Escuela Superior Politécnica del Litoral

Facultad de Ingeniería en Electricidad y Computacidn

"Diseño e implementación de un sistema de adquisición de datos y generación de señales de control con monitoreo gráfico para una bancada de prueba de motores eléctricos"

Tesis de grado

Previa a la obtención del Título de :

Ingeniero en Electricidad

Especialización:Industrial

Presentada por:

Edgar **Wilfrido** *Cañar* Vargas

Luis Eduardo **Baque** Realpe

Guayaquil - Ecuador

1998

AGRADECIMIENTO

Al Ing. Damián Alberto Larco Gómez, por su valiosa colaboración y guia en el presente trabajo.

A la ESPOL por la facilidad de acceder a los diferentes instrumentos utilizados en este trabajo

Y a todas las personas que colaboraron de u otra manera para la culminación de este trabajo

DEDICATORIA

Edgar Cañar V.

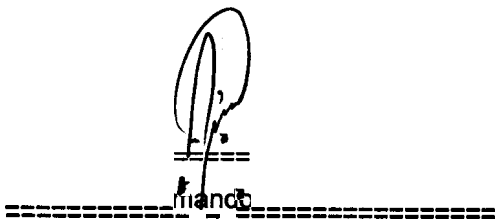
A mis padres, por su apoyo incondicional en proporcionarme los medios necesarios para culminar todos mis estudios.

A mi esposa e hijos, que supieron esperar y que me apoyaron en todo momento.

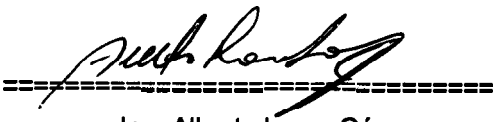
Luis Baque R.

A mis padres, que me apoyaron en todo momento.

TRIBUNAL DE GRADO



Ing. Arm ndo Altamirano
Presidente del Tribunal



Ing. Alberto Larco Gómez
Director de Tesis

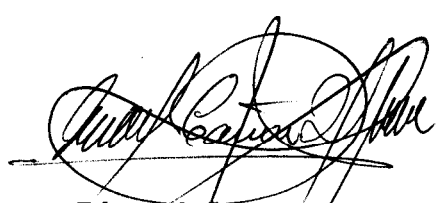
Ing. Norman Chootong
Miembro del Tribunal

Ing. Dennys Cortez
Miembro del Tribunal

DECLARACIÓN EXPRESA

"La responsabilidad por los hechos, ideas y doctrina expuestos en esta tesis, me corresponden exclusivamente; y el patrimonio intelectual a la ESCUELA SUPERIOR POLITECNICA DEL LITORAL".

(Reglamentos de Exámenes y Titulos profesionales de la ESPOL).



Edgar Cañar Vargas

Luis Baque Realpe

RESUMEN

El siguiente trabajo es un prototipo de interface para adquirir datos e interactuar con algún equip0 externo, con el objeto de visualizar sus par&metros fisicos y manipular dicho equipo a través de un computador.

El sistema implementado comprende un módulo que nos ayuda a establecer la comunicación entre dos tipos de elementos: eléctrico (motor de corriente continua) y digital (computador).

Para el funcionamiento del sistema que maneja el motor, se utilizan otros equipos del laboratorio de Electronica de Potencia: estos equipos forman lo que se conoce como bancada de prueba de motores eléctricos ellos son: el controlador de velocidad MV 4200, el medidor digital de torque, velocidad y freno de corrientes de Eddy MV 1045 y los tacometros AC y DC MV 1025 y MV 1024 respectivamente, todos ellos se los conecta para una regulación de velocidad por tacómetro, la cual se encuentra especificada en los manuales de manejo de estos equipos experimentales.

El controlador de velocidad recibe una señal externa proveniente del computador, la que previamente es acondicionada por nuestra interface, y produce un cambio de velocidad en el motor.

La unidad MV 1045, se emplea para tres iunciones básicas que son: la visualización digital de los par&metros de velocidad y torque, para sensar la señal de torque y para variar este mismo par&metro en el motor.

El computador produce una variación de torque de carga a traves de esta unidad enviando una seiial, que previamente es acondicionada por el módulo a realizar; esta variación es sensada a traves de esta misma unidad y devuelta hacia el computador, previamente tratada en el módulo de interface. Los tacometros son empleados para realimentación del controlador de velocidad y para sensar la velocidad.

El módulo de adquisición y envio de datos, que es el dispositivo a diseiiar. En este módulo existen dos tarjetas electronicas: una maneja las señales de fuerza que recibe del motor y las acopla para que las reciba la tarjeta digital, que es donde se encuentran todos los integrados que ayudan en el control al computador.

Por último, el computador es el equipo que coordina las acciones de los demás componentes mediante la ejecución de un programa elaborado en "C++" bajo DOS. Parte de este componente es el software o programa que sirve de interfaz entre el operador y el motor DC.

INDICE GENERAL

INDICE DE FIGURAS

INDICE DE TABLAS

INDICE DE TABLAS

INTRODUCCIÓN

En el mundo moderno se construyen sofisticados sistemas de control, muchos de los cuales son modificaciones de anteriores diseños. Con el avance de la tecnologia y la simplicidad en muchos de los casos para renovar este tipo de sistemas, hace posible mejores y novedosos sistemas de control.

Esta tesis es consecuencia precisamente de estos tipos de innovación. Cuando buscamos un tema para nuestra tesis se nos ocurrió controlar un motor de corriente directa (motor DC) a través de un computador. Pasamos meses investigando los detalles de la comunicación externa con el puerto paralelo; esencia de nuestro proyecto de tesis. En Internet consultamos horas y horas, los textos sobre el tema sólo hacian una referencia muy superficial.

Por otro lado, existen en nuestro medio diferentes maneras de realizar un seguimiento a las reacciones de un motor eléctrico: multimetros, osciloscopios, etc. Estos equipos "examinan" al motor y proporcionan información para la toma de decisiones con respecto de su funcionamiento. Los sistemas automáticos de control se limitaban a realizar operaciones ciclicas en respuestas a par&metros externos. Si queremos intervenir en la rutina programada de algún mecanismo, deberiamos "reprogramarlo". Los sistemas de control han ido evolucionando y la asistencia de un computador se ha hecho

indispensable; más adn, la asistencia de toda una red de computadores constituye parte fundamental de Cstos. El diseño de softwares dirigidos hacia ese tipo de controles es bastante complejo, y el factor económico hace inalcanzable dicha tecnologia en nuestro medio.

El propósito de este trabajo es supervisar: el voltaje y la corriente de armadura, la velocidad y el torque de carga; señales caracteristicas del motor DC. Estas señales son respuestas a otras señales que se envian desde el computador hacia el motor, a traves del *Módulo de Adquisición y Envio de Datos.*

En este trabajo se ha desarrollado un sistema de control en lazo abierto. Un motor DC es seleccionado como objeto de control, Un computador es seleccionado como objeto supervisorio 0 control. El control es realizado a traves del operador del computador, el cual, modifica la velocidad o carga del mismo a libre voluntad.

El proyecto se maneja a traves del computador, en nuestro caso es el controlador del sistema. Encendiendo el motor a una cierta velocidad, obtenemos inmediatamente su respuesta en el monitor. Podemos observar: la señal de voltaje de armadura, la señal de la corriente del motor, la señal de velocidad del motor y la seiial de torque de carga, una por una o todas ellas a la vez. Una vez iniciado el proceso, el operador puede manejar la velocidad del motor y a su vez puede aplicarle carga al mismo, observando en el monitor las señales de respuesta antes citadas.

CAPITULO I

CONCEPTOS Y FUNDAMENTOS PARA EL DISEÑO DEL SISTEMA

1.1 Generalidades

En este capitulo se analizan los fundamentos en los cuales se basa este trabajo, incluyendo las pruebas realizadas en sistemas que no fueron considerados al momento de culminar el proyecto.

El sistema de adquisición y envio de datos se basa en sistemas de control comunes, se hace una pequeña reseiia de su naturaleza entre los cuales tenemos los sistemas de lazo abierto y sistemas de lazo cerrado. Así mismo, definimos cada uno de los elementos que componen el sistema.

Uno de los componentes del sistema más importantes es el computador, se enfatiza en la información obtenida a través de Internet, se explica los diferentes periféricos de éste. Se hace una descripción detallada de los modos de operación del puerto paralelo, continuando con una explicación de su estructura fisica y lógica.

Por último se exponen los criterios por los que se utiliza el lenguaje C++ y porque no se utiliza Visual Basic en el control de las señales que maneja el computador.

1.2 Principios básicos de la adquisición y conversión de datos

1.2.1 Sistemas de Control

El ser humano es un sistema de control perfecto, lo mismo se puede afirmar de los animales y de los demds seres vivos. Nuestro método de acción locomotriz es simple, primero nos proporcionamos una meta, por ejemplo: cruzar un rio usando un tronco de árbol como un puente, luego recurrimos a la ley de control que nos permita cumplir el objetivo (en este caso la conservación del equilibrio), y por último, se realiza el proceso. A medida que damos un paso detectamos la ubicación del cuerpo con respecto al centro de gravedad y de manera automatica, generamos una nueva posición de brazos, tronco y piernas para guardar el equilibrio. Actuar, medir y compensar para cumplir una referencia es la estrategia general de los métodos de control presentes en la naturaleza.

En la gran mayoria de casos, las máquinas que el hombre ha desarrollado usan medios electrónicos para implementar un sistema de control automatic0 similar al que se encuentra en el comportamiento fisico de los seres. El control implica la

generación de una señal de salida en respuesta a una señal de entrada al sistema. Este proceso puede ser en "Lazo abierto" o en "Lazo cerrado". Apagar el calentador del agua de manera automdtica durante el dia y encenderlo a las doce de la noche, es un ejemplo de un control en **lazo abierto.**

Un esquema de control en lazo cerrado tradicional (Fig. 1), es el tipo de sistema más usado para el control de algdn fenómeno. En éste control lo ejercen los circuitos electrónicos a través de los diferentes sensores disponibles.

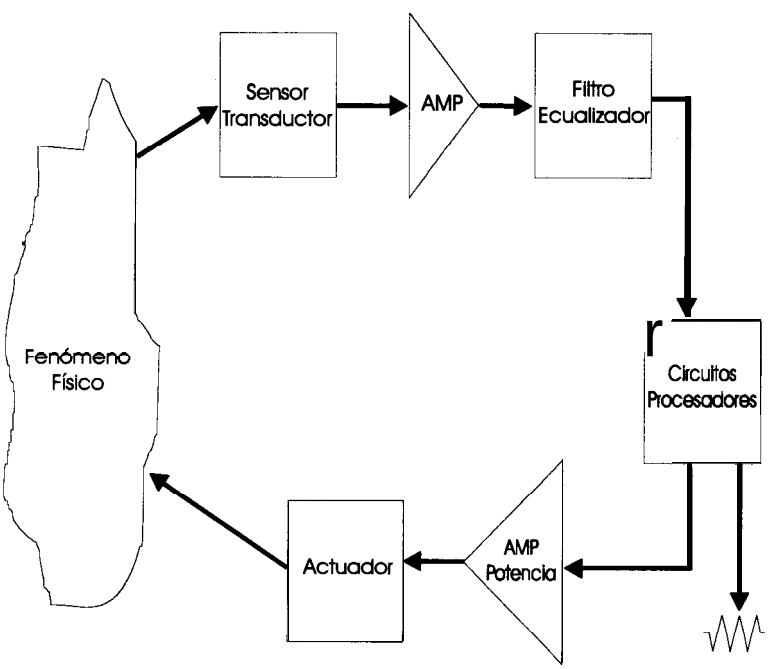


Figura 1 Sistema de Control Tradicional

Esta proyecco comprende el diseño de un sistema de control de lazo abierto, como se muestra en la Fig. 2. En éste sistema se va a controiar un motor de corriente directa, mediante un proceso en lazo abierto. Nuestro controlador es un computador standard, por el cual, mediante sensores correspondientes supervisamos las señales de cambios del motor. Estos cambios son provocados por dos señales controladoras: variando el voltaje de armadura y la corriente de carga de línea.

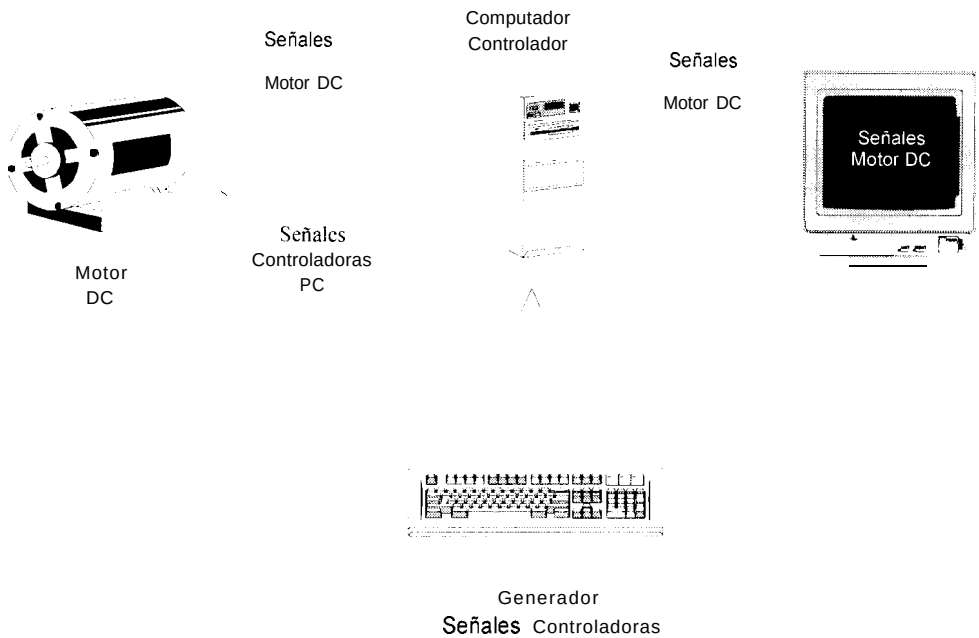


Figura 2 Sistema de Control en lazo abierto

1.2.2 Sistema de Adquisición de Datos

La aplicación de métodos digitales, al procesamiento de sefiales análogas es una práctica muy común en todos los

campos de influencia de la electronica. Su introducción a posibilitado los modernos sistemas de reproducción de sonido, el video y la telefonia digitales, el control de procesos por computador, la instrumentación digital, los juegos de video, el sistema de satélite digital (*DSS*) y otras tecnologias que han marcado nuestra época. En todos estos casos, la transición del mundo análogo al digital la proporciona invariablemente algún tipo de sistema de adquisicion y conversion de datos.

Los sistemas de adquisicion de datos, como su nombre lo indica, adquieren señales análogas de una o varias fuentes y las convierten en una secuencia de datos o códigos digitales, cada uno de los cuales representa el valor particular de esas seiiales en un instante dado. Esta conversion es esencial, siempre que se utilicen tecnicas digitales confiables para realizar un trabajo normalmente análogo, 0 se recurra a un procedimiento digital (computador, microprocesador, microcontrolador, etc.) para controlar un experiment0 0 proceso. Una vez digitalizada, es decir convertida en una colección de bits, una señal análoga se vuelve prácticamente inmune al ruido y se torna más flexible, pudiendo ser almacenada, analizada y manipulada de muchas formas.

Las señales análogas de entrada provienen, en la mayoria de los casos, de transductores que convierten parametros del

mundo real (presión, temperatura, humedad, flujo, luz, sonido, etc.), en sefiales electricas equivalentes. Una vez digitalizadas estas sefiales son procesadas por dispositivos terminales como computadores, redes de comunicaciones, etc., para ser analizadas, almacenadas y/o transmitidas de alguna forma. La habilidad del sistema de adquisicion de datos para preservar la exactitud e integridad de las sefiales originales, es la principal medida de su calidad.

1.2.3 Elementos Básicos de un Sistema de Adquisición de Datos

Un sistema de adquisicion y conversion de datos (Fig. 3), se compone básicamente de: un *Multiplexor* análogo de entrada, un amplificador de muestreo y retención, un convertidor análogo digital y un circuito de control.

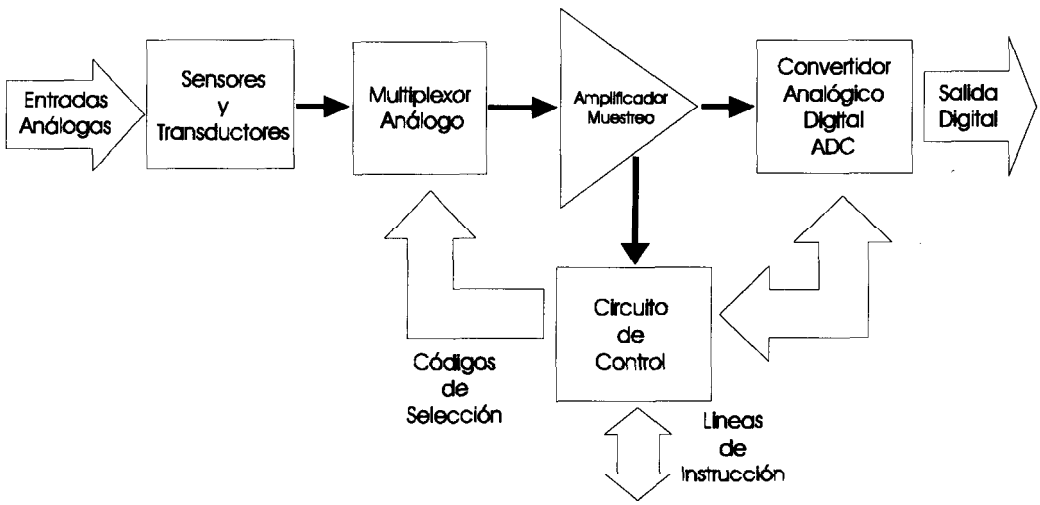


Figura 3 Elementos de **un** Sistema de Adquisicidn de Datos

1.2.3.1 Multiplexor Análogo

El Multiplexor análogo actúa como un interruptor rotatorio de varias posiciones que, periódicamente selecciona una seiial de entrada y la encamina al circuito de muestreo y retención. Este último sigue la seiial análoga seleccionada durante un breve instante y memoriza su amplitud instantánea, ignorándola del resto del tiempo. Este proceso se denomina ***muestreo.***

1.2.3.2 Transductores

Mide el parámetro que se quiere controlar y realiza la transición entre el mundo fisico y eléctrico. Los equivalentes eléctricos que produce un transductor en la entrada de un sistema son: voltaje, corriente, carga, capacitancia y resistencia. Hay una gama muy amplia de transductores con caracteristicas diversas en cuanto al nivel de señales la impedancia de salida y la respuesta de frecuencia.

1.2.3.3 Amplificador

Es un bloque fundamental en todo sistema de control. Su misión es acoplar la señal del transductor a los circuitos posteriores. Por lo general entrega un voltaje para el funcionamiento de los circuitos de procesamiento, si la señal captada es muy débil debe amplificar considerablemente. Si la seiial es muy fuerte, el amplificador escalará la seiial a niveles manejables. El amplificador se encarga además, de realizar el acople de impedancia entre un dispositivo que convierte un par&metro fisico en otro par&metro fisico (transductor) y el sistema de control.

1.2.3.4 Convertidor Analógico-Digital

El convertidor Analógico/Digital, asigna a cada muestra análoga un código o valor digital equivalente. Este último proceso se denomina ***cuantización***. El muestreo y la cuantización son la

1.2.3.5 Circuito de Control

Este circuito suministra las señales lógicas de sincronización y secuencias requeridas en cada instante, además realiza la comunicación con el dispositivo de procesamiento.

Adicionalmente, se pueden requerir filtros y circuitos de acoplamiento, previos a la acción de un dispositivo multilineas (multiplexor) para garantizar que todas las señales de entrada tengan un rango dinámico similar.

En la mayor parte de los bloques funcionales de un sistema de adquisicion y conversion de datos pueden venir integrados en un mismo chip, la práctica más comdn es utilizar tarjetas de adquisicion de datos especializadas. Estos módulos incluyen generalmente a una de las bahias disponibles de un Computador. Su utilización es sencilla, puesto que en ellos el fabricante ha resuelto la mayor parte de los problemas de: ruido, aislamiento, estabilidad, etc., que atentan la eficiencia de la conversion; problemas que debe enfrentarse el diseiiador cuando planea, desarrolla y construye el sistema desde el principio.

1.3 Comunicación de un medio externo con el computador

Para poder comprender los elementos complementarios que acompañan a un computador, se toman como ejemplo los mas conocidos, y entre ellos; los de mayor utilidad son los que utilizan el puerto paralelo. Por su tecnologia actualizada, ahora es un transportador de información de muy alta velocidad.

1.3.1 Periféricos más comunes que se enlazan con el computador

Existen en el mundo real diferentes fenómenos fisicos los cuales podemos comprenderlos a traves de nuestros sentidos. Un *multimetro* lee el voltaje de una bateria, Un barómetro mide la presión de un tanque, en todos estos casos la lectura de las señales se hace a traves de una pantalla digital de dichos instrumentos en caso de ser digitales, y de una pluma indicadora en caso de ser analógicos. En todo caso existen sensores apropiados y un circuito electrónico interno hacen esto posible.

Así mismo existen diferentes maneras de comunicar este tipo de señales a traves de un computador, por ejemplo la comunicación a traves del teclado, rastreadores de imagen, disqueteras, puerto serial, puerto paralelo, etc. (Fig. 4)

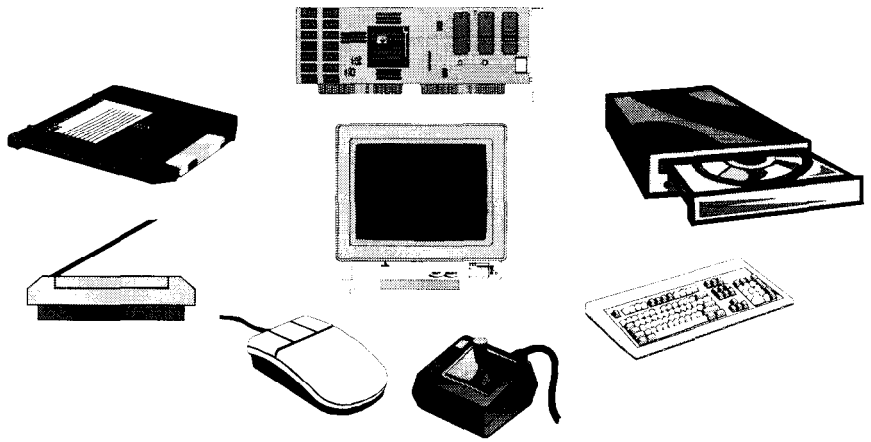


Figura 4 Periféricos más Comunes del computador

La via de comunicación que vamos a emplear ***es*** el ***puerto paralelo.*** Por la facilidad y velocidad en transferencia de datos, puede considerarse el medio más práctico, para llevar información de un parámetro fisico hacia un computador.

1.3.2 Elpuerto paralelo del computador

Existen varios modos de operación del puerto paralelo, ellos van apareciendo uno a uno en el tiempo, mejorando su rendimiento hasta llegar a estandarizar su norma, configuración y funcionamiento.

1.3.2.1 Aspectos fundarnentales del puerto paralelo

Cuando IBM Introduce el computador, en 1981, el puerto paralelo para impresora era incluido como

una alternativa al puerto serie. El puerto paralelo tenia la capacidad de transferir 8 bits de datos a la vez, mientras el puerto serie transmite un bit a la vez. El computador era introducido y las impresoras matriciales eran los principales perifericos que usaron el puerto paralelo. Con el avance de la tecnologia, la necesidad de mejorar la comunicación externa aumentaba, el puerto paralelo se volvió el medio por el cual uno podria conectar perifericos de muy alto rendimiento.

Los problemas enfrentados por diseñadores y clientes de estos perifericos entran en tres categorías. **Primero,** aunque el desempeño del computador aumenta dramaticamente, no hay virtualmente ningiin cambio en la actuación 0 arquitectura de éste. La maxima transferencia de datos que se obtiene con esta arquitectura, está en alrededor de 150 kilobytes por Segundo. **Segundo,** no hay ninguna norma para la interfaz electrica; esto causa muchos problemas al intentar garantizar el funcionamiento en varias plataformas. **Finalmente,** la falta de normas del plan forzó una limitación de distancia de sólo seis pies para los cables externos.

En 1991 hubo una reunion de fabricantes de impresoras para discutir una nueva norma para el control inteligente de impresoras sobre una red. Estos fabricantes incluido Lexmark, IBM, Texas Instruments y otros, form6 la Alianza de la red de impresoras (NPA).

El NPA propuso la creación de un comité elaborador de una nueva norma para un puerto paralelo bidireccional de alta velocidad. Era un requisito que esta nueva norma fuera totalmente compatible con el software del puerto paralelo original, pero aumentaria la capacidad de transferencia de datos a velocidades mayores de 1 megabytes por Segundo.

Casi todos los computadores personales compatibles con IBM tienen un puerto paralelo al cual se tiene acceso por medio de un conector detrás del panel. El propósito original del puerto era el proporcionar un modo de comunicación con la impresora, y sigue siendo su uso más común. Pero los usuarios y quienes desarrollan los nuevos productos han aprendido que las ocho salidas, las cinco entradas y las cuatro líneas bidireccionales del puerto son suficientes para las tareas básicas

de supervision, control, transferencia de datos y otras aplicaciones.

Los circuitos y el conector para el puerto paralelo, generalmente se encuentran en una tarjeta de expansion, aunque algunas veces, especialmente en los computadores portátiles, los circuitos se encuentran en el tablero principal.

El Sistema Operativo de Microsoft se refiere al puerto paralelo como LPT1 (Impresora en Línea # 1), o LPT2 y LPT3 para puertos adicionales. Otro nombre dado al puerto paralelo *es puerto de impresora,* reflejo de su uso más común. El panel conector trasero es un conector hembra de tipo D de 25 *pines* (Fig. 5)(no confundir con el conector macho DB-25 utilizado en muchos puertos seriales).

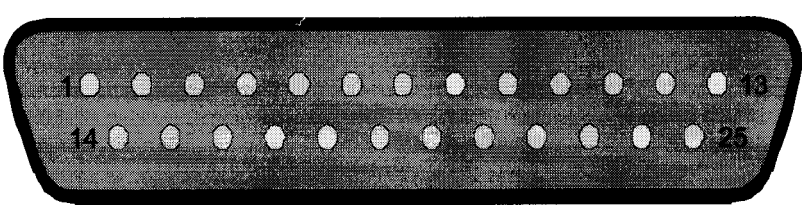


Figura 5 Distribución de los Pines del Puerto Paralelo

Cada puerto paralelo tiene una de tres direcciones base posibles: 378h, 278h o 3BCh. Para conectar la tarjeta de puertos con alguna de estas direcciones, es necesario realizar ajustes según como lo indica

el fabricante; por lo general, la direccih de puerto que se utiliza por omisión es la 378h.

La tabla I resume las señales en el conector del puerto paralelo; estas seiiales siguen más 0 menos la configuración del dispositivo comunicador de la impresora, popularizada por *Centronics* a pesar de que el conector de 25 pines no usa las 36 líneas del dispositivo original. Aunque cada señal tiene un nombre que sugiere una función particular, no necesariamente deben utilizarse para el mismo propósito. Así por ejemplo, este proyecto utiliza el registro de control para leer las seiiales analógicas en lugar del registro de datos.

PIN	SEÑAL	FUNCION	I/O	REGISTRO	BIT	INV
1	-STB	Strobe	I/O	Control	0	Y
2	D0	Bit 0	O	Datos	0	N
3	D1	Bit 1	O	Datos	1	N
4	D2	Bit 2	O	Datos	2	N
5	D3	Bit 3	O	Datos	3	N
6	D4	Bit 4	O	Datos	4	N
7	D5	Bit 5	O	Datos	5	N
8	D6	Bit 6	O	Datos	6	N
9	D7	Bit 7	O	Datos	7	N
10	-ACK	Acknowledge	I	Estado	6	N
11	BSY	Printer Busy	I	Estado	7	Y
12	PE	Paper End	I	Estado	5	N
13	SEL	Printer Selected	I	Estado	4	N
14	-AUTOLF	Automatic Line Feed	I/O	Control	1	Y
15	-ERR	Error	I	Estado	3	N
16	-INIT	Initialice Printer	I/O	Control	2	N
17	-SELIN	Select Printer	I/O	Control	3	Y
18-25	GND	Ground	I			

Tabla I **Descripción** de **los** Pines **del** Puerto Paralelo

Cuando el computador arranca, una rutina BIOS busca un puerto en cada una de las tres direcciones en el orden mencionado anteriormente. Segdn cuantos puertos estén conectados al computador se llamaran LPT1, LPT2, LPT3.

Las interrupciones de hardware que se asocian con estos puertos generalmente son la 5 y 7. Convencionalmente, el LPT1 utiliza la interrupción 7 y LPT2 la interrupción 5.

1.3.2.2 Modos de operación del puerto paralelo

Existen varios modos de operación del puerto paralelo, ellos van apareciendo uno a uno en el tiempo, mejorando su rendimiento hasta llegar a standarizar su norma, configuración Y funcionamiento.

1.3.2.2.1 Modo de operación standard (Nibble)

El modo standard es el modo más comdn. Todos los puertos paralelos standard están provistos de 5 líneas de entrada para información desde el periferico hacia el computador. Esta información es

llevada a través del *registro de estado* en secuencias de 8 bits (en lotes de 4 bits). Generalmente la quinta línea es utilizada para indicar cual de los dos lotes esta ingresando.

1.3.2.2.2 Modo de operación standard bidireccional (Byte)

El modo standard bidireccional utiliza las líneas de DATOS en ambos sentidos, para enviar y recibir información controlado por un pin de las líneas del registro de control.

1.3.2.2.3 Modo de operación Puerto Paralelo Extendido (EPP)

El modo EPP es el puerto paralelo extendido y su protocolo fue desarrollado originalmente por Intel, Xircom y Zenith Data Systems, como un medio para proveer de alto rendimiento al puerto paralelo y que a la vez fuera compatible con el SPP (Puerto Paralelo Standard).

1.3.2.2.4 Modo de operación Puerto de Capacidad Extendida (ECP)

El modo ECP es el puerto de capacidad extendida, su protocolo fue diseriado por Hewlett Packard y Microsoft como un modo avanzado de comunicación con perifericos tipo impresoras y escáneres.

1.3.2.3 Estructura del Puerto Paralelo

El puerto paralelo se divide en tres partes, entendidas ellas en dos puntos de vista: el fisico, que es en sí el puerto paralelo fisico, en nuestro caso el conector DB-25; y el lógico, que son los registros que utiliza en memoria el computador para procesar la información almacenada en ellos.

1.3.2.3.1 Normas para el uso del puerto

Para el manejo del puerto paralelo se ha cambiado ciertas formas tradicionales de uso. El puerto paralelo fisico está compuesto de tres partes: Líneas de Datos, que aqui llamaremos "Puerto de Datos"; Líneas de Estado, que aqui

llamaremos "Puerto de Estado"; y Líneas de Control, que aqui llamaremos "Puerto de Control". Así mismo tenemos el puerto paralelo virtual, que es el concept0 lógico que tiene el Sistema Operativo del puerto paralelo. Este se divide en tres partes: Registro de Datos, Registro de Estado y Registro de Control, cada uno es equivalente con su parte fisica.

1.3.2.3.2 Líneas de Datos

Las líneas de datos desde DO hasta D7, son ocho bits de salidas que llevan información para que la impresora los procese (Fig. 6). En caso de otras aplicaciones, se pueden utilizar las líneas de datos como salidas generales.

Para controlar el estado de los pines del 2 al 9 en el conector paralelo, solo hay que escribir los datos deseados en el registro de datos, cuya dirección es la dirección base del puerto. Por ejemplo para dar un alto lógico desde D4 hasta D7 y un bajo lógico desde DO hasta D3, se

escribiria F0h (11110000) en el registro de datos.

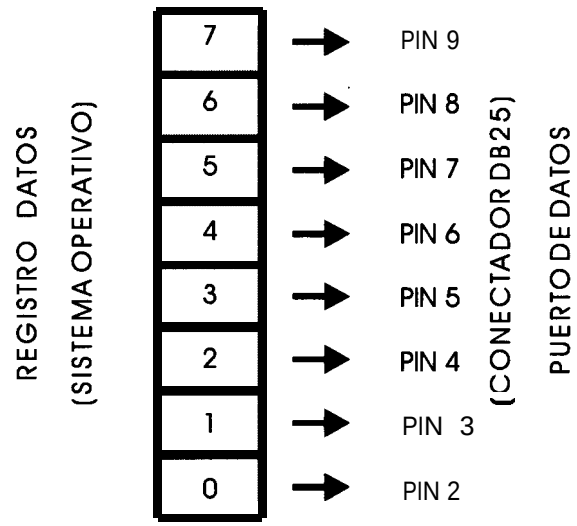


Figura 6 Distribución de los Bits del Registro de Datos del Puerto Paralelo

Algunos puertos tienen líneas de datos bidireccionales, que pueden ser utilizadas como entradas o salidas. Estos puertos son los EPP, ECP y BYTE como se explicó anteriormente.

1.3.2.3.3 Líneas de Estado

Las líneas de Estado son cinco entradas que se leen en un registro de estado, el cual se localiza en la dirección de Base del puerto + 1, por ejemplo: 379h para una dirección de base de puerto 378h. El

registro de Estado es solo de lectura, si escribimos sobre él no le afecta en modo alguno. Las cinco líneas de estado usan los bits del 3 al 7 en el registro (Fig. 7), correspondientes a los pines del 10 al 13 y el 15 en el conector (Fig. 5). Los bits 0, 1 y 2 no se utilizan. Para leer el estado de las entradas, se lee el puerto de estado. Sin embargo, el valor que se lee no es exactamente igual a los estados 16gicos del conector. Los bits en el registro de estado son iguales a los estados lógicos de sus pines correspondientes. Los bits del 3 al 6 se leen normalmente. Sin embargo, el puerto de estado contiene en complement0 el estado lógico del pin 11 del puerto paralelo *(printer* busy Tabla I). Por lo tanto, para encontrar el estado lógico actual del conector, se complementa, o se invierte el bit 7.

El operador de boole "or" exclusivo (XOR) representa una manera fácil de invertir uno o más bits en un byte, sin alterar los demás bits. Tal como lo muestra la

siguiente tabla de verdad, el resultado de una operación XOR es 1 sólo cuando las entradas están conformadas por un 1 y un 0:

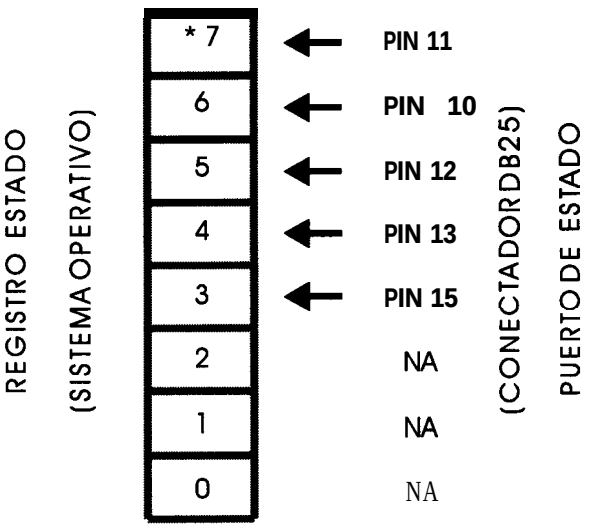


Figura 7 Distribución de los Bits del Registro de Estado del Puerto Paralelo (* Señal Invertida)

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Tabla II Tabla de Verdad del Operador Lógico XOR

Para invertir los bits seleccionados en un byte, primero se debe crear un byte máscara en el cual los bits que se desea invertir son "unos" y los que quiere

ignorar son "ceros". Por ejemplo, para invertir el bit 7, el byte mascara seria 10000000 u 80h. Si se hace la operación XOR de este byte con el byte leido del registro de estado, se obtiene el valor actual del conector. El byte mascara recibe su nombre por que los ceros enmascaran, o esconden, los bits que no se desean cambiar. He aqui un ejemplo:

1 0 1 0 1 X X X

Entradas del puerto de estado, bits del 3 al 7, presentes en el conector del puerto paralelo. En este caso lo que hay en X no interesa.

0 0 1 0 1 X X X

Resultado cuando se lee el registro de estado directamente sin mascara. Notase que el bit 7 está invertido.

1 0 0 0 0 0 0 0

Se enmascara un byte para que el bit 7 coincida con el conector.

1 0 1 0 1 X X X

1.3.2.3.4 Líneas de control

Además de los puertos de datos y estado, el puerto paralelo contiene un puerto de control que es bidireccional. Se pueden usar sus cuatro líneas como entradas o como salidas, en cualquier combinación. La dirección del registro de control es la base +2, o 37Ah, para puertos con una dirección de base 378h.

Las cuatro líneas de control utilizan los bits del 0 al 3 en el registro, que corresponde a los pines 1, 14, 16 y 17 en el conector (Fig. 8). Así como el puerto estado, el puerto de control tiene bits invertidos. En el conector, los bits 0, 1 y 3 son los complementos de los estados lógicos en el registro de control; sólo el bit 2 se leen normalmente. De tal manera que al escribir 0Fh en el registro de control, en realidad le está escribiendo 04h al conector. Para hacer que el valor que escriba coincida con el

resultado en el conector, realice la operación XOR con el valor 0Bh (00001011) .

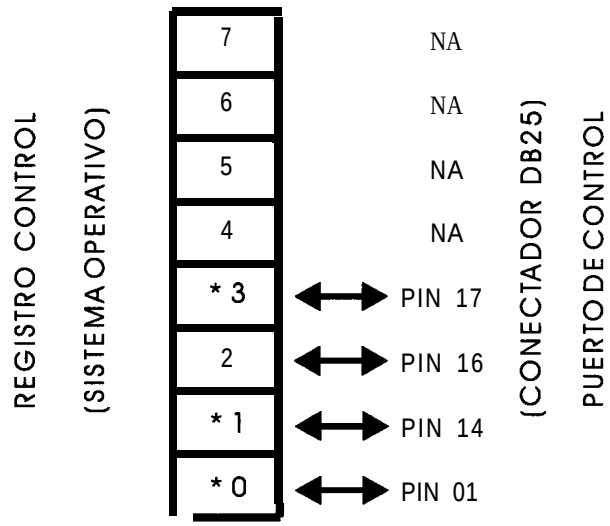


Figura 8 Distribucidn de los Bits del Registro de Control del Puerto Paralelo (* Señales Invertidas)

Tambien se pueden usar las líneas de control como entradas. En los primeros disefios, las salidas de los puertos de control eran salidas inversoras de colector abierto 7405 con resistencias de *4.7* KΩ[i]. Las salidas tambien conectan los *buffers* de entrada. Cuando las salidas de colector abierto son altas, se puede manejar a las entradas de *buffer* con otras salidas digitales. En los últimos diseiios de puertos, los componentes varian, pero el puerto deberia mantener

la capacidad de usar un bit de control como entrada cuando la salida es alta.

Si conecta una salida alta a tierra, la corriente fluye a través de la resistencia, pero con una resistencia de 4.1 KΩ, ésta es solo un poco mayor de 1 mA. Por estos pueden unir dos o más salidas de colector abierto, y cualquier entrada baja llevará un bajo a la salida combinada. A esto a menudo se le conoce como una compuerta "OR alambrada" (wired OR), aunque en lógica positiva (alto="1" y bajo="0"), las salidas forman una compuerta AND, con cualquier salida baja haciendo baja la salida combinada.

Las salidas de algunos son similares a los colectores abiertos y al escribir un "1" a una salida le permite usarla como una entrada. Los puertos en los microcontroladores **8051** y 80c51 son ejemplos de este tipo de salidas. Contrastando con los colectores abiertos, la mayoria de los dispositivos tienen un tipo de salida diferente, llamada "polo

doble" (tótem pole), con dos transistores puestos uno encima del otro, como una "pila tot&mica". Cuando la salida es baja, la salida inferior del transistor conduce, de manera similar a la salida de colector abierto. Pero cuando la salida es alta, el transistor superior conduce, creando una de resistencia baja a +5 voltios.

Si une dos salidas "polo doble", cuando una está alta y la otra baja el resultado es una gran cantidad de corriente en las salidas de los transistores a medida que Cstos se disputan el control. El resultado lógico es impredecible. Lo más importante de esto es que las corrientes elevadas pueden destruir los componentes. Las salidas de la mayoria de los dispositivos digitales tienen salidas complementarias que son similares a las salidas "polo doble".

Si une una salida "polo doble" con una salida de colector abierto, todo saldrd bien si la salida de colector abierto

permanece alta. Si se baja y la salida de "polo doble" está alta, el resultado es una corriente excesiva. Entonces, para evitar problemas, use salidas de colector abierto para manejar las salidas de control. Para llevar un alto a las cuatro salidas del puerto de control para usar los bits como entradas, escriba 04h al registro de control.

Debido a que los bits 0, "1" y 3 están invertidos en el conector, se escribe 04h, no 0Fh, para llevar a los bits del 0 al 3 a un alto. El valor leido tiene a los bits 0, 1 y 3 invertidos desde sus estados lógicos en el conector. Para invertirlos nuevamente y exhibir el valor actual en el conector se utiliza la operación XOR:

Dato XOR 0Bh

1.4 Sistema operativo requerido en el sistema

En el momento de tener lista la información a cerca del sistema, el puerto paralelo y los circuitos necesarios para la implementación,

hay que pensar en el software o programa que manejará todo el proceso. El siguiente pas0 era encontrar el lenguaje de programacion ideal para montar el sistema, para lo cual se pens6 en trabajar con Pascal, Fortran, Assembler, Basic, C++. Más aún se pens6 trabajar en Windows 95 con Visual Basic 5.0 o con Visual C++ 4.0.

De las pruebas hechas con los diferentes lenguajes y sistemas operativos empleados, comenzaremos con mencionar aquellos que se descartaron.

1.4.1 Pruebas efectuadas en Visual Basic 5.0 para Windows 95

Se estudió seriamente trabajar sobre Visual C++ 4.0, pero debido a lo complejo del lenguaje, desviamos la atención hacia Visual Basic 5.0.

Se habian elaborado y transportado ciertos módulos diseñados en lenguaje C++ hacia visual Basic 5.0, la facilidad gráfica que ofrecia este lenguaje era buena; pero, cuando probamos la mayor parte del sistema, éste en ciertos instantes no respondia como debia ser. La fase de lectura de datos la realizo muy bien, los datos se almacenaban en el arreglo y se podían luego graficar. Conectamos la parte de escritura de datos hacia los DAC[ii] y entonces el sistema realizaba interrupciones periodicas que no estaban contempladas en el

programa, se pens.6 que era falla del hardware, pero verificando con una "punta lógica", el sistema Windows 95 realizaba chequeos periódicos en su sistema, en el cual estaba incluido el puerto paralelo, entonces sus pines arrojaban información no deseada a los DAC.

Como se puede observar faltó un conocimiento más profundo a cerca del funcionamiento del Windows 95. Por eso se lo descartó.

1.4.2 Pruebas efectuadas en Lenguaje C++ v 3.0 para DOS

Una vez que se conoce como esta estructurado el puerto paralelo, debemos ahora trabajar sobre una plataforma la cual nos asegure el domino del motor sin interrupciones no deseadas.

Primero se trabajó sobre DOS; un sistema operativo elemental, se establecieron y se diseñaron las rutinas básicas de manejo del puerto paralelo. Probamos con C++ 3.0 y los módulos de programación funcionaron bien, se crearon las rutinas básicas de lectura de datos y envio a través de los Convertidores Digital a Analógico.

Cabe mencionar que primero se trabajó en C++ bajo DOS y esas rutinas fueron llevadas a Visual Basic, por los motivos ya mencionados se tuvo que regresar a "C++"

CAPITULO II

DESCRIPCIÓN DE LOS COMPONENTES DEL SISTEMA

2.1 Introducción

Las diferentes maneras de controlar la velocidad de un motor, sean AC[iii] o DC[iv], se las realiza de buena manera a través de equipos hechos a base de semiconductores, como es el caso de nuestra bancada de prueba de motores electricos.

Los equipos que se utilizan en el laboratorio de electronica de potencia, son de tecnologia suiza (marca Terco), y están dispuestos en una bancada para realizar pruebas de control de velocidad a un motor AC o DC; en el proyecto se emplea la regulación de velocidad por tacometro a un motor DC, cuyo control de velocidad se lo realiza conectando los equipos de la bancada para dicha regulación. Estos equipos son: el controlador de velocidad MV 4200[v], el medidor digital de torque, velocidad y freno de corrientes de Eddy MV 1045[vi], el motor DC MV 1006[vii] y los tacometros AC y DC MV 1025[viii] y MV 1024[ix] respectivamente {1}.

La bancada se la hace actuar con el módulo de adquisicion y envio de datos, el cual permite la conversion de seiiales analógicas a

digitales y además la manipulacih de estas señales, para que puedan ingresar al computador a través del puerto paralelo.

En el computador los datos se los emplea para la formación de gráficas en lenguaje de programación Ct+, las que son presentadas en el monitor. Además se puede enviar datos desde el computador, los que sirven para producir cambios en la velocidad y torque en el motor, previamente acondicionados por el módulo de interface.

2.2 Componentes de la bancada de prueba de los motores eléctricos

El sistema empleado para el control de velocidad de un motor DC (Fig. 9) representa un diagrama de conexiones para la regulación por tacómetro, el cual es factible realizar en nuestra bancada de prueba. Observe la colocación de los "puentes" en el controlador de velocidad MV 4200, los cuales son de fundamental importancia en esta configuración.

La bancada de prueba del Laboratorio está constituida por: La máquina DC (motor) MV 1006, el controlador de velocidad MV 4200, el medidor digital y freno de corrientes de Eddy MV 1045 y los tacómetros AC y DC MV 1025 y MV 1024 respectivamente. A continuación se detalla cada uno de estos componentes.

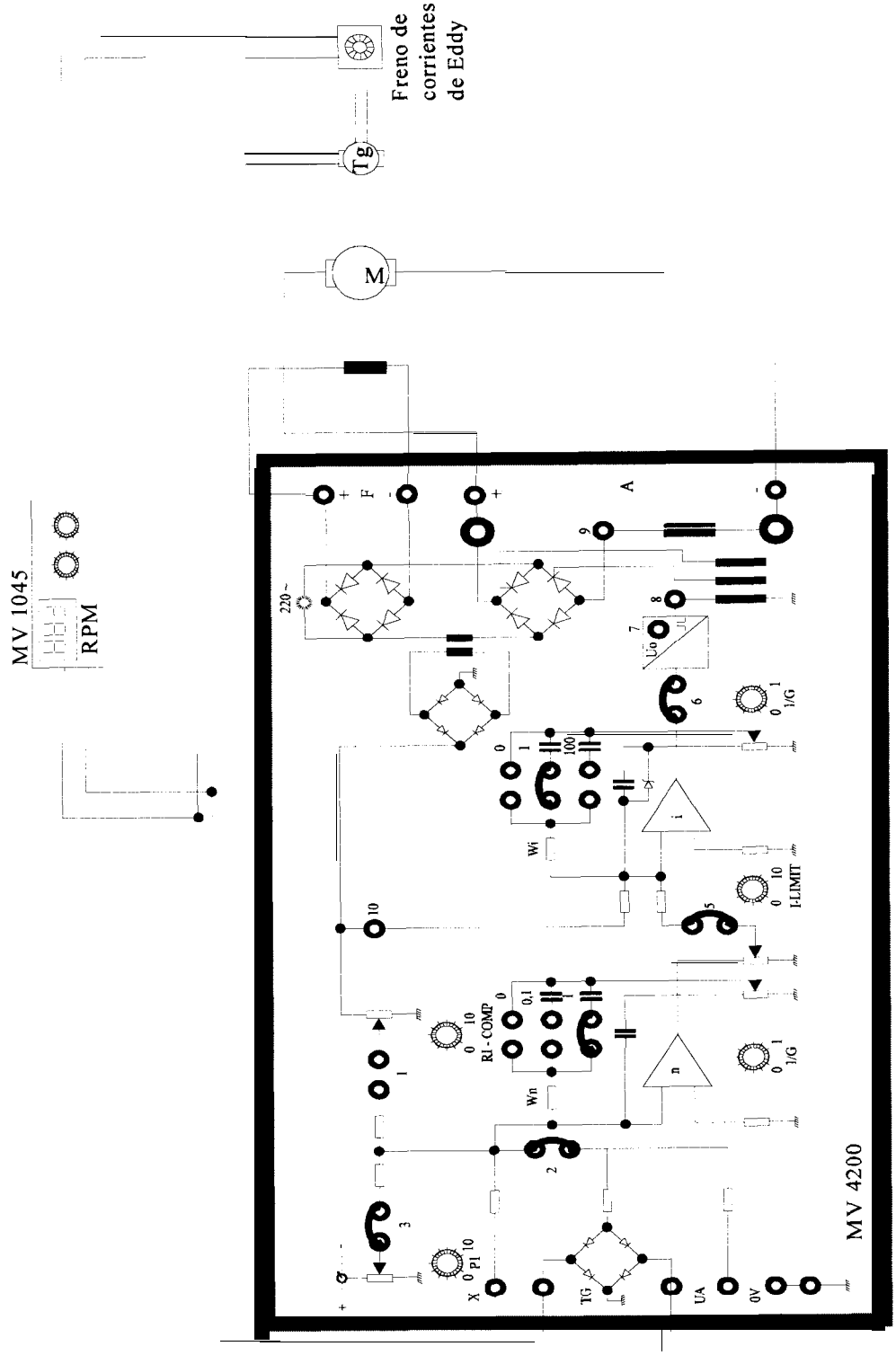


Figura 9 Diagrama de la bancada conectada para la regulación de velocidad por tacómetro

2.2.1 El controlador de velocidad

Este equipo estd diseñado para controlar la velocidad en motores DC, con una potencia de salida no mayor de 1.2 Kilovatios, y tiene una salida de voltaje del rotor o armadura de máximo 220 V_{DC} x. El equipo (Fig. 10) se lo emplea para realizar dos tipos de configuraciones para el control de la velocidad: la regulación por tacómetro y la regulación por voltaje del rotor, ambos casos se realizan variando el voltaje de armadura. Este equip0 utiliza un *control proporcional e integral,* aplicados a la corriente y velocidad de una máquina DC.

Para la regulación de velocidad por tacómetro, empleada en el proyecto, se.escoge una velocidad deseada con el modificador de resistencia (potenciómetro) P1 de la (Fig. 10), entonces el actual valor de la velocidad es medido con el tacómetro AC cuyo voltaje desarrollado es proporcional a la velocidad. {2}

En el regulador "n" (Fig. 10) este voltaje es comparado con el voltaje del *potenciómetro* P1 (Fig. 10), y la diferencia entre ellos es integrada, dando como resultado un voltaje que es empleado en el regulador de corriente "I" (Fig. 10).

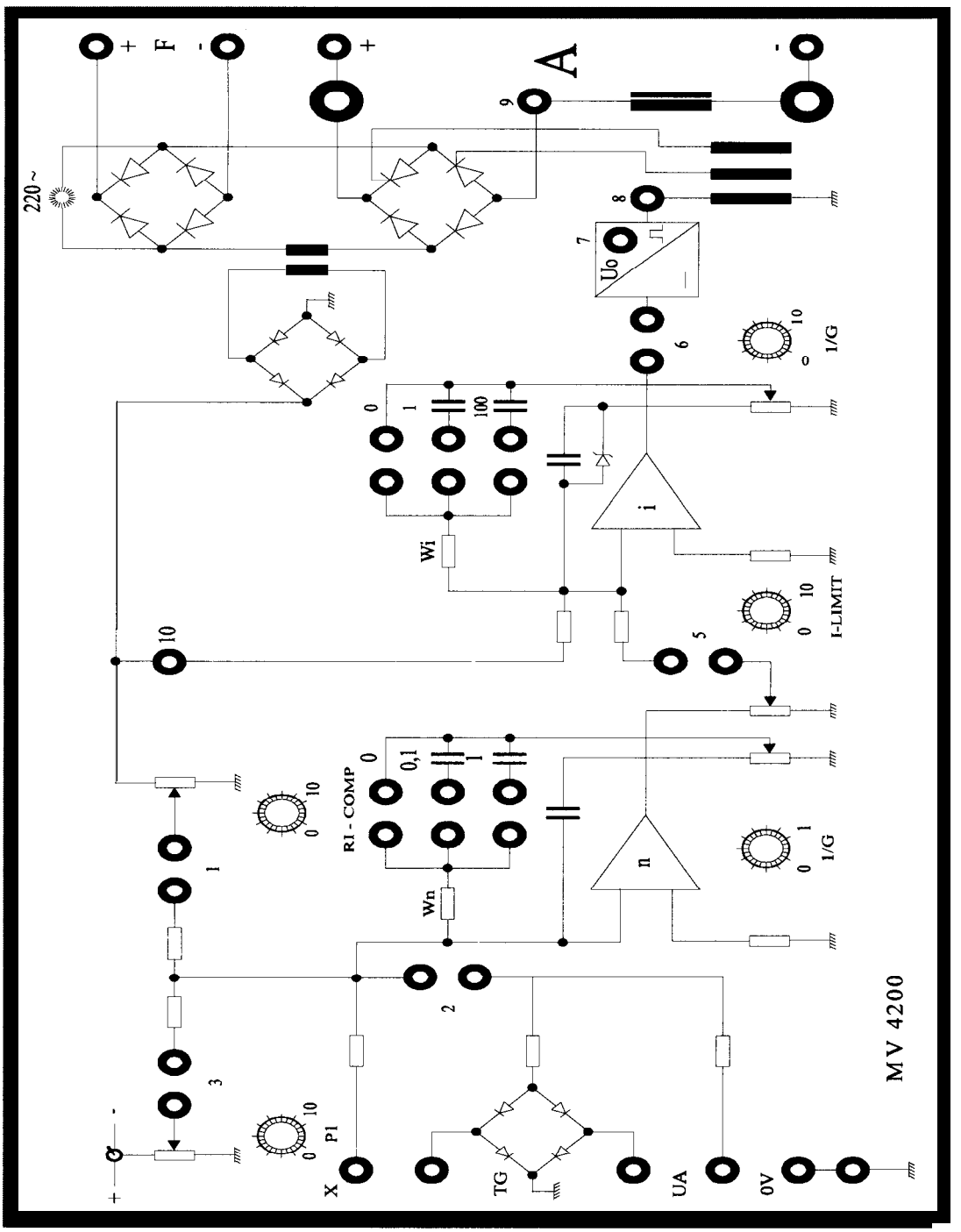


Figura 10 **Diagrama del** controlador de velocidad MV 4200.

Como nos damos cuenta el regulador **n** no controla la velocidad directamente, pero este da un voltaje que sirve como un valor deseado I_{des}[xi] para la corriente del motor.

Un transformador de corriente puesto en serie en la línea de alimentación de los *tiristores,* cuyo secundario desarrolla una corriente la cual es rectificada, y una caida de voltaje a través de una resistencia después del rectificador, representa el valor actual de la corriente I_{act}[xii] .

I_{des} e I_{act} son comparadas en el regulador I (Fig. 10), y la diferencia entre ellos va al generador de pulsos, el cual entrega disparos de pulsos a los *tiristores.* El regulador de corriente funciona, de tal manera que, la corriente del motor será la suficiente para mantener la velocidad del motor al valor deseado.

La ventaja de este arreglo es que se obtiene una limitación automática del motor, y el máximo voltaje de salida puede ser variado con el potencidmetro $I_{lím}$[xiii] (Fig. 10), de tal manera que, se puede seleccionar la corriente dependiendo de la capacidad de trabajo del motor conectado.

La regulación por voltaje del rotor es menos precisa que la regulación por tacómetro, especialmente porque esta depende de la temperatura del motor. Cuando la temperatura del motor

sufre un cambio, la resistencia en el devanado de campo variard, al igual que la corriente de campo, y como la f.e.m.[xiv] inducida depende del campo magnético entonces daria como resultado final un efecto sobre la velocidad.

En cuanto al arreglo de los *tiristores,* podemos ver en el diagrama que se trata de un rectificador puente *semicontrolado* monofásico, donde este arreglo sólo funciona en el primer cuadrante y la corriente es discontinua.

Los terminales de salida del controlador son A (+) y (-), y son los terminales donde se conecta la armadura del motor que se va regular su velocidad, posee además dos terminales de una fuente de 220 V_{DC} F(t) y (-), que sirven para la conexión del campo del motor, éstos terminales se los puede observar en la fig. 10.

2.2.2 El motor de corriente continua

La bancada cuenta con una máquina DC la cual puede trabajar como generador o como motor, Para este último caso puede trabajar como serie o como paralelo de excitación separada, debido que se emplea la configuración de regulación por tacdmetro se escoje el paralelo de excitación separada. Las caracteristicas se detallan a continuación

Motor	2.0 Kw, 1700 r.p.m.
Excitación	220 V_{DC}, 0,5 A
Armadura	220 V 6 A
Máximo Torque	11,7 Nm
Peso	48Kg

Tabla III Datos Caracteristicos del motor

A continuación describiremos los terminales de salida del motor los cuales se los presentan en la fig. 11.

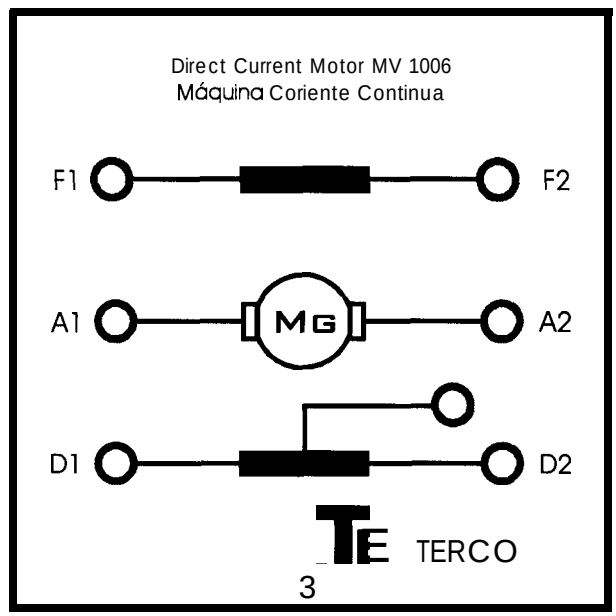


Figura 11 Máquina de corriente continua

Los terminales F1 y F2 (fig. 11), en el motor, son los de alimentación del circuito de campo del motor, y como vemos en las especificaciones debe de ser de 220 V_{DC}

Los terminales Al y A2 (fig 11), en el motor, son los de la armadura y van conectados al puente rectificador semicontrolado del controlador de velocidad.

2.2.3 El medidor digital de velocidad, torque y freno de corrientes de Eddy.

Esta unidad consta de dos partes principales las cuales son:

- Medidor Digital de Torque, Velocidad
- Freno de corriente de Eddy

El medidor digital de torque y velocidad tiene un circuito electrónico digital, para la presentación de una estimación rdpida de estos parámetros en pantallas digitales de 3 digitos. La velocidad es medida en revoluciones por minuto (r.p.m.) y el torque en Newton metro (Nm). El diagrama electrónico se lo ilustra en la Fig. 12. El diagrama de la parte frontal y posterior del dispositivo se muestran en la Fig. 13.

En la parte posterior del dispositivo se encuentran cuatro terminales tipo banana, dos de color rojo y dos de color amarillo, las cuales sirven para el ingreso de la señal de velocidad a través de tacómetros DC y AC respectivamente. La unidad cuenta con una terminal de salida en la parte posterior que proporciona una señal de corriente en un rango de 4-20 mA[xv], que sirve para sensar la velocidad y es utilizada por dispositivos externos.

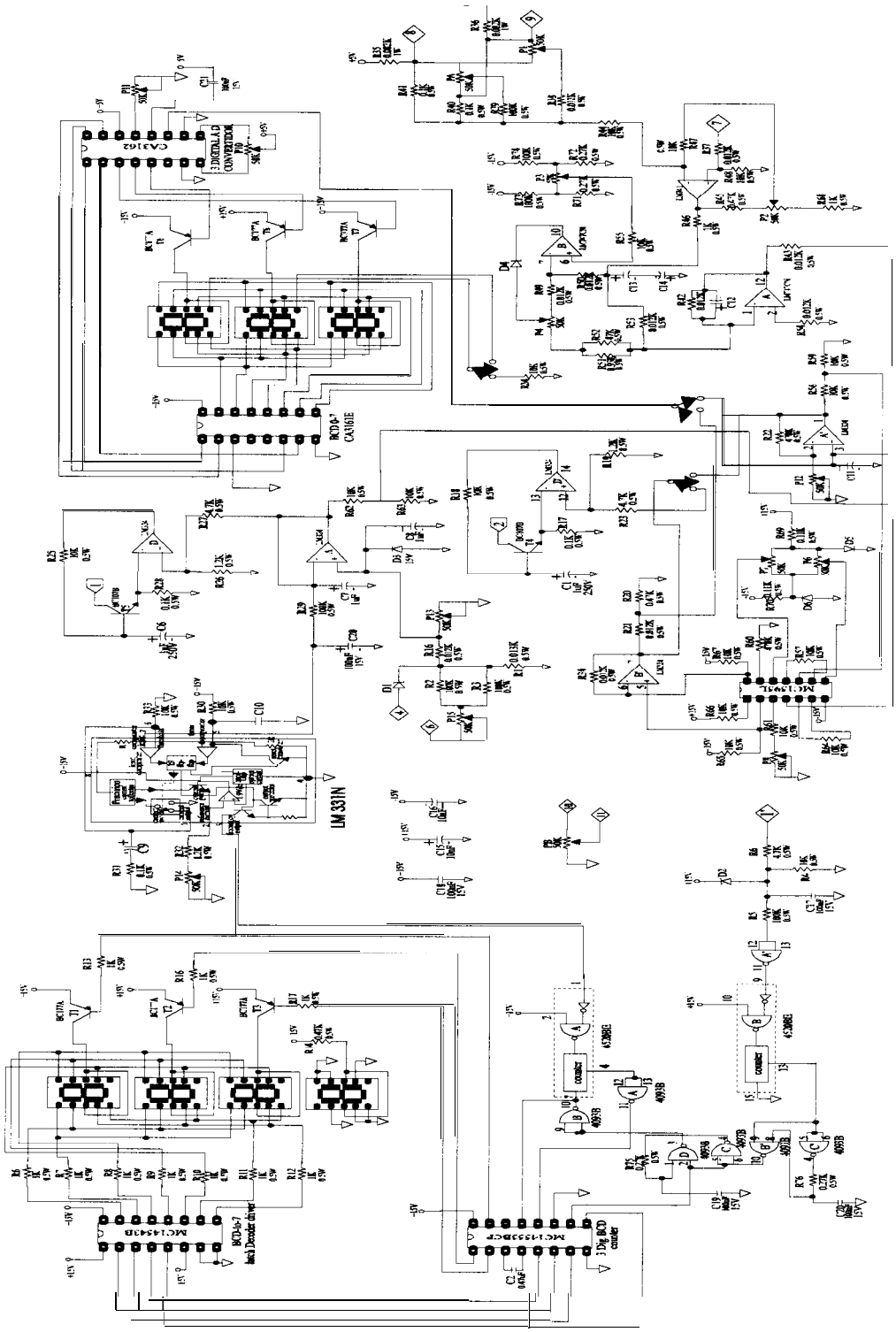
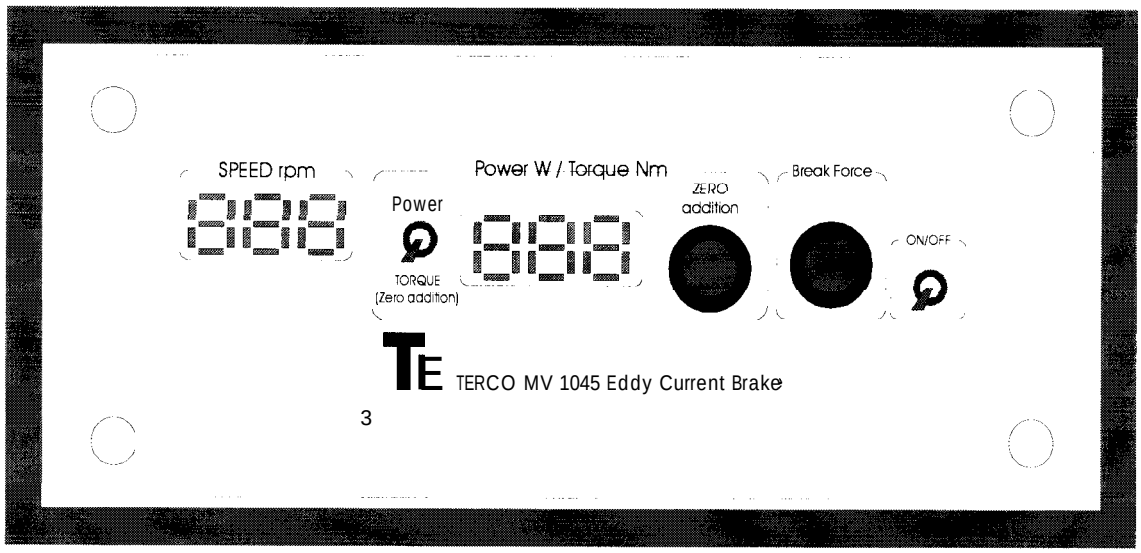


Figura 12 Diagrama electrónico de la sección de visualización y salidas del MV 1045

Vista Frontal



Vista Posterior

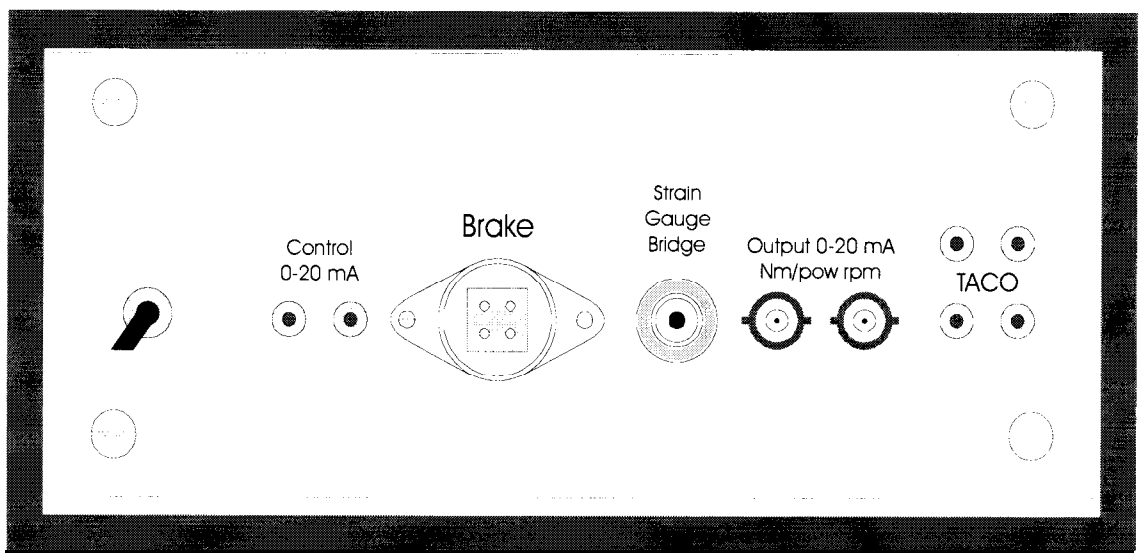


Figura 13 Diagramas de la parte frontal y posterior del MV 1045

La unidad se comunica a través de un terminal cuadrado con un sistema externo que sirve para el freno de corriente de Eddy el cual está acoplado al eje motriz de la bancada.

La carga puede ser manejada manualmente por un *potenciómetro* PB de 50 KΩ ("break force" en la fig.13) que se encuentra en la parte frontal de la unidad (Fig. 13). Aunque también puede ser controlada por una señal externa de corriente en un rango de 0-20mA, la cual ingresa a la unidad por la parte posterior (Fig. 13), a través de dos terminales tipo banana de color rojo y negro.

El manejo de la carga ya sea por el *potenciómetro* PB o por señal externa consiste en variar el dngulo de disparo de un *tiristor* (SCR[xvi]) UT1 (Fig. 14).

Además este sistema tiene una extensión " brazo de torque ", sobre el cual está montado un transductor medidor de deformación (Strain Gauge), que será nuestro sensor de torque de carga, esta serial ingresa a la unidad MV 1045 a través de un terminal redondo en la parte posterior (Fig. 13).

Al igual que con la velocidad la unidad proporciona o devuelve una señal de corriente en un rango de 4-20mA (Fig. 13), que es proporcional a los Newton metro del motor, la cual sirve para sensar el torque y es utilizada en el proyecto para ingresar los datos correspondientes a las variaciones de torque de carga.

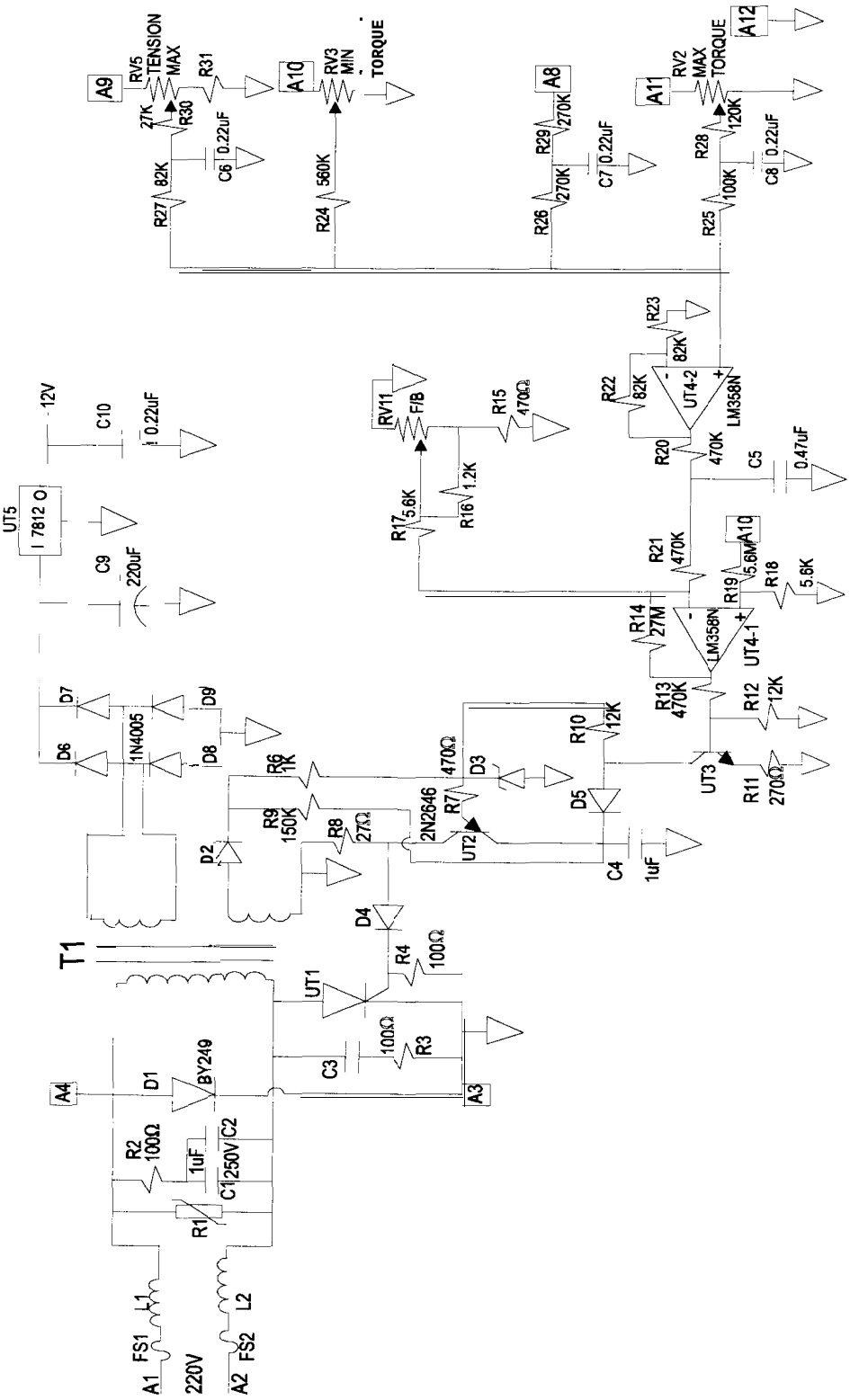


Figura 14 Diagrama de la sección del freno de corrientes de Eddy del MV 1045

2.2.4 El sensor de velocidad

La bancada emplea tacómetros para sensar la velocidad el tacómetro AC (MV 1025) y el tacómetro DC (MV 1024). El primer0 es empleado por el medidor digital de torque velocidad (MV 1045) y por el controlador de velocidad (MV 4200), en la regulación de velocidad.

La unidad MV 1045 utiliza el tacómetro AC como fuente de la señal de velocidad la cual será convertida a digital y presentada en las pantallas digitales de siete segmentos, mientras que el MV 4200 lo utiliza para el lazo cerrado de realimentación, en la configuración para la regulación de velocidad.

El MV 1045 puede utilizar la señal del tacómetro DC o del tacómetro AC, este último nos da una señal de voltaje alterna la cual es proporcional a la velocidad del motor.

El tacómetro DC nos da una señal de voltaje continua que es proporcional a la velocidad del motor, esta seiial es acondicionada para llevarla al computador convertida a un valor digital *(bits)* correspondiente al valor en revoluciones

por minuto (r.p.m.) del motor, la relación de transformación del tacómetro DC es: 20.8 V a 1000 r.p.m.

2.3 El Módulo de adquisicih y envio de datos

El Módulo de adquisicih y envio de datos está compuesto principalmente de dos bloques que son: el bloque de fuerza y el bloque digital.

El bloque de fuerza del módulo es la sección donde las sefiales análogas son acondicionadas para que sean recogidas en rangos óptimos de voltajes por el bloque digital.

El bloque digital es donde se convierten las sefiales análogas a digital, para que puedan ser leidas por el puerto paralelo, esta conversión se trata de que sea con el menor error posible, y a una velocidad que permita barrer la frecuencia de la señal análoga.

Los dos bloques los detallaremos a continuacih:

2.3.1 Bloque de fuerza del módulo de adquisicih y envio de datos

Nuestro bloque de fuerza lo constituye cuatro circuitos acondicionadores, para las sefiales análogas de: voltaje, corriente, velocidad y torque. Un diagrama simplificado de este bloque es presentado a continuacih en la Fig. 15.

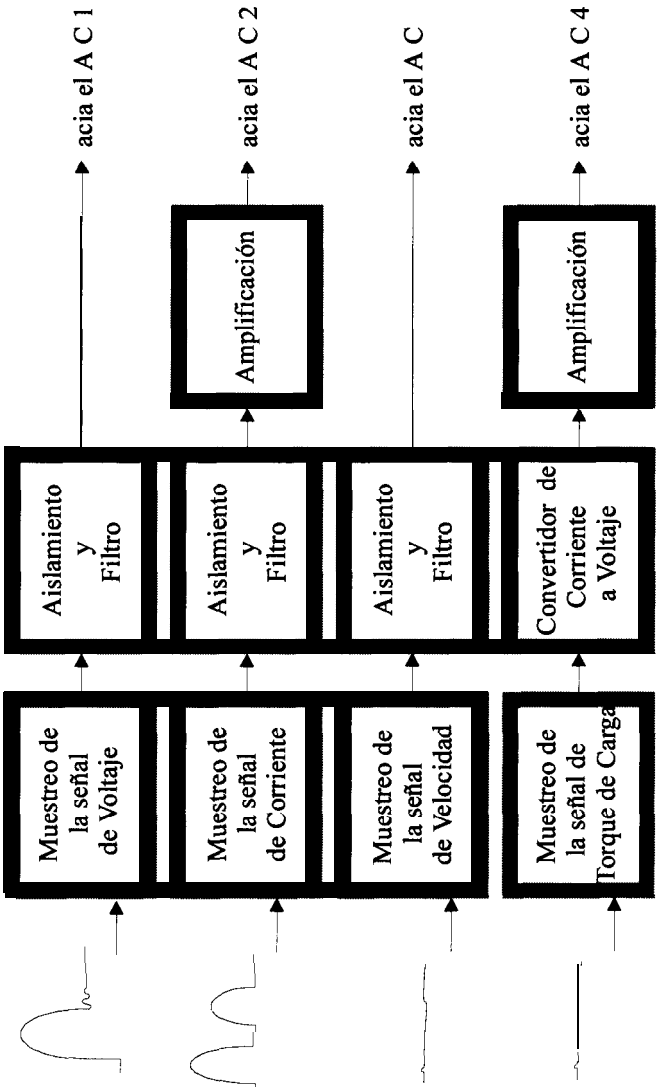


Figura 15 Diagrama simplificado del bloque de fuerza del Módulo de adquisición y envio de datos.

Para el circuito de voltaje, que se lo mide directamente desde los terminales del rotor, se emplea un arreglo de potenciómetros y un amplificador operacional como aislador de impedancia, mientras que para el circuito acondicionador de velocidad, que se la obtiene desde el tacómetro DC, se

utiliza un arreglo de resistencias con un amplificador operacional como aislador de impedancia y un filtro para eliminar el rizado.

Para el circuito sensor de la sefial de corriente se emplea una resistencia en serie con la armadura, la caida de voltaje en dicha resistencia nos dará una fiel copia de la sefial de corriente, además se utiliza un amplificador de muestreo y un filtro activo pasa bajo de segundo orden para evitar las frecuencias altas. Finalmente para el circuito sensor de torque, cuya seiial es sensada por el MV 1045 a través de un transductor medidor de deformación (strain gauge) ubicado en la bancada, dicha sefial analógica de torque se la toma desde una de las salidas del MV 1045 en forma de corriente, la cual es proporcional al torque de carga, como circuit0 acondicionador se utiliza un convertidor de corriente a voltaje. Estos circuitos serán desarrollados más adelante.

En el diagrama de la parte frontal del módulo de adquisición y envio de datos, que se muestra en al fig. 16. Podemos observar los terminales por donde ingresan las señales sensadas en el motor, así vemos que por el terminal V, ingresard la sefial de voltaje del motor; por el terminal I, ingresara la sefial de corriente del motor; por el terminal w, ingresará la señal de velocidad y por último por los terminales Tt y T- ingresara la sefial de torque de carga.

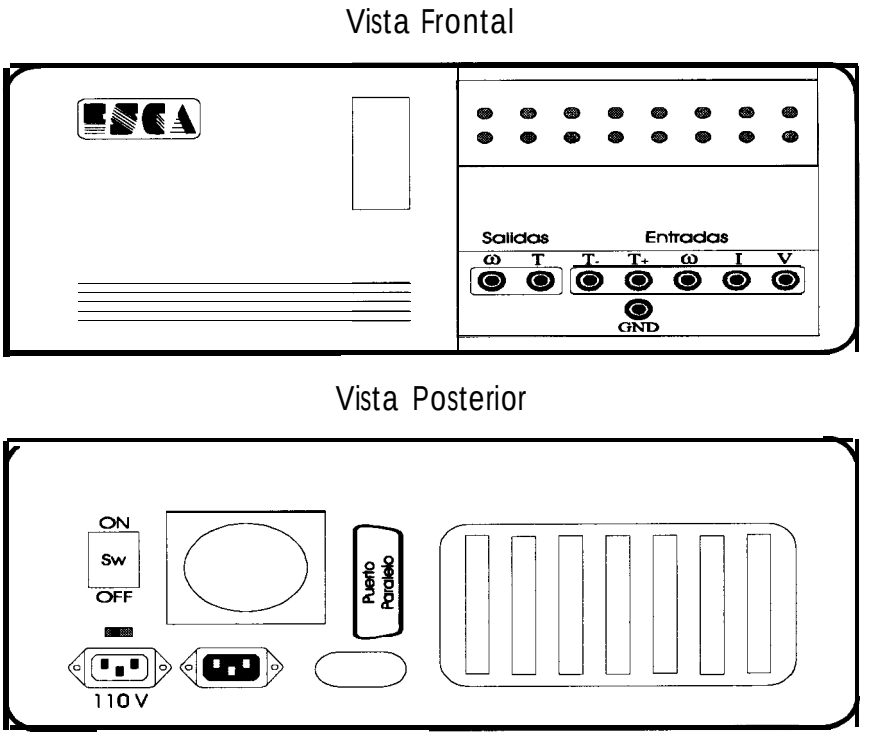


Figura 16 Vista frontal y posterior del módulo de adquisicih y envío de datos.

Cabe anotar que todos los terminales de ingreso de seiiales son de tipo banana y de color rojo.

2.3.2 Bloque digital del Módulo de adquisicih y envío de datos

El bloque digital de este módulo lo conforman dos secciones principales, una para adquirir datos hacia el computador y otro para enviar datos desde el computador, ambos gobernados por unos circuitos de multiplexación de datos desde el puerto paralelo del computador.

El bloque para adquisición de datos digitales lo conforman los convertidores analógico digital (ADC[xvii] 08161) y la multiplexacih de las señales digitales, mientras que el bloque para el envio de las señales de control estd compuesto

por una etapa de multiplexacih, por los convertidores digital analógicos y por dos circuitos acondicionadores de seiial basados en amplificadores operacionales los cuales sirven para acoplar las dos señales análogas, la señal de variación de velocidad y de torque de carga, en el controlador de velocidad (MV 4200) y en el medidor digital de torque, velocidad y freno de corriente de Eddy (MV 1045) respectivamente. Todos los circuitos mencionados en este bloque se estudiardn de manera más amplia en el capitulo III. El diagrama de bloques se lo presenta en la Fig. 17.

La sección digital del módulo de adquisición de datos, tiene su enlace con el computador a través de un terminal Centronic de 36 pines, el cual lo podemos observar en el diagrama de la parte posterior del módulo de adquisición y envio de datos en la fig. 16.

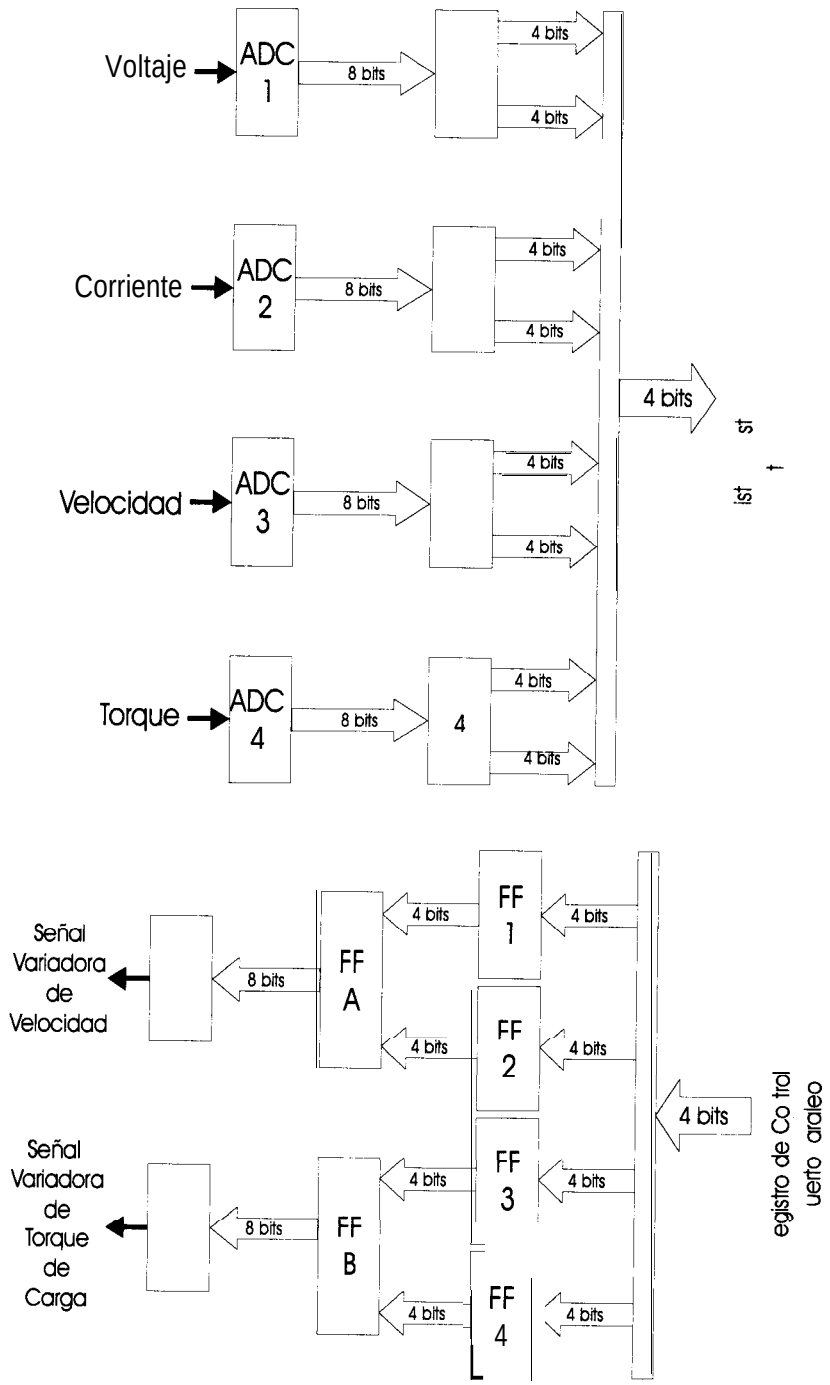


Figura 17 Diagrama simplificado del bloque digital del Módulo de adquisición y envio de datos

2.4 El computador

El computador es el controlador del sistema. Este maneja el ingreso y salida de datos, encargandose de multiplexar sincronizadamente la información en el módulo de adquisicion y envio de datos.

Todo esto se realiza a través del lenguaje de programación C++ version 3.0 bajo D.O.S, y se emplea el modo gráfico del C+t para la representación en el monitor de las señales, como por ejemplo la de voltaje y corriente en la armadura del motor DC.

2.4.1 Ingreso y salida de información a través del puerto paralelo

Como ya se mencionó en el capitulo uno, la función de los diferentes registros del puerto paralelo. El registro de DATOS es utilizado generalmente para la salida de información del computador hacia algún dispositivo, el registro de Estado es utilizado para el ingreso de información desde algún dispositivo hacia el computador y el registro de CONTROL generalmente se usa para controlar el sentido y el flujo de información.

En el proyecto hemos efectuado ciertos cambios para mayor facilidad de comunicación, dominio y enlace con el módulo de adquisicion y envio de datos. El registro Estado (Status)

sigue siendo utilizado para el ingreso de información del módulo hacia el computador.

El ingreso de la información, en este caso el valor digital de las seiiales externas que fueron convertidas a valores binarios determinado por los convertidores Analógico Digital, se lo realiza en dos series de 4 bits cada una; lógicamente se establece un sincronismo para identificar, primer0 el ingreso de los cuatro bits más significativos, y luego los cuatro bits menos significativos. En el momento que ingresan los primeros bits, estos son almacenados en una variable temporal, luego es convertida a variable de ocho bits mediante la operación "&=".

Variable temporal1 &= 0xF0

X X X X 0 0 0 0

Luego ingresan los siguientes 4 bits, que son los menos significativos a otra variable temporal y desplazada su información cuatro bits o sea OOOOXXXX. Una vez que la información fue desplazada a la parte menos significativa de la variable, esta es encerada con la operación:

Después de esto se procede a "unir" las dos variables mediante la operación "OR" exclusiva (XOR "|") y luego se almacena su resultado en el arreglo "T".

Variable temporal2 &= 0x0F

oooooxxx

T[x] [y] = variable temporal1 | variable temporal2

Para la salida de información desde el computador hacia el módulo de adquisicion y envio de datos, se utiliza el registro de control. Los ocho bits se los envía en dos series de cuatro bits, sincronizados los más significativos y los menos significativos. En el puerto paralelo el registro de control tiene un pin negado por lo cual debemos aplicar una mascara de salida para la información; ésta se la realiza con la operación OR exclusiva (XOR), y el operando 0Bh y para el control de entrada y salida hemos utilizado el registro de datos. La razón es muy sencilla, el registro de datos nos proporciona ocho bits de manejo o sea 256 posibilidades de comandos, pero en realidad lo que se necesitaba era sostener las señales habilitadoras de los flip-flop para lo cual utilizamos cuatro de ocho bits, en análisis posteriores concluimos que esta operación era mejor realizarla en la tarjeta del circuit0 del bloque digital del módulo de adquisicion y envio de datos, pues se pierden 128 comandos al mantener inmóviles estos cuatro bits. Los otros cuatro bits nos permiten manejar el multiplexor, que es el que gobierna la secuencia de entrada de los convertidores analógico digital.

Dato XOR 0Bh (00001011)

2.4.2 Software de Control

Debemos recalcar que el control del sistema lo realiza el computador, puesto que de este parte las órdenes de monitoreo a elección del usuario, o el cambio de velocidad y la carga aplicada al motor, así mismo a elección del usuario.

El software que se ha diseñado es el que controla todo el sistema electrónico del módulo de adquisición y envio de datos.

2.4.2.1 Lectura de datos

Este bloque de programa se encarga de la lectura de datos a través del puerto para lo cual utiliza los comandos *inportb y outportb* {3}, comandos que leen o escriben una palabra de ocho bits a traves del puerto.

El formato de los comandos es el siguiente:

Dato =Inportb (dirección del puerto)

En este caso, en dato se almacena la palabra que se encuentra en ese instante en el puerto paralelo.

Como la lectura es hecha a través del puerto de estado, a este llegan solo cuatro bits por lo cual se hace una doble lectura para recoger primero la parte más significativa y luego en la segunda, la parte menos significativa. La dirección del puerto es aquella destinada al puerto de estado, o sea la dirección base t 1.

2.4.2.2 Escritura de datos

En esta parte del programa se utilizan las otras dos componentes del puerto que son las registros de control y datos. Primero ingresa la palabra binaria de ocho bits al bloque, aqui la palabra es dividida en dos partes; luego por el puerto datos (Data) se le indica que salen los cuatro bits más significativos; entonces se envía esta porción de la palabra por el puerto de control. Se indica nuevamente por el puerto de datos (Data) que sale la otra parte, y por el puerto de control sale la segunda parte de la palabra.

Como se mencionó antes esta palabra debe salir enmascarada por el negado por la configuración que lleva en esa parte del puerto.

Outportb (dato, dirección del puerto)

2.4.2.3 Modo gráfico

En realidad lo que el bloque hace es detectar si la tarjeta de video disponible soporta la resolución del monitor que utiliza el programa. Una vez conseguido esto se selecciona el color y otros atributos para los graficos.

En el modo gráfico escogido se establece la resolución del monitor en 800 pixeles de ancho por 600 pixeles de alto

2.4.2.4 Configuración de variables y parámetros

En esta parte se enceran ciertos valores del puerto, e inicia las variables para el control de motor.

2.5 Conexión de los equipos

El sistema completo lo conforman tres partes principales:

La bancada de prueba que consta: del controlador de velocidad; el medidor digital de torque, velocidad y freno de corrientes de Eddy;

los tacómetros AC, DC y el motor DC, todos ellos están conectados para la regulación por tacómetro como se observó en la fig. 9.

El módulo de adquisicion y envio de datos, que es el nexo 0 puente entre la bancada y el computador. En 61 existen los bloques de fuerza y digital que convierten las señales en los dos sentidos: bancada - computador.

El computador que controla el manejo del motor a través del módulo de adquisicion y envio de datos. La conexión de estas tres partes se la puede observar en la fig. 18.

Como vemos el terminal del motor A2 se conectan al puente rectificador semicontrolado del controlador de velocidad A(+) y desde el mismo terminal A2 del motor, se lleva la señal de voltaje hacia el módulo de adquisicion y envio de datos por el terminal de entrada V localizado en su parte frontal (fig. 18).

El terminal Al del motor, se lo conecta con el terminal de entrada I situado en la parte frontal del módulo de adquisicion y envio de datos. El terminal A2, del motor se lo conecta a tierra.

Los terminales F1 y F2 del motor, se los conecta a los terminales F (t) y (-) del controlador de velocidad MV 4200, que son los que alimenta al circuito de campo del motor.

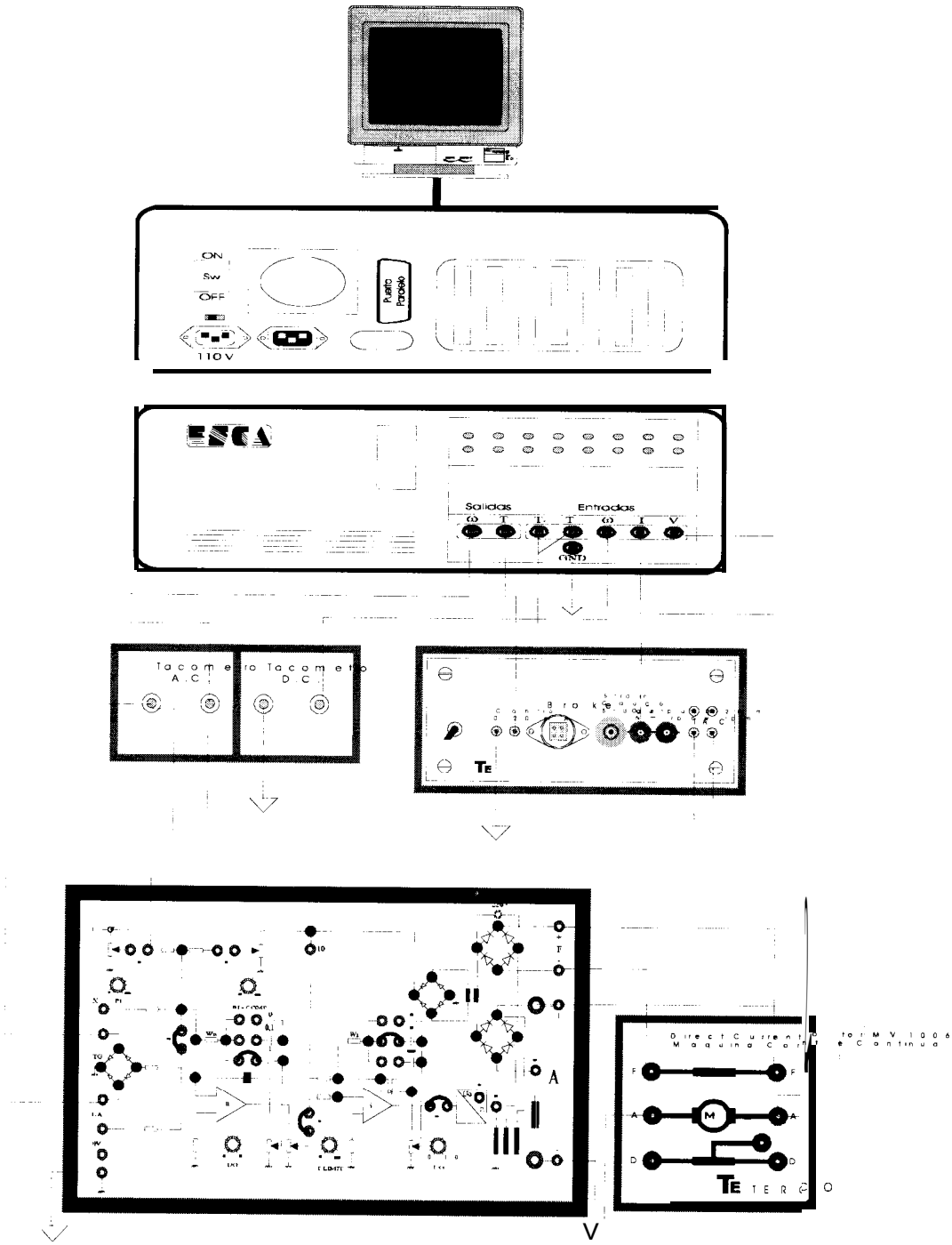


Figura 18 Diagrama de conexión del sistema completo

Los terminales del tacometro DC van conectados uno a tierra y el otro al terminal w en la parte frontal del módulo de adquisicion de datos como se ve en la fig. 18.

Los terminales del tacometro AC se los conecta a los terminales de ingreso para señal de tacometros; tanto en el controlador de velocidad MV 4200, como en el medidor digital de torque, velocidad y freno de corriente de Eddy MV 1045.

En la parte posterior del MV 1045 se encuentra un terminal de salida de O-20 mA que se conecta con los terminales Tt y T- de la parte frontal del módulo de adquisicion y envio de datos.

Los dos terminales de salida del módulo de adquisicion y envio de datos, el W se conecta al controlador de velocidad MV4200 en el terminal 3 del lado derecho, ya que este va a reemplazar la variación manual del potenciómetro por la señal proveniente del computador; y el terminal de salida T se conecta con la entrada de control de O-20 mA que tiene el MV 1045 en su parte posterior, tal como se observa en la fig. 18.

El terminal Centronic que tiene el módulo de adquisicion y envio de datos, sirve para conectar a éste, con el puerto paralelo del computador, por medio de un cable para impresoras.

CAPITULO III

PROCESO DE ADQUISICIÓN DE DATOS Y GENERACIÓN DE SEÑALES CONTROLADORAS

3.1 Introducción

La adquisicih y generación de señales controladoras se la realiza a través del diseño del módulo de adquisicih y envio de datos, el cual se lo implement6 teniendo en cuenta, la disponibilidad de elementos en el mercado para circuitos sencillos de adquisicih de datos, que nos permitan un manejo de espacio en cuanto a tarjetas y que su costo no fuese un obstáculo para la realización del proyecto.

3.2 Circuitos sensores de señales y datos

Los circuitos empleados para sensar y obtener datos sirven para que el computador pueda enlazarse con el componente controlador de velocidad (MV 4200), en el caso de la velocidad y corriente.

Además se enlaza con el componente medidor de velocidad y torque (MV 1045) para el caso del torque y por último con el tacómetro para el caso de la velocidad.

Debido a que el sistema utilizado para captar las seiiales del motor fue escogido de tal manera que, dichas señales fueran sensadas tratando de que exista la menor cantidad de perdidas, por eso se eliminó los elementos que distorsionan y desmejoran la forma real de la sefial como por ejemplo el transformador, la eliminación de estos dispositivos significaba tener que ubicar un solo punto de conexión neutro, es decir unir; el sistema de fuerza de la bancada, el Módulo de adquisición y envio de datos y el computador en un solo punto de conexión de neutro, para esto se tuvo que aislar todo el sistema.

Esto se pudo obtener aislando la alimentación de voltaje con un banco de transformadores de aislamiento, cuyos secundarios darán el suministro de voltaje de 220 V_{AC} al controlador de velocidad (MV 4200) y el medidor digital de torque, velocidad y freno de corrientes de Eddy (MV 1045).

La sefial de voltaje es sensada directamente de las terminales del motor, y escalada sin que sufra ninguna deformación lo cual se observó en la grdfica obtenida por el computador.

La sefial de corriente es obtenida a través de una resistencia conectada en la misma línea del motor, lo que garantiza una sefial prácticamente real en su forma.

Los datos de velocidad son obtenidos por medio de un tacómetro DC ubicado en el eje motor, y son escalados por medio de un divisor de voltaje, de forma semejante a la sefial de voltaje.

Los datos de torque son dados por el medidor digital de torque, velocidad y freno de corrientes de Eddy (MV 1045), a través de una salida de corriente en una de sus terminales de salida, para dispositivos externos ubicados en su parte posterior (Fig. 13).

En esta sección se desarrollan criterios de diseño y construcción, tanto teóricos como experimentales, para los circuitos de captura de señales y envios datos digitales.

3.2.1 Circuito sensor de voltaje

La sefial de voltaje es obtenida directamente de las terminales del motor; esto se realizó de tal manera que la sefial no sufra deformaciones y sea una copia prácticamente exacta de la sefial de voltaje del motor, los circuitos constan de dos etapas: la etapa de muestreo-escalamientq y la etapa de filtro-aislamiento de impedancia. El diagrama del circuito sensor de voltaje se ilustra en la Fig. 19.

El circuito de muestreo es un divisor de voltaje, en el cual la sefial de voltaje se escala para que no supere los valores nominales de entrada del ADC08161. Los potenciómetros fueron

escogidos sabiendo que, el máximo valor de voltaje controlado para la configuracih de regulación por tacómetro es de 185 V_{rms}[xviii], así uno de los potenciómetros es de 5 kΩ (POT2 Fig. 19) y el otro de 100 kΩ (POT1 Fig. 19), de esta manera, en el potenciómetro de 5 kΩ se puede calibrar la forma de onda de tal manera que guarde una relación proporcional con los bits de salida del ADC08161, y cumpla con las especificaciones de 5 V máximo para la seiial de entrada requerida por el ADC. Se emplea un amplificador operacional en configuracih de seguidor-unitario U1 (Fig. 19) para dar una salida de baja impedancia{4}.

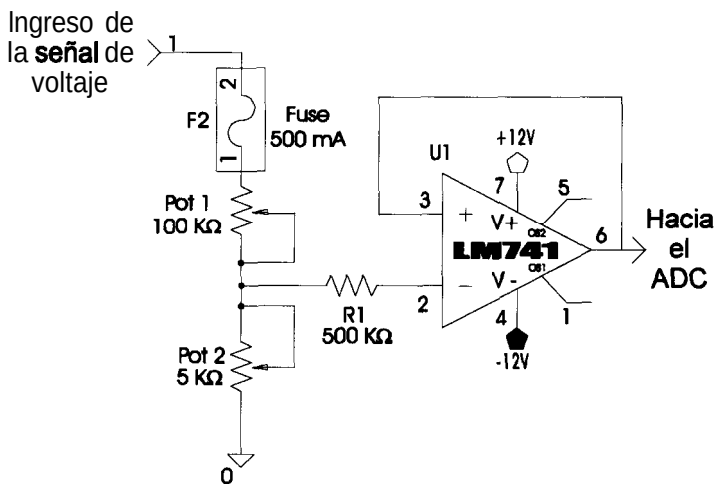


Figura 19 **Diagrama del circuito** sensor de voltaje **del** motor.

La etapa de filtro es igual al caso de la corriente, es decir, que consta de un filtro activo pasa bajo de Segundo orden realizado sobre la base de una configuracih de un amplificador operacional, cuya frecuencia de corte es

aproximadamente 450 Hz y tiene impedancia de salida baja por ser de amplificacih unitaria (seguidor-emisor) (esta etapa se explicará en la próxima sección).

3.2.2 Circuito sensor de corriente

La función del circuito sensor de corriente, es obtener una sefial analógica de voltaje a través de una resistencia ubicada en la línea de alimentación de armadura del motor, sefial que representa una copia fiel de la seiial de corriente, la que será convertida a digital (bits) para que el computador procese la gráfica y el valor promedio de la corriente.

Este circuit0 se divide en tres etapas: muestreo-escalamiento, filtro y de amplificación; las cuales son presentadas en la Fig. 20.

En la etapa de muestreo, la sefial de corriente en el lado AC del sistema de fuerza es sensada como una sefial alterna de voltaje producida por la caida a través de una resistencia R1 (Fig. 20) el cual tiene un valor de 0.1Ω, que es despreciable comparada con la resistencia de armadura del motor y está conectada en serie con la armadura del motor; este voltaje va a ser proporcional a la corriente, y es una fiel reproducción de su forma de onda. Debido a que la máxima

corriente permisible I_{DC}^{xix} por la armadura del motor es de 6 A (datos de placa), esto nos daria un voltaje promedio DC

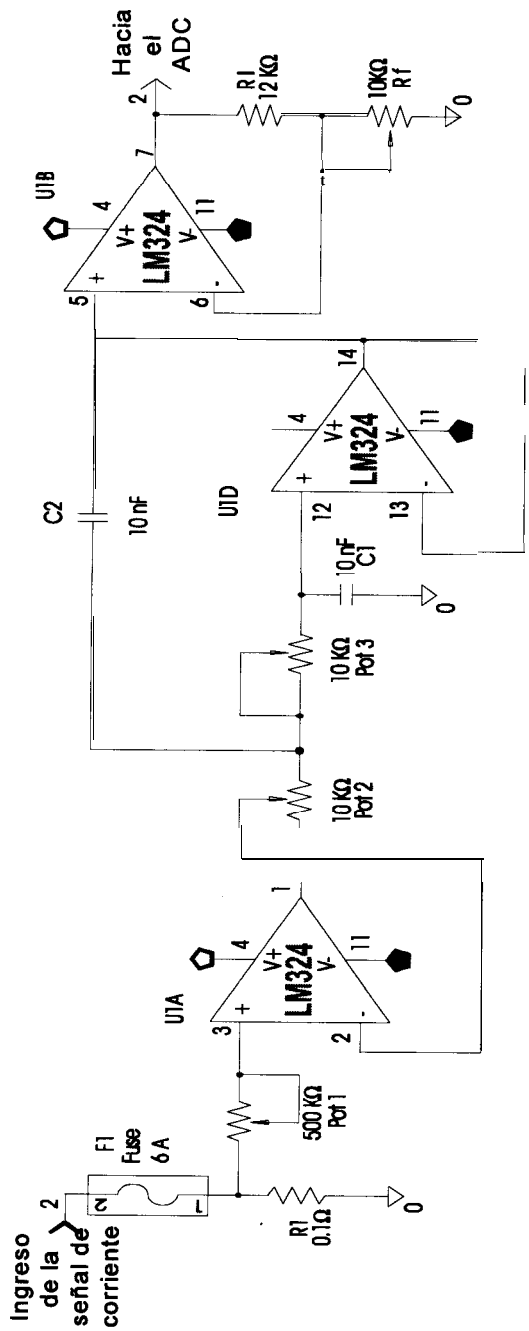


Figura 20 **Diagrama del circuito** sensor de corriente **del** motor.

máximo V_{DC} en la resistencia de 0.6 V, que seria entonces el máximo valor de excursion del sensor de corriente. La seiial es captada hacia el amplificador operacional U1A (Fig. 20) a través de una resistencia de polarización que puede estar entre 500 KΩ y 1 MΩ[xx].

El amplificador operacional (U1A) configurado como seguidor-emisor, el cual como sabemos tiene ganancia unitaria; y para nuestro caso es empleado para aislar la alta impedancia de la señal de entrada, de las siguientes etapas.

La segunda etapa está conformada por un filtro activo R-C, este filtro pasa bajo es de Segundo orden y es empleado para dejar pasar las frecuencias bajas, su frecuencia de corte es de aproximadamente 450 Hz[xxi] y a continuación se muestran las ecuaciones empleadas para diseñar este filtro {5}, los resultados para el valor de las resistencias y capacitores también son exhibidos.

$$C_1' = \frac{C_1''}{2\pi f_{cp}} \qquad C_2' = \frac{C_2''}{2\pi f_{cp}}$$

Donde C_1'' y C_2'' son escogidos por tabla para este filtro y f_{cp}[xxii] es la frecuencia de corte, la cual se requeria fuese de aproximadamente 450 Hz.

$$C_1 = \frac{C_1'}{R_1} \qquad C_2 = \frac{C_2}{R_2}$$

Los valores empleados fueron:

C_1= 10 ηF C_2= 10 ηF Pot2 ≈ 8KΩ Pot3 ≈ 10KΩ

Cabe anotar que los valores de Pot2 y Pot3 fueron calibrados experimentalmente hasta obtener que la frecuencia de corte fuese de aproximadamente 450 Hz; ademas este diseño tiene una configuración de amplificación seguidor-unitario que nos sirve para dar una impedancia de salida baja la cual es requerida por las especificaciones para la entrada del ADC08161.

La última etapa no es mas que un amplificador no-inversor U1B (Fig. 20) cuya ganancia no debe superar los 5V que es lo máximo permitido en la entrada analógica del ADC 08161, para esto la relación Rf/Ri del amplificador debe ser ≤ 4. En este caso Rf y Ri (Fig. 20) son resistencias variables las cuales se calibran experimentalmente, de forma tal que, no se amplifique la señal de corriente mas alla del rang0 establecido por los par&metros de resolución del monitor del computador, y ademas que la corriente guarde una

proporcionalidad o correspondencia con los bits de salida del ADC, los mismos que son explicados en el Apéndice A.

3.2.3 *Circuito sensor de velocidad*

Para sensar la velocidad del motor se ubicó un tacómetro DC en el eje motriz de la bancada, el cual tiene una relación de transformación lineal y además su señal también es sensada por el Medidor Digital de torque, velocidad y freno de corrientes de Eddy (MV 1045).

El circuito que nos sirve para sensar la velocidad se presenta en la Fig. 21.

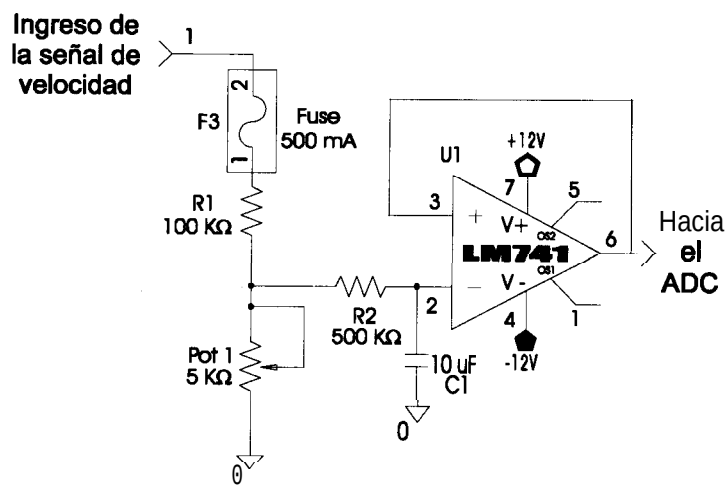


Figura 2 1 Diagrama del circuito sensor de velocidad del motor

Este circuito divide un voltaje, igual que en el sensor de voltaje; los valores de las resistencias fueron escogidos

sabiendo que el tacómetro DC tiene una relación de transformacih de 20.8V a 1000 r.p.m. (datos de placa).

Ya que en esta configuracih de control de velocidad del motor no se llega mas allá de 1000 r.p.m., entonces se usa los 20.8 V como voltaje a ser dividido, escogiéndose entonces los siguientes valores: para R1 (Fig. 21) una resistencia de 100 kΩ y para POT1 (Fig. 21) un potenciómetro de 5 KΩ. En el potenciómetro de 5 kΩ nosotros podemos escalar un voltaje de 5V máximo, el cual va con las especificaciones para la entrada analógica del ADC 08161.

Además se establece una configuración de un seguidor-emisor para el amplificador operacional U1 de la Fig. 21 que al igual que los casos anteriores nos sirve para dar una salida de voltaje de baja impedancia. El capacitor Cl es empleado para eliminar el rizado, ya que la señal del tacómetro DC no es completamente continua.

3.2.4 Circuito sensor de torque de carga

La carga del motor puede ser manipulada de dos formas dentro del Medidor Digital de Torque y Velocidad (MV 1045), Una manera de variar el torque de carga es por medio de un potenciómetro PB en la gráfica de la Fig. 12, ubicado en la parte frontal del MV 1045 (potenciómetro "break force" fig.

13), la otra es por el ingreso en la parte posterior de una sefial previamente convertida a analógica, proveniente del computador.

El torque de carga es cuantificado por medio de un transductor medidor de deformación (Strain Gauge), el cual se ubica en un extremo de la bancada, y se comunica con esta a través de un "brazo de torque".

La seiial del transductor ingresa al MV 1045 y procesada de tal manera que el dispositivo entrega esta sefial en forma de corriente continua escalada en un rango de 4-20mA. Y es esta sefial la que se emplea para acondicionarla antes de su ingreso al puerto paralelo del computador, el circuit0 encargado del acoplamiento se ilustra en la Fig. 22.

La sefial de corriente de 4-20mA sale de entre un voltaje V_{CC}[xxiii] y el colector del transistor T4 (Fig. 12), dicha señal de corriente circula a traves de una resistencia de 22Ω, y esta resistencia trabaja como carga del transistor T4; entonces al circular una corriente de 4-20mA por dicha resistencia el voltaje en el punto A de la Fig. 22 será:

$$V_a = 10 (V_c - V_{cc})$$

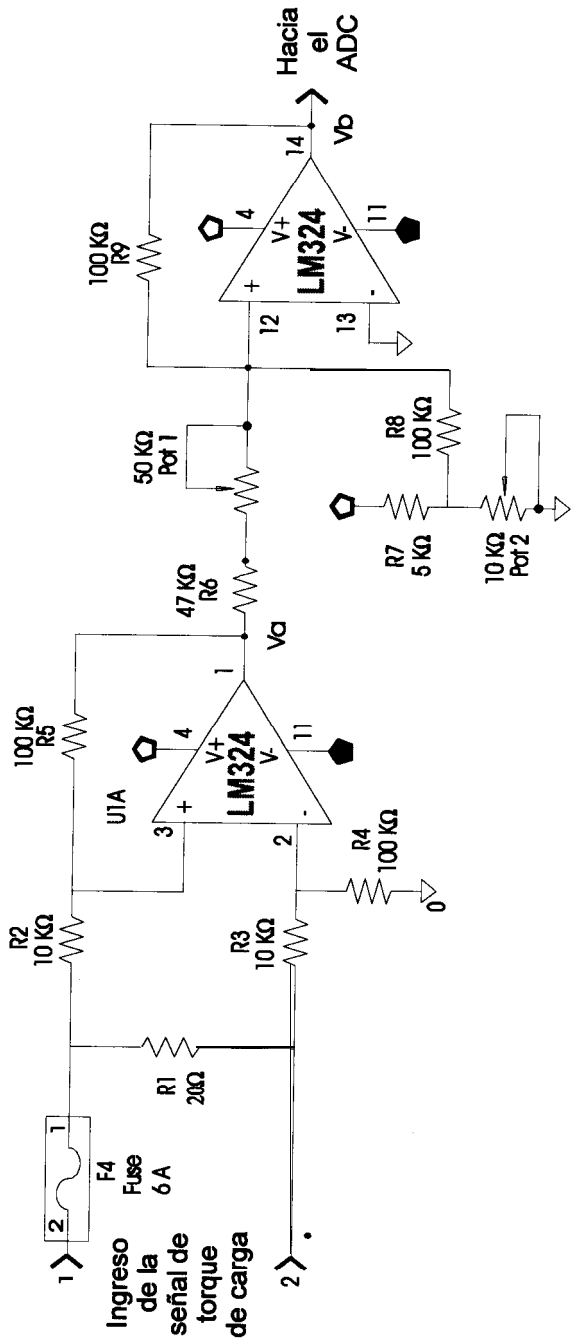


Figura 22 Diagrama del circuito sensor de torque de carga del motor.

Donde V_a es el voltaje a la salida del amplificador operacional U1A {6}, V_{cc} es el voltaje positivo de polarización del amplificador operacional y V_c es el voltaje de colector del transistor T4 como se aprecia en la Fig. 12.

Entonces el valor de V_a estara en un rango de -0.8 V. hasta -4.4 V., que son los valores correspondientes a la circulación de una corriente de 4-20mA en la resistencia de 22 Ω a la entrada del amplificador operacional U1A en la Fig. 22.

La siguiente etapa se trata de una configuración inversora para el amplificador operacional U1B con un potenciometro de 50 KΩ el cual nos permitira calibrar la ganancia del voltaje V_b y en consecuencia el máximo rango de salida de voltaje del convertidor.

Por lo tanto el rango de salida en el punto b V_b Serb:

$$V_b = G\ (V_a\ )$$

Donde G es la ganancia del amplificador y estará en un rango de -1 como mínimo y -2 como máximo, además dependera del potenciometro de 50 KΩ exclusivamente; y V_a es la salida de voltaje de la etapa anterior.

El arreglo de la resistencia R7 (Fig. 22) de 5 KΩ con el potenciómetro POT2 (Fig. 22) de 10 KΩ y la resistencia R8 (Fig. 22) de 100 KΩ nos sirve para ajustar el cero (offset) de referencia del voltaje de salida V_b, ya que este voltaje será el que ingrese al convertidor analógico-digital, el cual corresponde al bloque digital y es explicado en detalle más adelante.

3.3 Conversión y multiplexacih de la señal analógica a digital

El diagrama de la conversih y multiplexacibn de las sefiales analbgicas se muestran en la Fig. 23.

La conversih de las cuatro sefiales analógicas se las realiza por medio del ADC 08161, y la multiplexación de las seiiales es gobernada por los cuatro bits menos significativos del registro de datos del puerto paralelo del computador.

3.3.1 Equivalencias y conversih de datos para la señal analógica.

Las seiiales analógicas que salen de los bloques acondicionadores de señal, entran a una sección de conversih analógica digital la cual es idéntica para las cuatro sefiales, ya que se basa en un convertidor analógico a digital (ADC 08161), el cual es detallado ampliamente en el apéndice A.

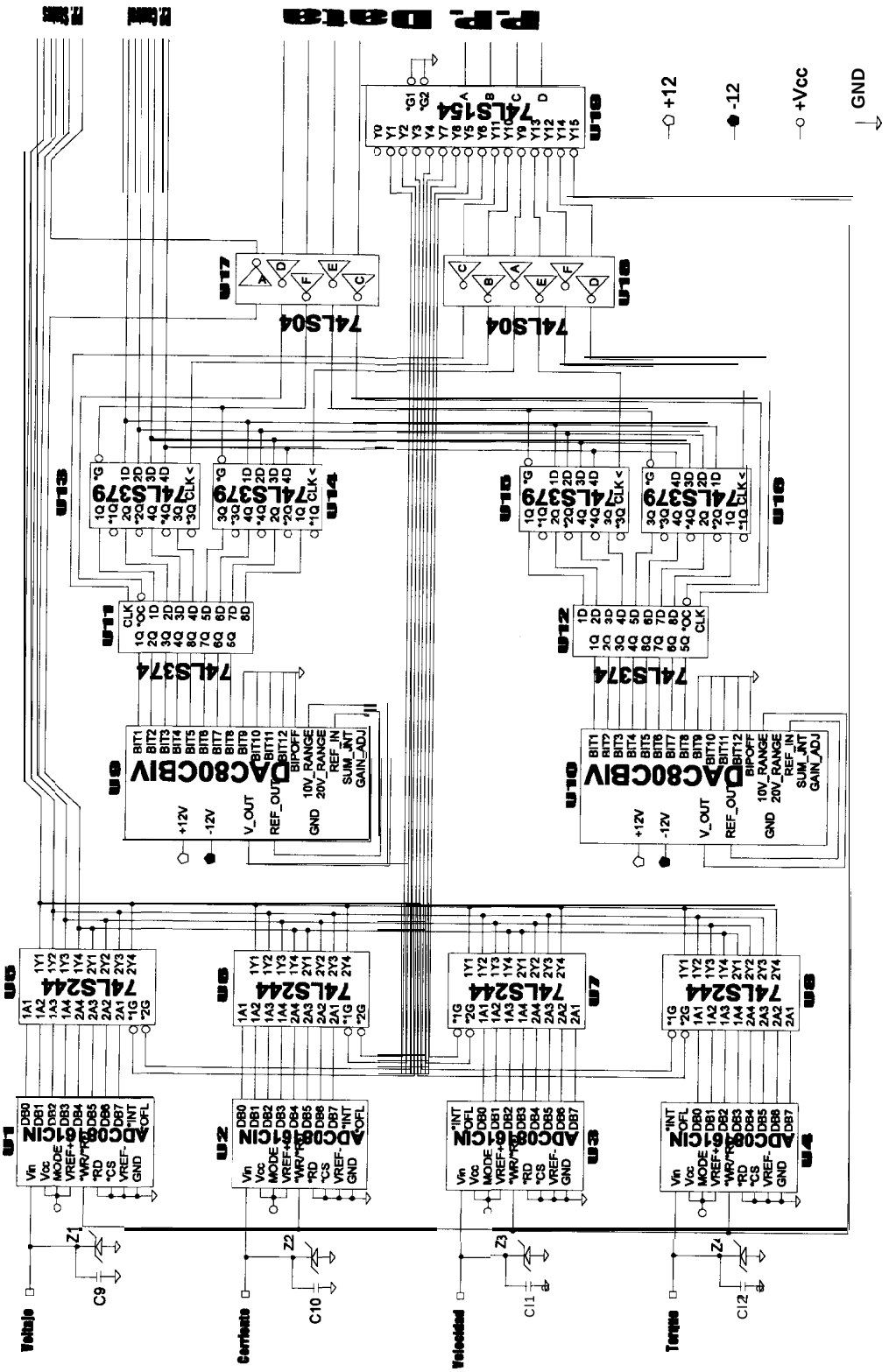


Figura 23 Diagrama esquemático de adquisición y envío de datos.

Los convertidores ADC 08161 son U1 para la conversih de la seiial analbgica de voltaje, U2 para la conversih de señal analógica de corriente, U3 para la conversih analógica de velocidad y U4 para la conversih de la señal analógica de torque de carga, como se ilustra en la Fig. 23.

Los capacitores C9, C10, C11, C12, (Fig. 23) sirven para evitar las seiiales de ruido que se acoplan a la entrada del convertidor. Los diodos zener Z1, Z2, Z3, Z4 (Fig. 23) sirven como protección de los ADC08161 contra señales analógicas que se extiendan más allá de los limites permitidos por la entrada analógica del ADC.

Las señales analógicas son convertidas a digital empleando un proceso de cuantificación, el niimero de bits del convertidor ADC determina el número máximo de intervalos, códigos de cuantización y además determina la resolución, esto es, el más pequeño increment0 de amplitud de señal que puede discernir el sistema.

Para nuestro caso el número de bits del convertidor es de 8, por lo que tendrá 2^N códigos, donde N es el ndmero de bits del ADC. Tendrá también 2^N -1 intervalos. Como resultado tendremos entonces 256 códigos y 255 intervalos de cuantizacih diferentes {7}.

La resolución del convertidor viene dada por:

$$R(V) = \frac{V_{FS}}{2^N - 1} \quad ,$$

Donde V_{FS} es la diferencia entre el rang0 de voltaje comprendido entre V_{MAX} - V_{MIN}, los cuales son los valores máximo y mínimo de cuantificación de la señal.

Por lo tanto la resolución será de

$$R(V) = \frac{5}{255} = 19.6mV$$

Lo que se denomina tamaño de paso, ya que el tamaño de 1 LSB[xxiv] será ≈ 19.6 mV.

Entonces para las cuatro señales que son calibradas en un rango de 0 a 5v, como se lo explicó en las secciones anteriores, nosotros tendremos valores binarios entre 0 y 255 decimales los cuales serán codificados de la siguiente manera:

Para la serial analógica de corriente, cuya forma de acondicionamiento de la señal entre 0 y 5V fue explicada en la sección 3.2.2, la equivalencia de códigos con valores en amperios se muestra en la tabla IV.

Código binario	Amperio
0	0
1	$23.5 * 10^{-3}$
2	$147 * 10^{-3}$
.	.
.	.
.	.
254	5.9765
255	6.0000

Tabla IV Tabla de equivalencias **binarias** de corriente

Con esta tabla logramos cubrir el rango de trabajo nominal para la corriente de armadura del motor que es de 6 A.

Para la señal analógica de voltaje, cuya forma de adaptación de señal entre 0 y 5V fue desarrollada en la sección 3.2.1, la equivalencia de códigos con valores en voltios se presenta en la tabla V.

Código binario	Voltios
0	0
1	0.78
2	1.56
.	.
.	.
.	.
254	219.22
255	220

Tabla V Tabla de equivalencias **binarias** de voltaje **del** motor

Con esto barremos el rango máximo de excursión para el voltaje de armadura, esto es, 220 V_{DC}.

Para la sefial analógica de velocidad, cuyo acoplamiento de sefial entre 0 y 5V se explicó en la sección 3.2.3, tiene una equivalencia para sus valores binarios, la cual se muestra en la tabla VI.

Códigos binarios	RPM
0	0
1	3.92
2	7.84
.	.
.	.
.	.
254	996.08
255	1000

Tabla VI Tabla de equivalencias binarias de velocidad del motor

Esto es debido a que debemos barrer un rango de mil r.p.m., recomiendación que hace el fabricante de los módulos (MV 4200) para la regulación de velocidad por tacometro.

Para la sefial analógica de torque de carga, cuyo acondicionamiento de la sefial a un rango entre 0 y 5V fue desarrollado en la sección 3.2.4, las equivalencias para los códigos binarios se muestran en la tabla VII.

Códigos binarios	Nm
0	0
1	$39.2 * 10^{-3}$
2	$78.4 * 10^{-3}$
.	.
.	.
.	.
254	9.9608
255	10.0000

Tabla VII Tabla de equivalencias binarias de torque del motor

Estas equivalencias comprenden un rango de 10 Newton metros (Nm), aunque en la prdctica no se llega a barrer todo este intervalo, ya que al aplicar un variación de torque produce la elevación brusca de la corriente del rotor, debido a que la relación no es lineal entre el torque y la corriente de armadura.

3.3.2 Multiplexación de la señal analógica.

Todos los ADC08161 reciben la seiial de inicio de conversih al mismo tiempo por el pin 6 (WR/RD Fig. 23)del circuito integrado, y esta señal de inicio de conversih viene dada por la salida 15 del decodificador 74LS154 (u19 en la Fig. 23), que a su vez recibe la orden desde el registro de datos del puerto paralelo.

Una vez que tenemos las señales digitalizadas, los ocho bits de salida del ADC 08161, entran a una sección de multiplexacion, la cual consta de cuatro *buffer* de ocho bits (74LS244), los cuales están denotados en el diagrama de la Fig. 23 como U5, U6, U7 y U8. Cada ***buffers*** contiene los datos que entraran por el registro de estado del puerto paralelo.

Se emplea los ***buffer*** 74LS244 de tres estados, debido a la facilidad que brindan para multiplexar los datos, ya que tienen dos pines de habilitacion (1G y 2G en la Fig. 23), los cuales sirven para seleccionar cuatro de los ocho bits de cada circuito integrado, entonces por el registro de estado del puerto paralelo ingresardn ocho bits, seleccionando por medio de las salidas del decodificador 74LS154, los pines 1 y *19* de los *buffer* 74LS244 (1G y 2G Fig. 23) y empleando las técnicas de lecturas del puerto, las cuales fueron detalladas en la sección 2.4.1 el computador ingresara sus ocho bits (byte) de datos.

Los cuatro primeros bit que ingresan se obtienen habilitando el pin 1 del *buffer* (1G Fig. 23), y corresponden a la parte más significativa de los ocho bits (byte), mientras que los cuatro bits siguientes que ingresan con la habilitacion del pin 19 del 74LS244 (2G Fig. 23) corresponden a la parte menos significativa del byte.

Todo esto ocurre para la lectura de los ocho bits (byte) correspondiente a un valor de voltaje de la señal analógica. Para la lectura de las tres seiiales restantes se utiliza el mismo principio y procedimiento anterior, es decir que se habilita los pines 1 y 19 de cada *buffer* (1G *y 2G* Fig. 23), empleando las señales de las salidas del decodificador,el cual recibe los códigos binarios desde el registro de datos del puerto paralelo.

3.4 Comunicación del sistema con el computador a través del puerto paralelo

El puerto paralelo controla el ingreso y salida de datos desde el computador, dicho puerto fue descrito en el capitulo 1.

El ingreso de datos, se lo realiza escribiendo en el registro de datos la codificación binaria, que habilita al circuito digital del módulo de adquisicion, para luego ingresar los datos realizando una lectura del registro de estado del puerto paralelo.

Para el envio de datos, se escribe el código binario en el registro de control, luego se habilita al circuito digital del módulo de adquisicion de datos escribiendo la codificación binaria en el registro de datos del puerto.

3.4.1 Ingreso de datos

El registro de datos, es de ocho bits de los cuales los cuatro bits menos significativos manejan directamente las entradas A, B, C, D del decodificador 74LS154 (U19 en la Fig. 23), con estos cuatro bits podemos habilitar dieciseis salidas de voltaje bajo, de las cuales, la salida YO, es empleada como un estado de reposo, es decir, un estado que sólo se utiliza para blanquear el registro de datos, en la Fig. 23 se muestra un diagrama completo de como el puerto maneja o se comunica con el circuito digital.

La salida Y15 (U19 Fig. 23) es usada por todos los pines de inicio de conversion de los ADC, U1 , U2 , U3 y U4 en la Fig. 23, como se ve todos los ADC empezarian la conversion al mismo tiempo, y debido a la conexión de mínimo circuito digital que se emplea para los ADC, éstos tienen un tiempo de conversion de aproximadamente 700 ηs, por lo que el computador no necesita usar el pin de fin de conversion para saber que una conversion se ha realizado sino que el computador lee el registro de estado inmediatamente, las especificaciones tecnicas del ADC se encuentran en el Apendice A, y esto se debe a que el computador se tarda aproximadamente 1μs a 2μs en realizar las instrucciones de lectura del puerto a través del programa, tiempo suficiente

para que el dato esté listo en el *buffer* 74LS244 (U5, U6, U7 y U8 Fig. 23).

Como vimos anteriormente las salidas del 1 al 8 del decodificador 74LS154 (U19 en la Fig. 23); son empleadas para seleccionar los cuatro bits de la señal digitalizada que se encuentran en los *buffers* 74LS244, los cuales ingresaran por el registro de estado del puerto paralelo, las salidas Yl-Y2 del decodificador U19, seleccionan el primer dato digital (1 byte) así:

Los cuatro bits menos significativos del registro de datos del puerto, por medio de instrucciones de escritura del puerto a traves del programa, envian un código binario hacia las entradas del decodificador U19 (Fig. 23) para activar la salida Y1 del mismo decodificador, esta salida habilita los primeros cuatro bits que se encuentran en el *buffer,* los cuales serán los bits más significativos y se los ingresa con instrucciones de lectura del registro de estado del puerto, desde el programa controlador; además por medio del mismo programa se los desplazan 4 lugares hacia la izquierda. Luego, los cuatro bits menos significativos del registro de datos del puerto a traves del programa con instrucciones de escritura del puerto, sacan un código binario que selecciona la salida Y2 del U19 (Fig. 23), que habilita los siguientes cuatro bits del *buffer,* y por medio

del mismo programa se leen los cuatro bits del registro de estado que, luego se los une a los anteriores cuatro bits, formando así el primer grupo de ocho bits (byte) de dato digitalizado para la primera señal analógica que ingresa por el canal uno o primer ADC, en nuestro caso este canal es utilizado para sensar la señal de voltaje del motor.

Las salidas Y3-Y4, Y5-Y6, Y7-Y8, del decodificador U19 en la Fig. 23, son empleadas para seleccionar y formar los tres bytes de datos, los cuales corresponden a la señal de corriente, velocidad y torque de carga respectivamente. Estas señales ingresarbn por el registro de estado de cuatro en cuatro bits, de la misma manera que se leyó los primeros ocho bits (byte).

Una vez que ya se han leido los primeros cuatro grupos de ocho bits (32 bits) de las cuatro señales digitalizadas, el programa utiliza una rutina menor que repite este procedimiento para leer los grupos de ocho bits (bytes) sucesivos, y de esta manera, obtener datos que permitan formar graficas que sean fiel copia del voltaje y corriente del motor, además obtener valores de velocidad y Torque de carga del motor.

3.4.2 Salida de datos.

Se ha empleado la gráfica de la Fig. 24 para una mejor visualización de como trabaja el bloque de envio de datos, la Fig. 24 es una gráfica simplificada del envio de datos por el puerto paralelo, cuyo diagrama completo es el mostrado en la Fig. 23.

El envio de datos, se lo realiza por el registro de control del puerto paralelo, y se lo emplea para cambiar la velocidad o el torque de carga, todo esto se gobernará con los restantes cuatro bits más significativos del registro de datos del puerto, junto a los cuatro bits menos significativos del puerto utilizados en el caso anterior, esto fue para el ingreso de señales digitalizadas, el manejo de esta sección del circuito por parte del puerto es como sigue.

Si en el *programa* se escoge un cambio de velocidad, entonces los cuatro bits más significativos del registro de datos del puerto paralelo, por medio de instrucciones de escritura del puerto en el programa, envía hacia los habilitadores de los flip-flop señales digitales que permitan su selección o habilitación, pero antes de llegar a los flip-flop estas seiiales son invertidas o pasadas por inversores U17 (74LS04 Fig. 23) para que sean de voltaje alto, de esta manera,

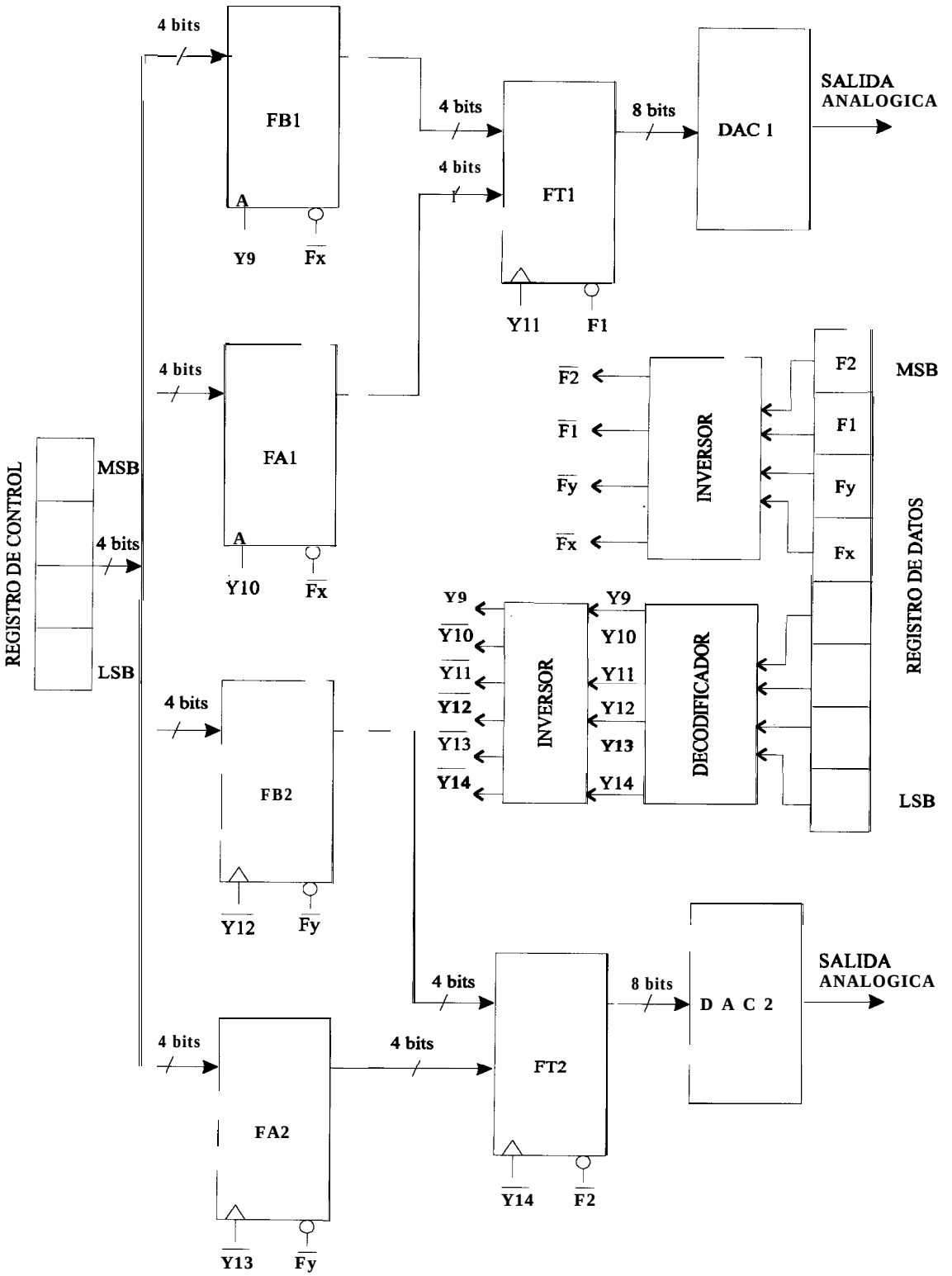


Figura 24 Diagrama de bloques de la comunicación con el PC a través del puerto paralelo.

saldrán la señal *Fx* que selecciona a los flip-flop FA1 y FB1, tambien sale *F1* que habilita a los flip-flop FT1, como se muestra en la Fig. 24, luego por medio de instrucciones de escritura en el puerto, el registro de datos, a travels de sus

cuatro bit menos significativos envía hacia las entradas del decodificador un código binario, que permite habilitar la salida Y9 del decodificador U19 en la Fig. 23, una señal digital que se emplea para el *clock* del circuito integrado FB1 (Fig. 24), la cual es previamente invertida para que sea de voltaje alto (*Y9* en la Fig. 24), entonces el programa realiza una escritura hacia el registro de control del puerto, sacando de esta manera los primeros cuatro bits de dato digital y almacenándolos en el flip-flop 74LS379, FB1 en la Fig. 24, siguiendo con la secuencia el programa, por medio de instrucciones de escritura del puerto saca los siguientes cuatro bits, y el registro de datos del puerto a traves de sus cuatro bits menos significativos, envía hacia las entradas del decodificador un código binario que habilita la salida Y10 del decodificador, esta señal se emplea para manejar el *clock* del flip-flop 74LS379, FA1 en la Fig. 24, seiial que es previamente invertida para que se vuelva de voltaje alto (*Y10* en la Fig. 24), siguiendo la rutina el programa a traves de instrucciones de escritura, envía los

siguientes cuatro bits y los almacena ahora en el flip-flop FA1, completando de esta manera los ocho bits del dato.

Como las señales Fx y F1 en la Fig. 24 son mantenidas durante todo este procedimiento, y continuando con el programa, a través de instrucciones de escritura del puerto, se envía al registro de datos un código binario; que se comunica con las entradas del decodificador, para habilitar la salida Y11, señal que al ser invertida (*Y11* en la Fig. 24), es empleada para manejar el clock del flip-flop 74LS374, FT1 en la Fig. 24, permitiendo así, que el dato se almacene en el flip-flop FT1 e ingrese al convertidor digital analógico a digital DAC80P.Luego, la señal es convertida a análoga y pasa a la sección de acondicionamiento para que pueda ser usada externamente , en nuestro caso por el MV 4200 que como se describió anteriormente, es por donde ingresa la señal de variacidn de velocidad del motor .

Si se escoge un cambio en el torque de carga, el procedimiento a seguir es igual al anterior sólo que ahora se trabajard con los circuitos integrados y señales de la parte inferior del gráfico de la Fig. 24, esto es con FT2, FA2, FB2 y DAC2.

Si se inicia esta rutina de envio de datos, escogiendo primero cualquiera de las dos variaciones para el motor, sea

de velocidad o de torque de carga, y luego se quiere hacer otro cambio, de tal manera que, las dos variaciones estén habilitadas al mismo tiempo, esto es factible de realizar debido a que el programa mantiene, mediante el registro de datos del puerto paralelo, siempre habilitadas las señales F1 y F2, que son las que sacan de funcionamiento a los flip-flop 74LS374 (FT1 y FT2 en la Fig. 24).

3.5 Conversión y multiplexación de la señal digital a analógica.

La señal digital tiene su origen en el programa de control y supervision, cuando nosotros escogemos si iniciamos un cambio ya sea para la velocidad o para torque de carga.

De esta manera se tiene a la salida del convertidor digital analógico (DAC) un voltaje entre cero y diez voltios la cual será entregada a la siguiente fase del circuit0 que es la de acondicionamiento de la señal de voltaje para que vaya hacia el controlador de velocidad MV 4200 para producir un cambio de velocidad y hacia el medidor digital de torque de carga MV 1045 para producir una variación de torque.

Cabe anotar que en el Apendice B se encuentra las especificaciones tecnicas del circuito integrado DAC80P.

3.5.1 *Equivalencias y conversión de datos para la seiial digital.*

Si nosotros escogemos un cambio de velocidad, éste tiene lugar luego del siguiente proceso:

La velocidad seleccionada se la envía por teclado en revoluciones por minuto (r.p.m.), es tomada por el programa y llevada a código binario con las siguientes equivalencias.

RPM	Códigos binarios
0	-255
3.92	254
7.84	253
.	.
.	.
.	.
996.08	1
1000	0

Tabla VIII Tabla de equivalencias binarias de la señal variadora de velocidad del motor

Con esto barremos el rango de velocidades permitido por el regulador de velocidad MV 4200 y además recomendado para la regulación por tacómetro, cuyo rang0 es de cero a mil revoluciones por minuto (r.p.m.).

Este código binario es puesto por medio de instrucciones de programa en el registro de control del puerto paralelo. Luego, es guardado en los dos registros 74LS379 (U13 Y U14 en la Fig. 23), cuatro bits por cada registro. Por medio del registro 74LS374 (U11 en la Fig. 23), son puestos en la

entrada digital del DAC, tal como se describió en la sección anterior. Si por otro lado, escogemos un cambio en el torque de carga éste se produce de la siguiente manera.

El cambio de torque de carga se realiza por medio de teclado en Newton metro (Nm) y llevado a código binario por medio del programa con las siguientes equivalencias.

Nm	Códigos binarios
0	255
$39.2 * 10^{-3}$	254
$78.4 * 10^{-3}$	253
.	.
.	.
.	.
9.9608	1
10.0000	0

Tabla IX Tabla de equivalencias binarias de la señal variadora de carga del motor

Los valores de la tabla IX, cubren el rango de torque máximo permisible, que es de diez Newton metro. Cuando este cambio entra en funcionamiento, produce interferencia que degenera las señales de ingreso al computador; por lo que no se utiliza en su rang0 total, más bien se lo emplea para visualizar la demanda de corriente del motor.

Debido a que en la línea de corriente del motor se encuentra un fusible F1 (Fig. 20) el cual nos permitird proteger el motor para un valor determinado de corriente en un rango de

cero hasta seis amperios nominales para el rotor, este fusible se lo ha puesto de un valor del 50% del valor de corriente nominal (tres amperios), debido tambien que, la relación entre el torque y la corriente de armadura no es lineal y ademds tampoco lo es la salida de la unidad MV 1045; por estos tres factores anteriores no es recomendable usar el rango total de cambios para torque de carga que permite el programa en el computador, sino que se emplea un 25% de su rango, ya que llega un instante que el cambio de un bit en el puerto produce un cambio brusco en el torque de carga, el cual puede quemar el fusible de protección antes mencionado, en conclusion se emplea un cambio máximo de aproximadamente 2 Newton metro.

La conversion digital-analógica la dan los DAC80P y es igual para cualquiera de los dos casos y se la realiza por medio de la técnica llamada red de escalera de resistencias cuyo resultado está descrito por la siguiente tabla.

Los valores resaltados en la tabla son los que usamos, entonces según la tabla para un cambio de un bit menos significativo (LSB) le corresponden a un cambio de 2.44 mV en la salida, además se debe notar que la correspondencia de los datos digitales es inversa, es decir, que a un código equivalente a 255 decimal o 11111111 binario le corresponde aproximadamente 0 voltios de salida, y para un código

equivalente a 0 decimal o 00000000 binario le corresponden aproximadamente 10 voltios de salida.

ANALOG OUTPUT				
DIGITAL INPUT	VOLTAGE (1)		CURRENT	
MSB LSB	0 to +10 V	±10V	Oto-2 mA	±1 mA
000000000000	+9.9976 V	+9.9951 V	-1.9995 mA	-0.9995 mA
011111111111	+5.0000 V	0.0000 V	-1.0000 mA	0.0000 mA
100000000000	+4.9976 V	-0.0049 V	-0.9995 mA	+ 0.000 5mA
111111111111	0.0000 V	–10.0000V	0.0000 mA	+1.000 mA
One LSB	2.44m V	4.88m V	0.488 mA	0.488 mA

NOTE: (1) To obtain values for other binary ranges:
0 to +5V range divide 0 to +10V range values by 2
±5V range: divide ±10V range values by 2.
±2.5V range: divide ±1OV range values by 4.

Tabla X Tabla de equivalencias binarias de voltaje del DAC

La calibración de los voltajes de salida entre 0 y 10 voltios, para ambos DAC80P, fue experimental; ya que sólo se empleó ocho de sus doce bits de entradas, y estos fueron los bits más significativos.

3.5.2 Multiplexación de la señal digital.

La multiplexación de las seiiales digitales de variación de velocidad y torque de carga, se las realiza con el control del puerto paralelo sobre el cicuito digital del módulo de adquisición y envio de datos.

Los datos digitales (ocho bits) son puestos por medio de instrucciones de programa en el registro de control, primero se habilita al registro de cuatro bits 74LS379(U13 fig. 23); luego, se realiza una escritura de código binario en el registro de control, y finalmente se envía una señal de reloj al registro 74LS379 (U13 en la Fig. 23), con lo que quedan almacenados los primeros cuatro bits.

Los restantes cuatros bits se los envía de igual forma sólo que se los almacena en el otro registro 74LS379 (U14 en la fig. 23).

Luego, los ocho bits son transferidos hacia la entrada digital del DAC (U9 fig. 23) por medio del registro 74LS374 (U11 en la Fig. 23).

Todo estos circuitos integrados intervienen para un cambio en la velocidad, si queremos un cambio en el torque de carga, el envio de datos se realiza de igual forma que para la

velocidad, sólo que ahora los datos (ocho bits) se van a almacenar en los circuitos integrados 74LS379 (U15 y U16 en la fig. 23), cada uno con cuatro bits, luego son transferidos hacia el DAC (U10 fig. 23) por medio del registro 74LS374 (U12 fig. 23).

El mantenimiento de la seiial digital se la realiza, cuando el puerto (regisro de datos) sostiene las seiiales de habilitacion de los registro de ocho bits 74LS374 (F1 y F2 de la Fig. 24), con esto se puede mantener un cambio de velocidad, un cambio de torque, o ambas cosas a la vez.

La señal F1 sirve para habilitar y mantener la señal digital de velocidad y la seiial F2 sirve para habilitar y sostener la señal digital de torque de carga. La habilitacion y sostenimiento fueron explicados en la sección 3.5.

3.6 Circuitos acopladores de señales controladoras

Los circuitos que acondicionan las seiiales de velocidad y torque de carga son empleados para acoplar la señal análoga entre el módulo de interface y el controlador de velocidad, en el caso de una variación en la velocidad; para acoplar la señal y como fuente de corriente para el caso de la señal analógica de torque de carga, las descripciones de estos circuitos se detallan a continuación.

3.6.1 Señal variadora de velocidad del motor

La señal que produce una variación en la velocidad, sale de la parte frontal del módulo de adquisición y envio de datos, por el terminal de salida w (fig. 16), y se introduce al controlador de velocidad MV 4200 por el terminal del lado derecho tipo banana, numero tres en la Fig. 10. El cual se lo considera, debido a que en dicho punto se está abriendo el potenciometro P1 (Fig. 10), que es el que sirve para variar la velocidad del motor en forma manual en el MV 4200.

De manera experimental se tabuló en este punto el rango de voltaje DC que servia para variar la velocidad entre 0 y 1000 r.p.m., velocidad recomendada para la regulación por tacometro; dicho rango de voltaje DC result6 entre 0 y 10 V_{DC}; por esta razón es que el circuito integrado DAC80p se lo configura para que tenga un voltaje de salida variable entre 0 y 10 V_{DC}, como se observó en la sección anterior.

La seiial de velocidad es acoplada al controlador de velocidad MV 4200 por medio del circuito ilustrado en la Fig. 25, compuesto por un amplificador operacional U1 en configuración de seguidor emisor, que no es mas que un aislador de impedancia entre el circuito digital del módulo y el MV 4200. el diodo D1 del diagrama de la Fig. 25 sirve para no permitir el retorno de señal alguna proveniente del controlador de

velocidad, que pueda perjudicar al amplificador operacional o el circuito del módulo de adquisicion de datos.

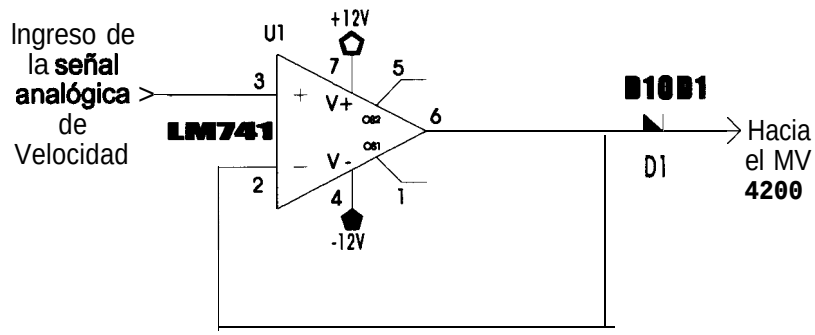


Figura 25 Diagrama del circuito acoplador de la señal variadora de velocidad.

3.6.2 Señal variadora de torque de carga del motor.

La señal que varia el torque de carga, sale desde la parte frontal del módulo de adquisicion y envio de datos, por el terminal de salida **T** (fig. 16), y se conecta hacia módulo medidor digital de torque, velocidad y freno de corriente de Eddy (MV 1045), ya que éste permite el ingreso de una señal externa de control **de** torque; la señal debe ser de corriente en un rango **de** 0 a 20 mA, ingreso que fue descrito en la sección 2.2.3.

Con esta información y como con el andlisis del diagrama de circuiteria de esta sección del MV 1045 (Fig. 26) se puede observar que la corriente requerida entre 0 y 20 mA debe circular por una resistencia R1 de 470 Ω (terminales C3 y C4 en la Fig. 26).

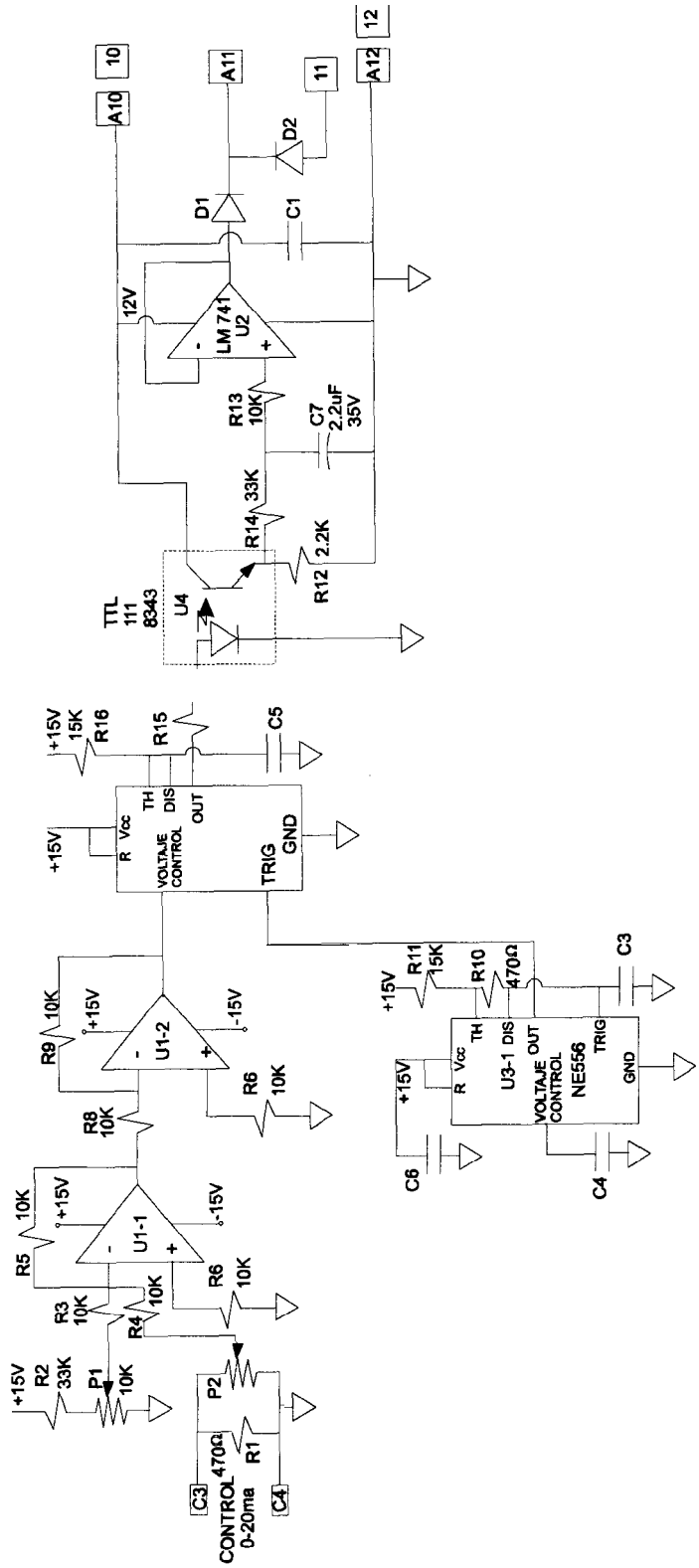


Figura 26 **Diagrama** de la **sección** de ingreso de la **señal** de torque en el MV 1045.

Se determinó que esto se puede lograr poniendo una fuente de voltaje variable entre 0 y 10 V_{DC}, con la capacidad de manejar hasta una corriente de 20 mA, esto debido a que la carga es constante (470 Ω).

Como se observó en la sección anterior el circuito integrado DAC80p da un voltaje variable de salida entre 0 y 10 V_{DC} con lo que se logra obtener la primera condición de la fuente de

voltaje, para lograr el manejo de la capacidad de corriente, se implement6 el circuito de la Fig. 27, que es un amplificador operacional U1 configurado como un seguidor emisor con un transistor Ql, el cual permite el manejo de la corriente requerida (20mA) evitando daños al amplificador operacional.

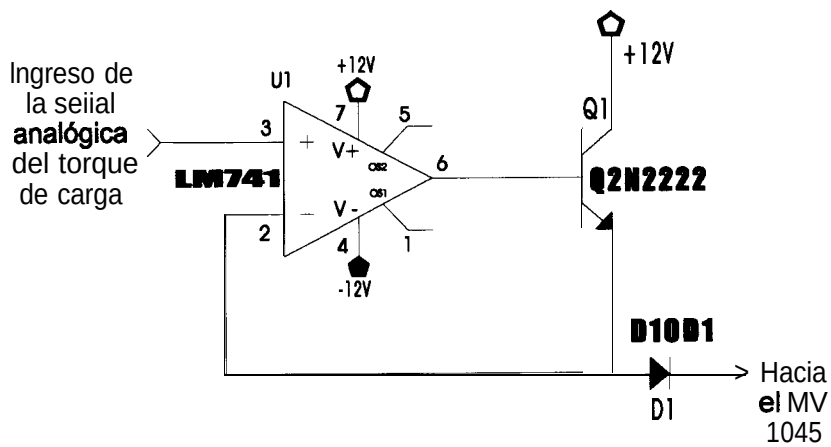


Figura 27 Diagrama del circuito acoplador de la señal variadora de torque.

Esta configuración además da una salida de baja impedancia, y al igual que el circuito empleado para la seiial de variación

de velocidad, emplea un diodo D1 (Fig. 27), el cual no permite el retorno de alguna señal del MV 1045 que puede causar perjuicio al circuit0 acoplador o al módulo de adquisición de datos.

CAPITULO IV

PROGRAMA DE SUPERVISIÓN Y CONTROL DEL SISTEMA

4.1 Introduccih

Como se mencionó antes, el programa o software fue diseñado en Lenguaje C++ v 3.0 de Borland bajo DOS. En el apendice C se puede encontrar el listado del programa.

En este capitulo, se explica como está estructurado el programa que controla las acciones del sistema. El computador utiliza el puerto paralelo fisico y el puerto paralelo virtual. El puerto paralelo virtual, es el **programa** encargado de controlar las acciones que gobiernan el sistema, para ello cambia el estado del puerto fisico enviando las respectivas señales electronicas. El proceso se inicia ingresando ciertos par&metros iniciales hacia el puerto, que permiten trabajar al resto del sistema. El programa está basado en dos procesos muy importantes: la lectura de datos y generación de datos, a través de ellos se controla el paso de la información (bits).

El programa fuente consta de 6 archivos PRINCIPA.CPP, GRAFICA.CPP, ACCION.CPP, OBJETO.CPP, CARGA.CPP, VELOCIDAD.CPP. Todos estos

archivos se enlazan para formar uno solo ejecutable el cual tiene por nombre ***motor.exe.***

4.2 Selección de la direccidn del puerto paralelo

Para seleccionar la dirección del puerto por el cual se van a enviar las seiiales de control hacia el motor el programa utiliza la funcion *inframe1()*. Esta funcion presenta un conjunto de direcciones opcionales enmarcada en un recuadro dibujada por la funcion frame(...), las direcciones que se presentan se encuentran almacenadas en el arreglo apuertos[ ], y son: 888, 632 y 956 (37811, 278h, 3BCh, en hexagesimal respectivamente).

4.3 Configuración del puerto paralelo

Una vez seleccionada la dirección del puerto, el programa procede a configurar el estado fisico de los pines del puerto paralelo, que permitiran continuar el trabajo al resto del programa.

Esta configuración se realiza para "encerar" el puerto paralelo, de tal manera que, al encender el motor y su bancada no realicen alguna acción inesperada y más bien se mantengan en estado pasivo.

La configuración del puerto paralelo, permite al programa colocar todos los pines del puerto, en un "estado inicial", para esto el programa utiliza la funcion de tipo entera ***iniciaguerto ()***. Esta

función emplea una de las direcciones del arreglo de tipo entero *apuerto[ ], para asignársela a la variable ***pto.***

Por medio de esta dirección base se envía el valor hexadecimal **0x0h** que encera los pines del puerto, mediante la instrucción ***Outportb (pto+2,0x0).*** Donde ***Pto+2,*** es la direccidn del registro de control

4.4 Configuración del puerto paralelo para uso del programa

El Sistema operativo del puerto paralelo utiliza tres registros, cada uno de los cuales son seleccionados al escoger la dirección correspondiente a ese registro. Esta dirección utiliza la variable ***pto.***

Debemos definir como se utilizan los **bits** del registro de datos en el programa (Fig. 28).

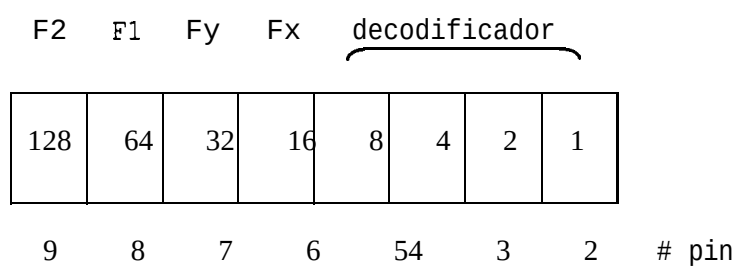


Figura 28 Registro de datos **del puerto** paralelo

F1 y F2 son las seiiales que habilitan los flip-flop que forman la palabra de ocho bits conectados con la entrada de los DAC. Fx y Fy

habilitan a los flip-flop (cuatro bits)para almacenar la parte alta o la parte baja de la información proveniente registro de control, segiin el caso, como se explicó en el capitulo tres. Los cuatro bits restantes del puerto de datos accionan al decodificador, que envía las señales a los diferentes dispositivos a su cargo.

El registro de estado (Fig. 29) utiliza los cinco bits más significativos, éste no tiene problemas con señales invertidas, pues utiliza un inversor en el pin 11 del puerto de estado. El programa debe realizar una "conjunción" con el valor 0Fh (dato&=OFOh) para eliminar los bits a los que no tiene acceso el puerto de estado.

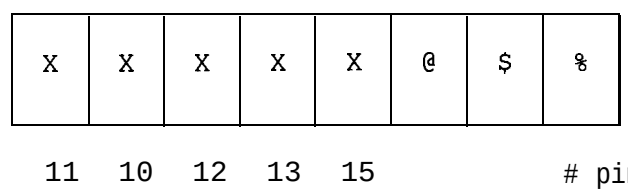


Figura 29 Registro de estado del puerto paralelo

El registro de control se utiliza para enviar información hacia el DAC, éste no utiliza inversores, por lo tanto, debemos enmascarar el registro de control al enviar los datos. El valor que utilizamos para enmascarar es 0Bh, que invierte los valores binarios para los pines 17, 14 y 1 del puerto paralelo. El registro de control utiliza solo los cuatro bits menos significativos, por lo tanto, los otros cuatro son información no requerida.

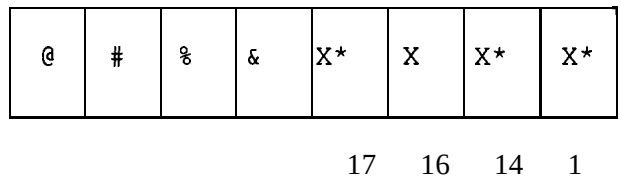


Figura 30 Registro de control del puerto paralelo

Cuando se utiliza el registro de datos, su información es dividida en dos partes: los cuatro bits más significativos controlan el manejo de los seis flip-flop en el proceso de escritura del puerto; los cuatro bits menos significativos controlan al decodificador que maneja el proceso de lectura del puerto y los cambios de estado (*clk*) de los flip-flop.

Se utiliza los valores: (Si la dirección base del puerto es 378h)

pto para el registro de datos -> 378h(888)

pto+1 para el registro de estado -> 379h(889)

pto+2 para el registro de control -> 37Ah(890)

4.5 Lectura de datos a través del puerto paralelo

La lectura de datos a través del puerto paralelo es realizada mediante la función lee ***datos()*** que se encuentra en el archivo GRAFICA.CPP.

El fragmento del programa lee_dato() (rutina), lee los datos del motor a través del puerto paralelo en un instante dado; 0 sea, los cuatro datos de las señales del motor en un instante dado. Esta rutina se repite 700 veces para formar una base de datos que luego es graficada en el monitor.

```
1.  int lee_dato()
2.  {
3.  int i,j,k,str,dat;
4.  double fvel;
5.  _SVOLTAJE=0;
6.  _SCORRIENTE=0;
7.  str=f1+f2+15;
8.  numero(_INVVELIN,150,getmaxy()-80,BLANCO);
9.  numero(_VELOCIDADIN,200,getmaxy()-80,BLANCO);
IO. numero(_CARGAIN,440,getmaxy()-80,BLANCO);
11. dobles(_INVCARIN,390,getmaxy()-80,BLANCO);
12. for (i=1;i<=datos;i++)
13. {
14. k=1+f1+f2;
15. outportb(pto,str);      // senal de STAR/CLEAR
16. j=0;
17. do{
18. outportb(pto,k);        // selecciona parte alta del ADC
19. dh=inportb(pto+1);      // lee senal "ALTA" del ADC
20. dh&=OxFO; // convierte a variable de 8 bits
21. k++;                    // da paso al siguiene 74LS244
22. outportb(pto,k);        // selecciona parte "baja" del ADC
23. dl=inportb(pto+1);      // lee senal "BAJA" de un ADC
24. dl>>=4;  // desplazamiento a la derecha
25. dl&=0x0F;               // convierte a variable de 8 bits
26. k++;                    // da paso al siguiene 74LS244
27. switch(j)
28. {
29. case 0:
30. _SVOLTAJE=_SVOLTAJE+dh|dl;
31. break;
32. case 1:
33. _SCORRIENTE=_SCORRIENTE+dh|dl;
34. break;
35. case 2:
36. _VELOCIDADIN=dh|dl;
37. break;
38. case 3:
39. _CARGAIN=dh|dl;
40. break;
41. }
42. t[i][j]=dh|dl;           // guarda el dato leido en el arreglo T
43. j=j+I;                   // da paso a la siguiente de las cuatro señales
44. }
```

```
45. while(j<=3);
46. }
47.
```

Listado de programa de la función lee_dato()

Cuando la rutina recibe la orden de leer, se prepara la seiial de inicio (*str*) en la línea 7 del listado. Con los valores correspondientes a F1 y F2 (Fig. 24), según sea el caso, se le agrega el valor de 15, que en el decodificador corresponde a la señal de inicio de conversion para los cuatro ADC. Una vez que el valor de *str* es tornado, ya se ha formado la palabra binaria que saldrá por el registro de datos. Como se mencionó antes la rutina se repite 700 veces que es el valor almacenado en la variable *datos* (línea 12) por omisión.

La variable *k* (línea 14) lleva el control de los ocho *buffers* que transportan la información exterior hacia el puerto de estado. Esta variable cambia de 1 hasta 8, reservándose el valor de cero en el decoficador como estado de espera. La variable *j=0* establece el lugar en la columna del arreglo *T* que va a ocupar cada una de las cuatro informaciones.

Con la instrucción ***outportb***(*pto,str*) *se* envía la seiial de conversion *(str)* en la dirección *pto,* inmediatamente se selecciona el primer *buffers* mediante la instrucción ***outportb(****pto,k****),*** luego se recibe la información en la variable temporal *dh,* que recibe la parte más significativa del dato ingresado, mediante la instrucción *dh=****inportb(****pto+1****)***. La instrucción ***dh&=0xF0*** de la línea 19,

convierte la información en variable de ocho bits. Pasamos al siguiente *buffer* incrementando *k* con la instrucción k+t de la línea 21. Con la instrucción ***outportb(pto,k)*** de la línea 22, seleccionamos la parte menos significativa del dato. Con la instrucción *dl=**inportb**(pto)* almacenamos el dato. Desplazamos la información cuatro lugares, para poder convertirla a 8 bits aplicamos la instrucción *dl>>4*, luego "unimos" *dh y dl* por medio del operador "|" y mediante la instrucción t[i][j]= dh|dl almacenamos la información en el arreglo *t*.

Este proceso se repite cuatro veces, por las cuatro señales a sensar, y se pasa al siguiente dato.

4.6 Generación de datos a través del puerto paralelo

La generación de los datos para controlar el motor son realizados por la función *dac1(int www)* que se encuentra dentro del archivo *VELOCIDAD.CPP* y por la función dac2(int www) que se encuentra dentro del archivo *CARGA.CPP.*

```
1.  int dacl(int www)
2.  {
3.  int ww;
    ww=www^0xb;
5.  ww=ww&0xf;
6.  outportb(pto,fx+f1+f2);          // habilta Fx y F1 y/o F2 si fuera el caso
7.  outportb(pto+2,ww);              // escribe parte baja del DAC
8.  outportb(pto,clk+1+fx+f1+f2);    // nivel alto para clk para FBx
9.  outportb(pto,fx+f1+f2);          // nivel bajo para el clk
10. www>>=4;                         // desplazamiento
11. ww=www^0xb;
12. ww=ww&0xf;
13. outportb(pto+2,ww);              // escribe parte alta del DAC
14. outportb(pto,clk+2+fx+f1+f2);    // clk para FAx
```

```
15. outportb(pto,fx+f1 +f2);              // baja el clk
16. outportb(pto,clk+3+fx+f1 +f2);        // clk para F1 o F2
17. outportb(pto,f1+f2);          // enclavamiento de Fl y/o F2
18. return 0;
19. }
```

La misma aplicación se repite en ambos archivos. La rutina recibe un valor www, el cuál es enmascarado en la línea 4 con la operación *"www^0xB"*, luego este mismo valor es convertido a ocho bits con la operación *"ww&0xF"*. La línea 6 selecciona los flip-flop que permiten el paso de los datos binarios hacia el DAC. La línea 7 "escribe" la palabra binaria ww hacia el puerto de control (*pto+2*). Esta información es "fijada" en el flip-flop temporal de la parte baja para el DAC con la línea 8 y 9. Se realiza un desplazamiento de cuatro bits, enmascara nuevamente la información y se la convierte en dato de ocho bits con las líneas 10,11 y 12. Se escribe nuevamente hacia el puerto de control, esta vez, la parte alta del DAC. Así mismo las líneas 14 y 15 fijan la información para el otro flip-flop y con la línea 16 y 17 se fija la información en el flip-flop F1 o F2 que va a ser recibida por el DAC1 o el DAC2, respectivamente.

4.7 Selección de la velocidad de inicio del motor

Para seleccionar el tiempo de aceleracion con que el motor comenzará a funcionar, el programa utiliza la función *inarranque()*, la cual presenta las opciones de velocidades almacenadas en el arreglo de tipo entero *arranque2[ ].

4.8 Variación manual de velocidad y carga del motor

La variación manual de carga se llevan a efecto en la funcion ***inon()*** que se encuentra dentro de archivo ACCION.CPP

La funcion *inon ()* permite desplazarse entre un menú de controles y se vale de las funciones *choiceon()* y *efectoon ()* para dar el efecto de selección

4.9 Variación automática de velocidad y carga del motor

La variación automatica de velocidad y carga se realiza por medio de una planificacion, donde el usuario ingresa hasta un máximo de 5 valores de velocidad y 5 de cargas.

El programa para receptar estos valores utiliza la funcion de tipo entera *inauto()*, la cual se encuentra dentro del archivo ACCIÓN.CPP Esta funcion se vale de las funciones de visualización de datos como son *numero(...)* y *dobles(...)* para presentar los ndmeros por el monitor además de *entra_num(...)* y *entra_dob(. ..)* para permitir ingresar ndmeros desde el teclado.

Esta funcion tambien verifica que el número de entradas no sea mayor a 5, y que la cantidad de digitos ingresados en un número sea de 3 para los enteros y 2 enteros más 2 decimales para los dobles.

4.10 Selección de las sefiales a mostrar en el monitor

Para seleccionar el tipo de sefiales que se van a visualizar en el monitor, el programa utiliza la funcion inframe3(), la cual presenta los cuatro tipos de sefiales que son posibles observar, los tipos de señales se encuentran almacenados en el arreglo de tipo char *gráficos[].

4.11 Visualización de las sefiales en el monitor

Para visualizar las sefiales de corriente, voltaje, torque y velocidad; el programa utiliza la funcion *traza ();* en esta funcion primeramente lee los datos del puerto paralelo por medio de la funcion *lee_dato()*, en dicha función se almacena los datos en un arreglo de 700 filas por 4 columnas, en donde cada columna almacena los datos de voltaje, corriente, velocidad y torque, como se muestra en el cuadro siguiente:

Almacenamiento de señales (Arreglo T[ ][ ])	
Arreglo de datos	Señales
T[0.....700][0]	Voltaje
T[0.....700][1]	Corriente
T[0.....700][2]	Velocidad
T[0.....700][3]	Torque

Tabla XI Arreglo que almacena los datos que ingresan por el puerto paralelo

Luego de que los datos están almacenados en el arreglo, el programa hace uso de la funcion gráfico(color), la cual dibuja en el monitor un punto por cada dato leido en una posición [x,y] donde la

posición x se incrementa de uno en uno a lo largo del monitor e [*y*] corresponde al dato almacenado en el arreglo.

CAPITULO V

MANUAL DE MANEJO DEL SISTEMA PARA EL USUARIO

5.1 Introducción

En este capitulo se trata de exponer ante el lector el manejo del sistema a través del software del computador. Para este punt0 es importante que haya leido los capitulos anteriores, por cuanto debe saber como conectar los equipos del sistema.

Los requerimientos del sistema, presenta las necesidades del computador para instalar el software que controlard el sistema. La guia de instalación que prepara el computador con una serie de archivos necesarios para el programa diseñado en lenguaje C++. La descripción del menú del Software que define una serie de parámetros para iniciar el sistema. Y el manejo de las opciones del menú del software que explica uno a uno, como configurar y arrancar el programa para el sistema.

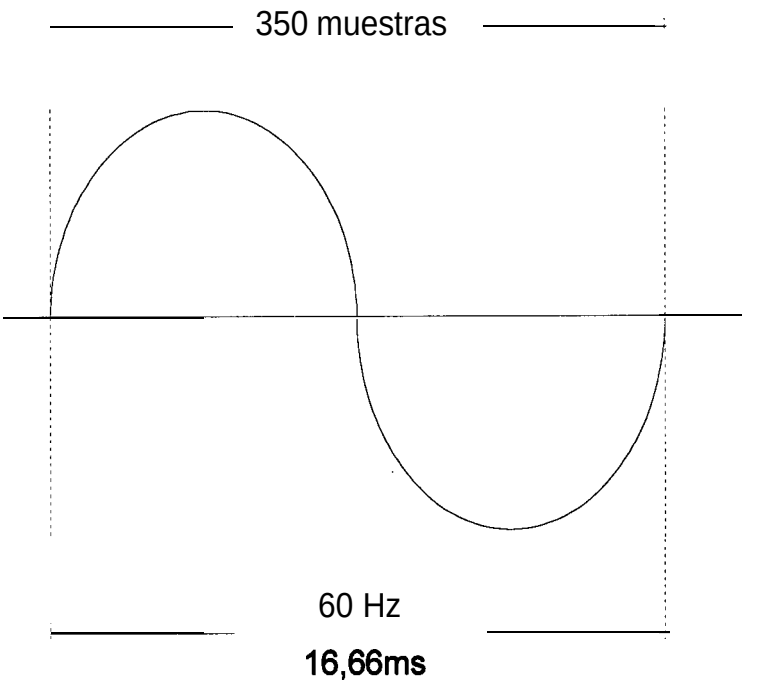
5.2 Requerimiento del sistema

El computador a utilizarse debe reunir ciertos requisitos que son mencionados en la tabla XII, y que son los minimos necesarios.

Por el tipo de muestreo que se realiza a las señales del motor DC; por cuanto se muestran las señales en el monitor con una resolución tan grande, la velocidad de la Unidad Central de Proceso (CPU) no debe ser menor a 100 Mhz. En las pruebas se llegó a establecer que se toman 700 muestras para formar 2 ondas (Fig. 31). Como sabemos, en el sistema electric0 nacional la frecuencia de la energia electrica es de 60 Hz, dando un tiempo de duración para una onda de 16,66 ms; por lo que, si dividimos las 350 muestras que forman una onda en la pantalla del sistema para el tiempo de duración de esa onda, tenemos una velocidad por cada muestra de 47 µs/muestra. Estas pruebas fueron hechas en un computador de 200Mhz con 64 Mb de RAM. Se hicieron otras pruebas en un computador de 66Mhz y 16 Mb de RAM en las cuales se observó una sola onda; por lo que se considera suficiente el procesador de 100 Mhz con 16 Mb de RAM antes mencionado.

Procesador	Pentium	100mhz
Monitor	Color	VGA
Memoria	16Mb ó más	
Puerto paralelo	Standar Parallel Port	
Sistema Operativo	DOS 6.0 o Superior	

Tabla XII Requerimientos del sistema



$$T_{muestra} = \frac{16{,}66ms}{350} = 47{,}62\,\mu s.$$

Figura 3 1 **Relación** de la velocidad de muestreo

5.3 Guia de instalación

Para la instalación de este programa se siguen los siguientes pasos:

1. Crear un directorio con el nombre de motor
2. Copiar en el directorio creado el programa **motor.exe**
3. Copiar en este directorio el archivo ***EGAVGA.BGI***

5.4 Descripción del menú del Programa

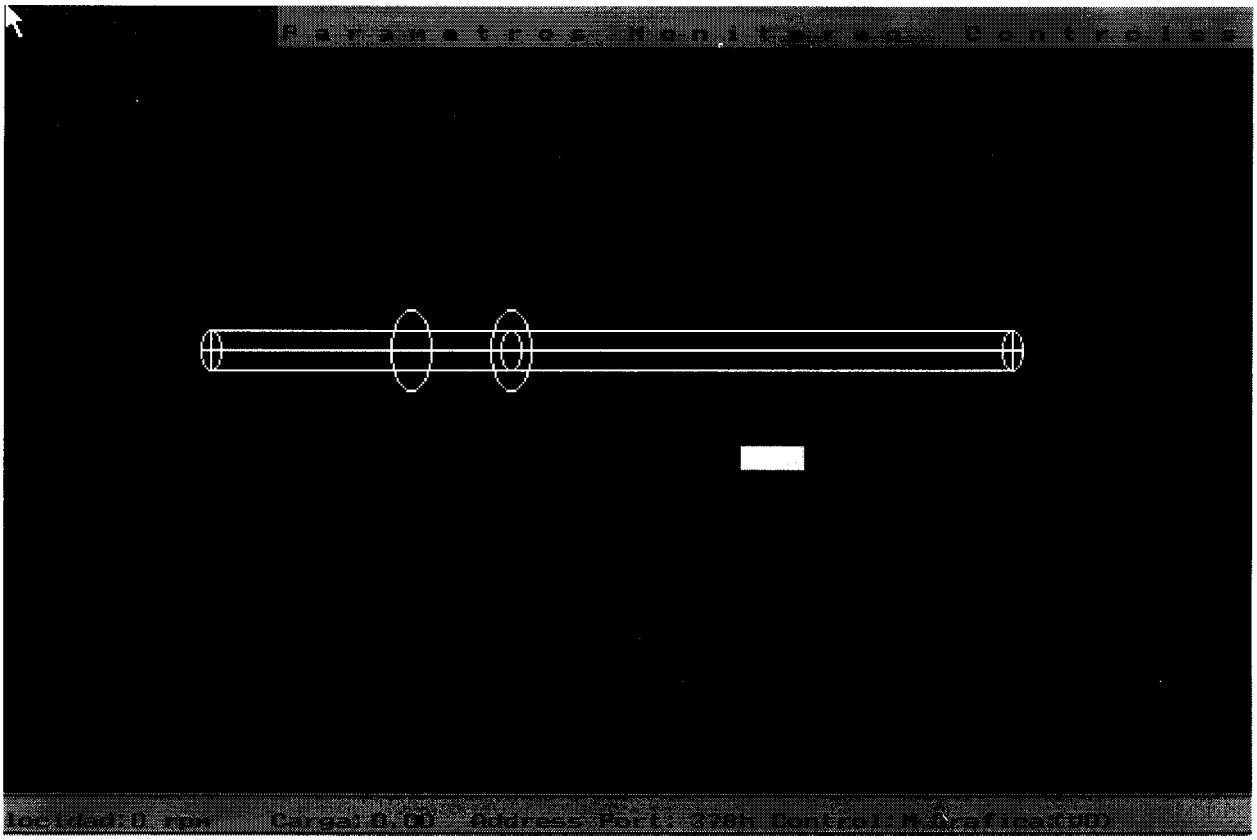


Figura 32 Pantalla inicial del programa

5.4.1 Menú pa&metros

5.4.1.1 Dirección de Puertos

Esta opción del menú contiene las direcciones base más usadas en el puerto paralelo del computador: 378h, 278h y 3BC.

Por defecto la dirección de puerto elegido es 378h. El puerto escogido se puede visualizar en la barra de estados.

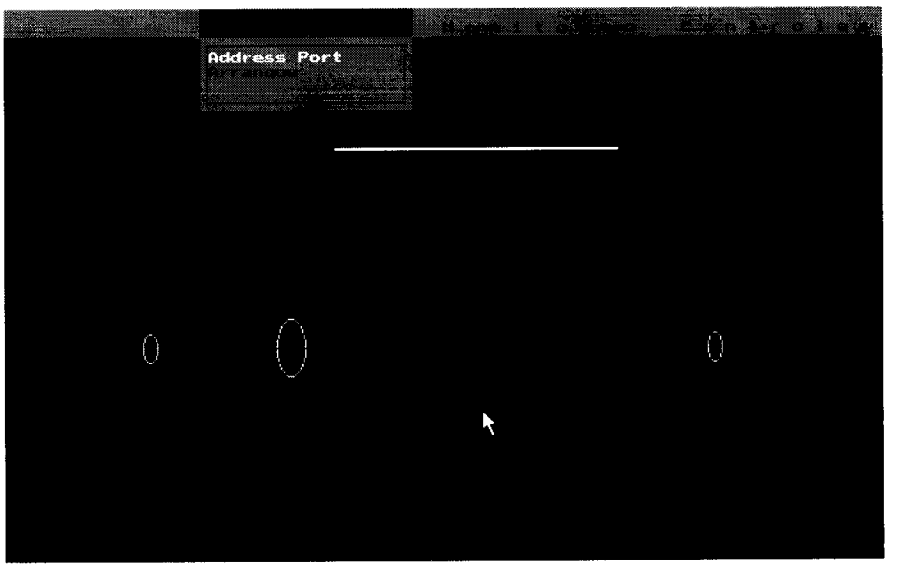


Figura 33 Menh de parámetros

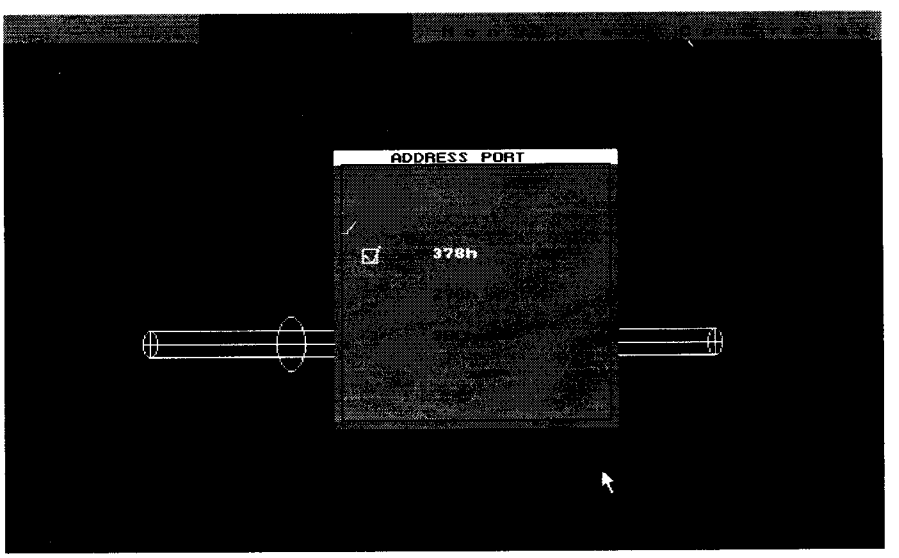


Figura 34 Selección de la direccidn de puertos

5.4.1.2 Arranque

Esta opción del menu permite escoger una velocidad de arranque con la cual el motor empezara a funcionar. Por la cantidad de corriente que maneja el motor DC en el instante del arranque, éste es limitado hasta 150 r.p.m. Los valores a escoger son 50, 100 y 150 r.p.m

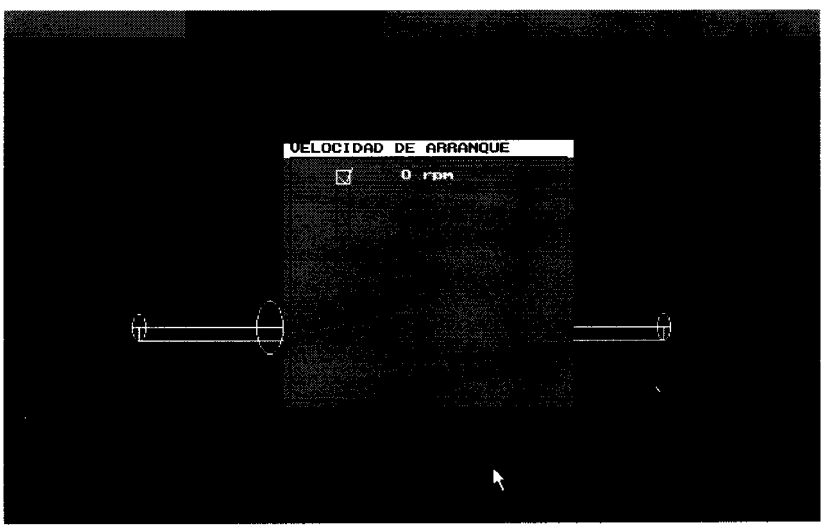


Figura 35 Selección de la velocidad inicial

5.4.2 Menú Monitoreo

5.4.2.1 Gráfico

Esta opción del men6 permite escoger el tipo de gráfica a visualizar las cuales son: Velocidad, Corriente, Voltaje y Torque.

Los tipos de ondas escogidas se verán en la barra de estado (vo=voltaje, co=corriente, ve=velocidad, to=torque). Cabe anotar que se puede hacer una selección indistinta de cada una de ellas, 0 sea, puede escoger una, dos, tres o todas a la vez.

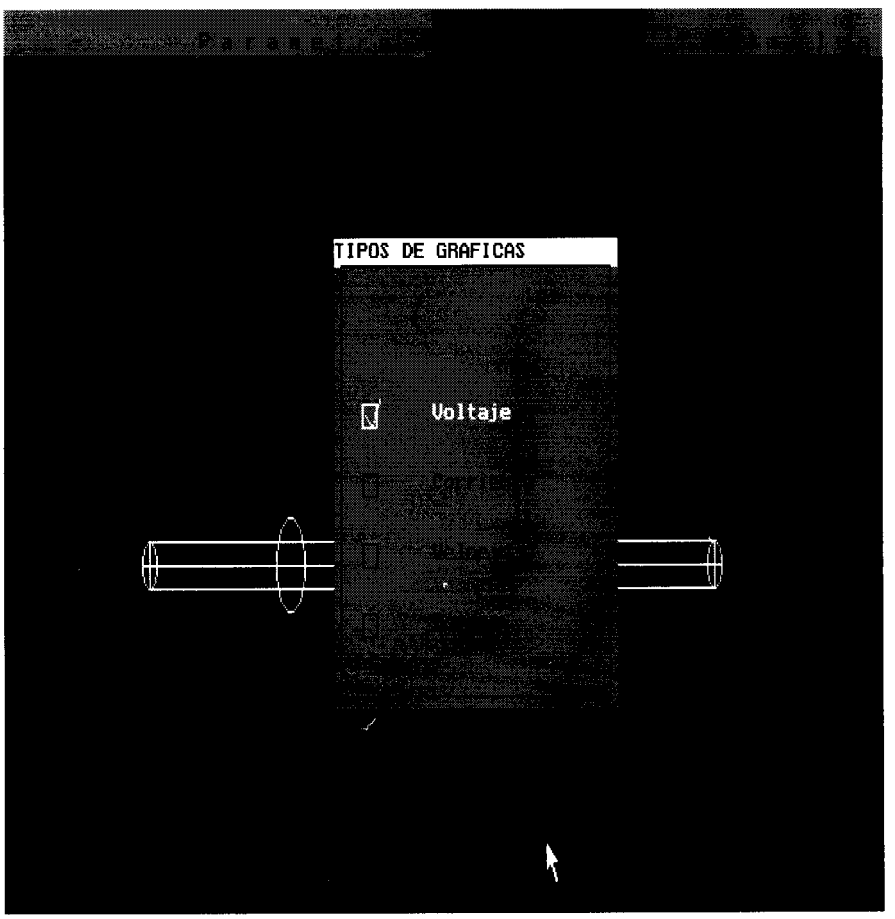


Figura 36 Seleccidn de las señales a monitorear

5.4.3 Menú controles

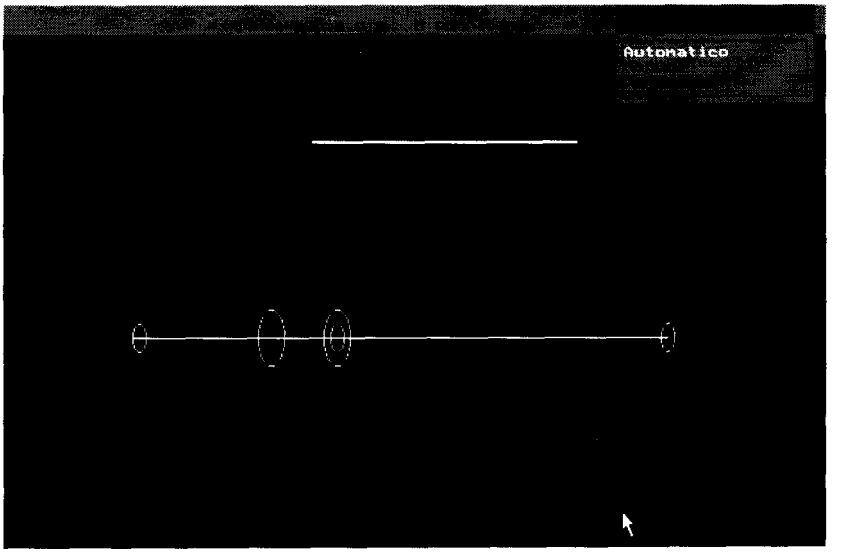


Figura 37 Menú de Controles

5.4.3.1 Operación Automática

Esta opción del menú presenta una ventana en la cual se puede ingresar un conjunto de valores para la velocidad y otro grupo de valores para la carga del motor. A Estos datos se le unen los valores de tiempo de duración para cada uno de esos valores.

El tiempo *t* se refiere al número de despliegues de pantalla que deben transcurrir para que se realice el cambio de velocidad y de carga, así por ejemplo si se coloca *t*=5, se espera ver 5 pantallas para ver cambiar la velocidad y la carga.

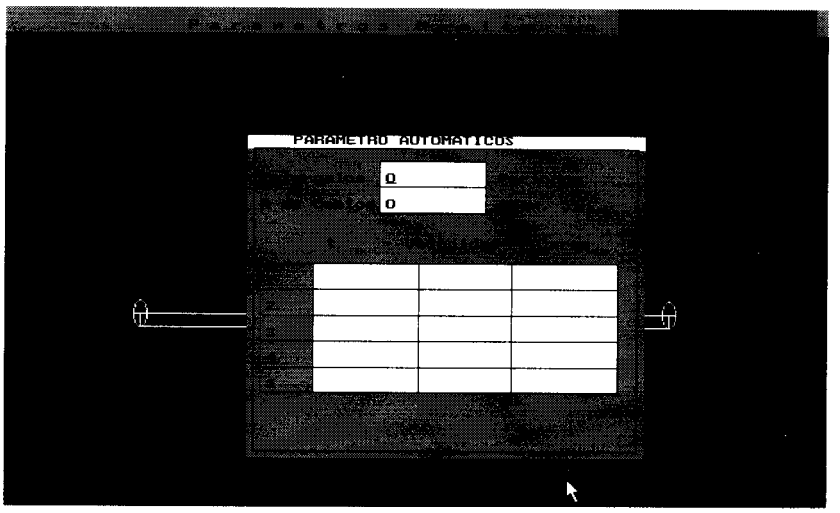


Figura 38 Operación automática

En el ejemplo de la Fig. 39, se muestra como se crea un control automático de 5 velocidades y 5 cargas que van a ser ejecutadas en un interval0 uniforme de 50 despliegues de ondas

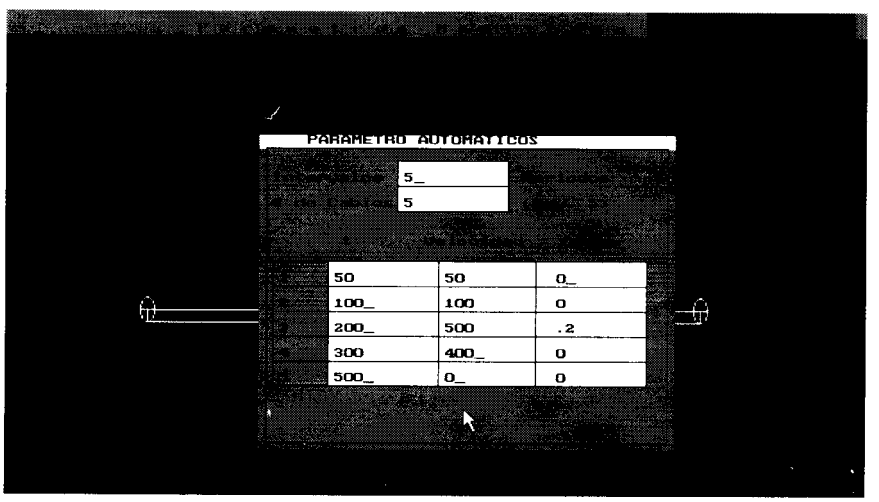


Figura 39 Ejemplo para una operación automática

5.4.3.2 Operación Manual

Esta opción del mend permite anular el modo automático del programa, permitiendo que el usuario tenga control del envio de órdenes al motor

5.4.3.3 Controlar

Con esta opción el usuario ingresa a una interface interactiva, en la cual visualiza, las ondas de corriente, voltaje, velocidad, torque que emite el motor, además, el usuario podrá controlar el movimiento del motor enviándole variaciones de velocidad o carga.

Esta pantalla se divide en 4 partes:

1. Una porción de la pantalla que permite visualizar las ondas de voltaje, corriente, velocidad, torque.

2. La franja de controles que permite manipular el motor y observar la respuesta del mismo.

3. La franja de parámetros que permite saber la velocidad con que inicio el motor, la carga con que inicio el motor, la dirección del puerto paralelo

por donde se envian los datos al motor, el modo de control (M=manual, A=automático), y el tipo de gráfico que se presenta en la pantalla.

4. La franja o barra de mensajes que da a conocer las principales teclas con las cuales manejar el programa.

5.5 Manejo de las opciones del menú del software

5.5.1 Selección de dirección de puerto

Para la selección de las direcciones del puerto, se escoge en el menú par&metros, la opción ***dirección del puerto***, cuando se presione la tecla **<enter>** en este submenu, se presentará en el centro de la pantalla una cuadro con 3 direcciones opcionales, el usuario podrá desplazarse en cada una de ellas por medio de las teclas up(↑) y down(&), cuando ha escogido la dirección deseada presiona la tecla **<enter>**, para salir de esta pantalla debe presionar la tecla escape <***esc***>.

En la barra de estado que se encuentra en la parte inferior de la pantalla, se visualizará la dirección del puerto que se ha escogido.

5.5.2 Selección del tiempo de aceleración

La selección de la velocidad de arranque se realiza escogiendo la segunda opción del menú par&metro. Cuando se escoge esta opción presionando la tecla <enter>, se presenta un cuadro en el centro de la pantalla, la cual contiene las velocidades de arranque que fueron tomadas experimentalmente; el usuario podrd desplazarse entre cada una de ellas por medio de las teclas up(↑) y down(k). La velocidad escogida será determinada presionando la tecla <enter>; al salir del cuadro presionando la tecla <esc>, la velocidad escogida se visualizará en la barra de estado que se encuentra en la parte inferior.

5.5.3 Selección del tipo de seiial

La selección del tipo de señal se realiza escogiendo la primera opción del menú parámetro. Cuando se escoge esta opción presionando la tecla **<enter>**, se presenta un cuadro en el centro de la pantalla, la cual contiene 4 tipo de señales que se pueden visualizar, el usuario podrd desplazarse entre cada una de ellas por medio de las teclas up(↑) y down(J). El tipo de señal escogida será marcada cuando se presione la tecla **<enter>**, en este cuadro se puede seleccionar hasta las cuatro opciones a la vez; al salir del cuadro presionando la

tecla <***esc***>, los tipos de señales escogidos se visualizara en la barra de estado que se encuentra en la parte inferior.

5.5.4 Control Automdtico

Para que el programa haga un control automático del motor el usuario debe planificarlo, y para esto debe seleccionar en el men6 controles, la opción automático.

Esta opción presenta un cuadro en el centro de la pantalla por medio del cual se pide ingresar la siguiente información.

Intervalos: Este parametro permite variar la velocidad y carga en un tiempo uniforme

T: Es el tiempo que va a durar cada valor de velocidad y carga; este parámetro es ignorado cuando la variación es uniforme es decir cuando el interval0 > 10 que es el minim0 tiempo para que se estabilice el motor bajo un conjunto de par&metro de velocidad y torque.

 # de cambios: Este parametro nos dice cuantos cambios vamos a ingresar

Velocidad: Es el arreglo en donde se van a colocar los valores de las velocidades que van a cambiar, en esta columna sólo se aceptan valores numéricos enteros, y la cantidad a ingresar esta limitada por <# de cambios>.

Carga: Es el arreglo en donde se van a colocar los valores de las cargas que van a cambiar, en esta columna se aceptan valores de tipo ***double*** y la cantidad a ingresar se limita por <# de cambios>.

5.5.5 Control interactivo del motor

Después de fijar todos los parametros iniciales, se debe escoger en el menú control la opción controlar **on**, esta opción nos presenta una pantalla en la cual se dibuja continuamente las señales indicadas anteriormente.

En esta pantalla encontramos 5 controles en la parte inferior, 4 de los cuales permiten controlar el motor.

El primer control, es una flecha con dirección hacia arriba, la cual, nosotros manipulamos con la tecla <↑> y así aumentamos el valor que se muestra junto a esta, el cual es

la velocidad que enviamos al motor, este increment0 se realiza en orden de una unidad hasta llegar a 1000 rmp.

Si presionamos la tecla ***<Page*** VP> cuando esta seleccionado este control, la velocidad se incrementa en orden de 50 rpm.

CONCLUSIONES Y RECOMENDACIONES

Una vez concluido el proyecto y obtenidos los resultados se observa, que trabajar con el puerto paralelo no es tan complicado. Si agregamos a ello, los conocimientos en la programación del lenguaje C++ bajo DOS; se podria desarrollar no solo un controlador para un motor sino, algún otro tipo de controlador con diferente propósito; por ejemplo, aplicable a la seguridad en el hogar o maquinaria en alguna industria.

El software del proyecto, estuvo planificado realizarlo en Windows 95, bajo el lenguaje Visual C++. Lo complicado es la programación, la cual demanda un mayor conocimiento para los estudiantes de electronica industrial. Esta complicación condujo a emplear Visual Basic 5.0. Todo marchaba bien, pero en una de las pruebas se observó que Windows 95 en su sistema interno, realiza algún tipo **de** inspección sobre los perifericos conectados al computador y desconfigura completamente al puerto paralelo en un momento determinado, tiempo que, no permitia mantener los cambios en los par&metros del motor. Por este motivo se decidió regresar al lenguaje Ctt 3.0 bajo DOS.

En las investigaciones realizadas, conocimos que a más del puerto SPP (Puerto Paralelo Standard), existen otros puertos diseñados bajo la norma IEEE 1284, que crea un puerto paralelo real de dos vías, esto es, los 8 bits del registro de DATA son bidireccionales. Esto sirve para simplificar el tipo de hardware que se requiera diseiiar. La

programación era igualmente complicada como el Visual C++, por lo que se decidió trabajar con el puerto SPP.

Lo interesante del proyecto es el trabajo con el puerto paralelo, realizando control sobre algún dispositivo externo de tipo electrónico.

Una de las recomendaciones seria el trabajar con ADC de varios canales, y realizar un controlador externo para la parte digital del sistema, de esta manera se hace más flexible o universal la aplicación del módulo supervisor, pudiendose crear con mayor facilidad otros programas con propósitos diferentes.

Así mismo, de las investigaciones realizadas, existen softwares para este propósito, diseñados en Windows 95 que superan el problema de la configuración del puerto, por lo cual, se recomienda investigar las técnicas que hacen esto posible. En la mayoria de los softwares mencionan el uso de una tarjeta de adquisición en el bus de datos, que manejan de manera más rápida y confiable la transferencia de información a través del puerto.

Por último, recomendamos investigar en Internet, pues la información es amplia y actualizada. Precisamente navegando en diferentes direcciones, se obtuvo la mayor cantidad de información.

APENDICE A

CONVERTIDOR ANALOGICO DIGITAL (ADC) 08161

El ADC08161 pertenece a la familia lógica Semiconductor de oxido metálico complementario (CMOS), este convertidor de 8 bits emplea la técnica de conversion multipasos, ofrece un tiempo de conversion de 500 ns; también posee internamente un circuito de muestreo y sostenimiento de referencia, disipando apenas 100mW.

El ADC08161 ejecuta una conversion con un voltaje estimado de 2 bits que son generados por los 2 bits más significativos (MSB"), y dos voltajes rápidos estimados de baja resolucibn que los generan los 6 bits menos significativos (LSB). Además puede ejecutar una conversion precisa de total escala para señales de entrada desde frecuencias DC (0 Hz) hasta frecuencias de 300 KHz. Puede adaptarse fácilmente hacia un microprocesador, y no necesita reloj externo.

DESCRIPCION DE LOS PINES

WR / RDY

Modo WR-RD (aplicando un voltaje alto al pin MODE)

WR: Con un voltaje bajo en CS la conversion es iniciada cuando WR es falso, voltaje alto. El resultado digital estaria habilitado en la salida al final de al conversion. (fig. 2 , 3, 4)

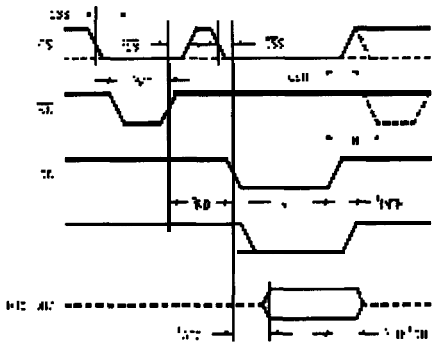


FIGURE 2. $\overline{WR}$-$\overline{RD}$ Mode with $t_{WR} \leq t_{WRL}$ (Mode Pin is High)

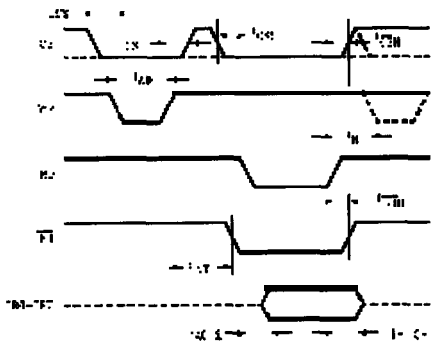


FIGURE 3. $\overline{WR}$-$\overline{RD}$ Mode with $t_{WR} > t_{WRL}$ (Mode Pin is High)

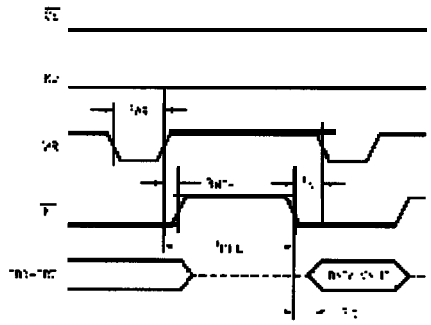


FIGURE 4. $\overline{WR}$ -$\overline{RD}$ Mode Reduced Interface System Connection with $\overline{CS}$ = $\overline{RD}$ = 0 (Mode Pin is High)

Modo RD (aplicando un voltaje bajo al pin **mode)**

RDY: Esta es una **salida** de drenador abierto. RDY entraria en estado verdadero, voltaje bajo, **después** de que CS **esté** en bajo, y retornaria al estado falso, voltaje alto, al final de la conversion.

MODE

Mode: Entrada de **selección** de modo (RD o WR-RD) .Este pin es puesto en un voltaje bajo por medio de un sumidero de corriente de 50 **μA** cuando se deja desconectado.

Modo RD.

Es seleccionado si el pin MODE se deja desconectado o si externamente es puesto a un voltaje bajo. Una conversion se realiza teniendo a RD en bajo hasta que el dato aparezca en la **salida.**

Modo WR-RD

Este es seleccionado cuando un voltaje alto es aplicado en el pin MODE. Una conversion se inicia cuando WR esta a un voltaje bajo y **los** datos se obtienen usando RD.

RD

Modo WR-RD (Aplicando un voltaje alto en el pin MODE).

Este es una entrada de **lectura** activada con un voltaje bajo. Con un voltaje bajo aplicado en el pin CS, el dato de **salida** debe ser activado cuando RD vaya al estado verdadero, voltaje bajo. **(fig. 2 , 3, 4)**

Modo RD

Este es seleccionado aplicando un voltaje bajo en el pin MODE. Con el pin CS en bajo. Una conversion se inicia cuando RD **está** en un estado verdadero, voltaje bajo. El dato de **salida aparecerá** en DBO-DB7 al finalizar la conversion. (fig. **1,** 5)

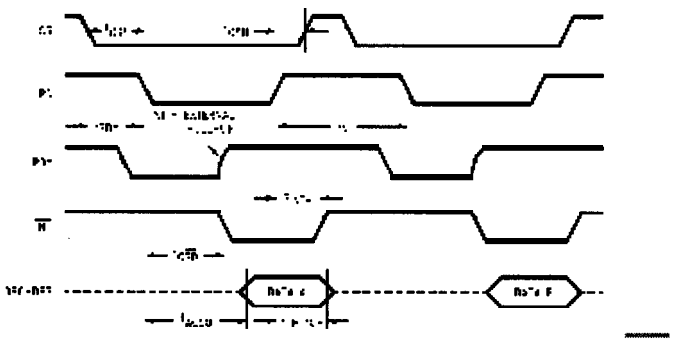


FIGURE 1. $\overline{RD}$ Mode (Mode Pin is Low)

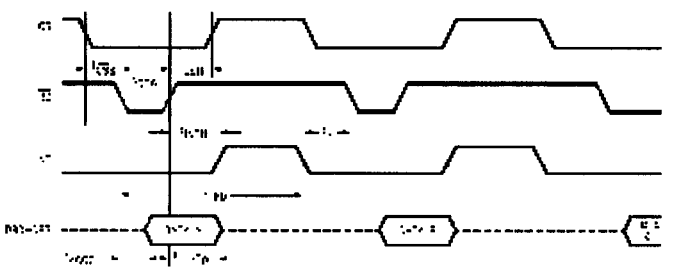


FIGURE 5. $\overline{RD}$ Mode (Pipeline Operation); t_{wow} must be between 200 ns and 400 ns. (Mode Pin is Low)

INT

Esta es una salida activa con voltaje en bajo, la cual indica que la conversion se ha realizado y el dato está en reforzador de salida. INT se borra cuando RD va a estado falso, voltaje alto.

GND

Este es el pin de tierra del suministro de voltaje. El pin de tierra debe ser conectado a un punto de referencia de tierra "neta".

VREF- , VREF+

Estas son las entradas de referencia de voltaje. Pueden ser puestas a un voltaje referencial entre GND -50mV y (V+) +50mV, pero VREF+ debe ser mayor que VREF- . Idealmente una entrada de voltaje igual a VREF- produce una salida de código cero , y una entrada de voltaje mayor que (VREF+) – 1.5 LSB produce una salida de código 255.

Para el ADC08161 una entrada de voltaje que exceda aV+ por mas de 100mV o si está por debajo de GND por mas de 100mV provocaria conversiones erróneas.

c s

Esta es una entrada que se activa con voltaje bajo y sirve para activar al chip. Un voltaje bajo aplicado a este pin habilita las entradas RD y WR.

OFL

Indica sobrecarga a la salida. Si una señal analógica de entrada es mayor que VREF+, OFL al final de la conversion estaria con un voltaje bajo , estado verdadero . Este pin se lo puede emplear para conectar dos ADC en cascada para lograr una mayor resolución. Esta salida está siempre activa, es decir, no entra en estado de alta impedancia.

V+

Entrada de voltaje positivo del suministro de voltaje. Su valor nominal de operacidn es +5v. Este deberia ser "puenteado" con un capacitor de Tantalio (burbuja) de 10μF en paralelo con un capacitor de cerámica de 0.1 μF. La longitud del conductor debe ser lo mas corta posible.

VREFOUT

Es un sumidero de referencia interno igual a 2.5V, el cual está disponible en este pin. Use un capacitor de 220μF "puenteado" entre este pin y tierra analógica.

INTERFASE DIGITAL

El ADC08161 tiene dos modos interface las cuales son seleccionadas conectando un voltaje alto o bajo en el pin MODE.

MODO RD

Con un voltaje bajo aplicado en el pin MODE, el wnvertidor es configurado en modo de lectura. En esta configuración, una conversion se realiza llevando a RD a un voltaje bajo, y manteniendolo hasta que la conversion se finalice y el dato aparezca a la salida, esto toma tipicamente unos 655 ns, y INT se va a estado verdadero (voltaje bajo) cuando finaliza la conversion. Un tipico retardo de 50 nS entre el cambio de estado falso (voltaje alto) de CS, después del final de la conversion, y el inicio de la siguiente conversion. La salida RDY va a estado falso (voltaje bajo) después que CS va a un estado verdadero (Voltaje bajo) , y vuelve a estado verdadero al final de la conversion. Esta señal puede ser usada para indicar que el convertidor está ocupado o como seiial para envio y respuesta.

MODO WR-RD

El ADC08161 está en modo WR-RD con el pin MODE puesto a un voltaje alto. Una conversion es iniciada cuando el pin WR va a estado falso (voltaje alto). Hay dos opciones de leer el dato de salida las cuales se relacionan con el tiempo que emplea su interface en mandar las señales.

Si se emplea un sistema de interface de reconocimiento de señales, usted puede esperar por la salida INT (voltaje bajo) para ir a leer el dato. Tipicamente INT va a estado verdadero (voltaje bajo) 690 ns después que WR entra en voltaje alto. Sin embargo si el tiempo de conversion requerido es bien pequeño no es necesario esperar por INT y se puede realizar una lectura de datos después de solo 350 ns.

Si RD es puesto a un voltaje bajo antes que INT vaya a un estado verdadero (voltaje bajo) , INT. Inmediatamente iria a voltaje bajo y el dato apareceria en la salida. Este es el modo de operación más rápido ($t_{RD} \leq t_{INTL}$) con un tiempo de conversion, incluyendo el tiempo de acceso de datos, de 560 ns.

CON INTERFACE REDUCIDA

Conectando CS y RD a un voltaje bajo, entonces se usa WR como control de inicio de conversion, para aplicaciones que requieran reducida interface digital mientras operan en modo WR-RD, el dato estaria en la salida aproximadamente 705 ns después que WR haya pasado a estado falso (voltaje alto).

ENTRADAS DE REFERENCIA

Las dos entradas de VREF de los ADC08161 son totalmente diferenciales y definen el rango de entrada desde cero a escala wmpleta del ADC08161. Esto le permite al diseñador variar el espacio que abarca la entrada analbgica ya que este rango será equivalente a la diferencia de voltaje entre VREF+ y VREF-. Transductores que tienen salidas de voltajes minimos sobre GND también pueden ser compensado conectando VREF- a un voltaje que es igual a este voltaje

minimo. Reduciendo VREF (VREF = VREF+ – VREF-) a menos de 5V, la sensibilidad del convertidor puede aumentarse (es decir, si VREF = 2.5V, entonces 1 LSB = 9.8 mV).

El arreglo de la referencia también facilita la operación radial y en algunos casos para los que el suministro de poder puede usarse por el poder del transductor así como el VREF de la fuente. El funcionamiento radial es logrado conectando VREF- a GND y conectando VREF+ y la entrada de suministro de poder de un transductor a V +.

La exactitud de los ADC08161 degrada cuando VREF+ – |VREF-| es menor de 2.0V. El voltaje a VREF- pone el nivel de la entrada que produce unas salidas digitales de todo cero. A través de VIN no está lo diferencial, el diseño de la referencia permite una capacidad de entrada aproximadamente diferencial para algunas aplicaciones de medida. La Figura 7 presenta una posible configuración diferencial.

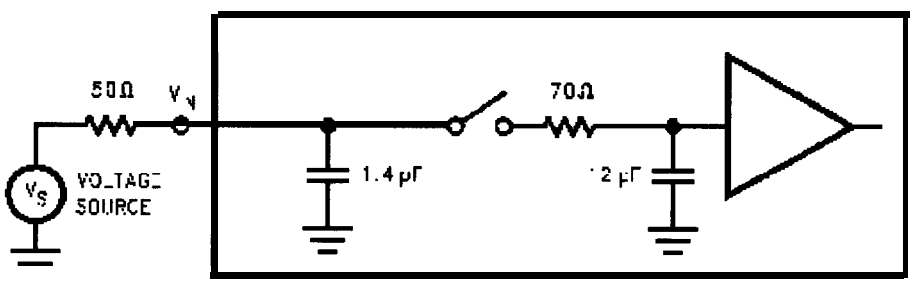


FIGURE 7. ADC08161 Equivalent Input Circuit Model

Debe notarse que, mientras las dos entradas de VREF son totalmente diferencial, la salida digital será cero para cualquier voltaje de entrada analógico sí VREF- ≥ VREF+.

MUESTREO Y SOSTENIMIENTO INHERENTE

Un beneficio importante de la arquitectura de la entrada de los ADC08161 es el muestreo y sostenimiento inherente y su habilidad para medir señales de velocidad relativamente altas. En un convertidor no muestreador, sin tener en cuenta su velocidad, la entrada debe permanecer estable a por lo menos ½ LSB a lo largo del proceso de conversion si es completa la exactitud a ser mantenido. Por consiguiente, para muchos seiiales de velocidad altas, esta señal debe ser externamente muestreada y sostenerse estacionaria durante la conversion.

Los ADC08161 son convenientes para los sistemas DSP porque controlan directo el S/H a través de la señal WR. La serial de entrada WR le permite al A/D ser sincronizado a un sistema de muestreo proporcional DSP o a otro ADC08161.

Los ADC08161 pueden realizar conversiones exactas de señales de entrada de maxima escala a frecuencias desde DC a más de 300 kHz (ancho de banda máximo) sin la necesidad de un externo.

REFERENCIA DE SUMIDERO INTERIOR

Los ADC08161 tienen un sumidero de referencia interior de 2.5V que puede usarse como la entrada de VREF+. Una combinación paralela de un condensador cerámico de 0.1 µF el y un condensador de tantalio de 220 µF debe usarse para desviar el pin VREFOUT. Esto reduce posible recogida de ruido que podria causar errores de la conversion.

ESQUEMA, TIERRAS, Y PUENTEADO

Para asegurar conversiones rápidas, exactas del ADC08161, es necesario usar técnicas apropiadas de esquema de circuitos para tarjetas. Idealmente, la referencia de tierra del convertidor analógico digital debe ser de baja impedancia y libre del ruido de otras partes del sistema. Los circuitos digitales pueden producir una gran cantidad de ruido en su retorno a tierra y, por consiguiente, debe tener sus propias líneas de tierra separadas. El mejor funcionamiento se obtiene usando líneas de tierra separadas debe ser prevista para las partes digitales y analógicas del sistema.

Las entradas analógicas deben aislarse de las señales ruidosas localizadas para evitar tener falsas señales acopladas a la entrada. Cualquier componente externo (ej un condensador de filtro de entrada) conectado por las entradas debe retornar a un punto de tierra muy “limpio”.

Los ADC08161 incorrectamente conectados con tierra puede tener una reducida exactitud de la conversión. El pin V+ de la fuente, VREF+, y VREF- (si no conectó con tierra) debe desviarse con una combinacidn paralela de un condensador cerámico 0.1 µF y un condensador de tantalio de 10 µF situados tan cerca como sea posible a los pines que usan pistas para corto circuito en algunas tarjetas. Vea Figura 8, 9.

APENDICE B

CONVERTIDOR DAC

CARACTERISTICAS INDUSTRIALES STANDARD DE LOS PINES DE SALIDA

Balance total ±1Ov con vcc = ±12vdc

Las entradas digitales son ffl - y cmos-compatible

Especificaciones garantizadas con fuentes de ±12v y ±15v

±1/2lsb maximo nolineal :0°c a +70°c

Tiempo estableciendo: 4ms max a ±0.01% de escala total

Garantizado monotonicity:0°c a +70°c

Dos opciones de paquete: plástico amoldado económico y hermético lado-soldado cerámico

DESCRIPCION

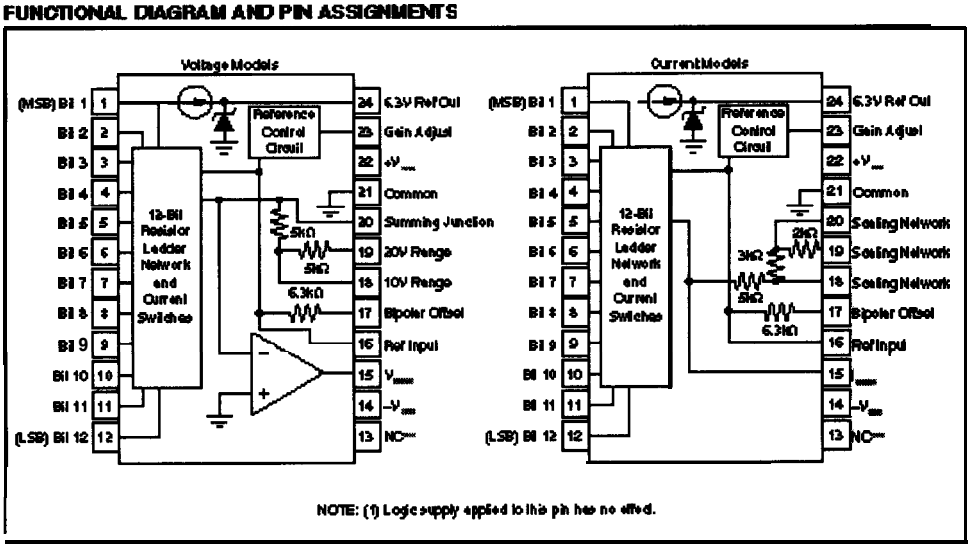
Este convertidor digital-analogico monolitico es pin a pin equivalente a la normas industriales DAC80 introducidos por Burr-Brown. El diseño de este chip incluye el amplificador de salida y proporciona una referencia muy estable capaz de proporcionar hasta 2.5mA a una carga extema sin degradación de funcionamiento del DAC.

Este convertidor usa técnicas de circuito probados para proporcionar funcionamiento exacto y fiable sobre variaciones de la temperatura yde suministm de poder. El uso de un diodo zener aterrizado como la base para la referencia intema conttibuye para la alta estabilidad y el bajo ruido del dispositivo. Métodos avanzados de arreglos laser da resultado de precisión en la coniente de salida y resistencias de retroalimentación en la salida del amplificador , así como bajo en-ores de linealidad integral y diferencial. El diseño innovador del circuito permite a los DAC80 operar a voltajes tan bajo como ill .4V sin la pérdida en actuación o exactitud sobre cualquier rango de voltaje de salida. La baja discipación de potencia de este chip de 118-mil por 121-mil resulta la más alta fiabilidad y el término mas largo de estabilidad .

El Burr-Brown ha reforzado la fiabilidad más alla del monolitico DAC80 ofreciendo un hermético, lado-soldado, de paquete cerámico. Además, la facilidad de uso ha sido reforzada eliminando la necesidad para un suministro de poder lógico de +5V .

Para aplicaciones que requieren fiabilidad y el bajo costo , los DAC80P en un paquete de plástico amoldado ofrece la misma función eléctrica para sobre temperaturas como el modelo cerámico. Los DAC80P solo están disponibles con salida de voltaje.Para diseños que requieren un rango de temperatura más ancho, vea el Burr-Brown modelos DAC85H y DAC87H. Los diagramas son presentados en la figuras siguientes.

FUNCTIONAL DIAGRAM AND PIN ASSIGNMENTS



PROTECCION

Una opción de protección está disponible para los modelos indicados en la Información de Clasificación. Quemadura-en duración es 160 horas al mdximo grado de temperatura especificado al que opera (o combinación equivalente de tiempo y temperatura).

Todas las unidades se prueban después quemadura-en asegurar que esas specificaciones de calidad se reline. Para pedir quemadura-en, agregue "–BI" al número ejemplar bajo.

DISCUSION DE ESPECIFICACIONES DE ENTRADA DE CODIGOS DIGITALES

Los DAC80 aceptan códigos de entrada digitales binarios complementatios. El modelo de CBI puede ser conectado para ser usado para cualquier de los tres códigos complementarios: CSB, COB , o CTC (vea tabla).

EXACTITUD

DIGITAL INPUT	ANALOG OUTPUT		
MSB ↓ LSB ↓	CSB Complementary Straight Binary	COB Complementary Offset Binary	CTC* Complementary Two's Complement
000000000000	+Full Scale	+Full Scale	–1 LSB
011111111111	+1/2 Full Scale	Zero	–Full Scale
100000000000	1/2 Full Scale –1LSB	–1LSB	–Full Scale
111111111111	Zero	–Full Scale	Zero

NOTE: (1) Invert the MSB of the COB code with an external inverter to obtain CTC code.

TABLE I. Digital Input Codes.

La linealidad de un convertidor DAC es la verdadera medida de su funcionamiento. El error delinealidad de los DAC80 se especifica sobre su rango entero de temperatura . Esto significa que el salida analógica no variará por más de ±1/2LSB, mdximo, desde una línea recta ideal dibujada entre los puntos extremos (entradas todos "1"s y todos "0"s) sobre el rango de temperatura especificado de 0°C a +70°C.

El error de linealidad diferencial de un convertidor DAC es la desviación de un ideal 1LSB cambio de voltaje de un estado del salida adyacente al próximo. Una especificación del error de linealidad diferencial de ±1/2LSB significa que los tamaños de paso para la salida de voltaje pueden ir de 1/2LSB a 3/2LSB cuando la entrada cambia de un estado de la entrada adyacente al próximo.

Monotonicity sobre un rango de 0°C a +70°C se garantiza en los DAC80 para asegurar que el salida analógico aumentará o permanecerá el mismo para la creciente de los códigos digitales de entrada.

MODELOS DE SALIDA DE VOLTAJE Y CORRIENTE

Se especifican tres tiempos del establecimiento a ±0.01% de rango de escala completa (FSR); dos para los máximos cambios de rango de escala completa de 20V, 1 OV y uno para el cambio de un 1 LSB . El cambio de 1 LSB es medido en el mayor acarreo (0111 . ..1 1 a 1000 . ..OO). el punto al que el peor caso establecido de tiempo ocurre.

Se especifican dos tiempos del establecimiento a ±0.01% de FSR. Cada uno se da para modelos de corriente conectados con dos cargas resistiva diferentes: 10W a 100W y 1000W a 1875W. Se mantienen resistencias interiores para conectar resistencias de carga nominales de aproximadamente 1000W a 1800W para voltaje del salida en el rango de ±1V y 0 a -2V (vea Figura 11 y 12).

$$V_{OUT} = -2mA\ [(R_L \times R_O) \div (R_L + R_O)]$$

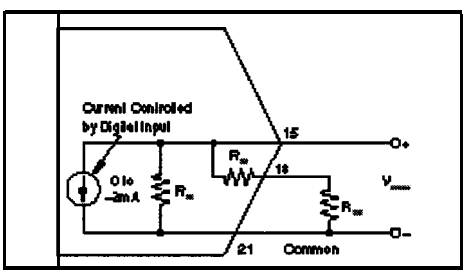


FIGURE 11. Current Output Model Equivalent Circuit Connected for Unipolar Voltage Output wib Resistive Load.

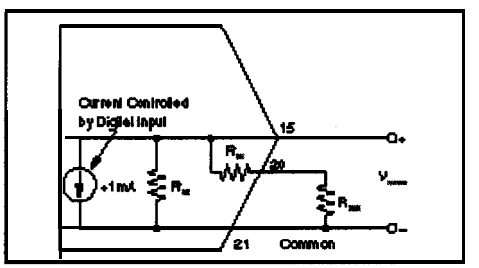


FIGURE 12. Current Output Model Connected for Bipolar Output Voltage with Resistive Load.

COMPLIANCIA

El voltaje de compliancia es el máximo voltaje de balance permitido en el nodo de salida de corriente para mantener la exactitud especificada. El máximo voltaje de compliancia de todos los modelos del salida de corriente es ±2.5V. El Máximo rango de voltaje seguro es ±1V y 0 a -2V (vea Figura 11 y 12).

REFERENCIA DEL SUMINISTRO

Todos los modelos DAC80 se proporcionan con un suministro de voltaje de referencia interior 6.3V . Este voltaje (pin 24) tiene una tolerancia de ±1% y debe conectarse a la Referencia de la entrada (pin 16) para el funcionamiento especificado. Esta referencia puede también usarse extemamente , pero la corriente de drenaje extema se limita a 2.5mA. Si una carga variable es manejada, un buffer amplificador extemo se recomienda para manejar la carga para aislar desplazamiento bipolar de las vatiaciones de carga. Ajuste de Ganancia y el desplazamiento bipolar debe hacerse bajo las condiciones de carga constantes.

COMPATIBILIDAD DE ENTRADAS LOGICAS

DAC80 entradas digitales son TTL, LSTTL y 4000B, 54/74HC CMOS compatible.El cambio en el umbral de entrada permanece en el umbral de TTL sobre el rango entero del suministro . Lógica "0" en la entrada de coniente sobre temperatura es bastante baja para permitir manejar DAC80 directamente de los salidas de 4000B y 54/74C dispositivos de CMOS.

INSTRUCCIONES DE OPERACION Y CONEXIONES DEL SUMINISTRO DE PODER

Conecte el suministro de poder de voltajes como mostrado en Figura 3. Para el funcionamiento óptimo y rechazo del ruido, Al poder de suministro deben agregarse condensadores de desacoplamiento como los mostrado. Estos condensadores (1mF tantalum) debe localizarse cerca de los DAC80.

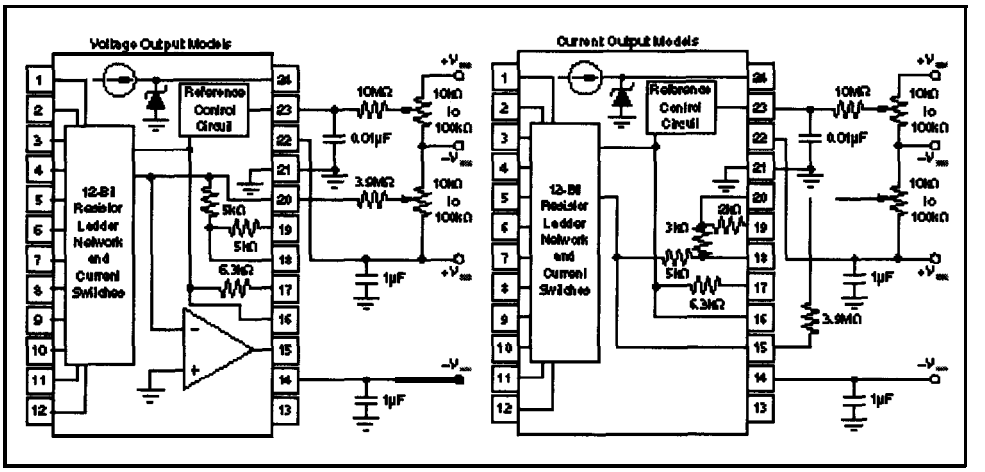


FIGURE 3. Power Supply and External Adjustment Connection Diagrams.

FUNCIONAMIENTO A ±12V

Todos los DAC80 modelos pueden operar encima del rango entero de suministro de poder de 11.4V a ±16.5V. Incluso con niveles del suministro drop-ping a ±11.4V, los DAC80 pueden escursionar un rango completo ±10V , suministrando una coniente de carga que se limita a ±2.5mA. Con suministro de poder mayor que ±12V, la salida DAC80 puede ser cargada hasta ±5mA. Para cambios de salida de ±5V o menos, la coniente de salida es ±5mA, minimo, encima del rango entero VCC.. Ninguna resistencia se necesita de +VCC para fijar 24, como se necesitó con versiones hibridas anteriores de Z de DAC80. Existiendo ±12V aplicaciones a las que están convirtiendose el monolitico DAC80 deben omitir la resistencia para fijar 24 para asegurar funcionamiento apropiado.

DESPLAZAMIENTO EXTERNO Y AJUSTE DE GANANCIA

El desplazamiento y ganancia pueden ser arregladas instalando Desplazamiento extemo y potentiometers de Ganancia. Conecte estos potenciómetros como mostrado en Figura 3 y ajuste como describió debajo. TCR del potentiometers debe tener 100ppm/°C o menos. Las resistencias 3.9MW y 10MW (20% carbono o mejor) deben estar localizada cerca de los DAC80 previniendo recoger ruido. Si no es conveniente usar estas resistencias de valor altas, una red equivalente "T" , como la mostrada en Figura 4, puede sustituirse.

Existiendo aplicaciones a las que están convirtiendo el monolitico DAC80 deben cambiar la ganancia del arreglo de resistencia en el pin 23 de 33MW a 10MW para asegurar rango de ajuste suficiente. El pin 23 es un punto de alta impedancia y un condensador cerámico de 0.001 µ F a 0.01 µ F debe conectarse de este pin al pin común (pin 21) para prevenir recogida del ruido. Refiérase a la Figurar 5 para la relación de Desplazamiento y ajustes de Ganancia para funcionamiento unipolar y bipolar del DAC .

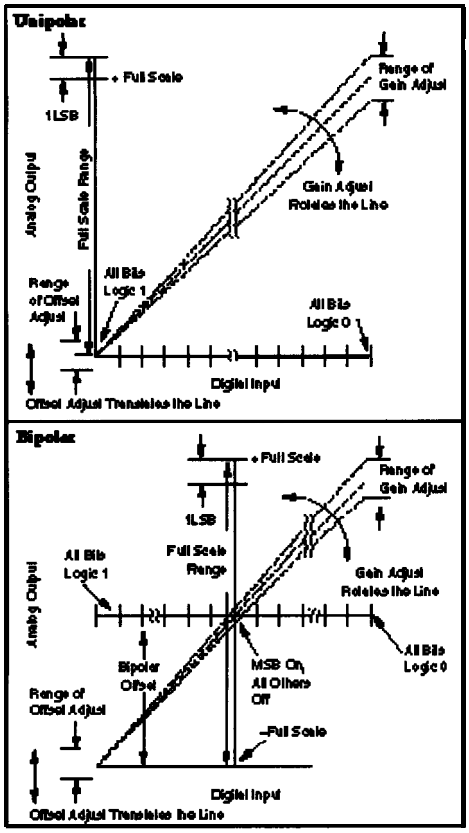


FIGURE 5. Relationship of Offset and Gain Adjustments for a Unipolar and Bipolar D/A Converter.

AJUSTE DE DESPLAZAMIENTO

Para las configuraciones unipolar (CSB) , aplique el código de entrada digital que debe producir potencial cero de salida y debe ajustar el potenciómetro del desplazamiento para salida cero . Para las configuraciones bipolar (COB , CTC), aplique el código de entrada digital que debe producir la maxima salida negativa. Ejemplo: Si el rango de escala completa se conecta para 20V, el máximo voltaje de salida negativo es -lOV. Vea tabla II para los códigos correspondientes.

AJUSTE DE GANANCIA.

Para las configuraciones unipolar o bipolares, aplique la entrada digital que debe dar maxima salida positiva. Ajuste el potenciómetro de Ganancia para esta salida positiva de escala completa. Vea tabla II para voltajes y corrientes de escala completa positivos .

DIGITAL INPUT	ANALOG OUTPUT			
	VOLTAGE(1)		CURRENT	
MSB ↓ LSB ↓	0 to +10V	±10V	0 to −2mA	±1mA
000000000000	+9.9976V	+9.9951V	−1.9995mA	−0.9995mA
011111111111	+5.0000V	0.0000V	−1.0000mA	0.0000mA
100000000000	+4.9976V	−0.0049V	−0.9995mA	+0.0005mA
111111111111	0.0000V	−10.0000V	0.0000mA	+1.000mA
One LSB	2.44mV	4.88mV	0.488µA	0.488µA

NOTE: (1) To obtain values for other binary ranges:
0 to +5V range divide 0 to +10V range values by 2.
±5V range: divide ±10V range values by 2.
±2.5V range: divide ±10V range values by 4.

TABLE II Digital Input/Analog Output

MODELOS DE SALIDA DE VOLTAJE

Pueden conectarse las resistencias de escalamiento intemas en las conexiones de rango de salida que proporcionaron los DAC80 para producir rangos de voltaje de salida bipolar de ±10V, ±5V, o ±2.5V; o rangos de voltaje de salida unipolar de 0 a +5V o 0 a +10V. Vea Figura 6.

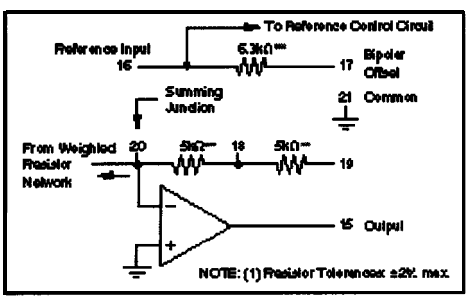


FIGURE 6. Output Amplifier Voltage Range Scaling Circuit.

Se minimizan ganancia y tendencia del desplazamiento debido al rastreo térmico de las resistencias de escalamiento con otros componentes interiores del dispositivo. Se muestran conexiones para varios rangos de voltaje de salida en la tabla Ill. El tiempo establecido para un cambio del rango de escala máximo se especifica como 4ms para el rango de 20V y 3ms para el rango de 10V .

Output Range	Digital Input Codes	Connect Pin 15 to	Connect Pin 17 to	Connect Pin 19 to	Connect Pin 16 to
±10	COB or CTC	19	20	15	24
±5	COB or CTC	18	20	NC	24
±2.5V	COB or CTC	18	20	20	24
0 to +10V	CSB	18	21	NC	24
0 to +5V	CSB	18	21	20	24

TABLE III. Output Voltage Range Connections for Voltage Models.

APENDICE C

```
// NOMBRE:                PRINCIPA.CPP */
/* DESCRIPCION:      Este programas contrala los movimientos de motor  */
#include<math.h>
#include<stdio.h>
#include<conio.h>
#include<string.h>
#include<dos.h>
#include<graphics.h>
#include<ctype.h>
#include<bios.h>
#include<stdlib.h>
#include<alloc.h>
#include "variable.cpp"
#include "objetos.cpp"
#include "velocida.cpp"
#include "carga.cpp"
#include "accion.cpp"
#include "grafica.cpp"
int choicesub(int);
void frame2(int);
void frame3(int);
int video_disponible();
//*************************************************************************
int video_disponible()
{
  int gdriver = DETECT, gmode, errorcode;
  initgraph(&gdriver, &gmode, "");
  errorcode = graphresult();
  if (errorcode != grOk) /* an error occurred */
  {
    printf("Graphics error: %s\n", grapherrormsg(errorcode));
    printf("Press any key to halt:");
    getch();
    exit(l); /* terminate with an error code */
  }
  return 0;
}
//*************************************************************************
// Funcion principal MAIN que llama a todas las subfunciones
//*************************************************************************
                          //Funcion principal del programa void main(void)
{
    int teclap,opcion0=1,i;
    farranque[0]=1;farranque[1]=0;farranque[2]=0;farranque[3]=0;farranque[4]=0;
    intervalo=0;
    ncambios=0;
    for(i=0;i<=5;i++){ secuencia[i][0]=0;secuencia[i][1]=0;secuencia[i][2]=0;}
    clrscr();                          //limpia pantalla
    video_disponible();                  //activa el modo grafico
```

```
setbkcolor(NEGRO);                              //se configura el color de fondo

inicia_puerto();                                //inicializa el puerto paralelo
      velocidad(CERO);      //entrar en 0
      carga(CERO);          //entrada en 0
      _INVCARGA=CERO;
      _INVVELOCIDAD=CERO;
do
{
      teclap=TAB;
      do                    //menu principal
      {
         titulo();
         estado(arranque[t_objf2-1],0.0,puertos[t_objf1-1],0);
          motor();
          choicemenu(opcion0);          //redibujo la pantalla con el frame escogido
          do
          {
                mover();   //Espere por que alguna tecla sea presionada
          }
          while(bioskey(1)==0);
                teclap=getch();
           switch(teclap)
           {
           case 0:
                    teclap=getch();
                     switch (teclap)
                      {
                      case LEFT:            //Moviendo hacia la izquierda
                                  if(opcion0==1){opcion0=4;} else {opcion0=opcion0-1;}
                                  if (versub==l) {cuadro(opcionO);}
                                  break;
                      case RIGHT:           //Moviendo hacia la derecha
                                  if(opcion0==4){opcionO=l;} else {opcion0=opcion0+1;}
                                  if(versub==1){cuadro(opcionO);}
                                  break;
                      case DOWN:
                                  versub=1;
                                  cuadro(opcion0);
                                  teclap=choicesub(opcion0);
                                        //Moviendo hacia abajo
                           }
                           break;
                           case ENTER:
                              if (versub==0) versub=1; else versub=0;
                              if (versub==l) cuadro(opcion0);else cuadro(0);
                              break;
                }
      }
      while(teclap!=ESC);
      setlinestyle(0,1,1);
      setviewport(0,0,getmaxx(),getmaxy(),1);
      clearviewport();
      cmdbottom("<ENTER> SALIR <ESC> CANCELAR",getmaxx()/2-200,getmaxy()/2-20,400,20,3);
```

```
        }
        while(27==getch());
        closegraph();
        velocidad(255);
        carga(255);

        }

//*****************************          0
int choicesub(int opmenu)
{
        int opm1=1,teclam1=0;
                do              //menu principal
                {
                        switch (opmenu)
                        {
                                case 1 :// return into-ml (20,20); break;
                                        choice-mO(opm1,posi[opmenu-1],20);        //redibujo la
                                                                        pantalla con el frame escogido
                                        break;
                                case 2: choice-ml(opm1 ,posi[opmenu-1],20);
                                        break;
                                case 3: choice_m2(opm1 ,posi[opmenu-1],20);
                                        break;
                            //*  case 4: choice_m3(opm1,posi[opmenu-1],20);
                            //          break;
                                case 4: choice_m4(opm1,posi[opmenu-1],20);
                                        break;
                                }

                                do{ mover(); /*espera por una tecla*/}while(bioskey(1)==0);
                                        teclaml =getch();
                                        switch (teclaml)
                                        {
                                                case 0:         //si es una tecla funcional
                                                        teclaml =getch();
                                                        switch (teclaml)
                                                        {
                                                                case UP:                //si es
                                                                                flecha abajo
                                                                        if(opm1==1)
{opm1=nsubm[opmenu-1];} else {opm1=opm1-1;}
                                                                        break;
                                                                case DOWN: //si es flecha aniba
if(opm1==nsubm[opmenu-1]) {opml =1;} else {opml =opml+l ;}
                                                                        break;
                                                        }
                                                        break;
                                                case ENTER:
                                                        switch(opmenu)
                                                        {
                                                                case 2:
                                                                        if(opm1==1){cuadro(0);
inframe1();frameno(POSFX1,POSFY1,POSFX2,POSFY2);cuadro(opmenu);
```

```
estado(arranque[t_objf2-1],0.0,puertos[t_objf11-1],1);}
                                                                    if(opm1==2)
                                                                        {
                                                                        cuadro(0);
                                                                        inarranque();
                                                    frameno(POSFX1-20,POSFY1,POSFX2,POSFY2);
                                                                        cuadro(opmenu);
                                                estado(arranque[t_objf21-1],0.0,puertos[t_objf11-1],1);
                                                                        inicia_puerto(); }
                                                                    break;

                                                            case 3:
                                                                    if(opm1==1){cuadro(0);
                                                                                inframe3();
                                                frameno(POSFX1,POSFY1,POSFX2,POSFY2);cuadro(opmenu);

                                                        estado(arranque[t_objf21],0.0,puertos[t_objf1-1],1);
                    _VELOCIDADOUT=arranque1[t_objf2-1];_INVVELOCIDAD=arranque2[t_objf2-1];  }
 break;

//                                                          case 4:
//
        setviewport(0,20,getmaxx(),getmaxy()-20,0);
//                                                                  if(opm1==1)
//                                                                  {
//                                                                      estado2();
//
        VOLT_VS_CORR();     getch();}
//                                                                  if(opm1==2)
//                                                                  {
//                                                                      estado2();
//
        TORQ_VS_CORR();     getch();}
//
//
setviewport(0,0,getmaxx(),getmaxy(),0);
//                                                                      titulo();
//                                                                      cuadro(opmenu);
//
        estado(arranque[t_objf2-1],0.0,puertos[t_objf1-1],1);
//
//                                                                  break;
                                                            case 4:
if(opm1==1){strcpy(control,"A");cuadro(0); inauto();cuadro(opmenu);}

        if(opm1==2){strcpy(control,"M");}

        if(opm1==3){estadoy=40;cuadro(0);inon();

                                                                    velocidad(CERO);

llentrar en 0
                                                                    carga(CERO);

llentrada en 0
                                                                    _INVCARGA=CERO;
```

```
_INVVELOCIDAD=CERO;
                                                                    estadoy=20;
        setviewport(0,0,getmaxx(),getmaxy(),0); clearviewport();
                                                                        titulo();
                                                                    cuadro(opmenu);}
                                        estado(arranque[t_objf2-1],0.0,puertos[t_objf1-1],1);
                                                                        teclam1=ESC;
                                                                        getch();
                                                                    break;
                                                                }
                                                                    break;

                                            }
                                }
                                while(teclam1!=ESC && teclam1!=LEFT && teclam1!=RIGHT );
        return 1;
}
//*************************
void choicefr(int)
{
}
int     inicia_puerto();
void    margen();
int     lee_dato();
void    grafico(int);
int     grafico4();
void    frametraza(int);
int     dac(int,int);
void    cuadros();
int     VOLT_VS_CORR();
void    choicefr(int);
void traza()
{
          lee_dato();
          grafico(NEGRO);
        clearviewport();
}
//*********************************

int VOLT_VS_CORR(void)
{
  int x,b=80;
  char xx[5];
  clearviewport();
  setcolor(BLANCO);
  if(cvi==0) return 27;
          line(20,getmaxy()-150,getmaxx()-20,getmaxy()-150);      //Eje x
          moveto(getmaxx()-20,getmaxy()-150);
          linerel(0,-5);linerel(5,5);linerel(-5,5);linerel(0,-5);
          outtextxy(getmaxx()-100,getmaxy()-120,"Corriente");

          line(20,getmaxy()-150,20,20);                    //Eje y
          moveto(20,20);
```

```
            linerel(-5,0);linerel(5,-5);linerel(5,5);linerel(-5,0);
            outtextxy(30,20,"Voltaje");

            settextstyle(2, 1, 1);
  for(x=1;x<=cvi-1;x++)
  {
    sprintf(xx,"%d",PVOLT_CORRI[x][1]*factor);
    outtextxy(PVOLT_CORRI[x][1]*factor+20,getmaxy()-150+10,xx);
    line(20+PVOLT_CORRI[x][1]*factor,getmaxy()-150-10,20+PVOLT_CORRI[x][1]*factor,getmaxy()-150);
    settextstyle(2, 0,1);

    sprintf(xx,"%d",PVOLT_CORRI[x][2]*factor);
    outtextxy(0,getmaxy()-150-PVOLT_CORRI[x][2]*factor,xx);
    line(20,getmaxy()-150-PVOLT_CORRI[x][2]*factor,30,getmaxy()-150-PVOLT_CORRI[x][2]*factor);
    settextstyle(2, 1, 1);
  }

   settextstyle(2, 0,1);
   moveto(20+PVOLT_CORRI[1][1]*factor,getmaxy()-150-PVOLT_CORRI[1][2]*factor); //punto inicial

   for(x=2;x<=cvi-1;x++)
   {
         if(x==1)
          linerel(PVOLT_CORRI[x][1]*factor,-PVOLT_CORRI[x][2]*factor);
         else
          linerel(PVOLT_CORRI[x][1]*factor-PVOLT_CORRI[x-1][1]*factor,-
PVOLT_CORRI[x][2]*factor+PVOLT_CORRI[x-1][2]*factor); //curva
        }
        frame("DATOS",getmaxx()-100,50,getmaxx(),getmaxy()-160); //cuadro de datos
        texto("Corr. Volt.",getmaxx()-95,50,NEGRO); //titulo de datos

         if( cvi>15){cvi=15;}


         for(x=1;x<=cvi-1;x++)      //despliegue de datos
         {
          numero(PVOLT_CORRI[x][1]*factor,getmaxx()-90,b,NEGRO);
          numero(PVOLT_CORRI[x][2]*factor,getmaxx()-40,b,NEGRO);
          line(getmaxx()-100,b+10,getmaxx(),b+10);
          b=b+20;

         }
 settextstyle(2, 0,1);
return 9;
}

//*******************
int TORQ_VS_CORR(void)
{
  int x,b=80;
  char xx[5];
  clearviewport();
  setcolor(BLANCO);
  // if(_NVI==0) return 27;
  if(cvi==O) return 27;
```

```
            line(20,getmaxy()-150,getmaxx()-20,getmaxy()-150);        //Eje x
            moveto(getmaxx()-20,getmaxy()-150);
            linerel(0,-5);linerel(5,5);linerel(-5,5);linerel(0,-5);
            outtextxy(getmaxx()-100,getmaxy()-120,"Corriente");

            line(20,getmaxy()-150,20,20);                              //Eje y
            moveto(20,20);
            linerel(-5,0);linerel(5,-5);linerel(5,5);linerel(-5,0);
            outtextxy(30,20,"Torque");

            settextstyle(2, 1, 1);
  for(x=1;x*=cvi-1;x++)
  {
    sprintf(xx,"%d",PVOLT_CORRI[x][1]*factor);
    outtextxy(PVOLT_CORRI[x][1]*factor+20,getmaxy()-150+10,xx);
    line(20+PVOLT_CORRI[x][1]*factor,getmaxy()-150-10,20+PVOLT_CORRI[x][1]*factor,getmaxy()-150);
    settextstyle(2, 0, 1);

    sprintf(xx,"%d",PVOLT_CORRI[x][2]*factor);
    outtextxy(0,getmaxy()-150-PVOLT_CORRI[x][2]*factor,xx);
    line(20,getmaxy()-150-PVOLT_CORRI[x][2]*factor,30,getmaxy()-150-PVOLT_CORRI[x][2]*factor);
    settextstyle(2, 1, 1);
  }

   settextstyle(2, 0, 1);
 //  moveto(20,getmaxy()-150);
   moveto(20+PVOLT_CORRI[1][1]*factor,getmaxy()-150-PVOLT_CORRI[1][2]*factor); //punto inicial

   for(x=2;x<=cvi-1;x++)
   {
         if(x==1)
          linerel(PVOLT_CORRI[x][1]*factor,-PVOLT_CORRI[x][2]*factor);
         else
          linerel(PVOLT_CORRI[x][1]*factor-PVOLT_CORRI[x-1][1]*factor,-
PVOLT_CORRI[x][2]*factor+PVOLT_CORRI[x-1][2]*factor); //curva
        }
        frame("DATOS",getmaxx()-100,50,getmaxx(),getmaxy()-160); //cuadro de datos
        texto("Torq. Volt.",getmaxx()-95,50,NEGRO); //titulo de datos

        if( cvi>15){cvi=15;}


        for(x=1;x<=cvi-1;x++)      //despliegue de datos
        {
          numero(PVOLT_CORRI[x][1]*factor,getmaxx()-90,b,NEGRO);
          numero(PVOLT_CORRI[x][2]*factor,getmaxx()-40,b,NEGRO);
          line(getmaxx()-100,b+10,getmaxx(),b+10);
          b=b+20;

        }
 settextstyle(2, 0, 1);
return 9;
}
```

```
//************************************************
//**********

int dac (int ff,int valor)       // ff puede tomar el valor 16 o 32 de fx fy respect.
{
        int w1,w2,clk;
        if (ff==16) clk=8;
        if (ff==32) clk=11;

        wl= valor; //atoi(argc[2])
        w2=w1^0xb;
        outportb(pto,ff+f1+f2);              // habilta Fx y F1y/o F2 si fuera el caso
        outportb(pto+2,w2);                  // lee parte baja del DAC
        outportb(pto,clk+1+ff+fl+f2);        // clk para FBx
        outportb(pto,ff+f1+f2);              // baja el clk
        w1>>=4;                      // desplazamiento
        w2=w1^0xb;
        outportb(pto+2,w2);                  // lee parte alta del DAC
        outportb(pto,clk+2+ff+f1+f2);        // clk para FAx
        outportb(pto,ff+f1+f2);              // baja el clk
        outportb(pto,clk+3+ff+f1+f2);        // clk para F1
        outportb(pto,f1+f2);         // enclavamiento de F1 ylo F2
        return 0:


//*****************************************************************//

void grafico( int color)
{
  int x,antes,centro;
  centro=getmaxy()/2+100;
          antes=getcolor();
           setcolor(color);

 for (x=1;x<=700;x++)
            {
                setcolor(AZUL); //torque
              if(_CANAL4==1)line(x,centro-t[x][3],x+1,centro-t[x+1][3]);
                setcolor(ROJO);  //velocidad
              if(_CANAL3==1)line(x,centro-t[x][2],x+1,centro-t[x+1][2]);
                setcolor(VERDE); /// corriente
              if(_CANAL2==1)line(x,centro-100-t[x][1],x+1,centro-100-t[x+1][1]);
                setcolor(BLANCO);   //voltaje
              if(_CANAL1==1)line(x,centro-t[x][0],x+1,centro-t[x+1][0]);


            }

    //      }
           setcolor(antes);
}
//*************
```

```
int grafico3()
{
int x,i;
_TG=1;
            for (x=1401;x<=datos;x++)
            {
                        line(x,getmaxy()/2-t[x][_TG]/2,x+1,getmaxy()/2-t[x+1][_TG]/2);
             }

return 0;
}

//*****************************************************************//
int inicia_puerto()
{
            // pto=_PUERTO;
            pto=apuertos[t_objf1-1];
            //pto=peek(0x40,0x8);
            outportb(pto+2,0x0);          // blamquea el puerto de control
            return 0;
}

//*****************************************************************//

int lee_dato()
{
            int i,j,k,str,dat;
            double fvel;

            _SVOLTAJE=0;
            _SCORRIENTE=0;
            str=f1+f2+15;
            numero(_INVVELIN,150,getmaxy()-80,BLANCO);
            numero(_VELOCIDADIN,200,getmaxy()-80,BLANCO);

            numero(_CARGAIN,440,getmaxy()-80,BLANCO);
            dobles(_INVCARIN,390,getmaxy()-80,BLANCO);

for (i=1;i<=datos;i++)
 {
  k=1+f1+f2;
  outportb(pto,str);            // senal de STAR/CLEAR
   j=0;
   do{
               outportb(pto,k);                        // selecciona parte alta de un ADC
               dh=inportb(pto+l);          II lee senal "ALTA" de un ADC
               dh&=0xF0;                  // convierte a variable de 8 bits
               k++;
               outportb(pto,k);             // selecciona parte “baja” de un ADC
               dl=inportb(pto+1);           // lee senal “BAJA” de un ADC
               dl>>=4;                             // desplazamiento a la derecha
               dl&=0x0F;                           // convierte a variable de 8 bits
               k++;
```

```
            switch(j)
            {
             case 0:
                    _SVOLTAJE=_SVOLTAJE+dh|dl;
                    break;
             case 1:
                    _SCORRIENTE=_SCORRIENTE+dh|dl;
                    break;
            case 2:
                    _VELOCIDADIN=dh|dl;
                    break;
            case 3:
                    _CARGAIN=dh|dl;
                    break;
            }
            t[i][j]=dh|dl;

            j=j+1;
  }
  while (j<=3);
 }
                                        fvel=_VELOCIDADIN;
                                        fvel=((255.0-fvel)*1000.0/250.0); //a la salida
                                             fvel=floor(fvel);
                                             _INVVELIN=fvel;

                                        fvel=_CARGAIN;
                                        fvel=((255.0-fvel)*10.0/250.0); //a la salida
//                              fvel=floor(fvel);
                                             _INVCARIN=fvel;



            numero(_INVVELIN,150,getmaxy()-80,NEGRO);
            numero(_VELOCIDADIN,200,getmaxy()-80,NEGRO);
            numero(_CARGAIN,440,getmaxy()-80,NEGRO);
            dobles(_INVCARIN,390,getmaxy()-80,NEGRO);

            if(PVOLT_CORRI[cvi][2]<=0){PVOLT_CORRI[cvi][2]=PVOLT_CORRI[cvi-1][2];}
            if(PVOLT_CORRI[cvi][1]<=0){PVOLT_CORRI[cvi][1]=PVOLT_CORRI[cvi-1][1];}
            if(_SVOLTAJE/700>=PVOLT_CORRI[cvi-1][2]){PVOLT_CORRI[cvi][2]=_SVOLTAJE/700;}
            if(_SCORRIENTE/700>=PVOLT_CORRI[cvi-1][1]) {
PVOLT_CORRI[cvi][1]=_SCORRIENTE/700;}
return 0;
}

void margen()
{
 int y,i=1,x=1,v,dec,sign;
 char c[2];
 setlinestyle(0,2,1);
 line(0,0,getmaxx(),0) ;  //linea superior del marco
 line(0,0,0,getmaxy());   // linea izquierda del marco
 line(getmaxx(),0,getmaxx(),getmaxy());
 line(O,getmaxy(),getmaxx(),getmaxy());
```

```
v=50;
for(y=1;y<getmaxy();y++)
{
 i=i+1 ;
 if( i==20)
 { v=v-1 ;
   line(0,y,getmaxx(),y);
   i=1;
 }
}
          for (x=1;x<=700;x++)
             {
                line(x,getmaxy()/2-t[x][_TG]/2,x+1,getmaxy()/2-t[x+1][_TG]/2);
                i=i+1,
                if((i==30)||(i==O))
                {
                     line(x,getmaxy(),x,getmaxy()-15);
                     outtextxy(x,getmaxy()-20,fcvt(x,0,&dec,&sign));
                    i=1;
                }
             }

 line(getmaxx()/2,0,getmaxx()/2,getmaxy());

 settextjustify(50,0);
 outtextxy(10,2," Grafica de velocidad del Motor" );
 outtextxy(getmaxx()-300,2," Presiones <Enter> Para salir");

}

void cuadros()
{

int y,i=1;
 setlinestyle(0,2,1);
i=10   ;
for(y=1;y<getmaxy();y++)
 {
 i=i+1 ;
 if( i==20)
 {
 line(0,y,getmaxx(),y);

 i=1;
 }
}
i=8 ;
for(y=1;y<getmaxx();y++)
 {
          i=i+1 ,
          if( i==30)
          {
          line(y,0,y,getmaxx());
          outtextxy(y+3,getmaxx()-20,"25");
          i=1;
```

BIBLIOGRAFIA

1. TERCO, Thyristor Unit 4200, Theory and Laboratory Exercises, Manual del Laboratorio ESPOL, Guayaquil 1990, pp. 7

2. TERCO, Thyristor Unit 4200, Theory and Laboratory Exercises, Manual del Laboratorio ESPOL, Guayaquil 1990, pp. 4

3. Javier Cevallos Sierra, Enciclopedia del Lenguaje C++, Ra-ma

4. Harry Fox, Master OP-AMP Applications Handbook, Folleto ESPOL, Guayaquil, 1988. pp102

5. D. Stout-M. Kaufman,Handbook of operational amplifier circuit design, Folleto ESPOL, Guayaquil 1989, pp. 180-181.

6. Harry Fox, Master OP-AMP Applications Handbook, Folleto ESPOL, Guayaquil, 1988. pp106-209

7. Ronald J. Tocci, Sistemas Digitales, Prentice Hall Hispanoamericana S.A, Mexico, 1987, pp. 452-463