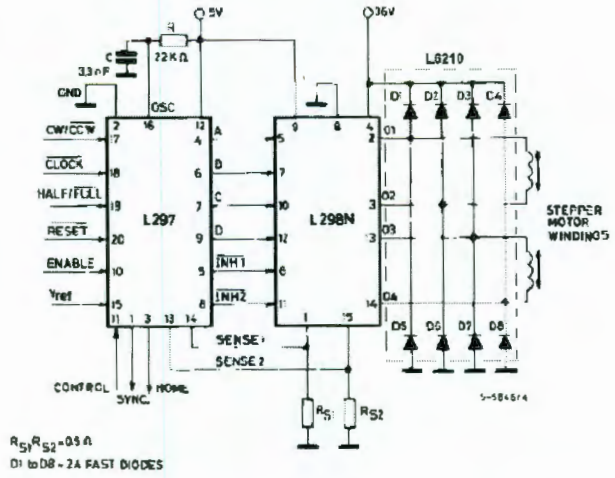


Figura 4.9. Conexión entre el L297 y el L298.

Pero en configuración de la Figura 4.9 existe un problema: De la Figura 4.8 se puede ver que se tienen dos conexiones de sensado SENCE1 y SENCE2, las cuales manejan un voltaje analógico el cual controlará la corriente de las bobinas para el chopper de la corriente. Si de alguna manera entra ruido en ellas puede ser que no se alcance la corriente de devanado máxima del circuito o que esta sea excesiva, es por ello que el circuito L297 se debe montar lo más cerca del L298 o que se use una forma de reducir el ruido. En este caso se separará el L297 de la tarjeta de control de movimiento, moviéndola a la tarjeta de la etapa de potencia, quedando la tarjeta de control de movimiento como se muestra en la Figura 4.10.

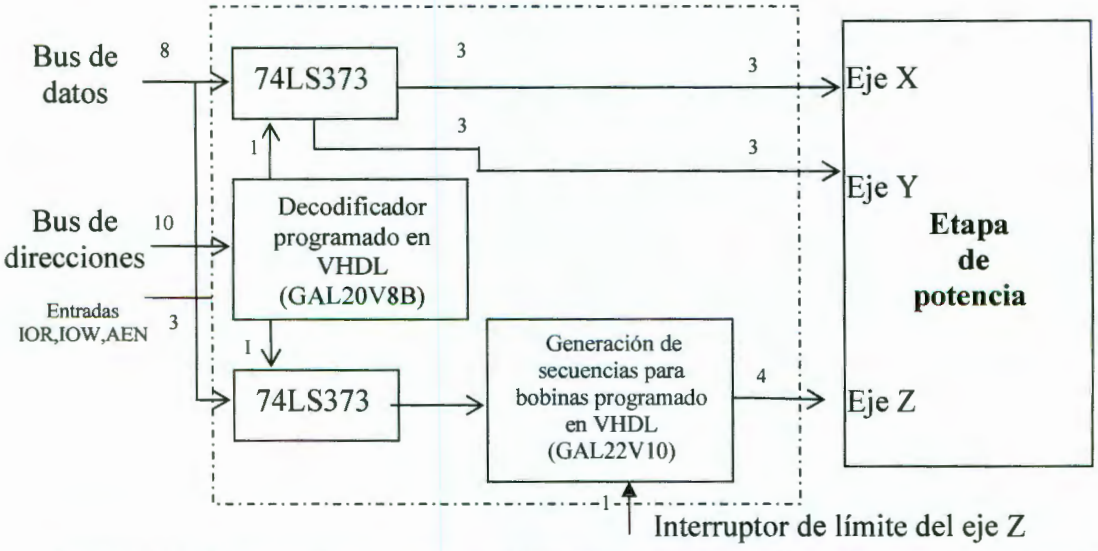


Figura 4.10. Diagrama esquemático mejorado de la configuración de la tarjeta de control de movimiento.

En el siguiente Capítulo se muestra el complemento de esta etapa, la cual es la etapa de potencia.

Capítulo 5

Etapa de potencia.

Debido a que las salidas de los generadores de secuencias usados en el Capítulo 4[1] no tienen la potencia necesaria para energizar las bobinas, en este Capítulo se analizan dos formas de energizar las bobinas por encapsulados comerciales: Mediante un buffer que contiene transistores el ULN2803[2] y el complemento del L297 el L298, para así seleccionar el tipo de etapa de potencia a utilizar.

5.1. Generación de potencia por el ULN2803.

Cuando se quiere que una salida digital controle corrientes o voltajes elevados en una bobina se pueden utilizar interruptores de transistor. El diodo que tiene el transistor sirve para que al mandar la señal de cero el circuito se desenergize en un tiempo dado por la relación L/R. Un transistor de potencia es capaz de manejar grandes cargas si es polarizado por una pequeña señal digital.

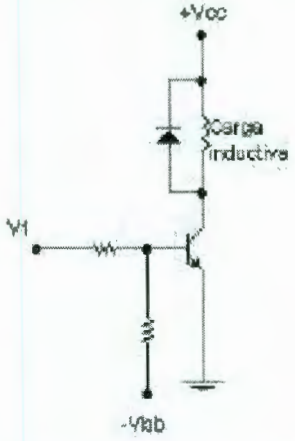


Figura 5.1. Etapa de potencia para una bobina.

En la Figura 1.1 aparece un interruptor para una carga inductiva. La resistencia de polarización de la base es usada para encender el transistor cuando el voltaje de señal se hace bajo. El diodo en paralelo con la carga inductiva permite que esta se desconecte sin picos abruptos cuando el transistor se apaga. Esto protege al transistor de una excesiva disipación de potencia durante la conmutación y de daños debidos a los picos de voltaje que pueden ocurrir cuando se apaga un inductor.

Para poder energizar un motor a pasos se necesita un arreglo de varios de estos transistores con una ganancia alta de corriente, y dentro de este tipo de circuitos

[1] La GAL22V10 programada y el L297.
[2] Hoja de especificaciones se encuentra en el anexo 5.1.

se encuentra el ULN2803 en el cual los transistores están en un arreglo Darlington.

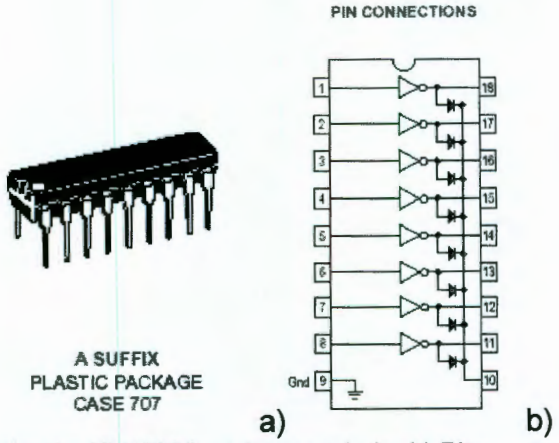


Figura 5.2. El circuito ULN2803. a) Encapsulado, b) Pines para conexión.

La forma de conectarlo es como se muestra en la Figura 5.3.

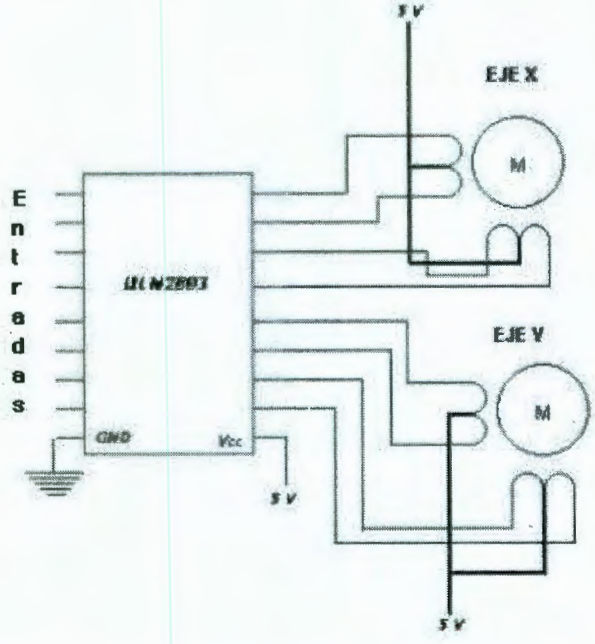


Figura 5.3. Conexión del ULN2803 y los motores a pasos.

5.1.1. Modificaciones para mejorar el desempeño de los buffers de transistores.

Debido a que el campo magnético que genera la bobina está dado por la intensidad de corriente que pasa por ella (Definido en el Capítulo 2), podemos ver que entre más rápido podamos cargar y descargar las bobinas, más rápido podremos dar pasos en el motor. Para poder hacerlo podríamos se debe agregar una resistencia en serie con el diodo para que nuestro circuito quede de la como se ve en la Figura 5.4.

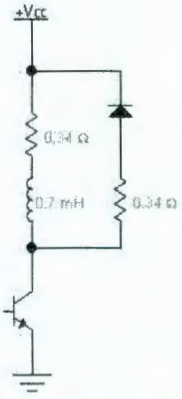


Figura 5.4. Etapa de potencia mejorada.

Podemos ver que al agregar esta resistencia sólo podremos bajar la constante de tiempo a la descarga de nuestro circuito, siendo la ecuación de la corriente de la descarga:

$$I(s) = \frac{\frac{1}{R1+R2}}{\frac{L}{R1+R2}s+1} Eo(s)$$

Aunque también podemos aumentar el voltaje y limitar la corriente mediante un circuito de "chopper" o limitador de corriente, esto con el fin de que aunque se tenga la misma constante de tiempo se pueda llegar a la corriente de excitación de la bobina de forma más rápida como se ve en la Figura 5.5.

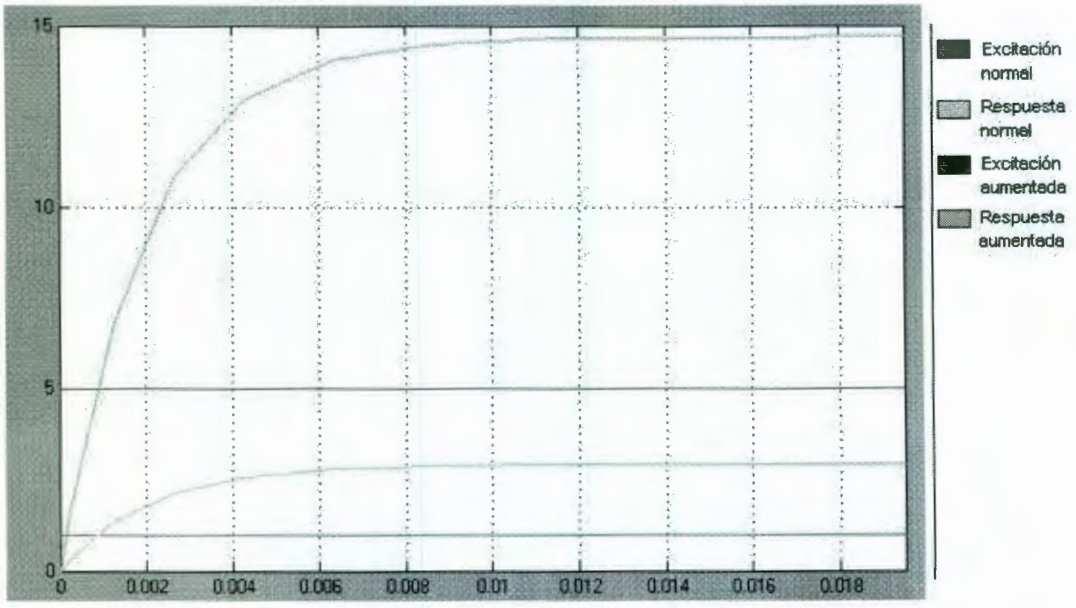


Figura 5.5. Respuesta mejorada del comportamiento de la corriente.

Como podemos ver, si la corriente de excitación de las bobinas para un campo magnético aceptable es de 3 amperes, el tiempo en que la bobina puede energizarse se ve disminuido si la excitación (Voltaje) se aumenta; sólo quedando el factor de recorte o "Chopper" de la corriente cuando llegue a la capacidad nominal de la misma. En la figura 5.5 aparece que con una excitación de 1 volt el tiempo en que se alcanza la corriente nominal de 3 amperes es de 6 milisegundos, mientras que, con la excitación de 5 volts el tiempo en el que alcanza los 3 amperes es de sólo 0.5 milisegundos.

Las modificaciones que podríamos hacer al circuito para que se comporte de esta manera serían de acuerdo a la Figura 5.6.

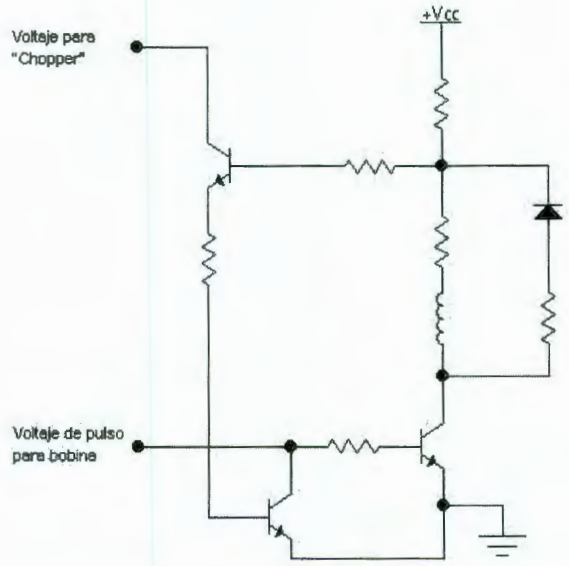


Figura 5.6. Circuito de transistores mejorado.

5.1.2. Ventajas del uso del ULN2803.

Para poder implementar una etapa de potencia con este circuito es muy simple puesto que sólo se requiere conectar los motores, la alimentación y las señales de activación además el costo del circuito es reducido y es fácil de encontrar en cualquier tienda de electrónica.

5.1.3. Desventajas del uso del ULN2803.

Como podemos ver nuestro circuito se hace cada vez más complejo y no hay un forma de modificar al ULN2803 para que pueda agregársele, por lo tanto éste se debe hacer a partir de transistores y otros elementos básicos.

Otra de las desventajas, quizá la mayor es la poca cantidad de corriente entregada por bobina la cual es de 500 mA por bobina.

5.2. Generación de potencia por el circuito L298.

Debido a que el L298 es un circuito complementario del L297 será difícil referirse a el sin mencionar el L297 es por ello que la mayoría de la descripción que hago se basa en parte de la combinación de los dos circuitos, además de ello se tendrán que tomar algunos conceptos básicos necesarios para entender el funcionamiento del L298.

5.2.1. El puente H.

Un puente H es un arreglo de transistores el cual sirve para energizar y desenergizar rápidamente una bobina. Existen en el mercado medios puentes H y completos, lo anterior es debido a que cuando se utilice un medio puente H (Figura 5.7) se requieren dos fuentes de alimentación, y cuando se utilice un puente H completo (Figura 5.8) solamente se requiere una fuente de alimentación. En este trabajo no se discutirá el funcionamiento del medio puente H esto es debido a que se explica el funcionamiento al puente H por usarse en un dispositivo.

La forma de operación del puente H es sencilla: Los cuatro transistores actúan como interruptores, conectando los extremos de la bobina ya sea a alimentación o a tierra. Cuando los interruptores Q1 y Q4 están encendidos, la corriente fluye de la alimentación a través del transistor Q1, la bobina, el transistor Q4 y por último llega a tierra. Cuando los interruptores Q1 y Q4 están apagados, la corriente del devanado no puede instantáneamente caer a cero debido a la inductancia de la bobina pues la corriente almacenada fluye a través de los diodos D2 y D3, creando un voltaje negativo en la bobina hasta que la corriente llega a cero que es cuando los diodos bloquean el flujo de corriente inverso y a pesar de las fugas la bobina se considera que está en circuito abierto. Similarmente cuando los interruptores Q2 y Q3 están encendidos, la corriente fluye de la alimentación positiva, a través del interruptor Q3, después a la bobina pero esta vez en sentido contrario, a través del interruptor Q2 y por último a la tierra, siendo que ahora el voltaje de la bobina es negativo. Cuando los interruptores Q2 y Q3 están apagados, la corriente de desenergización fluye través de los diodos D1 y D4, aplicando un voltaje positivo en la bobina hasta que la corriente se hace cero que es cuando los diodos bloquean la corriente e igualmente que el caso anterior se puede considerar la bobina como en circuito abierto.

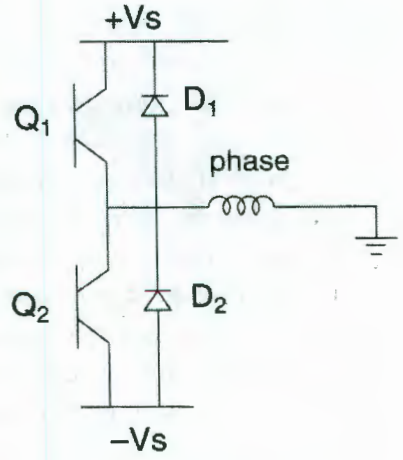


Figura 5.7. Medio puente H.

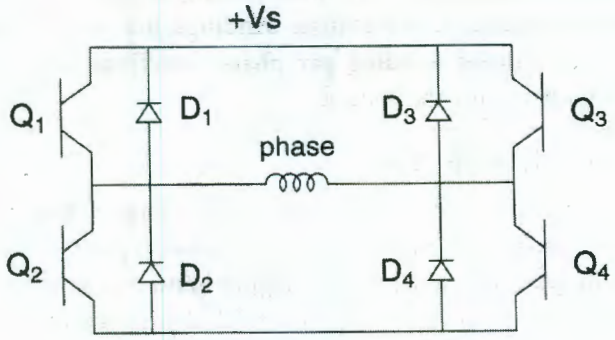


Figura 5.8. Puente H completo.

5.2.2. Limitación de corriente máxima (Chopper).

La limitación de corriente (Llamada comúnmente chopeado) consiste en aumentar la tensión de alimentación del motor para que éste alcance la corriente de regulación (Ireg) lo más rápidamente posible. Una vez alcanzada, se cortará y se conectará de forma que se mantenga ese valor de la corriente. Si no se empleara este método los tiempos de reacción serían enormes. En la Figura 5.9 se puede ver la regulación dada por uno de éstos circuitos

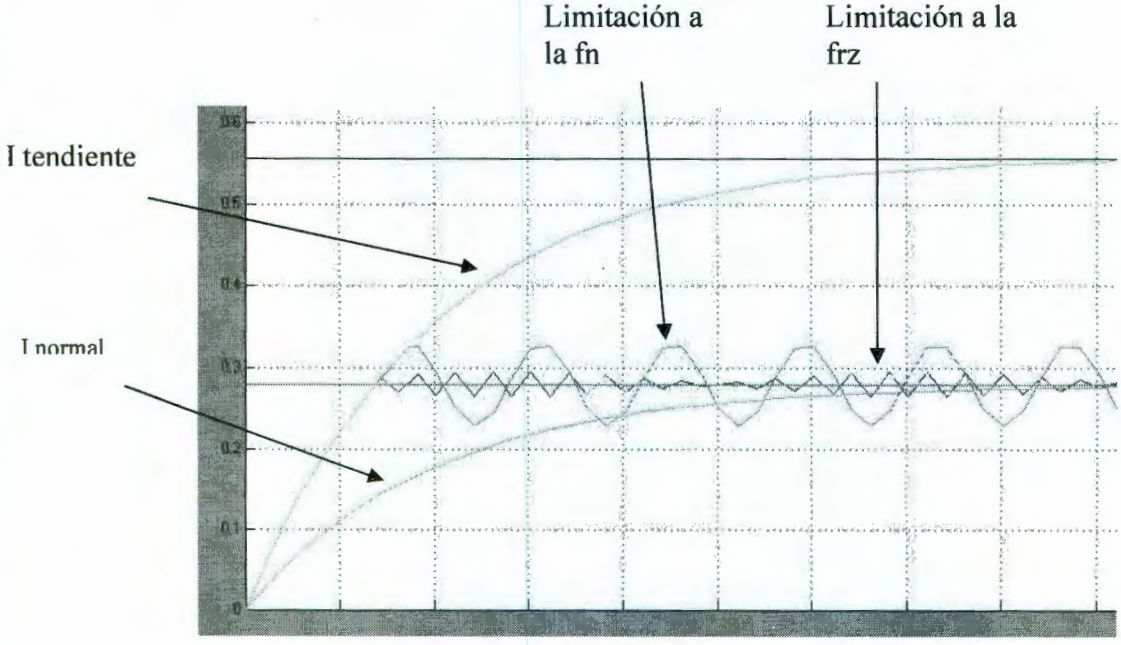


Figura 5.9. Limitación de corriente.

Siendo que para cada bobinado el esquemático es el siguiente:

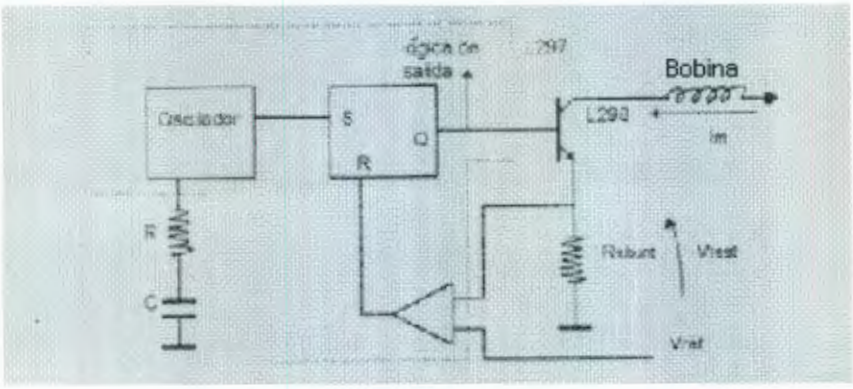


Figura 5.10. Circuito por bobina para chopper.

Mediante Vref y calculando adecuadamente Rshunt controlaremos la corriente Im que circulará por el motor, así haciendo:

$$Vtest = Vref$$

$$Im = Imax$$

que es la que circulará por el motor.

$$Rtest = Vtest / Im$$

$$Ptotal\ del\ motor = Pmax = R.Im$$

donde R es la resistencia que presenta cada devanado

En función de las fases activas:

$$1\ fase \rightarrow Im=(Pmax/R)^{1/2}$$

$$2\ fases \rightarrow Im=((Pmax/2)/R)^{1/2}$$

Siendo así la justificación teórica del cálculo de lo que en los esquemas se denomina Rsense o para nosotros la Rshunt donde verificamos el valor de la corriente que circula por los devanados del motor para su regulación posterior.

5.2.3. Cálculo de la frecuencia de chopper.

Como preliminares partiremos de los siguientes datos de partida:

Frecuencia inicial de prueba de la señal de reloj del L297: 100Hz.
Las señales de control CW/CCW y HALF/FULL se seleccionan manualmente.
Diodos rápidos para el puente en H: tr<200ns.
Vref inicial para control del chopeado: 1v.
Resistor de la red RC que precisa el oscilador de chopeo: RV de 0.34 ohms.

El diseño del sistema vendrá dado por:

Del motor:

Pmax = 4.65W
R= 0.34 ohms.
Imax 1 bobinado activo = (4.65 w / 0.34 ohms)^1/2 = 3.7 A
Imax 2 bobinados activos =((4.65 w / 2)/0.34 ohms)^1/2=5.2 A

Del L297:

Vref: 0-3v para nuestra práctica tomaremos p.e. 1v.
Rshunt = Vref / Imax motor = 3,9W .

El efecto de variar la frecuencia del oscilador de chopper es el de regular fehacientemente la corriente que entregaremos a las bobinas del motor, con la velocidad de circulación hacia ellas superior a la que por si mismas, por t, alcanzarían. Veamos el circuito equivalente que interviene:

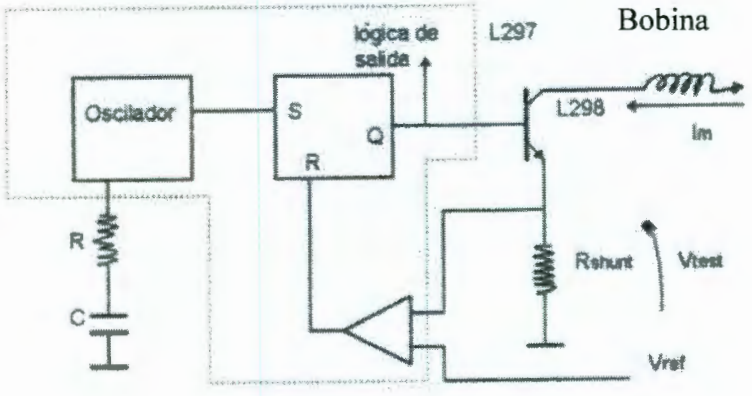


Figura 5.11. Diagrama del oscilador de chopper.

Si Vtest >= Vref resetearemos el flip-flop, cortando el transistor. A la par, la señal cuadrada del generador pondrá en set el FF saturando el transistor, así se tendrá:

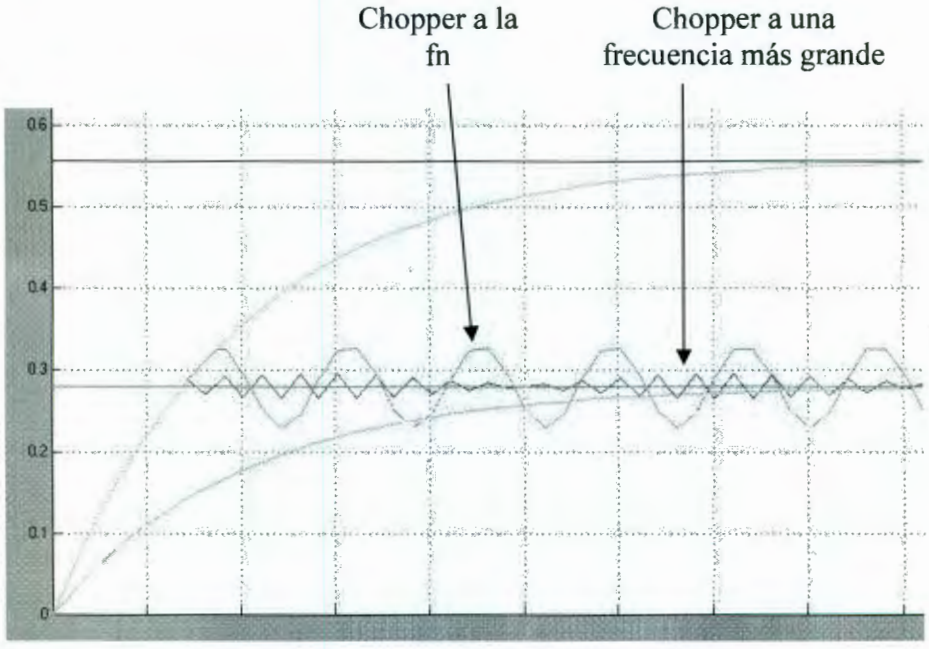


Figura 5.12. Efecto de la frecuencia en el chopper.

Aumentar la frecuencia de chopeo supone mejorar la regulación de la corriente por los bobinados del motor, en principio; mas en la práctica se observa que a partir de una frecuencia deja de regular adecuadamente. Fijando Vref=1v, C=3.3nF obtenemos la siguiente tabla:

Ro	Fr de Chopper	Rango de tensiones	Observaciones
0	1 Khz..	0,2 a 0,5	No hay regulación.
11K75	9 Khz.	0,3 a 0,5	Hay regulación.
23K5	18 Khz.	0,4 a 0,5	Hay regulación.
35K25	35 Khz.	0,3 a 0,6	Hay regulación.
47K	Continua.	1,1	No hay regulación.

Tabla5.1. Efecto de la frecuencia en la regulación.

Este fenómeno se produce al aumentar la frecuencia de chopper en exceso, haciendo que no le de tiempo al transistor de potencia del L298 a conmutar. De hecho, en sus hojas características nos dan como frecuencia máxima de conmutación de 25 a 40 Khz.

La variación de Vref conlleva al el control del nivel de tensión en Rshunt y con ello la corriente que circulará por los devanados del motor (o sea, valor de Ireg fijado anteriormente). Con esto y fijando una frecuencia de chopper adecuada, con Vref=1v de 18Khz (condiciones iniciales de diseño) se comenzó la experiencia de variar Vref desde 0 hasta 2v:

Vref	Fr. Chopeo	Actividad
0	18 Khz.	Nula.
0,25	18 Khz.	Nula.
0,5	30 Khz.	Gira torpemente.
0,75	30 Khz.	Giro con vibración.
1	18 Khz.	Giro.
1,5	5 Khz.	Giro.
2	1 Khz.	Giro.

Tabla 5.2.

Lo que aquí se refleja es Vref < Vtest -> R=L -> Q=L -> trt cortado ya que sobrepasa el motor la consigna: para 0v y 0.25v no hay movimiento, pues en el Rshunt se alcanzan rápidamente estos valores y no le da tiempo a la bobina a activarse.

A partir de 0.5v y 0.75v ya hay movimiento con más o menos dificultad (el motor empieza a vibrar) y vemos que el chopper entra en acción, ya que le da tiempo a la corriente a circular por los devanados y con ello, llegar y sobrepasar la consigna dando movimientos torpes al rotor.

Para 1v el funcionamiento es el normal y para tensiones mayores (1.5 y 2v) la situación se invierte, o sea:

Vref>Vtest -> R=H -> Q=H

Lo que viene a decirle al motor que puede consumir más corriente (no como para 0, 0.25, 0.5, 0.75, donde obligábamos al motor a consumir corrientes muy pequeñas y para los 2 primeros casos insuficiente para producir movimiento), apreciándose cómo interviene cada vez menos el chopper, pues no hace falta regular excesivamente la corriente por los devanados.

De toda esta experiencia, también nos damos cuenta de que los terminales INH1* y INH2* , al ser activadas (a nivel bajo) , inhiben el chopeado del bobinado por los pines A,B,C y D , ya que éste se realizará a través de los mencionados INH1* e INH2*.en el puente en H (L298) , haciendo que se produzca una recirculación rápida de las corrientes residuales en los devanados por medio de los diodos hacia la fuente de alimentación . Si no conectamos INH1* e INH2 del L297 al L298, el chopeado se realizará por medio de A,B,C y D, realizándose la recirculación de corrientes residuales de los bobinados de forma lenta; o sea, por disipación en el propio puente.

5.3. Selección de la etapa de potencia.

Debido a que dos de los motores a pasos que estamos usando pueden consumir hasta 3.7 ampers no es factible usar por potencia el ULN2803, además el ULN disipa la energía en si mismo lo que produce que se descargue la bobina con una velocidad demasiado baja para una aplicación de alta velocidad. Sin embargo el eje Z no requiere ni una potencia ni velocidad alta y es una buena opción implementar esta este eje con el UN2803.

Para los ejes X y Y se utilizará la etapa de potencia con el ULN298 por su rapidez de descarga de corriente y su potencia.

5.4. Implementación de la etapa de potencia.

Como premisa recordaremos que en Capítulo 4 se definió que el L297 debía estar cerca del L298 para evitar ruido, y es por lo mismo que se incluye en la tarjeta de potencia. En la Figura 5.13 se incluyen el diseño de la etapa de potencia.

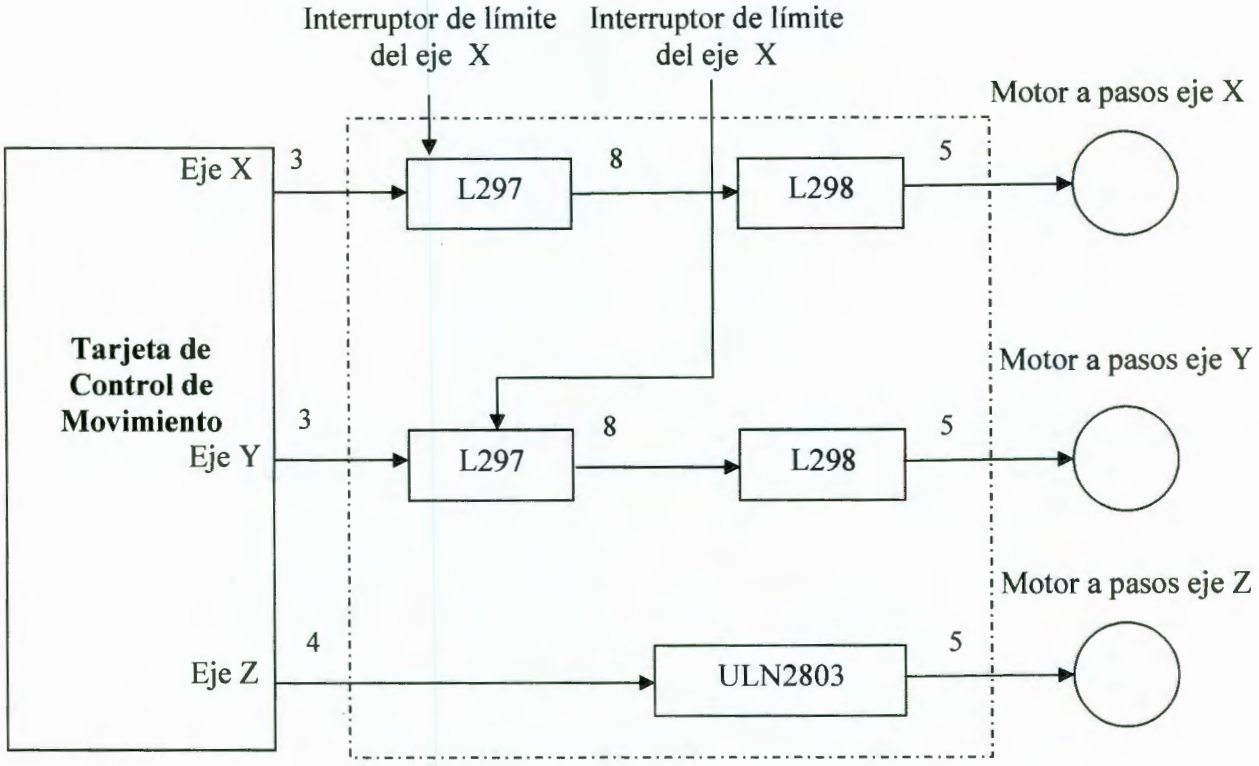


Figura 5.14. Diagrama esquemático de la etapa de potencia.

Hasta este Capítulo se definió todo el hardware necesario para poder lograr movimientos en una mesa lo, restante es generar una programación para poder generar trayectorias, lo que es tema del siguiente Capítulo.

Capítulo 6

Software de Generación de Trayectorias.

Ya sea para generar una línea o una curva se requiere generar movimientos coordinados de los diferentes ejes. Esto requiere la generación de señales de acuerdo al movimiento requerido, la generación de estas señales de referencia se logra por interpoladores. En las máquinas comerciales que hacen control numérico, el hardware consiste en circuitos digitales los cuales hacen la interpolación.

En este Capítulo se muestra la interpolación a través de la PC programada en lenguaje C y ensamblador para poder generar las señales de cantidad de pasos, velocidad y sentido, con las que la Tarjeta de Control de Movimiento trabajará.

6.1. Tipos de Interpoladores.

La forma de hacer estas interpolaciones es a través del integrador DDA(Analizador Digital Diferencial) ,el interpolador Referencia-Palabra, y el Método Tustin, sin embargo sólo se utilizará el integrador DDA debido a que es el más fácil de adaptar a la generación de señales para mover motores a pasos.

6.2. El Integrador DDA.

Los analizadores diferenciales digitales o DDA´s hacen un cálculo especial en el cual las variables se representan como palabras digitales, pero el cómputo es similar al hecho en computadoras analógicas, éste método combina la precisión de la computación digital con la continuidad del analógico. Su elemento principal es el integrador DDA, el cual tiene un rol similar al de un amplificador operacional y forma el block principal de la integración. La dificultad de transferir palabras de n bits se evita usando un método de cómputo incremental de transferencia. Ello significa que sólo los incrementos de las variables y su signo son transferidos, así que sólo dos líneas en vez de (n+1) se requieren.

6.2.1. Principio de funcionamiento del integrador DDA.

La forma de de funcionar es básicamente de sumas sucesivas usando métodos rectangulares o trapezoidales de aproximación. El método que se usa en los controles numéricos convencionales es el rectangular y es el mismo que se usará en este trabajo.

Asumamos que "p" es una variable del tiempo "t", como se ilustra en la Figura 6.1, la integración está dada por la aproximación del área bajo la curva de la suma de todos los rectángulos cada uno de ellos con una longitud de la base de Δt,

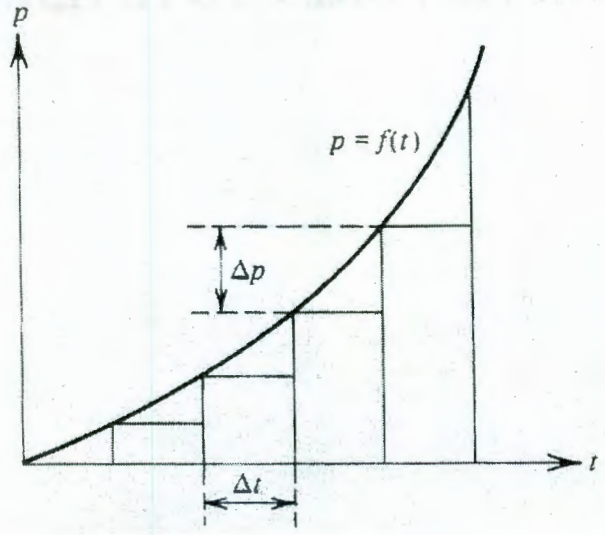


Figura 6.1. Integración digital.

Lo anterior nos lleva a la ecuación:

$$z(t) = \int_0^t p dt \cong \sum_{i=1}^{k} p_i \Delta t \tag{6.1}$$

El valor de z cuando t=k Δt se denota como z_k, el cual puede ser escrito como:

$$z_k = \sum_{i=1}^{k-1} p_i \Delta t + p_k \Delta t \tag{6.2}$$

o como

$$z_k = z_{k-1} + \Delta z_k \tag{6.3}$$

de donde

$$\Delta z_k = p_k \Delta t \tag{6.4}$$

La integración digital se hace en tres etapas. Primero el valor de la ordenada P_k se calcula añadiendo o quitando el incremento Δp_k de la ordenada anterior.

$$p_k = p_{k-1} \pm \Delta p_k \tag{6.5}$$

Después el incremento de salida Δz se calcula con ayuda de acuerdo a la con ecuación 6.4, y finalmente se le añade a la z previa de acuerdo a la ecuación 6.3.

El integrador DDA funciona de un modo iterativo a una frecuencia "f" dada por un reloj externo donde:

$$f = \frac{1}{\Delta t} \tag{6.6}$$

Después en cada iteración se calculan las operaciones 6.4 y 6.5.

Como ya se explicó, las entradas y salidas entre los integradores DDAs se transfieren como incrementos de 1 bit, y los valores de Δp y Δz deben ser unos o ceros. La forma de lograrlo es guardando la variable p en un registro de n bits o en un contador ascendente o descendente limitando así el valor máximo permisible de 2^n-1 así:

$$\frac{p_k}{2^n} < 1$$

El incremento de Δp ya sea de valor 1 o 0 únicamente se añade al bit menos significativo (LSB) del registro o contador de n bits, el cual se denota como el registro p. La salida incremental se calcula con la ayuda adicional de un registro adicional el cual llamaremos registro q. En cada iteración el contenido de la variable p se añade al contenido previo de q

$$q_k = q_{k-1} + p_k \tag{6.7}$$

Si el nuevo valor de q es mayor que (2^n-1) el cual es el valor máximo posible con una palabra de n bits, se genera un incremento Δz. Por ejemplo si tenemos una palabra de 3 bits, un valor de p_k =6 y el valor de q_{k-1} = 4, la operación suma binaria se ejecuta como sigue:

$$\begin{array}{rr} & q_{k-1} \\ + & p_k \\ \hline \Delta z & q_k \end{array} \qquad\qquad \begin{array}{rr} & 100 \\ + & 110 \\ \hline 1 & 010 \end{array}$$

De esta forma q_k = 2 y Δz = 1.

A continuación se muestra un diagrama esquemático de un integrador DDA (Figura 6.2):

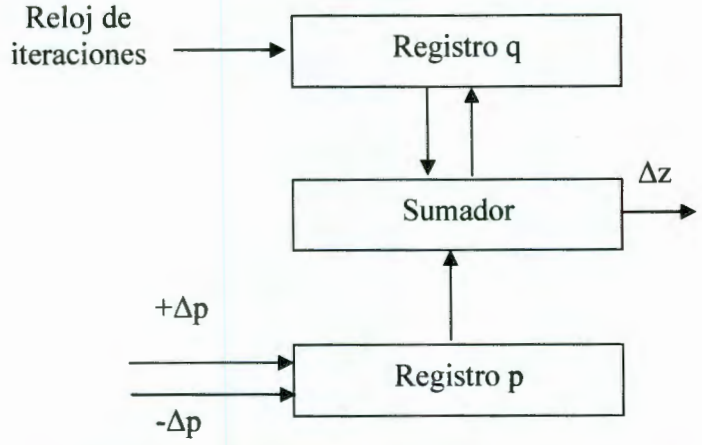


Figura 6.2. Diagrama esquemático del integrador DDA.

Como se puede ver el integrador DDA consiste de dos registros de n bits y de un sumador binario. Durante cada interación el valor de p se obtiene se obtiene a través de la ecuación 6.5 de la cual Δp es uno o cero. La integración se logra con la ecuación 6.7 y se ejecuta con la ayuda del sumador binario el cual añade el contenido de los registros p y q en cada iteración. El sobreflujo que se obtiene es el incremento de z (Δz). Matemáticamente estos incrementos se dan por :

$$\Delta z_k = 2^{-n} p_k \tag{6.8}$$

y combinando las ecuaciones 6.6 y 6.8 se pueden escribir como:

$$\Delta z_k = C p_k \Delta t \tag{6.9}$$

donde C es la constante de integración del DDA, el cual está definido por

$$C = \frac{f}{2^n} \tag{6.10}$$

y como se puede ver la ecuación 6.4 tiene la misma estructura que la 6.9 pero con la constante C haciéndola más completa.

La figura 6.3 muestra una representación del integrador DDA. La salida del DDA es el pulso de sobreflujo Δz, el cual puede ser conectado a la entrada del incremento Δp.

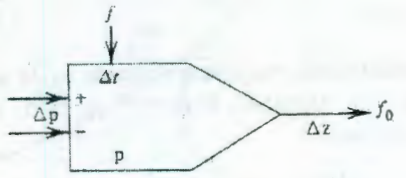


Figura 6.3. Representación simbólica del integrador DDA.

El DDA por si solo no acumula la cantidad de pulsos de Δz, si se requiere el resultado de la integración se requiere de un contador adicional el cual ejecute la ecuación 6.3.

El promedio de la frecuencia de la salida f_0 se obtiene de la ecuación 5.9:

$$f_0 = \left(\frac{\Delta z}{\Delta t}\right)_k = Cp_k = \frac{fp_k}{2^n} \tag{6.11}$$

El cual es directamente proporcional a la frecuencia de iteración y al valor de p e inversamente proporcional al valor de 2^n donde n es la longitud de los registros (en bits) de los registros del DDA. Es muy importante establecer la cantidad de bits a utilizar en los registros del DDA , ya que el número de bits define la resolución del proceso de integración pues entre más bits es más precisa la integración. Un ejemplo de cómo se integra digitalmente se muestra en la figura 6.4.

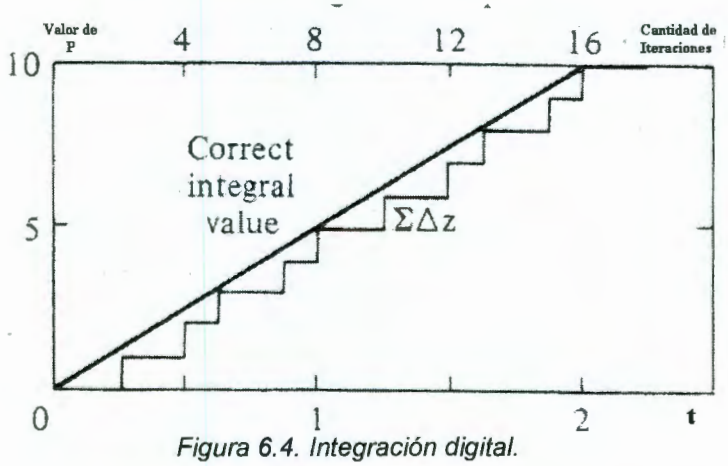


Figura 6.4. Integración digital.

De aquí es fácil comprender porque es tan fácil trasladar este sistema a motores a pasos; debido a que cada pulso Δz lo podemos interpretar como un paso que da el motor, la suma de pulsos es la distancia total recorrida por el eje y la velocidad esta dada por la frecuencia de los pulsos.

6.3. Aceleración y desaceleración exponencial.

Cuando se usan máquinas de control numérico generalmente se usan aceleraciones graduales del movimiento de los ejes para poder vencer la inercia del cuerpo en estado estático y cuando va a llegar a la posición deseada se desacelera gradualmente para poder vencer la inercia del cuerpo en movimiento. En estas máquinas además del integrador DDA para generar los Δz los cuales generan el movimiento, también se utiliza un DDA el cual produce una decadencia exponencial en los Δz para desacelerar el eje de las máquinas antes de que pare completamente.

La forma de implementarlo es la siguiente:

Supongamos la función exponencial:

$$p(t) = p_0 e^{-at} \tag{6.13}$$

Con la definición de la frecuencia dada por la ecuación 6.11 obtenemos:

$$f_0 = \left(\frac{\Delta z}{\Delta t}\right)_k = Cp_k = Cp_0 e^{-\alpha t} \tag{6.14}$$

derivando la ecuación anterior se tiene que:

$$dp = (-p_0 \alpha) e^{-\alpha t} dt \tag{6.15}$$

siendo que la ecuación de diferencias correspondiente es:

$$-\Delta p = (\alpha) p_k \Delta t \tag{6.16}$$

Comparando las ecuaciones 5.9 y 5.16 se puede ver que si ajustamos la constante C de tal manera que sea igual a α tenemos:

$$-\Delta p = \Delta z \tag{6.17}$$

Para completar la función exponencial de forma gráfica sólo tenemos que conectar la salida del integrador DDA al decremento de p (-Δp) como se muestra en la Figura 6.5.

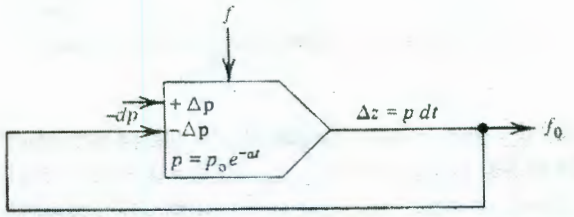


Figura 6.5. Diagrama del integrador DDA con desaceleración integrada.

Para lograr una curva de aceleración solamente se tendría que hacer que el registro p empezara en 0 conectar Δz al incremento de p (Δp) en lugar de al decremento. La forma en que se comportan los registros p y el resultado de la integración se muestran en la siguiente Figura:

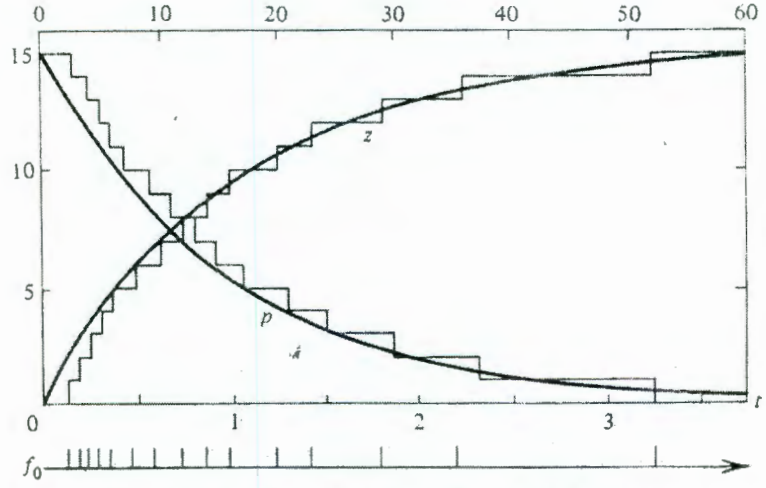


Figura 6.6. Contenido del registro p y el resultado de la integración de $15e^{-t}$.

6.4. Interpolación lineal.

Se le llama interpolación lineal a la habilidad de controlar el movimiento en una línea dada entre dos coordenadas. La interpolación se puede dar en un plano usando uno o dos ejes de movimiento o en tres dimensiones. El integrador da comandos de velocidad, en pulsos por segundo simultáneamente a los motores de los diferentes ejes y mantiene la relación entre ellos. Por ejemplo si quisiéramos una línea recta que tuviera una relación de movimiento de cinco pasos en el eje X por tres pasos en el eje Y tratando de obtener una respuesta igual a la que se muestra en las Figuras 6.7a y 6.7b.

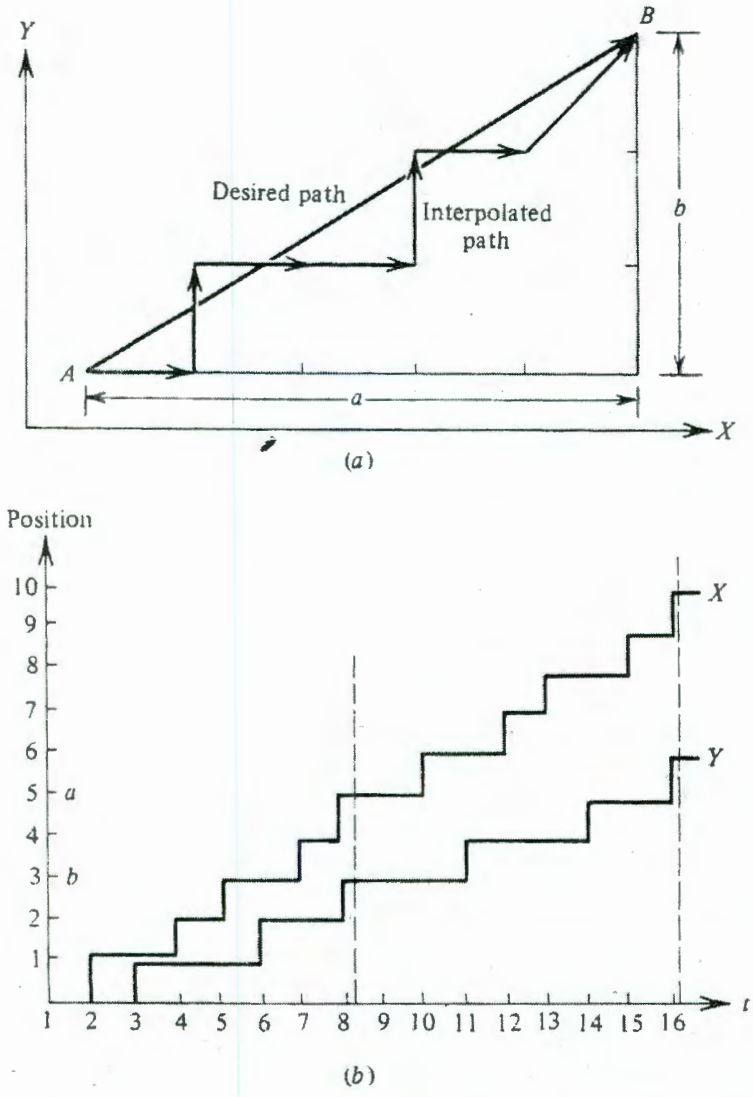


Figura 6.7. a) Trayectoria generada por los ejes X y Y, b) Gráficas del eje X y del Y de pasos dados contra el tiempo.

La forma de calcular una interpolación lineal en dos dimensiones es mediante el uso de dos integradores DDA (Figura 6.8).

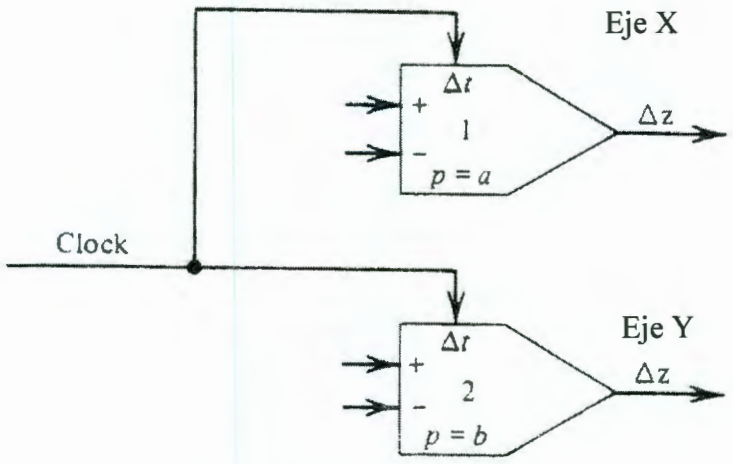


Figura 6.8. Interpolación lineal mediante el uso de dos integradores DDA.

Como se puede ver se requieren un integrador DDA por cada eje; el primer integrador da los pulsos para los pasos del eje X y el segundo integrador hace lo mismo para el eje Y. Los dos ejes están controlados por el mismo reloj así que las operaciones de integración se hacen al mismo tiempo. La diferencia radica en que cada registro p puede cargarse con un valor diferente en su respectivo DDA. La siguiente tabla muestra cómo sería la frecuencia de pulsos para el ejemplo de la relación 5:3 descrita anteriormente además de mostrar los resultados de las operaciones que muestran el desbordamiento.

Paso dado	PX	QX	ΔzX	PY	QY	ΔzY
Inicial	5	0		3	0	
1	5	5		3	3	
2	5	2	1	3	6	
3	5	7		3	1	1
4	5	4	1	3	4	
5	5	1	1	3	7	
6	5	6		3	2	1
7	5	3	1	3	5	
8	5	0	1	3	0	1
9	5	5		3	3	
10	5	0	1	3	6	

Tabla. 6.1. Frecuencia de pulsos de la interpolación lineal.

Con el interpolador lineal ya descrito sólo queda manejar la velocidad con la que se va a trazar la línea, esto se logra añadiendo un integrador DDA adicional el cual controla la frecuencia de alimentación del DDA (FRN) (Figura 6.12) controla la frecuencia de integración de los otros dos DDAs.

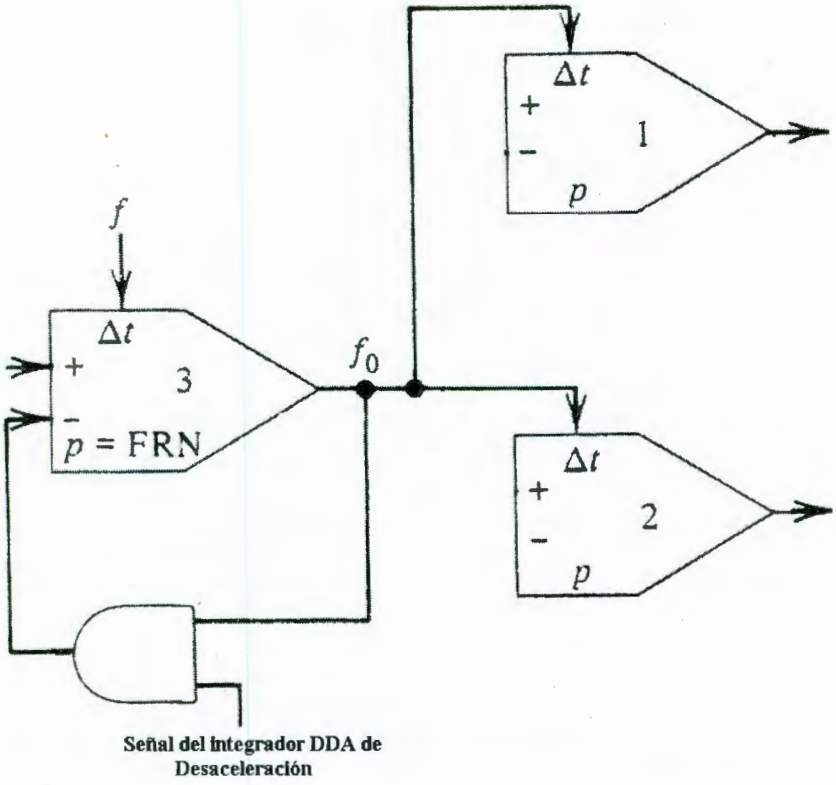


Figura 6.9. Interpolador lineal con controlador de frecuencia de los pulsos.

6.5. Interpolación circular

El interpolador circular elimina la necesidad de definir muchos puntos en un área circular. Sólo se necesita el radio para generar el arco. En la mayoría de los casos la interpolación circular se limita a sólo un cuadrante, y es por ello que el punto de inicio y el final deben estar en el mismo cuadrante también. Arcos más grandes se dividen en arcos sucesivos. La función que nos describe un arco es la siguiente:

$$(X - R)^2 + Y^2 = R^2 \tag{6.18}$$

donde R es el radio del círculo requerido y

$$X = R\,(1-\cos \omega t) \tag{6.20}$$

$$Y = R \operatorname{sen} \omega t \tag{6.21}$$

Siendo la velocidad la derivada de la posición tenemos:

$$V_X = \frac{dX}{dt} = \varpi R(sen\,\varpi t) \tag{6.20}$$

$$V_Y = \frac{dY}{dt} = \varpi R(\cos \varpi t) \tag{6.21}$$

Escritas de otra forma:

$$dX=\omega R(\text{sen } \omega t)dt=-d(R \cos \omega t) \tag{6.22}$$

$$dY=\omega R(\cos \omega t)dt=+d(R \text{ sen } \omega t) \tag{6.23}$$

Donde ωR es la velocidad de trazado del arco.

El interpolador circular consiste de dos integradores DDA cruzados en sus aumentos o decrementos del registro p como se muestra en la Figura 6.10.

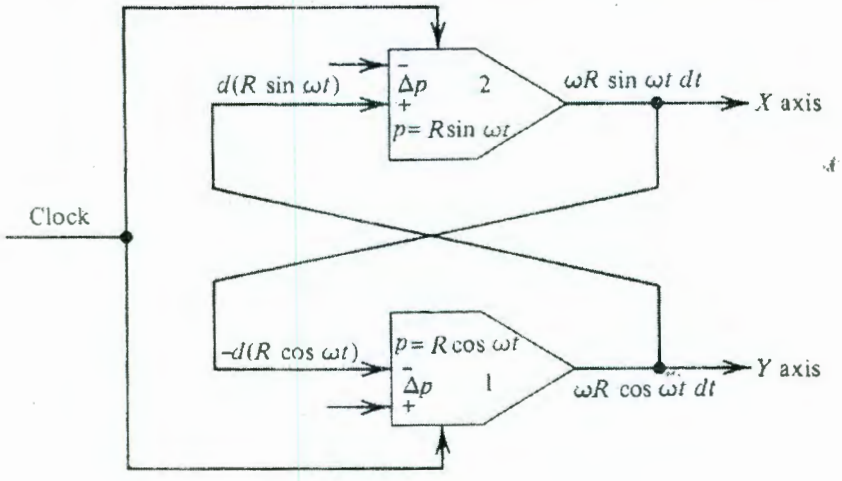


Figura 6.10. Interpolador circular.

Los registros p de los integradores se cargan con los valores i y j los cuales están definidos por

$$|i|=R \cos \omega to \tag{6.24}$$
$$|j|=R \text{ sen } \omega to \tag{6.25}$$

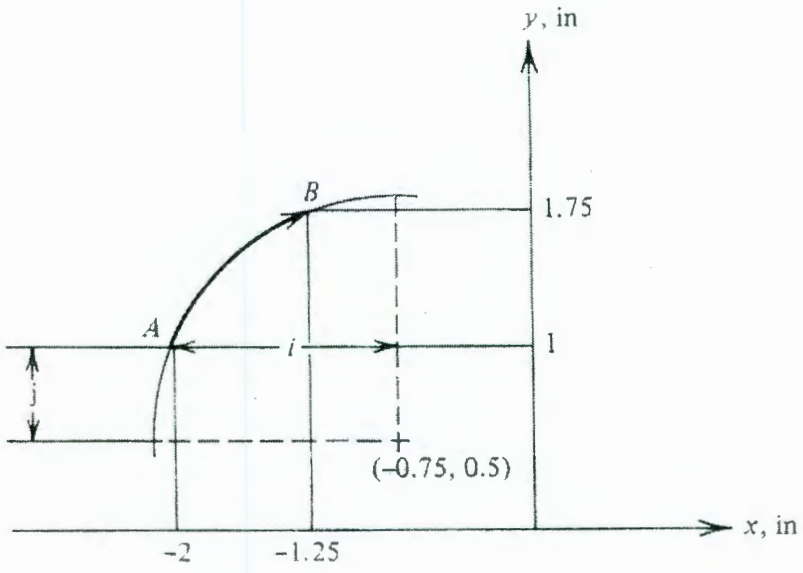
donde t_0 es el tiempo inicial. La salida del DDA esta dado por la ecuación 6.9 :

$$\Delta z1=CR \cos \omega t\, dt \tag{6.26}$$
$$\Delta z2=CR \text{ sen } \omega t\, dt \tag{6.27}$$

Y el valor que actualiza el valor del registro p esta dado por:

$$-\Delta p1=-d(R \cos \omega t) \quad (6.28)$$
$$\Delta p2=-d(R \operatorname{sen} \omega t) \quad (6.29)$$

Las ecuaciones 6.26 y 6.27 se parecen mucho a las ecuaciones 6.22 y 6.23 donde la constante de integración se ajusta de tal manera que C = ω.



A = Punto inicial
B = Punto final

Figura 6.11. Puntos inicial y final para calcular valores de i y j .

El interpolador funciona de la siguiente manera: El DDA 2 (ver Figura 6.10) se carga con el valor de p=j y sus pulsos de salida son los pasos del eje X , mientras al DDA 1 se carga con el valor de p=1 y sus pulsos dan los pasos del eje Y. El contenido de R sen ωt del registro p del DDA 2 se actualiza mediante el incremento ΔzY e cual está dado por dY=d(R sen ωt) obtenido del DDA 1, como se puede ver de la ecuación 6.23. Similarmente la salida del DDA 2 el cual es – d(R cos ωt) se conecta a la entrada –Δp del DDA 1 con el fin de actualizar su registro p que inicialmente es igual a R cos ωt.

Para dejar este procedimiento en claro se puede dar el siguiente ejemplo: Si tenemos un registro del DDA igual a 4 bits y queremos un arco que vaya de 0 a 90° con un radio igual a 15 pasos tenemos que de acuerdo a las ecuaciones 6.24 y 6.25:

$$i=15 \cos 180° = 15$$
$$j=15 \operatorname{sen} 180° = 0$$

Con los datos obtenidos se carga el registro p del DDA 1 con el valor i=15 y el registro p del DDA 2 con j=0. El resultado de las iteraciones se muestra en la siguiente tabla:

Iteración	PX	QX	AzX	PY	QY	AzY
Inicial	15	0		0	0	
1	15	15		0	0	
2	15	14	1	1	1	
3	15	13	1	2	3	
4	15	12	1	3	6	
5	15	11	1	4	10	
6	15	10	1	5	15	
7	15	9	1	6	5	1
8	14	7	1	7	12	
9	14	5	1	8	4	1
10	13	2	1	9	13	
11	13	15		9	6	1
12	12	11	1	10	0	1
13	11	6	1	11	11	
14	11	1	1	12	7	1
15	10	11		12	3	1
16	9	4	1	13	0	1
17	8	12		13	13	
18	8	4	1	14	11	1
19	7	11		14	9	1
20	6	1	1	15	8	1
21	5	6		15	7	1
22	4	10		15	6	1
23	3	13		15	5	1
24	2	15		15	4	1
25	1	0		15	3	1

Tabla 6.2. Interpolación circular.

Se puede ver que dado que el valor del registro p del DDA 1 es inicialmente alto i=15 y por lo mismo emite pulsos a alta frecuencia. El valor de p se reduce gradualmente por los pulsos que entran al decremento -Δp, reduciendo la frecuencia hasta a cero. Lo cual significa que la salida de frecuencia del DDA 1 corresponde a una función cos ωt. En contraste al inicio se carga en el registro p del DDA 2 un valor bajo j=0 lo cual hace que su frecuencia inicial sea igual a cero, pero conforme el registro p s va llenando con pulsos del DDA1 a través de la entrada Δp la frecuencia aumenta gradualmente emulando una función seno. La interpolación termina cuando el registro p del DDA 1 se vuelve cero.

A continuación se muestra la curva obtenida:

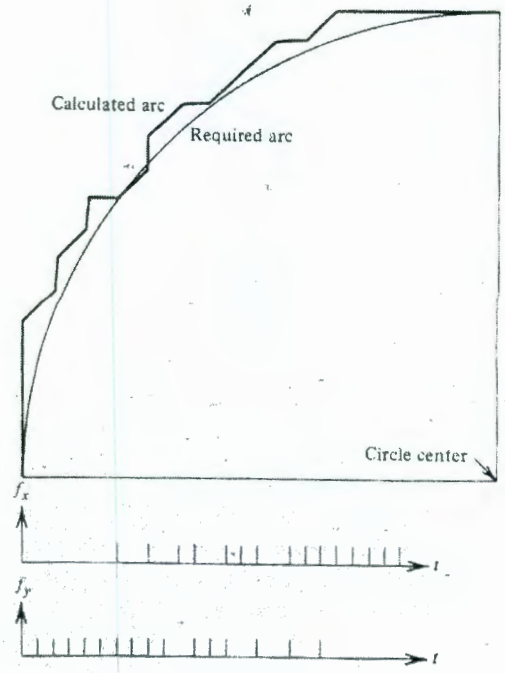


Figura 6.12. Arco deseado y arco obtenido por la interpolación circular.

Como se puede ver esta interpolación circular no es muy exacta, esto se debe a la poca cantidad de bits involucrados en la interpolación. Al aumentar el número de bits involucrados la resolución será mejor, con ello obteniendo una respuesta más parecida a lo requerido. En el control numérico convencional el número de bits de los registros varía de entre 14 a 20 bits de resolución.

Hasta el momento sólo se ha tratado un tipo de arco, el que va de 180 a 90° sin embargo hay varios tipos de arcos que se pueden dibujar (ver Figura 6.13), sin embargo esto se compensa fácilmente cambiando el sentido de la rotación de los motores.

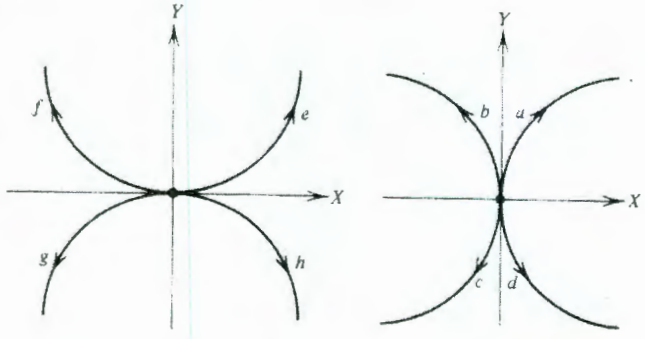


Figura 6.13. Tipos de arco que se pueden realizar.

6.6. Implementación del algoritmo.

El lenguaje en que se programa este trabajo es el lenguaje C debido a que es preferido por la industria por la facilidad de interactuar con los registros del microprocesador y entradas y salidas. Asimismo se programa en lenguaje ensamblador varias rutinas ya que es un lenguaje de menor nivel que el lenguaje C y por lo mismo sirve para optimizar rutinas más rápidas que las creadas en lenguaje C esto con el fin de mejorar el desempeño del softwàre de generación de trayectorias. Los programas completos se puedes ver en los anexos 6.1 y 6.2.

6.6.1. Algoritmo creado para la interpolación lineal.

Debido a que en la computadora se puede tener más bits de resolución que el que pueda dar un microcontrolador común, el valor de p lo definí como el valor de pasos para cada eje en cada integrador DDA y la resolución de los integradores DDA será del valor máximo más uno; esto es si se requiere una línea que requiere que el eje X de 5000 pasos mientras el eje Y de 4800; el valor de p del integrador DDA 1 (de acuerdo a la Figura 6.8) será de 5000 mientras que del DDA 2 será 4800 el valor del registro p, mientras que la resolución máxima será la de 5001 a continuación se muestra la función que hace la interpolación lineal.

```
do{
  qx=(qx+px);
  if(qx>=(pasos_mayores+1))
   { qx1=qx;Azx=1;qx=(qx1-(pasos_mayores+1));   Hace la interpolación en
   }                                            el eje X.
  qy=(qy+py);
  if(qy>=(pasos_mayores+1))
   { qy1=qy;Azy=1;qy=(qy1-(pasos_mayores+1));   Hace la interpolación en
   }                                            el eje Y.
  if(pasos_x>pasos_y)
```

6.6.2. Algoritmo creado para la interpolación circular.

El procedimiento de asignación de valores para p de los dos integradores DDA fue asignado de acuerdo al tipo de arco que se tratara, en este trabajo sólo se definieron arcos de 90° en las ocho direcciones posibles pero eso no implica que el algoritmo no pueda hacer arcos de diferentes amplitudes. El algoritmo de la interpolación circular es el siguiente:

```
do{
 qx_c=(qx_c+px_c);
 if(qx_c>=(radio+1))
 { qx1_c=qx_c;Azx=1;qx_c=(qx1_c-(radio+1));
     py_c++;
 }

 qy_c=(qy_c+py_c);
```

```
if(qy_c>=(radio+1))
  { qy1_c=qy_c;Azy=1;qy_c=(qy1_c-(radio+1));
        if (px_c>=1)
            px_c--;
  }
 }while(px_c>=1);
```

6.6.3. Algoritmo creado para la aceleración.

Para ajustar la aceleración lo cual nos permite tener una mayor velocidad de los motores debido a que se rompe el par de inercia lentamente, realizamos la siguiente función:

```
void ajustar_velocidad()
{
  NP=pasos_x-(200*cada_ciclo);
  if (NP==pasos_x)
    { if (espera>=3)
          espera--;
    }
 aceleracion++;
 cada_ciclo++;
}
```

6.6.4. Interfaz hombre máquina.

Para que la mesa fuera amigable con la operación del usuario se hizo una interfaz con el usuario para que el uso de la mesa fuera sencillo. Las funciones programadas fueron:

- Aceleración (X,Y) define la aceleración para el motor X y Y en pasos por segundo al cuadrado.
- Mueve_ Motor (X, Y) mueve el motor X y Y en X y Y cantidad de pasos a la velocidad y aceleración programada con anterioridad con movimiento en forma de interpolación.
- Jogging (motor,motor) mueve el motor indicado a velocidad programada sin importar la posición, puede indicar hasta dos motores i.e. jogging(x,y) o jogging (x).
- Stop(motor,motor) parar el motor indicado
- Rutina programada para escribir en la mesa las siglas UAQ.

7. Conclusiones:

La primera vez que hice este trabajo no requería justificar el porque estaba haciendo las cosas así que me concentré en el funcionamiento completo de la mesa, ya teniendo un funcionamiento completo creí que ya sabía todo lo referente a ella. En el aspecto del funcionamiento considero que fue aceptable pues fueron algunas linealidades mecánicas las que introdujeron variaciones muy pequeñas en la posición mecánica.

Sin embargo, al momento de redactar la tesis y al justificar cada paso dado me di cuenta que el crear una mesa de coordenadas a nivel industrial de alta calidad puede ser muy tardado si no se cuenta con un equipo multidisciplinario especialista en cada área, ya que no se pueden adquirir todos los conocimientos necesarios para crearla en el tiempo en que dura la carrera. Sin embargo no todo está perdido, pues si se tiene tiempo para poder diseñarla, la carrera de Ingeniero en Instrumentación y Control de Procesos ha proporcionado el conocimiento básico de los conceptos aquí mencionados, lo único que tuve que hacer fue el profundizar en ellos. De lo anterior ofrezco mis respetos a todos los maestros que me formaron pues muchas de sus enseñanzas fueron aplicadas aquí. A continuación puedo hacer algunas observaciones para las mejoras de algunos capítulos debido a que son muy extensas como para tratarse en un solo documento.

Del **Capítulo 1** Diseño de la parte mecánica, se puede usar un tornillo de bolas en lugar del tornillo de dientes en "V", pues el tornillo de bolas le podría dar mayor precisión, se podría usar acoplamientos flexibles en lugar de rígidos para poder quitarle carga al motor además se le puede dar otro diseño a la mesa para que funcione mejor

En el **Capítulo 4** Diseño de la tarjeta de control de movimiento, se puede usar el bus PCI, y poder generar micropasos, aunque creo que el sólo diseño de la tarjeta puede ser tema de tesis.

El **Capítulo 5** Etapa de Potencia puede ser mejorado si se hace un driver de mayor potencia.

Y el **Capítulo 6** Software de generación de trayectorias, puede crecer todo lo que se quiera, tal vez haya cosas que ni siquiera me puedo imaginar que pueden ayudar a crecer a este capítulo.

Esta tesis me conduce más al ámbito de las máquinas herramienta y me doy cuenta que no es tan difícil como parecía ser, sólo falta mucho estudio y trabajo por hacer.

BIBLIOGRAFÍA

1.- Robótica Industrial, Mikell P. Groover, Mitchel Weiss, Roger N. Nagel y Nicholas G. Odrey, Mc Graw Hill, Primera edición Febrero de 1989.
2.- Robot Dynamics and Control, Mark W. Spong, M. Vidyasagar, Impreso por Quinn-Woodbine, Inc., 1987.
3.- Electromechanical Systems, Electric Machines and Applied Mechatronics, Sergey E. Lyshevsky, Impreso por CRC Press, 2000.
4.- The Industrial Electronics handbook, Editor en jefe J. David Irwin, Impreso por CRC Press e IEEE Press, 1997.
5.- Mechanical Enginners Handbook, Editor Myer Kutz, Impreso por Wiley-Interscience, Segunda edición, 1998.
6.- Mechanical Design Handbook, Harold A. Rothbart, Editorial Mc Graw Hill, 1996.
7.- Manual del Ingeniero Mecánico, Baumeister y Marks, Editorial UTEHA, 1960.
8.- Autómatas programables, Joseph Balcells, José Luis Romeral, Editorial Alfaomega, 1998.
9.- Análisis de Circuitos en Ingeniería, William H. Hayt, Jr., Jack E. Kemmerly, Editorial Mc Graw Hill, Quinta edición 1993.
10.- Sistemas Electrónicos Digitales, Enrique Mandado, Editorial Alfaomega, Séptima edición 1998.
11.- Electrónica Industrial: Técnicas de potencia, J.A. Gualda, S. Martínez, P.M. Martínez, Editorial Alfaomega, Segunda Edición 1998.
12.- Electronics in Industry, George M. Chute, Robert D. Chute, Impreso por GLENCOE DIVISIÓN Macmillan/Mc Graw Hill, Quinta edición 2001.
13.- The Electronics Handbook, Editor en jefe Jerry C. Whitaker, Impreso por CRC Press e IEEE Press, 1996.
14.- The Electrical Engineering Handbook, Editor en jefe Richard C. Dorf, Impreso por CRC Press e IEEE Press, 1997.
15.- Eletrónica Industrial Moderna, Timothy J. Maloney, Editorial Prentice Hall, tercera edición 1997.
16.- VHDL Design Representation and Síntesis, James R. Armstrong, F. Gail Gray, Editorial Prentice Hall PTR, Segunda edición 2000.
17.- Fundamentos de los Microprocesadores, Roger L. Tokheim, Editorial Mc Graw Hill, Segunda edición 1985.
18.- The Mechatronics Handbook, Editor en Jefe Robert H. Bishop, impreso por CRC Press e ISA, 2002.
19.- Electrónica de Potencia, Circuitos, Dispositivos y Aplicaciones, Muhammad H. Rashid, Pearson Education, Prentice May, Segunda Edición 1993.
20.- Electrónica: Teoría de circuitos, Robert L. Boylestad, Louis Nashelsky, Pearson Education, Prentice May, Sexta Edición 1997.
21.- Inductrial Electronics and Robotics, Shuler, Mc Namee Editorial Mc Graw Hill, 1986.
22.- Analog and Computer electronics for Scientists, Publicada por Wiley-Interscience, John Wiley & Sons, Inc., Cuarta edición 1993.
23.- The designer's guide to VHDL, Peter J. Ashenden, Publicada por Morgan Kaufman Publishers Inc., 1996.
24.- Arquitectura de Computadoras y Procesamiento Paralelo, Kai Wwang/Fayé A. Briggs, Editorial Mc Graw Hill, 1988.
25.- Organización y Arquitectura de Computadoras, William Stallings, Editorial Prentice Hal, Quinta edición, 2000.
26.- The Circuits and Filtres Handbook, Wai-Kai Chen, impreso por CRC Press e IEEE Press, 1995.
27.- Fundamentals of Logic Design, Charles H. Roth, Jr. Publicado por PWS Publishing Company, Cuarta edición 1995.
28.- Digital Electronics with PLD Integration, Nigel P. Cook, Editorial Prentice Hall, 2001.
29.- A Guide to VHDL, Stanley Mazor, Patricia Langstrat, Publicado por Kluwer Academic Publishers, Segunda Edición 1993.

30.- Organización y Arquitectura de Computadoras, Jaime Martínez Garza, Editorial Prentice Hall, 2000.
31.- Ingeniería Computacional, Diseño de Hardware, M. Morris Mano, Editorial Prentice Hall Hispanoamericana, 1991.
32.- Programmable Logic Handbook, Geoff Bostock, Publicado por Butterworth Heinemann Ltd, Segunda edición, 1994.
33.- The Control Handbook, Editor William S. Levine, Impreso por CRC Press e IEEE Press, 1996.
34.- Ingeniería de Control Moderna, Katsuhiko Ogata, Editorial Pearson-Prentice Hall, Tercera edición 1998.
35.- Lenguaje Ensamblador para microcomputadoras IBM, J. Ferry Godfrey, Editorial Prentice Hall, 1991.
36.- Ensamblador básico, A. Rojas, Editorial Computec, 1995.
37.- Los microprocesadores Intel, Barry B. Brey, Editorial Prentice Hall, Quinta edición 2001.
38.- Computer Control of Manufactiring Systems, Yoram Koren, Mc Graw Hill,1983.

ANEXOS

Anexo 2.1.

Motor a pasos escap modelo P532.258.0.7.69.

High performance stepper motors
The disc magnet motor technology

The exceptional possibilities offered by the escap® line of disc magnet stepper motors are unequalled by any other kind of stepper motor. Their advanced technology, developed and patented by Portescap, allows for truly exceptional dynamic performance. The rotor of these motors consists of a rare earth magnet having the shape of a thin disc which is axially magnetized.

A particular magnetization method allows for a high number of magnetic poles, giving much smaller step angles than conventional two-phase permanent magnet stepper motors. Such a rotor design has a very low moment of inertia, resulting in outstanding acceleration and dynamic behaviour. These features, together with high peak speeds, mean that any incremental movement is carried out in the shortest possible time.

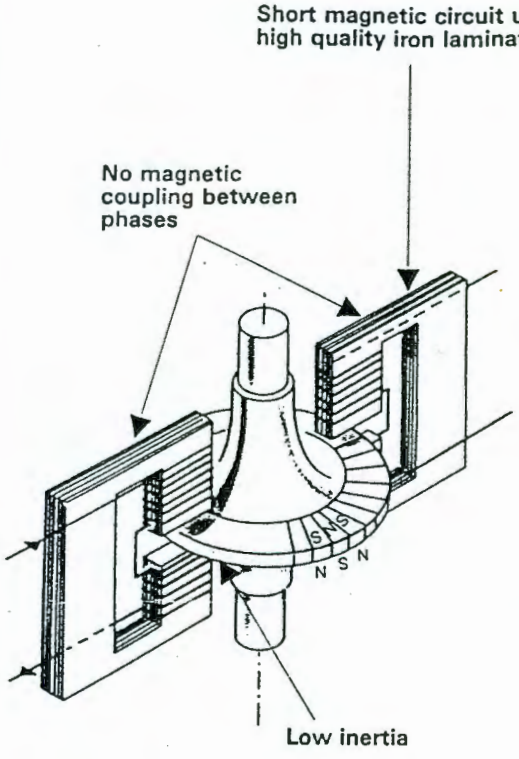
Low inertia also means high start/stop frequencies allowing to save time during the first step and to solve certain motion problems without applying a ramp.

The stator is designed for the shortest possible magnetic circuit, using high quality iron laminations. This gives low iron losses and more torque at high speed.

escap® disc magnet stepper motors therefore are ideally suited for applications where rapid and precise incremental motion is to be achieved in a simple open loop drive.

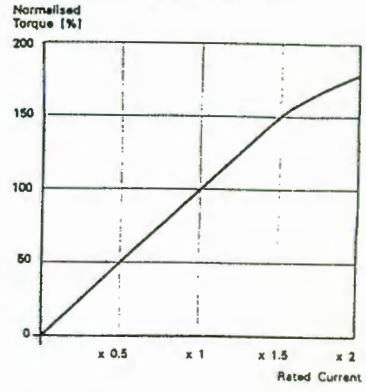
The unique electromagnetic characteristics of disc magnet motors permit them to operate well below any saturation of the magnetic circuit, where torque is truly proportional to current. By current boosting, performance can momentarily be pushed well above nominal values.

Those motors specially designed for microstepping feature a sinusoidal torque function with very low harmonic distortion and low detent torque. Excellent static and dynamic accuracy is obtained for any position and under any load or speed conditions.



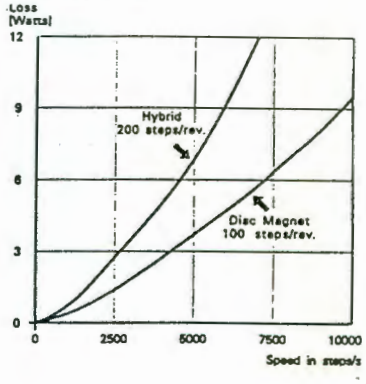
Iron saturation effects

Torque/Current
example: escap® motor type P532



Iron losses

Comparison DM/Hybrid same torque, losses due to magnet flux only.

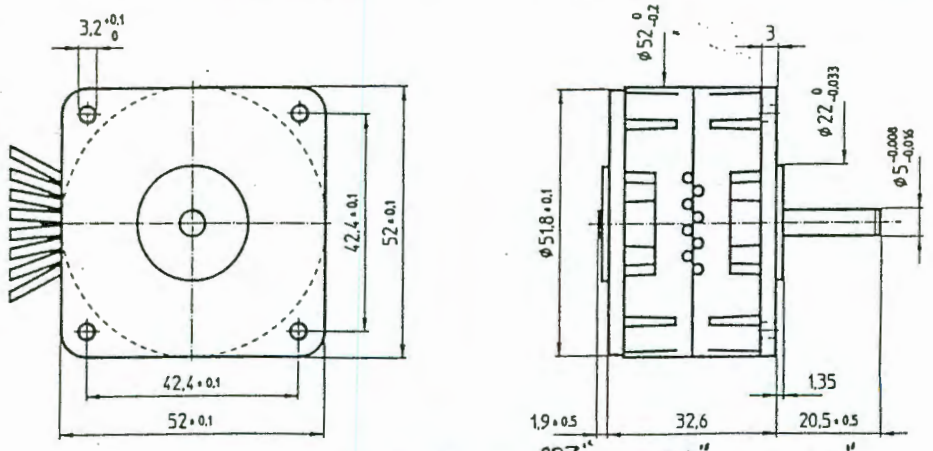


Features	Benefits
Low rotor inertia	High acceleration and power rate Reduced move time in incremental motion
	High start/stop frequencies
Very short magnetic circuit with low losses	More torque available at high speed
High quality iron laminations and no coupling between phases	Excellent linearity Pulse torque capability
High quality rare earth magnet	Largest magnetic energy in smallest envelope
Sinusoidal torque function	Precise positioning also in microstep mode

cap® P532

ep angle: 3.6°

A.4



.1968.5

.007" 1.28" .807"

2.1"

ale: 2/3
mensions in mm
ss: 250 g
d wires: 200 ± 15 mm
0.25 mm² (AWG 24)

P532-258...10

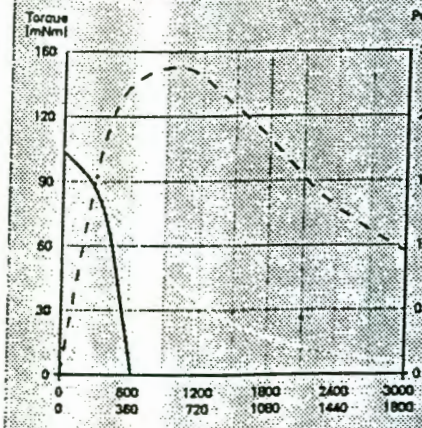
dard types available stock		P532-258 012 coils in series			P532-258 004 coils in series			P532-258 004 coils in parallel			P532-258 0.7 coils in parallel		
lependent parameters		min	typ	max	min	typ	max	min	typ	max	min	typ	max
resistance	ohm	24	27	29	8	8.8	9.5	2	2.2	2.4	0.3	0.34	0.38
inductance (1 kHz)	mH		64			20			5			0.7	
nal phase current (2 ph. on)	A		0.4			0.7			1.4			3.7	
nal phase current (1 ph. on)	A		0.56			1			2			5.2	
EMF amplitude	V/kst/s	17	21	25	10	12	14	5	6	7	1.8	2.3	2.8

ndependent parameters[1]		min	typ	max
e parameters				
ng torque (nominal current)	mNm (oz-in)	174 (24.6)	205 (29.0)	236 (33.4)
ng torque (twice nominal current)[2]	mNm (oz-in)	306 (43.3)	360 (51.0)	414 (58.6)
t torque amplitude and friction	mNm (oz-in)	14 (2.0)	28 (4.0)	40 (5.7)
nal parameters				
nal resistance coil-ambient[3]	°C/W		7.3	
emperature	°C			130
ting ambient temperature	°C	−20		50
lar accuracy				
ute accuracy (2 ph. on full-step mode)	% full-step		±3	±5
anical parameters				
inertia	$kgm^2 \cdot 10^{-7}$		12	
load[4]	N			20
load[5]	N			30
shaft play (5 N)	µm		10	25
shaft play (5 N)	µm		10	25
parameters				
oltage (1 min)	V_{RMS}		500	
al resonance frequency (nominal current)	Hz		330	
ical time constant	ms		2.3	
ar acceleration (nominal current)	rad/s^2		171000	
rate (nominal current)	kW/s		35	

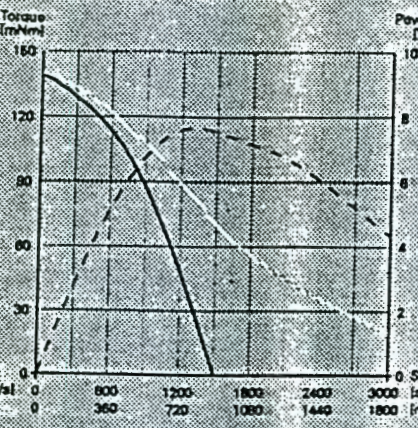
[1] Bipolar driver
[2] The maximum coil temperature must be respected
[3] Motor unmounted
[4] Load applied at 12 mm from mounting face
[5] Shaft must be supported for press-fitting a pulley or pinion

532 motor is also available from stock with the RG1/9 and K40 gearboxes (p. 60 and 62).
cular versions include options such as special shafts (hollow shaft), other gearboxes (R32, R40), optical encoders and rth.

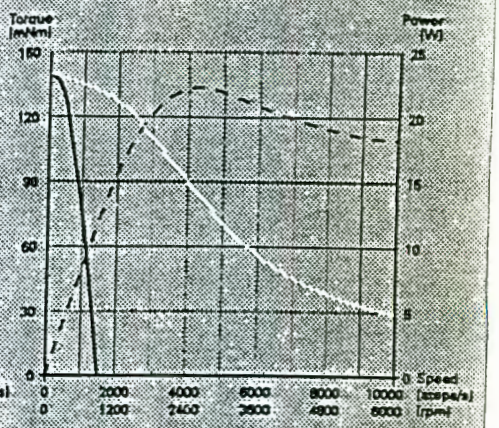
P532-258 004
Coils in series
escap® ELD-200
33Ω series resistor, 24V



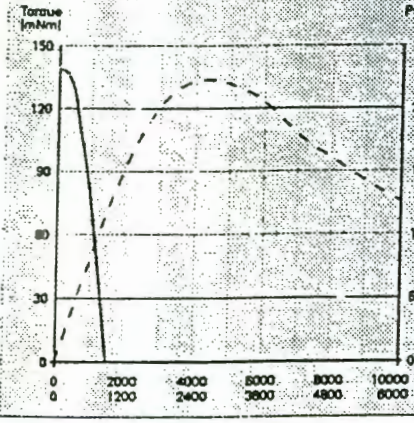
P532-258 012
Coils in series
escap® ELD-200
39Ω series resistor, 36V



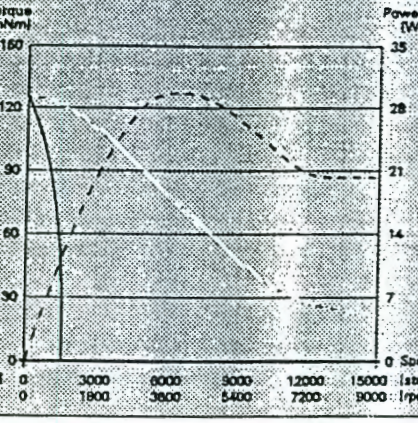
P532-258 004
Coils in parallel
escap® EDI-201
or EDM-453, 34V, 2A



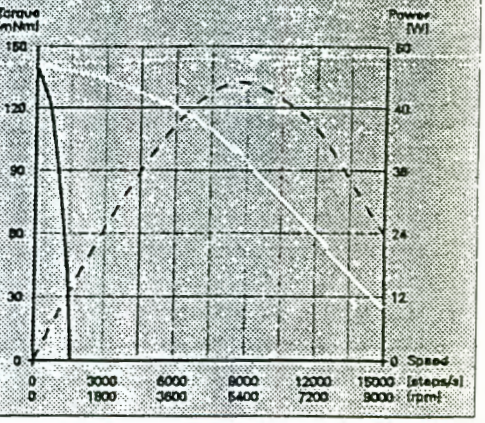
P532-258 004
Coils in parallel
escap® ESD-200
36V, 2A



P532-258 0.7
Coils in parallel
escap® ESD-300
36V, 3A



P532-258 0.7
Coils in parallel
escap® ESI-402
or EDB-106, 45V



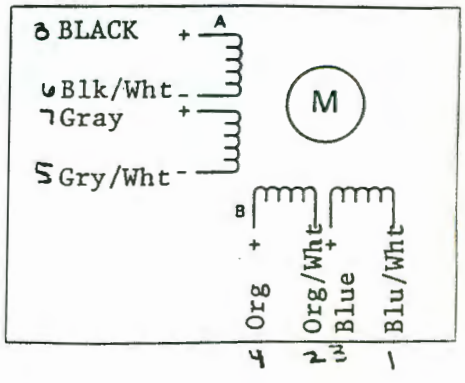
—— Pull-in range
Pull-out range
- - - Power output

Notes

The low inertia, extended pull-in range, high peak speed and boost torque capability of this motor are benefits for fast incremental motion.
The speed scale is indicated in full-steps/s for all drive modes.
The motor is driven in half-steps unless otherwise specified.
The motor is energised with nominal current unless otherwise specified.
Pull-in is measured with a load inertia equal to the rotor inertia.

The following escap® drive circuits are recommended with the P532 motor, depending on the drive mode and the dynamic performance required:
ELD-200, ESD-200, ESD-300, ESI-402, EDB-106.
Please refer to page 104/105 for more information on terminology and definitions.

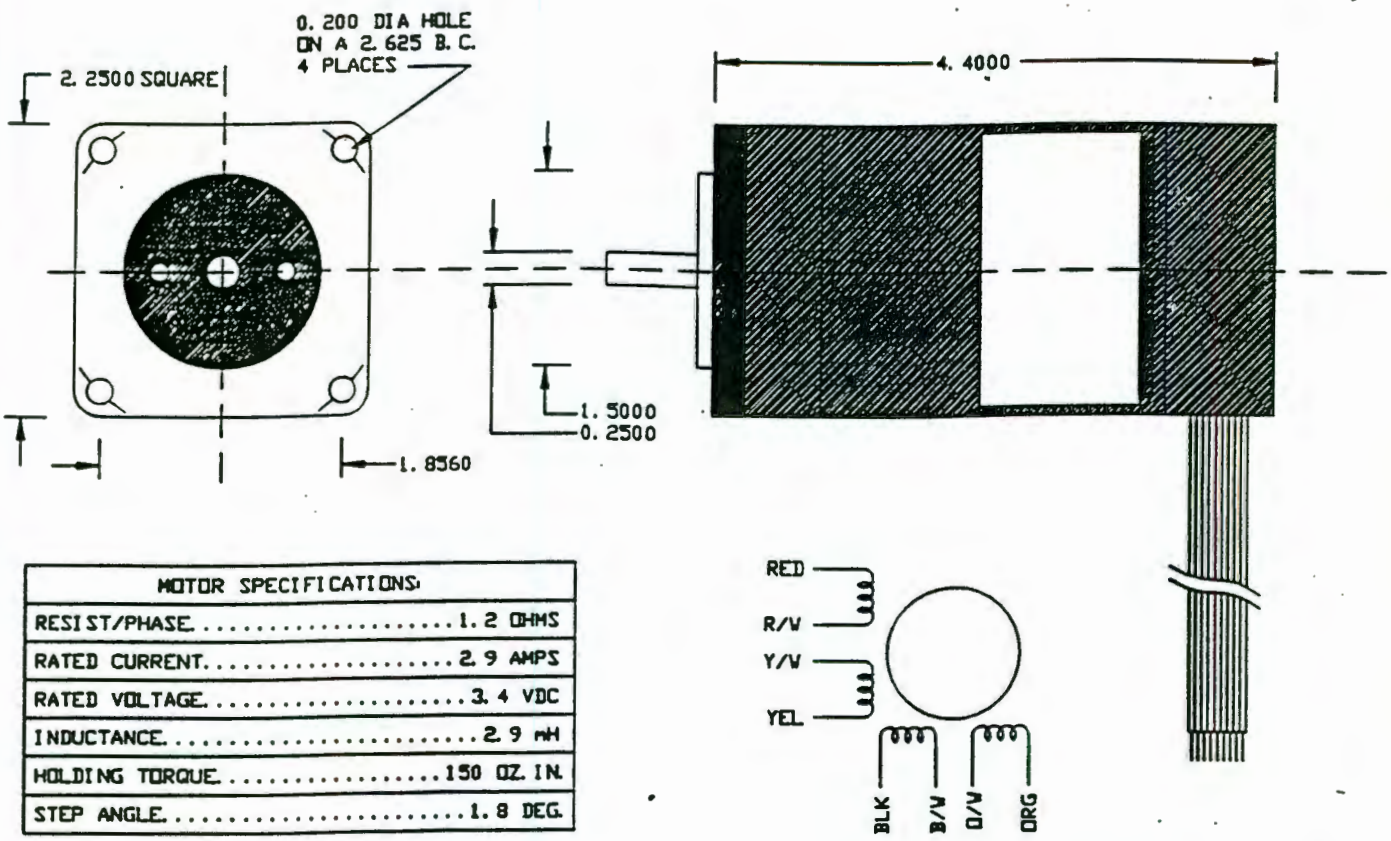
Motor connections



Anexo 2.2.

Hoja de especificaciones del motor a pasos Marca Bodine Electric modelo 23T3BEHH .

SERVO SYSTEMS CO.
STOCK NO: SM-242
BODINE ELECTRIC CO.
P/N 23T3BEHH



MOTOR SPECIFICATIONS	
RESIST/PHASE	1.2 OHMS
RATED CURRENT	2.9 AMPS
RATED VOLTAGE	3.4 VDC
INDUCTANCE	2.9 mH
HOLDING TORQUE	150 OZ. IN.
STEP ANGLE	1.8 DEG.

WIRE COLOR	DRIVE CONNECTION
ORANGE	A +
ORG / WHITE	A COMMON
BLK / WHITE	
BLACK	A -
RED	B +
RED / WHITE	B COMMON
YEL / WHITE	
YELLOW	B -
UNIPOLAR OR BIPOLAR DRIVE SERIES CONNECTIONS	

WIRE COLOR	DRIVE CONNECTION
ORANGE	A +
BLK / WHITE	
ORG / WHITE	A -
BLACK	
RED	B +
YEL / WHITE	
RED / WHITE	B -
YELLOW	
BIPOLAR DRIVE PARALLEL CONNECTIONS	

Anexo 4.1.

Hoja de especificaciones de la GAL20V8.

Lattice Semiconductor Corporation

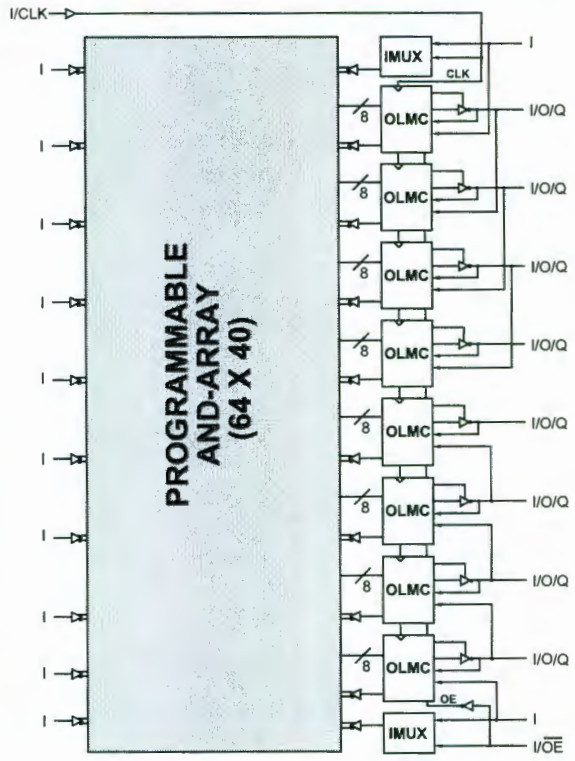
GAL20V8

High Performance E²CMOS PLD Generic Array Logic™

Features

- **HIGH PERFORMANCE E²CMOS® TECHNOLOGY**
  - — 5 ns Maximum Propagation Delay
  - — Fmax = 166 MHz
  - — 4 ns Maximum from Clock Input to Data Output
  - — UltraMOS® Advanced CMOS Technology
- **50% to 75% REDUCTION IN POWER FROM BIPOLAR**
  - — 75mA Typ Icc on Low Power Device
  - — 45mA Typ Icc on Quarter Power Device
- **ACTIVE PULL-UPS ON ALL PINS**
- **E² CELL TECHNOLOGY**
  - — Reconfigurable Logic
  - — Reprogrammable Cells
  - — 100% Tested/100% Yields
  - — High Speed Electrical Erasure (<100ms)
  - — 20 Year Data Retention
- **EIGHT OUTPUT LOGIC MACROCELLS**
  - — Maximum Flexibility for Complex Logic Designs
  - — Programmable Output Polarity
  - — Also Emulates 24-pin PAL® Devices with Full Function/ Fuse Map/Parametric Compatibility
- **PRELOAD AND POWER-ON RESET OF ALL REGISTERS**
  - — 100% Functional Testability
- **APPLICATIONS INCLUDE:**
  - — DMA Control
  - — State Machine Control
  - — High Speed Graphics Processing
  - — Standard Logic Speed Upgrade
- **ELECTRONIC SIGNATURE FOR IDENTIFICATION**

Functional Block Diagram



Description

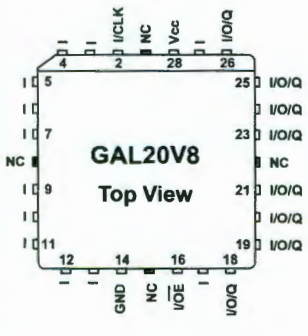
The GAL20V8C, at 5ns maximum propagation delay time, combines a high performance CMOS process with Electrically Erasable (E²) floating gate technology to provide the highest speed performance available in the PLD market. High speed erase times (<100ms) allow the devices to be reprogrammed quickly and efficiently.

The generic architecture provides maximum design flexibility by allowing the Output Logic Macrocell (OLMC) to be configured by the user. An important subset of the many architecture configurations possible with the GAL20V8 are the PAL architectures listed in the table of the macrocell description section. GAL20V8 devices are capable of emulating any of these PAL architectures with full function/fuse map/parametric compatibility.

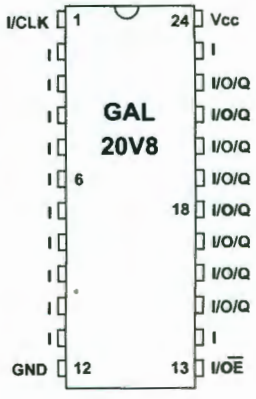
Unique test circuitry and reprogrammable cells allow complete AC, DC, and functional testing during manufacture. As a result, Lattice Semiconductor delivers 100% field programmability and functionality of all GAL products. In addition, 100 erase/write cycles and data retention in excess of 20 years are specified.

Pin Configuration

PLCC



DIP



LATTICE SEMICONDUCTOR CORP., 5555 Northeast Moore Ct., Hillsboro, Oregon 97124, U.S.A.
Tel. (503) 268-8000; 1-800-LATTICE; FAX (503) 268-8556; http://www.latticesemi.com

August 2000

Lattice Semiconductor Corporation

Specifications *GAL20V8*

GAL20V8 Ordering Information

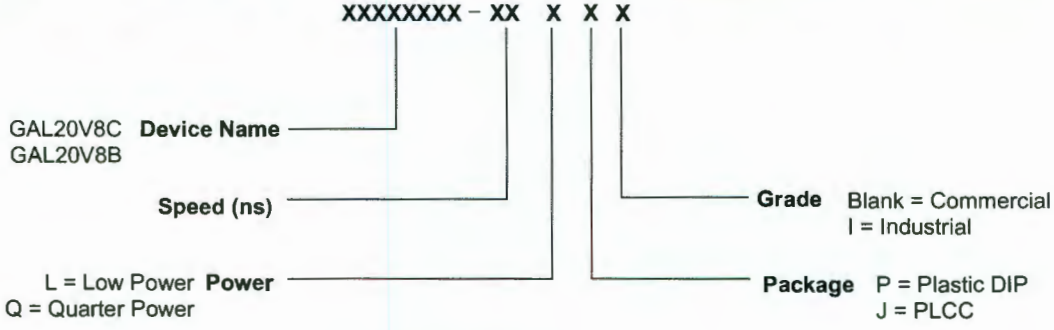
Commercial Grade Specifications

Tpd (ns)	Tsu (ns)	Tco (ns)	Icc (mA)	Ordering #	Package
5	3	4	115	GAL20V8C-5LJ	28-Lead PLCC
7.5	7	5	115	GAL20V8C-7LJ	28-Lead PLCC
			115	GAL20V8B-7LP	24-Pin Plastic DIP
			115	GAL20V8B-7LJ	28-Lead PLCC
10	10	7	115	GAL20V8C-10LJ	28-Lead PLCC
			115	GAL20V8B-10LP	24-Pin Plastic DIP
			115	GAL20V8B-10LJ	28-Lead PLCC
15	12	10	55	GAL20V8B-15QP	24-Pin Plastic DIP
			55	GAL20V8B-15QJ	28-Lead PLCC
			90	GAL20V8B-15LP	24-Pin Plastic DIP
			90	GAL20V8B-15LJ	28-Lead PLCC
25	15	12	55	GAL20V8B-25QP	24-Pin Plastic DIP
			55	GAL20V8B-25QJ	28-Lead PLCC
			90	GAL20V8B-25LP	24-Pin Plastic DIP
			90	GAL20V8B-25LJ	28-Lead PLCC

Industrial Grade Specifications

Tpd (ns)	Tsu (ns)	Tco (ns)	Icc (mA)	Ordering #	Package
10	10	7	130	GAL20V8C-10LJI	28-Lead PLCC
			130	GAL20V8B-10LPI	24-Pin Plastic DIP
			130	GAL20V8B-10LJI	28-Lead PLCC
15	12	10	130	GAL20V8B-15LPI	24-Pin Plastic DIP
			130	GAL20V8B-15LJI	28-Lead PLCC
20	13	11	65	GAL20V8B-20QPI	24-Pin Plastic DIP
			65	GAL20V8B-20QJI	28-Lead PLCC
25	15	12	65	GAL20V8B-25QPI	24-Pin Plastic DIP
			65	GAL20V8B-25QJI	28-Lead PLCC
			130	GAL20V8B-25LPI	24-Pin Plastic DIP
			130	GAL20V8B-25LJI	28-Lead PLCC

Part Number Description



Lattice Semiconductor Corporation

Specifications *GAL20V8*

Output Logic Macrocell (OLMC)

The following discussion pertains to configuring the output logic macrocell. It should be noted that actual implementation is accomplished by development software/hardware and is completely transparent to the user.

There are three global OLMC configuration modes possible: **simple**, **complex**, and **registered**. Details of each of these modes is illustrated in the following pages. Two global bits, SYN and AC0, control the mode configuration for all macrocells. The XOR bit of each macrocell controls the polarity of the output in any of the three modes, while the AC1 bit of each of the macrocells controls the input/output configuration. These two global and 16 individual architecture bits define all possible configurations in a GAL20V8 . The information given on these architecture bits is only to give a better understanding of the device. Compiler software will transparently set these architecture bits from the pin definitions, so the user should not need to directly manipulate these architecture bits.

The following is a list of the PAL architectures that the GAL20V8 can emulate. It also shows the OLMC mode under which the devices emulate the PAL architecture.

PAL Architectures Emulated by GAL20V8	GAL20V8 Global OLMC Mode
20R8	Registered
20R6	Registered
20R4	Registered
20RP8	Registered
20RP6	Registered
20RP4	Registered
20L8	Complex
20H8	Complex
20P8	Complex
14L8	Simple
16L6	Simple
18L4	Simple
20L2	Simple
14H8	Simple
16H6	Simple
18H4	Simple
20H2	Simple
14P8	Simple
16P6	Simple
18P4	Simple
20P2	Simple

Compiler Support for OLMC

Software compilers support the three different global OLMC modes as different device types. These device types are listed in the table below. Most compilers have the ability to automatically select the device type, generally based on the register usage and output enable (OE) usage. Register usage on the device forces the software to choose the registered mode. All combinatorial outputs with OE controlled by the product term will force the software to choose the complex mode. The software will choose the simple mode only when all outputs are dedicated combinatorial without OE control. The different device types listed in the table can be used to override the automatic device selection by the software. For further details, refer to the compiler software manuals.

When using compiler software to configure the device, the user must pay special attention to the following restrictions in each mode. In **registered mode** pin 1 and pin 13 (DIP pinout) are permanently configured as clock and output enable, respectively. These pins cannot be configured as dedicated inputs in the registered mode.

In **complex mode** pin 1 and pin 13 become dedicated inputs and use the feedback paths of pin 22 and pin 15 respectively. Because of this feedback path usage, pin 22 and pin 15 do not have the feedback option in this mode.

In **simple mode** all feedback paths of the output pins are routed via the adjacent pins. In doing so, the two inner most pins (pins 18 and 19) will not have the feedback option as these pins are always configured as dedicated combinatorial output.

	Registered	Complex	Simple	Auto Mode Select
ABEL	P20V8R	P20V8C	P20V8AS	P20V8
CUPL	G20V8MS	G20V8MA	G20V8AS	G20V8
LOG/iC	GAL20V8_R	GAL20V8_C7	GAL20V8_C8	GAL20V8
OrCAD-PLD	"Registered"[1]	"Complex"[1]	"Simple"[1]	GAL20V8A
PLDesigner	P20V8R[2]	P20V8C[2]	P20V8C[2]	P20V8A
TANGO-PLD	G20V8R	G20V8C	G20V8AS[3]	G20V8

1) Used with **Configuration** keyword.
2) Prior to Version 2.0 support.
3) Supported on Version 1.20 or later.

Registered Mode

In the Registered mode, macrocells are configured as dedicated registered outputs or as I/O functions.

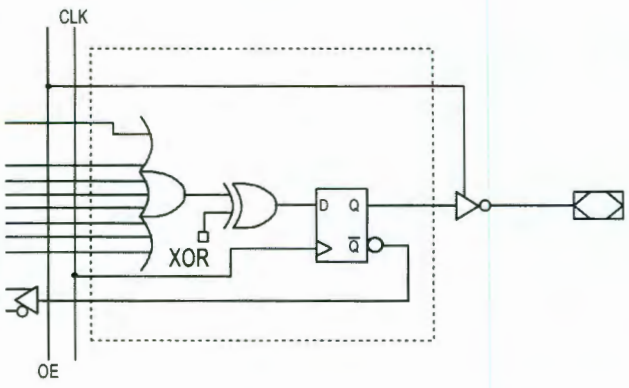
Architecture configurations available in this mode are similar to the common 20R8 and 20RP4 devices with various permutations of polarity, I/O and register placement.

All registered macrocells share common clock and output enable control pins. Any macrocell can be configured as registered or I/O. Up to eight registers or up to eight I/Os are possible in this mode.

Dedicated input or output functions can be implemented as subsets of the I/O function.

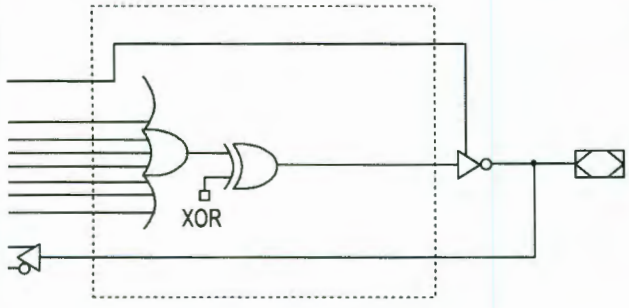
Registered outputs have eight product terms per output. I/Os have seven product terms per output.

The JEDEC fuse numbers, including the User Electronic Signature (UES) fuses and the Product Term Disable (PTD) fuses, are shown on the logic diagram on the following page.



Registered Configuration for Registered Mode

- SYN=0.
- AC0=1.
- XOR=0 defines Active Low Output.
- XOR=1 defines Active High Output.
- AC1=0 defines this output configuration.
- Pin 1 controls common CLK for the registered outputs.
- Pin 13 controls common $\overline{OE}$ for the registered outputs.
- Pin 1 & Pin 13 are permanently configured as CLK & $\overline{OE}$ for registered output configuration.



Combinatorial Configuration for Registered Mode

- SYN=0.
- AC0=1.
- XOR=0 defines Active Low Output.
- XOR=1 defines Active High Output.
- AC1=1 defines this output configuration.
- Pin 1 & Pin 13 are permanently configured as CLK & $\overline{OE}$ for registered output configuration..

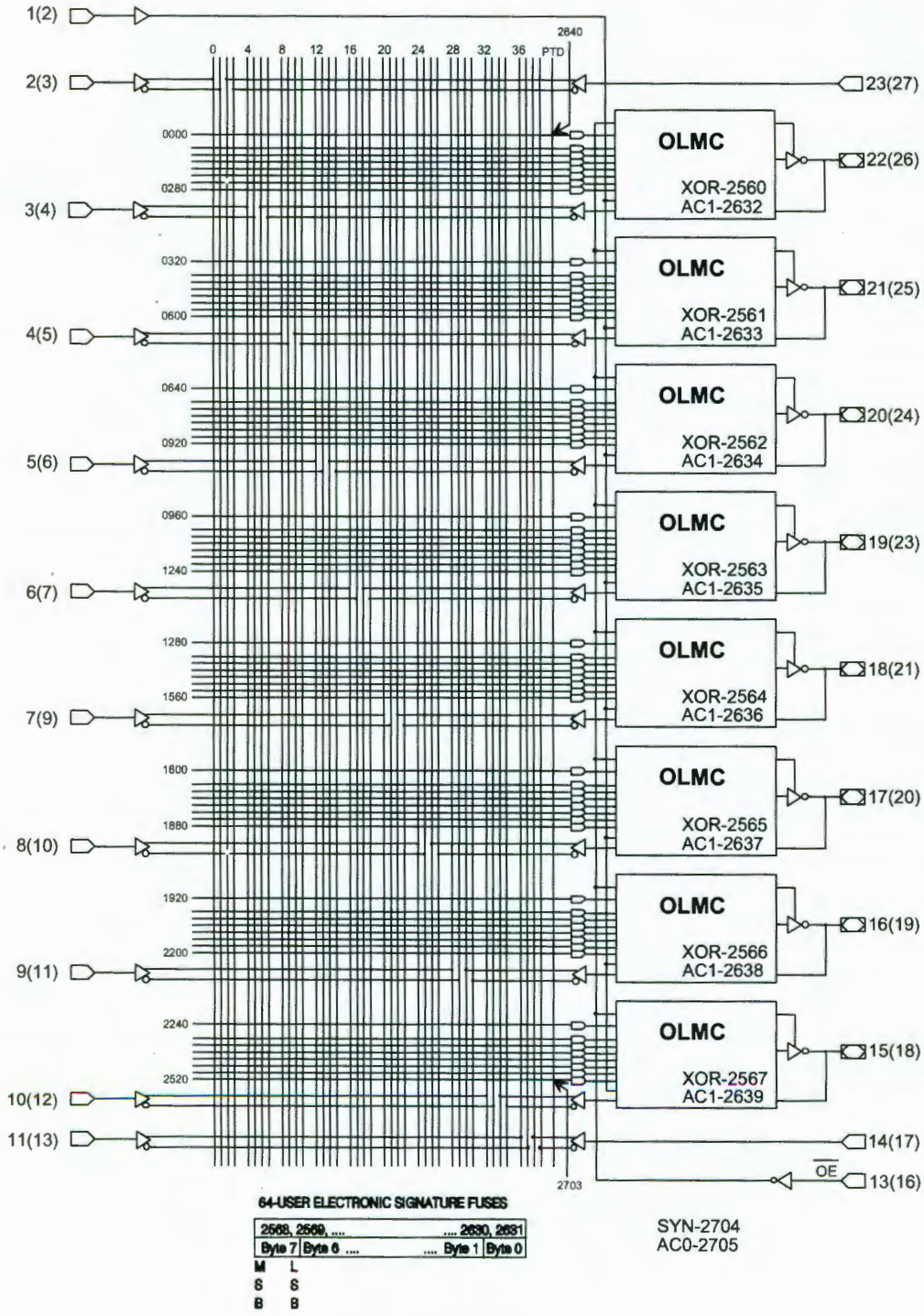
Note: The development software configures all of the architecture control bits and checks for proper pin usage automatically.

Lattice Semiconductor Corporation

Specifications *GAL20V8*

Registered Mode Logic Diagram

DIP (PLCC) Package Pinouts



Anexo 4.2. Decodificación de señales en lenguaje VHDL para selección de chip CS Y CS2.

```
library IEEE;
use IEEE.std_logic_1164.all;

entity component_1 is
        port (
                Entradas: in STD_LOGIC_VECTOR (13 downto 1);
                CS: out STD_LOGIC;
                CS2: out STD_LOGIC
        );
end component_1;

architecture component_1 of component_1 is
begin

process(Entradas)
begin
        case Entradas is
                when "1100000000010" => CS <= '1';
                when others => CS <= '0' ;
        end case;
end process;

process(Entradas)
begin
        case Entradas is
                when "1100000001010" => CS2 <= '1';
                when others => CS2 <= '0' ;
        end case;
end process;

end component_1;
```

Anexo 4.3. Archivo de reporte para la conexión a circuitos CS y CS2 usando una GAL20V8.

```
     | | | | | | |
   _________________
  -|                 |-
  -|                 |-
  -|                 |-
  -|     CYPRESS     |-
  -|                 |-
  -|                 |-   Warp VHDL Synthesis Compiler: Version 4 IR x66
  -|                 |-   Copyright (C) 1991, 1992, 1993,
   |_________________|                  1994, 1995, 1996 Cypress
Semiconductor
     | | | | | | |

========================================================================
Compiling:  compon~1.vhd
Options:    -q -yv2 -e10 -w10 -o2 -yga -fP -v10 -yb -yp -dc20v8 -
pPALCE20V8-15LMB compon~1.vhd
========================================================================

C:\WARP\BIN\VHDLFE.EXE V4 IR x66:  VHDL parser
Sat Nov 08 22:01:38 2003

Library 'work' => directory 'lc20v8'
Library 'ieee' => directory 'C:\warp\lib\ieee\work'
Using 'C:\warp\lib\ieee\work\stdlogic.vif'.

C:\WARP\BIN\VHDLFE.EXE:  No errors.


C:\WARP\BIN\TOVIF.EXE V4 IR x66:  High-level synthesis
Sat Nov 08 22:01:39 2003

Note:  Removing wires from arch. 'component_1' of entity 'component_1'.

C:\WARP\BIN\TOVIF.EXE:  No errors.


C:\WARP\BIN\TOPLD.EXE V4 IR x66:  Synthesis and optimization
Sat Nov 08 22:01:39 2003


------------------------------------------------------------
Detecting unused logic.
------------------------------------------------------------



------------------------------------------------------
Alias Detection
------------------------------------------------------

------------------------------------------------------
Aliased 0 equations, 0 wires.
------------------------------------------------------

------------------------------------------------------------
Circuit simplification
```

```
------------------------------------------------------------

Substituting virtuals - pass 1:


------------------------------------------------------------
Circuit simplification results:

      Expanded 0 signals.
      Turned 0 signals into soft nodes.
      Maximum expansion cost was set at 10.
------------------------------------------------------------
Created 15 PLD nodes.

C:\WARP\BIN\TOPLD.EXE:  No errors.

------------------------------------------------------------------------
---
PLD Optimizer Software:       DSGNOPT.EXE    17/JUL/96    [v3.22 ] 4 IR
x66

DESIGN HEADER INFORMATION  (22:01:40)

Input File(s): compon~1.pla
Device       : c20v8
Package      : PALCE20V8-15LMB
ReportFile   : compon~1.rpt

Program Controls:
                None.

Signal Requests:
    GROUP USEPOL ALL

Completed Successfully
------------------------------------------------------------------------
---
PLD Optimizer Software:       DSGNOPT.EXE    17/JUL/96    [v3.22 ] 4 IR
x66

OPTIMIZATION OPTIONS       (22:01:40)

Messages:
  Information: Selected logic optimization OFF for signals:
         cs2 cs


Summary:
                Error Count = 0      Warning Count = 0

Completed Successfully
------------------------------------------------------------------------
---
PLD Optimizer Software:       MINOPT.EXE     17/JUL/96    [v3.22 ] 4 IR
x66

LOGIC MINIMIZATION         (22:01:40)

Messages:


Summary:
```

```
                Error Count = 0      Warning Count = 0

Completed Successfully
-------------------------------------------------------------------------
---
PLD Optimizer Software:       DSGNOPT.EXE    17/JUL/96    [v3.22 ] 4 IR
x66

OPTIMIZATION OPTIONS       (22:01:40)

Messages:


Summary:
                Error Count = 0      Warning Count = 0

Completed Successfully
-------------------------------------------------------------------------
---
PLD Compiler Software:        PLA2JED.EXE    17/JUL/96    [v3.22 ] 4 IR
x66

DESIGN EQUATIONS           (22:01:41)


    cs =
          entradas_13 * entradas_12 * /entradas_11 * /entradas_10 *
          /entradas_9 * /entradas_8 * /entradas_7 * /entradas_6 *
          /entradas_5 * /entradas_4 * /entradas_3 * entradas_2 *
          /entradas_1

    cs2 =
          entradas_13 * entradas_12 * /entradas_11 * /entradas_10 *
          /entradas_9 * /entradas_8 * /entradas_7 * /entradas_6 *
          /entradas_5 * entradas_4 * /entradas_3 * entradas_2 *
          /entradas_1


Completed Successfully
-------------------------------------------------------------------------
---
PLD Compiler Software:        PLA2JED.EXE    17/JUL/96    [v3.22 ] 4 IR
x66

DESIGN RULE CHECK          (22:01:41)

Messages:
                None.


Summary:
                Error Count = 0      Warning Count = 0

Completed Successfully
-------------------------------------------------------------------------
---
PLD Compiler Software:        PLA2JED.EXE    17/JUL/96    [v3.22 ] 4 IR
x66

DESIGN SIGNAL PLACEMENT    (22:01:41)

Messages:
  Information: Checking for duplicate NODE logic.
```

None.

```
                              C20V8A
                 ____________________________________
  entradas_1 =|  1|                              |24|* not used
  entradas_2 =|  2|                              |23|* not used
  entradas_3 =|  3|                              |22|= cs2
  entradas_4 =|  4|                              |21|* not used
  entradas_5 =|  5|                              |20|* not used

  entradas_6 =|  6|                              |19|* not used
  entradas_7 =|  7|                              |18|* not used
  entradas_8 =|  8|                              |17|* not used
  entradas_9 =|  9|                              |16|* not used
 entradas_10 =|10|                               |15|= cs
 entradas_11 =|11|                               |14|= entradas_13
    not used *|12|                               |13|= entradas_12
                 ____________________________________
```

Summary:
Error Count = 0   Warning Count = 0

Completed Successfully

PLD Compiler Software: PLA2JED.EXE   17/JUL/96   [v3.22] 4 IR x66

RESOURCE ALLOCATION   (22:01:41)

Information: Macrocell Utilization.

Description	Used	Max
Dedicated Inputs	13	14
Output Macrocells	2	2
I/O Macrocells	0	6
	15 /	22 = 68 %

Information: Output Logic Product Term Utilization.

Node#	Output Signal Name	Used	Max
15	cs	1	7
16	Unused	0	7
17	Unused	0	7
18	Unused	0	7
19	Unused	0	7
20	Unused	0	7
21	Unused	0	7
22	cs2	1	7
		2 /	56 = 3 %

Completed Successfully

Anexo 4.4.

Hoja de especificaciones del 74LS373.

FAIRCHILD
SEMICONDUCTOR™

April 1986
Revised November 2001

DM74LS373 • DM74LS374
3-STATE Octal D-Type Transparent Latches and Edge-Triggered Flip-Flops

General Description

These 8-bit registers feature totem-pole 3-STATE outputs designed specifically for driving highly-capacitive or relatively low-impedance loads. The high-impedance state and increased high-logic level drive provide these registers with the capability of being connected directly to and driving the bus lines in a bus-organized system without need for interface or pull-up components. They are particularly attractive for implementing buffer registers, I/O ports, bidirectional bus drivers, and working registers.

The eight latches of the DM74LS373 are transparent D-type latches meaning that while the enable (G) is HIGH the Q outputs will follow the data (D) inputs. When the enable is taken LOW the output will be latched at the level of the data that was set up.

The eight flip-flops of the DM74LS374 are edge-triggered D-type flip flops. On the positive transition of the clock, the Q outputs will be set to the logic states that were set up at the D inputs.

A buffered output control input can be used to place the eight outputs in either a normal logic state (HIGH or LOW logic levels) or a high-impedance state. In the high-impedance state the outputs neither load nor drive the bus lines significantly.

The output control does not affect the internal operation of the latches or flip-flops. That is, the old data can be retained or new data can be entered even while the outputs are OFF.

Features

- Choice of 8 latches or 8 D-type flip-flops in a single package
- 3-STATE bus-driving outputs
- Full parallel-access for loading
- Buffered control inputs
- P-N-P inputs reduce D-C loading on data lines

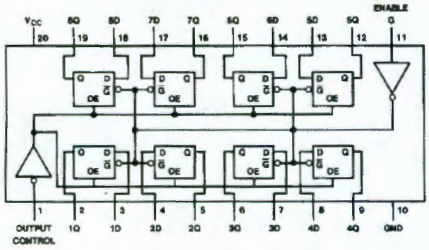
Ordering Code:

Order Number	Package Number	Package Description
DM74LS373WM	M20B	20-Lead Small Outline Integrated Circuit (SOIC), JEDEC MS-013, 0.300" Wide
DM74LS373SJ	M20D	20-Lead Small Outline Package (SOP), EIAJ TYPE II, 5.3mm Wide
DM74LS373N	N20A	20-Lead Plastic Dual-In-Line Package (PDIP), JEDEC MS-001, 0.300" Wide
DM74LS374WM	M20B	20-Lead Small Outline Integrated Circuit (SOIC), JEDEC MS-013, 0.300" Wide
DM74LS374SJ	M20D	20-Lead Small Outline Package (SOP), EIAJ TYPE II, 5.3mm Wide
DM74LS374N	N20A	20-Lead Plastic Dual-In-Line Package (PDIP), JEDEC MS-001, 0.300" Wide

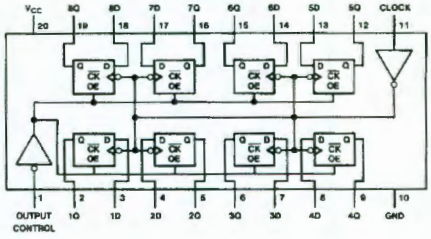
Devices also available in Tape and Reel. Specify by appending the suffix letter "X" to the ordering code.

Connection Diagrams

DM74LS373



DM74LS374



Function Tables

DM74LS373

Output Control	Enable G	D	Output
L	H	H	H
L	H	L	L
L	L	X	Q_0
H	X	X	Z

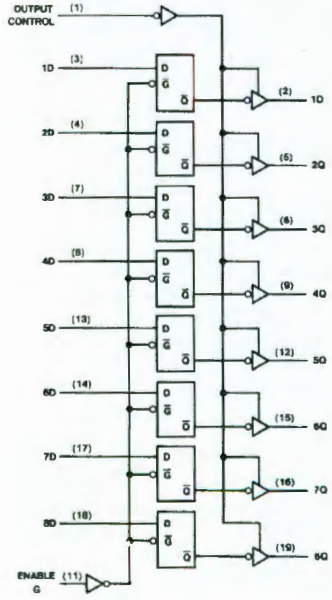
DM74LS374

Output Control	Clock	D	Output
L	↑	H	H
L	↑	L	L
L	L	X	Q_0
H	X	X	Z

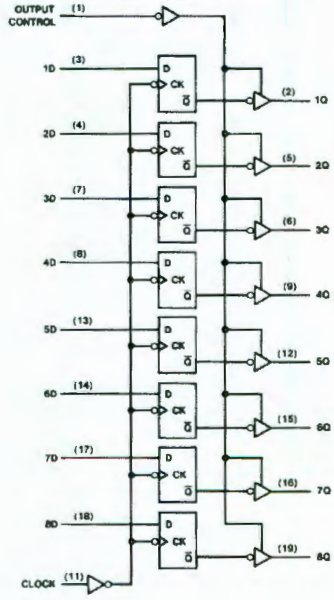
H = HIGH Level (Steady State) L = LOW Level (Steady State)
X = Don't Care Z = High Impedance State
↑ = Transition from LOW-to-HIGH level Q_0 = The level of the output before steady-state input conditions were established.

Logic Diagrams

DM74LS373
Transparent Latches



DM74LS374
Positive-Edge-Triggered Flip-Flops



DM74LS373 Switching Characteristics

at V_{CC} = 5V and T_A = 25°C

Symbol	Parameter	From (Input) To (Output)	R_L = 667Ω				Units
			C_L = 45 pF		C_L = 150 pF		
			Min	Max	Min	Max	
t_{PLH}	Propagation Delay Time LOW-to-HIGH Level Output	Data to Q		18		26	ns
t_{PHL}	Propagation Delay Time HIGH-to-LOW Level Output	Data to Q		18		27	ns
t_{PLH}	Propagation Delay Time LOW-to-HIGH Level Output	Enable to Q		30		38	ns
t_{PHL}	Propagation Delay Time HIGH-to-LOW Level Output	Enable to Q		30		36	ns
t_{PZH}	Output Enable Time to HIGH Level Output	Output Control to Any Q		28		36	ns
t_{PZL}	Output Enable Time to LOW Level Output	Output Control to Any Q		36		50	ns
t_{PHZ}	Output Disable Time from HIGH Level Output (Note 6)	Output Control to Any Q		20			ns
t_{PLZ}	Output Disable Time from LOW Level Output (Note 6)	Output Control to Any Q		25			ns

Note 6: C_L = 5 pF.

DM74LS374 Recommended Operating Conditions

Symbol	Parameter		Min	Nom	Max	Units
V_{CC}	Supply Voltage		4.75	5	5.25	V
V_{IH}	HIGH Level Input Voltage		2			V
V_{IL}	LOW Level Input Voltage				0.8	V
I_{OH}	HIGH Level Output Current				−2.6	mA
I_{OL}	LOW Level Output Current				24	mA
t_W	Pulse Width	Clock HIGH	15			ns
	(Note 8)	Clock LOW	15			
t_{SU}	Data Setup Time (Note 7) (Note 8)		20↑			ns
t_H	Data Hold Time (Note 7) (Note 8)		1↑			ns
T_A	Free Air Operating Temperature		0		70	°C

Note 7: The symbol (↑) indicates the rising edge of the clock pulse is used for reference.

Note 8: T_A = 25°C and V_{CC} = 5V.

DM74LS374 Electrical Characteristics

over recommended operating free air temperature range (unless otherwise noted)

Symbol	Parameter	Conditions	Min	Typ (Note 9)	Max	Units
V_I	Input Clamp Voltage	V_{CC} = Min, I_I = −18 mA			−1.5	V
V_{OH}	HIGH Level Output Voltage	V_{CC} = Min, I_{OH} = Max, V_{IL} = Max, V_{IH} = Min	2.4	3.1		V
V_{OL}	LOW Level Output Voltage	V_{CC} = Min, I_{OL} = Max, V_{IL} = Max, V_{IH} = Min		0.35	0.5	V
		I_{OL} = 12 mA, V_{CC} = Min		0.25	0.4	
I_I	Input Current @ Max Input Voltage	V_{CC} = Max, V_I = 7V			0.1	mA
I_{IH}	HIGH Level Input Current	V_{CC} = Max, V_I = 2.7V			20	µA
I_{IL}	LOW Level Input Current	V_{CC} = Max, V_I = 0.4V			−0.4	mA
I_{OZH}	Off-State Output Current with HIGH Level Output Voltage Applied	V_{CC} = Max, V_O = 2.7V, V_{IH} = Min, V_{IL} = Max			20	µA
I_{OZL}	Off-State Output Current with LOW Level Output Voltage Applied	V_{CC} = Max, V_O = 0.4V, V_{IH} = Min, V_{IL} = Max			−20	µA
I_{OS}	Short Circuit Output Current	V_{CC} = Max (Note 10)	−50		−225	mA
I_{CC}	Supply Current	V_{CC} = Max, D_n = GND, OC = 4.5V		27	45	mA

Note 9: All typicals are at V_{CC} = 5V, T_A = 25°C.

Note 10: Not more than one output should be shorted at a time, and the duration should not exceed one second.

DM74LS374 Switching Characteristics

at V_{CC} = 5V and T_A = 25°C

Symbol	Parameter	R_L = 667Ω				Units
		C_L = 45 pF		C_L = 150 pF		
		Min	Max	Min	Max	
f_{MAX}	Maximum Clock Frequency	35		20		MHz
t_{PLH}	Propagation Delay Time LOW-to-HIGH Level Output		28		32	ns
t_{PHL}	Propagation Delay Time HIGH-to-LOW Level Output		28		38	ns
t_{PZH}	Output Enable Time to HIGH Level Output		28		44	ns
t_{PZL}	Output Enable Time to LOW Level Output		28		44	ns
t_{PHZ}	Output Disable Time from HIGH Level Output (Note 11)		20			ns
t_{PLZ}	Output Disable Time from LOW Level Output (Note 11)		25			ns

Note 11: C_L = 5 pF.

Absolute Maximum Ratings(Note 1)

Supply Voltage	7V
Input Voltage	7V
Storage Temperature Range	−65°C to +150°C
Operating Free Air Temperature Range	0°C to +70°C

Note 1: The "Absolute Maximum Ratings" are those values beyond which the safety of the device cannot be guaranteed. The device should not be operated at these limits. The parametric values defined in the Electrical Characteristics tables are not guaranteed at the absolute maximum ratings. The "Recommended Operating Conditions" table will define the conditions for actual device operation.

DM74LS373 Recommended Operating Conditions

Symbol	Parameter		Min	Nom	Max	Units
V_{CC}	Supply Voltage		4.75	5	5.25	V
V_{IH}	HIGH Level Input Voltage		2			V
V_{IL}	LOW Level Input Voltage				0.8	V
I_{OH}	HIGH Level Output Current				−2.6	mA
I_{OL}	LOW Level Output Current				24	mA
t_W	Pulse Width	Enable HIGH	15			ns
	(Note 3)	Enable LOW	15			
t_{SU}	Data Setup Time (Note 2) (Note 3)		5↓			ns
t_H	Data Hold Time (Note 2) (Note 3)		20↓			ns
T_A	Free Air Operating Temperature		0		70	°C

Note 2: The symbol (↓) indicates the falling edge of the clock pulse is used for reference.

Note 3: $T_A = 25°C$ and $V_{CC} = 5V$.

DM74LS373 Electrical Characteristics

over recommended operating free air temperature range (unless otherwise noted)

Symbol	Parameter	Conditions	Min	Typ (Note 4)	Max	Units
V_I	Input Clamp Voltage	V_{CC} = Min, I_I = −18 mA			−1.5	V
V_{OH}	HIGH Level Output Voltage	V_{CC} = Min, I_{OH} = Max, V_{IL} = Max, V_{IH} = Min	2.4	3.1		V
V_{OL}	LOW Level Output Voltage	V_{CC} = Min, I_{OL} = Max, V_{IL} = Max, V_{IH} = Min		0.35	0.5	V
		I_{OL} = 12 mA, V_{CC} = Min			0.4	
I_I	Input Current @ Max Input Voltage	V_{CC} = Max, V_I = 7V			0.1	mA
I_{IH}	HIGH Level Input Current	V_{CC} = Max, V_I = 2.7V			20	µA
I_{IL}	LOW Level Input Current	V_{CC} = Max, V_I = 0.4V			−0.4	mA
I_{OZH}	Off-State Output Current with HIGH Level Output Voltage Applied	V_{CC} = Max, V_O = 2.7V, V_{IH} = Min, V_{IL} = Max			20	µA
I_{OZL}	Off-State Output Current with LOW Level Output Voltage Applied	V_{CC} = Max, V_O = 0.4V, V_{IH} = Min, V_{IL} = Max			−20	µA
I_{OS}	Short Circuit Output Current	V_{CC} = Max (Note 5)	−50		−225	mA
I_{CC}	Supply Current	V_{CC} = Max, OC = 4.5V, D_n, Enable = GND		24	40	mA

Note 4: All typicals are at $V_{CC} = 5V$, $T_A = 25°C$.

Note 5: Not more than one output should be shorted at a time, and the duration should not exceed one second.

Anexo 4.4.

Hoja de especificaciones de la GAL22V10.

Lattice Semiconductor Corporation

GAL22V10

High Performance E²CMOS PLD Generic Array Logic™

Features

- **HIGH PERFORMANCE E²CMOS® TECHNOLOGY**
  - — 4 ns Maximum Propagation Delay
  - — Fmax = 250 MHz
  - — 3.5 ns Maximum from Clock Input to Data Output
  - — UltraMOS® Advanced CMOS Technology
- **ACTIVE PULL-UPS ON ALL PINS**
- **COMPATIBLE WITH STANDARD 22V10 DEVICES**
  - — Fully Function/Fuse-Map/Parametric Compatible with Bipolar and UVCMOS 22V10 Devices
- **50% to 75% REDUCTION IN POWER VERSUS BIPOLAR**
  - — 90mA Typical Icc on Low Power Device
  - — 45mA Typical Icc on Quarter Power Device
- **E² CELL TECHNOLOGY**
  - — Reconfigurable Logic
  - — Reprogrammable Cells
  - — 100% Tested/100% Yields
  - — High Speed Electrical Erasure (<100ms)
  - — 20 Year Data Retention
- **TEN OUTPUT LOGIC MACROCELLS**
  - — Maximum Flexibility for Complex Logic Designs
- **PRELOAD AND POWER-ON RESET OF REGISTERS**
  - — 100% Functional Testability
- **APPLICATIONS INCLUDE:**
  - — DMA Control
  - — State Machine Control
  - — High Speed Graphics Processing
  - — Standard Logic Speed Upgrade
- **ELECTRONIC SIGNATURE FOR IDENTIFICATION**

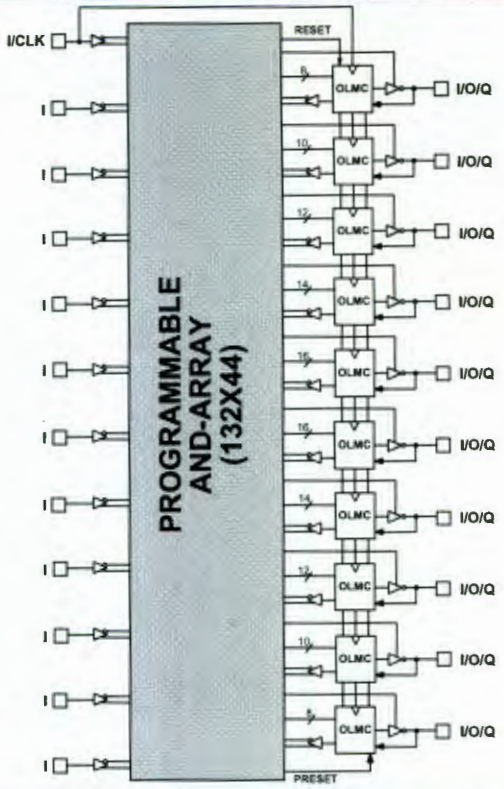
Description

The GAL22V10, at 4ns maximum propagation delay time, combines a high performance CMOS process with Electrically Erasable (E^2) floating gate technology to provide the highest performance available of any 22V10 device on the market. CMOS circuitry allows the GAL22V10 to consume much less power when compared to bipolar 22V10 devices. E^2 technology offers high speed (<100ms) erase times, providing the ability to reprogram or reconfigure the device quickly and efficiently.

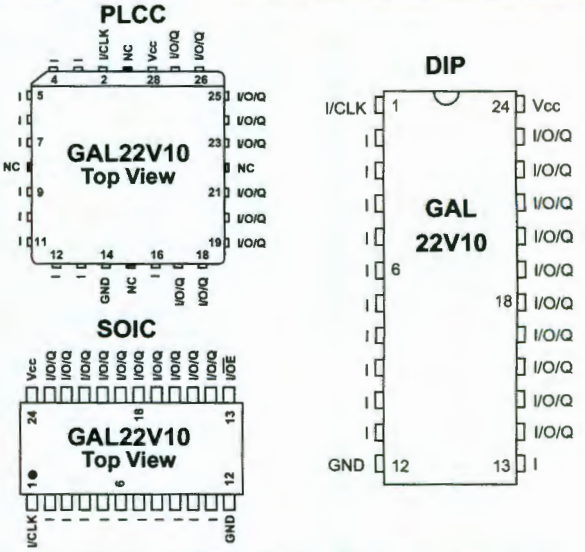
The generic architecture provides maximum design flexibility by allowing the Output Logic Macrocell (OLMC) to be configured by the user. The GAL22V10 is fully function/fuse map/parametric compatible with standard bipolar and CMOS 22V10 devices.

Unique test circuitry and reprogrammable cells allow complete AC, DC, and functional testing during manufacture. As a result, Lattice Semiconductor delivers 100% field programmability and functionality of all GAL products. In addition, 100 erase/write cycles and data retention in excess of 20 years are specified.

Functional Block Diagram



Pin Configuration



LATTICE SEMICONDUCTOR CORP., 5555 Northeast Moore Ct., Hillsboro, Oregon 97124, U.S.A.
Tel. (503) 268-8000; 1-800-LATTICE; FAX (503) 268-8556; http://www.latticesemi.com

May 2001

Lattice Semiconductor Corporation

Specifications *GAL22V10*

GAL22V10 Ordering Information

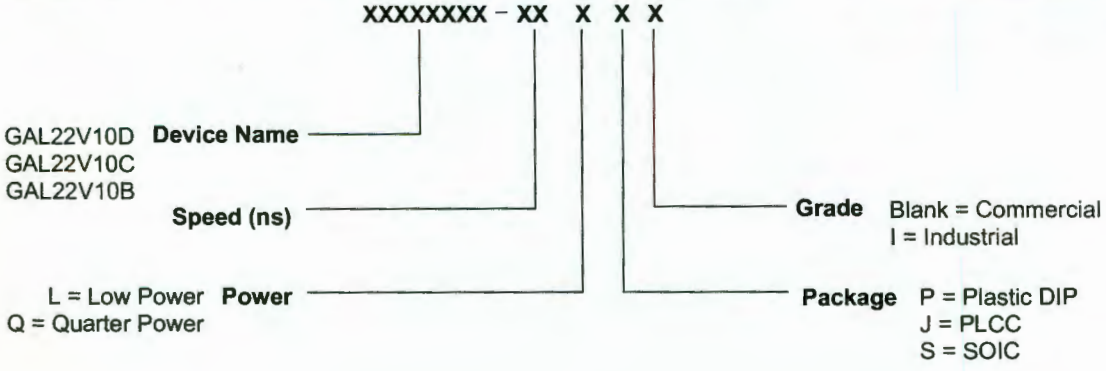
Commercial Grade Specifications

Tpd (ns)	Tsu (ns)	Tco (ns)	Icc (mA)	Ordering #	Package
4	2.5	3.5	140	GAL22V10D-4LJ	28-Lead PLCC
5	3	4	140	GAL22V10D-5LJ	28-Lead PLCC
			150	GAL22V10C-5LJ	28-Lead PLCC
7.5	4.5	4.5	140	GAL22V10D-7LP	24-Pin Plastic DIP
	5	4.5	140	GAL22V10C-7LP	24-Pin Plastic DIP
	4.5	4.5	140	GAL22V10D-7LJ or GAL22V10C-7LJ	28-Lead PLCC
	6.5	5	140	GAL22V10B-7LP	24-Pin Plastic DIP
			140	GAL22V10B-7LJ	28-Lead PLCC
10	7	7	55	GAL22V10D-10QP	24-Pin Plastic DIP
			55	GAL22V10D-10QJ	28-Lead PLCC
			130	GAL22V10D-10LP, GAL22V10C-10LP or GAL22V10B-10LP	24-Pin Plastic DIP
			130	GAL22V10D-10LJ, GAL22V10C-10LJ or GAL22V10B-10LJ	28-Lead PLCC
			130	GAL22V10D-10LS	24-Pin SOIC
15	10	8	55	GAL22V10D-15QP or GAL22V10B-15QP	24-Pin Plastic DIP
			55	GAL22V10D-15QJ or GAL22V10B-15QJ	28-Lead PLCC
			130	GAL22V10D-15LP or GAL22V10B-15LP	24-Pin Plastic DIP
			130	GAL22V10D-15LJ or GAL22V10B-15LJ	28-Lead PLCC
			130	GAL22V10D-15LS	24-Pin SOIC
25	15	15	55	GAL22V10D-25QP or GAL22V10B-25QP	24-Pin Plastic DIP
			55	GAL22V10D-25QJ or GAL22V10B-25QJ	28-Lead PLCC
			90	GAL22V10D-25LP or GAL22V10B-25LP	24-Pin Plastic Dip
			90	GAL22V10D-25LJ or GAL22V10B-25LJ	28-Lead PLCC
			90	GAL22V10D-25LS	24-Pin SOIC

Industrial Grade Specifications

Tpd (ns)	Tsu (ns)	Tco (ns)	Icc (mA)	Ordering #	Package
7.5	5	4.5	160	GAL22V10D-7LPI or GAL22V10C-7LPI	24-Pin Plastic DIP
	4.5	4.5	160	GAL22V10D-7LJI or GAL22V10C-7LJI	28-Lead PLCC
10	7	7	160	GAL22V10D-10LPI or GAL22V10C-10LPI	24-Pin Plastic DIP
			160	GAL22V10D-10LJI or GAL22V10C-10LJI	28-Lead PLCC
15	10	8	150	GAL22V10D-15LPI or GAL22V10B-15LPI	24-Pin Plastic DIP
			150	GAL22V10D-15LJI or GAL22V10B-15LJI	28-Lead PLCC
20	14	10	150	GAL22V10D-20LPI or GAL22V10B-20LPI	24-Pin Plastic DIP
			150	GAL22V10D-20LJI or GAL22V10B-20LJI	28-Lead PLCC
25	15	15	150	GAL22V10D-25LPI or GAL22V10B-25LPI	24-Pin Plastic DIP
			150	GAL22V10D-25LJI or GAL22V10B-25LJI	28-Lead PLCC

Part Number Description



Lattice Semiconductor Corporation

Specifications *GAL22V10*

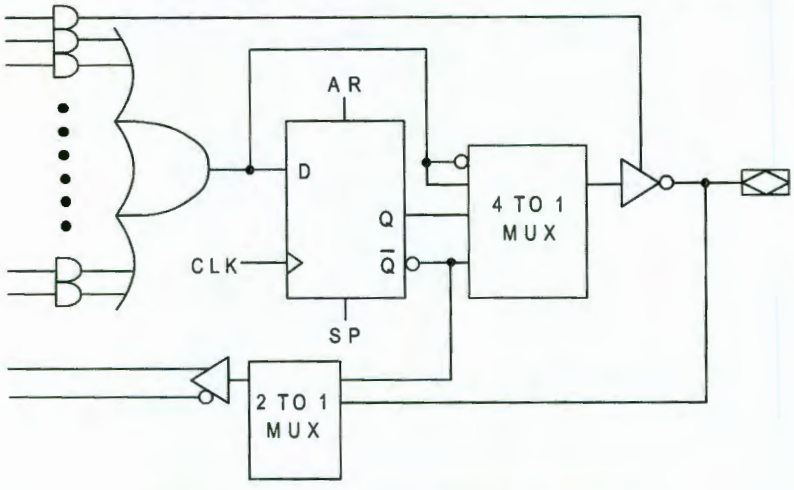
Output Logic Macrocell (OLMC)

The GAL22V10 has a variable number of product terms per OLMC. Of the ten available OLMCs, two OLMCs have access to eight product terms (pins 14 and 23, DIP pinout), two have ten product terms (pins 15 and 22), two have twelve product terms (pins 16 and 21), two have fourteen product terms (pins 17 and 20), and two OLMCs have sixteen product terms (pins 18 and 19). In addition to the product terms available for logic, each OLMC has an additional product-term dedicated to output enable control.

The output polarity of each OLMC can he individually programmed to be true or inverting, in either combinatorial or registered mode. This allows each output to be individually configured as either active high or active low.

The GAL22V10 has a product term for Asynchronous Reset (AR) and a product term for Synchronous Preset (SP). These two product terms are common to all registered OLMCs. The Asynchronous Reset sets all registers to zero any time this dedicated product term is asserted. The Synchronous Preset sets all registers to a logic one on the rising edge of the next clock pulse after this product term is asserted.

NOTE: The AR and SP product terms will force the Q output of the flip-flop into the same state regardless of the polarity of the output. Therefore, a reset operation, which sets the register output to a zero, may result in either a high or low at the output pin, depending on the pin polarity chosen.



GAL22V10 OUTPUT LOGIC MACROCELL (OLMC)

Output Logic Macrocell Configurations

Each of the Macrocells of the GAL22V10 has two primary functional modes: registered, and combinatorial I/O. The modes and the output polarity are set by two bits (SO and S1), which are normally controlled by the logic compiler. Each of these two primary modes, and the bit settings required to enable them, are described below and on the following page.

REGISTERED

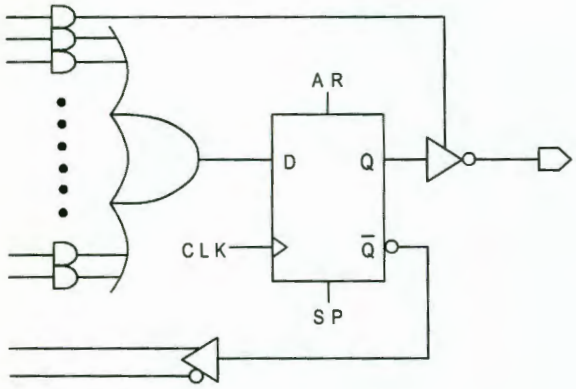
In registered mode the output pin associated with an individual OLMC is driven by the Q output of that OLMC's D-type flip-flop. Logic polarity of the output signal at the pin may be selected by specifying that the output buffer drive either true (active high) or inverted (active low). Output tri-state control is available as an individual product-term for each OLMC, and can therefore be defined by a logic equation. The D flip-flop's /Q output is fed back into the AND array, with both the true and complement of the feedback available as inputs to the AND array.

NOTE: In registered mode, the feedback is from the /Q output of the register, and not from the pin; therefore, a pin defined as registered is an output only, and cannot be used for dynamic I/O, as can the combinatorial pins.

COMBINATORIAL I/O

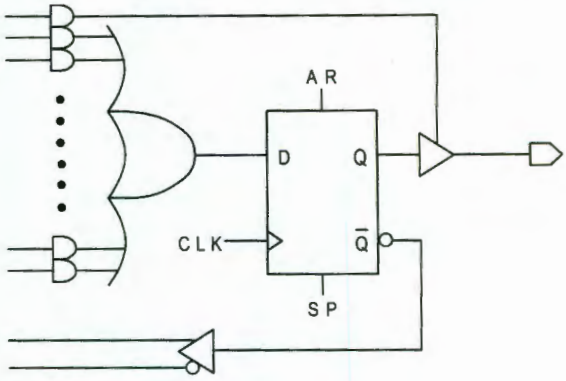
In combinatorial mode the pin associated with an individual OLMC is driven by the output of the sum term gate. Logic polarity of the output signal at the pin may be selected by specifying that the output buffer drive either true (active high) or inverted (active low). Output tri-state control is available as an individual product-term for each output, and may be individually set by the compiler as either "on" (dedicated output), "off" (dedicated input), or "product-term driven" (dynamic I/O). Feedback into the AND array is from the pin side of the output enable buffer. Both polarities (true and inverted) of the pin are fed back into the AND array.

Registered Mode



ACTIVE LOW

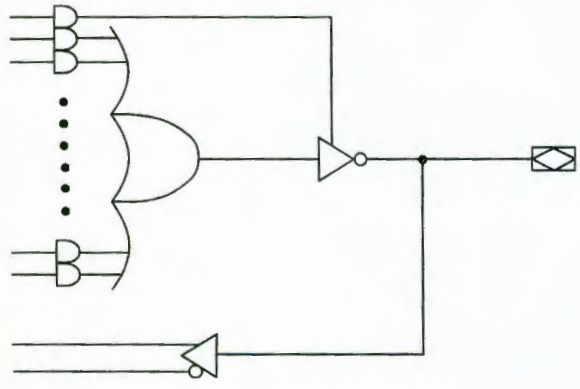
$S_0 = 0$
$S_1 = 0$



ACTIVE HIGH

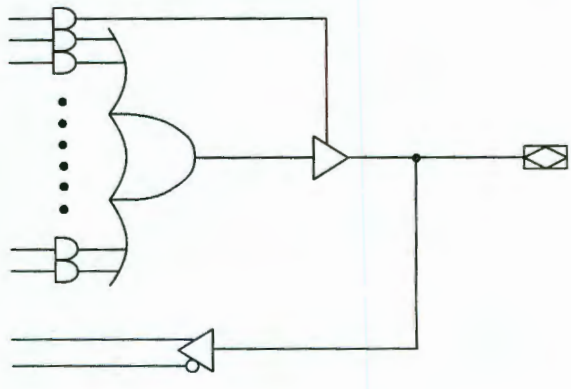
$S_0 = 1$
$S_1 = 0$

Combinatorial Mode



ACTIVE LOW

$S_0 = 0$
$S_1 = 1$



ACTIVE HIGH

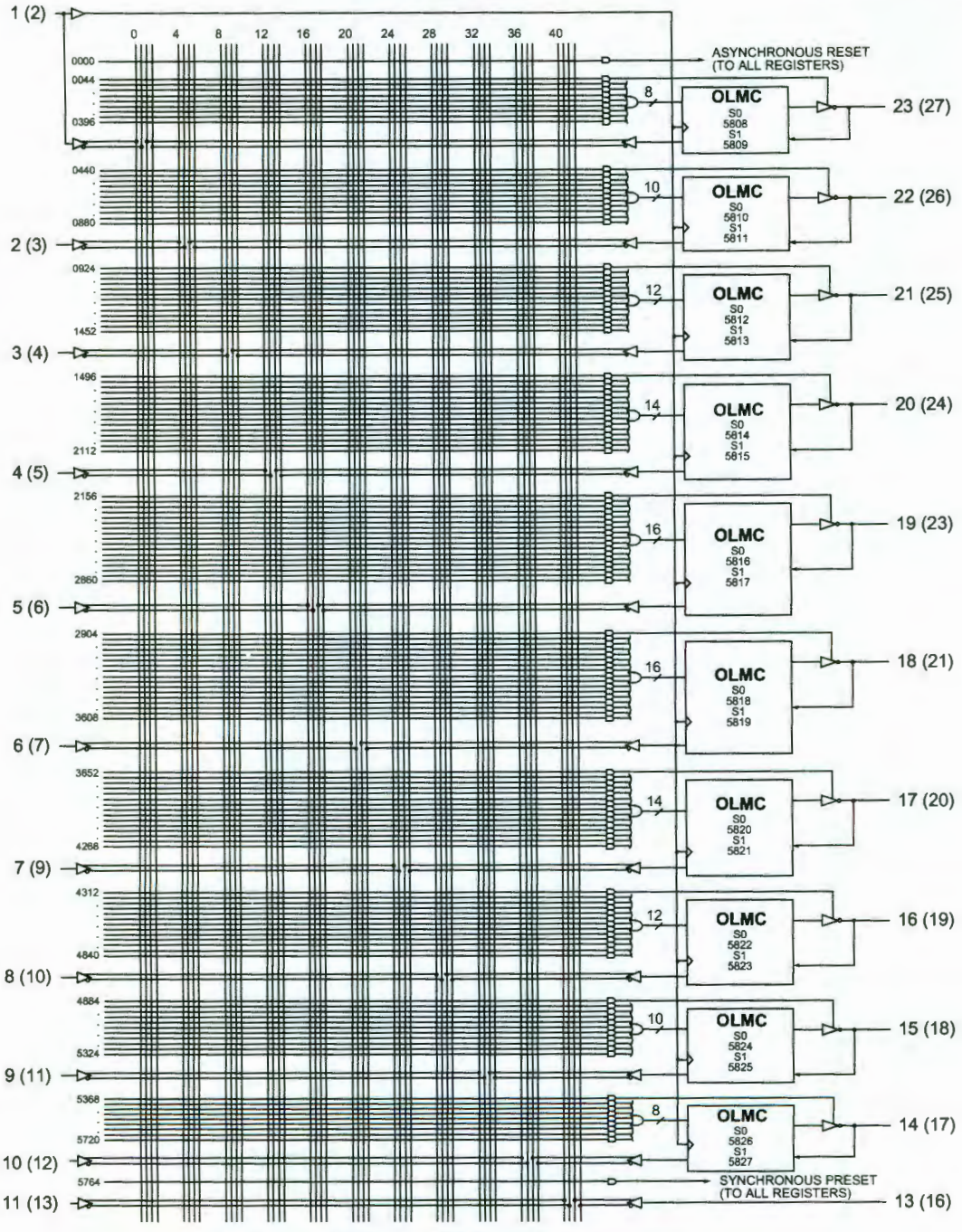
$S_0 = 1$
$S_1 = 1$

Lattice Semiconductor Corporation

Specifications *GAL22V10*

GAL22V10 Logic Diagram / JEDEC Fuse Map

DIP (PLCC) Package Pinouts



5828, 5829 ... Electronic Signature ... 5890, 5891							
Byte 7	Byte 6	Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0

MSB LSB

Anexo 4.5.

Decodificación de señales en lenguaje VHDL para generación de secuencias para motores a pasos.

```
library IEEE;
use IEEE.std_logic_1164.all;

entity generador is
        port (
                clk: in STD_LOGIC;
                clr: in STD_LOGIC;
                direccion: in STD_LOGIC;
                paso_comp_medio: in STD_LOGIC;
                y: out STD_LOGIC_vector(3 downto 0)
                );
end generador;

architecture generador of generador is
signal estado_presente,estado_proximo:std_logic_vector(3 downto 0);
begin

        Combinacional: process(estado_presente,direccion,paso_comp_medio)
        begin
                if (direccion='1'and paso_comp_medio='1')then
                        case estado_presente is
                                when "0000"=> estado_proximo <="0001";
                                when "0001"=> estado_proximo <="0011";
                                when "0011"=> estado_proximo <="0010";
                                when "0010"=> estado_proximo <="0110";
                                when "0110"=> estado_proximo <="0100";
                                when "0100"=> estado_proximo <="1100";
                                when "1100"=> estado_proximo <="1000";
                                when "1000"=> estado_proximo <="1001";
                                when "1001"=> estado_proximo <="0001";
                                when others=> estado_proximo <="0000";
                        end case;
                elsif (direccion='0'and paso_comp_medio='1')then
                        case estado_presente is
                                when "0000"=> estado_proximo <="0001";
                                when "0001"=> estado_proximo <="1001";
                                when "1001"=> estado_proximo <="1000";
                                when "1000"=> estado_proximo <="1100";
                                when "1100"=> estado_proximo <="0100";
                                when "0100"=> estado_proximo <="0110";
                                when "0110"=> estado_proximo <="0010";
                                when "0010"=> estado_proximo <="0011";
                                when "0011"=> estado_proximo <="0001";
                                when others=> estado_proximo <="0000";
                        end case;

                elsif (direccion='1'and paso_comp_medio='0')then
                        case estado_presente is
                                when "0000"=> estado_proximo <="0001";
                                when "0001"=> estado_proximo <="0010";
                                when "0010"=> estado_proximo <="0100";
                                when "0100"=> estado_proximo <="1000";
                                when "1000"=> estado_proximo <="0001";

                                when "0011"=> estado_proximo <="0010";
                                when "0110"=> estado_proximo <="0100";
                                when "1100"=> estado_proximo <="1000";
                                when "1001"=> estado_proximo <="0001";
```

```
                        when others=> estado_proximo <="0000";
                end case;
        else
                case estado_presente is
                        when "0000"=> estado_proximo <="0001";
                        when "0001"=> estado_proximo <="1000";
                        when "1000"=> estado_proximo <="0100";
                        when "0100"=> estado_proximo <="0010";
                        when "0010"=> estado_proximo <="0001";

                        when "1001"=> estado_proximo <="1000";
                        when "1100"=> estado_proximo <="0100";
                        when "0110"=> estado_proximo <="0010";
                        when "0011"=> estado_proximo <="0001";

                        when others=> estado_proximo <="0000";
                end case;
    end if;
    y<=estado_presente;
    end process Combinacional;


    Secuencial:process(clk,clr)
    begin
            if(clr='0')then
            estado_presente<="0000";
            elsif(clk'event and clk='1')then
            estado_presente<=estado_proximo;
            end if;
            end process Secuencial;
end generador;
```

Anexo 4.6.

Archivo de reporte para la conexión a circuito de bobinas usando una GAL22V10.

```
     | | | | | | | |
   _________________
  -|                 |-
  -|                 |-
  -|                 |-
  -|     CYPRESS     |-
  -|                 |-
  -|                 |-   Warp VHDL Synthesis Compiler: Version 4 IR x66
  -|                 |-   Copyright (C) 1991, 1992, 1993,
   |_________________|                  1994, 1995, 1996 Cypress Semiconductor
     | | | | | | | |

========================================================================
Compiling:  genera~1.vhd
Options:    -q -e10 -w10 -o2 -yga -fP -v10 -yb -yp -dc22v10 -pPALC22V10-25DMB
genera~1.vhd
========================================================================

C:\WARP\BIN\VHDLFE.EXE V4 IR x66:  VHDL parser
Sat Nov 08 22:05:39 2003

Library 'work' => directory 'lc22v10'
Library 'ieee' => directory 'C:\warp\lib\ieee\work'
Using 'C:\warp\lib\ieee\work\stdlogic.vif'.

C:\WARP\BIN\VHDLFE.EXE:  No errors.


C:\WARP\BIN\TOVIF.EXE V4 IR x66:  High-level synthesis
Sat Nov 08 22:05:40 2003


C:\WARP\BIN\TOVIF.EXE:  No errors.


C:\WARP\BIN\TOPLD.EXE V4 IR x66:  Synthesis and optimization
Sat Nov 08 22:05:41 2003


------------------------------------------------------------
Detecting unused logic.
------------------------------------------------------------



------------------------------------------------------------
Alias Detection
------------------------------------------------------------

------------------------------------------------------------
Aliased 0 equations, 11 wires.
------------------------------------------------------------

------------------------------------------------------------
Circuit simplification
------------------------------------------------------------

------------------------------------------------------------
Circuit simplification results:
```

```
      Expanded 0 signals.
      Turned 0 signals into soft nodes.
      Maximum expansion cost was set at 10.
------------------------------------------------------------
Created 25 PLD nodes.

C:\WARP\BIN\TOPLD.EXE:  No errors.

------------------------------------------------------------------------------
PLD Optimizer Software:       DSGNOPT.EXE    17/JUL/96    [v3.22 ] 4 IR x66

DESIGN HEADER INFORMATION  (22:05:42)

Input File(s): genera~1.pla
Device       : C22V10
Package      : PALC22V10-25DMB
ReportFile   : genera~1.rpt

Program Controls:
                None.

Signal Requests:
    GROUP USEPOL ALL

Completed Successfully
------------------------------------------------------------------------------
PLD Optimizer Software:       DSGNOPT.EXE    17/JUL/96    [v3.22 ] 4 IR x66

OPTIMIZATION OPTIONS       (22:05:42)

Messages:
  Information: Process virtual 'estado_proximo_0' ... expanded.
  Information: Process virtual 'estado_proximo_1' ... expanded.
  Information: Process virtual 'estado_proximo_2' ... expanded.
  Information: Process virtual 'estado_proximo_3' ... expanded.
  Information: Optimizing logic using best output polarity for signals:
        y_0.D y_1.D y_2.D y_3.D

  Information: Selected logic optimization OFF for signals:
        y_0.AR y_0.C y_1.AR y_1.C y_2.AR y_2.C y_3.AR y_3.C


Summary:
                Error Count = 0      Warning Count = 0

Completed Successfully
------------------------------------------------------------------------------
PLD Optimizer Software:       MINOPT.EXE     17/JUL/96    [v3.22 ] 4 IR x66

LOGIC MINIMIZATION         (22:05:42)

Messages:


Summary:
                Error Count = 0      Warning Count = 0

Completed Successfully
------------------------------------------------------------------------------
PLD Optimizer Software:       DSGNOPT.EXE    17/JUL/96    [v3.22 ] 4 IR x66

OPTIMIZATION OPTIONS       (22:05:43)

Messages:
  Information: Optimizing Banked Preset/Reset requirements.


Summary:
                Error Count = 0      Warning Count = 0
```

```
Completed Successfully
------------------------------------------------------------------------------
PLD Compiler Software:         PLA2JED.EXE     17/JUL/96     [v3.22 ] 4 IR x66

DESIGN EQUATIONS            (22:05:43)


    y_3.D =
          paso_comp_medio * y_3.Q * /y_2.Q * /y_1.Q * /y_0.Q
        + direccion * y_2.Q * /y_1.Q * /y_0.Q
        + /direccion * /y_2.Q * /y_1.Q * y_0.Q

    y_3.AR =
          /clr

    y_3.SP =
          GND

    y_3.C =
          clk

    y_2.D =
          paso_comp_medio * /y_3.Q * y_2.Q * /y_1.Q * /y_0.Q
        + /direccion * y_3.Q * /y_1.Q * /y_0.Q
        + direccion * /y_3.Q * y_1.Q * /y_0.Q

    y_2.AR =
          /clr

    y_2.SP =
          GND

    y_2.C =
          clk

    y_1.D =
          paso_comp_medio * /y_3.Q * /y_2.Q * y_1.Q * /y_0.Q
        + /direccion * /y_3.Q * y_2.Q * /y_0.Q
        + direccion * /y_3.Q * /y_2.Q * y_0.Q

    y_1.AR =
          /clr

    y_1.SP =
          GND

    y_1.C =
          clk

    y_0.D =
          paso_comp_medio * /y_3.Q * /y_2.Q * /y_1.Q
        + /y_3.Q * /y_2.Q * /y_1.Q * /y_0.Q
        + direccion * y_3.Q * /y_2.Q * /y_1.Q
        + /direccion * /y_3.Q * /y_2.Q * y_1.Q

    y_0.AR =
          /clr

    y_0.SP =
          GND

    y_0.C =
          clk


Completed Successfully
------------------------------------------------------------------------------
PLD Compiler Software:         PLA2JED.EXE     17/JUL/96     [v3.22 ] 4 IR x66
```

```
DESIGN RULE CHECK          (22:05:43)

Messages:
                None.


Summary:
                Error Count = 0      Warning Count = 0

Completed Successfully
-------------------------------------------------------------------------------
PLD Compiler Software:        PLA2JED.EXE    17/JUL/96    [v3.22 ] 4 IR x66

DESIGN SIGNAL PLACEMENT    (22:05:43)

Messages:
  Information: Checking for duplicate NODE logic.
                None.


                                C22V10
                          ____________________
           clk =| 1|                          |24|* not used
           clr =| 2|                          |23|= y_1
 paso_comp_me.. =| 3|                          |22|= y_3
     direccion =| 4|                          |21|* not used
      not used *| 5|                          |20|* not used
      not used *| 6|                          |19|* not used
      not used *| 7|                          |18|* not used
      not used *| 8|                          |17|* not used
      not used *| 9|                          |16|* not used
      not used *|10|                          |15|= y_2
      not used *|11|                          |14|= y_0
      not used *|12|                          |13|* not used
                ----------------------------

Summary:
                Error Count = 0      Warning Count = 0

Completed Successfully
-------------------------------------------------------------------------------
PLD Compiler Software:        PLA2JED.EXE    17/JUL/96    [v3.22 ] 4 IR x66

RESOURCE ALLOCATION        (22:05:43)

  Information: Macrocell Utilization.

                    Description        Used      Max
                ---------------------------------------
                | Dedicated Inputs   |    3  |   11  |
                | Clock/Inputs       |    1  |    1  |
                | I/O Macrocells     |    4  |   10  |
                ---------------------------------------
                                       8  /   22   = 36  %

  Information: Output Logic Product Term Utilization.

                 Node#  Output Signal Name  Used   Max
                ------------------------------------------
                | 14  |  y_0               |  4  |   8  |
                | 15  |  y_2               |  3  |  10  |
                | 16  |  Unused            |  0  |  12  |
                | 17  |  Unused            |  0  |  14  |
                | 18  |  Unused            |  0  |  16  |
                | 19  |  Unused            |  0  |  16  |
                | 20  |  Unused            |  0  |  14  |
                | 21  |  Unused            |  0  |  12  |
                | 22  |  y_3               |  3  |  10  |
```

```
| 23  |  y_1              |   3  |   8  |
| 25  |  Unused           |   0  |   1  |
                                  13  / 121   = 10  %

Completed Successfully
--------------------------------------------------------------------------------
PLD Compiler Software:          PLA2JED.EXE     17/JUL/96     [v3.22 ] 4 IR x66

JEDEC ASSEMBLE                (22:05:43)

Messages:
  Information: Output file 'genera~1.jed' created.


Summary:
                Error Count = 0       Warning Count = 0

Completed Successfully at 22:05:43
```

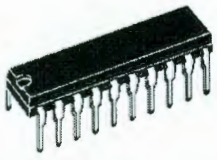
Anexo 4.7.

Hoja de especificaciones del L297.

L297

STEPPER MOTOR CONTROLLERS

- NORMAL/WAVE DRIVE
- HALF/FULL STEP MODES
- CLOCKWISE/ANTICLOCKWISE DIRECTION
- SWITCHMODE LOAD CURRENT REGULATION
- PROGRAMMABLE LOAD CURRENT
- FEW EXTERNAL COMPONENTS
- RESET INPUT & HOME OUTPUT
- ENABLE INPUT



DIP20

SO20

ORDERING NUMBERS : L297/1 (DIP20)
L297D (SO20)

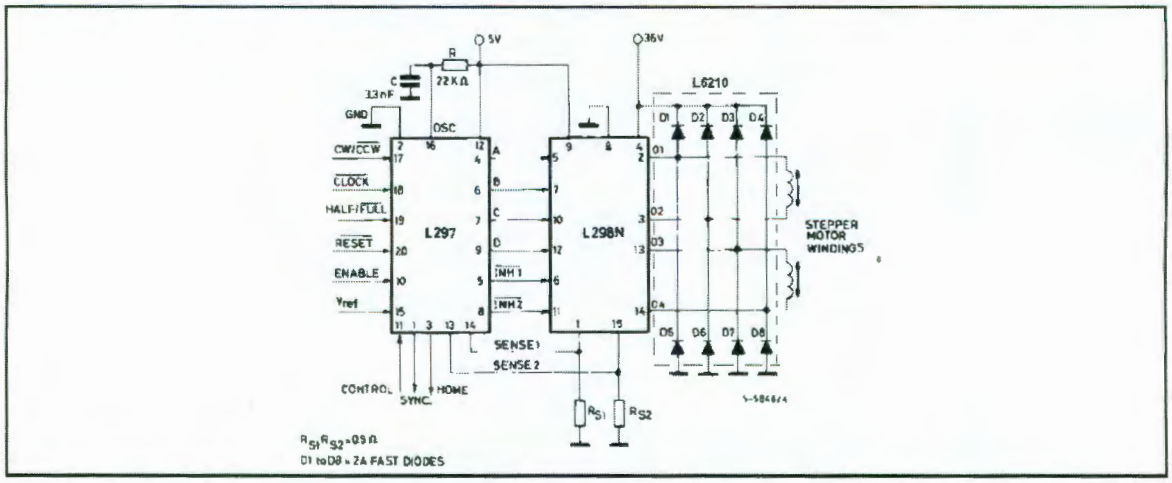
DESCRIPTION

The L297 Stepper Motor Controller IC generates four phase drive signals for two phase bipolar and four phase unipolar step motors in microcomputer-controlled applications. The motor can be driven in half step, normal and wawe drive modes and on-chip PWM chopper circuits permit switch-mode control of the current in the windings. A feature of this device is that it requires only clock, direction and mode input signals. Since the phase are generated internally the burden on the microprocessor, and the programmer, is greatly reduced. Mounted in DIP20 and SO20 packages, the L297 can be used with monolithic bridge drives such as the L298N or L293E, or with discrete transistors and darlingtons.

ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V_s	Supply voltage	10	V
V_i	Input signals	7	V
P_{tot}	Total power dissipation (T_{amb} = 70°C)	1	W
T_{slg}, T_j	Storage and junction temperature	-40 to + 150	°C

TWO PHASE BIPOLAR STEPPER MOTOR CONTROL CIRCUIT



PIN FUNCTIONS - L297/1 - L297D

N°	NAME	FUNCTION
1	SYNC	Output of the on-chip chopper oscillator. The SYNC connections The SYNC connections of all L297s to be synchronized are connected together and the oscillator components are omitted on all but one. If an external clock source is used it is injected at this terminal.
2	GND	Ground connection.
3	HOME	Open collector output that indicates when the L297 is in its initial state (ABCD = 0101). The transistor is open when this signal is active.
4	A	Motor phase A drive signal for power stage.
5	$\overline{INH1}$	Active low inhibit control for driver stage of A and B phases. When a bipolar bridge is used this signal can be used to ensure fast decay of load current when a winding is de-energized. Also used by chopper to regulate load current if CONTROL input is low.
6	B	Motor phase B drive signal for power stage.
7	C	Motor phase C drive signal for power stage.
8	$\overline{INH2}$	Active low inhibit control for drive stages of C and D phases. Same functions as INH1.
9	D	Motor phase D drive signal for power stage.
10	ENABLE	Chip enable input. When low (inactive) INH1, INH2, A, B, C and D are brought low.
11	CONTROL	Control input that defines action of chopper. When low chopper acts on INH1 and INH2; when high chopper acts on phase lines ABCD.
12	V_S	5V supply input.
13	$SENS_2$	Input for load current sense voltage from power stages of phases C and D.
14	$SENS_1$	Input for load current sense voltage from power stages of phases A and B.
15	V_{ref}	Reference voltage for chopper circuit. A voltage applied to this pin determines the peak load current.
16	OSC	An RC network (R to V_{CC}, C to ground) connected to this terminal determines the chopper rate. This terminal is connected to ground on all but one device in synchronized multi - L297 configurations. f $\cong$ 1/0.69 RC
17	CW/$\overline{CCW}$	Clockwise/counterclockwise direction control input. Physical direction of motor rotation also depends on connection of windings. Synchronized internally therefore direction can be changed at any time.
18	$\overline{CLOCK}$	Step clock. An active low pulse on this input advances the motor one increment. The step occurs on the rising edge of this signal.

PIN FUNCTIONS - L297/1 - L297D (continued)

N°	NAME	FUNCTION
19	HALF/$\overline{\text{FULL}}$	Half/full step select input. When high selects half step operation, when low selects full step operation. One-phase-on full step mode is obtained by selecting FULL when the L297's translator is at an even-numbered state. Two-phase-on full step mode is set by selecting FULL when the translator is at an odd numbered position. (The home position is designate state 1).
20	$\overline{\text{RESET}}$	Reset input. An active low pulse on this input restores the translator to the home position (state 1, ABCD = 0101).

THERMAL DATA

Symbol	Parameter		DIP20	SO20	Unit
$R_{th\text{-}j\text{-}amb}$	Thermal resistance junction-ambient	max	80	100	°C/W

CIRCUIT OPERATION

The L297 is intended for use with a dual bridge driver, quad darlington array or discrete power devices in step motor driving applications. It receives step clock, direction and mode signals from the systems controller (usually a microcomputer chip) and generates control signals for the power stage.

The principal functions are a translator, which generates the motor phase sequences, and a dual PWM chopper circuit which regulates the current in the motor windings. The translator generates three different sequences, selected by the HALF/FULL input. These are normal (two phases energised), wave drive (one phase energised) and half-step (alternately one phase energised/two phases energised). Two inhibit signals are also generated by the L297 in half step and wave drive modes. These signals, which connect directly to the L298's enable inputs, are intended to speed current decay when a winding is de-energised. When the L297 is used to drive a unipolar motor the chopper acts on these lines.

An input called CONTROL determines whether the chopper will act on the phase lines ABCD or the inhibit lines INH1 and INH2. When the phase lines are chopped the non-active phase line of each pair (AB or CD) is activated (rather than interrupting the line then active). In L297 + L298 configurations this technique reduces dissipation in the load current sense resistors.

A common on-chip oscillator drives the dual chopper. It supplies pulses at the chopper rate which set the two flip-flops FF1 and FF2. When the current in a winding reaches the programmed peak value the voltage across the sense resistor (connected to one of the sense inputs $SENS_1$ or $SENS_2$) equals V_{ref} and the corresponding comparator resets its flip flop, interrupting the drive current until the next oscillator pulse arrives. The peak current for both windings is programmed by a voltage divider on the V_{ref} input.

Ground noise problems in multiple configurations can be avoided by synchronising the chopper oscillators. This is done by connecting all the SYNC pins together, mounting the oscillator RC network on one device only and grounding the OSC pin on all other devices.

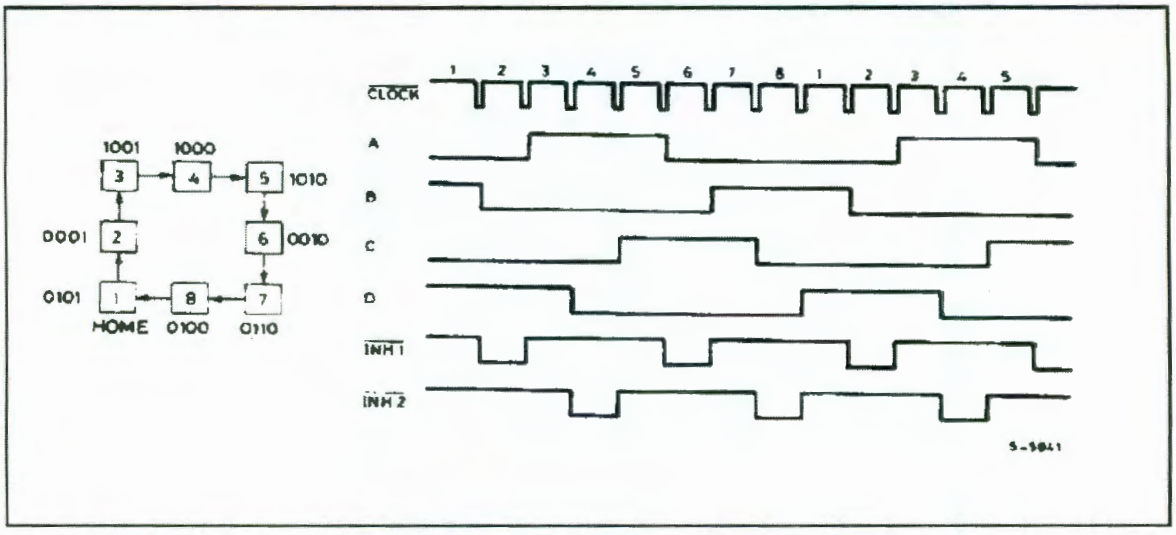
MOTOR DRIVING PHASE SEQUENCES

The L297's translator generates phase sequences for normal drive, wave drive and half step modes. The state sequences and output waveforms for these three modes are shown below. In all cases the translator advances on the low to high transistion of $\overline{CLOCK}$.

Clockwise rotation is indicate; for anticlockwise rotation the sequences are simply reversed $\overline{RESET}$ restores the translator to state 1, where ABCD = 0101.

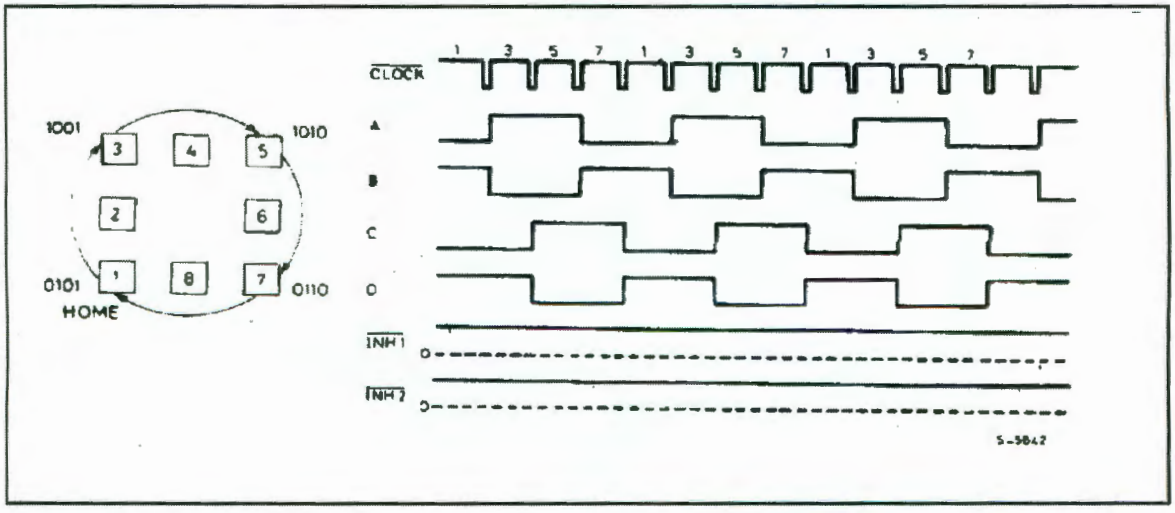
HALF STEP MODE

Half step mode is selected by a high level on the HALF/$\overline{FULL}$ input.

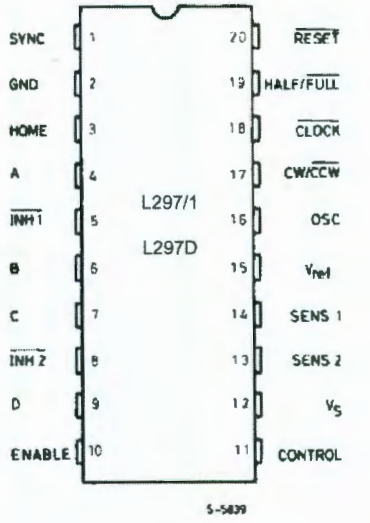


NORMAL DRIVE MODE

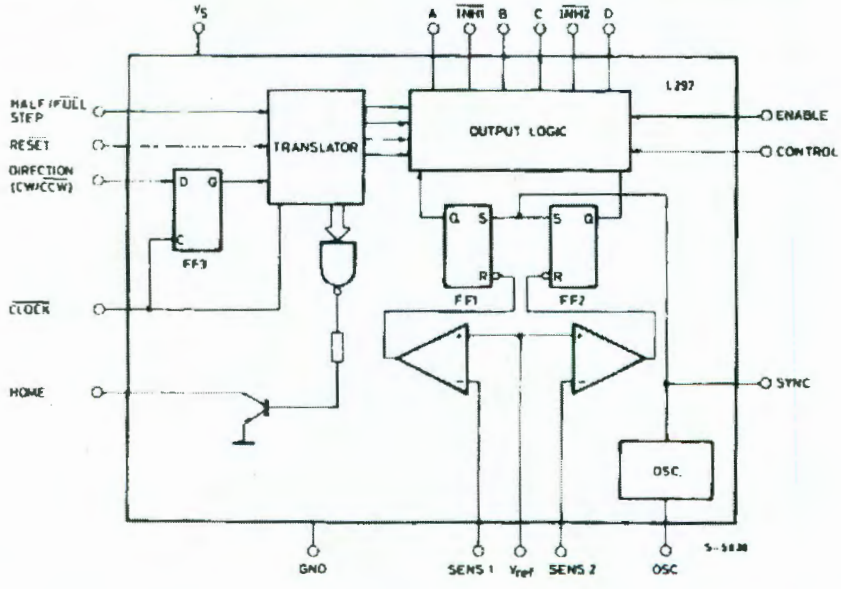
Normal drive mode (also called "two-phase-on" drive) is selected by a low level on the HALF/$\overline{FULL}$ input when the translator is at an odd numbered state (1, 3, 5 or 7). In this mode the $\overline{INH1}$ and $\overline{INH2}$ outputs remain high throughout.



PIN CONNECTION (Top view)



BLOCK DIAGRAM (L297/1 - L297D)



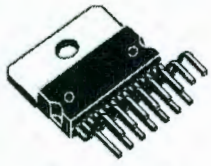
Anexo 4.8.

Hoja de especificaciones del L298.

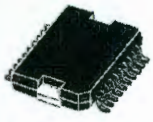
L298

DUAL FULL-BRIDGE DRIVER

- OPERATING SUPPLY VOLTAGE UP TO 46 V
- TOTAL DC CURRENT UP TO 4 A
- LOW SATURATION VOLTAGE
- OVERTEMPERATURE PROTECTION
- LOGICAL "0" INPUT VOLTAGE UP TO 1.5 V (HIGH NOISE IMMUNITY)



Multiwatt15



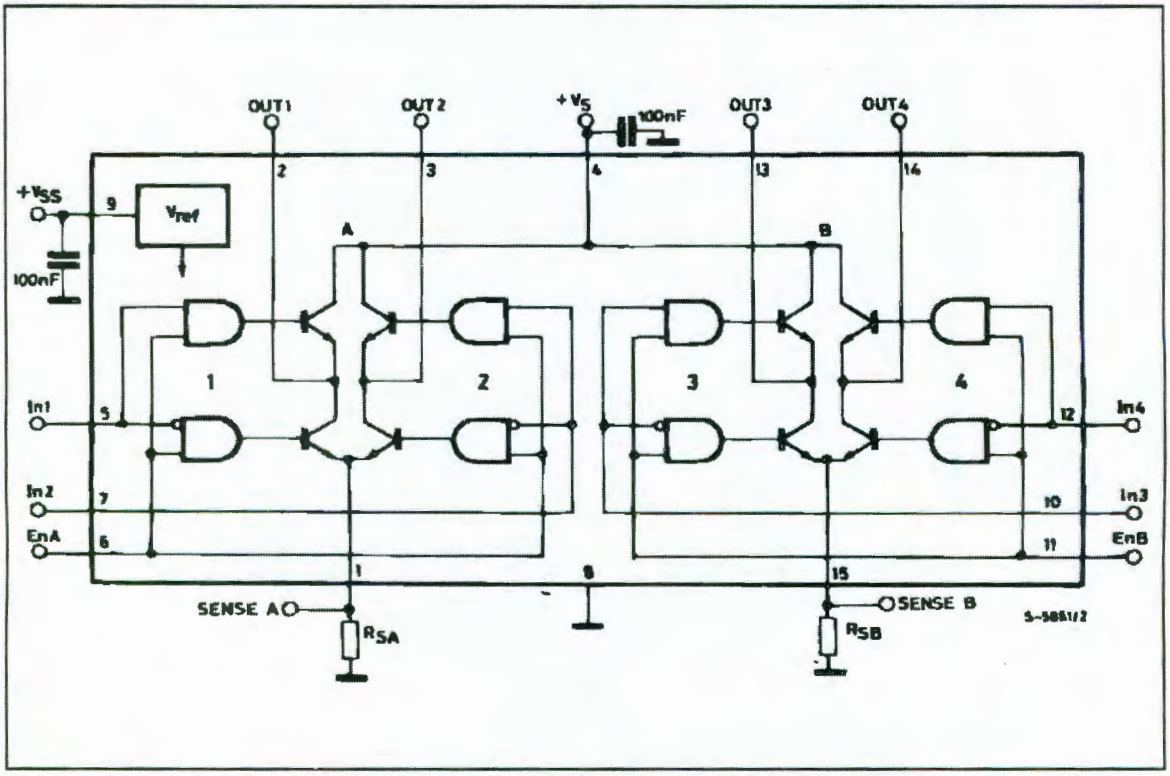
PowerSO20

ORDERING NUMBERS : L298N (Multiwatt Vert.)
L298HN (Multiwatt Horiz.)
L298P (PowerSO20)

DESCRIPTION

The L298 is an integrated monolithic circuit in a 15-lead Multiwatt and PowerSO20 packages. It is a high voltage, high current dual full-bridge driver designed to accept standard TTL logic levels and drive inductive loads such as relays, solenoids, DC and stepping motors. Two enable inputs are provided to enable or disable the device independently of the input signals. The emitters of the lower transistors of each bridge are connected together and the corresponding external terminal can be used for the connection of an external sensing resistor. An additional supply input is provided so that the logic works at a lower voltage.

BLOCK DIAGRAM



PIN FUNCTIONS (refer to the block diagram)

MW.15	PowerSO	Name	Function
1;15	2;19	Sense A; Sense B	Between this pin and ground is connected the sense resistor to control the current of the load.
2;3	4;5	Out 1; Out 2	Outputs of the Bridge A; the current that flows through the load connected between these two pins is monitored at pin 1.
4	6	V_S	Supply Voltage for the Power Output Stages. A non-inductive 100nF capacitor must be connected between this pin and ground.
5;7	7;9	Input 1; Input 2	TTL Compatible Inputs of the Bridge A.
6;11	8;14	Enable A; Enable B	TTL Compatible Enable Input: the L state disables the bridge A (enable A) and/or the bridge B (enable B).
8	1,10,11,20	GND	Ground.
9	12	VSS	Supply Voltage for the Logic Blocks. A100nF capacitor must be connected between this pin and ground.
10; 12	13;15	Input 3; Input 4	TTL Compatible Inputs of the Bridge B.
13; 14	16;17	Out 3; Out 4	Outputs of the Bridge B. The current that flows through the load connected between these two pins is monitored at pin 15.
–	3;18	N.C.	Not Connected

ELECTRICAL CHARACTERISTICS (V_S = 42V; V_{SS} = 5V, T_j = 25°C; unless otherwise specified)

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
V_S	Supply Voltage (pin 4)	Operative Condition	V_{IH} +2.5		46	V
V_{SS}	Logic Supply Voltage (pin 9)		4.5	5	7	V
I_S	Quiescent Supply Current (pin 4)	V_{en} = H; I_L = 0 V_i = L V_i = H		13 50	22 70	mA mA
		V_{en} = L V_i = X			4	mA
I_{SS}	Quiescent Current from V_{SS} (pin 9)	V_{en} = H; I_L = 0 V_i = L V_i = H		24 7	36 12	mA mA
		V_{en} = L V_i = X			6	mA
V_{iL}	Input Low Voltage (pins 5, 7, 10, 12)		–0.3		1.5	V
V_{iH}	Input High Voltage (pins 5, 7, 10, 12)		2.3		VSS	V
I_{iL}	Low Voltage Input Current (pins 5, 7, 10, 12)	V_i = L			–10	µA
I_{iH}	High Voltage Input Current (pins 5, 7, 10, 12)	Vi = H ≤ V_{SS} –0.6V		30	100	µA
V_{en} = L	Enable Low Voltage (pins 6, 11)		–0.3		1.5	V
V_{en} = H	Enable High Voltage (pins 6, 11)		2.3		V_{SS}	V
I_{en} = L	Low Voltage Enable Current (pins 6, 11)	V_{en} = L			–10	µA
I_{en} = H	High Voltage Enable Current (pins 6, 11)	V_{en} = H ≤ V_{SS} –0.6V		30	100	µA
$V_{CEsat(H)}$	Source Saturation Voltage	I_L = 1A I_L = 2A	0.95	1.35 2	1.7 2.7	V V
$V_{CEsat(L)}$	Sink Saturation Voltage	I_L = 1A (5) I_L = 2A (5)	0.85	1.2 1.7	1.6 2.3	V V
V_{CEsat}	Total Drop	I_L = 1A (5) I_L = 2A (5)	1.80		3.2 4.9	V V
V_{sens}	Sensing Voltage (pins 1, 15)		–1 (1)		2	V

ELECTRICAL CHARACTERISTICS (continued)

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
T_1 (V_i)	Source Current Turn-off Delay	0.5 V_i to 0.9 I_L (2); (4)		1.5		µs
T_2 (V_i)	Source Current Fall Time	0.9 I_L to 0.1 I_L (2); (4)		0.2		µs
T_3 (V_i)	Source Current Turn-on Delay	0.5 V_i to 0.1 I_L (2); (4)		2		µs
T_4 (V_i)	Source Current Rise Time	0.1 I_L to 0.9 I_L (2); (4)		0.7		µs
T_5 (V_i)	Sink Current Turn-off Delay	0.5 V_i to 0.9 I_L (3); (4)		0.7		µs
T_6 (V_i)	Sink Current Fall Time	0.9 I_L to 0.1 I_L (3); (4)		0.25		µs
T_7 (V_i)	Sink Current Turn-on Delay	0.5 V_i to 0.9 I_L (3); (4)		1.6		µs
T_8 (V_i)	Sink Current Rise Time	0.1 I_L to 0.9 I_L (3); (4)		0.2		µs
fc (V_i)	Commutation Frequency	I_L = 2A		25	40	KHz
T_1 (V_{en})	Source Current Turn-off Delay	0.5 V_{en} to 0.9 I_L (2); (4)		3		µs
T_2 (V_{en})	Source Current Fall Time	0.9 I_L to 0.1 I_L (2); (4)		1		µs
T_3 (V_{en})	Source Current Turn-on Delay	0.5 V_{en} to 0.1 I_L (2); (4)		0.3		µs
T_4 (V_{en})	Source Current Rise Time	0.1 I_L to 0.9 I_L (2); (4)		0.4		µs
T_5 (V_{en})	Sink Current Turn-off Delay	0.5 V_{en} to 0.9 I_L (3); (4)		2.2		µs
T_6 (V_{en})	Sink Current Fall Time	0.9 I_L to 0.1 I_L (3); (4)		0.35		µs
T_7 (V_{en})	Sink Current Turn-on Delay	0.5 V_{en} to 0.9 I_L (3); (4)		0.25		µs
T_8 (V_{en})	Sink Current Rise Time	0.1 I_L to 0.9 I_L (3); (4)		0.1		µs

1) 1)Sensing voltage can be –1 V for t ≤ 50 µsec; in steady state V_{sens} min ≥ –0.5 V.
2) See fig. 2.
3) See fig. 4.
4) The load must be a pure resistor.

Figure 1 : Typical Saturation Voltage vs. Output Current.

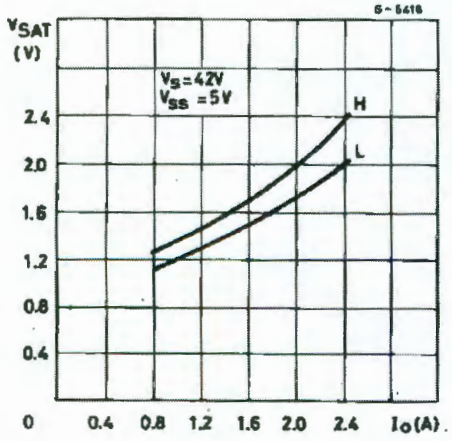
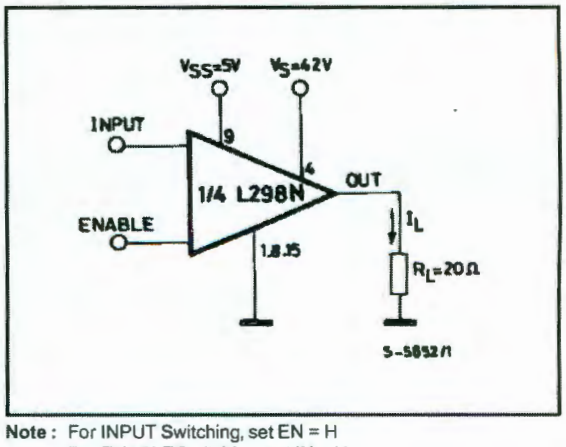


Figure 2 : Switching Times Test Circuits.



Note : For INPUT Switching, set EN = H
For ENABLE Switching, set IN = H

Figure 3 : Source Current Delay Times vs. Input or Enable Switching.

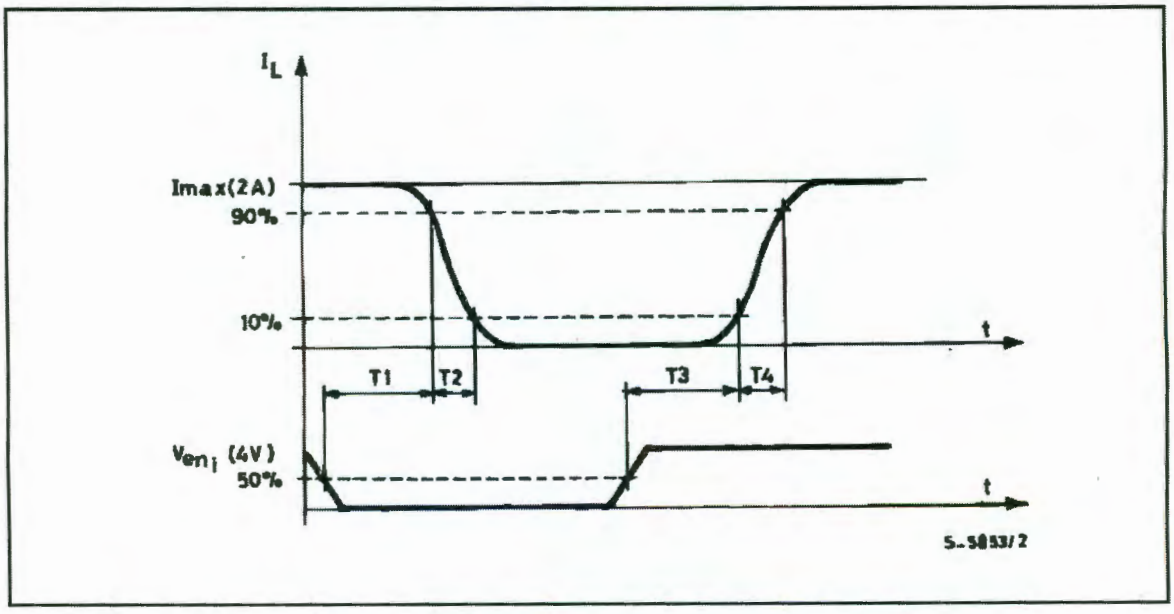
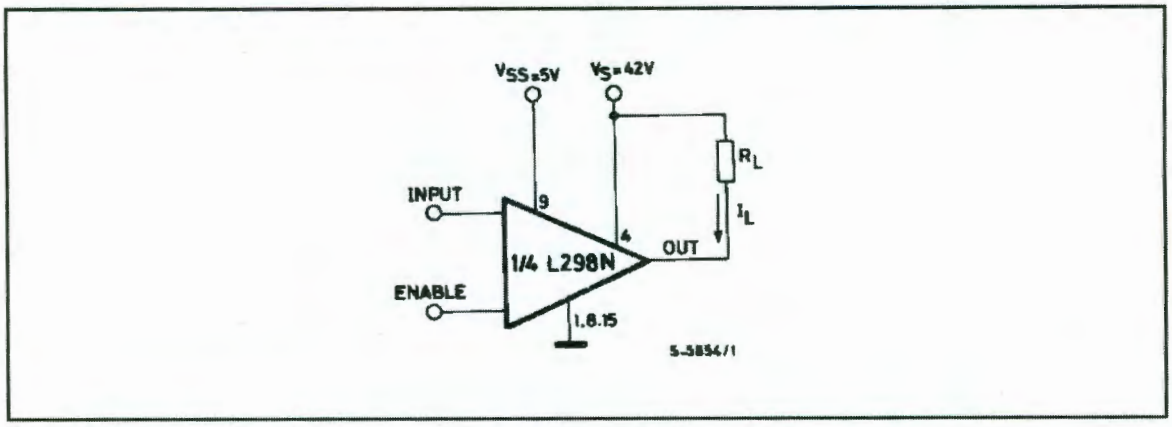


Figure 4 : Switching Times Test Circuits.

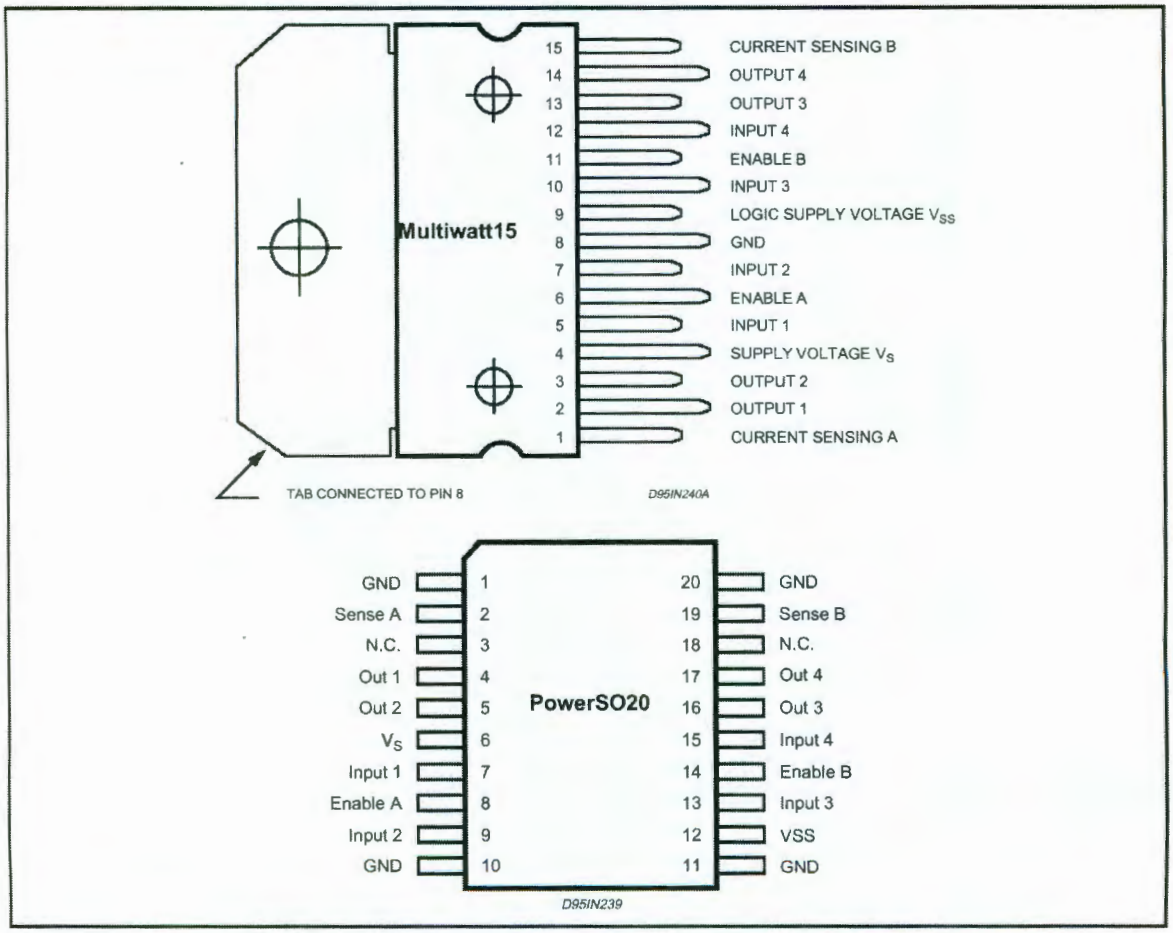


Note : For INPUT Switching, set EN = H
For ENABLE Switching, set IN = L

ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V_S	Power Supply	50	V
V_{SS}	Logic Supply Voltage	7	V
V_i, V_{en}	Input and Enable Voltage	–0.3 to 7	V
I_O	Peak Output Current (each Channel) – Non Repetitive (t = 100µs) –Repetitive (80% on –20% off; t_{on} = 10ms) –DC Operation	 3 2.5 2	 A A A
V_{sens}	Sensing Voltage	–1 to 2.3	V
P_{tot}	Total Power Dissipation (T_{case} = 75°C)	25	W
T_{op}	Junction Operating Temperature	–25 to 130	°C
T_{stg}, T_j	Storage and Junction Temperature	–40 to 150	°C

PIN CONNECTIONS (top view)



THERMAL DATA

Symbol	Parameter		PowerSO20	Multiwatt15	Unit
$R_{th\ j\text{-}case}$	Thermal Resistance Junction-case	Max.	–	3	°C/W
$R_{th\ j\text{-}amb}$	Thermal Resistance Junction-ambient	Max.	13 (*)	35	°C/W

(*) Mounted on aluminum substrate

Anexo 4.9.

Hoja de especificaciones del EDE 12400.

E-Lab Digital Engineering, Inc.

EDE12400

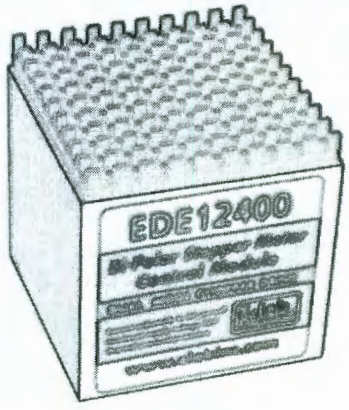
Bi-Polar Chopper Stepper Motor Control Module

Features

- Develops maximum possible motor torque by using dual coil-current sensing & control loop circuits
- Allows use of drive voltage beyond rated motor specification for enhanced torque & speed
- Chopper drive circuitry is current adjustable up to 2 Amps/coil using dual internal 5W sense resistors
- Motor supply voltage 0V to 46V
- Integral heatsink system and thermal potting compound eliminate need for an auxiliary heatsink or fan
- Eight internal 3A Schottky clamp diodes and large filter capacitors for enhanced noise suppression
- Two modes of current chopping provide efficient operation of both large and small stepper motors
- Internally generated voltage source for easily setting maximum coil current
- Primary drive circuit thermal overload protection
- Standard 24 pin DIP pin spacing for easy PCB placement & prototyping
- Threaded mounting coupler allows secure mount to PCB in rugged applications
- Chopping frequency generated internally; externally generated frequency may also be used

Overview

The EDE12400 stepper motor control module offers designers a compact, reliable stepper motor control system. Engineered with internal and external heatsinks and a highly thermally conductive potting compound, the need for cooling fans (known for short lifetimes) or a large heatsink plate is eliminated. An integrated chopper drive circuit safely provides the maximum motor torque for a given drive voltage, even one many times over the manufacture-specified voltage, offering tremendous torque and speed improvements over traditional stepper motor control circuits. Maximum coil current is easily set using a potentiometer or voltage divider, and can be dynamically adjusted. The highly efficient design of the EDE12400 drive circuitry combined with its unique PowerCube™ package makes it the ideal motor control solution for nearly any application.



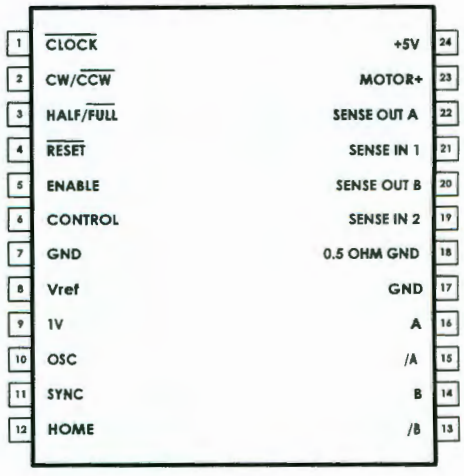
Specification Summary

Max. motor voltage 46V
Max. current 2 Amps per coil
Full/half stepping and direction control
Complete stepper motor control unit
Based on the proven L297/L298™ chipset

Typical Applications

CNC / Milling Machines
Robotics
Industrial Equipment
Remote-Positioning Equipment
Scientific Apparatus
Valve Controls

Module Pinout



Pin#	Pin Name	Description
1	$\overline{\text{CLOCK}}$	Step clock. An active low pulse on this input advances the motor one increment. The step occurs on the rising edge of this signal.
2	CW/$\overline{\text{CCW}}$	Clockwise/Counterclockwise direction control input. Physical direction of motor rotation also depends on connection of windings. Direction can be changed at any time.
3	HALF/$\overline{\text{FULL}}$	Half/full step select input. When high selects half step operation, when low selects full step operation. One-phase-on full step mode is obtained by selecting FULL when the translator is at an even-numbered state. Two-phase-on full step mode is set by selecting FULL when the translator is at an odd numbered position. (The home position is designated state 1).
4	$\overline{\text{RESET}}$	Reset input. An active low pulse on this input restores the translator to the home position (state 1, ABCD = 0101).
5	ENABLE	Chip enable input. When low (inactive) INH1,INH2,A,B,C, and D are brought low.
6	CONTROL	Control input that defines action of chopper. When low chopper acts on INH1 and INH2; when high chopper acts on phase lines ABCD.
7	GND	Ground connection.
8	Vref	Reference voltage input for chopper circuit. A voltage applied to this pin determines the peak load current. When using internal 0.5 Ohm current sense resistors, do not exceed 1V (sets 2 Amps).
9	1V	1V output voltage. May be used to feed a voltage divider circuit to set Vref input voltage. Using this output to drive a potentiometer or other voltage divider that sets Vref prevents (desirably) the possibility of sending a voltage higher than 1V into Vref.
10	OSC	RC oscillator to set chopper rate. In ordinary operation this pin may be left unconnected to use the internal RC oscillator. If multiple modules are to be utilized and their chopper outputs are to be synchronized, this pin should be grounded on all but one module. The module with the ungrounded OSC pin provides the chopper clock to the other modules via the SYNC pin.
11	SYNC	Output of the chopper oscillator. In ordinary operation this pin may be left unconnected. If an external chopper clock source is to be used it is injected at this pin. If multiple modules must have their chopper frequencies synchronized their SYNC pins should be connected.
12	HOME	Open collector output that indicates when the controller is in its initial state (ABCD = 0101). The output transistor is open when the signal is active.
13	/B	Motor phase 4 output drive signal. Connected to same coil as B.
14	B	Motor phase 3 output drive signal. Connected to same coil as /B.
15	/A	Motor phase 2 output drive signal. Connected to same coil as A.
16	A	Motor phase 1 output drive signal. Connected to same coil as /A.
17	GND	Ground connection.
18	0.5 Ohm Ground	Ordinarily connected to Ground if internal current sense resistors are to be used. If external current sense resistors are used, leave this pin floating.
19	Sense In 2	Input for load current sense resistor for coil across B and /B. For standard operation connect to Sense Out B.
20	Sense Out B	Output drive to load current sense resistor for coil across B and /B.
21	Sense In 1	Input for load current sense resistor for coil across A and /A. For standard operation connect to Sense Out A.
22	Sense Out A	Output drive to load current sense resistor for coil across A and /A.
23	MOTOR+	Motor power supply input. Maximum 46VDC.
24	+5V	Regulated +5V input.

Table One: Pin Functionality

Introduction

The EDE12400 Stepper Motor Control Module is built upon the L297/L298 chopper drive chipset manufactured by ST-Microelectronics. Support circuitry is incorporated to provide a complete bipolar chopper stepper motor interface. This datasheet may be used in conjunction with the L297 Datasheet, L298 Datasheet, and L297 Application Note for greater detail. These three documents are available from the E-Lab website (www.elabinc.com), the St-Microelectronics website (www.st.com), or the E-Lab Datasheet CD. As illustrated in Figure One, minimal external components are required to implement a full-featured chopper drive stepper motor control system.

Operational Overview

The EDE12400 contains all necessary circuitry for controlling a bipolar stepper motor at coil currents up to 2 Amps. Full & half stepping, directional control, motor enable/disable, and automatic current regulation provide a powerful, easy-to-use motion control system. The built-in chopper frequency generation and current sensing circuitry drives the motor at a presettable coil current which is determined by the voltage fed to the Vref input (pin 8).

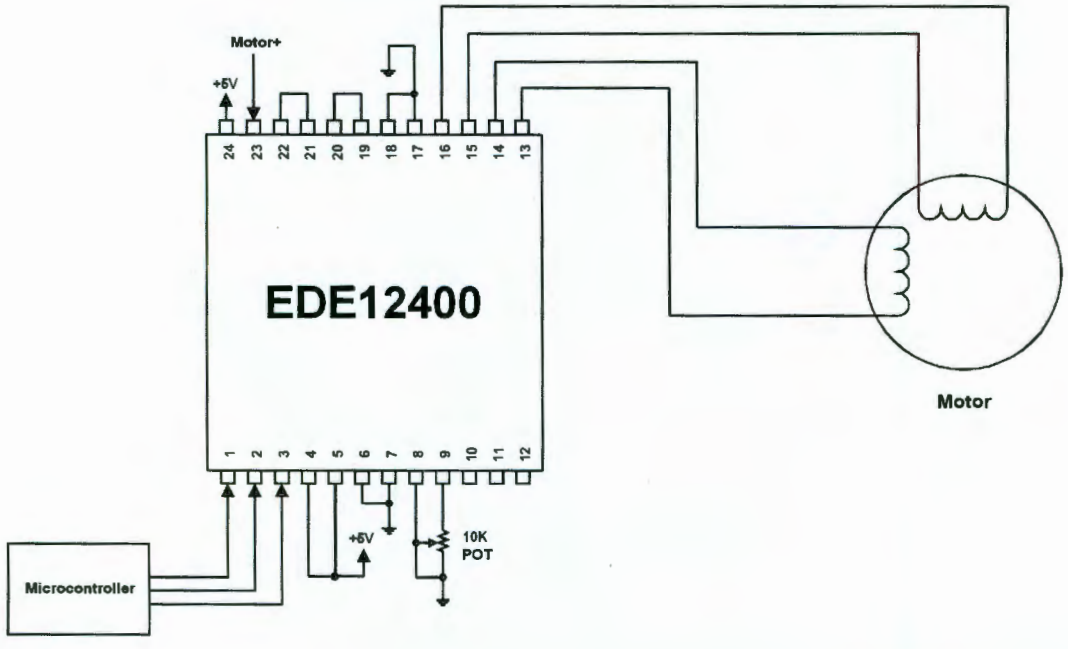


Figure One: Standard Module Hookup

When connected as shown in Figure One, the EDE12400 module will operate the motor based upon the inputs of the CLOCK, CW/CCW, HALF/FULL, and RESET pins. With the RESET pin high (its inactive state), a low-going (+5V to 0V) pulse on the CLOCK input will cause the motor to rotate one step at the low-to-high transition of the pulse. The CW/CCW pin determines the direction of shaft rotation. The HALF/FULL pin determines whether the module uses a standard full-step drive sequence (providing a 1.8°/per step rotation on a 1.8°/per step motor) or a half-step drive sequence (providing a 0.9°/per step rotation on a 1.8°/per step motor).

The drive outputs (A, /A, B, /B) that connect to the motor coils (see Figure One) cycle through one set of the following states depending upon whether full or half-stepping is selected. State 1, termed the 'Home' state, is the default power-on and reset state. The open-collector output HOME (pin 12) is active when the module is in state 1. Further discussion of output states may be found beginning page 5 of the L297 datasheet.

STATE	A	/A	B	/B
1	0	1	0	1
3	1	0	0	1
5	1	0	1	0
7	0	1	1	0

Table Two: Full Step Output States

STATE	A	/A	B	/B
1	0	1	0	1
2	0	0	0	1
3	1	0	0	1
4	1	0	0	0
5	1	0	1	0
6	0	0	1	0
7	0	1	1	0
8	0	1	0	0

Table Three: Half-Step Output States

Chopper Drive Fundamentals

Stepper motor torque is inversely proportional to motor rotation speed due to the inductance of the motor's coils. As rotational speed increases, it is more difficult to push the required amount of current into (and pull out of) the coils in the shorter period of time they are driven per step. As coil current decreases, so does motor torque. To overcome this, it is desirable to increase the drive voltage beyond the motor's rated voltage to increase current flow. Doing so leads to a problem, however, in that at lower speeds an overcurrent situation develops and the motor quickly overheats. The use of a chopper drive system, which places a higher voltage across the coils until the desired current setpoint is reached, allows coil current to remain at a desired level for both high and low speeds without the fear of overheating the motor or overdriving the coils. The EDE12400 applies motor input power to the coils as a square wave with varying duty cycle to dynamically control coil current. The drive frequency is set to 20KHz by an internal RC oscillator.

When connected as shown in Figure One, coil current is passed through internal 0.5 Ohm power resistors and then flows to ground. By measuring the voltage across these resistors the coil current may be determined. Following Ohm's law (i = v/r), the current through a resistor is equal to the voltage across the resistor divided by resistance, in this case 0.5 Ohms. As an example, if the voltage across one of the 0.5 Ohm sense resistors is 0.5 Volts, one Amp of current is flowing through the resistor, and therefore through the motor coil as well. When the EDE12400 detects that there is less current flowing through the coil than there should be it connects the Motor+ (pin 23) voltage input to the coil. As current begins to flow, the voltage across the sense resistor increases. When the increasing sense resistor voltage becomes equal to the Vref input voltage, the Motor+ voltage is

removed from the coil until the next chopper cycle (at 20KHz, the PWM period is 50us). Because the maximum module current is 2 Amps per coil, care should be taken to ensure that the maximum voltage applied to the Vref input is 1 Volt; otherwise current will exceed 2 Amps per coil and damage to the module may occur. To aid in usage, a 1V output (pin 9) is available to drive a voltage divider or potentiometer. Using this 1V signal (as opposed to +5V or more) ensures that the Vref current control input stays within the 0-1 Volt range. A simple voltage divider arrangement uses two resistors in series with one end connected to the 1V output from the module and the other end to ground. The connection point between the two resistors is then connected to the Vref input as the input voltage for the current limit. As an example, to limit current flow to .5 Amps (500mA) per coil one would need to place a voltage of 0.25V onto the Vref pin. This may be accomplished using a 5K and a 20K resistor is series, with the 5K resistor connected to ground on one end. A potentiometer may also be used as a voltage divider; one end connected to the 1V output of the module, the other to Ground, and the wiper to the module's Vref input. This arrangement is illustrated in Figure One.

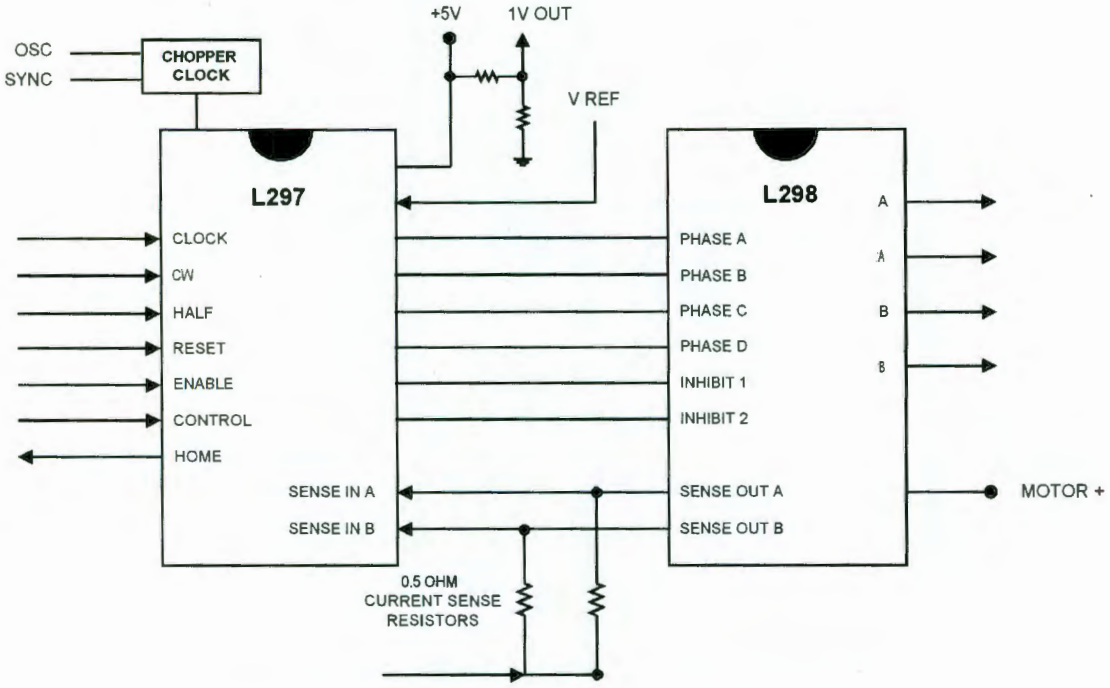


Figure Two: Internal 297-298 Connection Block Diagram

As illustrated by Figure Two, connection to the internal current sense resistors is available externally to the module at pin 18, the '0.5 Ohm GND' pin. Ordinarily, when using the internal sense resistors, Sense Out A (pin 22) is connected to Sense In 1 (pin 21), Sense Out B (pin 20) is connected to Sense In 2 (pin 19), and 0.5 Ohm GND (pin 18) is connected to GND. If the use of external sense resistors is desired instead (for instance, to reduce power consumption), one leg of each of the two external sense resistors should be connected to Sense Out A and Sense Out B with the other two legs grounded. The 0.5 Ohm GND (pin 18) should be left floating, and Sense In 1 & Sense In 2 should be connected to sense Out A & Sense Out B, respectively. Care should be taken to ensure that if external resistors are used they are capable of carrying the maximum coil current.

Anexo 5.1.

Hoja de especificaciones de la ULN2803.

Order this document by ULN2803/D

MOTOROLA

Octal High Voltage, High Current Darlington Transistor Arrays

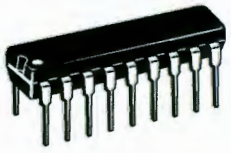
ULN2803
ULN2804

OCTAL PERIPHERAL DRIVER ARRAYS

SEMICONDUCTOR TECHNICAL DATA

The eight NPN Darlington connected transistors in this family of arrays are ideally suited for interfacing between low logic level digital circuitry (such as TTL, CMOS or PMOS/NMOS) and the higher current/voltage requirements of lamps, relays, printer hammers or other similar loads for a broad range of computer, industrial, and consumer applications. All devices feature open–collector outputs and free wheeling clamp diodes for transient suppression.

The ULN2803 is designed to be compatible with standard TTL families while the ULN2804 is optimized for 6 to 15 volt high level CMOS or PMOS.



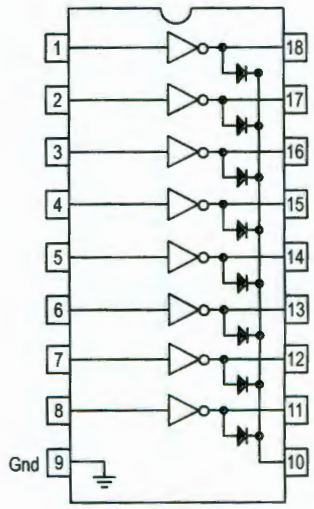
A SUFFIX
PLASTIC PACKAGE
CASE 707

MAXIMUM RATINGS (T_A = 25°C and rating apply to any one device in the package, unless otherwise noted.)

Rating	Symbol	Value	Unit
Output Voltage	V_O	50	V
Input Voltage (Except ULN2801)	V_I	30	V
Collector Current – Continuous	I_C	500	mA
Base Current – Continuous	I_B	25	mA
Operating Ambient Temperature Range	T_A	0 to +70	°C
Storage Temperature Range	T_{stg}	–55 to +150	°C
Junction Temperature	T_J	125	°C

$R_{\theta JA}$ = 55°C/W
Do not exceed maximum current limit per driver.

PIN CONNECTIONS



ORDERING INFORMATION

Device	Characteristics		
	Input Compatibility	V_{CE}(Max)/I_C(Max)	Operating Temperature Range
ULN2803A ULN2804A	TTL, 5.0 V CMOS 6 to 15 V CMOS, PMOS	50 V/500 mA	T_A = 0 to + 70°C

ULN2803 ULN2804

ELECTRICAL CHARACTERISTICS (T_A = 25°C, unless otherwise noted)

Characteristic		Symbol	Min	Typ	Max	Unit
Output Leakage Current (Figure 1)		I_{CEX}				µA
(V_O = 50 V, T_A = +70°C)	All Types		–	–	100	
(V_O = 50 V, T_A = +25°C)	All Types		–	–	50	
(V_O = 50 V, T_A = +70°C, V_I = 6.0 V)	ULN2802		–	–	500	
(V_O = 50 V, T_A = +70°C, V_I = 1.0 V)	ULN2804		–	–	500	
Collector–Emitter Saturation Voltage (Figure 2)		$V_{CE(sat)}$				V
(I_C = 350 mA, I_B = 500 µA)	All Types		–	1.1	1.6	
(I_C = 200 mA, I_B = 350 µA)	All Types		–	0.95	1.3	
(I_C = 100 mA, I_B = 250 µA)	All Types		–	0.85	1.1	
Input Current – On Condition (Figure 4)		$I_{I(on)}$				mA
(V_I = 17 V)	ULN2802		–	0.82	1.25	
(V_I = 3.85 V)	ULN2803		–	0.93	1.35	
(V_I = 5.0 V)	ULN2804		–	0.35	0.5	
(V_I = 12 V)	ULN2804		–	1.0	1.45	
Input Voltage – On Condition (Figure 5)		$V_{I(on)}$				V
(V_{CE} = 2.0 V, I_C = 300 mA)	ULN2802		–	–	13	
(V_{CE} = 2.0 V, I_C = 200 mA)	ULN2803		–	–	2.4	
(V_{CE} = 2.0 V, I_C = 250 mA)	ULN2803		–	–	2.7	
(V_{CE} = 2.0 V, I_C = 300 mA)	ULN2803		–	–	3.0	
(V_{CE} = 2.0 V, I_C = 125 mA)	ULN2804		–	–	5.0	
(V_{CE} = 2.0 V, I_C = 200 mA)	ULN2804		–	–	6.0	
(V_{CE} = 2.0 V, I_C = 275 mA)	ULN2804		–	–	7.0	
(V_{CE} = 2.0 V, I_C = 350 mA)	ULN2804		–	–	8.0	
Input Current – Off Condition (Figure 3) (I_C = 500 µA, T_A = +70°C)	All Types	$I_{I(off)}$	50	100	–	µA
DC Current Gain (Figure 2) (V_{CE} = 2.0 V, I_C = 350 mA)	ULN2801	h_{FE}	1000	–	–	–
Input Capacitance		C_I	–	15	25	pF
Turn–On Delay Time (50% E_I to 50% E_O)		t_{on}	–	0.25	1.0	µs
Turn–Off Delay Time (50% E_I to 50% E_O)		t_{off}	–	0.25	1.0	µs
Clamp Diode Leakage Current (Figure 6) (V_R = 50 V)	T_A = +25°C T_A = +70°C	I_R	–	–	50 100	µA
Clamp Diode Forward Voltage (Figure 7) (I_F = 350 mA)		V_F	–	1.5	2.0	V

ULN2803 ULN2804

TEST FIGURES

(See Figure Numbers in Electrical Characteristics Table)

Figure 1.

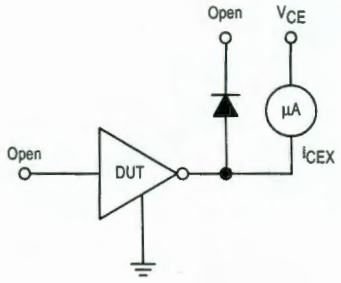


Figure 2.

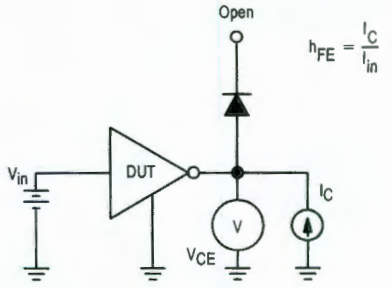


Figure 3.

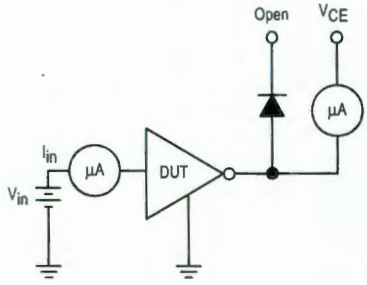


Figure 4.

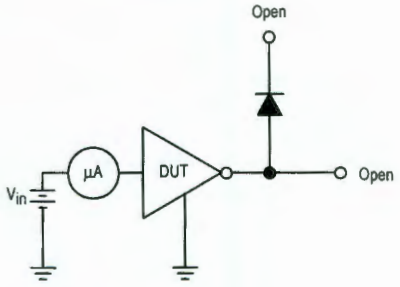


Figure 5.

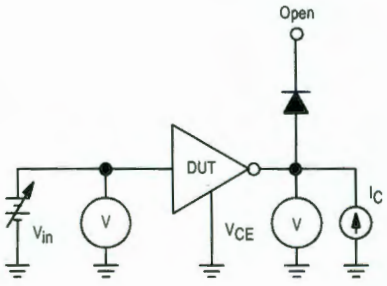


Figure 6.

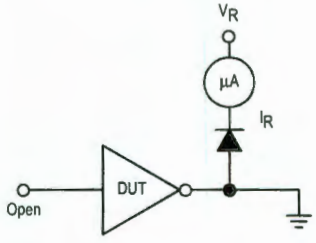
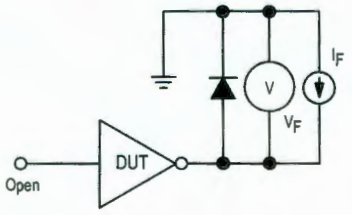


Figure 7.



TYPICAL CHARACTERISTIC CURVES – T_A = 25°C, unless otherwise noted

Output Characteristics

Figure 8. Output Current versus Saturation Voltage

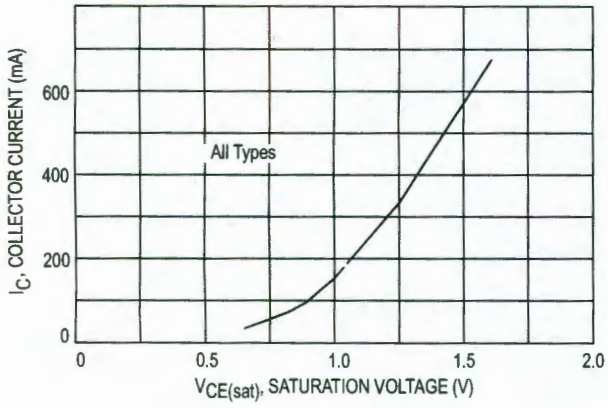
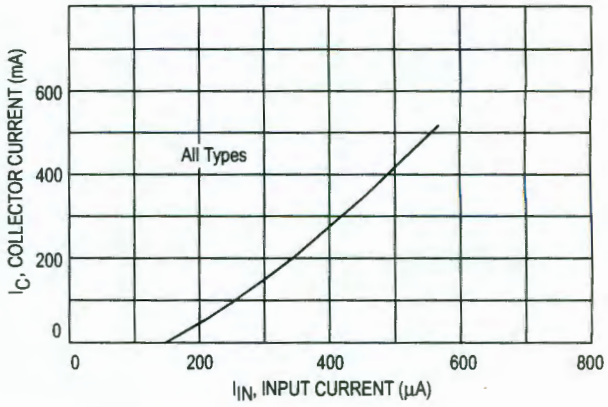


Figure 9. Output Current versus Input Current



Input Characteristics

Figure 10. ULN2803 Input Current versus Input Voltage

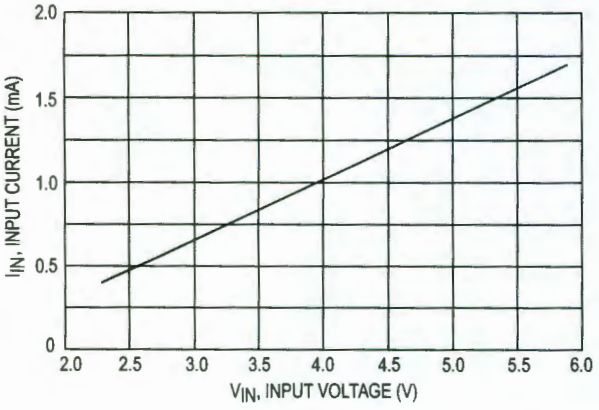


Figure 11. ULN2804 Input Current versus Input Voltage

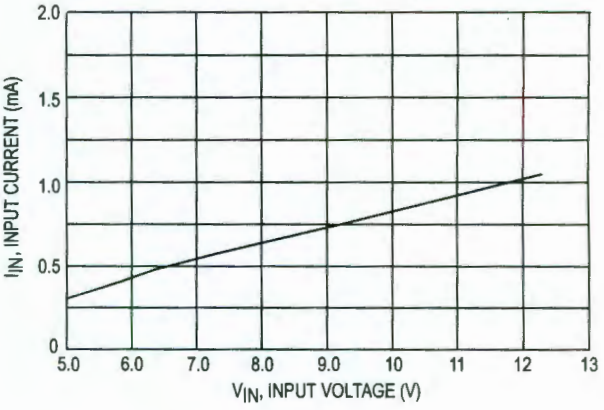
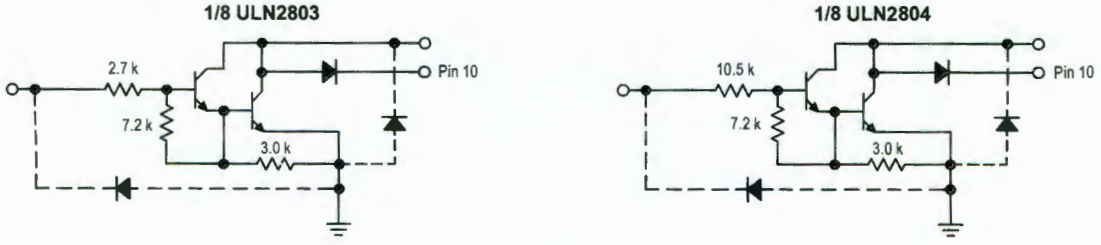


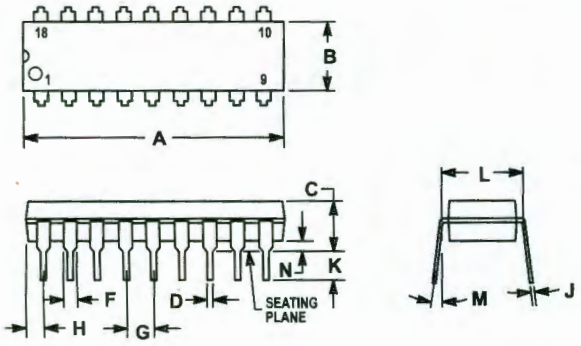
Figure 12. Representative Schematic Diagrams



ULN2803 ULN2804

OUTLINE DIMENSIONS

A SUFFIX
PLASTIC PACKAGE
CASE 707–02
ISSUE C



NOTES:
1. POSITIONAL TOLERANCE OF LEADS (D), SHALL BE WITHIN 0.25 (0.010) AT MAXIMUM MATERIAL CONDITION, IN RELATION TO SEATING PLANE AND EACH OTHER.
2. DIMENSION L TO CENTER OF LEADS WHEN FORMED PARALLEL.
3. DIMENSION B DOES NOT INCLUDE MOLD FLASH.

	MILLIMETERS		INCHES	
DIM	MIN	MAX	MIN	MAX
A	22.22	23.24	0.875	0.915
B	6.10	6.60	0.240	0.260
C	3.56	4.57	0.140	0.180
D	0.36	0.56	0.014	0.022
F	1.27	1.78	0.050	0.070
G	2.54 BSC		0.100 BSC	
H	1.02	1.52	0.040	0.060
J	0.20	0.30	0.008	0.012
K	2.92	3.43	0.115	0.135
L	7.62 BSC		0.300 BSC	
M	0°	15°	0°	15°
N	0.51	1.02	0.020	0.040

How to reach us:

USA/EUROPE/Locations Not Listed: Motorola Literature Distribution; P.O. Box 20912; Phoenix, Arizona 85036. 1–800–441–2447 or 602–303–5454

MFAX: RMFAX0@email.sps.mot.com – TOUCHTONE 602–244–6609

INTERNET: http://Design–NET.com

JAPAN: Nippon Motorola Ltd.; Tatsumi–SPD–JLDC, 6F Seibu–Butsuryu–Center, 3–14–2 Tatsumi Koto–Ku, Tokyo 135, Japan. 03–81–3521–8315

ASIA/PACIFIC: Motorola Semiconductors H.K. Ltd.; 8B Tai Ping Industrial Park, 51 Ting Kok Road, Tai Po, N.T., Hong Kong. 852–26629298

Anexo 6.1.
Programa en lenguaje C de la mesa XY.

```
#include <dos.h>
#include <conio.h>
#include <stdio.h>
#include <math.h>)

extern far void Recibir_Datos(int X);
extern far void Datos2(int X);

void subir_bajar_pluma();
void mueve_motor_x();
void mueve_motor_z();
void ajustar_velocidad();
void mover_por_pasos();
void menu();
void despedida();
void reporte_de_pasos();
void mueve_motor_y();
void jogging();
void hacer_linea();
void hacer_arco();
void pedir_radio();
void pedir_tipo_de_arco();
void linea_horizontal();
void linea_vertical();
void pedir_datos_linea();
void uaq();
void mandar_datos();

int
pasos_x,pasos_y,pasos_z,bobinas_x=1,bobinas_y=1,bo
binas_z=1;
unsigned long int
conteo_de_pasos_x,conteo_de_pasos_y,conteo_de_pas
os_z;

int opciones,sentido_x,sentido_y,sentido_z;
int joggear;

unsigned long int
qx,qy,qx1,qy1,Azx,Azy,px,py,px1,py1,Azz;
unsigned long int pasos_mayores,pasos_comparados;

unsigned long int
p0=0,px_c=0,py_c=0,qx_c=0,qy_c=0,qx1_c=0,qy1_c=0,r
adio;

int tipo_de_arco;

int espera=3,velocidad,cada_ciclo=1,aceleracion,NP;
int espera_inicial=300;
long int
posicion_global_x=0,posicion_global_y=0,posicion_glob
al_z;

int valor_x=0,valor_y=0,valor_z=0,valor_xy=0;

void main()
{
clrscr();
menu();
despedida();
}

void despedida()
{
clrscr();
gotoxy(14,10);printf("Thank you for using Israel and
Cadena Products");
gotoxy(15,15);printf("You have left the XYZ moving table
program");
getche();
}

void menu()
{
conteo_de_pasos_x=0;
conteo_de_pasos_y=0;
Recibir_Datos(0);
Datos2(0);
px=0,py=0,qx=0,qy=0,Azx=0,Azy=0,pasos_mayores=0;

clrscr();
gotoxy(14,5);printf("1) Mover individualmente los motores
por pasos");
gotoxy(14,6);printf("2) Jogging");
gotoxy(14,7);printf("3) Traza una l¡nea");
gotoxy(14,8);printf("4) Traza un c¡rculo");
gotoxy(14,9);printf("5) Escribir UAQ");
gotoxy(14,10);printf("6) Salir del programa");
gotoxy(14,1);printf("Introduce el n£mero de lo que
quieres hacer\n");
gotoxy(15,3);
scanf("%d",&opciones);
        switch(opciones)
                {
                case
1:pedir_datos_linea();mover_por_pasos();reporte_de_pa
sos();menu();break;
                case
2:clrscr();jogging();reporte_de_pasos();menu();break;
                case
3:pedir_datos_linea();hacer_linea();reporte_de_pasos();
menu();break;
                case
4:pedir_tipo_de_arco();menu();break;
                case
5:uaq();reporte_de_pasos();menu();
                case 6:break;
                default:gotoxy(12,14); printf("S¢lo
tenemos seis opciones int,ntalo de
nuevo");getche();menu();break;
                }
}

void reporte_de_pasos()
{
gotoxy(20,14);printf("Los pasos dados en el eje X son
%lu\n",conteo_de_pasos_x);
gotoxy(20,16);printf("Los pasos dados en el eje Y son
%lu\n",conteo_de_pasos_y);
gotoxy(20,18);printf("Ha salido del programa que estaba
usando");
gotoxy(12,22);printf("Presione ahora cualquier tecla para
volver al menu principal");getche();
Recibir_Datos(0);
Datos2(0);
}
```

```
void pedir_datos_linea()
{
clrscr();
gotoxy(1,1);printf("Dame los pasos en el eje X ");
gotoxy(1,2);scanf("%d",&pasos_x);
gotoxy(1,3);printf("Mover la mesa hacia:");
gotoxy(1,4);printf("1)Derecha");
gotoxy(1,5);printf("2)Izquierda\n");
gotoxy(1,6);scanf("%d",&sentido_x);
gotoxy(33,1);printf("Dame los pasos en el eje Y ");
gotoxy(33,2);scanf("%d",&pasos_y);
gotoxy(33,3);printf("Mover la mesa hacia:");
gotoxy(33,4);printf("1)Abajo");
gotoxy(33,5);printf("2)Arriba\n");
gotoxy(33,6);scanf("%d",&sentido_y);
}


void mover_por_pasos()
{
  do{
//     ajustar_velocidad();
     pasos_x--;
     Azx=1;
     mueve_motor_x();
     mandar_datos();
    }while (pasos_x>=1);
//  espera=15;
  do{
//     ajustar_velocidad();
     pasos_y--;
     Azy=1;
     mueve_motor_y();
     mandar_datos();
    }while (pasos_y>=1);
//  espera=15;
}

void mueve_motor_x()
{
if (Azx==1)
 {
  if (sentido_x<=1)
  {   posicion_global_x++;
      bobinas_x++;
      gotoxy(30,25);
      switch(bobinas_x)
            {
             case
2:valor_x=1;gotoxy(15,20);printf("Posici¢n en X: %ld
",posicion_global_x);break;
             case
3:valor_x=3;gotoxy(15,20);printf("Posici¢n en X: %ld
",posicion_global_x);break;
             case
4:valor_x=2;gotoxy(15,20);printf("Posici¢n en X: %ld
",posicion_global_x);break;
             case
5:valor_x=6;gotoxy(15,20);printf("Posici¢n en X: %ld
",posicion_global_x);break;
             case
6:valor_x=4;gotoxy(15,20);printf("Posici¢n en X: %ld
",posicion_global_x);break;
             case
7:valor_x=12;gotoxy(15,20);printf("Posici¢n en X: %ld
",posicion_global_x);break;
             case
8:valor_x=8;gotoxy(15,20);printf("Posici¢n en X: %ld
",posicion_global_x);break;
             case
9:valor_x=9;gotoxy(15,20);printf("Posici¢n en X: %ld
",posicion_global_x);break;
             default: printf("error");break;
            }
             if (bobinas_x>=9)
bobinas_x=1;conteo_de_pasos_x++;
             //ajustar_velocidad();
  }
   else
   {   posicion_global_x--;
             bobinas_x++;
             gotoxy(30,25);
             switch(bobinas_x)
             {
              case
2:valor_x=9;gotoxy(15,20);printf("Posici¢n en X: %ld
",posicion_global_x);break;
              case
3:valor_x=8;gotoxy(15,20);printf("Posici¢n en X: %ld
",posicion_global_x);break;
              case
4:valor_x=12;gotoxy(15,20);printf("Posici¢n en X: %ld
",posicion_global_x);break;
              case
5:valor_x=4;gotoxy(15,20);printf("Posici¢n en X: %ld
",posicion_global_x);break;
              case
6:valor_x=6;gotoxy(15,20);printf("Posici¢n en X: %ld
",posicion_global_x);break;
              case
7:valor_x=2;gotoxy(15,20);printf("Posici¢n en X: %ld
",posicion_global_x);break;
              case
8:valor_x=3;gotoxy(15,20);printf("Posici¢n en X: %ld
",posicion_global_x);break;
              case
9:valor_x=1;gotoxy(15,20);printf("Posici¢n en X: %ld
",posicion_global_x);break;
              default: printf("error");break;
             }
             if (bobinas_x>=9)
bobinas_x=1;conteo_de_pasos_x++;
             //ajustar_velocidad();
   }
  }
}

void mueve_motor_z()
{
if (Azz==1)
 {
  if (sentido_z==1)
  {   posicion_global_z++;
      bobinas_z++;
      gotoxy(30,25);
      switch(bobinas_z)
            {
             case
2:valor_z=1;gotoxy(15,20);printf("Posici¢n en Z: %ld
",posicion_global_z);break;
             case
3:valor_z=3;gotoxy(15,20);printf("Posici¢n en Z: %ld
",posicion_global_z);break;
             case
4:valor_z=2;gotoxy(15,20);printf("Posici¢n en Z: %ld
",posicion_global_z);break;
             case
5:valor_z=6;gotoxy(15,20);printf("Posici¢n en Z: %ld
",posicion_global_z);break;
```

```
            case
6:valor_z=4;gotoxy(15,20);printf("Posici¢n en Z: %ld
",posicion_global_z);break;
            case
7:valor_z=12;gotoxy(15,20);printf("Posici¢n en Z: %ld
",posicion_global_z);break;
            case
8:valor_z=8;gotoxy(15,20);printf("Posici¢n en Z: %ld
",posicion_global_z);break;
            case
9:valor_z=9;gotoxy(15,20);printf("Posici¢n en Z: %ld
",posicion_global_z);break;
            default: printf("error");break;
           }
            if (bobinas_z>=9)
bobinas_z=1;conteo_de_pasos_z++;
            //ajustar_velocidad();
 }
  else
  {   posicion_global_z--;
           bobinas_z++;
           gotoxy(30,25);
           switch(bobinas_z)
           {
            case
2:valor_z=9;gotoxy(15,20);printf("Posici¢n en Z: %ld
",posicion_global_z);break;
            case
3:valor_z=8;gotoxy(15,20);printf("Posici¢n en Z: %ld
",posicion_global_z);break;
            case
4:valor_z=12;gotoxy(15,20);printf("Posici¢n en Z: %ld
",posicion_global_z);break;
            case
5:valor_z=4;gotoxy(15,20);printf("Posici¢n en Z: %ld
",posicion_global_z);break;
            case
6:valor_z=6;gotoxy(15,20);printf("Posici¢n en Z: %ld
",posicion_global_z);break;
            case
7:valor_z=2;gotoxy(15,20);printf("Posici¢n en Z: %ld
",posicion_global_z);break;
            case
8:valor_z=3;gotoxy(15,20);printf("Posici¢n en Z: %ld
",posicion_global_z);break;
            case
9:valor_z=1;gotoxy(15,20);printf("Posici¢n en Z: %ld
",posicion_global_z);break;
            default: printf("error");break;
           }
           if (bobinas_z>=9)
bobinas_z=1;conteo_de_pasos_z++;
           //ajustar_velocidad();
  }
 }
}

void mueve_motor_y()
{
if (Azy==1)
 {
 if (sentido_y==1)
 {   posicion_global_y++;
     bobinas_y++;
     gotoxy(32,25);

     switch(bobinas_y)
           {
            case
2:valor_y=16;gotoxy(42,20);printf("Posici¢n en Y: %ld
",posicion_global_y);break;
            case
3:valor_y=48;gotoxy(42,20);printf("Posici¢n en Y: %ld
",posicion_global_y);break;
            case
4:valor_y=32;gotoxy(42,20);printf("Posici¢n en Y: %ld
",posicion_global_y);break;
            case
5:valor_y=96;gotoxy(42,20);printf("Posici¢n en Y: %ld
",posicion_global_y);break;
            case
6:valor_y=64;gotoxy(42,20);printf("Posici¢n en Y: %ld
",posicion_global_y);break;
            case
7:valor_y=192;gotoxy(42,20);printf("Posici¢n en Y: %ld
",posicion_global_y);break;
            case
8:valor_y=128;gotoxy(42,20);printf("Posici¢n en Y: %ld
",posicion_global_y);break;
            case
9:valor_y=144;gotoxy(42,20);printf("Posici¢n en Y: %ld
",posicion_global_y);break;
            default: printf("error");break;
           }
            if (bobinas_y>=9)
bobinas_y=1;conteo_de_pasos_y++;
 }
 else
 {   posicion_global_y--;
     bobinas_y++;
     gotoxy(32,25);

     switch(bobinas_y)
           {
            case
2:valor_y=144;gotoxy(42,20);printf("Posici¢n en Y: %ld
",posicion_global_y);break;
            case
3:valor_y=128;gotoxy(42,20);printf("Posici¢n en Y: %ld
",posicion_global_y);break;
            case
4:valor_y=192;gotoxy(42,20);printf("Posici¢n en Y: %ld
",posicion_global_y);break;
            case
5:valor_y=64;gotoxy(42,20);printf("Posici¢n en Y: %ld
",posicion_global_y);break;
            case
6:valor_y=96;gotoxy(42,20);printf("Posici¢n en Y: %ld
",posicion_global_y);break;
            case
7:valor_y=32;gotoxy(42,20);printf("Posici¢n en Y: %ld
",posicion_global_y);break;
            case
8:valor_y=48;gotoxy(42,20);printf("Posici¢n en Y: %ld
",posicion_global_y);break;
            case
9:valor_y=16;gotoxy(42,20);printf("Posici¢n en Y: %ld
",posicion_global_y);break;
            default: printf("error");break;
           }
            if (bobinas_y>=9)
bobinas_y=1;conteo_de_pasos_y++;
 }
 }
}


void jogging()
```

```
{
            int paro;
            joggear=0;
            gotoxy(20,2);printf("Presiona la tecla para
mover lo motores hacia alg£n lugar");
            gotoxy(20,4);printf("8)Mover hacia arriba");
            gotoxy(20,6);printf("2)Mover hacia abajo");
            gotoxy(20,8);printf("4)Mover hacia la
derecha");
            gotoxy(20,10);printf("6)Mover hacia la
izquierda");
            gotoxy(20,12);printf("5)Parar");
            gotoxy(20,14);printf("p)Salir de jogging
\n");
            gotoxy(20,16);printf("s)Subir la Pluma   \n");
            gotoxy(20,18);printf("b)Bajar la Pluma   \n");

do{
   joggear=getch();

   while(joggear=='s')
          {
          Azz=1;
          Azy=0;
          sentido_z=2;
          mueve_motor_z();
          Datos2(valor_z);
          joggear=0;
          }

   while(joggear=='b')
          {
          Azz=1;
          Azy=0;
          sentido_z=1;
          mueve_motor_z();
          Datos2(valor_z);
          joggear=0;
          }

   while(joggear=='4')
          {
          Azx=1;
          Azy=0;
          sentido_x=1;
          mueve_motor_x();
          mandar_datos();
          joggear=0;
          }

   while(joggear=='6')
          {
          Azx=1;
          Azy=0;
          sentido_x=2;
          mueve_motor_x();
          mandar_datos();
          joggear=0;
          }
   while(joggear=='8')
          {
          Azx=0;
          Azy=1;
          sentido_y=2;
          mueve_motor_y();
          mandar_datos();
          joggear=0;
          }
   while(joggear=='2')
          {
          Azx=0;
          Azy=1;
          sentido_y=1;
          mueve_motor_y();
          mandar_datos();
          joggear=0;
          }
            gotoxy(20,2);printf("Presiona la tecla para
mover lo motores hacia alg£n lugar");
            gotoxy(20,4);printf("8)Mover hacia arriba");
            gotoxy(20,8);printf("2)Mover hacia abajo");
            gotoxy(20,10);printf("4)Mover hacia la
derecha");
            gotoxy(20,12);printf("6)Mover hacia la
izquierda");
            gotoxy(20,14);printf("p)Salir de jogging\n");
            gotoxy(20,16);printf("s)Subir la Pluma   \n");
            gotoxy(20,18);printf("b)Bajar la Pluma   \n");
          valor_x=0;
          valor_y=0;
 }while(joggear!='p');
  Recibir_Datos(0);
  Datos2(0);

clrscr();
}

void hacer_linea()
{
  unsigned long int aux;

  if(pasos_x>pasos_y)
   {  px=pasos_x;
          py=pasos_y;
          pasos_mayores=pasos_x;
   }
  else if(pasos_y>=pasos_x)
   {  py=pasos_y;
          px=pasos_x;
          pasos_mayores=pasos_y;
   }
  pasos_comparados=0;
  do{
    qx=(qx+px);
    if(qx>=(pasos_mayores+1))
    { qx1=qx;Azx=1;qx=(qx1-(pasos_mayores+1));
    }
   qy=(qy+py);
   if(qy>=(pasos_mayores+1))
    { qy1=qy;Azy=1;qy=(qy1-(pasos_mayores+1));
    }
   if(pasos_x>pasos_y)
          if (Azx>=1)
            pasos_comparados++;
          else delay(0);
   else
          if (Azy>=1)
            pasos_comparados++;

   mueve_motor_y();
   mueve_motor_x();
   mandar_datos();
   Azx=0;
   Azy=0; gotoxy(20,12);
//   gotoxy(20,14);printf("Conteo en X
%lu\n",conteo_de_pasos_x);
//   gotoxy(20,18);printf("Conteo en Y
%lu\n",conteo_de_pasos_y);
  }while(pasos_comparados<=(pasos_mayores-1));
}
```

```
void hacer_arco()
{  Datos2(0);
if (tipo_de_arco==1)
  { px_c=radio;p0=radio;py_c=0;}
else
  {px_c=0;py_c=radio;p0=radio;}

switch(tipo_de_arco)
{
 case 1:
  do{
   qx_c=(qx_c+px_c);
   if(qx_c>=(radio+1))
    { qx1_c=qx_c;Azx=1;qx_c=(qx1_c-(radio+1));
          py_c++;
    }

   qy_c=(qy_c+py_c);
   if(qy_c>=(radio+1))
    { qy1_c=qy_c;Azy=1;qy_c=(qy1_c-(radio+1));
          if (px_c>=1)
               px_c--;
    }

   mueve_motor_y();
   mueve_motor_x();
   mandar_datos();
   Azx=0;Azy=0;
   }while(px_c>=1);break;

 case 2:
  do{
   qy_c=(qy_c+py_c);
   if(qy_c>=(radio+1))
    { qy1_c=qy_c;Azy=1;qy_c=(qy1_c-(radio+1));
          px_c++;
    }

   qx_c=(qx_c+px_c);
   if(qx_c>=(radio+1))
    { qx1_c=qx_c;Azx=1;qx_c=(qx1_c-(radio+1));
          if (py_c>=1)
               py_c--;
    }

   mueve_motor_y();
   mueve_motor_x();
   mandar_datos();
   Azx=0;Azy=0;
//   gotoxy(30,10);printf("Conteo en X
%d\n",conteo_de_pasos_x);
//   gotoxy(30,11);printf("Conteo en Y
%d\n",conteo_de_pasos_y);
   }while(py_c>=1);break;
 }
}

void pedir_radio()
{
clrscr();gotoxy(1,1);printf("Dame el radio");
gotoxy(1,2);scanf("%lu",&radio);
}

void pedir_tipo_de_arco()
{
int arco;
clrscr();
gotoxy(14,5);printf("1) Arco de 0ø a 270ø");
gotoxy(14,6);printf("2) Arco de 270ø a 180ø");
gotoxy(14,7);printf("3) Arco de 180ø a 90ø");
gotoxy(14,8);printf("4) Arco de 90ø a 0ø");
gotoxy(14,9);printf("5) C¡rculo completo sentido anti-
horario");
gotoxy(14,10);printf("6) Arco de 0ø a 90ø");
gotoxy(14,11);printf("7) Arco de 90ø a 180ø");
gotoxy(14,12);printf("8) Arco de 180ø a 270ø");
gotoxy(14,13);printf("9) Arco de 270ø a 0ø");
gotoxy(14,14);printf("10) C¡rculo completo sentido
horario");
gotoxy(14,15);printf("11) Regresar a men£ principal");
gotoxy(14,1);printf("Introduce el n£mero de lo que
quieres hacer\n");
gotoxy(15,3);scanf("%d",&arco);
gotoxy(20,12);

          switch(arco)
          {
           case
1:sentido_x=1;sentido_y=1;tipo_de_arco=2;pedir_radio()
;hacer_arco();reporte_de_pasos();menu();break;
           case
2:sentido_x=1;sentido_y=2;tipo_de_arco=1;pedir_radio()
;hacer_arco();reporte_de_pasos();break;
           case
3:sentido_x=2;sentido_y=2;tipo_de_arco=2;pedir_radio()
;hacer_arco();reporte_de_pasos();break;
           case
4:sentido_x=2;sentido_y=1;tipo_de_arco=1;pedir_radio()
;hacer_arco();reporte_de_pasos();break;
           case
5:sentido_x=1;sentido_y=1;tipo_de_arco=2;pedir_radio()
;hacer_arco();

          sentido_x=1;sentido_y=2;tipo_de_arco=1;hace
r_arco();

          sentido_x=2;sentido_y=2;tipo_de_arco=2;hace
r_arco();

          sentido_x=2;sentido_y=1;tipo_de_arco=1;hace
r_arco();reporte_de_pasos();break;
           case
6:sentido_x=1;sentido_y=2;tipo_de_arco=2;pedir_radio()
;hacer_arco();reporte_de_pasos();menu();break;
           case
7:sentido_x=1;sentido_y=1;tipo_de_arco=1;pedir_radio()
;hacer_arco();reporte_de_pasos();break;
           case
8:sentido_x=2;sentido_y=1;tipo_de_arco=2;pedir_radio()
;hacer_arco();reporte_de_pasos();break;
           case
9:sentido_x=2;sentido_y=2;tipo_de_arco=1;pedir_radio()
;hacer_arco();reporte_de_pasos();break;
           case
10:sentido_x=1;sentido_y=2;tipo_de_arco=2;pedir_radio(
);hacer_arco();

sentido_x=1;sentido_y=1;tipo_de_arco=1;hacer_arco();

sentido_x=2;sentido_y=1;tipo_de_arco=2;hacer_arco();

sentido_x=2;sentido_y=2;tipo_de_arco=1;hacer_arco();r
eporte_de_pasos();break;
           case 11:break;
           default: printf("S¢lo tenemos estas once
opciones\n");pedir_tipo_de_arco();break;
          }
}
```