UNIVERSIDAD NACIONAL AUTONOMA
DE MEXICO

FACULTAD DE INGENIERIA

ANALISIS ESPECTRAL POR COMPUTADORA

T E S I S

QUE PARA OBTENER EL TITULO DE

INGENIERO MECANICO ELECTRICISTA

P R E S E N T A

HECTOR GALVAN OSEGUERA

DIRECTOR DE TESIS: ING. MARIO IBARRA PEREYRA

MEXICO, D. F. 1990

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

A MIS PADRES

Por su apoyo y dedicación demostrado en todo momento

A MIS HERMANOS

Les dedico este trabajo como una muestra de cariño

A MIS MAESTROS

Este trabajo lo dedico a todos mis maestros y profesores de la Facultad de Ingeniería. Especialmente a mi maestro, el Ing. Mario Ibarra Pereyra por su paciencia y empeño durante el desarrollo de este trabajo.

A Bertha

Lo dedico este trabajo muy especialmente a Bertha por su ayuda y comprensión.

ANALISIS ESPECTRAL POR COMPUTADORA

ANALISIS ESPECTRAL POR COMPUTADORA

I- INTRODUCCION

I-1 DESCRIPCION Y PRINCIPIOS BASICOS

El objetivo principal de este trabajo es el diseño de una tarjeta para microcomputadora que pueda utilizarse como sistema de adquisición de datos de propósito general y desarrollar el complemento necesario para el análisis espectral.

El diseño de la tarjeta y su uso estarán apoyados por varios capítulos donde se fundamentan los conceptos y herramientas necesarias para el desarrollo del sistema.

Se aborda el diseño por medio de convertidores analógicos-digitales y digitales-analógicos para disponer de elementos y comprender la operación y limitaciones del sistema .

El análisis de Fourier es tratado con cierto detalle exponiendo la teoría básica de la transformada de Fourier así como los algoritmos principales de transformada rápida de Fourier.

El diseño de la tarjeta se fundamenta con una exposición en detalle sobre las formas de manejo con microprocesadores y sus enlaces básicos de entrada salida.

I-2 APLICACIONES PRINCIPALES

Los principales usos de este sistema comprenden las diferentes modalidades como instrumento de medición espectral, así como el despliegue de señales en el dominio del tiempo.

La captación de datos en el laboratorio o en otro medio diferente (zona agrícola, pesquera, cultivos marinos, etc) se efectua como un conjunto de valores de algunas variables (ventas, voltaje,corriente, Ph, etc.) contra alguna variable independiente,normalmente el tiempo. Para descubrir la presencia de tendencias significativas.

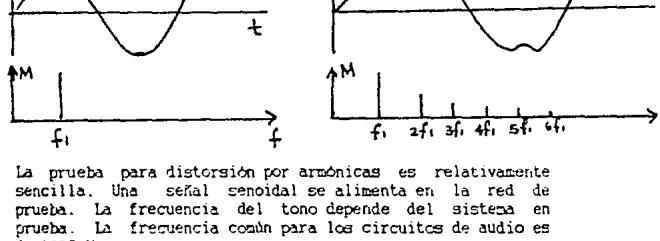
Algunas veces estas tendencias son difíciles de descubrir a causa de la presencia de ruido o alguna otra forma de perturbación de corta duración en los datos.
Este sistema tiene una aplicación directa en el análisis espectral con capacidad de expansión para permitir simular sistemas lineales con respecto a la respuesta en frecuencia y análisis de estabilidad.

La astronomía, la física, la química, la estadística, la biomédicina, la electrónica, la mecánica y otros campos tienen areas de estudio que pueden obtener beneficio de la

teoria de Fourier, especificamente de la transformada rapida de Fourier.

Análisis de Distorsión

La distorsión que se puede analizar espectralmente es la distorsión armónica consistente en que a un circuito se le alimenta una senoide pura y a la salida se obtienen sus armónicas como se ve en la figura :



La prueba para distorsión por armónicas es relativamente sencilla. Una señal senoidal se alimenta en la red de prueba. La frecuencia del tono depende del sistema en prueba. La frecuencia común para los circuitos de audio es de 1000 Hz.
Generalmente, el nivel de este tono se establece para producir la salida máxima permitida por el sistema.
La salida causada por el tono de prueba contiene las armónicas de distorsión causadas por la red de prueba.

I-3 LIMITACIONES EN EL USO

Es claro que todo instrumento de medición tiene limitaciones en cierto intervalo de operación.
De tal manera que sólo es capaz de mostrar o procesar datos con ciertas caracteristicas tales como velocidad, fuerza, voltaje, corriente, frecuencia, etc. Todas estas variables deben estar dentro de cierto intervalo para que las mediciones sean confiables.

Las limitaciones dependen principalmente la velocidad de proceso en el computador.
Ampliar el intervalo de operación implicaría una serie de cambios en el diseño del sistema.
El primer cambio sería el uso de un procesador interno y/o externo con alta velocidad de operación, perteneciente a cierto tipo de arquitectura y tecnología que lo clasificaría como no-convencional. Además de tener otras características que situan el diseño en una relación costo-beneficio muy atractiva y difícil de evaluar.

La justificación para el uso de un sistema con características sobresalientes implica un diseño más

rebuscado en cuanto a estructura y tecnología.
Para reducir las limitantes del sistema podemos involucrar dispositivos con características estructurales y tecnológicas diferentes.
La tecnología NMOS, CMOS y otras afines son las más usadas por los diseñadores y fabricantes de microcircuitos, en este caso los microprocesadores y circuitos lógicos adicionales comparten las mismas cualidades.

También podríamos considerar una restricción con respecto a la conversión analógica-digital, en el caso de adquisición de datos para procesamiento en tiempo real.
Los convertidores analógicos-digitales se dividen en varias categorías, según su arquitectura, que poseen diferentes características de funcionamiento tales como velocidad de conversión, frecuencia de operación, etc.

De tal manera que no podemos tomar datos de algún proceso físico, sin considerar la naturaleza del fenómeno que se inspecciona. Debemos tener en cuenta la naturaleza del fenómeno que se inspecciona para poder decidir sobre las características del convertidor analógico-digital.

I-4 CARACTERISTICAS PRINCIPALES

Para poder resolver problemas de telecomunicaciones o sistemas de control, es esencial conocer sus características.
Estas características determinan la llamada distorsión lineal de un sistema de comunicación, en otras palabras, las distorsiones de una señal de comunicaciones que se presenta debido a una función de transferencia dependiente de la frecuencia. En los sistemas de control, el comportamiento transitorio y la estabilidad son muy importantes. Las características de un sistema están determinadas por las funciones de transferencia del sistema de control.

Las señales de interferencia también afectan el comportamiento de los sistemas de comunicación y control. Pueden presentarse dentro del sistema, por ejemplo, ruido térmico o distorsiones causadas por no-linearidades o pueden venir del exterior tales como otros canales de comunicación o líneas de potencia.
Una señal sufre diversos cambios cuando pasa a través de un sistema. En general, es atenuada, distorsionada o intervienen perturbaciones en ella. La señal resultante a la salida del sistema no es exactamente la misma que la señal de entrada. Para asegurar que una señal no sea afectada por distorsiones o señales de interferencia durante la transmisión, dicho sistema y sus componentes deben mantener una cierta calidad.
En otras aplicaciones, sin embargo, se desea modificar señales en forma particular, por ejemplo, convertidores de

frecuencia, moduladores, filtros selectivos, cambiadores de fase, redes de retraso, amplificadores, atenuadores. Los siguientes parámetros determinan las características del sistema :

a) Distorsiones lineales
b) Distorsiones *no*-lineales
c) Señales de interferencia

Las distorsiones *no*-lineales y señales de ruido se pueden medir con analizadores de espectros, la distorsión lineal con un analizador de redes. Ambos análisis son necesarios si se desea entender el comportamiento real del sistema.

Los parámetros más importantes del equipo de medición para el análisis de los sistemas son :

1-Rango de frecuencia
2-Resolución mínima en ancho de banda
3-Rango en nivel de medición
4-Rango de despliegue
5-Rango en medición de fase
6-Rango en medición de retardo de grupo
7-Resolución
8-Impedancia de entrada

El circuito que se va a diseñar tendrá que especificarse adecuadamente en estos parámetros.

II- CONVERTIDORES D/A Y A/D

Cuando se desea guardar en una memoria los datos de salida de algun sensor tales como temperatura o presion, usualmente los datos deben ser convertidos de una forma analógica a una forma digital para que sean aceptados por la memoria. Un convertidor tipo analógico/digital (A/D) es necesario para hacer la conversión. Una cantidad analógica puede tener cualquier valor dentro de un rango. Una memoria digital de una cantidad tiene solamente un número fijo de valores posibles.

El inverso de un convertidor analógico/digital es el convertidor digital/analógico. El convertidor D/A produce una salida de corriente o voltaje proporcional al la magnitud de una palabra binaria aplicada a su entrada.

Los Convertidores digitales tanto el analógico/digital como el digital/analógico tienen un impacto definitivo en el advenimiento de la nuevas tendencias tecnológicas en diversos campos de la ingeniería con una repercusión intensa en los aspectos sociales y económicos: reflejandose en el campo de la televisión comercial, en los equipos de audio que utilizan técnicas de modulación por código de pulsos, en equipos biomédicos totalmente digitalizados, sistemas de medición en diferentes áreas, y en muchos otras aplicaciones.

Discutiremos convertidores D/A porque varios tipos de convertidores A/D usan convertidores D/A como parte su circuito.

II-1 CONVERTIDORES DIGITALES/ANALOGICOS

Convertidores D/A de resistencia por peso binario

El tipo más simple de convertidor usa resistencia por peso binario y un amplificador operacional, para convertir el valor binario lógico a un voltaje proporcional.

La figura 2-1 muestra el circuito para un convertidor de 4 bits. Un amplificador operacional tiene muy alta ganancia (usualmente mayor de 100.000), baja impedancia de salida y alta impedancia de entrada.
Tiene dos entradas : una inversora, marcada con el signo (-) en el simbolo esquematico triangular, y una entrada no-inversora, marcada con el signo (+). Probablemente lo más importante que recordar con respecto a los amplificadores operacionales es que cuando se realimente una parte de la salida a la entrada inversora (realimentación negativa) medirá el mismo voltaje en las entradas inversora y no-inversora.
La salida del amplificador operacional será una fuente o sumidero de la corriente necesaria para mantener estos voltajes al mismo nivel.
En el circuito de la figura 2-1 la entrada no-inversora está a tierra, por lo tanto, la entrada inversora será sostenida a 0 voltios. Ya que la entrada no-inversora presenta 0 voltios pero no está verdaderamente atada a tierra, algunas veces se le conoce como tierra virtual.

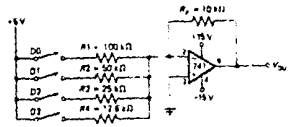


Figura 2-1. Convertidor D/A de resistencias por peso binario.

Convertidor R/2R

Para un convertidor D/A de más de 4 bits, un circuito integrado como la figura 2-1 tiene el problema que se requiere un rango demasiado grande de valores. Otro circuito para producir corrientes por peso binario con solo dos valores de resistencias se muestra en la figura 2-2. Como antes, las corrientes por peso binario se convierten a un voltaje proporcional por el operacional y R .
F
A primera vista, la configuración R/2R puede parecer complicada, pero por ser una proporción de resistencias es bastante fácil.
Primero suponemos que el interruptor S3, el bit más

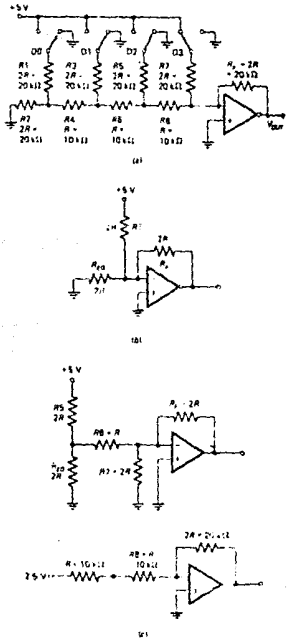


Figura 2-2. Convertidor D/A R/2R. a) Esquematico completo b) Circuito equivalente para MSB c) Circuito equivalente para MSB siguiente.

significativo esta conectado a una referencia de voltaje precisa de + 5V y los otros interruptores están cerrados a tierra Entonces R1 y R2 están en paralelo a tierra. Una resistencia de 2R en paralelo con otra 2R es equivalente a un solo R a tierra de su union.
Esta R equivalente se añade a R4 para formar otra 2R en paralelo con R3 a tierra. La combinación de R3 y los resistores previos se reducen a R en serie con R6. Usando la misma técnica a través del resto dá el circuito simplificado de la figura 2-2b. Como la tierra virtual del operacional está a 0 voltios, no fluye corriente a través de la resistencia equivalente. Por lo tanto, esta resistencia puede ser ignorada. Los 5V en R7 de 20K producen .25 ma a través del punto de suma y R .
F
El voltaje de salida producido por el siguiente bit más significativo se obtiene de la siguiente manera : mentalmente cerramos el interruptor D2 a + 5V y cambiamos D3 a tierra.

Todas las resistencias a la izquierda de R5 en la figura 2-2a se reducen a una sola resistencia de 2R a tierra. El circuito puede ser simplificado encontrando el equivalente de Thevenin del divisor de voltaje formado por R5 y 2R a tierra. como lo muestra la figura 2-2c. El voltaje de Thevenin es solo el voltaje de la union, o 2.5 voltios.
La resistencia de Thevenin es igual al valor de las dos resistencias en paralelo o R. Todo lo que se encuentre a la izquierda de R8 puede representarse por una sola resistencia de valor R conectada a 2.5 V.
Se puede ignorar R7 porque ambos extremos están a tierra.

La resistencia total entre los dos puntos se suma y el voltaje equivalente de thevenin de 2.5 V. La corriente en el punto de suma en 2.5V/20K ó 0.125 ma. Esta corriente en RF de 20K produce un voltaje de salida de -2.5V para el siguiente bit más significativo.
Con un análisis similar puede determinarse que el siguiente interruptor menos significativo produce una salida de 1.25V y el siguiente interruptor menos significativo una salida de .625 V. El voltaje de escala completa para todos los interruptores encendidos es la suma de los cuatros voltajes, ó 375V

Aunque el convertidor R/2R es más difícil de analizar que el convertidor por peso binario. es más fácil de fabricar con exactitud porque solamente necesita uno o dos valores de resistencias con pelicula de metal de precisión. El número de bits se puede expandir añadiendo secciones con los mismos valores R/2R Se puede conectar un contador binario TTL o CMOS en lugar de los interruptores en la figura 2-2a para dar una respuesta escalonada

Conclusion.- La senal de salida del operacional es el equivalente analógico del número binario alimentado a las resistencias.

Convertidores D/A Monoliticos e hibridos.

Monolitico significa "una sola piedra", y cuando se usa como referencia a circuitos integrados, indica que todos los circuitos están contenidos en la misma pieza de silicio. Un circuito híbrido contiene una o más piezas de silicio y redes de resistencias u otros componentes en un solo paquete integrado.

Un convertidor D/A monolitico es el MC1408L, cuyo diagrama se muestra en la figura 2-3a. El MC1408L tiene una presentación estándar de 16 patillas y requiere un Vcc de +5V y un VEE de cualquier voltaje de un mínimo de -5V a un máximo de -15V. En el convertidor MC1408L existe un arreglo R/2R que divide una corriente de referencia del amplificador de corriente en ocho corrientes por peso binario. Los interruptores de transistores bipolares entonces conmutan las corrientes por peso binario hacia una línea de salida de acuerdo a la palabra binaria en las entradas TTL compatibles A1-A8. Notese que las etiquetas para MSB y LSB están invertidas con respecto a la forma normal de designar los bits en los contadores.

Algunos convertidores D/A tienen estas caracteristicas y otros no, para saber sus caracteristicas es necesario leer sus hojas técnicas. El convertidor 1408L tiene una salida de corriente que puede ser convertida a voltaje con un amplificador operacional y una resistencia como se muestra en la figura 2-3b. Este voltaje puede ser calculado usando las formulas :

$$V_{out} = \frac{V_{ref}}{R14}\left(\frac{A1}{2} + \frac{A2}{4} + \frac{A3}{8} + \frac{A4}{16} + \frac{A5}{32} + \frac{A6}{64} + \frac{A7}{128} + \frac{A8}{256}\right)$$

Para los valores mostrados , el voltaje de salida es :

$$_4V = \frac{5\,k\Omega}{1\,k\Omega}\ \frac{255}{256}\ 9.961\,V$$

Esto es referido a un convertidor de 10V escala completa.

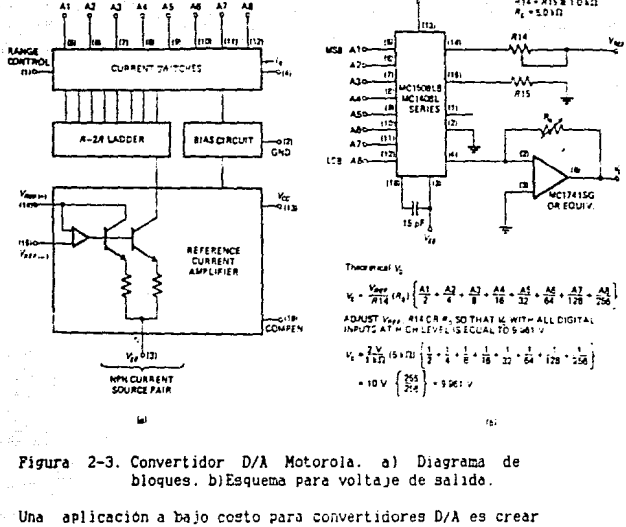
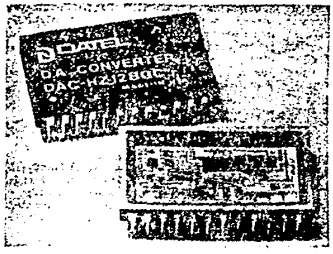


Figura 2-3. Convertidor D/A Motorola. a) Diagrama de bloques. b)Esquema para voltaje de salida.

Una aplicación a bajo costo para convertidores D/A es crear una amplia gama de sonidos y formas de onda. Las salidas de un contador binario de 8 bits se pueden conectar a las entradas del convertidor D/A. Conforme el contador recorre la cuenta binaria, el convertidor produce una forma de onda " diente sierra" hecha de 255 pasos. La frecuencia de salida es igual a la frecuencia de entrada de reloj del contador dividida entre 256. Se pueden crear formas de onda más elaboradas conectando el convertidor D/A a la salida de una memoria de 8 bits RAM o ROM. La memoria está programada con el valor binario deseado. La memoria es leída secuencialmente hacia el convertidor por el contador atado a sus entradas de direcciones. El número de sonidos posibles está limitado solo por la imaginación, la cantidad de memoria, y la paciencia para programar la memoria.

Para aplicaciones que requieran más bits de resolución, existen convertidores híbridos D/A tales como el DATEL DAC-HZ 12 BCG disponible en un integrado de 24 patillas. Este es un convertidor que requiere solamente +15V y -15V en la fuente de alimentación. En el convertidor están incluidos el amplificador operacional y las resistencias, de tal manera que se pueden obtener salidas de voltaje y de corriente. El voltaje de salida a escala completa puede ser cambiado usando diferentes valores en la resistencia de realimentación del operacional. Está incluida una referencia de voltaje de alta precisión para la red de corriente. Para lograr la precisión requerida para el convertidor, las



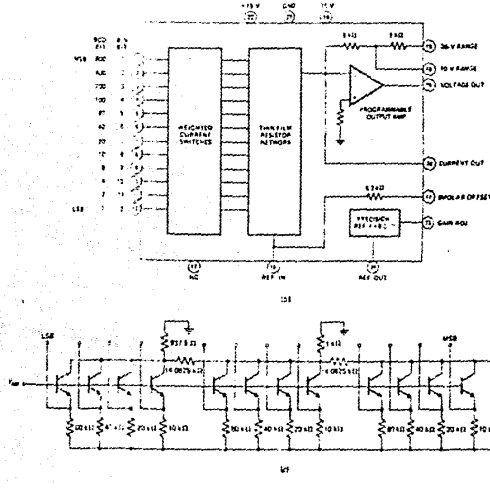


Figura 2-4. Convertidor DATEL hibrido. a) Fotografía. b) Diagrama de bloques. c) Fuentes de corriente por peso binario.

resistencias en la red son sometidas, a un proceso por rayo LASER para obtener la uniformidad en los valores. Ya que se debe ajustar cada resistencia, es necesario reducir el número de resistencias requerida en el convertidor. El arreglo R/2R requiere dos resistencias por bit, y el método de peso binario descrito anteriormente usa solamente una resistencia por bit. El problema con el método de peso binario es el gran rango de valores para un convertidor de más de 4 bits. Este problema se resuelve usando tres secciones de 4 bits tipo peso binario como se muestra en la figura 2-4c. Las resistencias en serie con las secciones menos significativas atenuan las salidas de estas secciones, de tal manera que puedan usarse resistencias de la fuente de corriente del mismo valor en todas las secciones. Como se muestra en la figura 2-4c, las fuentes de corriente de precisión están todas encendidas. Los circuitos de los interruptores no mostrados apagan las fuentes de corriente deseadas poniendo sus emisores en estado alto. Hay varios esquemas de conmutación posibles, siendo imposible tratarlos todos aquí. Lo importante para el diseñador, en general, con respecto a los convertidores D/A son las especificaciones técnicas más que el tipo o modalidad de conmutación que se usa internamente.

Características de convertidores D/A

Fuentes de error y especificaciones

La primera característica de un convertidor D/A a considerar es la resolución. Esto está determinado por el número de bits en la palabra de entrada. Un convertidor de 8 bits, por ejemplo, tiene 2^8 ó 256 niveles de salida; así que su resolución es una parte en 256. Un convertidor de 12 bits tiene una resolución de 1 parte en 2^{12} ó 4096.
La resolución algunas veces se expresa como un porcentaje. Una parte de 4096 es 0.024 por ciento.

Lo siguiente a considerar es la precisión del convertidor D/A. Por precisión se entiende una comparación de la salida real contra la salida esperada.
Si el convertidor tiene una salida de escala completa de 10 V y +/- 0.2 % de precisión, entonces el máximo error para cualquier salida será de 0.002 x 10 V, ó 20 mV. Idealmente, la precisión de un convertidor será, como peor caso, +/- 1/2 de su bit menos significativo. Un convertidor de 10 bits tiene una resolución de 1 parte por 1024, o aproximadamente 0.1 %.

Para ser uniformes, la precisión puede ser +/- 0.05 %. Hay varias formas en que este error puede aparecer. La figura 2-5 puede ilustrar tres de ellas.
El error de linealidad involucra la cantidad por la que la salida verdadera difiere de la salida ideal en forma lineal.

Este error usualmente es causado por errores en los valores de las resistencias de la fuente de corriente. Otro tipo de error conocido como error de ganancia, es frecuentemente causado por errores en la resistencia de realimentación de la parte conversora de corriente a voltaje en el amplificador operacional. El error de compensación significa que la salida no es cero cuando las entradas están a cero volts. El error por compensación añade un valor constante a todos los valores de la salida como se muestra en figura 2-5c. Esto puede ser causado por errores o corrientes de fuga en los interruptores de corriente.

La precisión de un convertidor D/A puede ser menor a su resolución a una parte por 256, ó 0.39 %. La hoja de datos dá una precisión para el MC1408L-8 de +/- 0.19 %, +/- 1/2 del bit menos significativo que coincide con la resolución. Sin embargo, Motorola declara también al convertidor MC1408L-8, que es un convertidor de 8 bits con una precisión de sólo +/- 0.78 %. Esto equivale a la resolución de un convertidor de 8 bits. Es mucho mas barato con menor resolución y se puede usar donde se requiere el rango de 8 bits pero no la precisión.
Otro término usado con los convertidores D/A es la monotonicidad. Un convertidor se dice monotonico si no pierde ningun paso cuando se extiende en todo el rango por un contador.

El tiempo de asentamiento en la salida es otra especificación dada para convertidores. El tiempo de asentamiento es usualmente definido como el tiempo requerido para que la salida del convertidor pueda ajustarse dentro del intervalo +/- 1/2 LSB (bit menos significativo) del valor final después de un cambio en la palabra de entrada. Si el convertidor es operado a una frecuencia muy alta, es posible que no haya tiempo para validar el dato con el valor correcto antes que haga la conmutación al siguiente valor.

II-2 CONVERSION ANALOGICA/DIGITAL

Aquí se expone el problema inverso a los convertidores D/A, es decir convertir una señal o voltaje analógico a una palabra binaria, para poder manipularla por medio de procesamiento digital; de igual manera que en los convertidores D/A, la resolución de un convertidor A/D depende del número de bits.

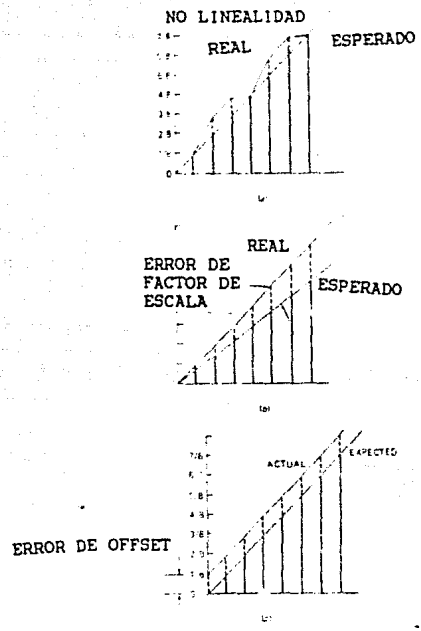
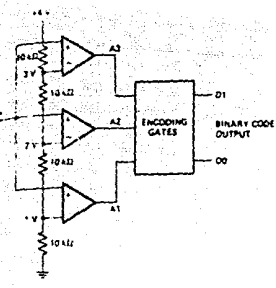


Figura 2-5. Errores de D/A. a) Linealidad b) Escala o ganancia. c) Compensación.

Un convertidor de 8 bits por ejemplo, puede representar un voltaje de entrada con 256 palabras digitales posibles. La resolución se dice ser de 1 a 256. Existen muchos tipos de convertidores.

Convertidores Paralelos, Simultáneos o "Flash"

El convertidor A/D más simple en concepto y el más rápido es el tipo paralelo mostrado en la figura 2-6a. Un divisor de voltaje resistivo define los voltajes de umbral y/o disparo en tres comparadores.



(a)

V_{in} (Volts)	Comparator Outputs			Binary Output	
	A1	A2	A3	D1	D0
0 to 1	0	0	0	0	0
1 to 2	1	0	0	0	1
2 to 3	1	1	0	1	0
3 to 4	1	1	1	1	1

(b)

Figura 2-6. Convertidor A/D por comparador paralelo
a) Esquema b) Salida para valores voltajes de entrada.

La referencia de voltaje en la parte alta del divisor es el voltaje de entrada máximo o de escala completa.
La salida de cada comparador se pone en estado alto si el voltaje en su entrada positiva es mayor que el voltaje de referencia en su entrada negativa. La señal analógica a ser digitalizada se aplica a todos los comparadores en paralelo. El número de comparadores en estado alto indica la amplitud del voltaje de entrada analógico. La figura 2-6b muestra las salidas para el circuito, en la figura 2-6a con varios voltajes de entrada. Si el voltaje de entrada es menor que 1 voltio, ningún comparador se pone en estado alto. Un voltaje entre 1 y 2 voltios se indica si el comparador de umbral más bajo tiene una salida alta. Un voltaje de entrada de 2 a 3 voltios, dá un estado alto a las salidas de los comparadores A1 y A2, y un voltaje arriba de 3 V produce una salida alta en todos los comparadores.
Este sistema es similar a una regla de 10 Cms. con sólo las marcas de los centímetros en ella. El convertidor resolvera una entrada de voltaje para solamente uno de los 4 niveles de entrada Esto es equivalente a 2 bits de resolución.
Se pueden usar mas comparadores para incrementar la

resolucion del convertidor. Son necesarios 7 comparadores para 3 bits de resolución, y 15 comparadores para 4 bits de resolución, o 16 niveles. Un convertidor de N bits requiere 2 comparadores, así que un convertidor de 8 bits requiere 256 comparadores. Este número excesivo de comparadores es la mayor desventaja del convertidor A/D paralelo.
Otra desventaja de este tipo de convertidor es que el código de salida no es binario. Sin embargo, se puede transformar a binario por medio de compuertas. La ventaja más grande del convertidor A/D tipo paralelo es su velocidad de conversión. La palabra digital completa se encuentra disponible después del tiempo de retardo de los comparadores y las compuertas codificadoras. Se le conoce algunas veces como "flash" o instantáneo.

El convertidor TRW TCD-1007J puede hacer una conversión de 8 bits en 33 ns.

Convertidor de Rampa sencilla o pendiente doble.

La figura 2-7a muestra un convertidor A/D usando un generador de rampa sencilla, comparador, contadores, BCD o binarios.

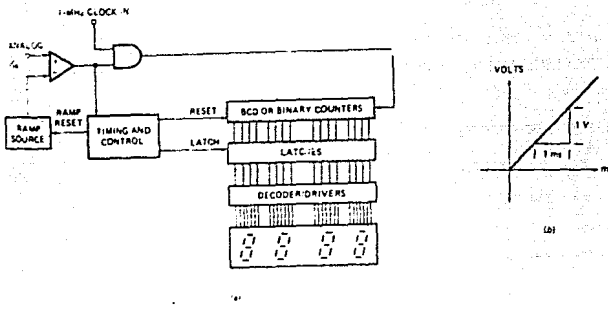


Figura 2-7. Convertidor A/D de pendiente unica. a) Diagrama de bloques. b) Pendiente de la rampa.

Al principio del ciclo de conversión los contadores y el generador de rampa sencilla se colocan en ceros (0's). Se alimenta un voltaje a la entrada positiva del comparador.

Como esta entrada es más positiva que la entrada negativa del comparador. La salida del comparador se pone en estado alto. Esto habilita la compuerta AND para dejar al reloj accesar los contadores y empezar a tomar la rampa sencilla. El voltaje rampa sencilla se va a positivo hasta que se excede el voltaje de entrada. Esto causa que la salida del comparador se ponga en estado bajo y deshabilite la compuerta AND. El reloj del contador, por lo tanto se inhibe.

Para inspeccionar la operación del circuito, supongamos una frecuencia de reloj de 1 MHz, cuatro contadores de BCD, y un voltaje de entrada de 2.000 V. Suponemos que la pendiente de la rampa sencilla es 1 V/ms, como se muestra en la figura 2-7b. Desde el principio del ciclo de conversión la rampa sencilla tomará 2 ms en alcanzar 2 V y apagar los pulsos de reloj de los contadores. En estos 2ms, los contadores habrán

alcanzado 2000 pulsos de reloj. La salida de los comparadores al ponerse en estado alto excitará a todos los Flip-Flops o latches y enviará la cuenta al despliegue. La lógica inserta un punto decimal en el lugar adecuado del despliegue de 7 segmentos para dar una lectura de 2.000, que corresponde directamente a la entrada analóga de voltaje de 2.000 V. Los voltajes restantes hasta 9.999 V se pueden convertir al equivalente BCD y desplegarlo de la misma manera. Este circuito es un voltmetro sencillo. Si se usan contadores binarios en lugar de contadores BCD, la salida será directamente codificada en binario.

Para aplicaciones tales como voltimetros digitales que puedan requerir resoluciones de una parte en 20000 o más. El convertidor A/D de rampa sencilla o de pendiente sencilla no es lo suficientemente estable. Las variaciones en el generador rampa sencilla son causadas por las variaciones de temperatura, o sensibilidad en el voltaje de entrada.

Convertidores de doble pendiente

El diagrama de bloques de un convertidor de doble pendiente A/D es el que se muestra en las figura 2-8a. El circuito es muy similar a un circuito de pendiente sencilla excepto para el interruptor en la entrada, el cual selecciona el voltaje de entrada o un voltaje de referencia, y las conexiones invertidas en las entradas del comparador.

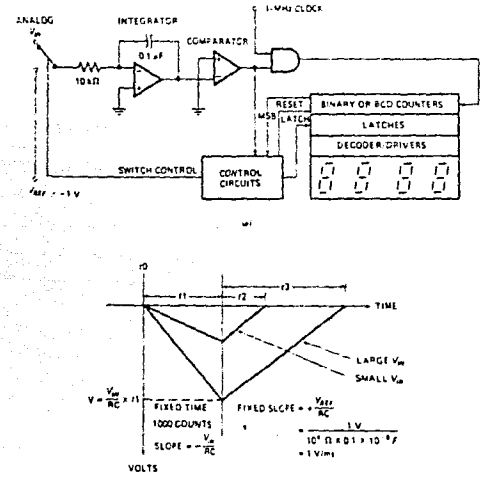


Figura 2-8. Convertidor A/D de doble pendiente. a) Diagrama de bloques. b) Salida integradora contra tiempo.

La primera parte del circuito es el generador de rampa, o integrador. La entrada inversora del amplificador operacional es sostenida a tierra virtual por el operacional. Un voltaje de, por ejemplo, 2 V aplicados en la resistencia de entrada de 10 K causará una corriente constante de 0.2 mA para que fluya a la resistencia en el punto o nodo de suma. Como esta corriente no puede fluir por

la alta impedancia del operacional, para conservar la entrada inversora del amplificador operacional en tierra virtual, el capacitor debe sustraer la misma corriente de su otra placa. Conforme el capacitor se carga, la salida de voltaje en el amplificador operacional se hace cada vez más negativa para mantener la corriente constante. El voltaje en un capacitor que está siendo cargado por una corriente constante es una rampa lineal.

Con un voltaje de entrada positivo, la salida del integrador es una rampa negativa. Como se muestra en la figura 2-8b. Un voltaje negativo causa que la salida sea una rampa lineal en la dirección positiva.
La pendiente de la rampa puede ser determinada fácilmente, usando la relación básica q=CV para capacitores y q=It. Igualando las dos expresiones y manipulando los términos resulta dV/dt=I/C. Ya que la corriente es igual a Vin/R, sabemos que dV/dt= Vin/(RC). Esto demuestra que para un voltaje Vin determinado la pendiente de salida, o dV/dt, es una constante. Para los valores mostrados y una entrada de +2V, la pendiente de salida es -2V/ms.

Ahora que se ha explicado el generador de rampa, podemos trazar un ciclo de conversión. Un ciclo de conversión empieza con la rampa en 0. Los contadores en cero. y el interruptor de entrada conectado al voltaje de entrada. El voltaje positivo dentro del generador causa una rampa de voltaje negativo.
Cuando la salida del integrador obliga a la entrada inversora del integrador a ser negativa, la salida del comparador se hace positiva y habilita la compuerta AND. Esto permite el acceso al reloj de los contadores. La salida del integrador puede ser una rampa negativa para un número fijo de ciclos. Esto se muestra para dos voltajes de entrada diferentes como t1 en la figura 2-8b. Cuando el contador alcanza la cuenta fijada, el circuito de control limpia los contadores a cero y conmuta la entrada del integrador a la referencia de voltaje negativo.

Un voltaje de entrada negativo causará que la salida del integrador sea una rampa positiva, como se muestra en t2 para la figura 2-8b.
Cuando la salida del integrador alcanza el cero, de nuevo , la salida del comparador se pone en estado bajo. El circuito de control detecta esta transición y habilita las salidas del contador dentro de los "latches".

Entonces limpia los contadores a cero y conmuta la entrada del integrador al voltaje de entrada. Esto empieza otro ciclo de conversión. El número de cuenta almacenado en los "latches" es proporcional al voltaje de entrada Vin.

La salida de un integrador en un tiempo fijo t1 produce una rampa de voltaje V, igual a (Vin/RC)t1. Para volver a cero,

el integrador debe producir un rampa de igual voltaje Para el periodo de referencia t2, el voltaje V es igual a (VREF/RC)*t2. Se pueden igualar las dos expresiones para

$$Vin/RC \times t1 = V_{REF}/RC \times t2$$

$$Vin \times t1 = V_{REF} \times t2$$

$$t2 = Vin \times t1/V_{REF}$$

Ya que RC aparece en cada lado de la ecuacion, se cancela. El significado práctico de ésto es que, como la misma resistencia y capacitor se usan para el periodo de integración y el periodo de referencia, las variaciones en R, y C no tendrán efecto en la precisión de la lectura de salida. Esta es una gran ventaja sobre el convertidor de una sola rampa. El resultado final de la ecuación anterior muestra que la salida del contador es directamente proporcional a Vin ya que VREF y t1 son constantes.
Para el circuito mostrado en la figura 8a. t1 es 1000 del reloj de 1 Mhz, ó 1 ms. y VREF es -1 V. Para una señal de entrada de 2V. t2 será (2V/1V) x 1000, ó 2000 cuentas. Un punto decimal en la derecha representa 2.000 V. La gráfica en la figura 2-8b representa la salida del integrador para un voltaje de entrada más pequeño. Por ejemplo, 0.8V. Para esto t2 será (0.8V/1V) x 1000, ó 800 cuentas. La lectura de salida será 0.800 V.
Ya que el convertidor de doble pendiente se usa en muchos instrumentos y voltimetros, es importante entender su operación. Para resumir : el voltaje de entrada desconocido se aplica a un contador para un número de ciclos fijo t1. El contador se limpia a cero y la entrada del integrador conectada a un voltaje de referencia. El número de cuentas requerido por la salida del integrador para regresar a cero es directamente proporcional al voltaje de entrada.

Las ventajas del convertidor de doble pendiente son su precisión, bajo costo, y la inmunidad a las variaciones de R y C por la temperatura. La desventaja del convertidor es su baja velocidad de conversión. Un convertidor capaz de resolver 1 parte en 20000 puede tomar 100 ms o más por ciclo de conversión.

Convertidores de carga balanceada

El convertidor de carga balanceada usa esencialmente el mismo circuito que el convertidor por doble pendiente de la figura 2-8a. Sin embargo, en lugar de estar conmutando continuamente entre el voltaje desconocido y el voltaje de referencia, la entrada se deja conectada al voltaje desconocido. Los pulsos de una corriente de referencia se insertan directamente al punto de suma del integrador por un tiempo fijo. El número de pulsos de la corriente de referencia requeridos para conservar el punto de suma en 0 voltios para este tiempo fijo es proporcional al voltaje de

entrada desconocido. La ventaja de esta técnica es que el voltaje en el capacitor de integración es conservado muy cerca de 0 voltios. Por lo tanto, la absorción dieléctrica por el capacitor no es un problema como sucede en los convertidores de alta precisión de doble pendiente.

Convertidores A/D mediante el uso de contadores y convertidores D/A

Tipo contador- Antes se mencionó la manera de producir una rampa conectando la salida de un contador a las entradas de un convertidor D/A. La rampa está hecha de pequeños pasos. El tamaño de los pasos está determinado por el número de bits de resolución del convertidor.

La figura 2-9 muestra como el voltaje de salida del convertidor D/A y contadores se puede usar, en lugar de un integrador para hacer un convertidor tipo rampa sencilla A/D. Al principio de un ciclo de conversión, los contadores se limpian y la salida del convertidor D/A es cero.

Cuando se alimenta un voltaje a la entrada del comparador, su salida se pone en estado alto y les habilita el reloj a los contadores. Cada pulso de reloj hace avanzar los contadores una cuenta e incrementa el voltaje del convertidor D/A en un paso.

Cuando la salida del convertidor D/A se hace más grande que el voltaje de entrada, la salida del comparador se mantiene baja. Esto apaga los pulsos de reloj a los contadores. Después el circuito de control habilita las salidas del contador y limpia los contadores para empezar el ciclo de nuevo.

Este método es más rapido que el tipo de doble rampa, pero requiere un convertidor D/A de precisión, otra desventaja es que para cada ciclo de conversión, el contador tiene que empezar en cero y contar hasta que la salida del D/A coincide con Vin.

Convertidor A/D por seguimiento

El convertidor A/D por seguimiento es un método que es algo más rápido para digitalizar señales tales como audio que cambia solamente en cantidades pequeñas de una muestra a la siguiente. Como se muestra en la figura 2-10, un convertidor A/D por seguimiento usa un contador ascendente/descendente en lugar de un contador solamente ascendente del tipo anterior. El reloj es dirigido tanto a la entrada ascendente

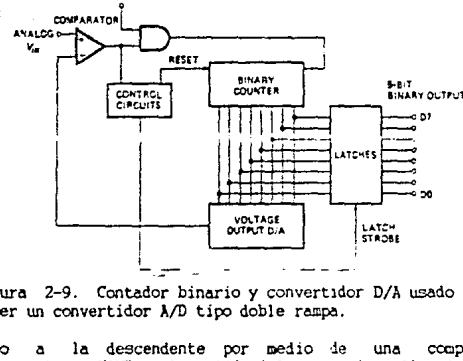


Figura 2-9. Contador binario y convertidor D/A usado para hacer un convertidor A/D tipo doble rampa.

como a la descendente por medio de una compuerta interruptora NAND en la salida del comparador. Al principio de una conversión para un voltaje de entrada, la salida del convertidor D/A es cero, así que la salida del comparador está en estado alto. Esto dirige al reloj hacia la entrada del contador ascendente. El contador cuenta en forma ascendente hasta que la salida del convertidor D/A es mayor que Vin.

La salida del comparador cambia a un estado bajo y dirige el reloj hacia la entrada del contador descendente. Sin embargo, si Vin no ha cambiado, se efectua un ciclo de la cuenta en forma descendente y producirá una salida del convertidor menor a Vin. La salida del convertidor cambiará al estado alto y dirige otra vez el reloj a la entrada del contador ascendente. Un ciclo de cuenta ascendente cambiara la salida anterior para repetir el proceso. El bit menos significativo de salida de este tipo de convertidor oscila, entonces, para un valor constante de Vin. Esto es una desventaja. Ya que la señal de entrada cambia, el contador cuenta en forma ascendente o descendente para "seguir el voltaje", de ahi el nombre del convertidor. La ventaja es que para digitalizar ondas senoidales solo tiene que contar hacia arriba o hacia abajo pocos ciclos de una muestra a otra en lugar de contar hacia arriba desde cero cada vez. El convertidor A/D tipo aproximaciones sucesivas es más rápido que los anteriores y no oscila.

Convertidor A/D de aproximaciones sucesivas

La gran ventaja de la conversión por aproximaciones sucesivas es que los N bits de resolución se pueden lograr solo con N pulsos de reloj. Por ejemplo, un convertidor de 8 bits por aproximaciones sucesivas requiere solamente 8 pulsos de reloj, comparado con 256 pulsos de un convertidor por contador. La figura 2-11 muestra un circuito por aproximaciones sucesivas para un convertidor. el corazón de este circuito es el registro de aproximaciones sucesivas

(SAR) tal como el MC14549, que funciona como sigue. Al inicio del ciclo de conversión el SAR, en el primer pulso de

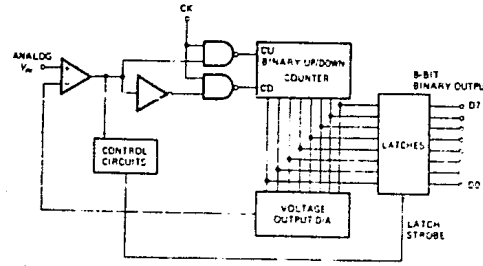


Figura 2-10. Contador binario asecendente/descendente y convertidor usado para hacer un convertidor A/D por seguimiento.

reloj, enciende su bit de salida más significativo al convertidor D/A MC1408. Entonces el SAR espera por la señal del comparador LM319 indicando si la salida del D/A es mayor o menor que la entrada de voltaje. Si la salida del comparador está en estado alto, entonces la salida del D/A es menor que Vin y el SAR mantendrá el bit más significativo encendido. Si la salida del comparador está en estado bajo, entonces la salida del D/A es mayor que Vin y el SAR limpiará el bit más significativo. En cualquier caso, en el siguiente pulso de relój el SAR encenderá el bit más significativo. Esto mantendrá o limpiará este bit dependiendo de la salida del comparador. El SAR continua hasta el bit menos significativo. Conserva un bit si la salida del D/A es menor que Vin. Se necesita solo un pulso de reloj de cada bit. Cuando todos los bits han sido completados, el SAR envia una señal de "fin de conversión" (EOC).

La señal EOC indica que las lineas de salida paralela tienen una palabra binaria válida representando el tamaño de una señal de entrada analógica. Si la señal EOC está atada a la entrada de "inicio de conversión" (SC), el convertidor ciclará continuamente. El MC14549 también tiene una salida serie que manda la salida digital de bit en bit hasta que el SAR lo determina.

El método de aproximaciones sucesivas para convertidores A/D es similar a encontrar la altura de una mesa apilando bloques por peso binario 128, 64, 32, 16, 8, 4, 2, y 1 Cmts. de altura. El bloque más significativo de 128 cmts está primero. Si está muy alto, es separado y se le graba un cero para esa posición de bit. Entonces se intenta con el siguiente bloque más significativo añadiendole al apilamiento.

En cualquier momento añadir mas bloques hace que la pila sea más grande que la mesa, ese bloque es marcado con un cero y separado. Si el añadir un bloque no hace la pila más alta que la mesa, entonces se graba un "1" para ese bit. Cada bloque requiere sólo un intento. Cuando el proceso ha concluido, el resultado es una palabra de 8 bits que representa la altura de la mesa.

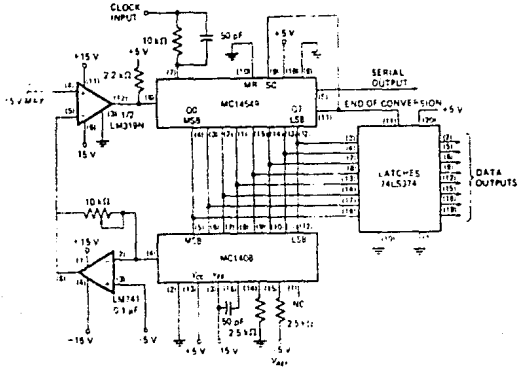


Figura 2-11. Convertidor A/D de aproximaciones sucesivas

El circuito mostrado en la figura 2-11 usa partes ya disponibles. El máximo rango de entrada es +/- 5 V. La R y C en la entrada del reloj son para proteger el MC14549 de daño si se interrumpe accidentalmente el suministro de energía antes de que el generador de pulsos alimente el reloj. Notese que la entrada positiva del convertidor de corriente a voltaje está atada a - 5V en lugar de a tierra. Esto recorre la entrada analógica en el intervalo de -5V a +5V, en lugar de 0 a 10V.

II-3 ESPECIFICACIONES DE A/D

Las especificaciones discutidas para convertidores D/A tales como resolución, precisión, linealidad, etc., se usan también para convertidores A/D. Otro término usado en convertidores A/D es el error de cuantización, o el error entre el valor analógico y su representación digital. Idealmente, el máximo error de cuantización es de +/- 1/2 del bit menos significativo, este es otro método de especificar resolución.
Otro parámetro importante para convertidores A/D es el tiempo de conversión, o el tiempo necesario para digitalizar cada muestra.

Este rango de nanosegundos a milisegundos depende del tiempo de conversión y el número de bits. Las aplicaciones de baja velocidad, alta resolución usualmente utilizan convertidores tipo doble pendiente. Las aplicaciones de alta velocidad, resolución media utilizan convertidores de aproximaciones sucesivas, y las aplicaciones de alta velocidad utilizan convertidores tipo paralelo o "flash".

III- PROCESAMIENTO DE SEÑALES DIGITALES

III-1 INTRODUCCION A LA TRANSFORMADA DISCRETA DE FOURIER

Muestreo y ventanas de una señal

Cuando tabulamos una f(t), estamos limitandola a un intervalo de tiempo y dentro de ese intervalo estamos calculando sus valores para ciertos instantes. Esto corresponde a las operaciones de ventana y muestreo.

Usemos una señal cosenoidal de 1 voltio de amplitud y una frecuencia de 12.5 Hz, es decir, $x(t) = \cos 2\pi 12.5t$. Tomemos el intervalo 4/12.5=0.32 seg. la gráfica tendrá un intervalo de 0.01 seg entre muestras. Los valores están tabulados en la tabla 3-1 y graficadas en la figura 3-1. Observando la figura 3-1, que es la gráfica correspondiente a una ventana de una onda cosenoidal muestrada, la ventana se define decidiendo la traza de la onda en un intervalo de tiempo finito, en este caso de 0.00 a 0.31 seg. Se muestra solamente por la decisión de encontrar y mostrar sus valores verdaderos solo en 32 puntos de ese intervalo de tiempo. En términos de procesamiento de señales, la figura 3-1 se puede entender como obtenida de la multiplicación mostrada en la figura 3-2. La figura 3-2a es la forma de onda en este caso una cosenoidal obtenida , a la cual se le tomará una muestra. La figura 3-2b es la ventana rectangular (pulso cuadrado). Su producto, mostrado en la figura 3-2c, es una onda cosenoidal con una ventana rectangular.

	x(t)	t	x(t)	t	x(t)	t	x(t)
0.00	1.000	0.08	1.000	0.16	1.000	0.24	1.000
0.01	0.707	0.09	0.707	0.17	0.707	0.25	0.707
0.02	0.000	0.10	0.000	0.18	0.000	0.26	0.000
0.03	0.707	0.11	0.707	0.19	0.707	0.27	0.707
0.04	1.000	0.12	1.000	0.20	1.000	0.28	1.000
0.05	0.707	0.13	0.707	0.21	0.707	0.29	0.707
0.06	0.000	0.14	0.000	0.22	0.000	0.30	0.000
0.07	0.707	0.15	0.707	0.23	0.707	0.31	0.707

Tabla 3-1. Los Valores de $x(t)= \cos 2\pi 12.5t$, cada 0.01 seg. sobre el intervalo de 0.00 a 0.31 s.

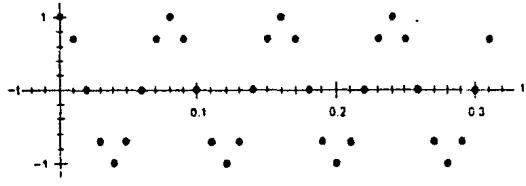


Figura 3-1. Una onda cosenoidal muestreada y con ventana.

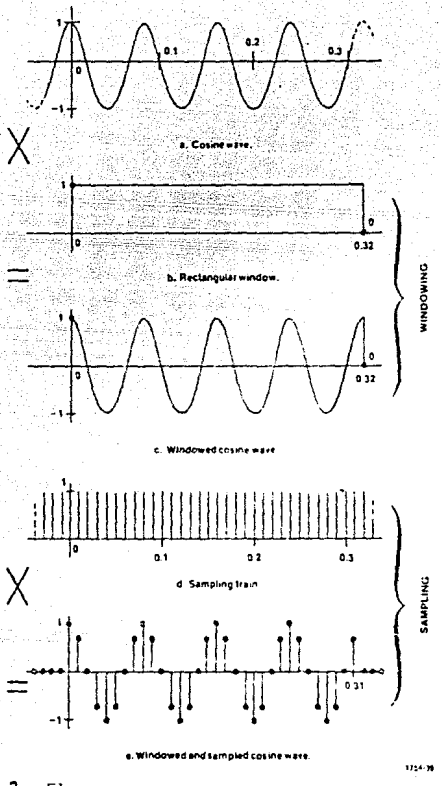


Figura 3-2. El muestreo y las ventanas equivalen a varias multiplicaciones de señales.

En las figuras 3-2b y c, obsérvese los puntos negros de terminación en los extremos de la ventana. Esto no es realmente parte de la ventana, pero es una forma de ilustrar los valores en los extremos de la misma.

Después de efectuar la ventana a la señal, el muestreo se representa por un tren de impulsos de amplitud unitaria mostrados en la figura 3-2d. Cada impulso empieza de cero y regresa instantáneamente a cero. Así, cada impulso es diferente de cero, sólo en cierto intervalo de tiempo.
Cuando este tren de impulsos, o tren de muestreo, se multiplica por la ventana de la onda senoidal, el producto (la fig 3-2e) contiene la ventana y la muestra de una onda coseno. Cada impulso entonces tiene una área correspondiente al valor del coseno en ese punto.
Las posiciones de la muestra en el tiempo, corresponden a las posiciones de los impulsos, en el tiempo.

En términos prácticos, los resultados de la figura 3-2e y la tabla 3-1 se pueden obtener usando un convertidor analógico-digital. Esto involucra realmente una multiplicación y generación de formas de onda, como se muestra en la figura 3-2.
La figura 3-2 corresponde al disparo de la entrada del convertidor analógico-digital para su activación o desactivación.
Cuando se habilita al convertidor, retenemos la forma de onda que va a ser digitalizada. Cuando se desactiva el convertidor, bloqueamos la señal. Y esto equivale a multiplicar.
En la práctica, el muestreo se efectua de la misma manera. La ventana de la señal se muestrea a intervalos iguales. Cuando se activa la ventana de muestreo, un capacitor toma el valor de la señal, durante el tiempo de muestreo. Entonces obtenemos un sólo valor por cada punto de muestreo, como se muestra en la figura 3-1 y 3-2e. Estos valores se pueden almacenar en una memoria de computadora. Luego con una computadora o un sistema digital determinado es posible procesar los datos.

III-2. TRANSFORMADA DISCRETA DE FOURIER DFT EN DATOS MUESTREADOS

La figura 3-2e se puede transformar al dominio de la frecuencia aplicando la transformada de Fourier (integral de Fourier) sobre el intervalo de la ventana al producto de la señal y el tren de pulsos. Tenemos :

$$X_d(k\Delta f) = \Delta t \sum_{n=0}^{N-1} x(n\Delta t) e^{-j2\pi k\Delta f n\Delta t} \qquad (3\text{-}1)$$

Esta expresión permite transformar un conjunto de muestras en el tiempo, tal como en la tabla 3-1. a un conjunto de muestras en el dominio de la frecuencia.

Con algunas manipulaciones adicionales, podemos desarrollar la transformada inversa de Fourier DFT.

La DFT inversa se expresa como :

$$x(n\Delta t) = \Delta f \sum_{k=0}^{N-1} X_d(k\Delta f) e^{j2\pi k\Delta f n\Delta t} \qquad (3\text{-}2)$$

Esta expresión permite transformar un conjunto de muestras, en el dominio de la frecuencia, calculadas por medio de TDF (DFT) a un conjunto de muestras en el dominio del tiempo.

En ambas expresiones tenemos las siguientes variables:

N= Número de muestras a considerar.
Δt= El tiempo entre muestras, conocido como intervalo de muestreo. De aqui. $N\Delta t$ produce la longitud de la ventana, también conocida como longitud de registro de tiempo.

Δf= El intervalo de muestreo en el dominio de la frecuencia y es igual a $1/N\Delta t$.
n= El índice de muestreo en tiempo. Sus valores son n=0,1,2,....N-1.

k= El índice para el conjunto de componentes discretas calculados en la frecuencia. Sus valores son k=0,1,2,..,N-1.

$x(n\Delta t)$= Es el conjunto discreto de muestras en el tiempo que definen la forma a transformar.

$X(k\Delta f)$= El conjunto de coeficientes de Fourier obtenidos por DFT de $x(n\Delta t)$.

e = Base del logaritmo natural.

j= El simbolo en notación compleja, indicando la parte imaginaria de una cantidad compleja ($j=\sqrt{-1}$).

Por medio de algunas manipulaciones, haciendo $\Delta t=1$, tal que $\Delta f=1/N$, tenemos una forma más común de DFT y su

inversa:

$$\text{DFT} \qquad X_d(k) = \sum_{n=0}^{N-1} x(n) e^{-j2\pi kn/N}$$

$$\text{DFT inversa} : \quad x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X_d(k) e^{j2\pi kn/N}$$

ya que 1/N es solamente un término de escalamiento, se debe incluir en cualquiera de las transformaciones pero no en ambas.
Por facilidad de cálculo, usamos la identidad de Euler ($e^{\pm j\Theta} = \cos\Theta \pm j \operatorname{sen}\Theta$) para cambiar la exponencial compleja y obtener la DFT y DFT inversa como sigue :

$$X_d(k) = \frac{1}{N} \sum_{n=0}^{N-1} x(n) \frac{\cos 2\pi kn}{N} - j x(n) \frac{\sin 2\pi kn}{N}$$

(3-3) y

$$x(n) = \sum_{k=0}^{N-1} X_d(k) \frac{\cos 2\pi kn}{N} + j X_d(k) \frac{\sin 2\pi kn}{N}$$

Habiendo desarrollado este par de expresiones, es más fácil calcular la transformada discreta de Fourier para cualquier forma de onda. Por ejemplo, se puede calcular la DFT de la onda cosenoidal de los valores listados en la tabla 3-1. Hay 32 muestras, así que k y n toman valores de 0,1,2,. . . .,31 Los valores de x(n) se toman directamente de la tabla, donde x(0) corresponde al tiempo cero y tiene el valor de uno, x(1) corresponde a 0.01 seg y tiene el valor de 0.70, etc. Cada coeficiente de Fourier, Xd(k), se calcula por la sumatoria [x(n)cos 2πkn/N - j sin 2πkn/N] para todos los valores de k.
Por ejemplo, si k=0 evaluamos la sumatoria anterior para n=0 hasta n=31. Luego hacemos k=1 y repetimos la evaluación de la sumatoria para todo el rango de n. Continuando hasta que en la sumatoria se obtiene el valor de k=31. Cuando este ciclo está completo, tenemos un conjunto de coeficientes de Fourier para las 32 muestras en el dominio del tiempo de la onda cosenoidal.
Estos coeficientes de Fourier definen la onda senoidal en el dominio de la frecuencia compleja en 32 puntos discretos de la frecuencia.

Existen 32 x 32 = 1024 operaciones para una transformada discreta de Fourier de 32 puntos.
El uso de un programa para calcular la transformada discreta de Fourier facilita las operaciones iterativas que requiere, ya que lo que cambia son los valores de los índices, k y n, y los valores de la muestra, x(n).
El uso del programa en Basic de la figura 3-3 calcula la transformada discreta de Fourier resultando los valores de la parte izquierda de la tabla 3-2.
Hay muchos números en la parte izquierda de la tabla 3-2, pero observamos que son muy pequeños comparados con los dos números mostrados en negrillas (0.5 y 0.5002). Los números más pequeños son debido al ruido digital y se deben

considerar nulos comparados con los dos números principales, que son los componentes del espectro de la onda cosenoidal dada en la figura 3-1.

```
100 DIMENSION  X(31) XR(31) XI(31)
105 LET PI=3.1416 LET X=0 LET EM=1
110 FOR I=0 TO 31
115 LET X(I)=COS(2*PI*12.5*I/100)
120 NEXT I
130 STOP

200 LET XR=0 LET XI=0
205 FOR K=0 TO 31
210 FOR N=0 TO 31
215 LET AG=2*PI*K*N/32
220 LET XR(K)=XR(K)+X(N)*COS(AG)/32
225 LET XI(K)=XI(K)-X(N)*SIN(AG)/32
230 NEXT N
235 NEXT K
240 STOP

300 PRINT "AJAL"
305 PRINT "32-POINT DFT " #20 "COEFFICIENT" #40 "FOURIER COEFFICIENTS"
310 PRINT #20," INDEX" #35,"REAL PART" #54 "IMAGINARY PART"
315 FOR I=0 TO 31 PRINT #23,I #32,XR(I) #54 XI(I) NEXT I
320 STOP
```

Definir arreglo de datos

Calcular TRF de 32 puntos del arreglo X el arreglo real XR y el imaginario XI.

resultados

Figura 3-3. Programa en BASIC para calcular una TRF de 32 puntos.

Si usaramos un algoritmo de transformada rápida de Fourier (FFT) evalua la mitad de los coeficientes de Fourier, sólo lo referente al dominio positivo de la frecuencia.
Es frecuente atribuir una gran diferencia entre la transformada rápida de Fourier (FFT) y la transformada discreta de Fourier (DFT), pero en realidad la FFT (TRF) no es diferente de la TDF (DFT), ya que sólo es un algoritmo para calcular la TRF (DFT). La forma exacta de operación no es tan importante mientras se recuerden los conceptos fundamentales de la teoría de Fourier y como se llega a la TRF (FFT) por medio de TDF.
Observamos la tabla 3-2. Excepto por algún ruido digital, y el orden de los coeficientes, los resultados de la TRF (FFT) y TDF (DFT) son iguales. La TRF (FFT) obtiene los mismos resultados de una manera más rápida.

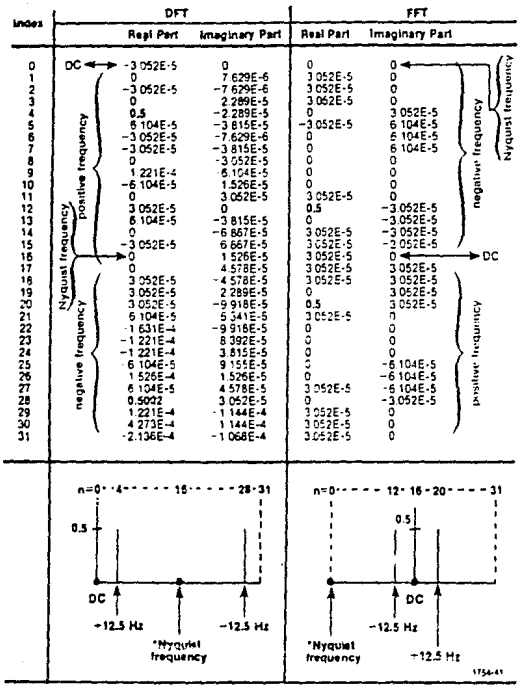
Regresando a las matemáticas de la DFT, la cual por medio de la FFT, se evalua de una manera más eficiente,

$$X_d(k) = \frac{1}{N}\sum_{n=0}^{N-1} x(n)e^{-j\pi nk/N} \qquad (3\text{-}4)$$

o por medio de Euler :

$$X_d(k) = \frac{1}{N}\sum_{n=0}^{N-1} x(n)e^{-j\pi nk/N} \qquad (3\text{-}5)$$

es evidente que los resultados vienen en dos partes - una parte real y una parte imaginaria, precedida por el operador imaginario, j. Esta es la forma rectangular más usada en la mayoría de las implementaciones de la FFT.



El intervalo de frecuencia esta dado por $\Delta f = 1/N\Delta t$. La letra tipo negrita indica los componentes de frecuencia para la onda cosenoidal en la tabla 3-1; las otras son practicamente cero.

La frecuencia de Nyquist es la frecuencia más alta que se puede definir en un muestreo. Para muestras igualmente espaciadas, la frecuencia de Nyquist es 1/2 .

Tabla 3-2. Transformada de Fourier de 32 puntos a partir de los datos de la tabla 3-1

III-3 TRANSFORMADA RAPIDA DE FOURIER

La transformada rápida de Fourier comprende un grupo de algoritmos, para calcular la TDF con una reducción considerable en el número de iteraciones en el cálculo. Naturalmente, el desarrollo de estos algoritmos representan el mayor paso en la realización del equipo digital moderno para análisis de Fourier.

Observando los pasos involucrados en el calculo de la DFT. La expresión :

$$(3.6)\quad X(m) = \sum_{n=0}^{N-1} x(n)\, W^{mn}$$

se puede expresar por el arreglo de ecuaciones

$$\begin{aligned}
X(0) &= x(0)W^0 + x(1)W^0 + x(2)W^0 + \cdots + x(N-1)W^0 \\
X(1) &= x(0)W^0 + x(1)W^1 + x(2)W^2 + \cdots + x(N-1)W^{N-1} \\
X(2) &= x(0)W^0 + x(1)W^2 + x(2)W^4 + \cdots + x(N-1)W^{2(N-1)} \\
&\cdots \\
X(N-1) &= x(0)W^0 + x(1)W^{N-1} + x(2)W^{2(N-1)} + \cdots + x(N-1)W^{(N-1)^2}
\end{aligned} \qquad (3\text{-}7)$$

Ya que $W^0 = 1$, se podrian hacer algunas simplificaciones en el arreglo. Sin embargo, es conveniente dejar el término W^0.

Bajo esta condición, se puede ver que el calculo de cualquier componente espectral en particular, requiere de N operaciones de multiplicación complejas y N operaciones de suma involucradas al calcular una componente de frecuencia dada.

De el arreglo, aparece como que deberiamos calcular N componentes espectrales. Sin embargo, si x(n) es real, solamente son necesarias N/2 componentes, y una expansión completa de la forma expresada en (3-7) necesitaría el calculo para la forma mencionada.
Esto resulta en un mínimo aproximado de $2N \times (N/2) = N^2$ de operaciones complejas. Para señales de magnitud considerable, este número puede ser muy grande, con un tiempo de proceso inadecuado.

El arreglo (3-7) se puede expresar en forma matricial como:

$$\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ \cdot \\ \cdot \\ \cdot \\ X(N-1) \end{bmatrix} = \begin{bmatrix} W^0 & W^0 & W^0 & \cdots & W^0 \\ W^0 & W^1 & W^2 & \cdots & W^{N-1} \\ W^0 & W^2 & W^4 & \cdots & W^{2(N-1)} \\ \cdot & \cdot & \cdot & & \cdot \\ \cdot & \cdot & \cdot & & \cdot \\ \cdot & \cdot & \cdot & & \cdot \\ W^0 & W^{N-1} & W^{2(N-1)} & \cdots & W^{(N-1)^2} \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ \\ \\ \\ x(N-1) \end{bmatrix}$$

Esto se puede expresar como

(3-8) $$\bar{X} = [W]\,\bar{x}$$

donde $\bar{X}$ es el vector columna definiendo la transformada, $\bar{x}$ es el vector columna que define el tiempo discreto de la senal, y [W] es la matriz cuadrada de NxN.

La base para la transformada rapida de Fourier (TRF) es un algoritmo presentado por Cooley y Tukey [5,18], y se puede entender como una factorización de matriz de (3-8). Asumimos que N puede escogerse como un multiplo entero de 2, y definir un entero L como :

(3-9) $$L = \log_2 N$$

(3-10) $$N = 2^L$$

Luego [W] se factoriza en L matrices de la forma

$$W=[W1][W2]\ldots[WL] \quad (3\text{-}11)$$

Cada renglon de las matrices individuales tiene la propiedad que contiene solamente dos terminos diferentes de cero : La unidad y W^k donde k es algún entero.

Substituyendo (3-11) en (3-8) resulta

$$\bar{X} = [W1][W2]..[W_L]\,\bar{x} \quad (3\text{-}11A)$$

donde el subindice de $\bar{X}_s$ se usa solamente para denotar el hecho que los componentes de este vector aparecen en diferente orden que el supuesto en la definición original de X. [WL] opera sobre $\bar{x}$ dando un nuevo vector. Cada componente del nuevo vector se puede obtener por una multiplicacion y una suma, ya que todos, a excepcion de 2 elementos son nulos. Como hay N componentes de $\bar{x}$, habrá N sumas y multiplicaciones complejas requeridas por este proceso

El nuevo vector opera en $\{W_{L-1}\}$, y el proceso se repite hasta que (3-11A) se satisface. Así, inicialmente aparece como que el número de operaciones es $NL=N\log_2 N$ sumas complejas y $N\log_2 N$ multiplicaciones complejas. Sin embargo, aprovechando algunas propiedades de simetría y el hecho que muchos de los términos W^k son unitarios, el verdadero número de operaciones complejas se reduce a la mitad o más.

Como un peor caso, establecemos el valor $N\log_2 N$ como el número de operaciones involucradas en el algoritmo Cooley-Tukey para TRF.
En la tabla 3-3 se muestran las proporciones del número requerido con el uso directo del TDF.

N	N(TDF)	N(TRF)	N(TRF/ N(TDF)
16	256	64	0.250
32	1024	160	0.156
64	4096	384	0.0938
128	16,384	896	0.0547
256	65,536	2048	0.0312
512	262,144	4608	0.0176
1024	1,048,576	10,240	0.0098
2048	4,194,304	22,528	0.0054
4096	16,777,216	49,152	0.0029

N= Número de puntos en la muestra

$N(TDF)=N^2$ = Número aproximado de operaciones aritmeticas complejas con TDF.

$N(TRF)= N \log_2 N$= Número aproximado de operaciones complejas con TRF.

Tabla 3-3. Comparación del número aproximado de calculos requeridos para TDF y TRF.

Hay un número de variaciones en los algoritmos de TRF indicados por (3-11A). La mayoría de estos algoritmos se pueden clasificar como a) locales (in-place) o b) salida/entrada natural. Un algoritmo local es el que dada una componente de un vector cualquiera intermedio se puede almacenar en la misma localidad ocupada por la componente correspondiente del vector precedente, reduciendo así, el total de almacenamiento requerido. Desafortunadamente, la mayoría de los algoritmos locales resultan, ya sea, una salida del espectro mostrando las componentes en desorden, o requieren que los datos de entrada sean ordenados antes de entrar al arreglo computacional.
Por otra parte, los algoritmos de entrada-salida natural, requieren de más memoria interna para mantener este orden natural. Así si el tamaño de la memoria es extremadamente critico, sería deseable usar un algoritmo local aunque los datos de entrada o salida tengan que ser ordenados. Este proceso de ordenación se conoce como operación de combinación.

III-4 ALGORITMOS DE LA TRANSFORMADA RAPIDA DE FOURIER

Inspeccionemos algunas variaciones de los algoritmos básicos de TRF y sus diversas estructuras operativas.
Los casos mostrados se han organizado con el fin de dar claridad en la presentación más que óptima eficiencia en la organización.
Para cada variación del algoritmo presentado, se dará un esquema del procesamiento de señales referido como gráfica de flujo de señales.

Esta terminología se usa, ya que existe similitud física y matemática con una gráfica de flujo de señales como la que se usa en teoría de sistemas. En todos los casos, usaremos el esquema de un sistema de 8 puntos. La tendencia general en cada caso la podemos observar de una inspección del sistema de 8 puntos, así que podemos generalizar el resultado a un número arbitrario de puntos satisfaciendo a (3-22) y (3-23).
Mostraremos un total de 8 gráficas, las primeras 4 (figuras 3-4 a 3-7) representan variaciones del algoritmo original de Cooley-Tukey.

Estos algoritmos se han derivado de un proceso llamado Decimación en tiempo (decimation in time) [5]. La segunda parte (figuras 3-8 a 3-11) representan variaciones de un algoritmo desarrollado, en forma independiente, por Sande, Tukey, Stockham y otros.
Las derivaciones de estas formas se han logrado por un proceso llamado decimación en frecuencia (decimation in frequency) [5]. Por convención, designaremos a la decimación en tiempo como algoritmos de tipo 1 y la decimación en frecuencia como algoritmos tipo 2.
Además de las gráficas de flujo de señales presentamos dos cartas de flujo para computadora (figuras 3-12 y 3-13) correspondiente a dos algoritmos tipo 1.
De la definición de TDF y TDF inversa como en (3-A) y en (3-B), se observa que las únicas diferencias en las dos funciones es la presencia del factor 1/N en la función inversa y el reemplazo de W por W^{-1}. En realidad, las formas exactas dadas aquí son algo arbitrarias, ya que hemos usado un número dado de formas. Una modificación común es la colocación del factor 1/N en la TDF, en lugar de en la TDF inversa.

(3-A) $$X(m) = \sum_{n=0}^{N-1} x(n) W^{mn}$$

(3-B) $$x(n) = \frac{1}{N} \sum_{m=0}^{N-1} X(m) W^{-mn}$$

Las gráficas de flujo de señales y cartas de computación están organizadas para el cálculo de la TDF usando (3-A).
Se puede usar el mismo esquema para calcular la TDF inversa con (3-B) con las siguientes modificaciones :

1) la cantidad $W = e(-j2\pi/N)$ se reemplaza por $W = e(j2\pi/N)$. Esto se logra invirtiendo los signos de todos los términos senoidales. b) Todos los términos se deben multiplicar por 1/N. Esto lo podemos hacer ya sea a la entrada o a la salida. c) La entrada es ahora la transformada X(m) y la salida es la señal en el tiempo x(n).

El procedimiento para leer gráficas de flujo de señales es el siguiente. Usar de referencia la figura 3-4. Hay 4 columnas y cada columna contiene 8 entradas. Por claridad,

la variable bidimensional y(l,m) se usara para denotar el valor de un nodo en el arreglo, donde l es el número de la columna y m es el número del coeponente dentro de la columna. Notese que en general, l varia sobre el rango 0<=l<=L con l=0 en la izquierda y m varia sobre el rango 0<=m<=N-1 con m=0 en la parte superior.

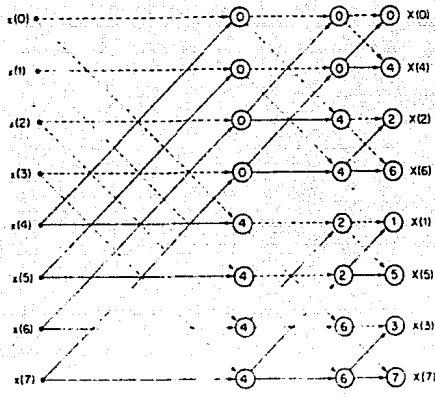


Figura 3-4. Algoritmo local con entrada natural y salida mezclada (tipo 1)

Lo anterior no implica que tengamos que apartar almacenamiento extra para todos los elementos de la variable bidimensional y(l,m). En realidad, con los algoritmos del tipo local, la columna y(l+1,m) se almacena en el mismo arreglo como y(l,m). Con otros algoritmos, es usualmente necesario mantener más almacenamiento interno, pero es rara vez necesario proveer almacenamiento total para cada variable del proceso, sin embargo, en este caso es muy conveniente ver todos los elementos del arreglo como un conjunto diferente de variables. A este respecto, la entrada x(m) se almacena en y(0,m) cuando no se requiere mezclado (scrambling) y en y(0,m) cuando se requiere mezclado en la entrada.
La entrada X(m) la obtenemos en la localidad y(L,m) cuando no se requiere mezclado de salida y la localidad y(L,m) cuando se requiere mezclado a la entrada.
El procedimiento que sigue hace referencia a algoritmos del tipo 1 de las figuras 3-4 a 3-7.
El nodo correspondiente a la columna l y renglon m, la variable y(l,m) se encuentra de una ecuación de la forma :

$$y(l,m) = y(l-1,m1) + W^{r}\ y\ (l-1,m2) \quad (3.12)$$

donde m1, m2, y r son funciones de la localidad dentro del arreglo y el algoritmo en particular.

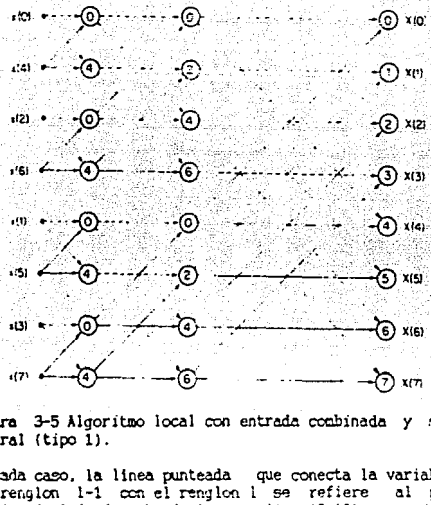


Figura 3-5 Algoritmo local con entrada combinada y salida natural (tipo 1).

En cada caso, la línea punteada que conecta la variable en el renglon 1-1 con el renglon 1 se refiere al primer término de lado derecho de la ecuación (3-12), por ejemplo, el término menos significativo.

La línea llena se refiere al segundo término en el lado derecho de la ecuación (3-12), por ejemplo los términos significativos. El número en el círculo es el grado de W como se indica por el entero r en (3-12).
El algoritmo de la figura 3-4 es uno del tipo local como ilustraremos en el diagrama de flujo de la figura 3-12. Esto significa que cuando una columna particular de datos se calcula, los valores se pueden almacenar en las mismas localidades ocupadas por la columna anterior. Así se requieren N valores complejos de la señal para almacenamiento.
En el algoritmo anterior, la señal de entrada aparece en orden natural y la salida es mezclada. Una modificación en donde la entrada es "mezclada" y la salida aparece en orden natural, se muestra en la figura 3-5.
La grafica de flujo de señales de un algoritmo con entrada-salida natural se muestra en la figura 3-6. Es posible demostrar que la propiedad local no se mantiene, así que es necesario mantener el almacenamiento por 2N valores complejos de la señal.
Mostramos un arreglo interesante en la figura 3-7. En esta forma la geometría de cada etapa es idéntica a las restantes.

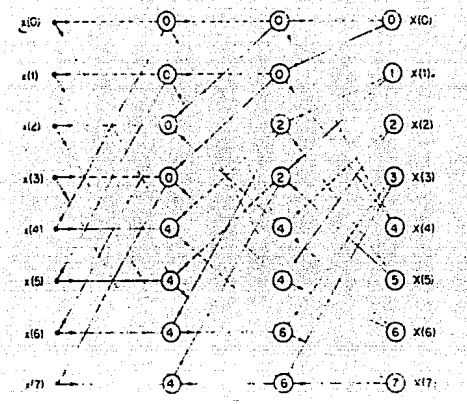


Figura 3-6 Algoritmo con entrada/salida natural requiriendo mas memoria interna (tipo 1).

Esta propiedad podria ser ventajosa en el diseño de ciertos sistemas. Sin embargo, esta forma requiere más almacenamiento interno y los datos de entrada se deben mezclar, así que el algoritmo no es ni local ni una versión de entrada-salida natural.
Las gráficas de flujo de señales ilustradas en las figuras 3-8 a 3-11 representan algoritmos tipo 2. Con todas estas gráficas de flujo, las ecuaciones que relacionan columnas sucesivas tienen una de las dos formas posibles, dependiendo de la localidad dentro del arreglo. Una forma es la ecuación.

$$y(l,m) = y(l-1,m1) + y(l-1,m2) \quad (3\text{-}13)$$

la otra forma es la ecuación

$$y(l,m) = W^{i} [y(l-1,m1) - y(l-1,m2) \quad (3\text{-}14)$$

donde m1, m2, y r son funciones de la localidad dentro del arreglo y el algoritmo particular.

Referirse a la figura 3-8 para ilustración. Las componentes que corresponden a (3-13) son aquellas con dos líneas punteadas Las componentes que corresponden a (3-14) son aquellas con líneas llenas terminando en el lugar correspondiente marcado con un circulo.

Figura 3-6 Algoritmo con entrada/salida natural requiriendo mas memoria interna (tipo 1).

Esta propiedad podria ser ventajosa en el diseño de ciertos sistemas. Sin embargo, esta forma requiere más almacenamiento interno y los datos de entrada se deben mezclar, así que el algoritmo no es ni local ni una versión de entrada-salida natural.
Las gráficas de flujo de señales ilustradas en las figuras 3-8 a 3-11 representan algoritmos tipo 2. Con todas estas gráficas de flujo, las ecuaciones que relacionan columnas sucesivas tienen una de las dos formas posibles, dependiendo de la localidad dentro del arreglo. Una forma es la ecuación.

$$y(l,m) = y(l-1,m1) + y(l-1,m2) \quad (3\text{-}13)$$

la otra forma es la ecuación

$$y(l,m) = W^{r} \{y(l-1,m1) - y(l-1,m2) \quad (3\text{-}14)$$

donde m1, m2, y r son funciones de la localidad dentro del arreglo y el algoritmo particular.

Referirse a la figura 3-8 para ilustración. Las componentes que corresponden a (3-13) son aquellas con dos lineas punteadas. Las componentes que corresponden a (3-14) son aquellas con lineas llenas terminando en el lugar correspondiente marcado con un circulo

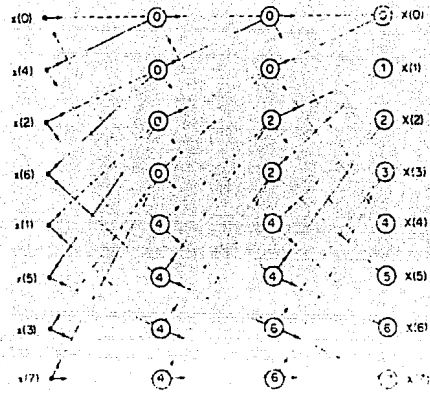


Figura 3-7 Algoritmo con geometria idéntica en cada etapa (tipo 1).

El entero en el circulo es el grado W indicado por r en la ecuación (3-14). El signo (-) aparece junto a la rama que se resta, y esto es siempre la rama más baja terminando en un circulo determinado.
Para ilustrar esta gráfica, las ecuaciones correspondientes a la primera columna de la figura 3-8 son :

$$
\begin{aligned}
y(1,0) &= y(0,0) + y(0,4) \\
y(1,1) &= y(0,1) + y(0,5) \\
y(1,2) &= y(0,1) + y(0,5) \\
y(1,3) &= y(0,3) + y(0,7) \\
y(1,4) &= W^{0}[y(0,0) - y(0,4)] \\
y(1,5) &= W^{1}[y(0,1) - y(0,4)] \\
y(1,6) &= W^{2}[y(0,2) - y(0,6)] \\
y(1,7) &= W^{3}[y(0,3) - y(0,7)]
\end{aligned}
$$

El algoritmo de la figura 3-8 es de la forma local, y los datos de salida se mezclan. Una forma modificada donde los datos de entrada se mezclan y la salida aparece en orden natural, se muestra en la figura 3-9.

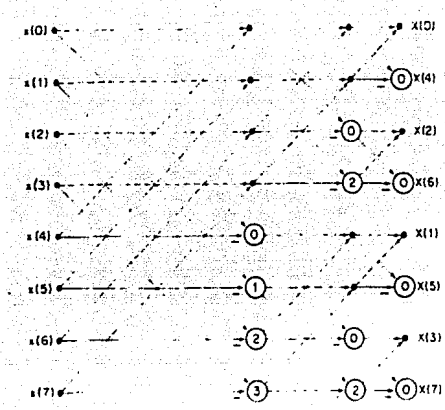


Figura 3-8 Algoritmo local con entrada natural y salida mezclada (tipo2).

La gráfica de señales de entrada-salida natural o tipo 2 está en la figura 3-10. Como en el caso de la versión de entrada-salida natural tipo 1, se requiere más memoria interna. Finalmente, un algoritmo que tiene geometría idéntica en cada etapa está en la figura 3-11.
Los diagramas de flujo para computadora correspondiente a dos algoritmos particulares previamente dados se muestran en la figuras 3-12 y 3-13, y también se ilustra un diagrama de flujo para efectuar la operación de mezclado, en la figura 3-14. Las ecuaciones dadas en estos diagramas de flujo representan una combinación de ecuaciones algebraicas normales y ecuaciones típicamente usadas en lenguajes de programación. Una ecuación de la forma x= x + y debe leerse como " agregar 'y' al valor encontrado en la localidad 'x' y almacenar el resultado en la localidad 'x'.

Mostramos el entero m en el diagrama de flujo en su rango natural. Por ejemplo, 0<m<N-1. Sin embargo, las variables dimensionadas en muchos lenguajes de programación sólo aceptan enteros positivos como argumentos.
El diagrama de flujo de la figura 3-12 corresponde al algoritmo de la figura 3-4.

Los enteros L y N en el diagrama de flujo corresponden a las definiciones de (3-9) y (3-10). La operación (^) sobre un entero se refiere al valor mezclado de ese entero en particular. Una rutina separada es capaz de efectuar la operación de mezclado: y se presenta al respecto, la figura 3-14.

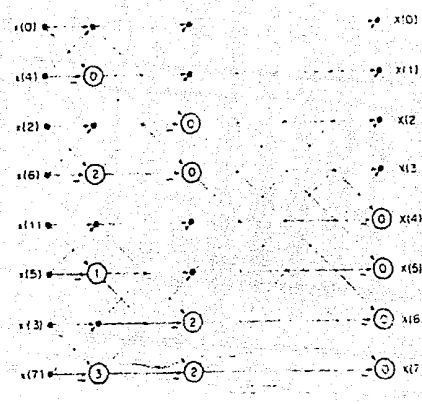


Figura 3-9 Algoritmo local entrada mezclada y salida natural (tipo 2).

Una de las operaciones se muestra en la figura 3-12 y está dada por :

$$Ie = [\ \overset{\frown}{Ic/Ia}\ ] \quad (3\text{-}15)$$

La operación $\overset{\frown}{[\ ]}$ se define como "la parte entera" de la cantidad dentro de los corchetes.
Esto es solamente el proceso de redondeo de las cifras al entero más grande igual o más pequeño que dicho número. Esta operación la conseguimos en algunos lenguajes de programación igualando una variable entera a la izquierda al cociente resultante a la derecha. En la operación particular de (3-15), el entero que resulta está mezclado.
El diagrama de flujo correspondiente al algoritmo de entrada-salida natural de la figura 3-6 se muestra en la figura 3-13. Como se espera, la cantidad de almacenamiento interno para esta forma es aproximadamente el doble que los algoritmos locales, ya que es necesario mantener los arreglos separados y1(m) y y2(m). Sin embargo, no se requiere la operación de mezclado ya que la organización es bastante más sencilla.

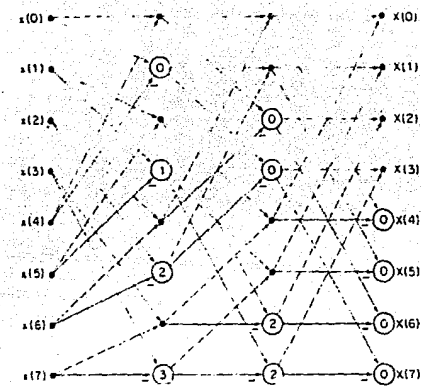


Figura 3-10. Algoritmo de entrada/salida natural requiriendo más almacenamiento interno (tipo 2).

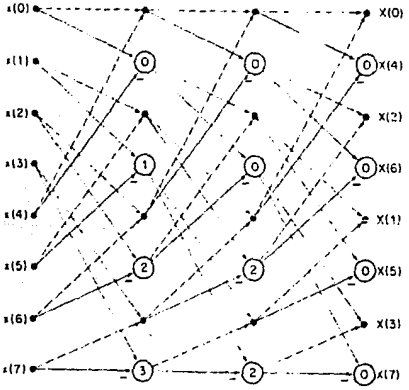


Figura 3-11.Algoritmo con geometria identica en cada etapa (tipo 2).

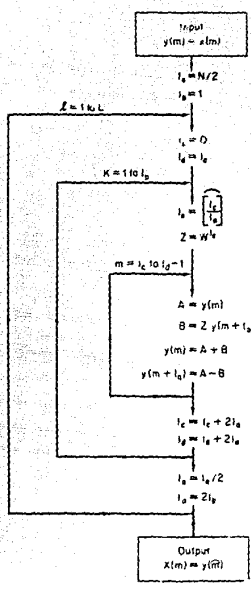


Figura 3-12. Diagrama de flujo correspondiente al diagrama de la figura 3-4.

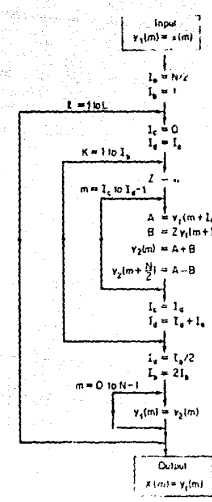


Figura 3-13. Diagrama de flujo correspondiente al diagrama de la figura 3-6.

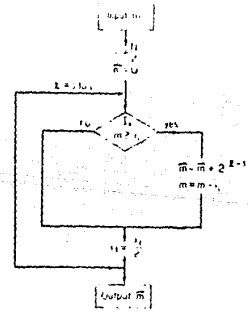


Figura 3-14. Diagrama de flujo del algoritmo de mezclado.

III-5 UN ALGORITMO PARA CALCULAR LA TDF

La transformada rápida de Fourier (TRF) no es un algoritmo único para evaluar la transformada discreta de Fourier (TDF).
Hay varios algoritmos que proveen una ventaja básica sobre las N^2 operaciones para la evaluación directa de de la TDF.
Una propiedad particular de los datos es el número de muestras.
La mayoría de los algoritmos de TRF operan en N muestras donde N es igual a dos (2) elevado a una potencia entera.
Pero existen algunos algoritmos para trabajar con N igual al producto de varios enteros, permitiendo más variaciones en los valores de N. Sin embargo, veremos a la clase de algoritmos diseñados para $N=2^k$, donde k es un entero. Los algoritmos de $N = 2^K$, también conocidos como algoritmos de la potencia 2, son muy directos y relativamente más rápidos de ejecutar que los algoritmos más grandes.
Es frecuente que la TRF esté basada en los algoritmos de Cooley-Tukey o Sande-Tukey.
Estos algoritmos difieren principalmente en su aproximación llamada, decimación en tiempo para el Cooley-Tukey y decimación en frecuencia para el Sandey-Tukey. Esta última, es la aproximación que vamos a analizar con la técnica de la potencia de 2.

El algoritmo de Sande-Tukey para evaluar la TDF.

Revisando una expresión para la TDF, vista anteriormente.

$$X_d(k) = \frac{1}{N} \sum_{n=0}^{N-1} x(n) e^{-j2\pi nk/N}$$

Por conveniencia la podemos expresar como :

$$A(n) = \sum_{k=0}^{N-1} x_d(k) W^{-nk}$$

Para n=0,1,...,N-1. Los datos en el dominio del tiempo están dados por $x_e(0)$, $x_o(1)$,... $x_e(N-1)$, y W es igual a la exponencial $e(j2\pi/N)$. Ya que el término 1/N que precede al signo de sumatoria en xd(k) es simplemente un término de escala, lo podemos omitir para mayor claridad.
La evaluación de la TRF de x (t) consiste de log N=M etapas, cada etapa requiere pares de calculos de la forma :

Y

Para los enteros definidos, r,s, y p entre 0 y N-1 y m entre 0 y M-1. Los resultados al final de las m etapas de calculo se denotan por $x_m(t)$, donde t=0,1,...,N-1. Tambien el algoritmo del ejemplo es el algoritmo local. Esto significa que los resultados presentes reemplazan los resultados anteriores.
Así, $x_{m+1}(t)$ sobre-escribe $x_m(t)$ al ir de la etapa m a la etapa m+1. El subindice m define una secuencia de arreglos que comprenden los contenidos de una localidad de almacenamiento asociada al final de la etapa m.
Para tener una mejor idea de como llevamos a cabo un calculo, consideramos una TRF de 16 puntos como ejemplo.
Con una inspeccion superficial podemos ver la esencia del algoritmo. Si queremos realmente implementar un algoritmo de TRF, revisar referencia [5,18].
La organización del calculo de la TRF se muestra en la figura 3-15.
Notese que los datos son separados por grupos en cada etapa, y conforme pasa cada etapa, los grupos se dividen en grupos más pequeños.
Esto continua hasta que la transformación está completa, con un dato por grupo.
Esta operación se conoce como decimación en frecuencia. Una decimación en tiempo está organizada en la forma inversa, el calculo lleva a grupos cada vez mayores.
Los datos de cada grupo del algoritmo de decimación en frecuencia son calculados de los pares de los correspondientes elementos del grupo anterior. Esto se indica en la figura 3-15 y se muestra con mayor detalles en el diagrama de la figura 3-16. Cada linea marcada en la figura 3-16 representa un punto del calculo. Aunque estas lineas están geometricamente separadas en la figura 3-16, en realidad están en el mismo lugar, punto o localidad de memoria en el algoritmo local.
Como habiamos visto, el calculo se efectua en pares y son de la forma :

$$x_{m+1}(r) = x_m(r) + x_m(s)$$

y

$$x_{m+1}(s) = [x_m(r) - x_m(s)]W^{r \cdot p}$$

Un ejemplo de tal par se muestra en la figura 3-16 en la etapa x_1 . $x_0(7)$ y $x_0(15)$ se añaden para obtener $x_1(7)$. Tambien, en el siguiente nodo, $x_0(0)$ se multiplica por W^0 y añadido al producto de x (8) y $-W^0$ para obtener x (8).

Notese en la figura 3-16 que las operaciones están agrupadas en cada etapa según si debemos multiplicar o no, los elementos por el factor $W^{r \cdot p}$. Esta eficiencia organizacional está incrementada por la ventaja de la simetria de las senoides y cosenoides para obtener los factores de giro (twiddle), indicados por W^0, W^1, y asi sucesivamente. Dependiendo del algoritmo de TRF en particular, estos factores de giro pueden generarse individualmente en cada

etapa, o pueden llamarse de una tabla generada de antemano

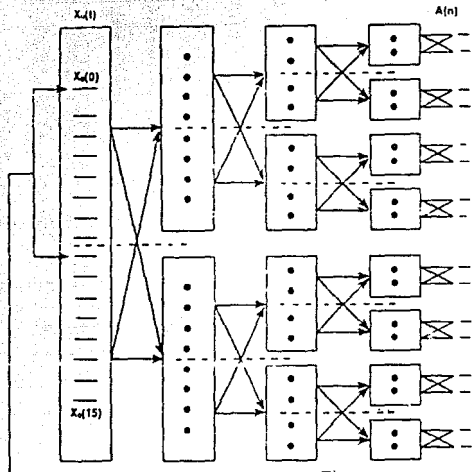


Figura 3-15. La organización general del algoritmo de decimación frecuencia.

Después de que la etapa final del algoritmo ha pasado. $X_0(t)$ será transformada a A(n), y los datos del dominio del tiempo se habrán transformado al dominio de la frecuencia. Como puede verse de la figura 3-16, A(n) ocurre en un orden mezclado.
El orden de estos coeficientes de Fourier puede reordenarse por un proceso llamado inversion de bits. Si los subindices de A(0) a A(15) son reexpresados en código binario puede determinarse la localidad correcta de cada componente de frecuencia invirtiendo o cambiando los bits de la dirección

		bit		
A(0)	= A(0000)	reversal	A(0000)	= A(0)
A(8)	A(1000)		A(0001)	A(1)
A(4)	A(0100)		A(0010)	A(2)
A(12)	A(1100)		A(0011)	A(3)
			.	
			.	
A(11)	A(1011)		A(1101)	A(13)
A(7)	A(0111)		A(1110)	A(14)
A(15)	A(1111)		A(1111)	A(15)

Tabla 3-4. La inversión de bits ordena los coeficientes de Fourier.

Esto se ilustra en la tabla 3-4 y es el paso final de la transformación.

Programa para el calculo de la TRF y TRF inversa

Los pasos anteriores para el calculo de la TRF, han sido implementados en el lenguaje BASIC en la figura 3-17.
Este programa provee los medios para el estudio adicional y la experimentación con el material de la TRF presentado.
Debemos señalar, sin embargo, que el programa en la figura 3-17 es solamente como programa de muestra, algo para experimentación preliminar. Cualquier aplicación formal debe contener rutinas de ensamblador o TRF por Hardware.

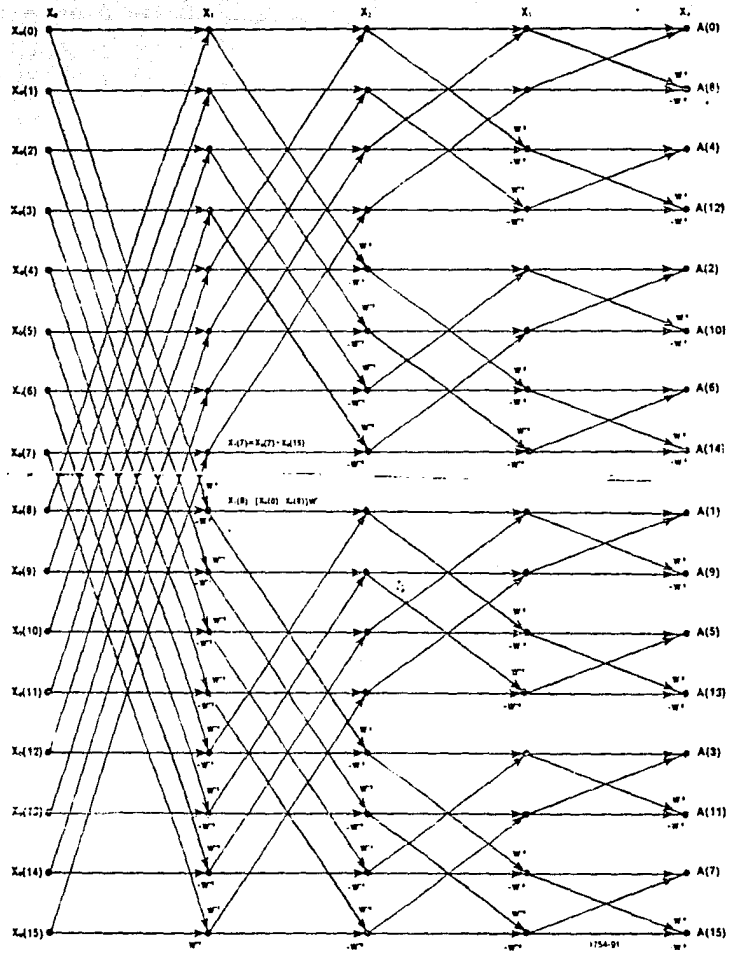


Figura 3-16 Diagrama de flujo para un algoritmo de TRF de decimación en frecuencia de 16 puntos

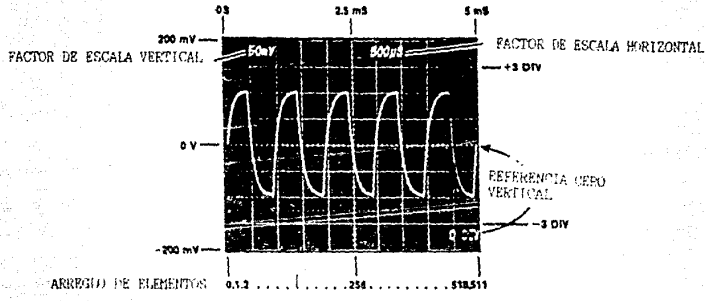
```
100 REM             B:FFT1FT.BAS
101 REM
105 PRINT "RECORD LENGTH MUST BE A POWER OF 2."
110 INPUT "INPUT RECORD LENGTH.";N
115 M=LOG(N)/LOG(2)
120 DIM INPUTRE(N-1),INPUTIM(N-1),BUFFERRE(N-1),BUFFERIM(N-1)
125 DIM TFRE#(N/2-1),TFIM#(N/2-1)
130 REM     ****INPUT DATA FOR XFORM HERE--REAL DATA INTO ARRAY
135 REM     INPUTRE AND IMAGINARY DATA INTO INPUTIM.  IF YOU
140 REM     HAVE REAL TIME-DOMAIN DATA ONLY, SET INPUTIM TO
145 REM     ALL ZEROS.****
150 PRINT "IS THIS AN INVERSE OR FORWARD"
155 PRINT "TRANSFORM? ENTER I OR F."
160 INPUT TYPE$
165 IF TYPE$="I" THEN SIGN=1
170 IF TYPE$="F" THEN SIGN=-1
175 IF SIGN=1 THEN 205
200 REM
201 REM ****REORDER DATA FOR INVERSE FFT****
202 REM
205 FOR I=0 TO N/2
210 BUFFERRE(I)=INPUTRE(I*(N/2-1))
215 BUFFERIM(I)=INPUTIM(I*(N/2-1))
220 NEXT I
225 FOR I=0 TO N/2-2
230 BUFFERRE(I+(N/2+1))=INPUTRE(I)
235 BUFFERIM(I+(N/2+1))=INPUTIM(I)
240 NEXT I
245 FOR I=0 TO N-1
250 INPUTRE(I)=BUFFERRE(I)
255 INPUTIM(I)=BUFFERIM(I)
260 NEXT I
300 REM
305 REM ****GENERATE TWIDDLE FACTORS****
310 REM
315 PI#=3.14159265358979#:P12#=2*PI#
320 FOR P=0 TO N/2-1
325 TFRE#(P)=COS(PI2#*(-P)/N)
330 TFIM#(P)=(SIGN)*SIN(PI2#*(-P)/N)
335 NEXT P
400 REM
405 REM ****COMPUTE FAST FOURIER TRANSFORM****
410 REM
415 FOR L=1 TO M
420 L=0:H=L
425 G=(N/2^L)
430 FOR K=0 TO (N-1) STEP G
435 TFI=0
440 TFIFLAG=(-1)^(L-1)
445 FOR J=0 TO (G-1)
450 TFI=J*2^(L-1)
455 R=K+J+G+J+H:T=J+G+H
460 IF TFIFLAG>0 THEN 490
465 BUFFERRE(R)=INPUTRE(S)+INPUTRE(T)
470 BUFFERIM(R)=INPUTIM(S)+INPUTIM(T)
475 GOTO 500
480 TEMPRE=INPUTRE(S)-INPUTRE(T)
485 TEMPIM=INPUTIM(S)-INPUTIM(T)
490 BUFFERRE(R)=TEMPRE*TFRE(TFI)-TEMPIM*TFIM(TFI)
```

Figura 3-17. Programa en BASIC para computar la TRF y TRF inversa.

```
495 BUFFERIM(R)=TEMPRE*TFIM#(TFI)+TEMPIM*TFRE#(TFI)
500 NEXT J
505 L=L*1:H=INT(L/2)*G*2
510 NEXT K
515 FOR II=0 TO N-1
520 INPUTRE(II)=BUFFEPRE(II)
525 INPUTIM(II)=BUFFERIM(II)
530 NEXT II
535 NEXT I
540 FOR I=0 TO N-1
545 INPUTRE(I)=INPUTRE(I)/N
550 INPUTIM(I)=INPUTIM(I)/N
555 NEXT I
600 REM
605 REM ****BIT REVERSAL ROUTINE TO UNSCRAMBLE FFT RESULTS****
610 REM
615 FOR I=0 TO N-1
620 INDEX%=I
625 IOUT%=0
630 FOR J=1 TO M
635 TEMP%=I AND INDEX%
640 IOUT%=IOUT%*2
645 IOUT%=IOUT%+TEMP%
650 INDEX%=INDEX%\2
655 NEXT J
660 BUFFERRE(I)=INPUTRE(IOUT%)
665 BUFFERIM(I)=INPUTIM(IOUT%)
670 NEXT I
675 IF SIGN=1 THEN 815
700 REM
705 REM ****ORDER FFT OUTPUT FOR NEG. FREQ. AT 0 TO
710 REM N/2-2, DC AT N/2-1, POS. FREQ. AT N/2 TO N-2,
715 REM AND NYQUIST AT N-1.****
720 REM
725 FOR I=0 TO N/2
730 INPUTRE(I+(N/2-1))=BUFFERRE(I)
735 INPUTIM(I+(N/2-1))=BUFFERIM(I)
740 NEXT I
745 FOR I=0 TO N/2-2
750 INPUTRE(I)=BUFFERRE(I+(N/2+1))
755 INPUTIM(I)=BUFFERIM(I+(N/2+1))
760 NEXT I
765 REM
770 REM     ****FFT OUTPUT IS HERE IN ARRAYS INPUTRE FOR REAL
775 REM     PART AND INPUTIM FOR IMAGINARY PART****
780 REM
785 STOP
800 REM
805 REM ****SCALE IFT OUTPUT BY FACTOR OF N***
810 REM
815 FOR I=0 TO N-1
820 INPUTRE(I)=BUFFERRE(I)*N
825 INPUTIM(I)=BUFFERIM(I)*N
830 NEXT I
835 REM
840 REM     ****IFT OUTPUT IS HERE IN ARRAYS INPUTRE FOR REAL
845 REM     PART AND INPUTIM FOR IMAGINARY PART****
850 REM
855 STOP
```

III-6 FORMATOS DE DATOS EN LA TRANSFORMADA RAPIDA DE FOURIER

En el sentido fundamental hay dos tipos de información en el dominio del tiempo, disponibles para cualquier tipo de análisis. Hay información que es directamente medible. Luego hay información teórica generada por fórmulas matemáticas o cualquier forma de expresión.
Para el caso de información medible, como una forma de onda, la información puede ser muestreada por medio de técnicas estándares de medición. Entonces las muestras de la ventana pueden almacenarse en un arreglo de datos para su procesamiento futuro. En el caso del software usado aquí para los ejemplos, la longitud del arreglo de datos de la forma de onda es de 512 elementos. Cada elemento del arreglo corresponde a una muestra de la señal de la ventana.
El elemento cero del arreglo corresponde al tiempo cero en el borde izquierdo de la ventana, como se indica en la figura 3-18, y el elemento número 511 está en el instante 512.
Los elementos restantes, del 1 al 510, corresponde a puntos igualmente espaciados entre los bordes de la ventana. Los valores digitales almacenados en cada elemento del arreglo corresponden a valores muestreados de la forma de onda en puntos de muestra igualmente espaciados en el tiempo. En resumen, el arreglo de formas de ondas corresponde a la tabla de valores que debemos de construir de mediciones de punto a punto de la amplitud de la señal. La forma de onda mostrada en la figura 3-18, correspondería a la tabla de mediciones punto a punto anotada inmediatamente después.
Para el caso de los datos teóricos, los valores computados son simplemente aceptados en el arreglo de datos de la misma manera que la tabla 3-1.
Sin embargo, en el caso de la tabla 3-1, el arreglo es de 32 elementos. Para colocar la misma onda cosenoidal sobre la misma ventana en un arreglo de 512 elementos, necesitamos evaluar muestras cada 0.625 ms (0.32 segundos divididos entre 511 incrementos de tiempo) de 0 a .32 segundos. En cada caso, podemos hacer la TRF de los datos. En la anterior, se efectuó una TRF de 32 puntos, y hacemos después una TRF de 512 puntos. La diferencia importante, obviamente, es que las 512 muestras sobre el mismo intervalo de tiempo provee mucha mayor resolución en tiempo y resulta un calculo de más puntos en el dominio de la frecuencia.
La mayor parte de los datos en el dominio del tiempo, medidos y teóricos son datos con valores reales. Esto es, los datos son reales, no necesariamente en el sentido físico, sino reales en el sentido que no son datos complejos. No tienen parte imaginaria o al menos la parte imaginaria no es considerada. Así, solamente necesitamos almacenar un arreglo de una forma de onda con datos reales en el dominio del tiempo. Sin embargo, requerimos dos arreglos de forma de onda adicional para almacenar los datos del dominio de la frecuencia. Uno de estos arreglos es para la magnitud y la otra es para la fase, dependiendo si los resultados se presentan en forma polar o rectangular.



Time = n Δt Δt = 500 µS/51.2	Formato del arreglo Arreglo de elementos	Valores almacenados
0.0 µS	0	-5.299E-3
9.766 µS	1	4.599E-3
19.53 µS	2	1.422E-2
29.3 µS	3	2.293E-2
39.06 µS	4	3.010E-2
.	.	.
.	.	.
.	.	.
2.481 mS	254	1.927E-3
2.49 mS	255	9.855E-3
2.5 mS	256	-8.125E-4
2.51 mS	257	-1.052E-2
2.52 mS	258	-1.879E-2
.	.	.
.	.	.
.	.	.
4.951 mS	507	-5.434E-2
4.961 mS	508	-3.938E-2
4.971 mS	509	-2.608E-2
4.981 mS	510	-1.397E-2
4.99 mS	511	-1.850E-3

Figura 3-18. Formato de arreglo para ondas en el dominio del tiempo.

Adicionalmente, necesitamos estar preparados para los requerimientos de tamaño para los arreglos.
Si el software provee los resultados de la TRF sólo para frecuencias positivas, cada arreglo para los resultados necesita solamente ser mayor en un elemento que la mitad de la longitud del arreglo en el dominio del tiempo. Por ejemplo, un arreglo en el dominio del tiempo de 512 elementos requiere dos arreglos en el dominio de la frecuencia de 257 cada uno para contener los coeficientes de Fourier desde CD (directa) hasta la frecuencia de Nyquist (F =mitad de la frecuencia de muestreo). O, si el algoritmo de TRF provee resultados para el dominio de frecuencia positivo y negativo. Para este caso, la CD estará en el centro de uno de los arreglos de elementos (ya sea, 255 o 256, dependiendo de si la frecuencia de Nyquist es desplegada en el extremo negativo o positivo de la frecuencia.

III-7 LOS RESULTADOS DE LA TRF EN LA FORMA RECTANGULAR

Regresando a las matematicas de la TDF

$$X_d(k) = \frac{1}{N}\sum_{n=0}^{N-1} x(n)e^{-j2\pi kn/N}$$

$$X_d(k) = \frac{1}{N}\sum_{n=0}^{N-1} x(n)\left(\cos\frac{2\pi kn}{N} - j\sin\frac{2\pi kn}{N}\right)$$

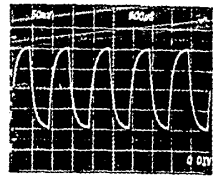
Es aparente que los resultados vienen en dos partes- una parte real y una parte imaginaria precedida del operador j. Esto es la forma rectangular de resultados provistos por la mayoría de las implementaciones de la TRF

Como un ejemplo específico, veamos la transformación de la forma de onda de la figura 3-18 al dominio de la frecuencia. La implementación de la TRF usada para este ejemplo es el DPO TEK BASIC, un paquete para procesamiento de señales desarrollado por Tektronix Inc..

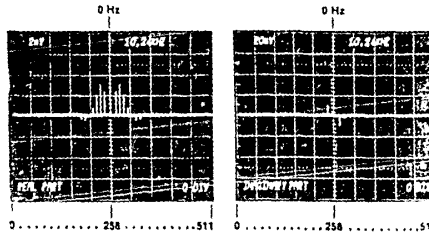
Procediendo en el ejemplo, deducimos que la señal a transformar se almacena en el arreglo de 512 elementos designados como arreglo A. Mostramos un despliegue de estos datos en la figura 3-19a. Este arreglo de datos, arreglo A, se transforma al dominio de la frecuencia por :

FFT A,B,C

Donde B y C son también arreglos de 512 elementos para almacenar los resultados del dominio de la frecuencia, los cuales en este caso, abarcan las frecuencias negativas y positivas. La parte real del dominio de la frecuencia se almacena en B (en la figura 3-19b); la parte imaginaria es almacenada en C (en la figura 3-19c). Notese que estos resultados recuerdan el espectro de una señal periódica.



a) Forma de onda a transformar



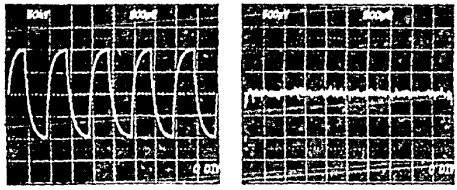
b) Dominio de la frecuencia de parte real

c) Dominio de la frecuencia parte imaginaria

Figura 3-19. La TRF de una señal en el dominio del tiempo resulta en dos arreglos que representan el dominio de la frecuencia compleja en forma rectangular.

Un punto de vista periódico surge de la figura 3-19a.
El arreglo de los coeficientes de Fourier en el dominio de la frecuencia es tal que los 0 Hz (componente de directa) ocurren en el elemento 256.
Esto corresponde al centro de cada despliegue en la figura 3-19b y c.
La porción positiva del dominio de la frecuencia se extiende a la derecha del centro (del elemento 257 al 511), y el dominio negativo se extiende sobre la parte izquierda del despliegue (elemento 0 al 255). Dependiendo del paquete de software usado, el elemento cero (la izquierda del despliegue) corresponde a la frecuencia más alta que puede definirse por el intervalo de muestreo en el dominio del tiempo y se refiere como la frecuencia de Nyquist.
Otros paquetes de software formatean los resultados de manera ligeramente diferente. Por ejemplo, la frecuencia de Nyquist puede aparecer como el último elemento en el arreglo (correspondiente al borde derecho de los datos desplegados).

Esto es usualmente el caso cuando la implementacion de la TRF presenta sólo la frecuencia positiva. También para un despliegue de frecuencias positivas solamente, el borde izquierdo del despliegue coresponderá a CD ó 0 Hertz.
Cualquier formato que usemos para una implementación de la TRF, el mismo formato lo requerimos para que los datos de la frecuencia sean transformados por la transformada inversa de Fourier al dominio del tiempo. La transformada inversa de Fourier usa esencialmente el mismo algoritmo de la TRF con algunos cambios menores a las rutinas de entrada y salida.
Así, en el caso del ejemplo de la figura 3-19, los arreglos del dominio de la frecuencia (B y C) podrían invertirse a otro arreglo de datos (D). El resultado de esta inversión (usando un comando similar IFT B,C,D) se muestra en la figura 3-20a.
La diferencia entre el resultado a partir de IFT y la onda original (figura 3-19a) se muestra en la figura 3-20(b).
Notese que esta diferencia es pequeña- efectivamente cero comparada con las magnitudes de ondas- y puede atribuirse a errores por redondeo aritmético.



a) IFT de la fig. 3-19a y b.

b) Diferencias entre la señal original y la invertida.

Figura 3-20. La TRF inversa del dominio de la frecuencia en forma rectangular regresa la forma de onda original.

III-8 LA TRANSFORMADA RAPIDA DE FOURIER EN FORMA POLAR

La forma rectangular es el resultado más directo para la TRF y es también más apropiado para el procesamiento.
Convertir los resultados de la TRF de coordenadas rectangulares a forma polar es una simple operación.
La magnitud es obtenida de la parte real e imaginaria, tomando la raíz cuadrada de la suma de los cuadrados (magnitude $\sqrt{Re^2 + Im^2}$). La fase es el arcotangente ($\tan^{-1}$) de la parte imaginaria entre la parte real (fase=$\tan^{-1}$ Im/Re).

Modulo 2ff. Por conveniencia, algunas implementaciones de la TRF contienen un argumento polar o interruptor. Este argumento extra permite especificar los resultados en forma polar.
En otras palabras, se provee una rutina de conversión a forma polar.
para el caso del interruptor o la opción de la forma polar, el algoritmo de la TRF no es cambiado al añadir el argumento.
Los resultados intermedios están todavía en forma rectangular. El interruptor polar solamente causa un paso adicional para convertir los resultados a forma polar.

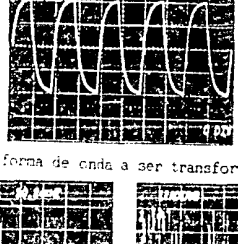
Adicionalmente, el formato de los arreglos de resultados permanece sin cambios, pero un arreglo contiene la magnitud y otro la fase. El arreglo de magnitud es el resultado de evaluar la raíz cuadrada de la suma de los cuadrados de las

partes real e imaginaria y el arreglo de fase es el resultado de evaluar el arcotangente de la parte imaginaria entre la parte real.
En todos los casos, requerimos una rutina de arcotangente.
También debemos notar que la mayoría de las rutinas de arcotangente regresan los resultados por lo general en el rango de 2π radianes, de $-\pi$ a π. Así, la fase resultante de usar conversión polar estará en ese rango y es conocida como fase módulo 2π.
Un ejemplo de resultados polares TRF se muestra en la figura 3-21. La señal en el dominio del tiempo en la figura 3-21a es la misma señal usada para el ejemplo rectangular.
Los resultados de la forma polar de la señal transformada con la TRF se muestran en la figura 3-21b y c.
La porción de la magnitud, está claro, es lo que esperabamos para una señal cuadrada filtrada con un filtro tipo paso-bajo. El espectro de magnitud también recuerda el espectro de línea.
La fase no puede existir, para un componente de frecuencia que no existe. Si vemos la figura 3-21a como una forma de onda periódica e interpretamos los resultados de la magnitud de la TRF en la figura 3-21b como el espectro de línea, la figura 3-21c parece ser contradictoria.
Parece haber fase en la figura 3-21c donde la magnitud en la figura 3-21b aparece como cero, eso es, en frecuencias entre las líneas espectrales.

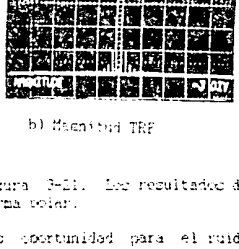
La razón de ésto es que todos estos espectros tienen componentes de bajo nivel de ruido residiendo entre las armónicas.
Para interpretar los resultados desde un punto de vista de continuidad y periódico, los componentes de bajo nivel de ruido son suficientemente bajos para considerarse cero.
Sin embargo, desde el enfoque computacional, estos componentes de bajo nivel todavía existen y son significativos para calcular la fase. Por ejemplo un valor de 10^{-6}, podría ser para propositos de interpretación, considerada cero comparada con el valor 1. Pero, para el proposito de procesamiento en computadoras, arcotangente de 1/1 y arcotangente de $10^{-6}/10^{-6}$ es lo mismo.
Aún los componentes más pequeños pueden contribuir significativamente a la fase. Esto es deseable cuando estamos buscando información contenida en componentes de bajo nivel, pero para el caso del ejemplo puede conducir a confusiones.

Como se presentan las componentes de bajo nivel? En realidad, la figura 3-21 no muestra una forma de onda ideal, continua y periódica.
La forma de onda usada para este ejemplo, fué una forma de onda verdadera, resultante de un proceso físico. Estuvo sujeta a distorsión, ruido, y todo lo relacionado con una señal física producto de un fenómeno real.

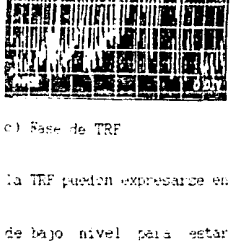
Así, algunas componentes de bajo nivel de ruido estaban realmente presentes como ruido de banda amplia. Añadiendo a ésto, la forma de onda fue adquirida por medio de una ventana y muestreada, las muestras fueron digitalizadas.



a) La forma de onda a ser transformada



b) Magnitud TRF



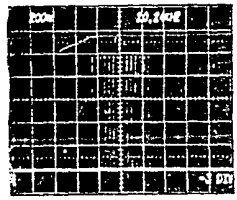
c) Fase de TRF

Figura 3-11. Los resultados de la TRF pueden expresarse en forma polar.

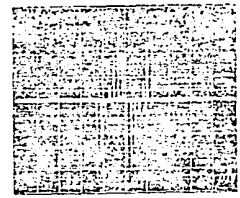
Más oportunidad para el ruido de bajo nivel para estar presente. Luego las muestras digitalizadas fueron transformadas por un algoritmo de TRF ejecutado por un sistema de computo con limitaciones tales como la velocidad de redondear algunos números, más ruido, de hecho es un tributo a la tecnología moderna que el ruido en estos componentes sea de bajo nivel.
Como podemos ver, muchas veces separan la realidad del análisis digital y las mediciones de la teoría pura.
Por ahora aceptemos el hecho, lo que aparezca en los resultados de la TRF, aunque inesperado, es probablemente válido o al menos explicable.
La señal cuadrada filtrada del figura 3-23a está denotando un punto de vista continuo y periódico.
Sin embargo, para tener un punto de vista periódico para interpretar los resultados de fase, debemos considerar la

fase valida solamente en las frecuencias de cada linea espectral en el despliegue de magnitud. Con eso en mente, podemos escoger los componentes significativos de frecuencia del espectro de magnitud.
Entonces, pueden colocarse en un arreglo y darle el valor de 1, mientras que todos los elementos del otro arreglo son cero. El arreglo resultante se muestra en la figura 3-22a. Entonces, de la multiplicación de elemento por elemento del arreglo de la figura 3-22a con el arreglo mostrado en la figura 3-21c, podemos obtener un espectro de linea para la fase (la figura 3-22b).
Este espectro de línea para el espectro de fase y magnitud en la figura 3-21b es lo que desamos observar si queremos interpretar el dominio de la frecuencia de la señal cuadrada filtrada desde el punto periódico, punto de vista relacionado a las series Fourier.
Pero, un punto de vista no-periódico puede ser más apropiado para algunas situaciones de análisis. Por ejemplo la figura 3-23a. El pulso, ciertamente aparece desde un punto de vista no-periódico.
Probablemente quisieramos interpretar su dominio de la frecuencia en términos de lo que esperariamos de la integral de Fourier.
El dominio de la frecuencia del pulso de banda limitada en la figura 3-23a se muestra en 3-23b y c. En particular, notese el despliegue de fase en la figura 3-23c. Está ordenada, lo que no sucede en la figura 3-21c, y parece coincidir con lo que esperabamos desde el punto de vista no-periódico. De hecho, comparemos el dominio de la frecuencia de la figura 3-23 con el dominio de la frecuencia teórica de la integral de Fourier en la figura 3-6c. Notese como casi coinciden.
De hecho, la diferencia aparente de los resultados de la figura 3-23, estriba en el pico de amplitud del espectro. Donde la integral de Fourier lo pone como VTo=25 x 200 x 10^{-6} = 5 x 10^{-3}, la TRF lo pone como 1. La razón es la periodicidad supuesta de la TRF. Con la periodicidad supuesta, es como si el pulso cuadrado fuera un tren de pulsos con un periodo igual a la longitud de la ventana. A reserva de discutir más sobre este punto, causa que la magnitud sea escalada al valor medio del pulso sobre la longitud de la ventana. Este valor medio es el término de las series de Fourier y puede evaluarse por la figura 3-23a como VTo/T = 5 x 10^{-3}/ (10x500x 10^{-6})= 1, donde T es la longitud de la ventana de adquisición (500 microsegundos por división para 10 divisiones). Tal que los resultados de la figura 3-23 son predecibles.
El ejemplo de la figura 3-23 comprende el enfoque no-periódico para interpretar los resultados de la TDF y TRF. Y es lógico ya que la TDF se deriva de la integral de Fourier. Pero recordemos que la integral de Fourier es también derivable de las series de Fourier. De tal manera que el punto de vista o enfoque periódico a la TDF y TRF es válida siempre y cuando tomemos en cuenta las modificaciones que ocurren a través de las transformaciones.

Fase continua. Hasta este punto, la fase ha sido ilustrada en términos del módulo 2π de fase; esto es, la fase ha sido restringida al rango $\pm\pi$, tal como se muestra en la figura 3-23c. Esto es suficiente para la mayoría de las interpretaciones. Pero la fase puede en teoría y práctica exceder el rango $\pm\pi$. Esto se muestra en la figura 3-24.



a) Componentes significativos puestos en valor 1

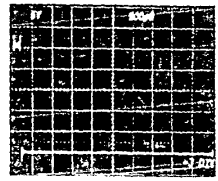


b) El espectro de línea para la fase es el producto de las figs. 3-22b y 3-21c.

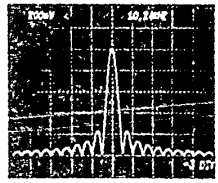
Figura 3-22. Interpretando la fase del enfoque periódico requiere sólo ver las componentes de frecuencia significativas.

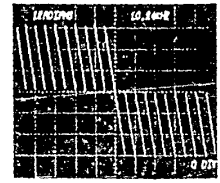
En términos de analizar las formas de onda compuestas de senoidales, la fase de cada componente de frecuencia se determina por la opción del tiempo cero. En la figura 3-25, por ejemplo, mostramos un pulso arreglado en el tiempo cero tal que el pulso es una función par del tiempo; por consiguiente, está hecho solamente de términos cosenos.
Un punto marca un lugar de referencia fijado al pulso. La misma referencia está marcada en dos componentes senoidales en la figura 3-25a.
Ahora como el pulso está recorrido en el tiempo (ver la fig. 3-25b), cada componente senoidal conlleva el mismo defasamiento. La relación de tiempo y amplitud de cada componente con otras no cambia, ya que todas las componentes se deben añadir de la misma manera para formar el pulso. Un mismo tiempo de retardo representa diferentes ángulos de fase para señales de diferentes frecuencias.
Por ejemplo una onda de 1000 hz que se desplace 0.25 ms se habrá defasado $(360 * 0.25 * 10^{-3}) / (1/1000) = 90$ y otra onda de 3000 hz que se desplace el mismo tiempo de 0.25 ms se habrá defasado $(360 * 0.25 * 10^{-3}) / (1/3000) = 270$.
Como se indica por las dos componentes en la figura 3-25b,

un componente de más alta frecuencia, a causa de su periodo más corto, presenta mayor defasamiento que uno a frecuencia más baja.
El pulso de la figura 3-22 es un ejemplo para ilustrar la diferencia entre el despliegue de modulo 2π y el de fase continua.
Este pulso se muestra otra vez en la figura 3-26a con su posición ligeramente defasada en el tiempo. Como el tiempo cero está al borde izquierdo del despliegue, el defasamiento en la figura 3-26a debería reflejarse en el dominio de la frecuencia como un incremento en la fase de cada uno de los componentes.



[illegible] de modo limitado.





[illegible]

Figura 3-22. Un caso donde el enfoque periódico es la opción natural y coincide con los resultados de la TRF.

La figura 3-26b muestra la magnitud del dominio de la frecuencia y la fase del módulo 2π para el pulso de la figura 3-26a. Este despliegue de fase es ciertamente diferente de la figura 3-23, pero notese que está restringido al rango de $\pm\pi$ radianes.
La figura 3-26c muestra el mismo dominio de la frecuencia como la figura 3-26b. Sin embargo, notese que la fase ahora excede el rango $\pm\pi$. Aquí, la fase se muestra en un formato

continuo, no con interrupcion ni repetido cada $\pm\pi$. Aùn mas interesante es el hecho que en la figura 3-23, a pesar de que fué computado con el módulo 2π, no pareceria diferente como con fase continua. Esto es a causa de la localización del pulso relativa al tiempo cero en la figura 3-23, el tiempo de defasamiento, o retardo, no es lo suficientemente grande para causar que la fase exceda a + . Sin embargo, con el tiempo de retardo incrementado en la figura 3-26a, la fase no excede el rango $\pm\pi$.

Algunas implementaciones de TRF proveen resultados polares en módulo 2π y otras proveen en fase continua. Aùn hay otros paquetes que contienen las dos opciones para seleccionar resultados polares en las dos formas.

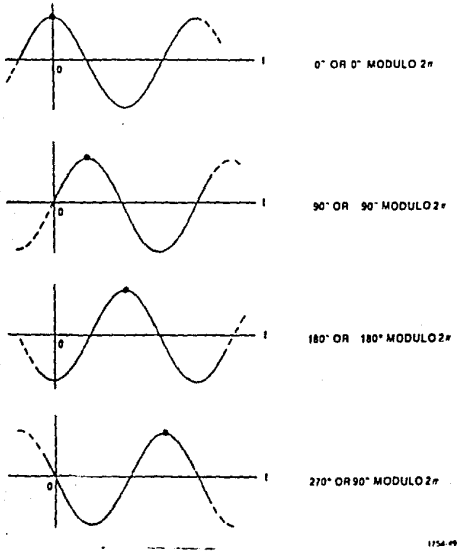
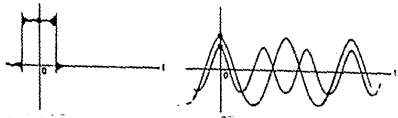


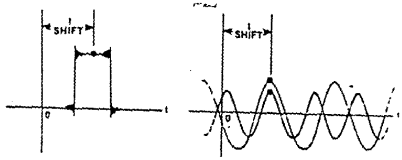
Figura 3-24. Especifica la fase depende de las referencias. Si la referencia está fijada a un punto especifico de la señal defasada, la fase relacionada al tiempo cero puede exceder el rango $\pm\pi$.

Es importante tomar en cuenta que clase de fase esta siendo usada por la implementación, para poder evaluar los resultados de acuerdo a lo anterior.

Es también muy importante darse cuenta que la fase de módulo 2π y fase continua son nada más que dos medios de presentar la misma información. La fase continua presenta los efectos del retardo en tiempo (adelantado o retardado) directamente alrededor de -360° de fase se representa como 360° de fase. Por otro lado, la fase del módulo 2 visualiza el retardo en tiempo en el sentido de que una senoidal con 0 de fase no aparece diferente de una retardada 360°. Así, las fase de 360° 720°, están representadas como 0° en el módulo 2π. Esta misma idea se aplica a todas las fases en exceso de $\pm$ radianes.



a) El pulso cuadrado mostrado con dos componentes arbitrarios arreglados para la misma relación de fase. La fase es 0°.



b) El mismo pulso cuadrado y los componentes arbitrarios recorridos en el tiempo. Un componente tiene fase de -270°; y el otro -400°.

Figura 3-25. El mismo retardo en el tiempo corresponde a un retardo en fase distinto para diferentes frecuencias.

Se pueden representar en el rango de $\pm\pi$ sin cambiar la apariencia de la senoide alrededor del tiempo cero.
Como una nota final, hay otra cosa interesante que debemos señalar en las figuras 3-24 y 3-27. Esto no es algo nuevo, sino más bien, algo para revisar y reforzar lo visto anteriormente. Veamos ahora los pulsos en las figuras 3-24a y 3-19a. La unica diferencia entre ellos es su localización en el tiempo.
Ahora veamos las magnitudes y fase del dominio de la

frecuencia en las figuras 3-5b y c y 3-27b. Notese que no hay ninguna diferencia en los despliegues de las magnitudes. Los despliegues de fase, sin embargo, son diferentes!. Esto señala otra vez a la propiedad de la integral de Fourier que el corrimiento en tiempo sólo afecta al ángulo de fase, en el dominio de la frecuencia.

IV- APLICACIONES DE LA TRF

Además de las comunicaciones, la astronomía, la física, la química, la estadística, la biomédicina, la electrónica, la mecánica y otros campos tiene áreas de estudio que pueden obtener beneficio de la teoría de Fourier, específicamente de la transformada rápida de Fourier o TRF (FFT).

IV-1 ANALISIS DE DISTORSION

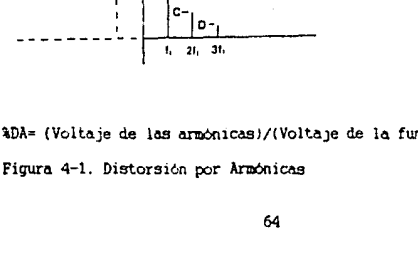
La distorsión ocurre cuando la onda de salida de un circuito tiene diferente forma que la onda de entrada.
A veces se busca intencionalmente está distorsión, como en el caso de un filtro al que se alimenta una señal con varias componentes, algunas de las cuales no aparecen a la salida.

La distorsión es :

Por frecuencia.- Cuando la respuesta a la frecuencia del dispositivo no es plana.

Por retardo.- Cuando las diversas componentes de una onda tardan diferentes tiempos en alcanzar la salida del dispositivo.
Armónica.- Cuando se alimenta una sola señal senoidal pura al dispositivo y éste la deforma.
Por intermodulación.- Cuando el dispositivo recibe una señal de cualquier forma no senoidal y la deforma.

Si la red ha causado distorsión por armónicas, La magnitud en el dominio de la frecuencia tendrá componentes que están relacionados con las armónicas del tono de prueba. Estas armónicas son distorsiones; son frecuencias en la salida que no están presentes en la entrada. El porcentaje de distorsión por armónicas puede ser determinado de la manera mostrada en la figura 4-1.



%DA= (Voltaje de las armónicas)/(Voltaje de la fundamental) x 100

Figura 4-1. Distorsión por Armónicas

Si varios tonos de prueba se alimentan a una red alineal dentro de ésta, las señales tienden a modularse entre ellas y producir la suma y diferencia de frecuencias. A la producción de estas sumas y diferencias de frecuencia (bandas laterales) se le llama distorsión por intermodulación. La visualización de la distorsión por intermodulación magnitud en el dominio de la frecuencia a la salida de una red se muestra en la figura 4-2.

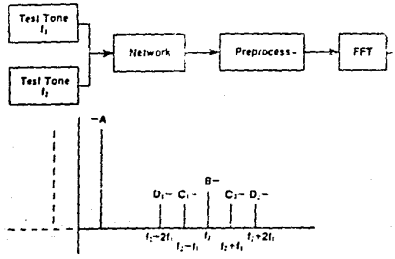


Figura 4-2 Visualización de distorsión por intermodulación.

Naturalmente, cualquier análisis de armónicas hecho por medio de TRF debe ser hecho con las debidas consideraciones por escurrimiento. Puede ser necesario preprocesar los datos para asegurar un número entero de ciclos, o se puede crear una ventana con los datos para reducir el escurrimiento en el dominio de la frecuencia.

IV-2 ANALISIS MECANICO Y DE VIBRACIONES

Los motores, bombas compresores, roladores, y otras maquinarias rotatorias vibran durante la operación. Y los resultados de un balero o rodamiento gastado pueden causar pérdidas en equipo valioso y más aún el tiempo muerto de la producción.
Una solución mejor es monitorear las vibraciones de una máina a través de transductores de desplazamiento, velocidad, y aceleración. Se pueden tomar lecturas periódicas de los niveles de vibración, de tal manera que las tendencias se puedan vigilar antes de que alcancen niveles peligrosos.

Todavía, el nivel de vibración por sí sólo no puede dar información suficiente. Una componente puede estar defectuosa y fallar o puede haber fallado sin afectar seriamente el nivel de vibración.

Aún esos defectos o vibraciones por falla están allí y con frecuencia se hacen visibles cuando la forma de onda de la vibración total se transforma al dominio de la frecuencia desbalanceo, desalineación, e inestabilidades de baleros agregan sus componentes al espectro
Un espectro de vibración se conoce como inspección mecánica, y la determinación de la información de ella se le conoce como análisis de inspección. Generalmente, las inspecciones mecánicas se obtienen cuando una máquina esta en operación, y se obtiene una inspección estándar de una máquina en buenas condiciones como referencia. Entonces la inspección mecánica tomada en fechas posteriores se compara con el estándar y se reportan los cambios significativos en los diferentes componentes

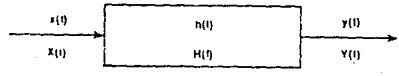
IV-3 ESTIMACION DE RESPUESTA EN FRECUENCIA (FUNCION DE TRANSFERENCIA)

Una gran parte del trabajo científico e ingeniería se enfoca en los sistemas. Estos sistemas pueden ser sencillos o complicados. Pueden ser mecánicos, eléctricos, o biológicos. Pero de cualquier manera que se presenten, las incógnitas principales son : Que hace?, Como reacciona a los estímulos? Que pasaría si ?
Estas incógnitas se pueden establecer a menudo por pruebas reales a los equipos. Pero someter el equipo a pruebas bajo ciertas condiciones puede ser impráctico, es factible que esté involucrado un elemento de peligro al efectuar estas pruebas.
Entonces se tendrían que predecir los resultados de las pruebas o al menos lograr algunas predicciones antes de efectuar pruebas. Lo se necesita es algún medio de caracterizar el sistema. Se necesita saber como va a responder a cada componente de frecuencia de una señal de entrada cualquiera.
La manera en que un sistema reacciona es llamada respuesta en frecuencia del sistema. Esta respuesta en frecuencia del sistema se expresa como una traza de amplitud vs frecuencia y fase vs frecuencia, o donde es posible, como un conjunto de ecuaciones describiendo estas gráficas.
Para un sistema lineal e invariante en el tiempo la respuesta en frecuencia caracteriza completamente al sistema. Si el sistema es no-lineal o variante en el tiempo, una traza de respuesta en frecuencia puede caracterizar para algunas condiciones de operación específicas.
Hay una variedad de métodos para obtener la respuesta en frecuencia de un sistema. La forma más sencilla es alimentar una señal senoidal a varias frecuencias, una a la vez, dentro del sistema. Entonces se grafica la amplitud y la fase en la salida. Algunos medios comunes de obtención de gráficas de respuesta a la frecuencia son con osciladores de barrido en frecuencia y hay otros esquemas de prueba como medio para obtener las trazas de respuesta en frecuencia. Un analizador de espectros con un generador de seguimiento es otro modo. Típicamente, sin embargo, la instrumentación

estándar está limitada para dar solamente un trazo de amplitud contra frecuencia. También , los osciladores por barrido de frecuencia y generadores de seguimiento tienen rangos limitados de frecuencia. Estas limitaciones pueden resultar en una caracterización incompleta del sistema, solamente se obtiene una traza de amplitud vs. frecuencia y solamente para una banda limitada de frecuencia. Dependiendo de las necesidades de análisis puede ser obligatorio más trabajo de análisis.
Otro punto a considerar es que las senoides no son señales de prueba prácticas para algunas situaciones. Esto es particularmente cierto para muchos sistemas mecánicos y geológicos. Considere, por ejemplo, la exploración petrolera, Como aplicar una señal senoidal a la superficie de la tierra?, En cambio, Las inspecciones geológicas se conducen normalmente explotando pequeñas cargas bajo la superficie de la tierra.Las respuestas son recogidas por micrófonos puestos en lugares estratégicos, y despues se usan para caracterizar la estructura de la superficie.
Se le conoce como impulso a la alta concentración de energia, tal como una carga de prueba geológica. La respuesta impulso es la forma en que un sistema reacciona a un impulso. La respuesta impulso es la contraparte en el dominio del tiempo de la respuesta en frecuencia.
Si se aplica transformada rápida de Fourier a una respuesta al impulso, se obtendrá la respuesta en frecuencia del sistema, que se conoce como la función de transferencia del sistema
En un sentido teórico, un impulso es una función de amplitud infinita, ancho igual a cero, y área unitaria. Su dominio de la frecuencia tiene una amplitud unitaria en todas las frecuencias.
En términos prácticos, sin embargo, un impulso tiene una amplitud finita y un ancho de pulso diferente de cero. Aunque diferente de cero, el ancho debe ser mucho menor que la respuesta en el tiempo esperada del sistema.
Hay también casos donde un impulso no es una señal de prueba apropiada. Por ejemplo, en la prueba de redes electrónicas, es a menudo más fácil aplicar una función escalón a la entrada. Entonces el voltaje a la salida se refiere como respuesta escalon. Esto está relacionado a la respuesta impulso porque la derivada de un escalón es un impulso. Y para sistemas, lineales e invariantes en el tiempo, la derivada de la respuesta escalón es la respuesta impulso. Y, de nuevo, la respuesta en frecuencia es la FFT de la respuesta impulso.

(Notese: La mayoria de los paquetes de procesamiento de señales digitales tiene rutinas de diferenciación que permiten fácilmente calcular la derivada de una forma de onda almacenada.)
Hay algunos casos, también, donde no son apropiadas ninguna de las señales estándares, senoides,frecuencias de barrido, impulsos, y escalones. Una red de comunicaciónes puede ser un ejemplo de esto.

Supongamos que la red está instalada y en uso. No se desea interrumpir para prueba de respuesta en frecuencia. Así que las señales de operación deben ser las señales de prueba. Su relación a la respuesta en frecuencia se muestra en la figura 4-3, donde x(t) es la señal de entrada, h(t) es la respuesta impulso , y y(t) es la señal de salida.
La relación más simple entre x(t) y y(t) se dá cuando estos



$$y(t)=\int_{-\infty}^{\infty} h(\tau)x(t-\tau)d\tau$$

$$Y(f)=X(f)H(f)$$

Figura 4-3. Implementación por Hardware de la correlación.

términos se encuentran en el dominio de la frecuencia. Entonces la salida en el dominio de la frecuencia es Y(f), es igual al producto de la respuesta en frecuencia, H(f), y la entrada, X(f). La respuesta en frecuencia puede ser determinada adquiriendo x(t) y y(t), transformandolas por medio de la FFT al dominio de la frecuencia.

H(f) se encuentra dividiendo Y(f) entre X(f). Sin embargo, se deben observar precauciones en esta operación, ya que existe el peligro de un división entre cero al calcular Y(f)/X(f).
Se puede evitar una situación de división por cero, colocando algoritmos de verificación en los puntos anteriores al calculo.

IV-4 CONVOLUCION

Una vez obtenida la respuesta a impulso o la respuesta en frecuencia de un sistema, se tiene caracterizado el sistema. Desafortunadamente, sin embargo, usualmente es difícil determinar como va a reaccionar un sistema a una entrada solamente observando su respuesta a impulso o la respuesta en frecuencia.

Para predecir la forma de onda de la señal de salida del sistema para una señal determinada, necesita resolver la integral de convolución. Esto se muestra en la figura 4-3. Establece que la señal de salida del sistema y(t), es la convolución de la respuesta impulso, h(t), y la señal de entrada, x(t).

El símbolo t en la integral de la figura 4-3 es solamente una variable auxiliar que facilita el corrimiento en el tiempo en la operación de convolución.

La integral, como se muestra en la figura 4-3, se puede evaluar de una manera directa en el dominio del tiempo usando técnicas digitales. Esto, sin embargo, puede consumir mucho tiempo; es más rápido y fácil efectuar la convolución aprovechando que la convolución en el dominio del tiempo corresponde a la multiplicación en el dominio de la frecuencia. El proceso es simplemente transformar la señal de entrada al dominio de la frecuencia por medio de FFT y obtener X(f). Entonces la respuesta a impulso es transformada al dominio de la frecuencia para obtener la respuesta en frecuencia, H(f). El producto de estos dos conjuntos de datos complejos se usa para obtener Y(f), que es la función del dominio de la frecuencia para la salida del sistema causado por la entrada, x(t).

La función en el dominio del tiempo para la salida de este sistema se obtiene simplemente efectuando la transformación inversa de Fourier de Y(f) para obtener y(t). Conociendo la respuesta en frecuencia del sistema, la técnica de convolución descrita anteriormente se puede usar para predecir la forma de onda del sistema para cualquier entrada.
Hay algunos casos, sin embargo, cuando se conoce la forma de onda de salida y se desea conocer cual fué la entrada que lo causó. Esto se puede evaluar siempre que se haya definido H(f) del sistema.
La forma de onda de la entrada se determina evaluando

$$X(f) = Y(f)/H(f) \qquad (4\text{-}1)$$

Para X(f), y después obteniendo la transformada inversa de X(f) para obtener x(t).
Esta operación es conocida como deconvolución y hay necesidad de tener precaución al usarla. Específicamente la división compleja del arrreglo de datos Y(f) y el arreglo de datos H(f) que puede producir una división inválida en algunos puntos. También, la deconvolución es sensible al ruido. Las componentes del ruido en los datos que son sometidos a deconvolución podrían presentar gran modificación de los resultados. Sin embargo es posible obtener mejoras razonables aplicando filtrado digital.
El filtrado digital es, en el sentido más sencillo, una forma de generar una función de filtrado paso-bajo por medio de programación. La función generalmente se construye como una respuesta en frecuencia. Luego la forma de onda que será filtrada se transforma al dominio de la frecuencia y se multiplica por la respuesta del filtro.
En el dominio del tiempo esto corresponde a convolucionar la respuesta impulso del filtro con la forma de onda.

IV-5 CORRELACION

La definición matemática de correlación es :

$$r(\tau) = \lim_{T \to \infty} \frac{1}{2T} \int_{-T}^{T} x(t)y(t+\tau)dt$$

donde r(ℓ) = Función de correlación formada por la suma de los productos de x(t) y y(t).

(ℓ) = Tiempo de retardo entre x(t) y y(t)

La correlación es entendida como la comparación de componentes o la prueba de similitud entre formas de ondas. El circuito equivalente a esta definición, en la figura 4-4, hace esta operación más sencilla de visualizar.
En términos de procesamiento de digital de señales, la correlación se simplifica bastante usando FFT. Las dos formas de onda a correlacionar, x(t) y y(t) se transforman al dominio de la frecuencia. Conforme a lo anterior, un término es conjugado y el producto complejo se forma para dar R(f)= X(f)* Y(f) =(X(f) * Y(f)). Aquí el asterisco se usa para denotar conjugación. El paso final es la transformación inversa R(f) al dominio del tiempo para obtener r( ).
Se pueden efectuar, dependiendo de las formas de onda que sean usadas, dos tipos de correlación. Si las dos formas de onda son iguales x(t)=y(t) se dice ser una autocorrelación. Si las formas de onda son diferentes, x(t)≠y(t), su correlación se conoce como correlación cruzada.

Autocorrelación. Es útil hacer notar que la función de autocorrelación de una señal periódica es periódica.
También la función de autocorrelación de una señal no-periódica es no periódica. Estas dos cosas se demuestran en la figura 4-5a y b, donde una onda senoidal está autocorrelacionada y el ruido aleatorio está correlacionado. Las funciones de autocorrelación en la figura 4-5 se arreglan de tal manera que el eje vertical queda a la mitad de la gráfica. El tiempo positivo está a la derecha y el tiempo negativo está a la izquierda.
En el caso de la función de autocorrelación de una onda senoidal, la máxima correlación ocurre en el tiempo cero (centro de la gráfica). Esto es donde la onda senoidal se superpone perfectamente. La máxima correlación ocurre para la onda senoidal en cualquier intervalo igual al periodo de la onda senoidal.

En la figura 4-6, sin embargo, la función de autocorrelación aparece como una ventana triangular, por consiguiente, reduciendo los máximos excepto en cero.
Esta ventana triangular ocurre a causa de arreglos de ceros extra que se añaden a los arreglos de formas de onda antes

de la correlacion (ver figura 4-7). Estos ceros agregados preveen errores de correlación ciclica, pero también hacen aparecer a la onda senoidal como pulsada en lugar de continua. Este efecto pulsado causa una envolvente triangular para la función de correlación.

Notese en la figura 4-5b que la función de correlación para el ruido es grande en el tiempo cero y muy pequeña en los demás tiempos. Un acoplamiento exacto (correlación perfecta) se obtiene solamente cuando el ruido se superpone exactamente sobre si mismo (tiempo cero).

Para otros intervalos de tiempo, hay poco acoplamiento o ninguno y el ruido se dice estar no-correlacionado.
La figura 4-5c muestra la función de autocorrelación para lo que parece ser ruido aleatorio. Sin embargo, de la periodicidad de la función de autocorrelación, es obvio que la señal periódica está presente en el ruido.

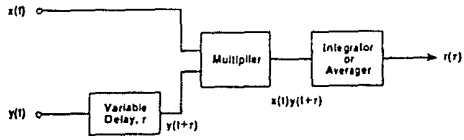


Figura 4-4. Implementación por Hardware de la correlación.

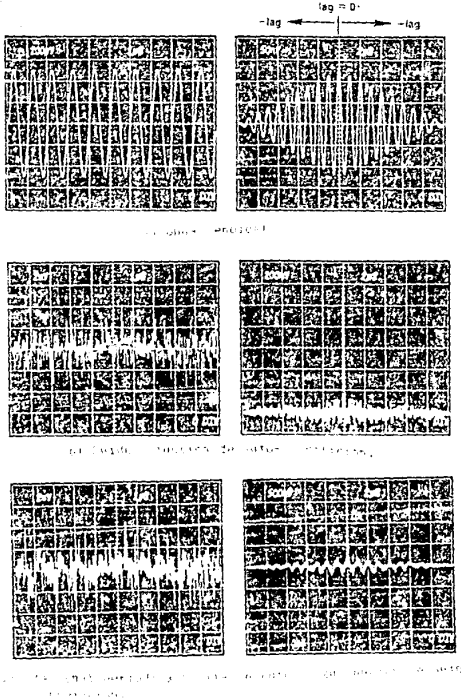


Figura 4-5. la autocorrelación detecta señales ocultas en ruido.

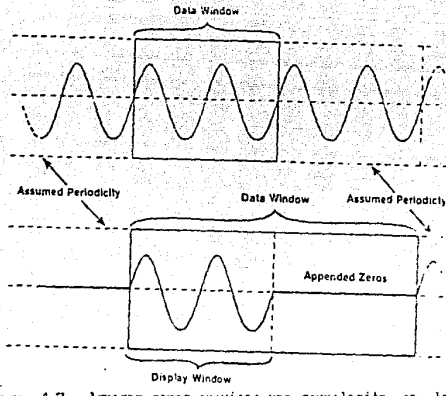
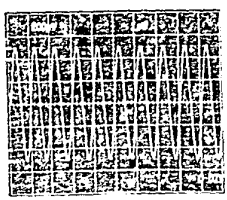


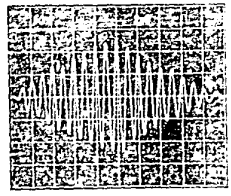
Figura 4-7. Agregar ceros previene una correlación en la TRF.

Por lo tanto la autocorrelación es una herramienta útil para detectar la presencia de señales periódicas ocultas en el ruido. Los estudios biomédicos , astronómicos y sistemas de control por tonos son unas de las posibles áreas de aplicación para las técnicas de detección por autocorrelación.

Correlación Cruzada- En la autocorrelación, una señal se multiplica por una versión de la misma señal retardada. El proceso de correlación cruzada difiere solamente en que se usan dos señales. Una se multiplica por la versión retardada de la otra. La función de correlación cruzada resultante contiene solamente aquellas componentes de frecuencia comunes a ambas señales.
Para visualizar la utilidad de la correlación cruzada, volvemos al ejemplo donde se detecta una señal oculta en el ruido. Suponemos que se recibe una señal que está distorsionada por el ruido, pero se sabe qué tipo de señal se está buscando. Esto a menudo es el caso del radar, sonar, y controles de tonos, donde la señal transmitida está bien definida pero inmersa en ruido se encuentra la señal recibida. La situación se demuestra en la 4-8, donde una señal senoidal se ilustra oculta en el ruido. Obsérvese que



a) La señal de Referencia. b) Es la señal de Referencia ?.



c) La correlación cruzada resulta positiva.

Figura 4-8. Si conocemos la señal buscada, la correlación es útil para encontrarla.

en la figura 4-7 no existen componentes de ruido en la función de correlación cruzada. Esto se porque el ruido no se combina a las señales que se correlacionan.
Si estamos interesados en tener una aproximación de una respuesta impulso de un sistema lineal, el ruido puede convertirse en una señal de prueba útil. Todo lo que se necesita es estimular al sistema con una señal de ruido de amplio ancho de banda y efectuar la correlación cruzada de esta entrada con la salida del sistema resultante. Si la prueba es conducida cuidadosamente, los resultados aproximan la forma de onda de la respuesta impulso del sistema.
Además de detectar señales ocultas en el ruido y aproximar respuestas a impulsos, la correlación cruzada encuentra muchas aplicaciones donde se debe medir retardos. Los retardos en el tiempo son un parámetro en la solución de diversos problemas: técnicas de medición de eco caracterización de sistemas de transmisión, etc. Con la correlación cruzada, se encuentra el mejor acoplamiento entre una señal transmitida y otra recibida.
Lo básico de este concepto se ilustra en la figura 4-8.

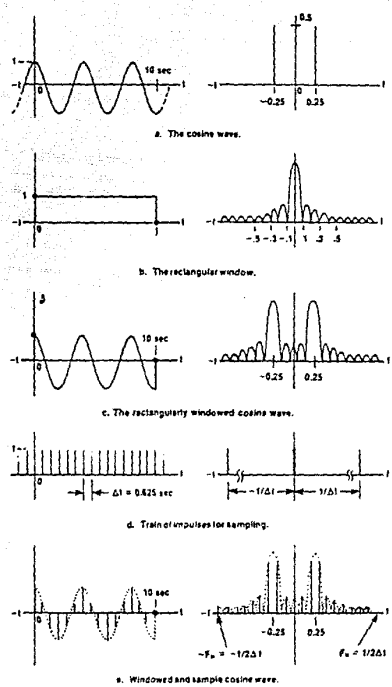


Figura 4-6 Proceso de transformación de una onda cosenoidal que tiene un número de ciclos no-entero en la ventana.

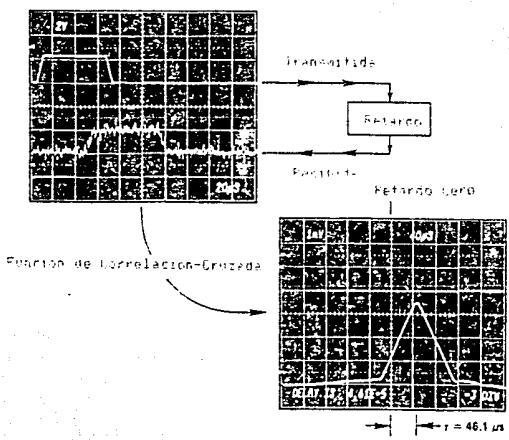


Figura 4-9. La localización de la máxima correlación cruzada indica el retardo entre las señales.

IV-6 ESPECTRO DE POTENCIA

Como parte final, se debe mencionar que la transformada rápida de Fourier o FFT de una función de correlación produce lo que se conoce como espectro de potencia.

La FFT de una función de autocorrelación se conoce generalmente como autoespectro. También referido como densidad espectral de potencia, el autoespectro se usa en forma muy amplia en el análisis de vibraciones. Cuando una función de correlación cruzada se transforma a Fourier en el dominio de la frecuencia, el resultado se conoce como espectro cruzado.

El espectro cruzado contiene los productos de magnitudes y las diferencias de fases de los componentes de frecuencia comunes a las señales involucradas en la correlación cruzada. De la misma manera que el autoespectro, el espectro cruzado se usa frecuentemente en el análisis de vibraciones.

Existe una relación importante entre parámetros al usar la transformada discreta de fourier (TDF) o la transformada rapida de Fourier (FFT). Suponemos que el tiempo verdadero de la señal discreta se deriva de una función continua en el tiempo o al menos referida a tal función para propósitos de análisis. Revisaremos los parámetros de interes :

T = separación entre las muestras en el tiempo.

fs = frecuencia de muestreo (en Hertz) = 1/T

F = incremento entre componentes de frecuencia (en Hertz)

resolución en frecuencia.

tp = longitud de registro (en segundos) = periodo efectivo de la señal de tiempo = 1/F

fo = frecuencia "folding" = fs/2 Hertz

fh = frecuencia más alta esperada en el espectro de la señal (Hertz)

N = Número de muestras en el registro.

Supongamos que la función a analizar es una señal de información. Tal como una señal en la cual el espectro se concentra principalmente en frecuencias más bajas y que no existe translación del espectro.

Para evitar superposición de colas espectrales, es necesario que :

$$f_s > 2f_h \quad (4\text{-}3)$$

este resultado implica que T debe selecionarse de acuerdo a

$$T < 1/2f_h \quad (4\text{-}4)$$

El incremento entre componentes espectrales F puede verse como la resolución en frecuencia. Para una resolución en frecuencia, la longitud de registro mínima tp la seleccionamos por

$$tp = 1/F \quad (4\text{-}5)$$

La revisión de (4-4) y (4-5) conduce a la conclusión que hay un compromiso entre el ancho de banda de la señal a muestrear y la resolución del espectro de la señal muestreada.

Para incrementar el ancho de banda de la señal, es necesario reducir T. Para una N determinada, esto reduciría la longitud del registro, por consiguiente, se decrementa la resolución en frecuencia. Por lo tanto, para aumentar la resolución es necesario incrementar tp. Para una N dada, ésto incrementaría T, por lo tanto se decrementa la capacidad de alta frecuencia.
La única manera de incrementar la capacidad de alta frecuencia o la resolución en frecuencia manteniendo la otra constante es aumentar el número de puntos N en la longitud de registros. Si fh y F se especifican, N debe satisfacer

$$N > 2fh/F \qquad (4\text{-}6)$$

La relaciones anteriores son condiciones mínimas que se deben satisfacer con un algoritmo de procesamiento básico como TDF o también TRF sin incluir ninguna técnica de modificación de datos.
Las modificaciones en los procedimientos básicos podrían alterar estas condiciones.

V- PROCESADORES PROGRAMABLES

V-1. PROCESADORES PARA TRANSFORMADA RAPIDA DE FOURIER

Los algoritmos se pueden implementar directamente usando una microcomputadora. El procesamiento en tiempo real es posible para señales con anchos de banda hasta varios cientos de Kilohertz.
Esto ha abierto en nuestros dias, grandes áreas para aplicaciones del procesamiento de señales digitales, pero es insuficiente para satisfacer los requerimientos de análisis en tiempo real de procesamiento de imágenes, radar, video, etc, que requieren de circuiteria de proposito especial y técnicas para superar las limitaciones de velocidad y memoria de las microcomputadoras de proposito general.

Una manera de implementar estas tareas es por medio de un arreglo de procesamiento de alta velocidad, que releva al microprocesador central de tareas iterativas en conjunto con tareas criticas en el manejo de la memoria, dejando al procesador libre para un control y supervisión muy eficiente en el proceso de transformación.

Dos caracteristicas importantes del arreglo de procesamiento son : la eficiencia de las operaciones aritméticas requeridas en la ejecución del algoritmo de TRF y la capacidad para manejar arreglos de procesamiento mucho mayores que la memoria del procesador.
Podemos obtener una mejora considerable en la velocidad de procesamiento con el uso de arquitectura "rebanada de bits" (bit-slice), empleando dispositivos de lógica acoplada por emisor (Emitter Coupled Logic o ECL).

Con el uso de dispositivos de 4 bits o mas con arquitectura de "rebanada de bits" se efectuan las operaciones aritméticas básicas requeridas en el calculo de TRF con operación "mariposa" [2,4]. Este tipo de implementación puede ser modular, aunque con el advenimiento de la tecnología superior, encontramos procesadores dedicados, llamados "procesadores de señales digitales" en un solo módulo. Pero existen varios que son implementados en módulos. Se requiere un cierto número de elementos aritméticos tipo "rebanada de bit" para lograr una estructura completa para la operacion "mariposa".
El calculo eficiente de la TRF de arreglos muy grandes de datos almacenados externamente requiere de métodos que consuman el mínimo de tiempo para el manejo de matrices almacenadas mayores que la capacidad de almacenamiento de alta velocidad. Existen varios algoritmos para efectuar este intercambio de disco a cinta.
Cuando el problema de intercambio de matrices mayores que el almacenamiento involucra un arreglo de procesamiento unido a un sistema principal basado en microprocesador, la transferencia se efectua entre la memoria de alta velocidad del sistema externo y la memoria del sistema principal, para

lo que requerimos un método para transferencia de alta velocidad.
Un ahorro en tiempo lo logramos cuando los datos se seleccionan en el orden requerido por el algoritmo de la FFT (podemos ver la figura 5-1). Un método para lograr esto es emplear un algoritmo implementado físicamente. Una matriz de datos de 2^m por x^n se transpone en 2^j bloques menores, los cuales pueden contenerse dentro de la memoria rápida. El algoritmo procede a ciclar estos bloques en la memoria uno a la vez, asegurando que, a pesar de la manipulación que se efectua sobre ellos durante el proceso, sean regresados a las áreas de almacenamiento externo. El número de pasos completos de los datos a través de la memoria depende del tamaño de la matriz externa y está dada por (m+j-1)/j, donde m>n. Durante estos pasos el direccionamiento del almacenamiento externo es tal que los datos se presentan de una manera ordenada requerida por las operaciones aritméticas de la TRF.

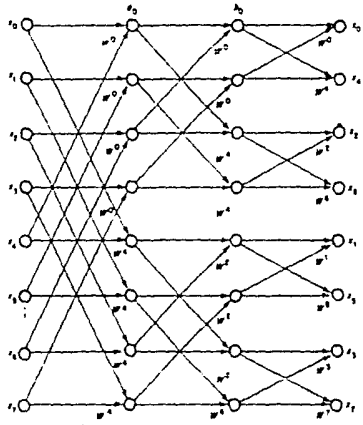


Figura 5-1. Diagrama de flujo para valores mezclados de salida arreglados o mezclados.

Para permitir que algoritmos de TRF se apliquen a un número diferente de situaciones en el procesamiento de señales, requiriendo quizás, un cambio en velocidad y/o configuración, normalmente se usan procesadores de TRF de propósito especial, los cuales son dispositivos programables de alta velocidad arreglados para funcionar como procesadores en cascada o paralelo.

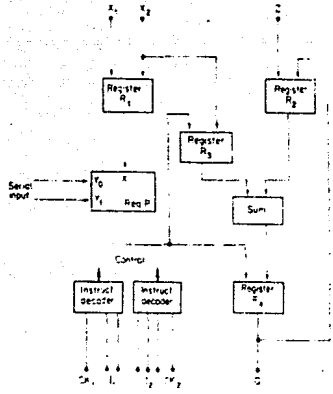


Figura 5-2. Diagrama de un elemento aritmético 'bit slice'

Estos consisten de un conjunto de módulos incluyendo un módulo de mariposa para manejar datos complejos, y módulo de almacenamiento soportando varios tipos de submódulos, en compañía de un procesador de TRF.

En la figura 5-3 mostramos un módulo "mariposa" típico. El funcionamiento puede extenderse a 16, 24 y 32 bits interconectando varios módulos. El módulo incluye multiplicadores y operadores de suma y resta. Se incluye una memoria RAM para almacenamiento de datos complejos y una memoria PROM que contiene las constantes rotacionales usadas en el calculo de TRF. La entrada de datos al módulo de "mariposa" consiste de un conjunto de 4 números complejos alimentados simultaneamente ya sea del módulo de almacenamiento o de cualquier módulo mariposa. Estas entradas se almacenan en la memoria de datos RAM, y cuando un bloque de datos reside en la memoria, se leen grupos de 4 valores complejos y se efectua la operación de "mariposa" en ellos.
Los cuatro nuevos números complejos creados por la operación de "mariposa" pueden salir del módulo "mariposa" a un módulo adyacente o módulo de almacenamiento o puede regresar a la RAM del módulo "mariposa para usarse en la siguiente etapa del calculo de TRF.

Para obtener mayor rendimiento podemos operar módulos de "mariposa" en paralelo. Podemos usar módulos en paralelo para procesar cada mitad de las entradas complejas de la TRF. Esta separación en los dos flujos de datos no está completamente terminada como se ve en la figura 5-2, cada salida de la TRF es un función de todas las entradas. Es necesario modificar para intercambiar algunas de las entradas de la configuración en un punto de la trayectoria paralela. Podemos obtener velocidad adicional con la operación de 4 ó más módulos paralelos con interconexión en la entrada.

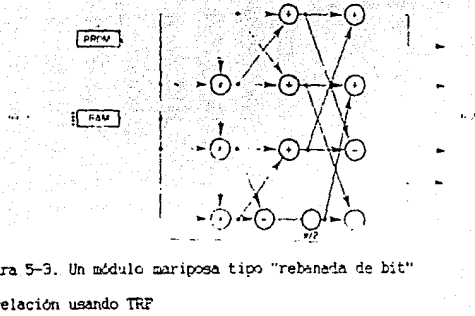


Figura 5-3. Un módulo mariposa tipo "rebanada de bit"

Correlación usando TRF

La operación de convolución :

$$R_{xy} = \frac{1}{N-r} \sum_{i=1}^{N} x_i y_{i+r} \qquad (5\text{-}1)$$

tiene su equivalente en el dominio de la frecuencia relacionado por la trasformada de Fourier.

El método para expresar esto en términos de Fourier es encontrar la TRF de cada serie, multiplicar los valores y encontrar la transformada inversa de Fourier del producto.

$$R_{xy} = x_i * y_i \qquad (5\text{-}2)$$

$$\uparrow \quad \downarrow \quad \downarrow$$

IFFT FFT FFT

$$\downarrow \quad \downarrow$$

$$X(f) \cdot Y(f) = Z(f)$$

Este método indirecto de correlación será más rápido que el de la evaluación discreta de la correlación.
Otro método indirecto aplicable a la autocorrelación que requiere un número similar de multiplicación y sumas es la evaluación de la correlación calculando primero el espectro de potencia. Son necesarios 4 pasos para llevar a cabo la autocorrelación por este método :

1- La transformación de una serie en el tiempo, x_i a una serie en la frecuencia X_j, $j = 0,1,\ldots N-1$ usando TRF.

2- La derivación del espectro de densidad de potencia, el cual es el promedio del cuadrado del valor de la transformada de Fourier.

3- La suavización del espectro de potencia por medio de una función de ventana.

4- La transformación inversa del espectro suavizado para dar la función de autocorrelación.

Hay dificultades, sin embargo, cuando queremos implementar este procedimiento en una computadora digital o un microprocesador. Suponemos que una subrutina de TRF que tiene dos argumentos N (= número de puntos) y X (= serie de valores) aceptará N puntos en un arreglo complejo que puede almacenar 2N + 2 elementos. Antes de transformar los datos añadimos un cierto número de ceros a cada valor de las series compleja y real- un cero para cada juego de puntos(por ejemplo si N=5 entonces la parte real de X podría ser 1,2,3,4,5,0,0,0,0,0). La transformada se evalua por la TRF y su resultado es reescrito en el arreglo sobre los datos de entrada. Una forma multiplexada de almacenamiento se usa donde las partes real e imaginaria de los valores en el arreglo son entrelazadas por ejemplo Re X1, Im X1, Re X2, Im X2,....

La parte real cumple la transformada de Fourier del coseno

$$X_k = \sum_{n=0}^{N-1} x_n \cos\left(\frac{2\pi nk}{N}\right) \qquad (5.3)$$

La parte imaginaria cumple con la transformada de Fourier :

$$X_k = \sum_{n=0}^{N-1} x_n \sin\left(\frac{2\pi nk}{N}\right) \qquad (5.4)$$

La densidad del espectro de potencia sin la suavización por medio de una ventana será :

$$S_k = \frac{X_k^2 + X_k^2}{N} \qquad (5-5)$$

Estos valores sobreescriben la parte real X en la memoria de la computadora.
La parte imaginaria se convierte a cero. Antes de llevar a cabo la transformada inversa de Fourier en una versión suavizada de este resultado es necesario reordenar los datos de tal manera que la parte real sea simetrica con respecto al punto de frecuencia de traslape mostrado en la figura 5-4. La autocorrelación se escribe sobre la parte real de X y, debido al translape, se convierte en 2N puntos de longitud, empezando con el punto de corrimiento de cero fase (k=0) y repetido en orden inverso después que se alcanza el punto medio (N). La razón para la adición de ceros y la forma repetitiva del autocorrelograma así formado es debido a la longitud finita de los datos correlacionados. Lo que sucede cuando dos trasformadas de Fourier se multiplican es que los elementos que han sido defasados o corridos en el tiempo más que la longitud permitida en la serie de datos reaparecen al principio otra vez, causando distorción en el correlógrama. El procedimiento detallado anteriormente evita este efecto.

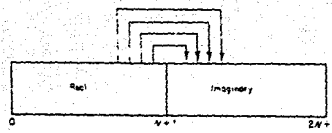


Figura 5-4. Espectro de Fourier por translape

Una fuente de error posterior ocurre cuando el máximo retardo r aparece como una fracción apreciable de la longitud de la señal. Podemos evitar ésto multiplicando el resultado dado anteriormente por un factor de escala,

$$W_k = \frac{N-k}{N} \qquad (5\text{-}6)$$

Como una consecuencia de estas fuentes de errores vemos que tenemos que modificar el procedimiento dado anteriormente con los siguientes dos pasos :

3(a) Suma de N ceros a la densidad del espectro suavizado antes del paso 4.
4(a) La multiplicación del correlógrama resultante por W^k según el paso 4.

La programación para llevar a cabo ésto forma parte de la libreria de rutinas de la mayoría de los compiladores de escala media y grande. Existe mayor disponibilidad en la actualidad con respecto a microprocesadores, también especialmente donde los procesadores están asociados con periféricos especiales para una multiplicación rápida.

V-2. EL PROCESADOR PROGRAMABLE DE SEÑALES

Podemos reconocer tres tipos de arquitectura física de procesadores. Primero tenemos los dispositivos "alambrados" que son capaces de efectuar una función específica tal como la TRF de una secuencia de datos de entrada. Debemos hacer arreglos para aplicar estos datos como flujo de entrada múltiple o serial y manejar la salida producida por el dispositivo. Se requiere un microprocesador de control o una microcomputadora de acuerdo al nivel del dispositivo.
Podemos reconocer el hecho de que el procesamiento de señales está involucrado con el filtrado digital y transformada de Fourier o transformaciones similares en grandes bloques de datos y diseñar nuestra arquitectura de acuerdo con lo anterior. También requerimos otras funciones tales como promedio, integración y formato de salida. Entonces podemos organizar el procesador de señales sobre una base funcional, de acuerdo a estas funciones que son efectuadas por dispositivos individuales o hardware asociado con rutinas de software particulares, cada una de las

cuales es programable.
Los módulos pueden pertencer a los microprocesadores bipolares tipo "rebanadas de bits" tales como el AM2903 usado como elemento aritmético paralelo.
Este enfoque modular nos lá nuestra segunda arquitectura como el control programado de un conjunto de funciones básicas de programación y circuitos requeridas en una operación de procesamiento de señales.

El principio básico es que el microprocesador genera un flujo de comandos o instrucciones de alto nivel a los diferentes módulos que se ejecutan en secuencia. Este tipo de arquitectura modular interconectada dá más flexibilidad de interconexión que los dispositivos alambrados pero aún requiere algunos cambios en la interconexión y posiblemente en módulos de memoria para diferentes tipos de procesamiento.
Los nuevos desarrollos en circuitos complejos de una sola pieza o paquete, ofrecidos por los diseños con la tecnología VLSI, ahora permiten la posibilidad de diseño de un procesador especial como el descrito anteriormente. O dispositivos más complejos donde la arquitectura se diseña para proveer un enfoque paralelo (pipe-lined) para las operaciones sencillas repetitivas de suma, substracción y multiplicaciones complejas. Obtenemos, así un gran rendimiento adecuado para aplicaciones en línea del procesador de propósito general, junto con la capacidad de programar en lenguaje ensamblador o aún efectuar programación de funciones en lenguajes de alto nivel.
La tercera posibilidad en arquitectura del diseño para procesamiento de señales tiene como resultado el procesador de señales programable (PSP).

V-3. DISEÑO DEL SISTEMA FISICO (HARDWARE)

Ya que el módulo de procesamiento especializado efectua operaciones básicas, las velocidades de transferencia en E/S son relativamente bajas en un PSP. La flexibilidad en la trayectoria de datos requerida, incluye la posibilidad de un uso no demasiado eficiente debido a una limitación en las patillas del circuito integrado. Sin embargo, la mejora considerable en eficiencia de programación de control y capacidad para escribir ésto en lenguaje de alto nivel que es más fácil y barato escribir y dar mantenimiento a los dispositivos PSP.
En cuanto a arquitectura el PSP generalmente está diseñado como un número de elementos controlados independientemente, los cuales operan principalmente, en forma paralela.
Estos elementos incluyen una unidad paralela, un control de canal de entrada/salida con funciones de decodificación y direcciones y un procesador de control operando con un programa interno.
Cada elemento tiene su memoria de trabajo para prevenir conflictos de acceso o "cuello de botella" que ocurren con el uso de una memoria principal única. Hay un uso extensivo

de ROM y RAM en la programacion y control de cada módulo. La reprogramación sólo requiere la escritura del nuevo código o el intercambio. El sistema se puede expandir fácilmente usando mayor capacidad de memoria.
Son posibles una gran variedad de diseños. Varios de ellos usan microprocesadores interconectados, donde un procesador provee el control de la memoria principal, el segundo se encarga del control un tercero sirve de unidad aritmética. Otra división de funciones es diferenciar entre TRF y operaciones que no involucran TRF y ejecutar estas instrucciones en unidades de procesamiento separadas. Esto puede ser extremadamente eficiente para operaciones en arreglo tales como la función fundamental de multiplicación/adición :

$$\sum_{i=0}^{n-1} a_i x \qquad (5\text{-}7)$$

que forma el nucleo de todos los procesos de filtrado digital y transformadas de Fourier.

Un PSP tipico contiene la mayoria (si no todos) los elementos mostrados en la figura 5-5. Donde podemos identificar cinco (5) elementos conectados por un Bus principal :

1) Procesador de control

2) Procesador aritmético

3) Memoria

4) secuenciador

5) Procesador de E/S

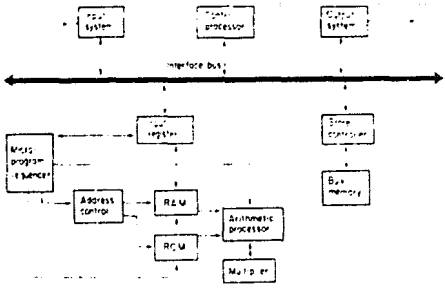


Figura 5-5. Procesador de Señales Programable

El procesador aritmetico es importante ya que es esencial para efectuar las operaciones de multiplicación/adición definidas por la ecuación (5-7), con precisión y velocidad el cual efectua un cierto número de procesos en paralelo.
Es necesario un microprograma horizontal de control para efectuar un cierto número de procesos en paralelo. Para muchas aplicaciones solamente una implementación tipo "rebanada de bit" (bit slice) es práctica y se logra por medio de un proceso paralelo (pipe-line), por ejemplo, efectuamos un cierto número de calculos por etapas en secuencia para determinados datos donde las etapas funcionan en paralelo.

Es posible efectuar varios ciclos de busqueda (fetch) anticipadamente a la instrucción que está siendo ejecutada, de tal manera que puede efectuarse la decodificación de instrucciones anteriores.
La secuenciación de estas operaciones puede controlarse por un secuenciador adicional tal que durante un ciclo del procesador se efectuan tres o más funciones en paralelo :

1- Los datos a multiplicar se transfieren al registro de entrada.

2- Los datos direccionados en ROM/RAM se envian al procesador donde se añaden al contenido ya presente efectuando una suma del producto.

3- Almacena el siguiente par de direcciones en el registro de direcciones en espera de la siguiente instrucción.

Un PSP es un microcircuito integrado que incorpora muchos de estos conceptos; se desarrolló en los laboratorios BELL.
Esto está diseñado usando tecnología MOS con canal N que opera a 5 MHZ con 40 patillas.
Este procesador consiste de elementos independientemente controlados :

Una unidad aritmética de direcciones para controlar el acceso a memoria; y una unidad I/O para proveer interfaz de datos serie. Otros elementos consisten de una unidad de control que sincroniza estos elementos y provee una decodificación temporal de instrucciones, RAM para almacenamiento temporal de resultados y ROM para el programa.
El procesador BELL se programa en ensamblador, pero tiene caracteristicas adicionales requeridas en el procesador paralelo (pipe-line) tal com la capacidad de manejar un formato de comando múltiple por instrucción. El software de soporte incluye un simulador interactivo para rastrear (debug) el programa y un editor de pantalla.

Un sistema comercialmente disponible tambien basado en un tipo modular, es el procesador digital Advanced Micro Devices 29500. Este incluye un procesador aritmetico con un

multiplicador por hardware, un controlador de direcciones responsable de la generación de la secuencia de direcciones requerida por la TRF y otros algoritmos de procesamiento de señales, y un secuenciador para controlar todo el sistema.
La figura 5-6 muestra la arquitectura usada para la unidad aritmética. Ya que la operaciones del procesamiento de señales involucra la manipulación de números complejos requiriendo el doble de operaciones de suma/resta y cuatro veces los productos cruzados que los números reales, es necesario un sistema con proceso paralelo utilizando multiplicadores por hardware separados y buses de datos diferentes y una unidad logica aritmética para calculos de números reales e imaginarios.

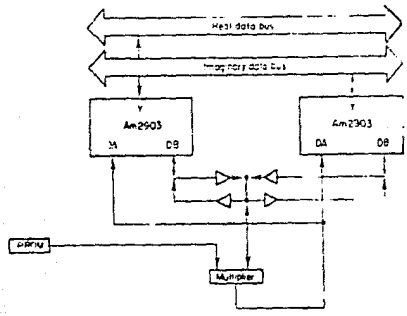


Figura 5-6. La unidad Aritmética de un PSP (Advanced Micro Devices Ltd.)

Las constantes se toman de un bus distinto pero esto produce una disminución en el tiempo del ciclo de todo el proceso.

Una arquitectura paralela de bus multiple se adopta para los procesadores de señales programables que incluyen el control de la unidad lógica aritmética. Los diversos registros y la unidad lógica aritmética hacen que estos sistemas se conecten por medio de 10 buses separados de 8 bits, permitiendo la operación de los datos complejos y reales en paralelo. La secuencia de direcciones se provee en un componente de un solo microcircuito integrado, el cual puede generar varias secuencias de direcciones requeridas para un rango de los algoritmos de la TRF.
Los parametros que se pueden microprogramar o alambrar en el sistema son la transformada en longitud, decimación en tiempo o frecuencia, radio-2 o radio-4 e inversión de bits. La señal de status se provee para la realización en columna del módulo mariposa y las transformadas. Para

habilitar las operaciones de escritura/lectura de tal manera que sean concurrentes, y para permitir que los calculos de los módulos de mariposa se efectuen de una manera traslapada, las direcciones de los registros se diseñan en forma paralela (pipe-line). Ellos consisten de 4 registros de 8 bits los cuales se pueden programar como pilas (stacks) dobles o como pilas sencillas de 4 niveles. Las salidas de registros se conectan por medio de un multiplexador a la salida del dispositivo.
El uso de tales tecnicas paralelas es esencial para el procesamiento en tiempo real. En el caso del AMD29500, por ejemplo, los módulos de mariposa de la FFT se pueden ejecutar en 400 ns dando una transformada en forma completa de radio-2 de 1024 puntos en aproximadamente 2 ms.

V-4. SOFTWARE PARA LOS PSP

La complejidad de los PSP en comparación con otros sistemas con microprocesador ha indicado la necesidad para dos niveles de programación.
Primero, un lenguaje de microprocesador para el diseñador de sistemas. Esto debera de ser orientado al flujo o estructura de datos. Requerimos acceso programable para controlar detalladamente el funcionamiento de Hardware. Los microprogramas son necesarios para controlar especificamente el procesador de control, el procesador aritmético y en la implementación del almacenamiento del control de escritura/lectura. Muchos de estos controles necesitan derivarse de una forma de instrucción con formato tipo múltiple y debe incluir la habilidad para controlar el acceso y decodificación de instrucciones simultáneas.
Segundo, un lenguaje propio del procesamiento digital de señales para el programador de sistemas.
Este es orientado con su arquitectura y también los juegos de instrucciones diseñados de acuerdo a las aplicaciones de procesamiento de señales donde la función multiplicación/suma fundamental (ec. 5-7) es sobresaliente. También encontramos que el lenguaje de procesamiento de señales es estructurado por bloques. Esto es, el programa describe "acciones" por comandos o instrucciones y datos por medio de "declaraciones". El vocabulario de comandos necesita incluir asignaciones, "cases", "for", instrucciones condicionales (if, then, else), "go-to" restringido, "return", "exit", instrucciones de procedimientos (procedure) y multiprogramación. Estas ultimas incluyen manipulación de tareas, instrucciones de señales y sincronización y comprende instrucciones múltiples para llevar a cabo las operaciones de los módulos "mariposa" y filtrado recursivo, ambas basadas en operaciones de multiplicación/suma. Muchos dispositivos PSP bajo desarrollo o ya desarrollados incluyen un lenguaje de procesamiento de señales, tales como PASCAL, como un base para añadir comandos con funciones especiales tales como las mencionadas.
Las instrucciones reales que deben llevarse a cabo, es

conveniente clasificarlas en 4 grupos. Primero tenemos la clase de juego de instrucciones para efectuar tareas generales y de supervision que encontramos en computadoras de proposito genera. Segundo, las instrucciones de transferencia de datos que obedecen a estructuras de comandos ya establecidos para proveer transferencia rápida de datos entre la unidad almacenamiento y la unidad del procesador y su multiplicador asociado. El tercer grupo es el que comprende las instrucciones de entrada/salida que podrían controlar los convertidores A/D y D/A así como equipo más sofisticado, por ejemplo, un equipo para reconstrucción de imágenes.
Finalmente, tenemos las instrucciones para funciones complejas para llevar a cabo algoritmos de TRF y operaciones similares.
Con un conjunto de instrucciones de este tipo, el procesador de señales programable puede ser visto como una mejora considerable con respecto a los sistemas alambrados o con estructura por bloques orientada, además de la facilidad con la que podemos usar el mismo procesador de señales programable

processor el set de instrucciones de este dispositivo tiene comandos como FFT
Para procesamiento en tiempo real hasta el rango de voz (10KHz) los dispositivos de un solo integrado usualmente ofrecen la solución más conveniente
Las señales de video requerieren de mayor nivel de procesamiento.
Esto puede efectuarse por distintos medios que incluyen el diseño de un procesador de señales por microcodigo basado en LSI. Esta opción es la que se ha adoptado en muchas compañías que se han esforzado en este campo.
Al igual que en la solucion de un único integrado DSP, la clave es la multiplicación por hardware. El Plessey PDSP16112 lo efectua en 100 ns. el TRW MPY016K en 45 ns. etc.
El diseño involucrando integrados multiples es mas comun.
El bloque básico es el procesador aritmético , este consiste usualmente de un procesador 'bit slice' o 'word slice' y el multiplicador por hardware, el controlador de microprograma que incluye un secuenciador de instrucciones y los generadores de direcciones y datos.
Un sistema con componentes estándares es el 2901 que es un procesador 'bit slice' de 4 bits, un multiplicador/sumador por hardware, almacenamiento, y 2 microcontroladores 2911. Usando esta clase de implementaciones una TRF de 1024 puntos se efectua en 10 ms.
Se han reportado sistemas basados en dos Plessey PDSP163164, que son acumuladores para el manejo de números complejos y un multiplicador por hardware, siendo capaz de generar el calculo del algoritmo para una configuración 'mariposa' en 50 ns calculando una TRF de 1024 puntos en 256 .
La tendencia creciente en algunos sistemas es hacia el calculo con punto flotante, algunos fabricantes como TRW, Analog Devices y AT&T ofrecen integrados de este tipo.
Los integrados para procesamiento digital han alcanzado el precio y la eficiencia suficientes, para su uso a bajo costo.
El apoyo obtenido para herramientas de desarrollo es bastante aceptable.
En el area de sistemas con integrados múltiples de arquitectura avanzada, han abierto el desarrollo de procesadores de señales con arreglos de procesadores.
También con el apoyo de microsecuenciadores y multiplicadores de alta velocidad se ha logrado un avance considerable.

APLICACIONES Y TECNICAS PARA EL TMS32010

Las aplicaciones típicas para DSP en el procesamiento de señales incluye el filtrado digital, correlación, transformada rápida de Fourier, transformada Hilbert, filtros adaptables, procesamiento de sonar y radar, procesamiento sísmico, procesamiento de voz y generación de señales

En el area de instrumentacion comprende el analisis espectral (normalmente basados en la TRF) (.7' filtrado digital, PLL, análisis de transitorios y generación de ondas.
La figura 5-8 muestra el diagrama de bloques del TMS32010 de la cual puede verse que el dispositivo esta basado en una arquitectura Harvard modificada, la discrepancia estriba en que el dispositivo permite transferencia en programas y datos. En una arquitectura Harvard modificada la memoria de programas y datos tienen espacios diferentes, lo que permite un traslape total en los ciclos de 'fetch' y ejecución.
Por medio de esta modalidad, los coeficientes pueden almacenarse en RAM.
Una amplia gama de aplicaciones requieren correlacion y filtrado digital como parte de un proceso. La figura 5-9 muestra el TMS32010.
Los pasos principales se muestran en la figura 5-10. para implementar un filtro de respuesta impulso finita. Los coeficientes , h(k), que se almacenan en ROM deben ser inicialmente leidos y escritos en RAM, usando la instruccion TBLR. Despues de esto, la entrada x(n)m, es leida hacia la RAM, y entonces se efectuan las operaciones de multipicación, suma y corrimiento.

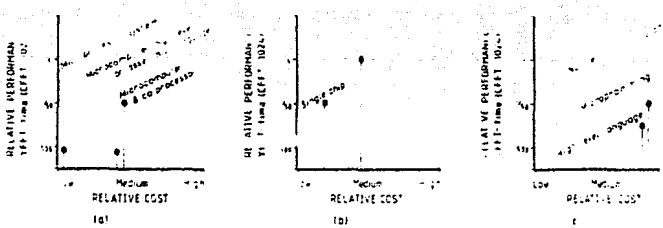


Figura 5-7 Comparación de costo/eficiencia de DSP's

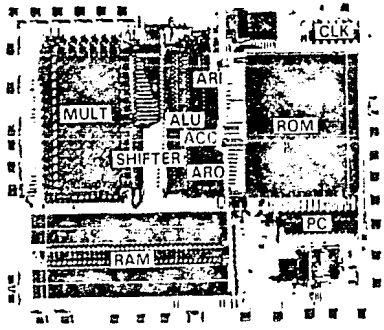


Figura 5-9. TMS32010

Alternatives	Execution speed (ms)	Programming effort	Hardware design effort	System Cost Relative	System Cost Average $	System flexibility
GP high level lan	250	High	None	Some average	1 200	High
GP microprogramming	50	High	None	Some average	1 000	Above average
GP - number cruncher	5	High	None	High	15 000	Above average
Microcomputer	1000	Below average	None	Medium	200	High
Microcomputer - control	20	Below average	None	Medium	1 000	Above average
Microprocessor sys	1000	Average	Medium	Low	75	Average
Single chip SP	50	Above average	Medium	Above average	200	Average
Multi-chip SP	5	High	High	Medium	1 000	Low

Tabla 5-1. Tabla comparativa para desarrollo

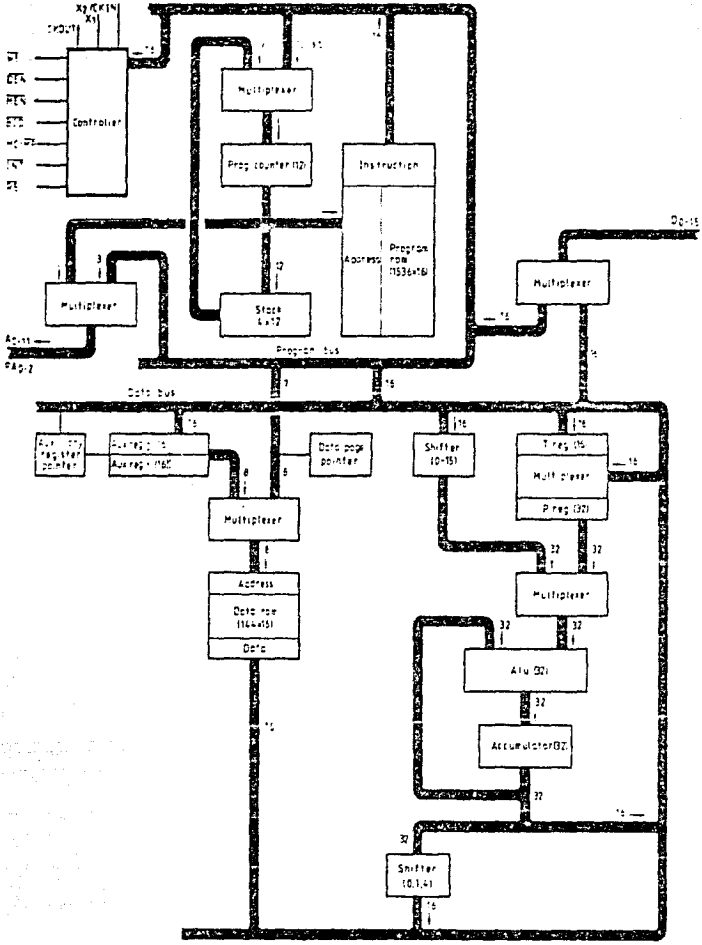


Figura 5-8. Diagrama interno del TMS32010.

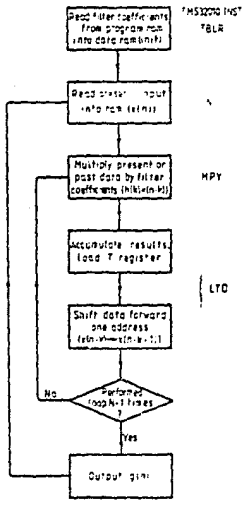


Figura 5-10 Diagrama de flujo para la implementacion de un filtro tipo FIR.

```
* THIS SECTION OF CODE IMPLEMENTS THE FOLLOWING EQUATION:                *
* x(n-4)h(4) + x(n-3)h(3) + x(n-2)h(2) + x(n-1)h(1) + x(n)h(0) = y(n) *
*
NXTPT   IN    XN,PA0          * GET THE NEW INPUT VALUE XN FROM PORT PA0 *
*
        ZAC                   * ZERO THE ACCUMULATOR *
*
        LT    XNM4            * x(n-4)h(4) *
        MPY   H4
*
        LTD   XNM3            * x(n-4)h(4) + x(n-3)h(3) *
        MPY   H3
*
        LTD   XNM2            * SIMILAR TO THE PREVIOUS STEPS *
        MPY   H2
*
        LTD   XNM1
        MPY H1
*
        LTD   XN
        MPY   H0
*
        APAC                  * ADD THE RESULT OF THE LAST MULTIPLY TO *
                              * THE ACCUMULATOR                        *
*
        SACH YN,1             * STORE THE RESULT IN YN *
*
        OUT   YN,PA1          * OUTPUT THE RESULT TO PORT PA1 *
*
        B     NXTPT           * GO GET THE NEXT POINT *
```

Figura 5-11. Código a implementar para un filtro FIR long-5

VI- MANEJO DE ENTRADA/SALIDA EN MICROCOMPUTADORAS

VI-1 FUNDAMENTOS DE TRANSFERENCIA DE DATOS

La verdadera definición de una operación de entrada/salida es la acción de transferir desde o hacia un dispositivo periférico. Un ejemplo es la manera de mandar un solo bit a un periférico.
La figura 6-1 ilustra una transferencia de un solo bit. El dispositivo de salida consiste de un registro disparado por flancos de subida y un diodo emisor de luz (LED) para indicar el estado del registro. De acuerdo a un diagrama de tiempos, el microprocesador presenta datos validos y una línea de salida al registro. La unidad de control levanta la línea de salida a un nivel lógico de "1" en el punto medio del intervalo donde el dato es válido. Mientras el dato se transfiere del microprocesador al registro del dispositivo de salida, se efectua un ciclo de salida de un bit. El LED o diodo emisor actua como dispositivo periférico de salida.

La mayoría de los microprocesadores tienen ciclos de salida. Los datos sobre la línea usualmente reflejan un byte en el acumulador mientras el ciclo de salida se genera por una instrucción de salida.

La figura 6-1(b) muestra la extensión lógica de la transferencia de un solo bit: la transferencia de datos paralela.

En este caso, se envian ocho bits de datos a un registro de 8 bits donde se usan para manejar un despliegue de 7 segmentos, en lugar de un LED. Otra vez se aplica el diagrama de tiempos de de la figura 6-1(a).
La figura 6-1(c) muestra un ciclo de entrada de un solo bit. En este caso, la señal de entrada del microprocesador se usa para muestrear los datos a la entrada del registro 'D', un breve tiempo despues del flanco de subida del reloj de entrada. los datos se vuelven estables en la salida del registro y éste es muestreado por el microprocesador.
Otra vez, la unidad de control del microprocesador provee todas las señales de tiempo apropiadas para la transferencia de datos y usualmente transfieren los datos válidos en la línea a uno de los registros del microprocesador donde pueden accesarse por el programa del usuario.

Extendiendo estos datos de entrada a 8 bits la unidad de control dá la lógica de transferencia de entrada al teclado de la figura 6-1(d).
Un codigo de 8 bits se genera presionando una tecla en el teclado. El microprocesador procede a examinar el codigo de 8 bits moviendo el dato hacia el registro y enviandolo al acumulador del microprocesador o a uno de sus registros.

Todo lo descrito anteriormente es verdadero siempre que no

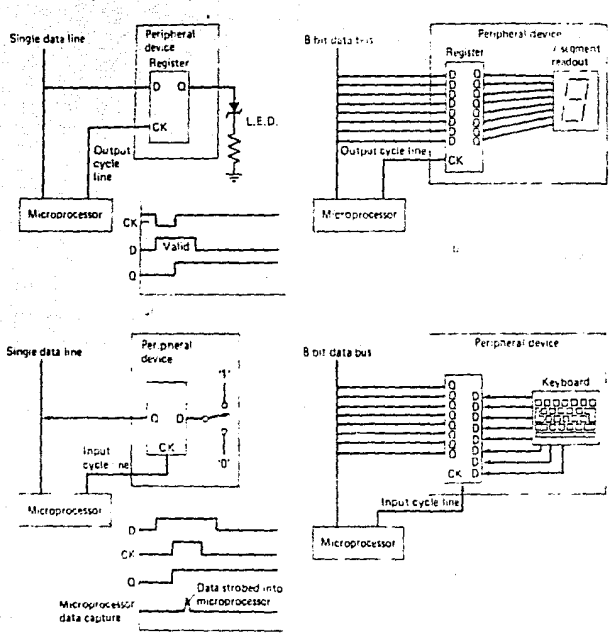


Figura 6-1. Transferencia sencilla de entrada y salida. a)Transferencia de un sólo bit. b)Transferencia de salida paralela c)Entrada de un sólo bit d)Entrada paralela

existan mas dispositivos de entrada/salida o perifericos en el bus del microprocesador. Si usamos dos o más dispositivo, surge la duda de qué dispositivo debe recibir los datos. Esto se resuelve con lógica de selección.
La figura 6-2(a) muestra tres dispositivos de salida manejados por las mismas líneas de datos y control. El microprocesador especifica a que dispositivo corresponde la entrada o la salida por medio de un codigo de dispositivo. El decodificador en los dispositivos perifericos activa o desactiva la transferencia de datos a un dispositivo particular dependiendo del código del dispositivo.

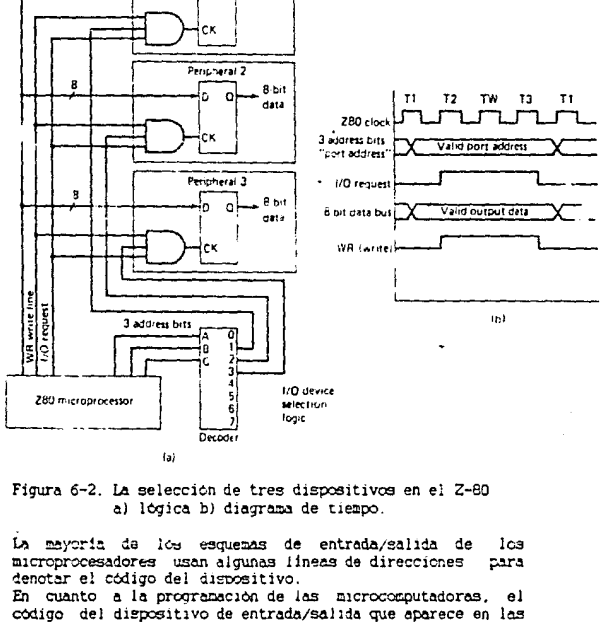


Figura 6-2. La selección de tres dispositivos en el Z-80
a) lógica b) diagrama de tiempo.

La mayoría de los esquemas de entrada/salida de los microprocesadores usan algunas líneas de direcciones para denotar el código del dispositivo.
En cuanto a la programación de las microcomputadoras, el código del dispositivo de entrada/salida que aparece en las líneas de direcciones se especifica en la instrucción de entrada/salida. Una instrucción OUT 5 de un microprocesador 8080, por ejemplo, pone el valor 5 en la línea de direcciones y los contenidos del acumulador en las líneas de datos.

Los ciclos de entrada/salida del Z-80 de Zilog son ejemplos ilustrativos de entrada/salida. La figura 6-2(b) ilustra el ciclo de salida del Z-80

Notese que en lugar de una linea de salida, hay dos lineas que deben combinarse para crear una salida: la linea de IORQ (I/O request) y la linea WR (WRITE). Notese que la direccion del puerto o codigo de seleccion que aparece en los 8 bits menos significativos de las lineas de direcciones es válida antes de que las lineas de salida se activen. Este margen sirve para permitir a los decodificadores de seleccion estabilizar y seleccionar adecuadamente el dispositivo de salida antes de enviar el dato.

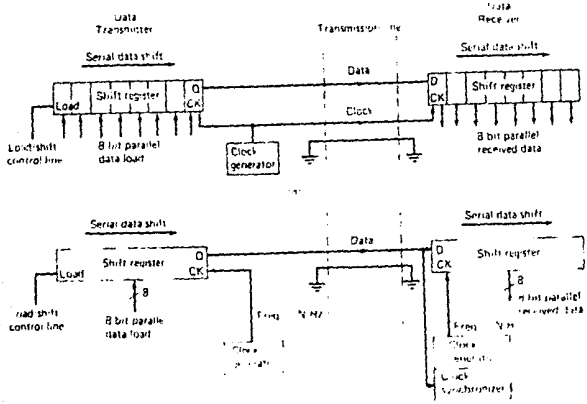


Figura 6-3. Esquemas de comunicacion a) sincrono b) asincrono

VI-2. INTERFAZ SERIE

Enviar un byte de datos a un registro de salida y proveer un reloj de entrada para mover los datos, es una manera sencilla y rápida de enviar datos al dispositivo periférico. No es practico, sin embargo, tener 8 ó más lineas de datos y una linea de reloj en todos los periféricos.
En esta situación es ventajoso remplazar la transferencia de datos paralela con la transmisión serie de datos.

La transmisión de datos en serie es el proceso de dividir los bytes en unicamente bits y enviarlos a los dispositivos periféricos uno por uno

Existen algunos problemas en la implementación de la comunicación serial. Primero, es necesario un método efectivo de convertir de la forma paralela a forma serial. Esta tarea la puede efectuar un microprocesador bajo control de programación o un registro de corrimiento con el control adecuado. Una cadena de ocho registros se carga con la palabra de 8 bits en el primer pulso de reloj de la operacion de la transmisión serial, y los bits se mueven secuencialmente a través de la cadena de registros a razon de uno por pulso de reloj.
De una manera similar, los bits que llegan a la parte receptora son recorridos en forma serial a un registro hasta que se acumulan 8 bits. Esta palabra digital completa puede ser usada por un dispositivo periferico. Como se ilustra en la figura 6-3, una linea de entrada, una linea de salida, y una línea de reloj es todo lo que se necesita para una comunicación serial bidireccional entre dos dispositivos.

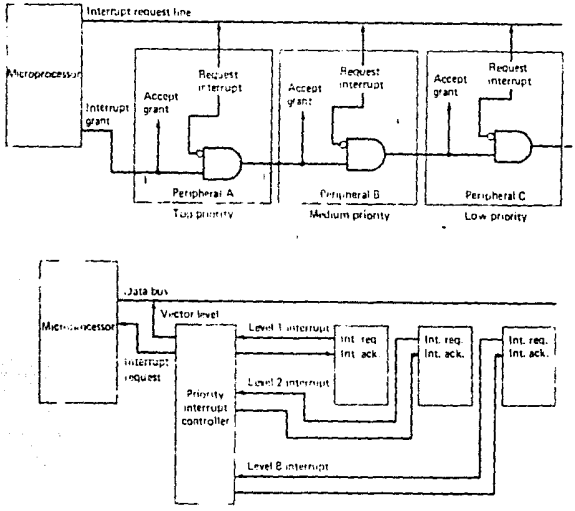


Figura 6-4. Sistema para manejo de interrupciones. a) Daisy-chain. b) sistema vectorizado de interrupciones.

Si los dos dispositivos están en forma separada y poseen diferentes fuentes de poder, también debe existir una línea común o de tierra entre ambos.

VI-3. INTERFAZ ASINCRONA

Una forma común de transmisión de datos que elimina la necesidad de sincronizar por reloj es la comunicación asíncrona. La Comunicación serial asíncrona reside en el hecho de que dos relojes de aproximadamente la misma frecuencia permanecen sincronizados por cortos periodos de tiempo.
Un transmisor de datos asíncrono envía una señal inicial llamada bit de inicio seguido de 8 bits de datos y un bit de parada (que puede ser de 1, 1 1/2 ó 2). El receptor asíncrono sincroniza su reloj a la recepción del bit de inicio y mueve 8 bits de datos usando el receptor sincronizado como guía.
El bit de parada al final del flujo de datos serial es usualmente utilizado por el equipo de recepción para determinar si el reloj presenta desviaciones para proveer datos adecuados.

La comunicación serial requiere muchos mas circuitos de control que la comunicación paralela. Ademas de la interfaz paralela estándar al canal del microprocesador, debemos usar conversión serie-paralelo, sincronización de reloj y lógica asociada.

El uso común de la comunicación serial ha dado como resultado microcircuitos integrados para manejar la interfaz compleja y la tarea de conversión. Entre estos circuitos integrados están el UART (Universal asynchronous receiver/transmitter), el USRT (Universal synchronous receiver/transmitter), y ACIA (Asynchrounous communication interface adapter).

VI-4. METODOS DE ENTRADA/SALIDA PARA MICROCOMPUTADORAS

Cada microcomputadora tiene su propia manera de transferir datos. Con la excepción de algunos microprocesadores exóticos, la mayoría de ellos usan el método del canal de datos o mapeo de memoria.

Canal de datos

En el inicio de la computación, cuando la optimización lógica y de circuitos no era entendida completamente, las computadoras se mostraban como una máquina consistiendo de un procesador, memoria y una unidad de entrada/salida.

La memoria conectada al procesador por medio de una interfaz y los periféricos por medio de otra interfaz.

Fue necesario tener instrucciones separadas para referencia a memoria y para operaciones de entrada/salida.
Este esquema de entrada/salida se llama canal de datos.

Mapeo de memoria

Conforme la computación avanza, es urgente la optimización y un conjunto de instrucciones fáciles de usar. Se optimizarón los circuitos y el conjunto de instrucciones tuvó una simplificación por medio del manejo de la lógica de entrada/salida y de memoria del procesador en forma simultánea mas que en forma independiente.

Una interfaz por mapeo en memoria es casi identica a una interfaz de memoria, pero en lugar de usar RAM o ROM, se incorporan registros de salida. Como en una interfaz de memoria, la interfaz del periférico debe contener un reconocedor de direcciones de 16 bits, buffers y lógica de protocolo asociado con el bus del microprocesador.

El procedimiento de entrada/salida por mapeo en memoria conduce por si mismo a la organización del programa y una cierta sección localizada para dispositivos periféricos del espacio de direccionamiento del procesador (usualmente bloques de 4K u 8K). Estas localizaciones en memoria se conocen como registros de dispositivos (lo contrario a puertos para canal de datos). En los programas los datos se transfieren desde y hacia estas localidades tal como se maneja en memoria.

VI-5. VENTAJAS DE LOS DIVERSOS TIPOS DE E/S

El argumento común es que la interfaz en el canal de datos requiere menos circuitos a causa de las direcciones de puertos que son pequeñas, en lugar de direccionamiento completo de memoria que se utiliza para seleccionar el dispositivo.

Otro argumento en contra del mapeo en memoria es que ocupa memoria real. Este argumento no es válido porque los dispositivos entrada/salida usualmente se agrupan en un area de memoria localizada, y el espacio util tomado de la memoria del procesador usualmente es despreciable.
El gran espacio de direccionamiento del mapeo en memoria tiene ventajas en aplicaciones especiales. Ocasionalmente es necesario tener más de 256 puertos de entrada/salida.
Con el mapeo en memoria, se pueden tener cientos de periféricos.
Los argumentos acerca de las ventajas de los dos metodos de E/S son interminables, pero es interesante hacer notar que la mayoria de las computadoras modernas usan mapeo en memoria.

Inicio de la Transferencia de E/S

Hemos supuesto que el microprocesador sabe qué periférico desea enviar el dato y cuando enviarlo. Pero en la realidad no es tan sencillo. Los dispositivos periféricos solamente pueden aceptar datos con determinada velocidad, y son capaces de aceptar datos en cualquier momento. Hay dos métodos para saber cuando empezar la transferencia: la inspección del 'status del dispositivo' bajo el programa de control o "polling" y el programa de interrupción iniciado por el periférico o manejo de E/S por interrupción.

Muestreo (Polling)

Uno de los métodos más comunes y más sencillos para el manejo de la información es tener registros de entrada separados dentro de la interfaz donde el procesador puede saber el 'status' de la información de nuevos datos mandados a la interfaz.

La gran ventaja del manejo de E/S por muestreo es su simplicidad de Hardware. La mayor desventaja es la cantidad de tiempo de procesamiento que consume. El monitoreo de las banderas de status consume mucho tiempo, al dejar el ciclo de muestreo (polling), aún en un corto periodo de tiempo para efectuar algun otro procesamiento puede causar que los datos se pierdan.

La cantidad de tiempo empleado en los lazos de muestreo se puede reducir con los bits de status para muchos dispositivos codificados o empacados en una sola palabra de status, ya sea con un bit representando cada dispositivo o un código indicando que puerto está requiriendo un ciclo de E/S; pero persiste la necesidad de un continuo muestreo (polling).

VI-6. E/S POR INTERRUPCIONES

Ya que es la acción del periférico, la que inicia la transferencia de datos, parece más razonable que el periférico 'avise' al procesador cuando está listo para mandar nuevos datos, en lugar de tener al procesador continuamente preguntando si tiene un pendiente. Esto es principalmente la idea que hay en el fondo del manejo de E/S por interrupciones : cuando un periférico tiene datos que transferir, le avisa al procesador.
Debemos incluir un sistema de interrrupciones en la estructura de control del microprocesador si usamos este método de E/S. Una interrupción causa que un programa se detenga, y efectua la ejecución de un programa adicional que recibe o envia datos.
La forma mas sencilla es el sistema de interrupcion de una sola linea.
En este sistema, una interfaz en el bus del microprocesador

simplemente pone un 1 lógico en la línea de interrupción que conduce a la lógica del control del procesador.
Al final de la instrucción en ejecución , el programa se dirige a una dirección fija de interrupción. Un programa localizado en esta dirección acepta datos o da servicio a la interrupción.
Con el sistema de interrupción de una sola línea, es posible tener muchos dispositivos emitiendo interrupciones; pero cuando la ejecución se dirige a la dirección de interrupción fija, debemos tener una rutina de muestreo para que el periférico coloque su dirección de inicio del programa sobre el bus de direcciones o datos; cuando el dispositivo interrumpe, el procesador puede tomar esta dirección y la ejecución de la rutina de servicio apropiada. Este tipo de sistema de interrupción trabaja bien pero involucra mayor complejidad de interfaz que el esquema sencillo de usar líneas de interrrupción múltiple. Se requieren circuitos generadores de direcciones y circuitos más complejos en cada interfaz de los diferentes dispositivos.

VI-7. PRIORIDADES DE INTERRUPCION

En sistemas de interrupción muchos periféricos pueden requerir una o varias interrrupciones, y eventualmente dos o más interrupciones simultaneas. En este caso una lógica de control de prioridades, o arbitro de prioridades, debe decidir cual interrupción es más importante.

El sistema de prioridades por cadena (Daisy-chain) es uno de los métodos disponibles para decidir las prioridades. Con este método todos los dispositivos que emiten petición de interrupción activan una única línea de interrupción hacia el procesador como se ilustra en la figura 6-4. El procesador, en este punto, detecta que hay una interrupción y empieza a procesar la interrupción después de completar la ejecución de la instrucción pendiente.

El procesador empieza la secuencia de interrupción emitiendo una señal de aceptación de interrupción que envia al primer periferico en la cadena de perifericos. Si este dispositivo causó la interrupción, previene que la señal de aceptación de interrupción pase al siguiente dispositivo y efectua la acción de interrupción poniendo su dirección de servicio de interrupción sobre el bus.
Con este método, el primer dispositivo en la cadena tiene la mayor prioridad, porque es el primero en tener acceso a la señal de aceptación si ocurren interrupciones múltiples por diferentes dispositivos.

Un segundo esquema de control de prioridades involucra líneas con prioridades de multinivel. Normalmente puede existir un circuito discreto o un circuito integrado que maneja varias líneas de interrupción (8 líneas en forma usual), llamado controlador de prioridad de interrupciones.

que trabaja con el procesador. Cada linea representa un nivel de interrupción.
La linea de interrupción con número de nivel bajo tiene más prioridad que las lineas con número más alto. El nivel de interrupción cero es normalmente el primero que obtiene respuesta.

Simplemente colocando los dispositivos en niveles de interrupciones diferentes, establecemos ordenes de servicios a interrupciones. La figura 6-4(b) muestra un sistema de interrupción de multinivel.

Asignar prioridades a los periféricos conlleva ciertos cuidados. Usualmente, los dispositivos rápidos que no pueden esperar que sus datos sean asignados, necesitan prioridades altas. Por ejemplo, una unidad de disco blando tiene una relación de flujo de datos relativamente alta, y por lo tanto debemos de asignarle un prioridad alta.

Si asignamos al disco una prioridad baja, hay posibilidad de perder información cuando el procesador esta procesando interrupciones emitidas por dispositivos lentos que podrían esperar.

VI-8 PROCESAMIENTO DE INTERRUPCIONES

Después que se acepta una interrupción y la ejecución del programa es dirigida a la rutina correspondiente, debemos seguir algunos pasos para asegurar que el programa interrumpido puede recuperar su estado original. Primero, la localidad donde el procesamiento fué interrumpido dentro del programa, debe guardarse, de tal manera que sea posible regresar a la localidad original. La lógica en la unidad de control del microprocesador, efectua esta función automaticamente cuando ocurre una interrupción. La dirección de "reingreso" usualmente se guarda la parte alta del stack.

Una manera rapida de explicar qué se está guardando en el stack cuando una interrupción se inicia, es la descripción de la instrucción "regreso de interrupcion", la cual debe ejecutarse al final de cada rutina de interrupción. Como un ejemplo veamos la instruccion RTI del microprocesador 6800.
La instrucción RTI consiste de un stack de 7 niveles en un apuntador de pila (STACK).
Un evento que ocurre por medio de hardware, durante la aceptación de una interrupción, es la colocación de una mascara (MASK). Si sucede más de una interrupción, y los bits de la mascara de interrupcion no estuvieran activos, el procesador inmediatamente seria interrumpido otra vez. Esto sucederia hasta la interrupción final. La figura 6-5 ilustra el concepto con respecto a un sistema de interrupciones.

Estructura del Bus (canal)

Un conjunto de líneas o conductores que comunican información entre los componentes en una computadora se llama Bus.
Si un bus conecta dos componentes menores dentro de un componente mayor, se llama bus interno. Cuando un Bus conecta dos componentes mayores, tal como el CPU y una interfaz, se llama bus externo.
La figura 6-6 ilustra la figura fundamental de un bus y su relación con varios componentes.

Configuración basica del 8086/8088

Para adaptarse a tantas situaciones como sea posible el 8086 y el 8088 tienen dos maneras de operar, modo mínimo y modo máximo. El modo mínimo se usa para sistemas pequeños con un solo procesador, un sistema en el cual 8086/8088 genera todas las señales de control al bus necesarias, por lo tanto minimiza la lógica de control. El modo máximo es para sistemas de tamaño medio y grande, los cuales frecuentemente usan dos o más procesadores. En el modo máximo, el 8086/8088 codifica las señales de control básicas del bus en una palabra de status de tres bits, y usa las señales de control restantes para proveer información adicional que se necesita para soportar una configuración de multiprocesador.

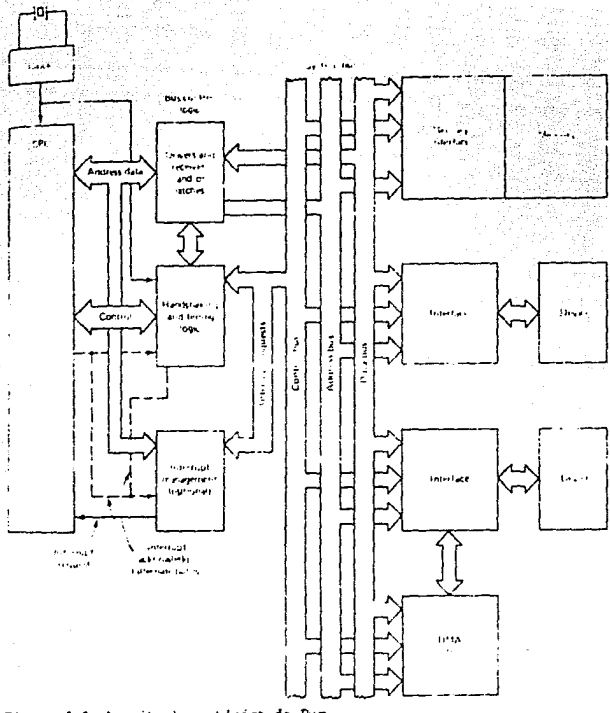


Figura 6-6. Arquitectura típica de Bus.

Los diagramas para el 80886 y 8088 se muestran en la figura 6-7. La patilla 33 (MN/ MX) determina la opcion de configuración. Cuando está a tierra se encuentra en modo máximo y cuando esta a +5V, en modo mínimo. Ambos procesadores multiplexan las señales de datos y direcciones y ambos tienen 20 patillas de direcciones. Sin embargo, ya que el 8088 puede transferir solamente 8 bits de datos al mismo tiempo, solamente 8 de sus patillas se usan para datos, lo contrario al 8086. Excepto por las patillas 28 y 34 los dos procesadores tienen la misma configuración.
En el 8086, la patilla 34 (BHE) denota si se efectuará o no la transferencia de menos de un byte sobre AD15 hasta AD8.
Un cero en esta patilla indica que se usarán las líneas de datos más significativas; de otra manera, solo se utiliza AD7 a AD0.
Las señales BHE y A0 indican a la interfaz conectada al Bus como aparecen los datos sobre el mismo.

Patillas	símbolo	E/S	Descrip.
1	tierra		Tierra
2-16	AD14-AD0	I/O	La dirección de salida durante la primera parte del ciclo de bus y acepta o saca datos durante la parte restante del ciclo de bus.
17	NMI	I	Peticiones de interrupción no-mascarables-flanco positivo.
18	INTR	I	Petición de interrupción mascarable.
19	CLK	I	Reloj- 33 % ciclo de trabajo: 5 MHZ para 8086 8 MHZ " 8086-2 10 MHZ " 8086-1
20	GND		Tierra
21	RESET	I	Termina actividad, borra PSW,IP,DS,SS,ES, y la cola de instrucciones., y coloca CS en FFFF. El procesamiento empieza en FFF0 cuando la señal baja. La señal debe ser 1 al menos 4 ciclos de reloj.
22	READY	I	Reconocimiento de memoria o interfaz de E/S que el CPU puede completar el ciclo de bus en proceso.
23	TEST		Usado junto con WAIT en un ambiente de multiprocesamiento.
24-31	-	-	Depende del modo.

Figura 6.7 Asignación de patillas.

32	RD	0-3	Indica que se efectuará una operación de E/S.
33	MN/MX	I	El CPU está en modo mínimo cuando se encuentra a +5V y en modo máximo si está a tierra.
34	BHE/S7	0-3	Si es 0 durante la primera parte del ciclo de bus esta patilla que al menos un byte de la transferencia en curso está a punto deefectuarse sobre las patillas AD15-AD8; si es 1 la transferencia se hace sobre AD7-AD0. El status S7 es disponible durante durante la ultima parte del ciclo del bus, pero mientras S7 no tiene un significado asignado.
35-38	A19/S6-A16/S3	0-3	Durante la primera del ciclo del bus los 4 bits superiores de la dirección son sacados y durante el resto del ciclo del bus es sacado el status. S3 y S4 indican el registro del segmento que esta siendo usado como sigue S3 S4 Registro 0 0 ES 0 1 SS 1 0 CS o nada 1 1 DS S5 da la configuracion de IF S6 es siempre 0

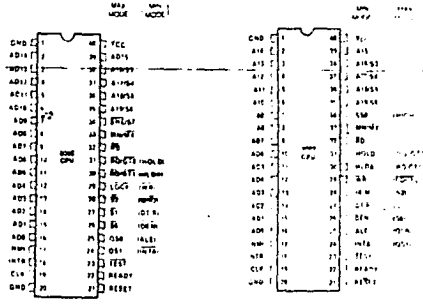
39		I/O-3	Lo mismo que AD14-AD0.
40	Vcc	-	Voltaje de alimentación de +5V + 10%.

1 En el 8088, AD15-AD8 son A15-A8 y son solamente para sacar bits de direcciones.

2 5 Mhz para el 8088, y 8 MHZ para el 8088-2.

3 En el 8088, esta patilla se denota por SSO y se usa en el modo mínimo para denotar status. Logicamente equivalente a S0. Es siempre 1 en modo máximo.

c) Definicion de patillas.



(a) 8086 pin diagram (b) 8088 pin diagram

Ya que el 8088, solamente puede transferir datos sobre AD7-AD0, esta patilla no se necesita para indicar la parte alta o baja del bus de datos, y está disponible para proveer información de status.
Las patillas 1 y 20 están a tierra. Las patillas del 2 al 16 y 39 (AD15-AD0) contienen la dirección necesaria para la transferencia durante la primera parte del ciclo del bus, y quedan libres para transferir los datos durante el resto del ciclo.
Las patillas 17 y 18 (NMI e INTR) son para requerimientos de interrupción. La patilla 19 (CLK) es el reloj procesador.
La patilla 21 (RESET) es para reiniciar el sistema.
La patilla 22 (READY) es para aceptar un reconocimiento de memoria o de alguna interfaz de E/S donde los datos de entrada se pondrán sobre el bus de datos o los datos de salida serán aceptados del bus de datos dentro del siguiente ciclo de reloj.
en cualquier caso, el procesador y su lógica de control del bus pueden completar el ciclo del bus en ejecución después del siguiente ciclo de reloj.
la patilla 23 (TEST) se usa en conjunto con la instrucción WAIT y se emplea principalmente en situaciones de multiprocesamiento.

VI-9. MODO MINIMO

Un procesador está en modo mínimo cuando su patilla MN/MX se encuentra a +5V. Las definiciones para las patillas 24 hasta 31 se ilustran en la figura 6-8 y un sistema típico están en la figura 6-9.

Las direcciones deben detenerse ya que están disponibles solamente durante la primera parte del ciclo del bus. Para señalar que la dirección está lista para detenerse se pone un "1" lógico en la patilla 25, la patilla de habilitación de dirección (ALE). Típicamente, la detención se efectua usando el microcircuito INTEL 8282, como se muestra en la figura 6-10. Ya que 8282 es un detenedor de 8 bits, se requieren dos para las direcciones de 16 bits. En un sistema 8086, BHE también deberá ser detenido.
Para un sistema pequeño que solo tiene 64 KB de memoria se requieren dos 8282. Una señal aplicada en la patilla STB controla los bits aplicados a la entrada a las líneas D17-D10. Por consiguiente, STB se conecta a la patilla ALE del 8086 y se unen D17-D10 a ocho de las líneas de dirección. Una señal baja activa en OE habilita las salidas D07-D00, y un 1 en esta patilla obliga a las salidas a un estado de alta impedancia.
En un sistema mínimo las líneas de control no necesitan pasar a través de transreceptores, sino que pueden usarse directamente. Las líneas M/IO, RD, y WR especifican el tipo de transferencia de acuerdo con :

M/IO	RD	WR	
0	0	1	E/S Lectura
0	1	0	E/S Escritura
1	0	1	Lectura a memoria
1	1	0	Escritura a memoria

VI-10. MODO MAXIMO

Un procesador esta en modo maximo cuando su patilla MN/MX está a tierra. Las definiciones del modo máximo de las patillas 24 al la 31 están en la figura 6-13 y una configuración de modo máximo típica se muestra en la figura 6-14. Es claro que la diferencia entre modo máximo y modo mínimo es la necesidad de circuitos adicionales para traducir las señales de control.
Este circuito es para convertir los bits de status S0, S1, y S2 a las señales de transferencia de E/S y memoria necesarios para transferencia de datos directa, y para controlar los detenedores 8282 y los transreceptores 8286. Normalmente se implementa con un controlador de bus INTEL 8288.

VI-11. MANEJO DE PRIORIDADES DE INTERRUPCION

El manejo de las interrupciones se puede efectuar por varios métodos, desde la técnica de programación pura, a la configuración de interrupciones en cadena, pero si se desea una configuración de mayor eficiencia, debemos usar más circuitos.

Existen muchos circuitos integrados comerciales que efectuan el manejo de prioridades de interrupción, sin embargo, evitamos problemas de compatibilidad, si los dipositivo pertenecen al mismo fabricante.
Trataremos el dispositivo Intel 8259A controlador de interrupciones programable, que ha sido diseñado especificamente para trabajar con el 8086/8088.

Sistemas de Interrupcion Basado en el 8259A

El 8259A está contenido en un circuito integrado de 28 patillas que requiere solamente +5V.
Para un 8088 las dos direcciones asociadas con un 8259A son normalmente consecutivas, y la línea A0 se conecta a la patilla A0, pero como hay solamente 8 líneas de datos en el 8259A y el 8086 siempre acepta la interrupcion de la parte baja del bus de 16 bits, todas las transferencias de datos desde y hacia el 8259A deben de hacerse sobre el byte bajo del bus. La manera más fácil de garantizar que todas las

transferencias usaran la parte baja del bus, es conectar la línea A1 a A0 y usar dos direcciones pares consecutivas con la primera divisible entre 4. Sin embargo para simplificar nos referiremos, a la segunda dirección como dirección impar en ambos casos.

La porción de control del 8259A contiene varios bits programables que pueden ser vistos como contenidos en siete registros de 8 bits. Estos registros se dividen en dos grupos, con uno conteniendo los comandos de inicialización (ICW) y el otro grupo conteniendo los comandos de operación (OCW). Los comandos de inicialización normalmente actuan como una rutina cuando la computadora arranca y permanecen activados durante su operación.
El registro para requerimiento de interrupción (IRR), el discriminador de prioridades, y el ISR (registro de servicio) reciben y controlan las interrupciones que llegan a las patillas IR7-IR0. El IRR detiene peticiones y, en conjunto con el discriminador de prioridades, permite colocar un "1" en la patilla INT, a las peticiones no mascarables con suficiente prioridad. La logica del discriminador de prioridades determina las prioridades de las peticiones en el IRR y el ISR es para detener las peticiones que están siendo procesadas.
Después de colocar un bit en "1" en el IRR, se compara con la mascara de bit correspondiente en el IMR(registro de mascara para interrupción). Si el bit de mascara es 0, la petición se pasa al discriminador de prioridades, pero si es 1, la petición se bloquea.
Cuando una petición de interrupción entra al discriminador de prioridades, éste examina su prioridad y, de acuerdo al estado actual del discriminador de prioridades, la interrupción se envia al CPU. Se activa la línea INT.

Patilla	simbolo	E/S 3-ST	Descrip.
24	INTA	0-3	Indica reconocimiento de una petición de una interrrupción. Consiste de una serie de pulsos de flanco negativo en dos ciclos de bus consecutivos.
25	ALE	0	Tiene como salida un pulso al principio del ciclo del bus y es para indicar que una dirección está disponible en las patillas de direcciones.
26	DEN	0	Activa durante la ultima parte del ciclo de bus y es para informar a los transreceptores que el CPU está listo para enviar ó recibir datos.
27	DT/R	0-3	Indica al conjunto de transreceptores si ellos transmiten o reciben datos.
28	M/IO	0-3	Distingue una transferencia de memoria de una transferencia de E/S. Para una transferencia de memoria es 1.
29	WR	0-3	Cuando 0, indica que se esta ejecutando una operación de escritura. Se usa en conjunto con 28(M/IO) y 32 (RD) para especificar el tipo de transferencia.
30	HDLA	0	Tiene como salida un "bus grant" a una petición de maestro (master). Las patillas con compuertas tres estados se ponen en alta impedancia, mientras HLDA=1.
31	HOLD	I	Recibe una peticion de bus del master. El 8088/80886 no ganará el control del bus hasta que esta señal es liberada.

1. Para el 8088, el simbolo es IO/M y un 1, indica una transferencia de E/S.

Figura 6-8 Definicion de patillas para modo minimo

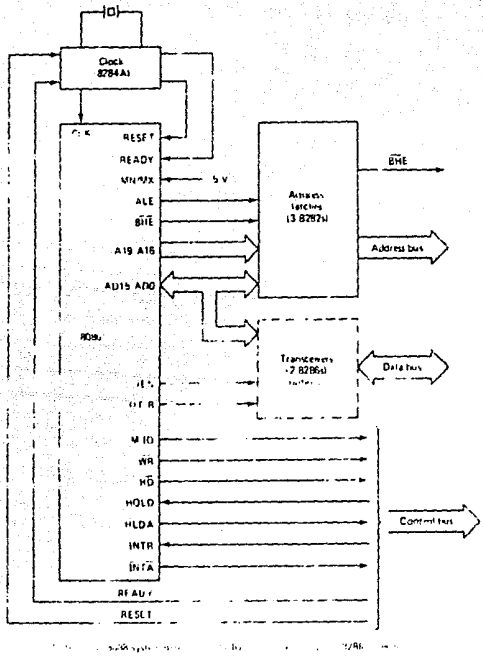


Figura 6-9. Sistema en modo mínimo

Si suponemos que la bandera IF en el CPU es "1" el CPU iniciará su secuencia de interrupción al completar su instrucción y regresará dos pulsos negativos sobre la línea INTA. A la llegada del primer pulso, los detenedores de IRR se deshabilitan de tal manera que el IRR ignorará señales posteriores en las líneas IR7-IR0. Este estado se mantiene hasta el final del segundo pulso INTA. También, el primer pulso obliga a que el bit adecuado de ISR se ponga en nivel alto y el bit de IRR sea puesto a cero.

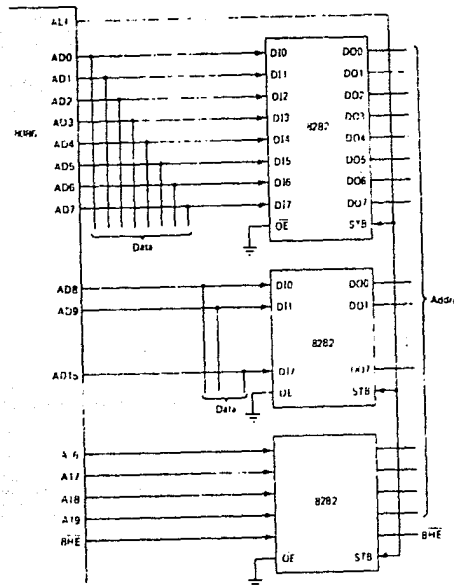


Figura 6-10. Aplicación de latches 8282.

El segundo pulso INTA causa que los contenidos actuales de ICW2 sean colocado en D0-D7, y el CPU usa este byte como el tipo de interrupción. Si el bit de fin automático de interrupción (AEOI) en ICW4 es "1", al final del segundo pulso INTA el bit de ISR, el cual fue puesto por el primer pulso de INTA, se limpia, de otra manera el bit de ISR no se borra hasta que se envia el fin de interrupción apropiado a OCW2.
Como se indicó anteriormente, los comandos de inicialización normalmente son ejecutados por una rutina de inicialización cuando se enciende el sistema y contiene los bits de control que se mantienen constantes durante la operación del sistema. El 8259A tiene una dirección par (A=0) y una impar (A=1) asociada y el comando de inicialización debe ejecutarse sucesivamente usando la dirección par ICW1 y la impar para las ICW's restantes.

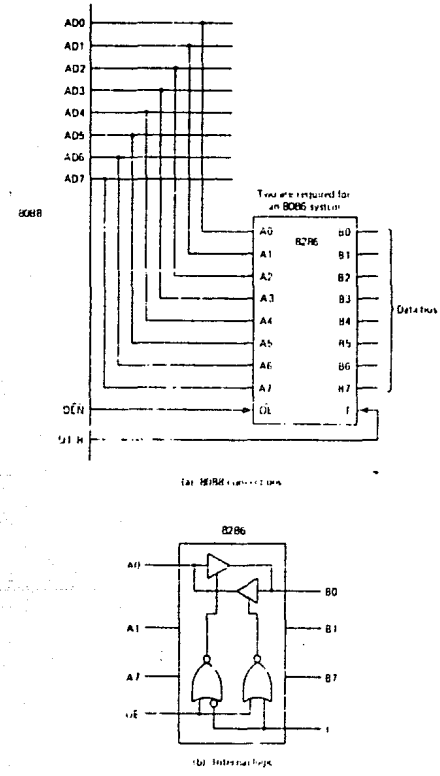


Figura 6-11. Logica interna y aplicaciòn del 8288.

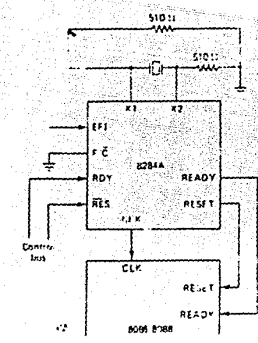


Figura 6-12. Configuracion tipica del 8284A.

Un programa tipico para establecer los contenidos de los ICW, el cual asume que la direccion del 8259A es 0080, es :

```
MOV AL,13H
OUT 80H,AL
MOV AL,18H
OUT 81H,AL
MOV AL,0dH
MOV 81H,AL
```

Las primeras dos instrucciones causan que las peticiones de interrupcion sean disparadas por flanco, denotan que solamente se usa un 8259A, además informan al 8259A que saldrá un ICW4. Las dos instrucciones siguientes causan que los 5 bits más significativos del tipo de interrupción se pongan en 00011.

ICW3 no es salida porque SNGL=1; por consiguiente, las dos últimas instrucciones colocan ICW4 a 0D, que informa al 8259A que el modo de anidación especial no será utilizado, el SP/EN se usa para deshabilitar los transreceptores, si el 8259A es el dispositivo maestro, debemos usar instrucciones EOI para borrar el bit ISR. si el 8259A es parte de un sistema 8086/8088.

Hay tres formas de OCW3. Se usa OCW1 para mascarar la petición de interrupción; cuando el bit de la mascara correspondiente a una petición de interrupción es "1"; entonces la petición se congela. OCW2 y OCW3 controlan el modo del 8259A y la recepción del comando EOI. La forma de colocar un byte en OCW1 es usando las direcciones impares asociadas al 8259A y hacia el OCW2 y OCW3 en los demás pares. OCW2 se distingue de OCW3 por el contenido del bit 3 del byte de datos. Si el bit 3 es "0", el byte se dirige hacia OCW2, y si es "1", es para OCW3. La diferencia entre OCW2 y OCW3 y OCW1 es el bit 4 de los datos, aunque todos

usan la direccion par. Si el bit 4 es "0", entonces el byte es para OCW2 o OCW3 conforme el bit 3 del byte de datos. No hay confusión en ICW2, ICW3, ICW4, y OCW1 cuando todos usan la direccion impar ya que los comandos de inicializacion deben seguir a ICW1 según la secuencia de arranque, y no existe ninguna salida hacia OCW1 durante esta secuencia.

Con referencia a la figura 6-15, los bits L2-L0 de OCW2 designan un nivel de ISR, el bit 5 es para los comandos de EOI, bits 6 y 7 controlan los niveles IR. Cuando el bit AEOI en ICW4 es "1", el bit ISR, el cual se coloca por medio de la peticion de interrupcion, se borra en forma automatica al final del segundo pulso INT, pero si AEOI=0, entonces el bit ISR debe borrarse directamente por un comando EOI, que consiste en enviar un OCW2 con el bit 5 igual a "1". Cuando enviamos un comando EOI el significado de las 4 posibles combinaciones del bit 7, el bit R (rotar), el bit 6 o bit L (colocar nivel), son :

R	SL	
0	0	no especificado, modo prioridad normal
0	1	especificamente borra el bit ISR indicado por L2-L0 de haber sido atendido tiene la prioridad mas baja.
1	0	rotar las prioridades de tal manera que un dispositivo despues de haber sido atendido tenga la prioridad mas baja
1	1	rotar la prioridad hasta que la posición especificada por L2-L0 sea la menor.

Patilla	simbolo	E/S 3-ST	Descrip.
24,25	QS1,QS0	0	Releja el status de la cola de instrucciones. Este status indica la actividad en la cola de espera durante el ciclo de reloj anterior.
26,27,28	S0,S1,S2	0-3	Indica el tipo de transferencia a efectuar durante el ciclo de bus.

S2	S1	S0	
0	0	0	Reconocimiento de interrupción
0	0	1	Lectura de E/S
0	1	0	Escrit. de E/S
0	1	1	Alto
1	0	0	Fetch de instr.
1	0	1	Lect. a memoria
1	1	0	Escrit a mem.
1	1	1	inactivo.

29	LOCK	0-3	Indica que el bus no será cedido a otros masters potenciales. Es iniciado por un prefijo de la instruccion LOCK y es mantenido hasta el final de la siguiente instrucción. También está activo durante y entre los dos pulsos de INTA.
30	RQ/T1	I/O	Para aceptar peticiones de bus y efectuar "bus grant".
31	RQ/GT0	I/O	Igual a la anterior solo que RG/GT0 tiene mayor prioridad.

Nota: en modo maximo el 8086 y el 8088 tienen la misma definición de patillas excepto la patilla 34, la cual en el 8088 es siempre 1.

Figura 6-13. Definición de patillas en modo máximo.

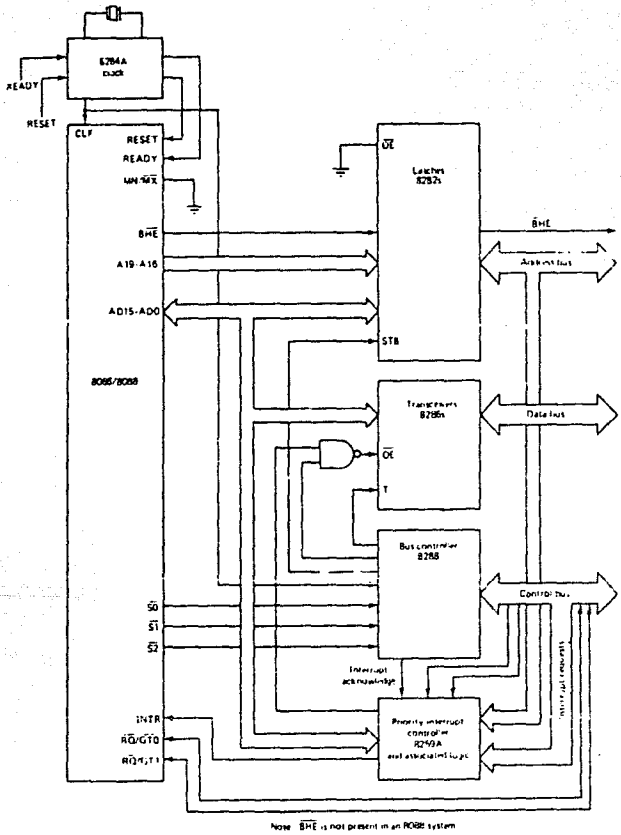


Figura 6-14. Configuración en modo máximo típica.

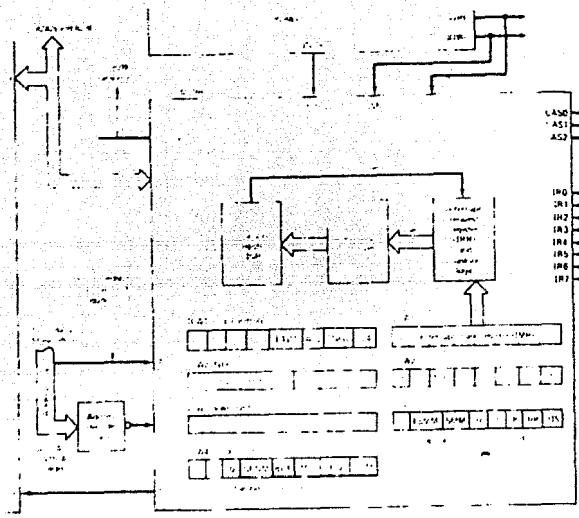


Figura 6-15 Organización del contador de interrupciones programable.

Los bits en OCW2 son retenidos temporalmente por el 8259A hasta que se efectuan las acciones contenidas en ellos. Lo anterior es particularmente importante con respecto al bit EOI.
Normalmente, una petición en IR0 tiene la prioridad más alta, y sobre IR1 tiene la siguiente prioridad más alta, y así sucesivamente. Cuando llega el primer pulso INTA el discriminador de prioridad permite que solamente las peticiones no-mascaradas con las más altas prioridades coloquen su bit ISR. Ya que los tres bits menos significativos de ICW2, donde ICW2 indica el tipo de interrupcion y determina la dirección del apuntador de interrupcion, se definen dependiendo de cuales bits ISR están prendidos. Por consiguiente, la rutina de interrupción asociada con el dispositivo conectado a la patilla de más alta prioridad IR es la primera en iniciar y las otras peticiones deben esperar hasta que se permitan interrupciones posteriores.
Bajo el modo de prioridad normal. Si el enesimo ISRn se prende, el discriminador de prioridades no reconocera

ninguna peticion sobre IR7 hasta IR(n+1), pero reconocera peticiones no-mascarables en IR(n-1) hasta IR0. Por consiguiente si la bandera en el CPU está en "1", las peticiones que tengan mayor prioridad que las que están siendo procesadas pueden interrumpir la rutina en proceso mientras que aquellas de menor prioridad esperan su turno. Las peticiones de menor prioridad se procesan de acuerdo a su prioridad, conforme el bit ISR se borra. Si AEOI= 1, el bit ISR correspondiente a una interrupcion se borra automáticamente al final del segundo pulso INTA. Cuando AEOI= 0 el bit ISR debe borrarse o limpiarse por medio de la rutina de interrupción poniendo en "1" el bit 5 de OCW2.

VII- INTERFAZ ENTRADA/SALIDA

VII-1 EL DISEÑO DE UNA INTERFAZ E/S

Las interfaces de memoria y de entrada/salida son la contraparte de la lógica de control al bus. Lo que existe entre la lógica de control del bus y la interfaz son solamente los conductores en el bus. Por consiguiente, la interfaz debe diseñarse para aceptar y enviar señales que son compatibles con la lógica de control del bus y su asignación de tiempos.

Una interfaz de E/S debe ser capaz de :

1- Interpretar las señales de selección de entrada/salida y direcciones para determinar sus características y saber que registros están siendo accesados.

2- Determinar si es una entrada o salida la seleccionada, y aceptar datos de salida o información de control del bus. Colocar datos de entrada o información de status sobre el bus.

3- Aceptar datos o enviar datos a un dispositivo periférico de E/S y convertir los datos de forma paralela a un formato aceptable para el dispositivo y vice versa.

4- Enviar una señal de reconocimiento (READY) cuando los datos han sido aceptados de o hacia el bus de datos, informando al procesador que se ha completado una transferencia.

5- Enviar peticiones de interrupción, si no existe manejo de prioridad de interrupción en la lógica de control del bus, recibir reconocimientos de interrupción y enviar un tipo de interrupción.

6- Recibir una señal de reinicio y efectuar el arranque por si misma, y tal vez su dispositivo asociado.

La figura 7-1 contiene un diagrama de bloque de una interfaz típica de E/S. La función de una interfaz típica es esencialmente traducir las señales entre el bus del sistema y el dispositivo de E/S y proveer los detenedores necesarios para satisfacer los dos conjuntos de restricciones de tiempos. La mayor parte del trabajo hecho por la interfaz se efectua por el bloque en la parte derecha de la figura. Normalmente este bloque se implementa en un solo circuito integrado, pero su función podría ser dividida entre muchos dispositivos. Claramente su función varia radicalmente, dependiendo del dispositivo de entrada/salida con el cual se comunica.
Una interfaz se divide en dos partes, una parte que sirve de interfaz al dispositivo de entrada/salida y una parte que

sirve de interfaz al bus del sistema. Aunque poco puede decirse del dispositivo de entrada/salida sin conocer sus cualidades.

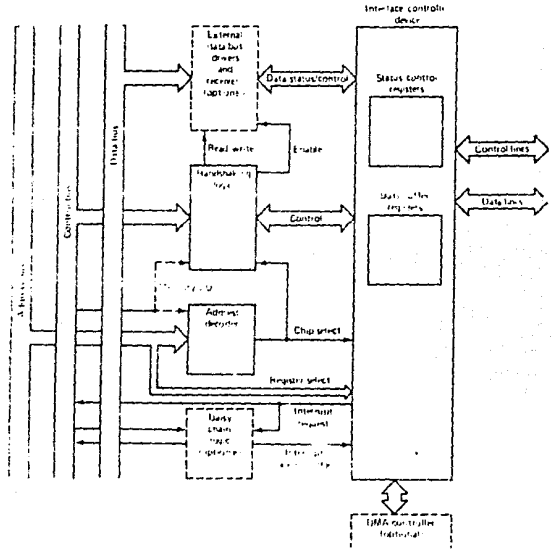


Figura 7-1. Diagrama típico de una interfaz de E/S.

Para soportar la lógica de la interfaz principal debe haber manejadores de bus de datos y receptores, lógica para traducir las señales de control de la interfaz en señales de protocolo adecuadas, y la lógica para decodificar las direcciones que aparecen en el bus.

En sistema 8086/8088, los transreceptores 8286 podrían manejar el bus de datos, tal y como son usados para manejar el bus y su lógica de control. Sin embargo, los dispositivos de la interfaz principal pueden contener receptores y transmisores internos que son suficientes para sistemas independientes.

La lógica de protocolo no puede diseñarse hasta que se conocen las señales de control necesarias para el dispositivo de la interfaz principal, y estas señales pueden variar de una interfaz a otra. Típicamente, esta lógica debe

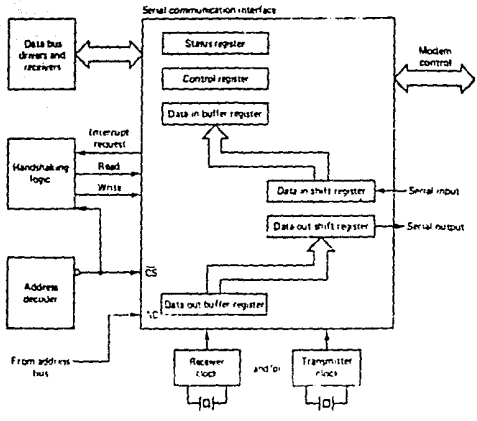


Figura 7-2. Interfaz serie.

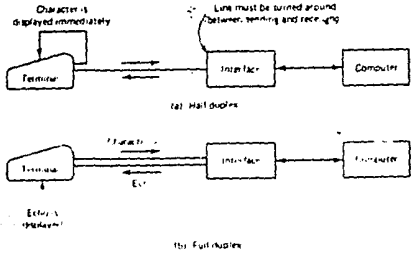


Figura 7-3. Modos de transmision basicos.

aceptar señales de READ/WRITE para determinar la direccion de los datos y usar las señales OE y T requeridas por el 8286.

En modo máximo el sistema 8086/8088 recibiria las señales IOWC (AIOWC) y IORC del controlador de bus 8288 y en modo mínimo recibiria las señales RD, WR y M/IO (o IO/M). Las peticiones de interrupción, las lineas de READY y RESET también se incluyen en la lógica.

El decodificador de dirección debe recibir la dirección y, quizas, un bit indicando si la dirección está en el espacio de E/S en memoria. En el modo mínimo este bit lo podriamos obtener de la linea M/IO.

Pero en modo máximo la determinación entre memoria y E/S se obtiene a partir de IOWC e IORC. Si el decodificador determina que su interfaz está siendo seleccionada, entonces el decodificador debe enviar señales al dispositivo principal indicando que ha sido seleccionado y cual registro es accesado. Los bits que designan el registro pueden ser la parte baja de la dirección, pero frecuentemente se generan dentro de la interfaz del dispositivo a partir de las señales de control READ/WRITE así como las señales de direcciones.

Por ejemplo, si hay dos registros A y B de lectura y dos registros C y D de escritura, entonces las señales read y write y el bit 0 del bus de direcciones podrian usarse para especificar el registro como sigue :

Write	Read	Bit0	Reg. Accesado
0	1	0	A
0	1	1	B
1	0	0	C
1	0	1	D

Si incluimos un sistema de interrupción en cadena en lugar de un dispositivo para manejo de prioridades, entonces cada interfaz debe contener lógica para interrupción en cadena, tal como se muestra en la figura 7-2b, y debe incluir lógica para generar el tipo de interrupción.
También la interfaz debe estar relacionada con un controlador de DMA.

Muchas interfaces las designamos para detectar al menos dos tipos de errores. Ya que las lineas que conectan una interfaz a su dispositivo estan casi siempre sujetas a ruido, se agregan bits de paridad a los bytes de información durante la transmisión. Si la paridad es par el bit de

paridad es "1" de tal manera que el numero total de unos (1's), incluyendo el bit de paridad, es par. Para paridad impar el numero total de unos es impar. Conforme se efectua la recepción de los bytes el sistema verifica la paridad, si existe un error, se utiliza un bit de status en un registro de status.
El otro tipo de error que la mayoria de las interfaces pueden detectar se llama error de sobre-ejecución. Como hemos visto, cuando una computadora acepta datos los trae de un registro de contenedores (buffers). Si, por alguna causa, los contenidos de este registro se reemplazan por nuevos datos antes de entrar a la computadora, ocurre un error de sobre-ejecucion. Como en el caso del error de paridad, el error de sobre-ejecución causa que coloque cierto bit de status.

VII-2 INTERFAZ DE COMUNICACION SERIE

Interfaz de Comunicacion Programable

Como un ejemplo de un dispositivo de interfaz serie consideremos el 8251A de Intel llamado Interfaz programable de comunicación. El diagrama se muestra en la figura 7-3.
Es capaz de programarse para comunicación asincrona y sincrona. Los registros para contenedores de entrada y salida comparten la misma dirección de puerto. Para la entrada, el flujo de bits en forma serie llega a la patilla RxD y es aceptada por el registro de corrimiento y los datos se transfieren a los registros contenedores de entrada, donde pueden entrar al CPU. Contrariamente, una salida de datos pone los datos en el registro contenedor de salida y son transferidos al registro de corrimiento transmisor y, con los bits de sincronia necesarios, son enviados por la patilla TxD.

Entre otras cosas los contenidos del registro de "modo", que es inicializado por el programa en ejecucion, determinan si el 8251A esta en modo asincrono o sincrono y el formato de los caracteres que se reciben y transmiten.

El registro de control, que tambien lo configura el programa, controla la operacion de la interfaz, y el registro de status proveen cierta información al programa en ejecución. Por supuesto, los registros de sincronia de la palabra SYNC son para almacenar los caracteres SYNC necesarios para la comunicación sincrona.

Aunque el procesador puede accesar los siete registros del lado izquierdo de la figura 7-3, el 8251A esta asociado con solamente dos direcciones de puertos.

La patilla C/D esta a la linea de direcciones A0 y tambien A0 diferencia las dos direcciones de puertos. El 8251A internamente interpreta las señales C/D, R/D y WR como sigue:

C/D(=A0)	RD	WR	
0	0	1	Datos al contenedor de entrada
0	1	0	Datos al contenedor de salida
1	0	1	El registro de status se coloca en el bus de datos
1	1	0	El bus de datos esta en modo de control, registro de caracter Sync.

Donde "1" significa que la patilla está en estado alto y "0" significa estado bajo. Cualquier otra combinación causa que las patillas D7-D0 se pongan en tercer estado o de alta impedancia.

La selección del registro de modo, el de control o el registro de sincronia del caracter sync depende de la secuencia de acceso. Mostramos un diagrama de flujo en la figura 7-4. Después de una señal de reeinicio por hardware o una instrucción con el bit de reinicio en "1", la siguiente salida con A0=1 (por ejemplo, con C/D=1, RD=1 y WR=0) se dirige al registro de modo. Los formatos del registro de modo para los casos asincronos y sincronos se definen en la figura 7-5. Si los dos bytes menos significativos (BmS) del "modo" son cero, entonces la interfaz se pone en modo sincrono y el byte más significativo determina el numero de caracteres de sync. En el modo sincrono, la siguiente salida de 1 ó 2 bytes con A0=1 se convierte en caracteres de sync. Si los dos bytes menos significativos del modo no son cero, entonces el 8251A entra en modo asincrono. En cualquier caso, todos los bytes subsecuentes anterior a otro reinicio se dirige hacia el registro de control si A0=1 y hacia el registro de control de salida si A0=0.
En el modo sincrono las proporciones de baudaje del transmisor y receptor, que son las proporciones de los registros de corrimiento, son las mismas que las frecuencias de las señales aplicadas a TxC y RxC, respectivamente, pero en el modo asincrono las tres posibles combinaciones restantes para los dos bytes menos significativos (BmS) en el registro de modo definen el factor de baudaje. La relación entre las frecuencias del reloj de TxC y RxC y el baudaje del transmisor y receptor es:

Frecuencia de Reloj =Factor de proporcion de baudaje * proporción de Baudaje

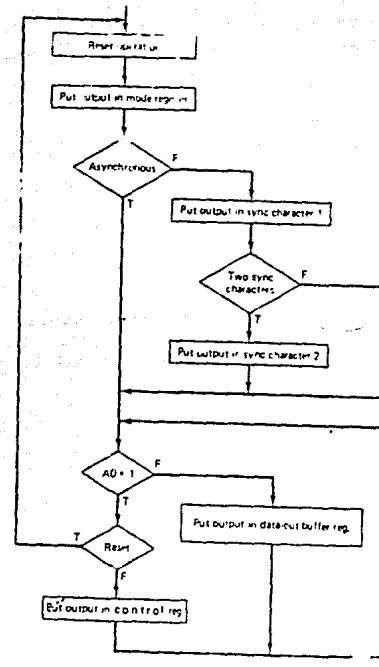


Figura 7-4. Diagrama de flujo de la salida.

Si los dos Bms son 10 en el registro de modo y las proporciones de baudaje en el transmisor y receptor son 300 y 1200, respectivamente, entonces la frecuencia aplicada a TxC debería ser 4800 Hz y la frecuencia en RxC debería ser 19.2 KHz.
En ambos modos, síncrono y asíncrono, los bits 2 y 3 indican el numero de bits de datos en cada caracter, el bit 4 indica si hay o no bit de paridad , y el bit 5 especifica el tipo de paridad (par o impar).
Para el modo asíncrono los dos BmS indican el número de bits de parada, pero para el modo síncrono el bit 6 determina si la patilla SYNDET se usa como entrada o como salida, y el bit 7 indica el número de caracteres syn.

Si la patilla SYNDET la usamos como salida se activa, cuando se encuentra una coincidencia bit a bit entre el flujo de entrada y los caracteres sync.

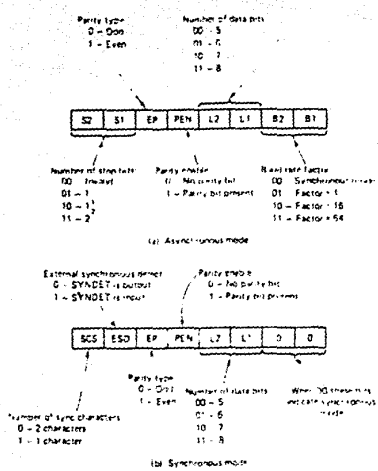


Figura 7-5. Formato del registro de modo.

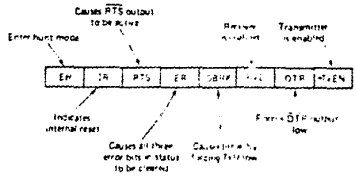


Figura 7-6. Formato del registro de control.

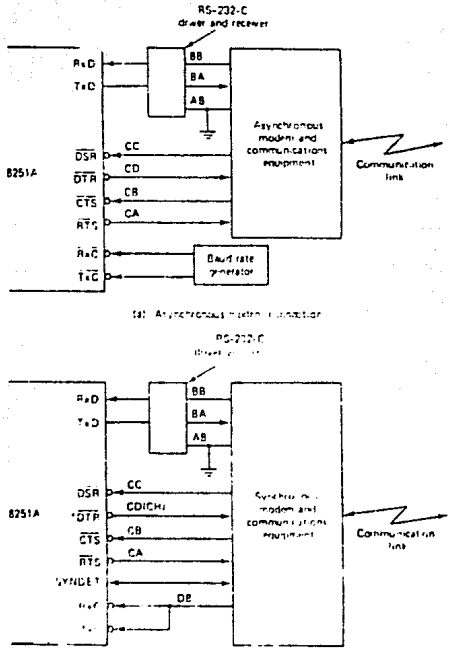


Figura 7-7. Conexion del 8251A como Modem.

Si la busqueda de los caracteres sync se efectua por un dispositivo externo, entonces SYNDET puede usarse como señal de entrada, indicando que se ha encontrado una coincidencia por el dispositivo externo. Esta patilla también tiene un significado durante la operación asíncrona, pero en este caso, solo puede ser una salida. Esta salida se llama la señal para detección de ruptura y se pone en alto cuando recibe un caracter de solamente 0's.

El formato para el registro de control se muestra en la figura 7-6. El bit 0 de este registro debe ser 1 antes de que los datos puedan ser recibidos.
La auto-respuesta de un modem se efectua poniendo el bit 1 a "1" y esto obliga a la pantalla DTR a "0" y el complemento de DTR esta normalmente conectado a la linea CD desde el modem.
El bit 3 en "1" obliga a TxF a cero, esto causa que se transmitan señales de deteccion. Al poner el bit 4 a "1" causa que todos los errores de bits en el registro de status sean borrados. El bit 5 se usa para enviar una señal " Request to Send" al modem (requerimiento de envio).

Si el complemento de la patilla RTS se conecta a la linea CA del modem, entonces un "1" en el bit 5 causará que la linea tome un estado alto "1". El bit 6 en "1" provoca el reinicio del 8251A. El bit 7 solamente se utiliza con el modo sincrono.
Un diagrama tipico se muestra en la figura 7-7. Con respecto a la conexion asincrona, suponemos que la asignación de tiempos es controlada por el modem y los equipos respectivos.
También, si el equipo se usa para detectar los caracteres de sincronia *syn* al inicio del mensaje recibido, entonces podemos informar al 8251A por medio de la patilla SYNDET.
Por otro lado, si el 8251A busca el caracter de sincronia syn puede usar la señal SYNDET como salida para decirle al modem que se encontró el caracter syn.
Una secuencia de programación que inicializa el modo de registro y dá una instrucción para habilitar el transmisor y empezar una transmisión asincrona de 7 bits, paridad par y dos bits de parada es :

```
MOV AL, 11111010 B
OUT 51H,AL
MOV AL, 00110011 B
OUT 51H,AL
```

Esta secuencia asume que los registros de control y modo están en la direccion 51H y las frecuencias de reloj son 16 veces las correspondientes al baudaje. La secuencia siguiente :

```
MOV A1,00111000B
OUT 51H,AL
MOV AL, 16
OUT 51H,AL
OUT 51H,AL
MOV AL,10010100B
OUT 51H,AL
```

pone al 8251A en modo sincrono y empieza a buscar dos caracteres ASCII sync. Como antes, los caracteres consisten de 7 bits y un bit de paridad par, pero no habrá, obviamente ... de parada.

El formato del registro de status está en la figura 7-8. Los bits 1 y 2, y 6 reflejan la señal en RxRDY, TxE y las patillas *SYNDT*. TxRDY indica que el buffer de salida está vacio. A diferencia de la patilla TxRDY, este bit no es afectado por la patilla de entrada CTS o el bit de control TxEN.

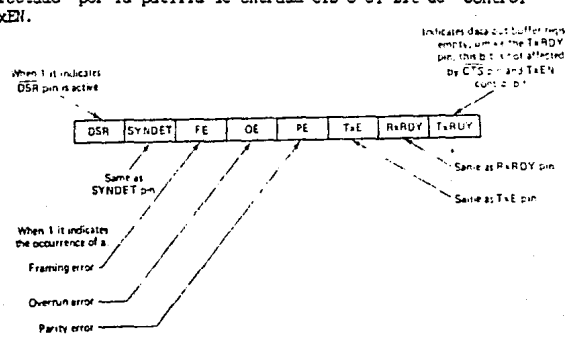


figura 7-8 Formato del registro de status.

RxRDY indica que se ha recibido un caracter y está listo para entrar al procesador. Los bits TxRDY y RXDY se pueden usar para E/S programados o las señales sobre las patillas correspondientes conectadas a las líneas de petición de interrupción para proveer interrupción de E/S.

El bit TxRDY es automáticamente borrado cuando un caracter está disponible para transmisión y el bit RxRDY automáticamente se borra cuando el caracter que lo causó entra al procesador. El bit 2 indica que el registro de corrimiento transmisor está esperando por enviar un caracter del registro de salida. Durante las transmisiones sincronas, mientras este bit este en "1", el transmisor llevará los datos desde el registro de caracter syn hasta que los datos sean puestos en el registro de salida. Los bits 3,4, y 5 indican paridad, sobre-ejecución, y error de protocolo, respectivamente.

Cuando se detecta un error, el bit correspondiente al tipo de error se pondra en "1". Si el complemento de la patilla DSR se conecta a la línea "Data Set Ready" (cc), entonces el bit 7 refleja el estado del modem y es "1" cuando el modem se enciende.

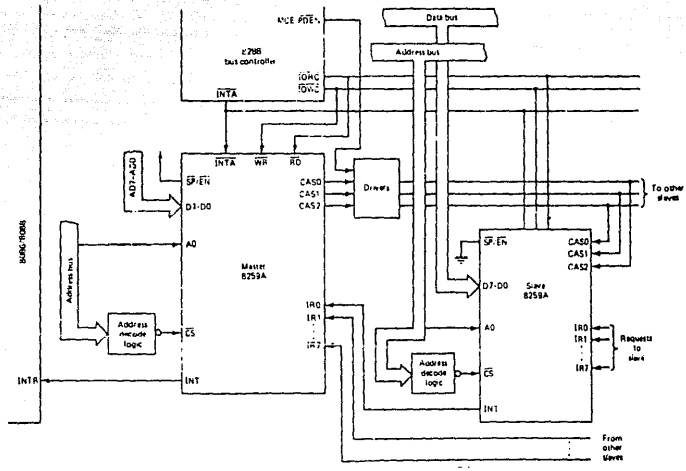


Figura 7-9. Sistema de interrupción con 8259A multiple.

La figura 7-9 da una secuencia tipica donde hay un puerto de E/S del 8251A programado a 80 caracteres, con los registros de datos en la dirección 0050, que pone los datos que lleguen en el buffer de memoria empezando en LINE. El lazo interno prueba continuamente el bit RxRDY hasta que se activa por un caracter en el registro de entrada. Entonces el nuevo caracter es llevado al detenedor (buffer) y los bits de error son checados.

```
        MOV AL,00110101B    ; Habilita transmisor y
                            ; receptor
        MOV 51H, AL         ; Borra bits de error
        MOV DI,0            ; Inicializa indice
        MOV Cx,80           ; Poner cuenta en cx
BEGIN:  In AL,51H           ; Esperar por entrada
        Test AL,02H         ;
        Jz BEGIN            ;
        In AL,50H           ; Entrada al caracter o dato
     MOV LINE[DI],AL        ; Poner en el buffer LINE
         Inc DI             ;
         In AL,51H          ; Checar bit de error
   Test AL,00111000B        ;  Y
         Jnz Error          ; Si no hay error
        loop BEGIN          ; Continua
     Jmp Short Exit         ;
Error: CALL NEAR PTR ERR_ROUT; Llama a la rutina ERR_ROUT
```

Entonces una vez que la entrada llega a su fin, efectuamos una llamada a la rutina de error que examinará los bits de error, imprimiendo un mensaje.

VII-3 LA COMUNICACION PARALELA

La comunicacion paralela se efectua simultaneamente transfiriendo bits sobre lineas separadas. Su ventaja sobre la comunicación serie, es que es posible obtener velocidades de transferencia mayores, con la desventaja del alto costo, por el número excesivo de lineas, ademas del ruido inducido sobre el sistema. La comunicación paralela se utiliza sobre grandes distancias solo si son necesarias altas velocidades de transferencia.

A diferencia de la comunicación serie, la comunicación paralela no ha sido estándar, la interfaz paralela puede efectuarse simplemente poniendo los datos en el registro de entrada de la interfaz o sacando los datos de la interfaz.

VII-4 EL 8255 INTERFAZ PARALELA PROGRAMABLE

El 8255 es un dispositivo de E/S de propósito general. Provee 24 líneas de E/S organizadas como tres puertos de E/S llamados A,B,C. La figura 7-10 muestra la definición de la patillas su diagrama de bloques. Todos los bits de la parte alta y baja del puerto C se pueden programar en forma separada.

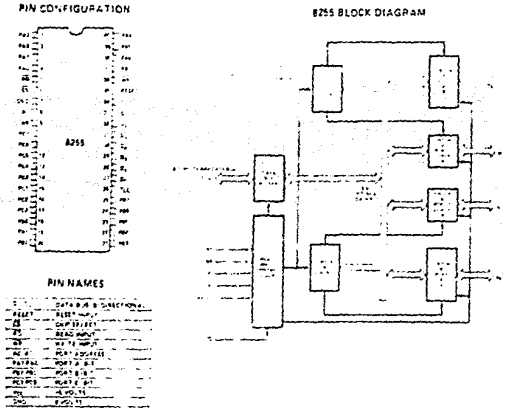


Figura 7-10. El 8255. Interfaz programable. 24 patillas de E/S (Cortesia de Intel Corporation).

El 8255 es un dispositivo versatil. Es posible programarlo como tres puertos sencillos de E/S (modo 0),, dos puertos con protocolo de señales (modo 1), o un puerto bidireccional con 5 señales de protocolo (modo 2). Estos modos se pueden combinar. Por ejemplo, el puerto A es programado para operar en modo 2 mientras el puerto B opera en modo 0. Hay también un bit de modo habilitar/deshabilitar que permite a bits individuales del puerto C se activen o desactiven para propositos de control.

Interfaz con 8255. El 8255A es directamente compatible con la arquitectura de triple Bus del microprocesador 8080, pero tambien compatible con el 8088/8086 y otros tipos de microprocesadores, siempre que tengamos cuidado con las señales de control y diagramas de tiempo.
La tabla 7-1 resume las operaciones de escritura y lectura posibles.

A_1	A_0	$\overline{RD}$	$\overline{WR}$	$\overline{CS}$	
					Input operation (READ)
0	0	0	1	0	Port A → data bus
0	1	0	1	0	Port B → data bus
1	0	0	1	0	Port C → data bus
					Output operation (WRITE)
0	0	1	0	0	Data bus → port A
0	1	1	0	0	Data bus → port B
1	0	1	0	0	Data bus → port C
1	1	1	0	0	Data bus → control
					Disable function
X	X	X	X	1	Data bus → 3-state
1	1	0	0	0	Illegal condition
X	X	1	1	0	Data bus → 3-state

Source: Courtesy of Intel Corporation

Tabla 7.1 Tabla de verdad para el PPI 8255A.

Cuando RD es bajo, es posible leer cualquier de los tres puertos con la combinación adecuada. Cuando A0 y A1 estén altas, el puerto de control se accesa. Esto es un registro especial en el 8255A que controla el modo operativo del dispositivo.

Como determinamos el modo 0?. Escribimos una palabra de control al puerto de control (F3H en la figura 7-11), la cual determina la forma de operación del 8255A.
La figura 7-12 muestra los tipos de palabra de control. Cuando el bit 7 de la palabra de control es un 0, seleccionamos el bit del modo set/reset. Si el bit 7 es un 1, podemos seleccionar de los tres modos.
Estudiando la figura 7-12, podemos ver que los tres puertos se dividen en dos grupos para la selección del modo. El puerto A y la parte alta del puerto C pueden programarse desde el modo 0 hasta el modo 2. El puerto B y la parte baja del puerto C solamente pueden programarse desde el modo 0 hasta el 1.

El Modo Bit Set/Reset

Cuando el bit 7 de la palabra de control es 0, se activa el modo bit set/reset. En este modo cualquier bit del puerto C puede estar en 1 ó 0. Notese que solamente se puede manipular un bit a la vez. Podemos aprovechar estas características para generar señales de control.

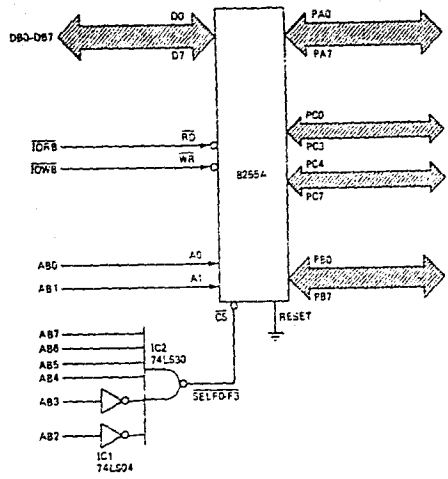


Figura 7-11. El 8255 se conecta directamente a la arquitectura del 8080.

Caracteristicas Electricas

Los tres puertos tienen especificaciones de Iol de 1.7 mA y un Ioh de 200 microamperes. Esto significa que puede manejar una carga TTL estandar o 4 cargas LSTTL. Una caracteristica especial del puerto B y C es que cualquier grupo de lineas puede proveer 1 mA a 1.5V. Esto es útil para manejar relevadores de estado sólido y transistores de potencia baja.

Modo 1 . Entrada/Salida por línea. El modo 1 es para interfaces por interrupcion y señalizacion. En este modo los puertos A y B están programados como puertos de datos y el puerto C para llevar señales de status. Una de las ventajas de este modo, es que puede efectuarse la transferencia de datos sin la intervención del CPU.
Hay cuatro posibles configuraciones para el 8255A cuando se opera en el modo 1. Esto comprende las 4 combinaciones de puertos A y B como entrada y salida.

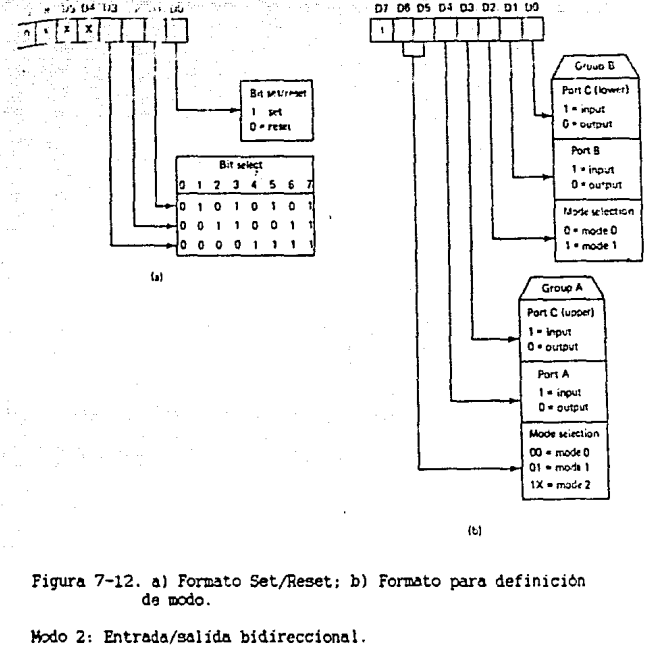


Figura 7-12. a) Formato Set/Reset; b) Formato para definición de modo.

Modo 2: Entrada/salida bidireccional.

Cuando operamos en modo 2, el puerto A del 8255A se convierte en un puerto bidireccional de datos soportado por cinco señales de protocolo
Esto se muestra en la figura 7-13. Los protocolos de señales son idénticos a los del modo 1 excepto que ahora solo existe la referencia al puerto A.
Este modo es útil cuando queremos transferir datos entre computadores.

Cuando programamos el puerto A para operar en modo 2, el puerto B puede operar en modo 0 o modo 1. Si programamos para modo 0, entonces PC0-PC2 pueden ser programados como entrada o salida en modo 0. Si el puerto B se programa para modo 1, entonces PC0-PC2 se convierten en señales de protocolo para este puerto.

Considerando todas las posibles combinaciones, hay 4 configuraciones del 8255A en modo 2. Estas se muestran en la figura 7-14. Como ejemplo, si escogemos programar el puerto A en modo 2, el puerto B como puerto de entrada en modo 0, y PC0-PC2 como salida en modo 0, la palabra de control es 11xxx010.

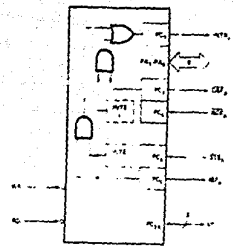


Figura 7-13. Modo 2 para el 8255. El puerto es bidireccional en este modo. Soportado por 5 señales del puerto C. (Cortesia de Intel Corporation.)

Intercambio de datos entre Computadores

La figura 7-15 ilustra una técnica para intercambio de datos entre computadores usando el 8255A en modo 2.
El puerto A se convierte en una trayectoria común para datos entre los dos procesadores con las señales de control del puerto de salida conectadas a las señales de control del puerto de entrada

Cuando los datos de salida del CPU-A llegan a su 8255A la señal $\overline{OBF}$ habilita al 8255-B, causando que la señal IBF (invertida) regrese como un pulso de reconocimiento. Esto, a su vez, pone el dato en el bus de datos bidireccional. El CPU-B lee el dato, terminando el pulso IBF y forzando a un nivel alto a la señal $\overline{ACK}$ del 8255-B. Esto libera $\overline{OBF}$ y completa la transferencia.

Cada procesador en la figura 7-15 requerira 4 rutinas separadas para la transferencia de datos :

1- Inicializacion del 8255A
2- Rutina de interrupción o muestreo para IBF y $\overline{OBF}$
3- Un programa receptor que lee los bytes de datos cuando IBF está alta.
6- Un programa transmisor para datos cuando $\overline{OBF}$ esta baja.

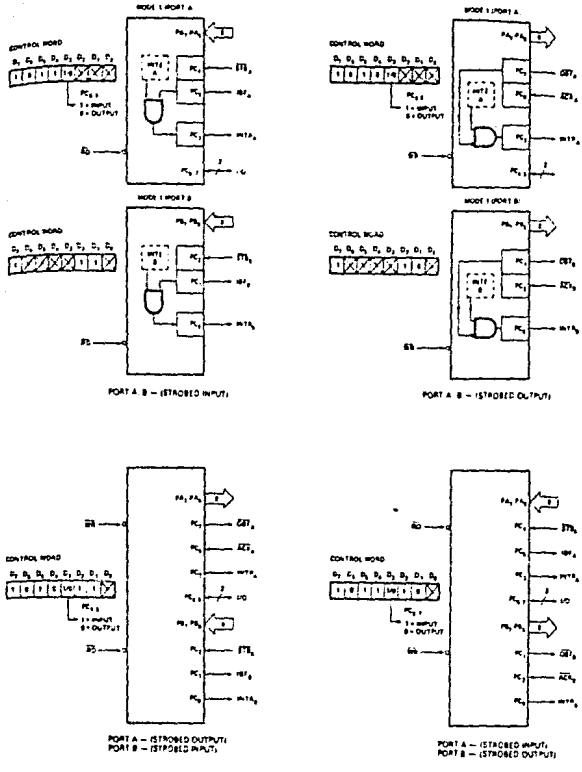


Figura 7-14. Cuatro configuraciones posibles de puertos A y B cuando el 8255 se programa en el modo 1 de operacion. El puerto C se convierte en el puerto de status en este modo.

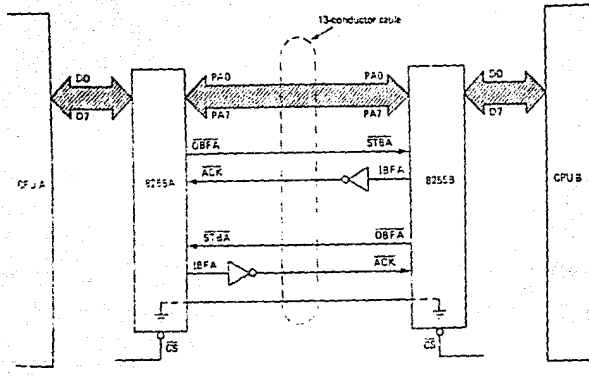


Figura 7-15. Dos 8255's pueden conectar dos computadoras diferentes sobre un bus de datos común.

VII-5 EJEMPLO DE A/D/ Y D/A

La figura 7-16 muestra como el 8255A podría conectarse a un subsistema A/D y D/A. Debido a que durante una conversión del A/D el voltaje analógo debe permanecer sin cambios, se necesita un circuito muestreador/sujetador para mantener la señal analógica constante mientras se efectua la conversión. El grupo A se configura como entrada en el modo 1. La conversión empieza por medio de una señal de la patilla PC7 del 8255A, la cual hace que el convertidor ponga una señal BUSY (ocupado).

Esta señal está conectada a la patilla de control del muestrador/sujetador y a un monoestable de flanco negativo. Mientras la señal busy está alta el muestreador/sujetador mantiene una salida constante, y cuando la señal "busy" está baja al final de la conversión el monoestable se dispara.
La salida del monoestable se complementa y la aplicamos a la entrada STBA (PC4) del 8255A. Esto causa que la muestra digitalizada entre al puerto A. El puerto B está configurado como un puerto de salida en modo 0, el cual se conecta directamente a la entrada binaria del D/A, y no es necesaria ninguna señal de protocolo.
Dado el puerto A, el puerto B, el puerto C y los registros de control con direcciones FFF8, FFF9, FFFA y FFFB, respectivamente, la secuencia :

```
MOV DX,0FFFEH
MOV AL, 10110000B
OUT DX,AL
```

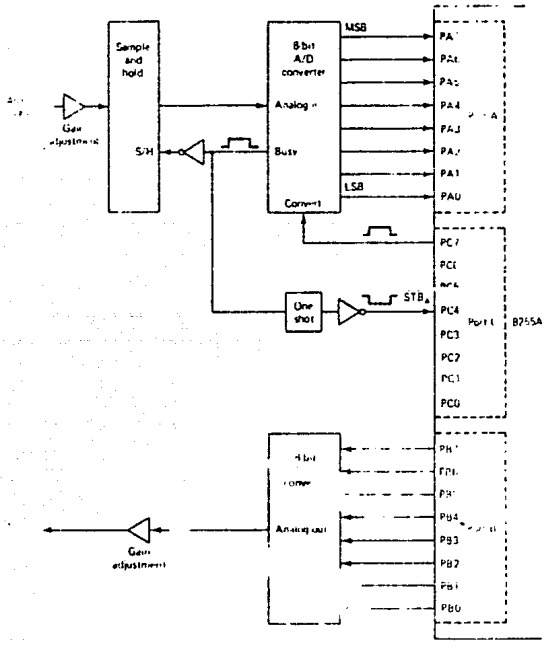


Figura 7-16. Interfaz de un A/D y D/A con un 8255's.

causa que el puerto A este en modo 1, puerto B en modo 0, y PC7 como salida. La secuencia :

```
MOV DX,0FFFEH
MOV AL,10110000B
OUT DX,AL
MOV AL,00001110B
OUT DX,AL
```

saca un pulso a la patilla de conversión en el A/D. La primera instrucción pone la dirección asociada con la instrucción Set/Reset, la cual es la misma que la dirección del registro de control, en el registro DX. Las dos siguientes instrucciones ponen a PC7 en "1" y las dos últimas lo borran. Una secuencia para proveer una entrada programada de los datos convertidos es :

```
      MOV DX, OFFFAH
AGAIN: IN AL,DX
      test AL,00100000B
      jz AGAIN,OFFF8H
      IN AL.DX
```

Para sacar un byte de AL al convertidor D/A :

```
      MOV DX,OFFF9H
      OUT DX,AL
```

En este ejemplo, hemos supuesto que el tiempo de conversión lo da el programa y que las ganacias de entrada y salida en los operacionales se ajustan manualmente.
Para obtener intervalos pares entre las muestras de entrada o salida del programa debemos cuidar tiempos de ejecución de las instrucciones.
Esto significa que debemos usar exactamente las mismas instrucciones entre cada muestra y que el tiempo total de ejecución sea conocido con precisión. Es necesario deshabilitar las interrupciones porque podrían adicionar la ejecución de un número diferente de instrucciones. Este tiempo de muestreo puede ajustarse incluyendo un retardo :

```
      MOV Cx.N
 IDLE: NOP
      loop  IDLE
```

Entre las salidas o entradas. La figura 7-17 muestra un diagrama de flujo con la entrada de un bloque de muestras al A/D.

Frecuentemente,los relojes programables y los dispositivos de control de ganancia están asociados con los subsistemas A/D y D/A de tal manera que el tiempo de las muestras y la salida del D/A las podemos controlar con mayor precisión y es posible cambiar la ganancia de forma dinámica. Tambien es común el uso de un controlador en conjunto con el subsistema A/D y D/A para obtener entrada/salida de alta velocidad de transferencia.
Solamente usamos convertidores de 8 bits en la figura 7-16. Esto limita la resolución a solamente 1 parte en 256. Si el rango de voltaje fuera de -10V a 10V, la resolución sería

Para obtener resoluciones mas altas, requerimos convertidores de 10, 12 o 14 bits. Para lograr conectar más de 8 bits usamos una combinación de los puertos A, B y C; podemos usar dos o más 8255's en paralelo.

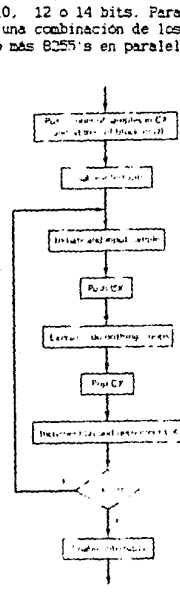


Figura 7-17. Muestreo de tiempo programado.

VII-6 TEMPORIZADOR PROGRAMABLE

Con cierta frecuencia es necesario marcar intervalos de tiempo tanto para el procesador y el dispositivo externo, el temporizador programable cuenta eventos externos y le pasa la cuenta al procesador, y provee los tiempos necesarios los cuales pueden programarse en el procesador.
Tales dispositivos se llaman temporizadores programables y algunos usos son:

1- Interrumpir un sistema operativo de tiempo compartido a intervalos iguales tal que pueda conmutar programas.

2- Producir señales con tiempos precisos que puedan tener periodos programados a un dispositivo (por ejemplo, un

A/D).

3- Servir como generador de baudaje.

6- Medir retardos de tiempo entre eventos

5- Contar el numero de veces que ocurre un evento en un experimento externo y proveer un medio para enviar el dato o la cuenta a la computadora.

6- Provocar que el procesador sea interrumpido despues del numero programado de eventos externos.

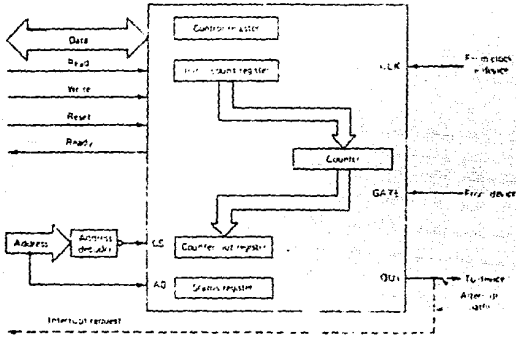


Figura 7-18 Temporizador tipico.

Un diseño representativo lo mostramos en la figura 7-18. En este diseño los cuatro registros a la izquierda son accesibles por la computadora, con los dos superiores como puertos de salida y los inferiores como puertos de entrada. El contador no está directamente disponible al procesador, sino que debe ser inicializado desde el registro de conteo inicial y solo puede ser leido transfiriendo su contenido al registro de salida. El contador opera empezando un valor inicial y decrementando su valor hasta cero. La entrada CLK determina la proporción del conteo. GATE es para habilitar y deshabilitar CLK, y la salida OUT se pone activa cuando la cuenta llega cero. Posiblemente cuando recibe una señal GATE. La señal OUT debe conectarse a una linea de peticion de interrupcion en el bus del sistema asi que una interrupción ocurrira cuando alcance 0. o a un dispositivo de E/S que lo usa para iniciar una actividad especifica.
La operación básica es poner una cuenta en el registro de conteo inicial, transferir la cuenta al contador y hacer

que el contador cuente en forma regresiva conforme aplicamos pulsos a la entrada CLK. Los contenidos del contador pueden leerse en cualquier momento sin perturbar la cuenta transfiriendola al registro de salida y accesando el contenido del contador. Deteniendo la cuenta en el registro de salida no es necesario que el dato sea accesado por el procesador inmediatamente. La indicación de la cuenta cero normalmente se aplica a la patilla de salida y a uno de los bits en el registro de status.

Es posible utilizar ya sea E/S programadas o E/S por interrupción para detectar la cuenta cero.
Entre otras cosas el registro de control incluye el modo de operación. El modo determina exactamente lo que sucede cuando la cuenta es cero y/o se aplica una señal a la entrada GATE. Algunas posibles acciones son:

1- La entrada de GATE se usa para habilitar y deshabilitar la entrada CLK.

2- La entrada GATE puede causar reinicio del contador.

3- La entrada GATE puede detener la cuenta y poner OUT en estado alto.

6- La cuenta dará una señal OUT y se detendrá cuando alcance cero.

5- La cuenta dará una señal OUT y automáticamente reinicia desde el registro de cuenta inicial cuando la cuenta alcance cero.

Los modos podrían definirse como una combinación de estas posibilidades.
Como un ejemplo, considerese la aplicación de un temporizador de intervalos a un sistema operativo de tiempo compartido.
En este caso conectamos un reloj a la entrada CLK y OUT a una línea de peticion de interrupción, posiblemente a una línea no-mascarable. La entrada GATE no es necesaria.
Cuando el sistema inicia, el registro de conteo inicial es :

Cuenta inicial = Frecuencia inicial x T

donde T es la longitud de cada instante en segundos, y el modo se coloca de tal manera que cada vez que la cuenta alcanza cero los contenidos del registro de cuenta incial son transferidos al contador y OUT se vuelve activa. Ya que OUT se usa como una peticion de interrupción, llamamos una rutina de interrupción para conmutar programas al final de cada periodo de T segundos.

Mostramos un diagrama del 8254 en la figura 7-19. El 8254 consiste de tres circuitos de conteo idénticos, cada uno de los cuales tiene entradas CLK y GATE y las salida OUTPUT.

Puede verse cada uno como si tuviera un par de registros de control y status, un registro de conteo (CR) para recibir la cuenta inicial, un elemento de conteo (CE) que ejecuta el conteo pero no es directamente accesible desde el procesador, y un "latch" de salida (OL) para detener los contenidos de CE para poder leerlo.

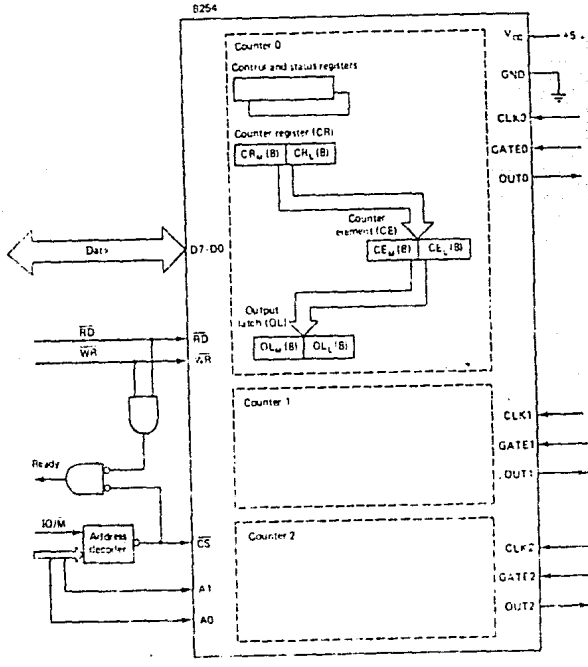


Figura 7-19. Diagrama del 8254.

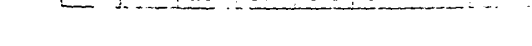
Los CR, CE y OL son considerados como pares de registros de 8 bits.

Los registros pueden ser accesados de acuerdo a la siguiente tabla:

CS	RD	WR	A1	A0	
0	1	0	0	0	Transferir al contador 0 CR
0	1	0	0	1	Al contador 1 CR
0	1	0	0	0	Al contador 2 CR
0	1	0	1	1	Al registro de control o indica un comando.
0	0	1	0	0	Del contador 0 OL o registro de status.
0	0	1	0	1	Del contador 1 OL o registro de status.
0	0	1	1	1	Del contador 2 OL o registro de status.

Donde 0 significa estado bajo y 1 estado alto. Todas las demás combinaciones ponen las lineas de datos en alta impedancia.
Cuando A1=A0=1, los bits más significativos del byte que está a la salida, determinan si se está escribiendo sobre un registro de control o se dá un comando.
Para las tres últimas combinaciones si leemos un registro de status. OL depende del comando anterior.
Hay dos tipos de comandos, el comando del "latch" que obliga a CE en el contador a que sea detenido especificamente por los dos bytes mas significativos en el OL correspondiente, y el comando de prelectura (read back), que causa una combinación de la detención del CE o 'prepara' una combinación de registros de status para su lectura.
Preparar un registro de status significa hacer que se realice en la siguiente ocasion una operacion de lectura desde el contador, pero si es 11 se ejecuta un comando de prelectura. En un comando latch los bits 5 y 4 deben ser 0, y los bits restantes no son usados.

El comando de prelectura tiene el siguiente formato :



Si el bit COUNT es cero, entonces los CE's para todos los contadores cuyos bits CNT sean 1 son detenidos por el latch. Si el CNT0=CNT2=1 pero CNT1=0, entonces los CE's son detenidos pero el CE en el contador 1 no lo es. Similarmente, STAT=0 causa que los registros de status de los contadores se preparen para la entrada.
Los CE's pueden ser detenidos por latch y los registros de status se pueden preparar con el mismo comando.

Los formatos de control y registros de status se muestran en la figura 7-20A. Si los 8 bits de los dos bytes más significativos de una salida son 1, indica que la salida es un comando de prelectura; de otra manera, especifican un contador. Si especifican un contador y los bits 4 y 5 son cero, entonces se indica un comando de latch y es dirigido al registro de control del contador especificado por los bits superiores, pero si no son ambos cero, entonces indican el tipo de entrada del OL o la salida al CR. La combinación 01 indica que las operaciones READ/WRITE son de o hacia OL /CR , 10 indica que las operaciones READ/WRITE son desde o hacia OL / CR , y 11 indica que estas operaciones ocurren en pares, con el primer byte desde o hacia OL / CR y el segundo desde o hacia OL /CR . Una escritura de un byte a CR provocará que el otro byte sea puesto en ceros. Los bits 1,2 y 3 determinan el modo y el bit 0 especifica el formato de la cuenta.

Dado que N es la cuenta inicial, los modos son :

Modo 0 (interrupción en la cuenta terminal)- GATE = 1 habilita el conteo y GATE =0 deshabilita el conteo, y GATE no tiene efecto sobre OUT. Los contenidos de CR se transfieren a CE en el primer pulso CLK despues de escribir a CR por medio del procesador. OUT se pone en estado bajo cuando hay una salida al registro de control y permanece baja hasta que la cuenta va a cero. El modo 0 es principalmente para conteo de eventos.

Modo 1 (mono-estable redisparable por hardware)- Despues que CR ha sido cargado con N, una transición de 0 a 1 sobre GATE causará que CE sea cargado, una transición de 1 a 0 en OUT empieza la cuenta. Cuando la cuenta alcanza el 0, OUT se pone en estado alto, produciendo un OUT con flanco negativo de N periodos de reloj.

Modo 2 (temporizador de intervalos periodicos). Despues de cargar CR con N, se hace una transferencia de CR a CE en el siguiente pulso de reloj, OUT va de 1 a 0 cuando la cuenta se hace "1" y permanece en estado bajo por un pulso CLK; entonces regresa a "1" y CE es recargado desde CR, dando asi un pulso negativo en OUT despues de N ciclos de reloj.

GATE=1 habilita la cuenta y GATE =0 deshabilita. Una transicion de 0 a 1 sobre GATE tambien causa que la cuenta sea reiniciada en los siguientes pulsos de reloj. Este modo se usa para proveer un temporizador de intervalos periodicos programables.

Modo 3 (Generador de ondas cuadradas)- Es similar al modo 2 excepto que OUT se pone en estado bajo cuando alcanza la mitad de la cuenta y permanece bajo hasta que la cuenta es 0. Como ha sido cambiado el ciclo de trabajo, GATE habilita y deshabilita la cuenta y una transicion 0 a 1 sobre GATE reinicia la cuenta.
Este modo puede ser usado para generación de baudaje.

Modo 4 (Señal disparada por Software)- Es similar al modo 0 excepto que OUT está alta mientras se efectua la cuenta y produce un pulso de reloj negativo cuando alcanza la cuenta.

Modo 5 (señal disparada por hardware- redisparable)- Después que se ha cargado CR, una transición en GATE causará una transferencia de CR a CE durante el siguiente pulso CLK. OUT estará alta durante el conteo, pero estará en estado bajo por un periódo de CLK cuando la cuenta sea cero. GATE puede reiniciar la cuenta en cualquier instante.

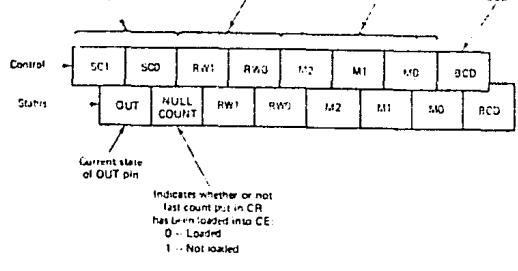


Figura 7-20A. Registros de control y de status para los contadores del 8254.

VII-7. PROGRAMACION DE E/S

Consideraciones sobre E/S

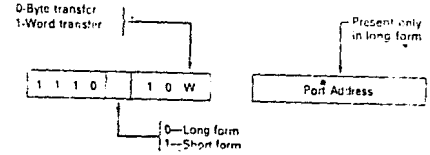
La transferencia de datos desde o hacia un puerto la podemos efectuar de dos maneras. Una manera es ejecutar una instrucción que causa la transferencia de un byte o palabra y la otra es ejecutar una secuencia de instrucciones que involucra un componente especial del sistema asociado con la interfaz para transferir una secuencia de bytes o palabras de o hacia una localidada definida por un bloque de la memoria. Lo ultimo se conoce como un controlador de acceso directo a memoria (DMA) y es una transferencia de bloques.
La figura 7-20, muestra la arquitectura basica de un sistema de un solo BUS.

De los tres tipos principales de E/S : 1) E/S programada, 2) E/S por interrupción, y 3) Transferencia de bloques; la E/S programada y E/S ppr interrupcion estan basadas en transferencia de bytes o palabras y, si los datos van a moverse entre la memoria y un puerto, deben pasar por registros del CPU. Por ejemplo, si acepta una palabra desde un puerto hacia una localidad de memoria.

Para transferir un flujo de bytes o palabras enviadas a la memoria desde una E/S o dispositivo de almacenamiento el programa necesita aceptar los bytes o palabras, una por una, y ponerlas en forma consecutiva en la localidad de memoria. Esto podria hacerse con un programa conteniendo lazos de procesos iterativos.
Una transferencia de bloques es supervisada por la interfaz y el controlador de DMA y es tal que cada byte o palabra se mueve directamente del puerto a la memoria tan pronto como aparece en el puerto.
El programa necesita solamente tener los comandos requeridos por la interfaz y el controlador de DMA para iniciar la transferencia.
Similarmente, la interfaz y su controlador de DMA pueden tomar bytes consecutivos de la memoria y mandarlos al dispositivo externo.
Llegamos a dos preguntas : Como sabe el sistema cuando una interfaz tiene datos disponibles o esta lista para recibir datos; y si mas de una interfaz necesita atencion simultaneamente, quien debe ser atendida primero ?.

En los casos de E/S programadas, el programa determina cual interfaz necesita atencion muestreando los bits "READY" en el registro de status. El muestreo o prueba de estos bits de "READY" se conoce como polling. Para E/S por interrupción, una interrupcion externa se envia al CPU desde la interfaz cuando ésta tiene datos que transferir o aceptar, y la operación se efectua por medio de un rutina de interrupción.
Para una operación de DMA la interfaz pide el uso del BUS enviando una señal sobre la linea de control y efectua las transferencias necesarias sin la ayuda del CPU

Si dos o mas dispositivos requieren un servicio simultaneo, el orden en que las peticiones se atienden depende del arreglo de prioridades del sistema.
Hay varias maneras de designar las prioridades, algunas están basadas en software, algunas en hardware y otras son combinaciones de ambas.
Aunque el arreglo de prioridad puede diferir de acuerdo al tipo de E/S, debe haber siempre un método para resolver conflictos.
En el 8086, todas las comunicaciones programadas con puertos de E/S se efectuan por instrucciones IN y OUT definidas en la figura 7-21. Ambas instrucciones deben transferir un byte o una palabra y ambas tienen un formato largo y uno corto.
El primer operando en la instruccion IN, el operando destino, debe ser, AL (para transferencia de bytes), AX (para transferencia de palabra). Una transferencia de palabras requiere de dos direcciones consecutivas con el byte de menor orden en AX moviendose del puerto con la parte baja de la dirección. Si el segundo operando en la instrucción IN evalua una constante, entonces la constante se usa como la dirección del puerto cuyo contenido está siendo accesado. En este caso la instrucción es de 2 bytes de longitud con la dirección del puerto ocupando el segundo byte. Si el segundo operando es DX, entonces, hay un sólo byte en la instrucción y los contenidos se usan como la dirección del puerto.
A diferencia del direccionamiento a memoria, los contenidos de DX, no se modifican por ningún registro de segmento. Esta forma permite acceso variable a puertos de E/S en el rango de 0 a 64K. El código de lenguaje de máquina para la instrucción IN es :



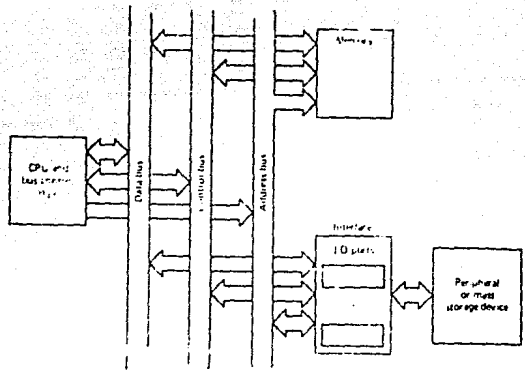
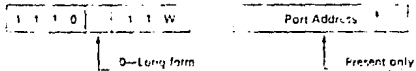


Figura 7-20. Sistema de bus sencillo.

Aunque AL o AX este implicado como el primer operando en una instrucción IN. AX o AL debe especificarse para que el ensamblador pueda determinar el bit W.
La instrucción OUT es similar excepto que la dirección del puerto es el destino, y es indicado por el primer operando, y el segundo es AL o AX.
Su código es :



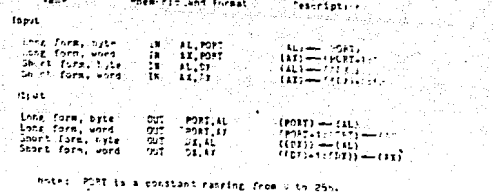


Figura 7-21. Instrucciones IN y OUT.

Notese que si usamos la forma larga de la instruccion IN u OUT la direccion del puerto debe estar en el rango de 0000 a 00FF, pero para la forma corta puede ser cualquier direccion en el rango de 0000 a FFFF (por ejemplo, cualquier direccion en el espacio de E/S). Ninguna de las instrucciones afecta las banderas.

La instrucción IN puede usarse para aceptar datos del registro de datos o el status de un registro de status. Las instrucciones :

```
IN AX,28H
MOV DATA_WORD, AX
```

Mueven la palabra en los puertos cuyas direcciones son 28H y 0029H a la localidad de memoria DATA_WORD. Las instrucciones

```
IN AL,27H
TEST AL,00000100B
JNZ ERROR
```

acepta el byte del puerto 27H y salta a ERROR si el bit 2 de este byte está en 1.

La Instruccion OUT se usa para sacar datos o mandar un comando a un puerto de E/S especificado. Si el bit 7 del puerto 26H causa que empiece una transferencia de bloque, entonces la transferencia podria iniciarse con la secuencia

```
COM_REG   EQU  26H
BEGIN_BIT EQU  10000000B

          .
          .
MOV AL,OTRCNTBITS
OR  AL,BEGIN_BIT
OUT COM_REG,AL
```

Donde OTRCNTBITS contiene otros bits en el comando a enviarse al registro de comando.
Algunas veces es suficiente traer solamente un byte o palabra y usar u operar en el mismo, pero la mayor parte del

tiempo necesitamos una secuencia de bytes o palabras antes que el dato sea significativo. Si varios bytes o palabras se requieren, entonces es necesario transferirlos a un conjunto de localidades consecutivas en la memoria antes de que sean usados.

Un conjunto de localidades usadas para tales propositos se conoce como buffer de memoria, o simplemente buffer.
Por ejemplo, los caracteres en una línea representan un comando al sistema operativo. Ya que el comando no podría interpretarse hasta que se complete la línea. La cadena de caracteres necesitaría almacenarse en buffer de memoria hasta que encuentre la combinación "carriage return/linefeed".

También, es frecuentemente necesario almacenar datos de entrada en forma temporal hasta que puedan convertirse en una forma adecuada u otras operaciones puedan efectuarse. Algunas operaciones pueden emplear más de un buffer de memoria, de tal manera que un buffer está siendo accesado mientras sobre el otro se efectua otra operación: entonces, cuando el buffer que está siendo llenado, o se encuentra lleno, puede operarse y dirigir su contenido a otro buffer Este modo de operación se conoce como doble buffer. Algunas situaciones podrían requerir más de dos buffers y sería "multiple buffer"

Otro aspecto importante con respecto a E/S es como los datos pasan entre procedimientos. Normalmente, los procedimientos que efectuan E/S están separados de aquellos que operan sobre los datos.

VII-8. E/S PROGRAMADA.

E/S programada consiste en examinar continuamente el status de una interfaz y efectuar una operación de E/S con la interfaz cuando su status indica que tiene datos por aceptar o su registro de salida está listo para recibir datos del CPU. Una operación típica de entrada se muestra en la figura 7-22. El diagrama de flujo supone que una secuencia de bytes o palabras aceptada y cada palabra o byte traido al CPU, es modificado y transferido al buffer de memoria.

Despues de que todos los datos han sido transferidos al buffer empiezan a procesarse.
Como un ejemplo más complejo, suponemos que desde una terminal se reciben caracteres a un arreglo de 82 bytes en BUFFER hasta que encontramos un regreso de carro (carriage return o CR) o acepta más de 80 caracteres. Si no encuentra

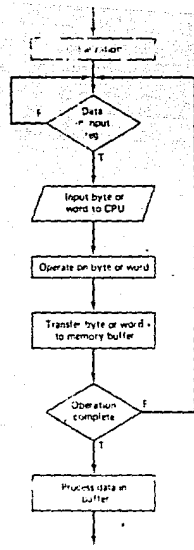


Figura 7-22 Entrada programada.

un CR en los primeros 81 caracteres entonces presenta el mensaje "BUFFER OVERFLOW"; de otra manera, añade un "line feed" automaticamente. Ya que el código ASCII es de 7 bits, el octavo bit, bit 7, se usa como bit de paridad durante la transmisión. Supongamos que el bit 7 es puesto para paridad par y detecta un byte de paridad impar, salta a una rutina de error ERROR.. Si no hay error de paridad, el bit 7, se limpia antes de transferir el byte al buffer de memoria. La interfaz de la terminal se muestra en la figura 7-23, con la dirección del registro de entrada en 0052, y el registro de salida en 0053, el registro de status en 0054. Un "1" en el bit 1 del registro de status indica que el registro de entrada contiene un byte para aceptar y un "1" en el bit 0 indica que el registro de salida está vacio.
La interfaz esta diseñada tal que el bit 1 esta activo cuando un byte entra al dispositivo y es borrado cuando el byte entra a la interfaz. El bit 0 se borra cuando un byte sale a la interfaz y es un "1" cuando un byte sale hacia el dispositivo.
El programa para resolver el problema está en la figura 7-24. Los registros ES y DI se usan como los registros de segmentos y de índice para almacenar los caracteres conforme se aceptan y CX se usa como el contador para limitar procesos iterativos. Las primeras cuatro instrucciones ejecutables que se ilustran para incializar los registros DS

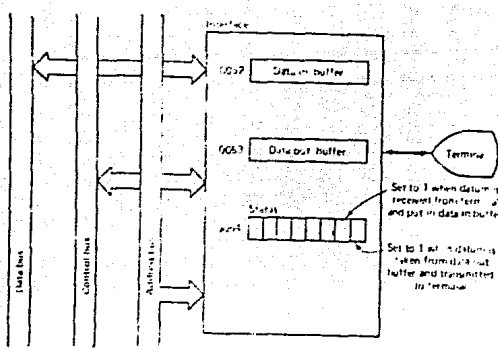


Figura 7-23. Interfaz para E/S programada.

y ES, y las cuatro siguientes ponen la compensación BUFFER en el registro DI y COUNT, ponen CX a la longitud máxima, y borran la bandera DF.
A continuación, hay un proceso iterativo de dos niveles que aceptan caracteres. Las primeras tres instrucciones componen el lazo interno, el cual causa que la computadora espere hasta que el bit de status para entrada de datos sea "1". Luego, entra un caracter al registro AL, donde se checa la paridad, si es par, se borra. Después de borrar el bit de paridad se almacena el caracter y es comparado con el código ascii para un CR.
La instrucción LOOPNE aceptará otro caracter siempre que no se detecte un CR ni tampoco la cuenta en CX se decremente a cero. Si el ultimo caracter es un CR, se añade un line feed y el número total de caracteres se pone en COUNT. En el evento donde la cuenta alcanza cero sin encontrar un CR salta a una rutina que despliega un mensaje de sobreflujo a la terminal.
Esta secuencia incluye un lazo de dos niveles, con el lazo interno simplemente esperando a que el bit de status para la salida de datos sea "1".

Si la línea aceptada fuera llenada con numeros que deben convertirse a enteros y otros formatos antes de usarlos, tenemos que incluir una secuencia para efectuar la conversion.
Si hay más de un dispositivo usando E/S programada, es necesario muestrear los bits de READY de cada dispositivo. La figura 7-25 muestra una secuencia de programa para aceptar datos de tres dispositivos.

Las direcciones de sus registros de status son STAT1, STAT2, y STAT3 y los procedimientos PROC1, PROC2, y PROC3 son llamados para efectuar la entrada de datos.

```
  DATA_SEG   SEGMENT
  MESSAGE     DB     'BUFFER OVERFLOW',ODH,OAH
                .
                .
                .
DATASEG      ENDS
COM_SEG      SEGMENT    COMMON
    BUFFER   DB         82 DUP(?)              ;RESERVE BUFFER AREA
    COUNT    DW         ?                      ;AND COUNT
COM_SEG      ENDS
                .
                .
                .
  IN_BUFF     EQU         52H                 ;ASIGNA NOMERES A
                                                ;DIRECCIONES
  OUT_BUFF     EQU         53H                ;REGISTRO DE INTERFAZ
  STATUS       EQU         54H                ;Y BITS DE READY
  RRDY         EQU         00000010B          ;EN REGISTRO DE
  TRDY         EQU         00000001B          ;STATUS

ASSUME DS:DATA_SEG, ES:COM_SEG
                MOV         AX,DATA_SEG        ;INICIALIZA DS
                MOV         DS,AX              ;Y REGISTRO ES
                MOV         AX,COM_SEG
                MOV         ES,AX
                .
                .
                .
                MOV         DI,OFFSET BUFFER   ;INICIALIZACION
                MOV         COUNT,DI
                MOV         CX,81              ;PARA ENTRADA
                CLD                            ;BORRA DF
    NEXT_IN:    IN          AL,STATUS          ;CARACTER
                TESTAL,RRDY                    ;HAY DATO
                JZ          NEXT_IN            ;EN REGISTRO DE BUFFER
                IN          AL,IN_BUFFER       ;ENTRA CARACTER
                OR          AL,0               ;CHECA PARIDAD
                JPE         NO_ERROR           ;SALTA A ERROR
                JMP         NEAR PTR ERROR     ;SI PARIDAD=0
    NO-ERROR:   AND         AL,7FH             ;BIT PARIDAD=0
                STOSB                          ;MOVER A BUFFER
                CMP         AL,ODH             ;CHECA CR
                LOOPNE      NEXT_IN            ;SIGUE SI
                                               ;NO HAY CR
                                               ;OVERFLOW
                JNE         OVERFLOW           ;SALTA EN
                                               ;OVERFLOW
                MOV         AL,OAH             ;AGREGA OAH
                STOSB
                SUB         DI,COUNT
```

```
                MOV     COUNT,DI        ;SALVA NO.
                                        ;CARACTERES

OVERFLOW:       MOV     SI,OFFSET MESSAGE ;INICIA
                MOV     CX,17           ;PARA SALIDA
NEXT_OUT        IN      AL,STATUS       ;ESPERA
                TEST    AL,TRDY         ;REG. BUFFER
                JZ      NEXT_OUT        ;VACIO
                LODSB                   ;CARAC. SALIDA
                OUT     OUT_BUFF,AL
                LOOP    NEXT_OUT        ;HASTA FIN
```

Figura 7-24. Ejemplo de E/S programada.

El bit 5 es tomado como el bit de entrada "READY" en todos los registros de status. La variable FLAG es para terminar el proceso de entrada y es inicialmente puesta a cero. Se supone que el primer procedimiento de entrada checa una condición de terminación y pone FLAG en "1", por lo tanto causa que el proceso de entrada termine después que todos los procesos pendientes sean completados. La secuencia de programa da al dispositivo una prioridad.
El dispositivo correspondiente a STAT1 tiene la prioridad más alta y los otros dos dispositivos deben esperar hasta que este dispositivo esté libre. El dispositivo correspondiente a STAT3 será servido hasta que uno de los dos dispositivos esté listo (READY)

```
                MOV     FLAG,0          ;CHECAR BANDERA
INPUT:          IN      AL,STAT1        ;CHECAR STAT1
                TEST    AL,20H          ;SI NO HAY ENTRADA
                JZ      DEV2            ;IR A DEV2
                CALL    FAR PTR PROC1   ;ENTRADA DEL DISP. 1
                CMP     FLAG,1          ;SI NO ES PUESTA
                JNZ     INPUT           ;ACEPTAR OTRO DATO
DEV2.           IN      AL,STAT2        ;CHECAR STAT2
                TEST    AL,20H          ;SI NO HAY ENTRADA
                JZ DEV3                 ;IR A DEV3
                CALL    FAR PTR PROC2   ;ACEPTAR DEL DEV2
                CMP     FLAG,1          ;SI BANDERA =0
                JNZ     INPUT           ;ACEPTA OTRO DATO
DEV3:           IN      AL,STAT3        ;CHECAR STAT3
                TEST    AL,20H          ;Y SI
                JZ      NO_INPUT        ;HAY DATO DISPONIBLE
                CALL    FAR PTR PROC3   ;ACEPTAR DATOS DE DEV3
NO_INPUT:       CMP     FLAG,1          ;CHECA BANDERA, SI ES 0
                JNZ     INPUT           ;ACEPTA OTRO DATO
                 .                      ;SI NO CONTINUA
                 .
```

Figura 7-25. Muestreo de prioridad.

La figura 7-25 muestra como el dispositivo podria ser servido por turnos. Esto se conoce como arreglo ROUND-ROBIN. Tal arreglo esencialmente dá a los tres dispositivos la misma prioridad. En este ejemplo. FLAG es checado solamente al final del lazo, y si es "1". el lazo es terminado sin probar entradas adicionales.

VII-9 E/S POR INTERRUPCIONES

Aunque la E/S programada es conceptualmente sencilla puede tener perdidas considerables de tiempo esperando a que los bits de ready esten activos.

En el ejemplo anterior. si la persona que teclea en la terminal pudiera hacer 10 caracteres por segundo y solamente 10 μs se necesitan para que la computadora acepte cada caracter. entonces aproximadamente :

$$99.99/100.00 \times 100\ \% = 99.99\ \%$$

del tiempo no se utiliza. Esto no importaría si no existieran otros procesos para efectuar.
Una interrupción es un evento que causa que el CPU inicie una secuencia fija, conocida como secuencia de interrupción. Antes que empiece una secuencia de interrupcion del 8086, la instrucción ejecutandose en ese momento debe completarse a menos que la instrucción sea HLT o WAIT. (la instrucción WAIT se usa para el coprocesador).

Para un prefijo, ya que el prefijo es considerado como parte de la instrucción, la petición de interrupcion no está reconocida entre el prefijo y la instrucción. En el caso de la instrucción REP. la petición de interrupción se reconoce después de que la operacion siguiente al REP está completa, y la dirección de retorno es la localidad del prefijo de REP.

```
        MOV   FLAG.0              ;BORRAR BANDERA
INPUT:  IN    AL,STAT1            ;ACEPTAR DEL DISP.
        TEST  AL,20H              ;SI ENTRADA ESTA LISTA
        JZ    DEV2
        CALL  PTR PROC1
DEV2:   IN    AL,STAT2            ;ACEPTAR DE DISP2
        TEST  AL,20H              ;SI LA ENTRADA ESTA LISTA
        JZ    DEV3
        CALL  PTR PROC2
DEV3:   IN    AL,STAT3            ;ACEPTAR DE DISP. 3
        TEST  AL,20H              ;SI LA ENTRADA ESTA LISTA
        JZ    NO_INPUT
        CALL  PTR PROC3
NO_INPUT: CMP FLAG,1              ;REPETIR SI LA BANDERA
        JNZ   INPUT               ; ESTA EN CERO
```

Figura 7-26. Muestreo Round-Robin.

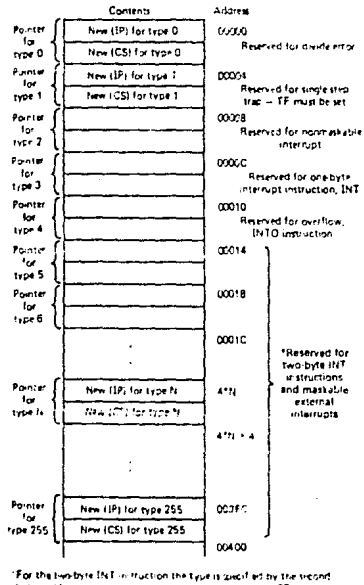


Figura 7-27. Estructura de los apuntadores de interrupcion.

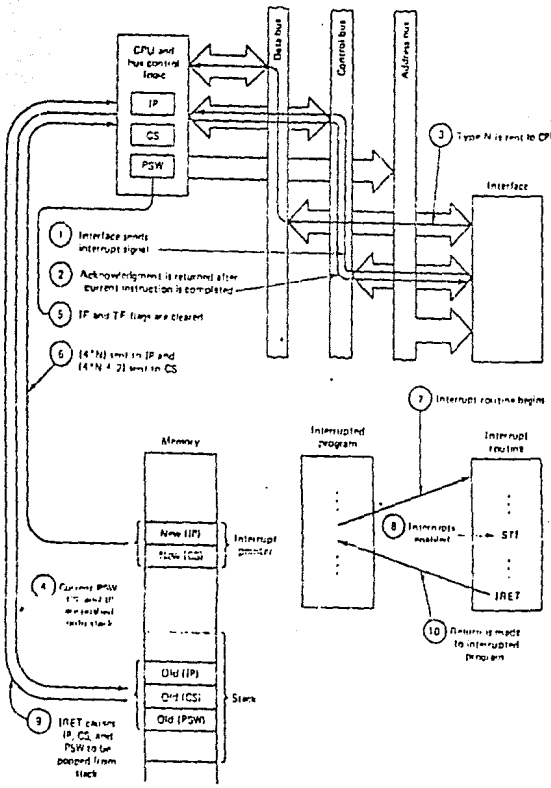


Figura 7-28. Secuencia de eventos durante una interrupción mascarable y su consecuente retorno.

Para las instrucciones MOV y POP en donde el destino es el registro de segmentos, una petición de interrupción no es reconocida hasta después de que se ejecuta la instrucción MOV o POP. Siempre que el registro de segmentos esta lleno primero, esto permite que los contenidos de ambos registros de segmentos y un apuntador sean cambiados sin interrupción. Para el 8086, una vez que la petición ha sido reconocida la secuencia de interrupción consiste de :

1- Establecimiento de un tipo N.

2- Guardando los contenidos actuales de PSW, CS y IP en el apuntador de pila (stack).

3- Borrar las banderas IF y TF.

4- Poner los contenidos de las localidades de memoria 4*N en IP y los contenidos de 4*N+2 en CS.

Así, una interrupción causa que la secuencia normal del programa sea suspendida y derive hacia una localidad indicada por la doble palabra al principio de cuatro multiplicado por el tipo (ejemplo, el apuntador de interrupción).
El control puede regresarse al punto donde la interrupción ocurrió colocando una instrucción IRET al final de la rutina de interrupción.

Hay dos clases de interrupciones, internas y externas, las interrupciones externas son causadas por una señal enviada al CPU a través de una de sus patillas, para el 8086 es NMI o INTR.
Una interrupción iniciada por la señal en la patilla NMI se llama interrupción no-mascarable y causará una interrupción tipo 2 sin importar la bandera IF . Las señales de interrupción no-mascarables son normalmente causadas por circuitos para detectar eventos críticos tales como falla de energía, o por eventos externos o pulsos de reloj que deben ser procesados inmediatamente. Una interrupción en la patilla INTR es mascarada por la bandera IF tal que si esta bandera es cero, la interrupción no es reconocida hasta que IF sea '1'. Cuando IF=1 y ocurre una interrupción mascarable, el CPU regresará una señal de reconocimiento o aceptación a la interfaz del dispositivo y empieza la secuencia de interrupción. La señal de reconocimiento causará que la interfaz que envió la señal de interrupción mande al CPU (sobre el bus de datos) el byte que especifica el tipo, y naturalmente la dirección del apuntador de la dirección. El apuntador, en cambio, provee la dirección de inicio de la rutina de interrupción. Como se indica en la figura 7-27, el tipo de una interrupción externa mascarable debe estar en el rango de 5 a 255, inclusive.

El proceso completo se muestra en la figura 7-28. Para interrupciones externas no-mascarables (tipo2), la señal

INTA no se envia. A menos que se indique otra cosa, suponemos interrupciones mascarables en el ejemplo que sigue.

Regresando al ejemplo que involucra la entrada de una linea de caracteres de una terminal, es claro que si usamos interrupciones, el sistema no tiene que esperar cuando tecleamos. Diseñando la interfaz de tal manera que envie una interrupcion al CPU cuando el buffer de su registro de entrada reciba datos, el CPU puede efectuar otras tareas mientras espera por una interrupcion. Cuando un caracter llega a la interfaz, el CPU recibe el aviso por medio de la señal de interrupcion para que pueda aceptar el caracter, y entonces el CPU puede regresar a otros procesos.
Como un ejemplo, considere los componentes del proceso y su relacion ilustrada en la figura 7-29. El programa principal es para inicializar los apuntadores de la interrupcion necesarios y entonces empezar su proceso normal . Mientras, se ejecuta, se usa la E/S por interrupcion para aceptar una linea de caracteres al buffer que está apuntado por BUFF_POINT.

Suponemos que todas las variables se definen en el segmento DATA_SEG cuya direccion de segmento ha sido almacenada en DS. La localidad de CODE, que inicialemente es puesta a cero, es usada para indicar cuando ha entrado una linea completa (CODE=2) o el buffer de entrada tiene "sobreflujo" (CODE=1).
Ocurre un "sobreflujo" cuando se reciben 81 caracteres sin un CR detectado. En el evento de un sobreflujo, se desactivan las entradas de interrupción, las salidas de interrupción se activan, y usamos la E/S por interrupción para sacar el mensaje de error de MESSAGE. Despues de que una linea ha sido aceptada es llamado un subprograma LINE_PROC para procesar la linea.

Este procedimiento podria consistir en mover datos a un área de buffer para textos, convirtiendo los datos en la linea a un formato diferente, o ejecutando una variedad de otras tareas.
La rutina de interrupcion para ejecutar la entrada y (en el evento de un sobreflujo) la salida se muestra en la figura 7-30. La rutina de entrada aparece primero y la rutina de salida aparece despues. En ambos casos los registros que se usan por la rutina son guardados en el apuntador de pila (stack) y luego recuperados del apuntador de pila (POP) inmediatamente despues del regreso. Como se indica por las directivas EQU que las direcciones de los registros de buffer de entrada y salida son 0052, y 0053, la dirección del registro de control es 0054, y los bits 1 y 0 del registro de control habilitan y deshabilitan las interrupciones de entrada y salida, respectivamente. Las variables COUNT y MSGCOUNT se definen como variables de palabra y ambas son puestas en cero.

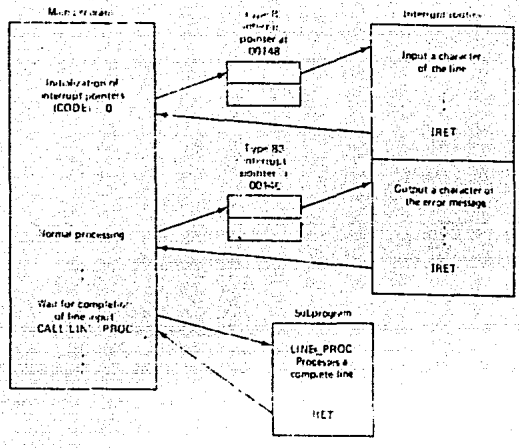


Figura 7-29. Relación entre componentes de procesamiento para el ejemplo de la aceptación de línea que involucra E/S por interrupción.

A continuación presentamos una rutina de un solo buffer:

```
INT_VEC SEGMENT AT 0
ORG 148H
DD  INT_ROUT
DD  OVERFLOW
INT_VEC ENDS
```

en el ejemplo anterior solamente hemos incluido un buffer de memoria y cuando el buffer ha recibido una línea de texto, llama a LINE_PROC para mover los datos o procesarlos.
Se supone que la entrada al buffer será suspendida hasta que LINE_PROC haya completado su tarea, y cuando LINE_PRO termina el buffer estará disponible para recibir una nueva línea. En ese tiempo BUFF_POINT se reinicializa hasta la dirección de inicio del buffer (posiblemente por LINE_PROC). Por otro lado, un programa que continuamente recibe nuevos datos y no puede suspender la entrada mientras procesa un buffer, al menos necesita dos buffers.

```
INT_SEG             SEGEMENT
     ASSUME  CS:INT+SEG, DS:DATA_SEG          ;PARAM. VIA DS
     IN_BUF         EQU        52H
     OUT_BUF        EQU        53H
     CONTROL        EQU        54H
     ENABLE_OUT     EQU        00000001H
```

```
INT_ROUT    PUSH    AX              ;GUARDA REGIST.
            PUSH    BX
            IN      AL,IN_BUF       ;ENTRA CARACT.
            MOV     BX,BUF_POINT    ;ALMACENA EN
            MOV     [BX],AL         ;BUFFER DE MEM.
            INC     BX              ;INC. APUNT. BUFF
            INC     COUNT           ;Y CUENTA
            MOV     BUFF_POINT,BX   ;ALM. APUNT. BUFFER
            CMP     AL,ODH          ;CHECA CR
            JNZ     NO_CR           ;REGRESA
            MOV     BYTE_PTR [BX+1]
                    ,OAH            ;AGREGA UN LF
            INC     COUNT
            MOV     CODE ,2         ;RUTINA 2
            XOR     AL,AL           ;LLAMAR PROC_PROC
            OUT     CONTROL,AL      ;DESACTIVA EL DISP.
            JMP     CONT
NO_CR:      CMP     COUNT,81        ;CHECA OVERFLOW
            JP      CONT            ;SI NO, REGRESA
            MOV     CODE,1          ;CODE=1
            MOV     MSGCOUNT,0      ;LIMPIA MSGCOUNT
            MOV     AL,ENABLE_OUT   ;DESACTIVA ENTRADA
                                    ;Y HABILITA
            OUT     CONTROL,AL      ;SALIDA
CONT:       POP     DX
            POP AX
            IRET

;LA SIGUIENTE RUITNA DE SERVICIO SACA UN CARACTER
;DEL MENSAJE CUANDO OCURRE UNA INTERRUPCION DE UN DISPOSI-
;TIVO DE SALIDA

OVERFLOW:   PUSH    AX              ;GUARDA REGISTROS
            PUSH    BX
            MOV     DX,MSGCOUNT
            MOV     AL,MESSAGE[BX]  ;SALIDA DE CARACTER
            OUT     OUT_BUF,AL
            INC     MSGCOUNT        ;INC. CONTADOR
            CMP     AL,OAH          ;ULTIMO CARACTER?
            JNE     RETURN          ;NO,REGRESA, SI NO.
            XOR     AL,AL           ;DESACTIVA INTERRUP.
                                    ;FUTURAS
            OUT     CONTROL,AL      ; DE LA SALIDA
RETURN:     POP     BX              ;RESTAURA REGISTROS
            POP     AX
            IRET
INT_SEG     ENDS
```

Figura 7-30. Rutina de interrupción para aceptar caracteres.

La figura 7-32 muestra un diagrama de flujo de como podría estructurarse LINE_PROC cuando usamos doble buffer. COUNT lleva una cuenta activa del numero de caracteres que han sido puestos en la linea. Notese que si ocurrio un sobreflujo, las instrucciones

```
MOV AL, ENABLE_OUT
OUT CONTROL, AL
```

Cesan las interrupciones de entrada que son generadas por la interfaz y permite que las interrupciones de salida sean generadas.
Esto permite que el mensaje de sobreflujo sea enviado al usuario de la terminal.
Una secuencia para inicializar los apuntadores de interrupcion, la cual asume que la interrupcion de entrada tiene tipo 82 y la interrupcion de salida tiene tipo 83, se muestra en la figura 7-31.
Las ultimas dos instrucciones son para habilitar las interrupciones de entrada. El código para cargar los apuntadores de interrupción puede aparecer en el programa de usuario, pero es parte de la inicialización efectuada por el sistema operativo si las interrupciones se manejan a través del sistema operativo. Si los apuntadores de interrupción son puestos por el programa de usuario, podrían también colocarse cuando se carga el programa insertando las siguientes directivas al principio del programa :
En el diagrama de flujo BUFFER1 y BUFFER2 representa la direccion de inicio de las areas. Se asume que el procesamiento normal inicialmente pone FLAG en 0 y la direccion BUFFER1 en BUF_POINT. Cuando la primera línea está lista para procesarse. LINE_PROC pone FLAG a "1" y conmuta los contenidos de BUF_POINT apuntando a BUFFER2. Pone la direccion BUFFER1 en BX y la cuenta de caracteres en DI, y entonces el procesamiento de la línea usa los contenidos de estos dos registros.

```
PUSH DS                    ;GUARDA DS
XOR AX,AX                  ;
MOV  DS,AX                 ;BORRA DS  PARA  DIRECCIONAR  UNA
                           ;LOCALIDAD ABSOLUTA.
MOVE AX,OFFSET INT_ROUT
MOV BX,148H
MOV [BX], AX               ;MUEVE OFFSET DE INT_ROUT A 148H
MOVE AX,OFFSET OVERFLOW
MOV [BX+4],AX              ;MOVER OFFSET DE OVERFLOW A 14CH
MOV AX,INT_SEG
MOV [BX+2],AX              ;MOVER  SEGMENTO A 14AH
MOV [BX+6]                 ;MOVER SEGMENTO A 14EH
POP DS                     ;RESTABLECER DS
MOV AL, 00000010B
OUT CONTROL,AL             ;HABILITA EL DISPOSITIVO
```

Figura 7-31. Secuencia de programa para inicializar los apuntadores de interrupciones.

La lectura de cada linea causa que los usos de las direcciones de BUFFER1 y BUFFER2 sean conmutados.
Por consiguiente, los buffers son llenados y procesada su información, mientras un buffer esta siendo llenado la

información del otro esta siendo procesada y viceversa. Notese que el tiempo requerido para procesar una línea debe ser menor que el tiempo necesario para aceptar una línea. Para asegurar que la inicialización está completa antes de aceptar la siguiente línea, es necesario deshabilitar las interrupciones mientras los buffers son conmutados. Una vez que IF es "1", los caracteres enviados por la terminal son capaces de causar interrupción y deben ser aceptados en el buffer que se está llenando.
De acuerdo a lo anterior, la posibilidad de interrupciones simultáneas (o al menos dentro del mismo ciclo de instrucción) no fue tomada en cuenta. También, nada se ha dicho de la incorporación de un esquema de prioridades en el sistema de interrupción. Hay varios modos de combinar las prioridades con E/S por interrupción, algunos involucrando solamente software, algunos solamente hardware, y combinaciones.

Consideremos los siguientes medios para dar prioridades a un sistema de interrupción :

1-Muestreo (polling)
2-Cadena de interrupcion (Daisy-chain)
3-Hardware para manejo de prioridades de interrupción.

Aunque hay diversas formas de interrupciones externas disponibles para su uso por cada tipo de interfaz. Dado un tipo de interfaz, el muestreo será capaz de establecer las prioridades. Por medio de una secuencia de programa (similar a la figura 7-25) al principio de la rutina de interrupción, la prioridad de la interfaz, podría establecerse por el orden en que los datos son muestreadas por la secuencia.

La cadena de interrupción (daisy-chain) es simplemente un medio vía hardware de tener un esquema de prioridad.
Consiste en asociar un circuito lógico con cada interfaz y pasando el reconocimiento de interrupción por medio de estos circuitos como se muestra en la figura 7-33a.

Los detalles de un circuito lógico de interrupción por cadena se muestran en la figura 7-33b. Si una interfaz ha hecho una petición, la señal activa baja INTA causa que se mande una señal de reconocimiento a la interfaz y se bloquea INTA a la siguiente interfaz, si una interfaz no ha hecho el requerimiento, entonces se permite el paso a través de la cadena de interrupción. Por consiguiente, conforme la señal INTA procede a lo largo de la cadena, la interfaz solicitante que se encuentra más cercana al CPU intercepta la señal de reconocimiento, envía el tipo, y completa la secuencia de interrupción, eventualmente liberando su petición.

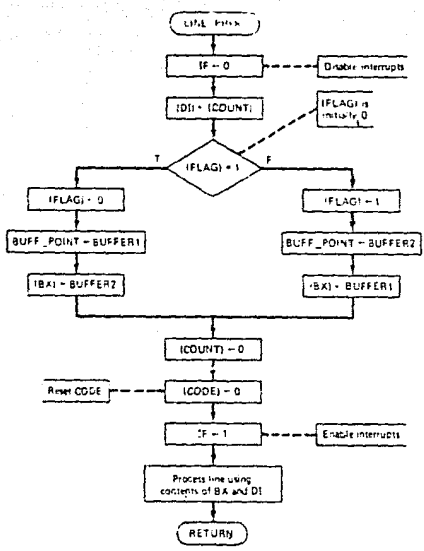


Figura 7-32. Diagrama de flujo de LINE_PROC cuando se requiere doble buffer.

Otros tipos de interfaz que se encuentran alejadas de la cadena no recibiran la señal de reconocimiento y mantendrán su petición. Despues de que una instrucción STI habilita de nuevo la bandera IF, o IF es habilitada por PSW almacenado por una instrucción IRET, el CPU reconocerá peticiones posteriores y enviará otra señal INTA. Esta vez la interfaz cuya petición ha sido atendida liberará su requerimiento y no aceptara la señal INTA. Deducimos que la prioridad de la interfaz, está determinada por su posición en la cadena. Entre más cercana al CPU, mayor su prioridad.

Podemos tener un arreglo más flexible de prioridad hardware/software, diseñando un circuito manejador de prioridad e incluyendolo en la lógica del bus de control. Tipicamente tal circuito se muestra en la figura 7-34. Las patillas INTR e INTA no se conectarian a las interfases, pero serían conectadas al circuito manejador. Las líneas de petición de interrupción que salen de la interfaz se conectan directamente al circuito manejador. El circuito manejador contendría la lógica necesaria para asignar

prioridades a las peticiones entrantes. Por ejemplo, la prioridad más alta podría darse para IR0, la siguiente para IR1, etc. Cuando una petición de interrupción es reconocida por la lógica de prioridad como la más alta, entonces los tres bits menos significativos del registro de tipo se ponen con el número de la línea de petición, se pone un bit a "1" en el registro de servicio, y es enviada una interrupción al CPU. Si IF=1, el CPU regresa una señal de reconocimiento y el circuito de manejo envía al CPU el tipo. todas las peticiones con menor prioridad se bloquean hasta que el bit en el registro de servicio se borra, una acción tomada normalmente por la rutina de interrupción.

Por consiguiente, cuando IF es habilitada por una instrucción STI, las peticiones de mayor prioridad pueden interrumpir la rutina de interrupción en curso, pero las peticiones de menor prioridad serán bloqueadas por la lógica de prioridad hasta que el bit que fue puesto en el registro de servicio se borra. Esto dá a la rutina de interrupción control para que las peticiones de menor prioridad sean reconocidas. Para que el programa sea capaz de limpiar en el registro de servicio, este registro debe ser programable, por ejemplo, debe tener una dirección de E/S de tal manera que sea accesado por IN y OUT
Además de la prioridad interna, se incluye un registro de mascara de 1 byte para permitir el mascaramiento de peticiones individuales. El bit n en este registro sería para mascarar IRn. Se supone que este registro es programable.
En el ejemplo anterior, los tres bits menos significativos del registro de tipo se determinan por la petición seleccionada por medio de la lógica de prioridad. Si este registro es programable, los bits podrían ser inicializados cuando el sistema arranca.
Muchos fabricantes de microprocesadores producen dispositivos para el manejo de interrupciones para complementar sus microprocesadores. El INTEL 8259A es un controlador de interrupciones programable y está diseñado para trabajar con el INTEL 8086/8088.

El manejador de prioridades es similar al circuito de la figura 7-34, pero tiene más propiedades no consideradas.
Para un sistema que involucra varios dispositivos de E/S que tiene que mandar o aceptar información mientras procesa, el software y hardware para procesamiento de interrupciones puede ser muy complejo. Esto es particularmente cierto para sistemas con multiprogramación. Cuando un sistema de software es de propósito específico sirviendo solamente una aplicación o de propósito general capaz de satisfacer numerosas aplicaciones, hay generalmente una componente del programa principal que controla a las demás.
En un ambiente de propósito general esta componente principal es el monitor residente. En cualquier caso el componente principal es el que inicializa cuando arranca el sistema. Entre otras cosas inserta los apuntadores de

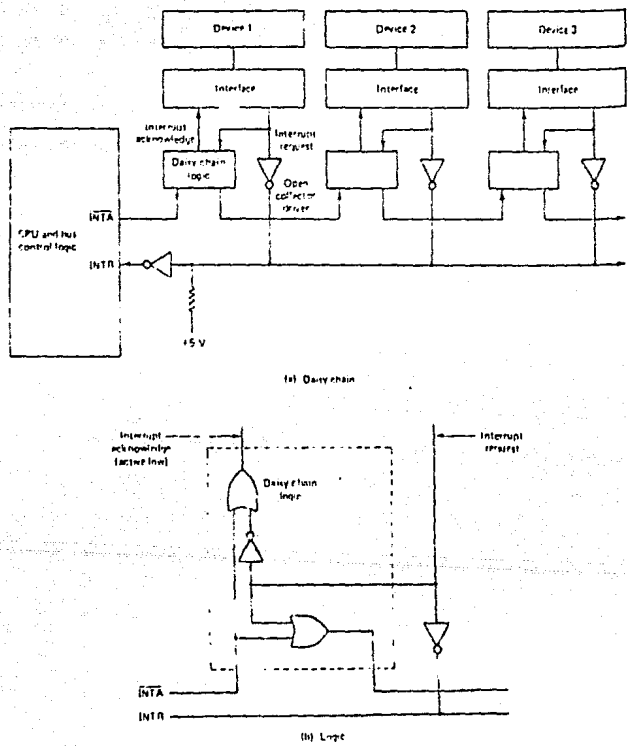


Figura 7-33. Arreglo de una cadena de interrupciones

interrupcion en la parte baja de la memoria (0000H a 03FFH). en un sistema de proposito especial la colocación de los apuntadores de interrupción normalmente permaneceria fijo hasta que el sistema sea apagado, pero en una situación de multiprogramación el sistema operativo puede cambiar algunos de los apuntadores conforme el sistema los necesite.

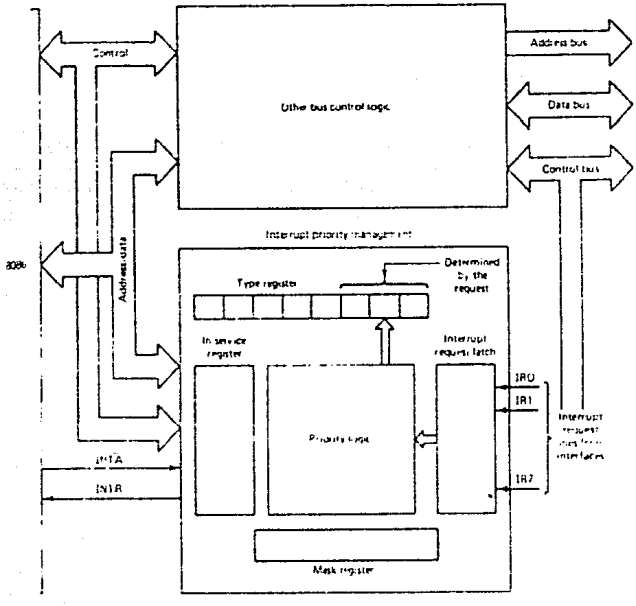


Figura 7-34. Diseño de un manejador de prioridad de interrupciones representativo.

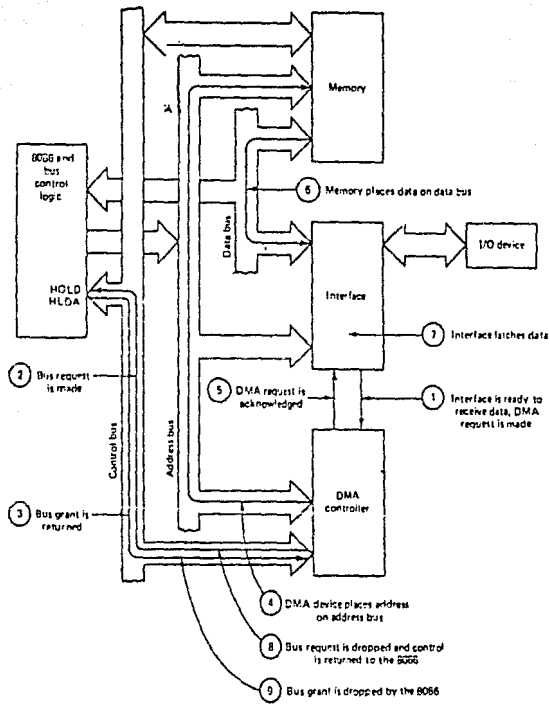


Figura 7-35 Salida de un sólo dato durante la transferencia de un bloque

VII-10 TRANSFERENCIA DE BLOQUES POR DMA (ACCESO DIRECTO A MEMORIA)

Si la velocidad de transferencia de datos desde o hacia un dispositivo periférico es relativamente baja, entonces la comunicación puede ser efectuada por medio de E/S programada o por interrupciones. Pero la ejecución de instrucciones o efectuar secuencias de interrupción lleva más tiempo que el disponible. Algunos dispositivos, tales como las cintas magnéticas y unidades de disco y convertidores A/D, pueden operar a velocidades de transferencia que son demasiado altas para manejarse por transferencia de bytes o de palabra.
Las velocidades de transferencia de E/S para dispositivos de almacenamiento son frecuentemente determinadas por los dispositivos, y no por el CPU, y la computadora debe ser capaz de ejecutar las E/S de acuerdo a la máxima velocidad del dispositivo. Para una unidad de disco la velocidad de transferencia se determina por la velocidad con la que los datos pasan por la cabeza de escritura/lectura y a menudo excede los 200,000 bytes por segundo.

Así, hay menos de 5 microsegundos disponibles para transferir cada byte desde o hacia la memoria. Para transferencias de tal magnitud, requerimos transferencia de bloques, con el uso directo de controladores de DMA para comunicación directa con la memoria.

La actividad involucrada en la transferencia de un byte o palabra sobre el bus del sistema se llama ciclo del bus. La ejecución de una instrucción puede requerir más de un ciclo de bus. Por ejemplo, la instrucción

```
MOV   AL,TOTAL
```

usaría un ciclo de bus para traer el contenido de TOTAL además de los ciclos requeridos para efectuar el Fetch de la instrucción. Una transferencia de bloques de 100 bytes requeriría 100 ciclos de bus, si movemos un byte a la vez, y 50 ciclos si movemos palabras iniciando con dirección par.

Durante cualquier ciclo de bus, uno de los componentes conectado al bus del sistema, toma el control. Este componente se puede convertir en el "maestro" durante ese ciclo y el componente con el que se comunica se dice ser el "esclavo". El CPU con su lógica de control al bus es normalmente el maestro, pero otros componentes especiales pueden ganar el control al bus enviando una petición de bus al CPU. Después que se completa el ciclo del bus del CPU regresará una señal de "bus grant" y el componente que envía la petición será el maestro. Tomar el control del bus por un ciclo del bus se le conoce como "robo de ciclo". Tal como en la lógica de control al bus un maestro debe ser capaz de colocar las direcciones en el bus de direcciones y dirigir la actividad del bus durante un ciclo. Los componentes

capaces de convertirse en maestros son los procesadores (y su lógica de control al bus) y los controladores de DMA. Algunas veces un controlador de DMA se asocia con una sola interfaz, pero se diseñan con frecuencia para adaptarse a más de una interfaz.
El 8086 recibe una petición de bus (bus request a traves de la patilla HOLD y emite "grants" de su patilla HLDA ("sujetar reconocimiento"). Normalbente, se efectua un petición cuando un maestro potencial manda un 1 a la patilla HOLD. Después que se completa el ciclo de bus en curso el 8086 responderá poniendo un "1" sobre la patilla HLDA. Cuando el dispositivo que efectua la petición recibe esta señal "bus grant" se convierte en el maestro. Permanece como maestro hasta que suelta la señal en la patilla HOLD, al tiempo que el 8086 suelta la señal en la patilla HLDA. Una excepción a la secuencia normal es que si una palabra que empieza en una dirección impar se accesa, entonces los dos ciclos del bus se requieren para completar la transferencia y una señal "grant" no sera emitida hasta despues del segundo ciclo.

Cuando un controlador de DMA se convierte en maestro coloca una dirección en el bus de direcciones y envia a la interfaz las señales necesarias para causar que se reciban los datos o se envien al bus de datos. Ya que el controlador de DMA determina cuando se libera la petición de bus, puede regresar el control al CPU después de que cada dato es transferido y la petición de control cada vez que el siguiente dato está disponible, puede retener el control hasta que el bloque completo es movido. Lo anterior es el caso usual ya que permite al CPU continuar su trabajo hasta que el siguiente dato está disponible. Ilustramos la secuencia de acciones tomadas durante una sola salida del DMA en la figura 7-35.

Una transferencia de bloques es una sucesión de transferencias de datos descritas antes. Cada DMA sucesivo usa la localidad de memoria siguiente. Aunque los controladores de DMA pudieran ser designados alrededor de una variedad de configuraciones, todas estas configuraciones deben satisfacer ciertos requisitos.

Ya que cada controlador de DMA podría proveer la dirección cuándo es el maestro, debe ser capaz de almacenar la siguiente dirección de memoria que será accesada. Debe ser capaz también de saber cuando detener la transferencia de bloques.

Mostramos, en seguida una configuración mínima del DMA en la figura 7-36. El diseño del controlador contiene dos registros además de los registros de status y de control, uno para detener la dirección de la siguiente localidad de la memoria, y una para anotar el número de bytes a ser transferidos. Estos registros son puertos de E/S, tal que pueden ser inicializados antes que empiece la transferencia

de bloques. Despues de que cada dato es movido, se incrementa el registro de direcciones (o se decrementa, dependiendo del diseño) y el registro de cuenta se decrementa. El incremento y decremento es de 1 para transferencia de byte, y de dos para transferencia de palabras.
La siguiente secuencia ocurre conforme el dato es enviado de la interfaz a la memoria :

1. La interfaz envia al controlador un peticion para el servicio de la DMA.

2. El controlador gana el control del bus.

3. Los contenidos del registro de direccion se ponen sobre el bus.

4. El controlador envia a la interfaz un reconocimiento de DMA el cual informa a la interfaz que ponga los datos sobre el bus. (Para una salida pone una señal a la interfaz para que detenga o tome los siguientes datos puestos en el bus.)

5. El byte de datos es transferido a la localidad de memoria indicada por el bus de direcciones.

6. El controlador libera el bus.

7. El registro de direcciones se incrementa en 1

8. El registro de cuenta de byte se decrementa en 1

9. Si el registro de cuenta de byte es diferente de cero, regresa al paso 1; de otra manera, se detiene.

El diseño del controlador muestra lineas de direcciones bidireccionales conectadas y solamente las lineas de direcciones van a la interfaz. Esto es naturalmente, porque el controlador puede convertirse en el maestro y colocar las direcciones en el bus mientras la interfaz puede sólo recibir direcciones.

Las lineas de datos bidireccionales se muestran con el sentido hacia la interfaz y el controlador aunque solamente la interfaz pueda transferir datos entre el bus de datos y el dispositivo de E/S. Al igual que la interfaz el CPU debe ser capaz de comunicarse con los registros en un controlador y debe hacerse sobre las lineas de datos.

Si una interfaz se conecta a un dispositivo de E/S, debe de haber un bit en los registros de status y control que indique el tipo de transferencia a efectuar. En suma, estos registros deben contener un bit de "arranque" para iniciar la actividad de E/S (un bit que puede ser sensado por el periferico) y un bit para indicar si el dispositivo esta

ocupado (busy) o no.
Los registros de status y de controlador deben contener un bit de habilitación que define si reconoce o no la petición de la DMA de la interfaz y un bit que determina si el bus será liberado entre las transferencias o si el bus será sostenido en nivel alto para la transferencia. Deben incluir también un bit de dirección de datos tal que cuando el controlador se convierta en maestro sepa si debe supervisar la transferencia de entrada o salida. Para iniciar una transferencia de bloques estos bits de la interfaz y control del DMA tienen que ser puestos en "1" así como poner valores en los registros de cuenta de byte y de dirección. El bit en el registro de control de la interfaz que dirige al dispositivo de E/S para efectuar la operación, que es el bit de inicio, tendría que ser puesto en "1" al último.
Una secuencia típica para empezar una transferencia de bloques se muestra en la figura 7-37.
Esta secuencia asume las siguientes definiciones de bits:

Bit 2 de INTSTAT- Bit de "ocupado (busy) para el dispositivo de E/S.

Bit 1 de DMACON- Informa al controlador de la dirección: 1 es para la entrada.

Bit 5 de DMACON- Habilita el controlador tal que acepte peticiones del DMA

Bit 6 de DMACON- Se borra cuando el bus se libera entre transferencias.

Bit 0 de INTCON- Informa a la interfaz de la dirección de la transferencia: 1 es para la entrada.

Bit 2 de INTCON- El bit de inicio que empieza la actividad de E/S.

Despues de que la secuencia en la figura 7-37 se ejecuta, el dispositivo de E/S empezará a aceptar datos, y el controlador del DMA hará un robo de ciclo del bus y transferirá un byte de la interfaz a la memoria cada vez que un byte se coloca en el registro de la interfaz.
Si el controlador del DMA conectado a un sistema 8086 solamente usa la parte baja de los 16 bits de una dirección, entonces puede solo supervisar una transferencia de bloques a un segmento de la memoria de 64K. Para tener acceso a toda la memoria el controlador debe ser capaz de comunicar la parte alta de los 4 bits de la dirección inicial al controlador así como la parte baja de los 16 bits
Por consiguiente, al menos un byte de direcciones adicional al que se indica en la figura 7-35, se necesitaría para enviar al controlador Normalmente, este byte sería enviado a un puerto adicional de E/S el cual no podría incrementarse durante e la transferencia de bloques Esto implica que la

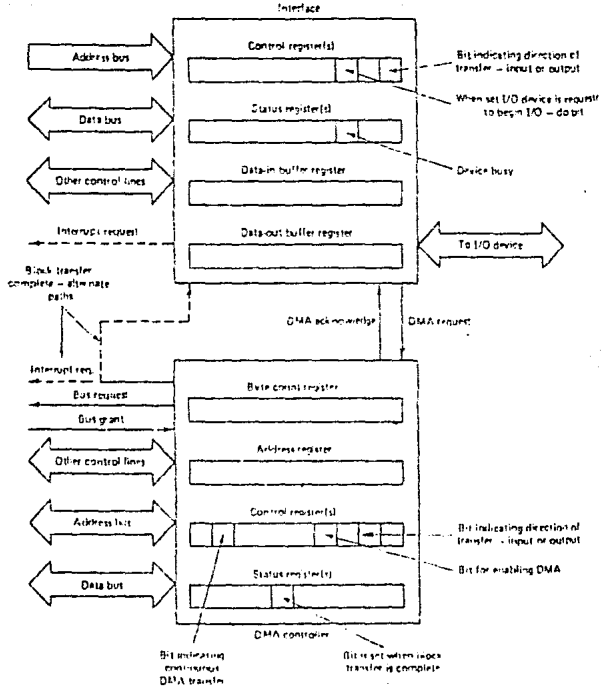


Figura 7-36. Configuración mínima de controlador/interfaz del DMA

transferencia debe efectuarse desde o hacia un solo segmento de 64K, aunque el segmento pudiera diferir de un bloque a otro durante la transferencia.

```
IDLE:   IN      AL,INITSTAT         :HASTA QUE ESTA LIBRE
        TEST    AL,4
        JNZ     IDLE
        MOV     AX,BYTE_COUNT       :PONER TAMANO BLOQUE
        OUT     BC_REG,AX           :REG. DE CUENTA BYTES
        LEA     AX,BUFFER           :PRINCIPIO DE DIR.
        OUT     ADDR_REG,AX         :EN REG. DE DIRECCION
        MOV     AL,DMAC             :PON DIRECCION Y ACTIVA
        OR      AL,0AH              :BITS EN EL BYTE DE CONTROL
        OUT     DMACON,AL           :ENVIAR AL CONTROLADOR
        MOV     AL,INTC             :EN EL REG. DE COMANDOS
        OUT     INTC,AL             :DE LA INTERFAZ
```

Figura 7-37. Secuencia típica para inicializar una transferencia de bloques.

Después de completar una transferencia de bloques, tenemos un bit de "efectuado" u "operación realizada", el cual será puesto en "1" en el registro de status del controlador y será colocada una señal en una de sus patillas. El programa que se está ejecutando puede detectar el fin de la transferencia de bloques checando periódicamente el bit de "efectuado". O la señal enviada a una de las patillas la podemos usar para mandar una interrupción. En el último caso, la señal podría enviarse directamente a la lógica de control del bus o podría enviarse a la interfaz y esta releva la petición y maneja la interrupción. Normalmente un controlador de DMA no está diseñado para proveer el tipo de interrupción; por consiguiente, si la señal se envía directamente a la lógica de control del bus, esta lógica debe incluir el circuito para manejo de interrupciones. La señal que indica la operación completa, se envía normalmente a causa de una petición de interrupción efectuada por otras razones adicionales aparte de la transferencia de bloques, por ejemplo, condiciones de error que solo involucran a la interfaz y no al controlador.
La interfaz y el controlador incluyen registros de status que graban la información vital durante una transferencia. El tipo de información depende del diseño, pero se deben tomar en cuenta cosa tales como la ocurrencia de errores de transmisión o falta de datos antes de que la cuenta de bytes llegue a cero (por ejemplo, fin de cinta).
Después de que la transferencia está completa o ha sido terminada anormalmente, el programa en ejecución o la rutina de interrupción debe checar los registros de status y tomar las acciones apropiadas.
Estas acciones consisten en reingresar la misma información y/o imprimir un mensaje de error. Al procedimiento o rutina de interrupción que efectua estas tareas y acciones se le conoce como rutina de terminación

La figura 7-38 es un diagrama de flujo de una rutina de terminación que esta diseñada para una rutina de transferencia a disco. La rutina checa errores de transmisión o terminaciones anormales.
Si la transferencia se termina anormalmente, un mensaje de error se imprime y la localidad de memoria CODE se pone en "1". Cuando no hay una terminación anormal pero hay un error de transmisión, la variable ERRCNT, la cual se supone ha sido definida en un valor de cero (0) antes de que inicie la transferencia, se compara con 5. Si ERRCNT es menor a 5, es incrementada en 1 y la transferencia vuelve a iniciarse; de otra manera, CODE se pone en cero y se imprime un mensaje de error. Esto significa que la transmisión debe ocurrir en transferencias de cinco bloques consecutivos antes de que acabe el intento de ejecución de transferencia. En el caso de que la transferecia sea completada sin un error, la única acción de la rutina de terminacion es poner CODE a 0.

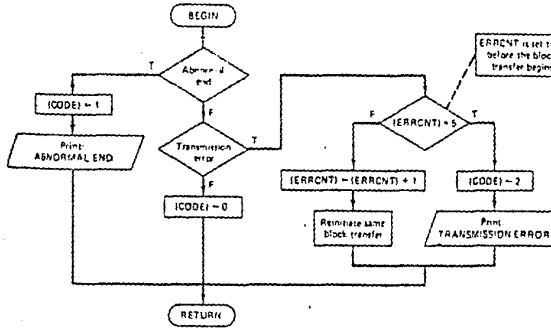


Figura 7-38. Diagrama de flujo de una rutina de terminación.

Si hay más de una interfaz conectada al controlador de DMA, entonces algunos de los registros deben ser duplicados. Si las interfaces son capaces de efectuar transferencias de bloques al mismo tiempo, entonces el controlador tiene que contener un "byte contador" y registro de direcciones para cada interfaz. También, el controlador debe ser capaz de priorizar las peticiones de la interfaces. Los registros de control del controlador deben incluir bits para habilitar en forma individual las líneas de peticiones que entran y de esa manera controlar las transferencias de bloques simultaneas. Sus registros de status deben contener bits para indicar en forma individual que interfaces han

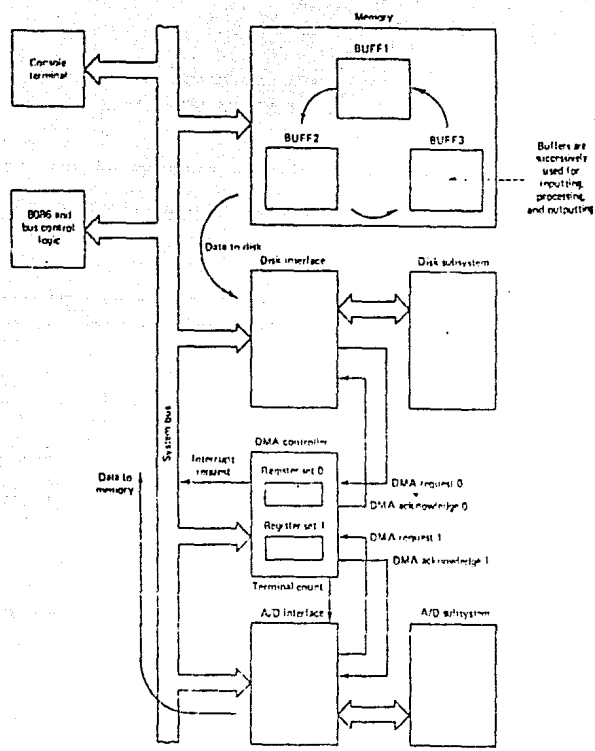


Figura 7-39. Configración de hardware para el ejemplo de triple buffer

completado sus transferencias. Si el controlador tiene solamente una patilla para señalar una terminación, entonces la señal debe usarse para indicar que alguna transferencia está completa, y se le dejaría a la rutina de terminación determinar cual transferencia, checando los bits de status del controlador.

Aunque la configuración del controlador de DMA es independiente del dispositivo de E/S al que sirve, el diseño de la interfaz debe reflejar las características del dispositivo de E/S asociado. Si una interfaz se conecta a un dispositivo que no sea de almacenamiento, entonces la configuración mínima mostrada en la figura 7-36 podría ser adecuada, pero para un dispositivo de almacenamiento la interfaz necesita buscar y direccionar la información al dispositivo. La interfaz para un subsistema de conversión A/D de un solo canal no requiere de más de 2 ó 3 bits de control e información de status, pero necesitaría contener bits para

1- Habilitar la capacidad de interrupción.

2- Indicar errores.

3- Especificar la frecuencia de muestreo.

4- Habilitar la capacidad de DMA.

5- Iniciar la entrada (por ejemplo, el bit de "ejecución").

Con respecto al punto 4, algunas interfaces como aquellas conectadas al subsistema A/D, se diseñan para efectuar ambas transferencias de bytes y palabras y transferencias de bloques. Aunque estas interfaces estén conectadas a controladores de DMA, la utilización de dichas interfaces es opcional. Si el subsistema comprende más de un canal, entonces debe la interfaz contener registros para especificar un número de canal y un modo de multiplexaje

En una unidad de cinta magnética los bits de control especifican parámetros como son

1) Unidad de cinta seleccionada.

2) Densidad de información en bits por pulgada (bpi) y velocidad de cinta.

3) Movimiento de la cinta (Reversa, adelante)

Los bits de status se incluyen para indicar

1) Errores de almacenamiento y transmisión

2) Fin de archivo

3) Fin de cinta

4) La unidad seleccionada esta encendida

5) La unidad selecionada está ocupada.

Ejemplo de diseño

Supongamos que ciertos datos tienen que ser continuamente aceptados a la memoria dsde un convertidor A/D, procesados, y luego escritos a disco.
Para llevar a cabo esto, las transferencias de bloques se usaran en conjunto con la tecnica de triple buffer.
La interfaz a disco y la interfaz a A/D se conectan juntas al controlador de DMA. Un diagrama del sistema de hardware está dado en la figura 7-39. Para simplificar se supone que los tres buffers están en los 64K de la parte baja de la memoria de tal manera que el controlador necesita solamente 16 bits de dirección.

El programa requerido esta dividido en tareas y las relaciones entre estas tareas se muestra en figura 7-40. Hay un programa principal, una rutina de interrupción, y cinco procedimientos. Cuatro de los procedimientos son llamados por el programa principal y uno de ellos, el procedimiento de entrada de datos, es también llamado por la rutina de interrupción. El quinto procedimiento, ERR_ROUT, es llamado cuando ocurre un error en cualquiera de los componentes del programa. El procedimiento INPUT es para iniciar la transferencia de un bloque desde el sistema A/D, PROCESSES es para efectuar el procesamiento, OUTPUT es para iniciar una transferencia a disco, y OUTCOMP es para hacer las tareas necesarias despues de cada salida de bloque.
Una interrupción ocurre siempre que se termina una transferencia de entrada, pero no después de una transferencia de salida. La rutina de interrupción llama al procedimiento INPUT para empezar una nueva transferencia, siempre que se requiera, y entonces efectua las tareas de terminación para la tarea de entrada que esta siendo completada.
Un diagrama de flujo del programa principal se muestra en la figura 7-41 y un resumen de las variables importantes en la figura 7-42. El programa principal primero acepta de una terminal el número de bloques de datos a ingresar y el número de bytes en cada bloque. Estas cantidades se ponen en COUNT y BUF_SIZE, respectivamente. Si entonces se decrementa COUNT, ajusta los bits de control necesarios en el controlador de DMA, incluyendo bits para permitir interrupciones al final de las transferencias desde el sistema de A/D desactivandose al final de las transferencias a disco, e iniciando la primera transferencia de entrada al llamar a INPUT. A continuación, entra a un lazo (LOOP) que

recibe, procesa y saca bloques sucesivos de datos entrando a un buffer de memoria, mientras se procesa otro buffer y tiene como salida otro buffer. Las primeras localidades en los tres buffers de memoria son, BUF1, BUF2, y BUF3. El loop se termina despues de que COUNT es decrementado a cero (0) y los bloques que han sido aceptados han sido procesados y sacados. Siguiendo el loop , el procedimiento OUTCOMP completa la salida final y se efectua el procesamiento de terminacion.

Como a menudo es el caso, se deben tomar pasos especiales al empezar y terminar un proceso. Para saltar el procedimiento de terminacion al inicio o arranque del sistema usamos una bandera de inicio FLAGF.
FLAGF tiene un valor inicial de cero y es puesta a uno por el procedimiento OUTPUT despues de que empieza la primera transferencia de salida. Despues de que se inicia la última entrada de datos el loop debe repetirse hasta que todos los bloques han sido procesados y sacados. Para estar seguros que se ha efectuado lo anterior, usamos una variable FLAGL que controla el loop. FLAGL se incrementa cada vez que se inicia una entrada de datos y es decrementada al principio de cada salida. Ya que FLAGL puede solamente regresar a cero despues de que todas las entradas de datos y procesamientos han terminado y la última salida ha finalizado, el loop es repetido mientras FLAGL es diferente de cero.

Cada vez el programa debe checar, por medio del loop, el siguiente bloque a ser procesado para asegurarse que su entrada es completa, y entonces procesar y empezar la salida del siguiente buffer. Antes de empezar la salida, sin embargo, el programa debe determinar si el disco está ocupado o no lo está con la salida anterior. Si el disco está ocupado, el programa tiene que marcar el tiempo hasta que está disponible. Se inicia un nueva entrada cada vez que ocurre una interrupción al final de la entrada anterior; por consiguiente el procedimiento de entrada no es llamado en forma explícita desde dentro del LOOP.
Para rotar el uso de los buffers de memoria dentro de las tareas de entrada, procesamiento y salida, y al mismo tiempo garantizar que nunca dos tareas van a usar el mismo buffer, se establece una tabla BUFF_TAB. La tabla consiste en tres pares de palabras, la primera palabra en cada par contiene la dirección del buffer y la segunda contiene un indicador. El indicador para un buffer se pone en "1" cuando la entrada al buufer se inicia y se pone en cero (0) despues de la terminación de la entrada. También, se coloca un terminador cuando el procesamiento de su buffer se empieza y es limpiado despues de que el buffer es vaciado.
Antes de que se inicie la entrada o el procesamiento de un buffer se checa el indicador y si el buffer está siendo usado, se bloquea su uso posterior. Esto previene a la entrada de datos de sobreejecución con respecto al procesamiento y salida de información, y viceversa. Notese

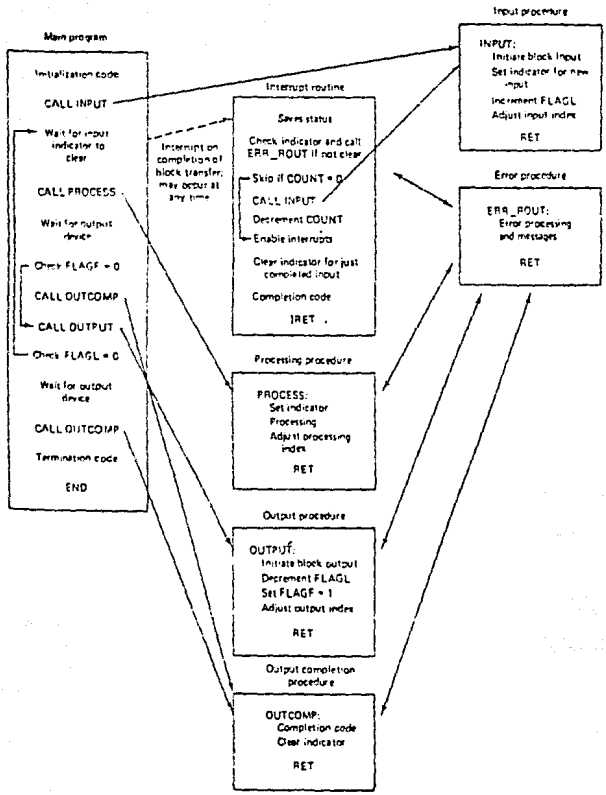


Figura 7-40 Diagrama de la relación de componentes

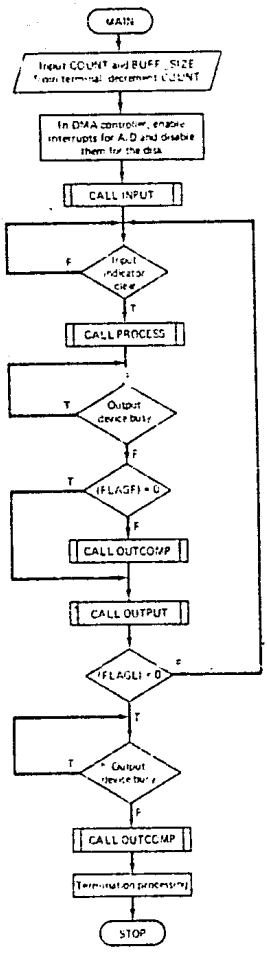


Figura 7-41 Diagrama del programa principal.

que, a diferencia de las tareas de entrada , las cuales deben ser aceptadas por medio de una interrupción que puede ocurrir en cualquier tiempo, las tareas de procesamiento deben seguir una secuencia fija. Esto asegura que el procesamiento y la salida del mismo bloque de datos no se pueden traslapar.
Asociado con la tabla BUF_TAB son los tres índices INDEX_I, INDEX_P, e INDEX_O, que apuntan a los pares en la tabla BUF_TAB. Conforme una entrada esta siendo iniciada, INDEX_I da la posición relativa al par dirección/indicador correspondiente al buffer en el que se efectuará la entrada. Se ajusta a cuatro inmediatamente despues de que se empieza una entrada de datos de tal manera que será correctamente colocada para la siguiente entrada. Los índices INDEX_P e INDEX_O se usan para apuntar al procesamiento en turno y pares de buffers de salida. Estos apuntadores rotan conforme el uso de buffers tiende a rotar.

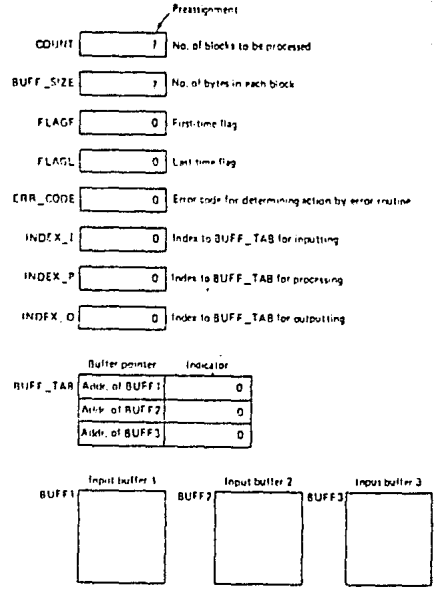


Figura 7-42. Resumen de las variables para el ejemplo de triple buffer.

Las transferencias de entrada y salida son seguidas por conjuntos de códigos de terminación los cuales checan errores que pueden haber ocurrido durante la transferencia previa y alistar el sistema para la siguiente entrada o salida. Los errores son detectados examinando los bits de status en la interfaz apropiada. Asociado con cada tipo de error hay un código, y si encuentra un error, su código es puesto en ERR_CODE y se efectua una llamada a ERR_ROUT. ERR_ROUT procesará el error y regresará, y para errores incorregibles llamados fatales, obligará al programa a terminar. Por ejemplo, el sistema A/D debe ser capaz de aceptar datos en forma continua. Ya que la secuencia de interrupción y el procesamiento que debe llevarse a cabo antes de que pueda iniciar la siguiente entrada, lo que toma tiempo, y es posible perder datos durante la transición entre buffers. Esto normalmente podría ser detectado examinando ciertos bits de status en la interfaz A/D después de que empieza la nueva entrada. Si los datos se han perdido, ponemos entero, por ejemplo 3, en la variable ERR_CODE. El procedimiento ERR_ROUT determinará el entero en ERR_CODE y, después de encontrar un 3, podría derivarse a un código que causa el despliegue en la terminal como se indica:

DATA MISSED IN BLOCK (NO. DE BOLQUE)

El diagrama de flujo detallado de la rutina de interrupción y el procedimiento INPUT se dan en la figura 7-43 y 7-44. Cuando ocurre una interrupción, el status del sistema A/D es aceptado desde la interfaz del A/D. Para minimizar la oportunidad de perder datos, éste status no es checado inmediatamente, sino que es guardado en el apuntador de pila (stack) y luego es examinado después de que inicia la siguiente transferencia de entrada. Al mismo tiempo que se aceptan INPUT, PROCESS y OUTPUT se supone que estos esperan que el registro BX contenga la dirección de BUFF_TAB y el registro SI contenga el índice apropiado.
Por consiguiente, la rutina de interrupción o el programa principal deben llenar estos registros antes de llamar a INPUT. También, después de llamar a INPUT se prueba el indicador para la siguiente entrada al buffer y el valor de COUNT. Si el indicador está en estado alto, se pone -1 en ERR_CODE y se llama ERR_ROUT. ERR_ROUT imprime un mensaje y pone COUNT en cero. Count es probada después del indicador, y por consiguiente, no hay más entradas si el siguiente buffer está siendo procesado, o COUNT ha llegado a cero (por ejemplo, El numero total de bloques asignados ha sido transferido). Si la COUNT es diferente de cero. La rutina INPUT es llamada y COUNT es decrementada. Una instrucción STI entonces habilita las interrupciones y el indicador para la entrada previa es limpiado. Por último, el status correspondiente a la entrada previa se examina por medio del código de terminación y se ejecuta la instrucción IRET.

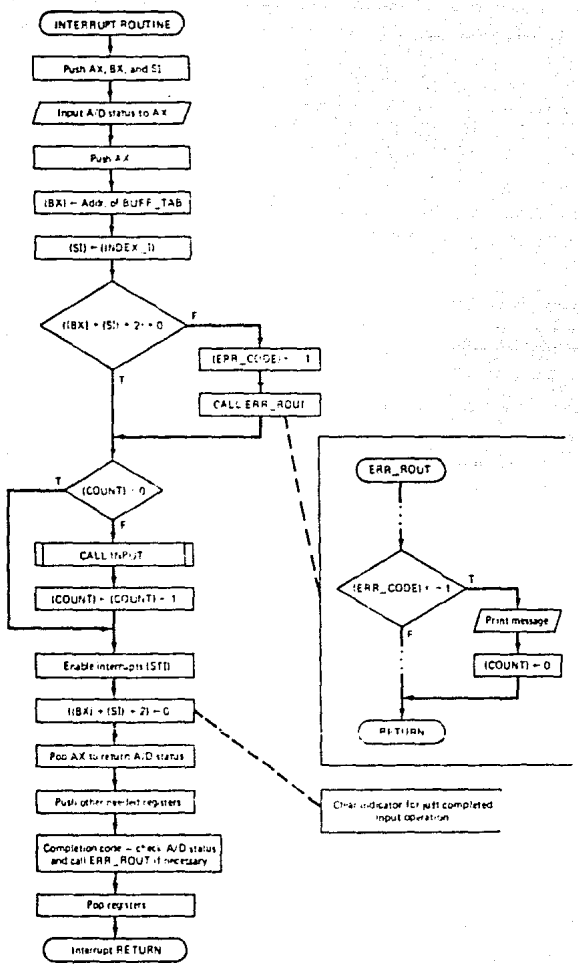


Figura 7-43. Diagrama de flujo de la rutina de interrupción

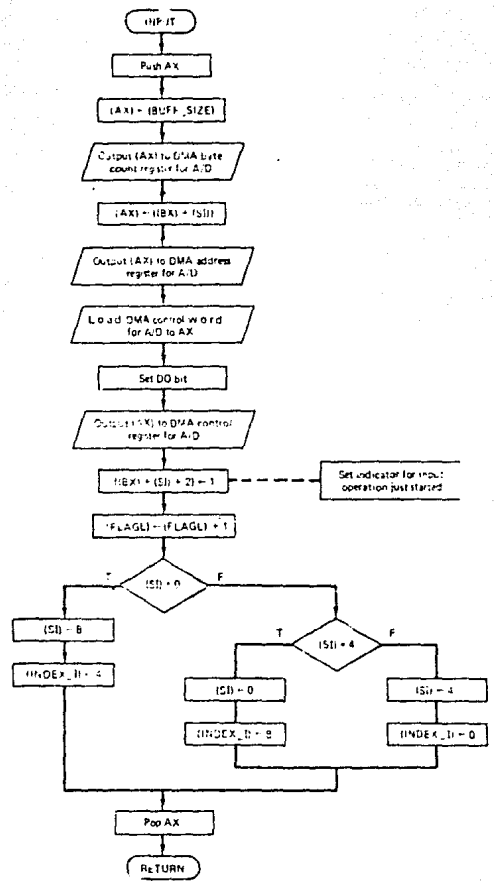


Figura 7-44. Diagrama de flujo del PROCEDURE INPUT.

El procedimiento INPUT pone los contenidos de BUF_SIZE y la dirección del siguiente buffer en el registro de cuentas y el registro de direcciones en 1 en el controlador de DMA, y entonces pone el bit de "ejecucion" que causa a que inicie la transferencia de entrada. A continuación, pone el indicador del buffer a "1", incrementa FLAGL, ajusta INDEX_I para apuntar al buffer positivo, y coloca el registro SI de tal manera que la rutina de interrupción puede usarlo para limpiar el indicador para que el buffer pueda ser llenado.
Para evitar la perdida de datos, la cantidad de procesamiento entre el tiempo que ocurre una interrupción y el bit de "ejecución" se pone en "1" debe ser lo mas pequeño posible. A riesgo de complicar el programa y degradar su estructura de control, la cantidad de este procesamiento mostrado en los diagramas de flujo se podría reducir.
Sin embargo, aún si el código de inicio de entrada se incorpora al principio de la rutina de interrupción, la secuencia de interrupción y los tiempos de conmutación entre buffers todavía limitarían la velocidad máxima de muestreo en el convertidor A/D. Si solamente se llenara o usará un solo buffer, de tal manera que no sea necesaria la conmutación, la velocidad de muestreo estaría limitada básicamente por el tiempo requerido para enviar un dato sobre el bus. Es importante hacer notar que debemos tener muy en cuenta el análisis de los diagramas de tiempos cuando están involucrados : la interrupciones, transferencias de bloques y buffers multiples. Estos casos se pueden manejar con lo siguiente .

1- Los datos podrían ponerse en un solo buffer a una velocidad de muestreo limitada por el tiempo del ciclo del bus. El tiempo de procesamiento y salida podrían ser ilimitadas.

2- Los datos podrían ponerse en numero limitado de buffers (al menos 3) a una velocidad de muestreo limitada por el tiempo de conmutación entre buffers. El tiempo de procesamiento y salida podrían ser ilimitados.

3- Cualquier numero de bloques podrían ser aceptados a una velocidad de muestreo limitada por el tiempo de conmutación. Ni el tiempo de procesamiento de bloques ni el tiempo de salida de bloques puede exceder el tiempo de entrada de bloque.

VIII- DISEÑO DE UN PROTOTIPO

VIII-1 BASES PARA EL ANALISIS Y DISEÑO DE UN SISTEMA.

El diseño de una interfaz para la adquisición de datos en una microcomputadora compatible requiere de los siguientes pasos :

a) decodificar las líneas de direcciones
b) receptores para el bus de datos

El circuito electrónico puede ser tan complicado como se requiera, puede incluir su propio procesador, en algunos casos del mismo nivel o mayor nivel que el CPU central.
Es deseable para el analisis espectral, ya que involucra transformada rapida de Fourier, que la interfaz tenga su propio procesador y en algunos casos puede tratarse de un procesador de señales digitales (DSP), los cuales ultimamente han tenido un auge increible, dando por resultado unos dispositivos de alta velocidad de procesamiento (hasta 25 mips), siendo posible realizar sistemas en tiempo real en forma independiente o como parte de un sistema integrado a una microcomputadora.
El diseño basico consiste en solo una lógica para decodificar las direcciones, y una interfaz paralela en un circuito integrado 8255A, el cual fue descrito anteriormente.
Al incorporar un microprocesador a la interfaz, el cual operará en combinación con el CPU central de la microcomputadora, se incrementa el grado de complejidad tanto de hardware como del software, y especialmente en el desarrollo de software.
Es recomendable tener en cuenta el mapa de direcciones en detalle de una microcomputadora, como se muestra en la figura 8-1 y figuras 8-1A a 8-1I. El circuito mostrado en la figura 8-2 describe el diagrama electronico de la interfaz principal al slot (ranura) del microcomputador.
En este caso se ha optado por una configuración en el A/D que permite una conversion continua. Pero podriamos usar una configuración donde podemos controlar el "inicio de conversion" por software y en algunos casos por hardware o ambos casos.
Normalmente el espacio de direcciones disponible de E/S es de 0300H a 03FFH, en este intervalo es permitido mapear dispositivos a memoria. Si no respetamos este rango de direcciones podriamos causar un conflicto de E/S.
El integrado 74LS138 provee 8 lineas de activación. Solamente usamos una línea para el 8255A. Las otras 7 líneas pueden conectarse a 7 integrados 8255A para obtener más líneas de E/S. La dirección es 0780H. La tabla 8-1 da las direcciones de E/S para las 8 líneas del 74LS138.
La logica de decodificación no usa las líneas A14 y A15. Esto significa que si otro dispositivo usa una dirección mayor a 03FFH, podría haber conflicto.
Hay tres modos de operación, seleccionables por software,

para el 8255A : modo 0 (entrada/salida basica), modo 1 (entrada/salida por señal), y modo 2 (bus bidireccional)
El integrado tiene tres puertos de datos : A, B y C.
El puerto C puede configurarse en forma diferente, esto es, podemos cambiar bits separados sin afectar los otros bits del puerto. El patrón de bits requerido para poner en "1" o en "0" un bit se muestra en la figura 8-3. Para poner en "1" el bit 5 en el puerto C, ponemos un 11 (decimal) al registro de control de palabra. Para poner el bit 2 en "0", poner un 4 al registro de palabra de control.
Hay 4 registros internos en el integrado 8255A. Los primeros 3 son para los tres puertos de datos y el cuarto es el registro de control. En la configuración dada por la lógica de decodificación, el puerto A tiene la dirección 1920(decimal), el puerto B tiene la dirección de E/S de 1921, el puerto C tiene una dirección de 1922 y la del registro de control es 1923.
Para cambiar la dirección de los puertos, usar la tabla 8-2. Por ejemplo, para poner todos los puertos como salidas, poner la palabra de control en 128. Para poner todos los puertos como entrada es 155.
Para poner el puerto A como salida, el puerto B como puerto de entrada y el puerto C dividido en dos, donde la mitad es entrada y la otra mitad es salida, se requiere una palabra de control de 128+8+2 o 138.
Ya que el A/D está convirtiendo en forma continua, el puerto del 8255A toma una muestra válida cada 200 µs. El muestreo exacto lo determina el valor del capacitor de 150 pF y la resistencia de 15K conectados a la patilla 4 del ADC0804, por otra parte, si el integrado ADC0804 se conectara directamente a las líneas del bus, entonces el software debe enviar un comando de "inicio de conversión", y esperar a que el A/D complete la conversión, luego aceptar el dato.
El 8255A se muestra en la figura 8-4. La comunicación entre la microcomputadora y el mundo real es por medio de 24 líneas de E/S. Estas están divididas, como ya mencionamos, en 2 grupos de 8 líneas, los puertos de datos A y B, y el puerto C, en dos grupos de 4 líneas. Este modo soporta transferencia de datos sin señalización.
En modo 1, los puertos A y B pueden ser configurados como entrada o como salida. No pueden definirse en forma individual como sucede en el caso de algunos dispositivos similares.
En modo 2 se usan las 8 líneas del puerto A para transferencia bidireccional. La señalización es lograda por medio del los 5 bits más significativos del puerto C.

LA PROGRAMACION DEL 8255A

El modelo de programación del 8255A, como ya mencionamos, es de 4 registros de 8 bits, puertos A, B y C y un registro de control. El mapa de direcciones se muestra en la figura 8-5.

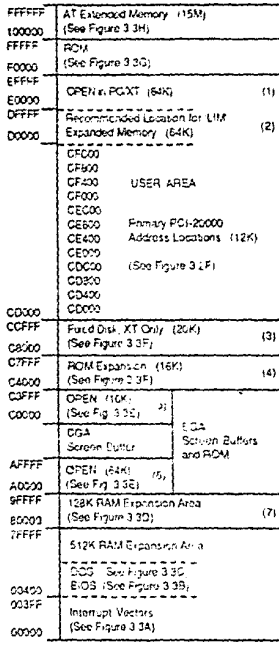


Figura 8-1. mapa de dispositivos en memoria para un IBM PC.

Pin number	Name	I/O address (hex)
15	Y0	0780–0783
14	Y1	0784–0787
13	Y2	0788–078B
12	Y3	078C–078F
11	Y4	0790–0793
10	Y5	0794–0797
9	Y6	0798–079B
7	Y7	079C–079F

Tabla # 1 Direcciones de E/S

D_4	D_3	D_1	D_0	PORT A	PORT C (UPPER)	PORT B	PORT C (LOWER)
0	0	0	0	OUTPUT	OUTPUT	OUTPUT	OUTPUT
0	0	0	1	OUTPUT	OUTPUT	OUTPUT	INPUT
0	0	1	0	OUTPUT	OUTPUT	INPUT	OUTPUT
0	0	1	1	OUTPUT	OUTPUT	INPUT	INPUT
0	1	0	0	OUTPUT	INPUT	OUTPUT	OUTPUT
0	1	0	1	OUTPUT	INPUT	OUTPUT	INPUT
0	1	1	0	OUTPUT	INPUT	INPUT	OUTPUT
0	1	1	1	OUTPUT	INPUT	INPUT	INPUT
1	0	0	0	INPUT	OUTPUT	OUTPUT	OUTPUT
1	0	0	1	INPUT	OUTPUT	OUTPUT	INPUT
1	0	1	0	INPUT	OUTPUT	INPUT	OUTPUT
1	0	1	1	INPUT	OUTPUT	INPUT	INPUT
1	1	0	0	INPUT	INPUT	OUTPUT	OUTPUT
1	1	0	1	INPUT	INPUT	OUTPUT	INPUT
1	1	1	0	INPUT	INPUT	INPUT	OUTPUT
1	1	1	1	INPUT	INPUT	INPUT	INPUT

Tabla 8-2 Palabras de control para el 8255A

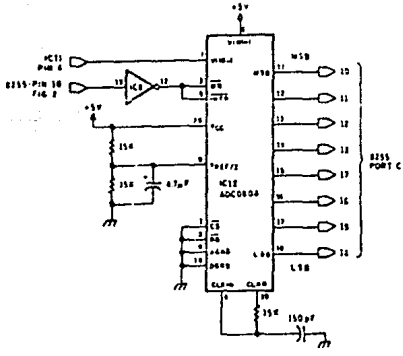


Figura 8-2A Diagrama del convertidor A/D

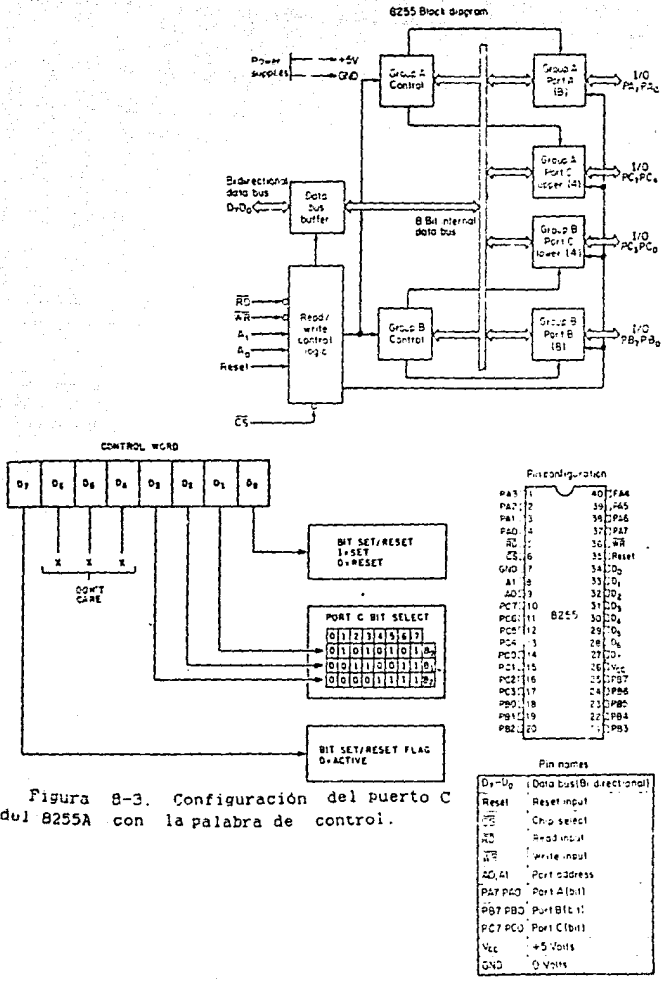


Figura 8-3. Configuración del puerto C del 8255A con la palabra de control.

Figura 8-4. Diagrama esquemático del 8255A y descripción de patillas

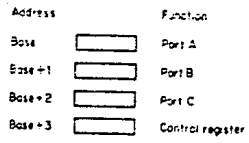


Figura 8-5 . Modelo de programación para el 8255.

La operación de los puertos de E/S se controla mediante el formato de la palabra de 8 bits que se escribe en el registro de control, localizado en al dirección (BASE + 3). El formato de la palabra de control se muestra en la figura 8-6. La tabla 8-3 representa la configuración en modo 0.

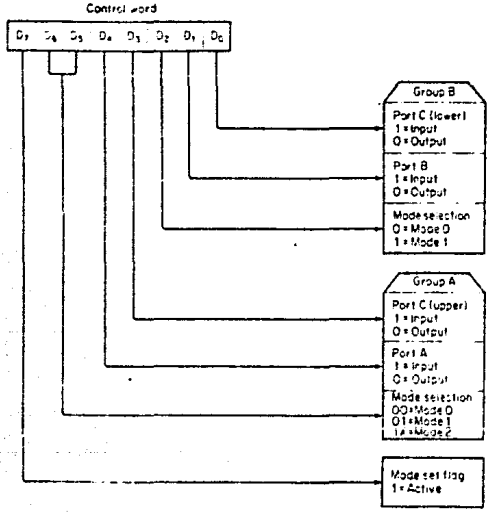


Figura 8-6. Palabra de control para el 8255

Como se muestra en la figura 8-7, el bus de una IBM PC o compatible es un bus de 8 bits implementado en un conector de 62 contactos.
Muchas de las señales involucran manejo de DMA o acceso directo a memoria (ver capitulo VII-10) y el uso de interrupciones.
Las líneas de direcciones A0-A19 pueden direccionar hasta 1 MB de espacio de direcciones. Aunque el 8088 puede usar 16 líneas (A0-A15) para accesar 64 KB de espacio para E/S, solo se decodifican las líneas A0-A9, restringiendo el número de puertos a 1024. La tabla 8-4 muestra las localidades de memoria de E/S.

No	Control Word Bits								Group A		Group B	
	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Port A	Port C (Upper)	Port B	Port C (Lower)
0	1	0	0	0	0	0	0	0	OUTPUT	OUTPUT	OUTPUT	OUTPUT
1	1	0	0	0	0	0	0	1	OUTPUT	OUTPUT	OUTPUT	INPUT
2	1	0	0	0	0	0	1	0	OUTPUT	OUTPUT	INPUT	OUTPUT
3	1	0	0	0	0	0	1	1	OUTPUT	OUTPUT	INPUT	INPUT
4	1	0	0	0	1	0	0	0	OUTPUT	INPUT	OUTPUT	OUTPUT
5	1	0	0	0	1	0	0	1	OUTPUT	INPUT	OUTPUT	INPUT
6	1	0	0	0	1	0	1	0	OUTPUT	INPUT	INPUT	OUTPUT
7	1	0	0	0	1	0	1	1	OUTPUT	INPUT	INPUT	INPUT
8	1	0	0	1	0	0	0	0	INPUT	OUTPUT	OUTPUT	OUTPUT
9	1	0	0	1	0	0	0	1	INPUT	OUTPUT	OUTPUT	INPUT
10	1	0	0	1	0	0	1	0	INPUT	OUTPUT	INPUT	OUTPUT
11	1	0	0	1	0	0	1	1	INPUT	OUTPUT	INPUT	INPUT
12	1	0	0	1	1	0	0	0	INPUT	INPUT	OUTPUT	OUTPUT
13	1	0	0	1	1	0	0	1	INPUT	INPUT	OUTPUT	INPUT
14	1	0	0	1	1	0	1	0	INPUT	INPUT	INPUT	OUTPUT
15	1	0	0	1	1	0	1	1	INPUT	INPUT	INPUT	INPUT

Tabla 8-3. Carta para la definicion del modo 0.

Para desarrollar la programación que controla la interfaz tanto en adquisición de datos como procesamiento por medio de la transformada rápida de Fourier y su representación gráfica con todos los parámetros, los compiladores de lenguaje 'C', tal como el Microsoft C Compiler version 5.1 soportan las funciones inp() y outp(), definidas en el programa de libreria conio.h.
El acceso de datos desde localidades especificas de memoria es vital en las interfaces, no importando el lenguaje que se use. Por esta razón se incluyen ejemplos basicos de E/S en GW-BASIC y Microsoft C Compiler, basados en los manuales respectivos como se muestra en la figura 8-8

Ejemplo 1

```
10 REM ' uso de peeek en E/S'
20 p=inp(768): REM ' lee el puerto A'
30 print p
```

```
ejemplo 2
10 REM Ejemplo de poke en E/S
20 out 769,50:REM ' escribir al puerto B'
```

```
Ejemplo 3
/*****************
uso de E/S
*****************/
#include <stdio.h>
#include <conio.h>
main ()
{
unsigned char p;
p=inp(768);
/************
Leer el puerto A
************/
```

Ejemplo 4:

```
/*****************
* Escritura a un puerto *
*****************/
#include <stdio.h>
#include <conio.h>
main ()
/***********
* Escribir a un puerto B*
***********/
}
```

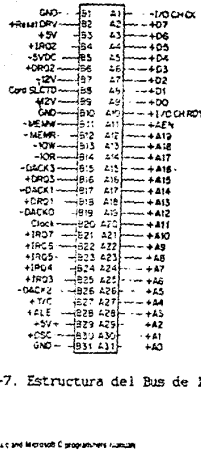


Figura 8-7. Estructura del Bus de la IBM PC.

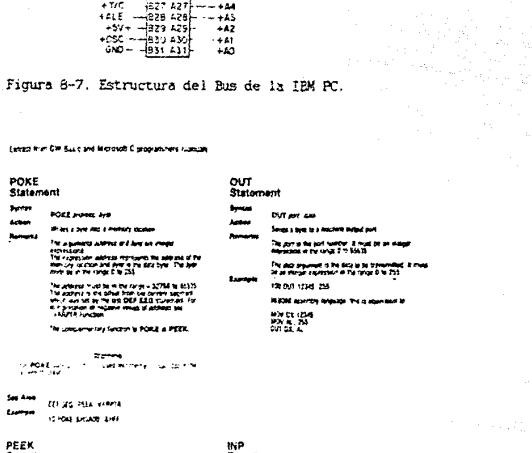
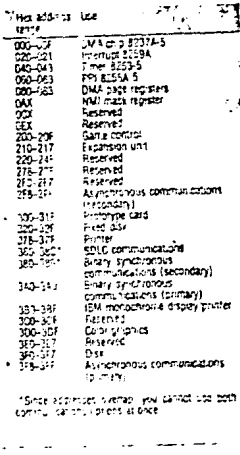


Figura 8-8. Instrucciones de E/S.



Hex address range	Use
000–01F	DMA chip 8237A-5
020–021	Interrupt 8259A
040–043	Timer 8253-5
060–063	PPI 8255A-5
080–083	DMA page registers
0A0	NMI mask register
0C0	Reserved
0E0	Reserved
200–20F	Game control
210–217	Expansion unit
220–24F	Reserved
278–27F	Reserved
2F8–2FF	Reserved
2F8–2FF	Asynchronous communications (secondary)
300–31F	Prototype card
320–32F	Fixed disk
378–37F	Printer
380–38C	SDLC communications
380–389	Binary synchronous communications (secondary)
3A0–3AF	Binary synchronous communications (primary)
3B0–3BF	IBM monochrome display/printer
3C0–3CF	Reserved
3D0–3DF	Color graphics
3E0–3F7	Reserved
3F0–3F7	Disk
3F8–3FF	Asynchronous communications (primary)

*Since addresses overlap, you cannot use both communications adapters at once

Tabla 8-4. Espacio de direcciones para los dispositivos de la IBM PC.

Primero examinaremos la estructura general de todos los programas en 'C'.

Listado 1

```
/************************
* Estructura general*
************************/
#include <stdio.h>
main ()
{
/************************
Codigo
************************/
}
```

El archivo stdio.h comprende la información necesaria para aceptar datos del teclado y su despliegue en el monitor.

Para el uso de las variables y tipos de datos revisar [5.6]

LECTURA DEL STATUS DE UN PUERTO DE ENTRADA

La operación de cada puerto se determina por el formato de una palabra de 8 bits en el registro de control del 8255A mostrado en la figura 8-9

Ejemplo

```
/*****************
*Lectura del puerto*
*****************/
#include <stdio.h>
#include <conio.h>
main ()
{
int port_A, control_reg, word;
unsigned int contents;
/**************
Declarar el tipo de datos
**************/
port_A=768;
control_reg=771;
/*********************
Direccion del registro
*********************/
word=155;
outp (control_reg,word);
/*****************
Inicializar el registro de control
*****************/
contents=inp(port_A);
/*****************
Leer puerto A
*****************/
printf("puerto A contiene %d\n",contents);
```

Conversion sincronizada en un A/D. El programa de la figura 8-9, sincroniza la operación del convertidor A/D, la captura de datos y la rutina de despliegue.
Normalmente los convertidores tienen 2 patillas una para "inicio de conversion" y la otra "fin de conversion". la primera señal la provee la interfaz, y la misma interfaz debe recibir la señal de "fin de conversion" la cual depende del tiempo de conversion requerido por el A/D.
Si eliminamos el uso de estas señales, ya que podemos confiar en la hoja de especificaciones del fabricante, donde asegura el tiempo de conversion del A/D, por tanto, podemos omitir, tanto el envio de la señal "inicio de conversión" asi como la recepción de la señal "fin de conversión", la cual puede estar asociada al registro de status para checar su activación.

Tambien es posible usar una señal de activacion en forma de "burst" o una serie de pulsos para lograr una ventana de muestreo de la señal con el fin de analizarla, por medio de técnicas de Fourier.
Podemos lograr un mejor desempeño si incluimos una o varias rutinas de ensamblador para controlar al convertidor A/D, dentro del programa principal, logrando así una mayor velocidad de muestreo y procesamiento. El lenguaje 'C' estándar, tanto Turbo C como Microsoft C Compiler version 5.1, soportan las directivas #asm y #endasm para ensamblador.

PROCESAMIENTO DE SEÑALES POR MEDIO DE LENGUAJE 'C'.

El procesamiento de señales con lenguaje 'C', puede comprender desde un desarrollo de software propio hasta adaptar el software a un paquete comercial, por ejemplo una hoja de calculo especial para el despliegue de señales, conocida como DADISP.
Consideremos el problema de calcular y desplegar la función de autocorrelación de los datos capturados, el algoritmo requerido para el procesamiento de la señal para señales continuas se da por :

$$r_{xx}(\tau) = \lim_{T \to \infty} \left[\frac{1}{T} \int_{-T/2}^{T/2} x(t)x(t+\tau)\,dt \right]$$

Esta formula anterior puede reescribirse como :

$$r(n) = \frac{1}{N} \sum_{i=0}^{N-1} x_i x_{i+n}$$

Los datos son muestreados por medio de cualquier A/D que cubra los requerimientos necesarios para capturar 512 bytes de datos en tiempo real y almacenar los datos secuencialmente en un arreglo (ver figura 8-10). El programa evalua los coeficientes de correlación y los grafica en la pantalla.

El programa (ver figura 8-10) esta hecho basicamente de un par de lazos anidados, para proveer la estructura necesaria de multiplicación/sumatoria y corrimiento.

```
for (k=0;k<=255;++k)
{
sum=0; /* poner en cero*/
for (i=0;i<=255;++i)
{
sum  +=contents[i] * contents[i+k];
```

Hacemos notar que los datos capturados se almacenan en un arreglo identificado por parentesis cuadrados.

El arreglo de datos, el cual ya ha sido defasado y multiplicado a una sumatoria por medio del operador sum+=, dividiendose después por N para dar cada coeficiente de Fourier, es desplegado en la pantalla para dar la función de autocorrelación.

ANALISIS DE ESPECTROS

Como ejemplo, evaluemos la transformada de las señales en el tiempo. Iniciemos con la transformada discreta de Fourier y posteriormente con la transformada rápida de Fourier.

El proceso de la transformada continua de Fourier se representa por la expresión :

$$X(\omega) = \int x(t) e^{-j\omega t} dt$$

La transformada discreta de Fourier es :

$$X(k) = \frac{1}{N}\sum_{n=0}^{N-1} x(n) e^{-j\frac{2\pi}{N}kn}$$

Expandiendo los terminos exponenciales podemos expresar X(h) como la suma de las partes real e imaginaria.

$$\mathrm{real}[X(k)] = \frac{1}{N}\sum_{n=0}^{N-1} x(n)\cos\left(\frac{2\pi}{N}kn\right)$$

$$\mathrm{imag}[X(k)] = \frac{1}{N}\sum_{n=0}^{N-1} x(n)\sin\left(\frac{2\pi}{N}kn\right)$$

El programa de la figura 8-11, tiene como entrada el numero de muestras, hasta 32 puntos, en este caso.
Los datos capturados x(i) = 0.039 * (a- 128) se someten a una compensación (offset) y son validados antes de ser

almacenados en un arreglo. Lograremos un aumento en la velocidad de muestreo, al efectuar estos calculos después, en lugar de en tiempo real.
Para reducir el escurrimiento espectral debido a que la señal de entrada se trunca durante la adquisicion. Hemos elegido procesar los datos por medio de la ventana de Hanning como un paso anterior al procesamiento con TDF.
Los datos muestreados multiplicados por la función de ventana se almacenan en el arreglo x(i) anterior al procesamiento por medio de TDF.

TRANSFORMADA RAPIDA DE FOURIER

La TRF reconoce que muchos terminos son redundantes y pueden eliminarse. Esto resulta en una operación con aproximadamente Nlog (N) pasos.
Para entender el enfoque práctico, consideremos el comportamiento del coeficiente complejo W_N para 8 valores muestreados, N=8
W es elevado a la potencia kl, donde k e l son enteros en el intervalo de 0 a 7. La naturaleza iterativa del algoritmo tiene como resultado 64 repeticiones de W_8^{kl}
La inspección de los coeficientes calculados revela solamente 8 productos, el resultado del producto kl, sobre el rango de 0 a 7. La TRF reconoce que muchos de los calculos son redundantes y usa un proceso de decimación en tiempo para bisectar el arreglo de datos hasta que solamente quedan 2 puntos de la transformada.

Refiriendose al diagrama de la figura 8-12, notese que el efecto del algoritmo fué el de mezclar el orden de los datos de salida, escribiendo tanto la entrada como la salida en binario.
El programa de la TRF en la figura 8-13 evalua la TRF de 128 muestras, los datos que se generan por medio de síntesis dentro del programa. Cambiando el signo de la función senoidal nos permite calcular la transformada inversa de Fourier, aplicando el algoritmo de la TRF, a la salida, regenerando así los datos originales.

La interfaz puede escoger, por medio de una conexión directa, la dirección a usar dentro del espacio de E/S, que es de 0200H hasta 02FFH. la tabla 8-5 muestra las opciones de las direcciones. La tabla 8-6 ilustra los valores de la palabra de control requerida para lograr varias combinaciones de E/S.
Como se muestra en la figura 8-14, el rango de direcciones se decodifica por medio de IC2, un demultiplexor SN74S138. El integrado SN74LS138 tiene tres entradas y decodifica varias combinaciones en 8 salidas. Este integrado también tiene una entrada activa alta y dos entradas activas en estado bajo (G2A y G2B), todas usadas para habilitación
Las lineas A8 y A9 manejan las entradas de control (figura 8-13) por medio de la señal AEN (Address Enable), la cual se encuentra en estado bajo cuando el microprocesador puede

accesar el bus. cuando A8 y AEN estan en estado bajo y A9 en estado alto, el integrado IC2 decodificará las líneas de direcciones A5-A7, ofreciendo una sola salida activa baja. Si usamos el tercer contacto, escogemos la dirección base 0240H, de tal manera que accesariamos el puerto A en la dirección 0240H, el puerto B en 0241H y el C en 0242H, y el registro de control en 0243H.

Control Word		Port		
Hex	Decimal	A	B	C
80	128	Out	Out	Out
82	130	Out	In	Out
85	133	Out	Out	In
87	135	Out	In	In
88	136	In	Out	Out
8A	138	In	In	Out
8C	140	In	Out	In
8F	143	In	In	In

Tabla 8-4 Configuracion de puertos en el 8255.

	Address	
Position	Hex	Decimal
1	200	512
2	220	544
3	240	576
4	260	608
5	280	640
6	2A0	672
7	2C0	704
8	2E0	736

Tabla 8-5. Posición de conectores y direcciones de puertos.

```
/*Capturar 512 bytes*/
/*Calcular la funcion de autocorrelacion*/
/************************/
#include <stdio.h>
#include <dos.h>
#include <conio.h>
main ()
{
int *portb;
int *ddrb;
int *pcr;
int *ifr;
int i;
int k;
long int sum;
float y;
int a;
unsigned char contents [512];
char flag;
portb=(int*)65120;
ddrb=(int*)65122;
pcr=(int*)65132;
ifr=(int*)65133;
*ddrb=0;
start:clg();
mode (1);
vdu(198,32);
draw(0,0)
draw(1279,0);
draw(0,1023);
/* Captura de 512 bytes*/
for (i=0; i <=511;++i)
{
*pcr=240;
*pcr=208;
*pcr=240;

do
  flag=*ifr;
  while (!(flag& 16));

  a=*portb;
   contents[i]=a-128;
}
/*****************
* Algoritmo de autocorrelacion *
*****************/
for (k=0; k<=255;++k)
{
sum=0;
```

Figura 8-10. Programa para calcular y graficar la funcion de autocorrelación.

```
/*******************
* Poner suma en cero *
*******************/
for (i=0;i <255.-k;
{
sum+=contents[i] * contents[i+k]
}
y=102.4 * 0.00153/256*sum;
/*********************
* Coeficiente de correlacion *
*********************/
gcol(0,2);
plot (69.4*k,y);
}
goto start;
}
```

Figura 8-10. Programa para calcular y graficar la funcion de autocorrelacion (continuación)

```
/* Sincronizar  el A/D*/
#include <stdio.h>
main ()
{

int *portb;
int *ddrbp;
int *pcr;
int *ifr;
double a;
char flag;
double contents;
portb = (int*)65120;
ddrb  = (int*)65122;
pcr   = (int*)65132;
ifr   = (int*)65133;
/*************

* DECLARAR APUNTADORES*

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

* ddrb=0;
/* Puerto b como entrada*/
for (;;)
{
*pcr=240;
*pcr=208;
*pcr=240;

/**************
INICIA EL A/D
**************/

do
flag=*ifr;

/*****************
* Leer ifr *
*****************/

/*******************
* Está la bandera puesta ? *
*******************/
a = *portb;
contents = 0.039 * (a-129);
printf ("Digital i/p: %f\n", contents);
}
}
```

Figura 6-9. Programa para sincronizar el A/D e imprimir el valor decimal de la señal de entrada.

```
/*****************************
* TDF / ventana de Hanning *
*****************************/
#include <stdio.h>
#include <stdlib.h>
#include<math.h>
main()
{
int *portb;
int *ddrb;
int *pcr;
int *ifr;
int i;
int k;
int n;
int y;
int z;
double realsum;
double imagsum;
double modulus;
double angle;
double angle1;
double a;
double x[32];
double ar[32];
double ai[32];
double window[32];
char flag;
portb = (int*)65120;
ddrb  = (int*)65122;
pcr   = (int*)65132;
ifr   = (int*)65133;
ddrb=0;
printf("numero de muestras:");
scanf ("%d,&n);
start: clg();/*limpiar area gráfica*/
mode (1);
draw(0,0);/*inicio de coordenadas*/
draw(1279,0);/* eje x*/
draw(0,0);/*inicio de coordenadas*/
draw(0,1023);/*eje y*/
for (i=0;i<=(n-1);i++)
{
*pcr=240;/*iniciar ADC/
*pcr=208;
*pcr=240;
do
flag=ifr;/*leer bandera (flag)*/
while (!(flag & 16));
a=*portb;
x[i]=0..39*(a-128);
/*Almacena los datos capturados en un arreglo*/
}
```

Figura 8-11. Programa para obtener la TDF mediante la ventana de Hannning

```
for (i=-;i<=(n-1);i++)
{
angle1=360*i/(n-1);
window[i]=0.5*(1-cos(rad(angle)));
x[i]=x[i]*window[i];/*ventana de Hanning*/
printf("%f\t",x[i]);
/*Algoritmo de la TDF*/
for (k=0;k<=(n-1);k+1)
{
realsum=0;/*iniciar variable*/
imagsum=0;
modulus=0;
for (i=0;i<=(n-1);i++)
{
angle=360*i*k/n;
realsum +=x[i]*cos(rad(angle));/n;
imagsum +=x[i]*sin(rad(angle));/n;
}
modulus=realsum**2+imagsum**2;
y=1000*modulus;
draw(1023*k/(n-1),0);
draw(1023*k/(n-1),0);
}
goto start;
}
```

Figura 8-11 Programa para obtener la TDF mediante la ventana de Hanning.

```
/*Transformada Rapida de Fourier
  y su inversa en 128 puntos*/
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
main ()
{
double ar[128];
double ai[128];
int n;
int n1;
int n2;
int a;
double b;
int c;
int d;
double e;
int f;
int g;
int h;
int j;
double k;
double l;
double co;
double si;
int m;
int p;
int q;
int r;
int s;
int t;
int u;
int w;
int z;
n=128;
n1=7;
for (z=1;z<=n;z++)
{
ar[z]=z;/* Sintesis de datos*/
ai[z]=0;
}
start:printf("Escoger: TRF o ITRF (inversa)[+1/-1]\n");
 scanf("%d",&n2);
a=n;
b=360/n);
for (c=1;c<=n1;c++)
{
d=a;
a=a/2;
e=0;
for (f=1;f<=a;f++)
{
co =cos(e);
si =sin(e)*n2;
e=e+b;
```

```
u=1;
for (g=d;g<=n;g=u*d
{
u=u+1.
h=g-d+f.
i=h+a.
k=ar[h]-ar[j];
l=ai[h]-ai[j];
ar[h]=ar[h]+ar[j].
ai[h]=ai[h]+ai[j];
ar[j]=co*k+si*l;
ai[j]=co*l-si*k;
}
b=2*b.
/* Reordenar la salida mezclada*/
m=1;
p=n/2;
g=n-1;
for (r=1;r<=q;r++)
{
if(f>(m-0.1)) goto label1.
k =ar[m].
l =ai[m];
ar[m]=ar[r];
ai[m]=ai[r];
ar[r]=k;
ai[r]=l.
label1:s=p;
label3:if(s>(m-0.1))goto label2;
m=m-s;
s=s/2.
goto label3;
label2:m=m+s.
}
for (w=1;w<=n;w++)
{
/************************
No de muestra/coef. real/coef. imaginario
************************/
printf("%d\t%f\%f\n ,w-1.ar[w]/n.ai[w]/n);
}
goto start.
}
```

Figura 6-13 Programa para TRF

```
/* Captura.h*/
#include <dos.h>
/*definir la direccion*/
#define base 0x240H
/*  Este macro define la direccion del multiplexor,  de  tal
manera,  que  se puede seleccionar cual entrada va a leer el
A/D */
#define MUX(channel) outportb (base+channel,0)
#define START_AD() outport (base+9,0)
/*Esto checa PB7. Esta completa la conversion?/

#define completa() (inportb(base+0xd & 0x80)
/* Este macro da el valor de la conversion A/D*/

#define READ_AD(result) (result=(inportb(base+8)\
      <<4); result+=(inportb(base+9) >>4;)
/Este macro empieza la lectura, espera hasta que el A/D
completa la conversion, y lee el resultado*/
#define GET_VALUE(result)\
   START_AD();\
   while completa()\
     ;\
   READ_AD(result);
/**********************
*  Aqui  empieza la declaracion de funciones  definidas  en
   captura.c*/
/* Despliega el valor como un voltaje entre -5 y +5V*/
void print_value(unsigned int reading);
/*  Esta  funcion espera a que la entrada al A/D  exceda  el
  umbral  y entonces llena "buf" con muestras de "bufsize",
tomadas  a  una  velocidad  de  muestreo  controlada  por
software.*/


void  captura(unsigned *buf,  int bufsize,  unsigned umbral,
int rate);
```

Figura 8-14.Archivo de referencia para la definicion de parametros.

```
 /*captura.c*/
/*Turbo C (Borland)*/
#include "captura.h"
#include <stdio.h>
/*Imprime los ceros y unos, para poder observar el patron de
bits del convertidor A/D.*/
void print_binary(unsigned int binary) {
int i;
for (i=15;i>=0;i--)
putchar(binary&(binary & (1<i) ? '1':'0');
}
/* Esta es una secuencia de prueba/*
/*imprime el valor como un voltaje para comparar con un
voltmetro*/
void print_value(unsigned int reading) {
if (reading & 0x0800) {
putchar('+');
reading &=0x7ff;
}
else {
  putchar('-');
 reading=(-reading)&0x07ff;
}
/*Ahora usamos la escala e imprimimos el valor */
printf("%1.3f,5.0 *(float)reading)((float) 0x7ff));
}
/* Captura un conjunto de muestras /*
void captura (unsigned * buf,int bufsize,unsigned umbral,int rate){
unsigned int sample;
int i,j;
/*primero, esperamos a que se exceda el umbral*/
while (1) {
GET_VALUE(sample);
/* El bit mas alto (MSB) se anade al umbral para hacer un valor
positivo */
if (sample(umbral : 0x800))
break;/*salir*/
}
/*captura un conjunto de muestras*/
for (i=0;i<bufsize;i++) {
GET_VALUE(buf[i]);
for (j=0;j<rate;j++)
; /*retardar por un factor 'rate'"*/
}
}
```

Figura 8-15. Programa para la adquisicion de datos.

```
/*display.c*/
/*  funciones  que  soportan el despliegue de señales  en  la
pantalla usando Borland BGI*/
#include <stdio.h>
#include <stdlib.h>
include <stdarg.h> /*va_start(), etc.*/
#include <string.h>
include <math.h>
#include "captura.h"
#include "display.h"
void gprintf(char * format,...) {
static int textline=0;
char textbuf[80];
va_list argptr;
va_start(argptr, format);
vsprintf(textbuf, format,argptr);
va_end(argptr);
outtextxy(0,textline,testbuf);
textline +=textheight("X");
}
void display_series(unsigned *series,  int series_size, int
offset,  scale_factor  vertical_scale_factor,  scale_factor
horizontal_scale_factor, int color){
int i,j,step;
/* crea una copia local del arreglo*/
int *lseries= (int*)calloc(series_size,sizeof(int));
memcpy(lseries,series, serie_size *sizeof(int));
/*primero,  efectuar  escalamiento  vertical  del  arreglo
local*/
switch (vertical_scale_factor ){

case quarter:for(i=0;i<series_size;i++)
     lseries[i]>>2;/*divide por 4*/
     break;

case half:for(i=0;i<series_size;i++)
          lseries[i]>>1;/*divide por 2*/
          break;
case full: break/*no hay cambio*/
case twice:for(i=0;i<series_size;i++)
           lseries[i]<<1;/*multiplica por 2*/
           break;
case quad: for(i=0;i<lseries_size;i++)
           lseries[i]<<2;/*multiplica x 4*/
}
if(offset)
for(i=0;i<series_size;i++)
lseries[i] +=offset;
/*Despliega  el  arreglo  de  acuerdo al  factor  de  escala
horizontal.*/
```

Figura 8-16. Programa para el despliegue de gráficos.

```
switch(horizontal_scale_factor){

case quarter: for(i=0;i<(series_size/4);i++)
              putpixel(i<<2,lseries[i],color)
              break;

case half:    for(i=0;i<(series_size/2);i++)
              putpixel (i<<<1,lseries[i];color),
              break;

/* Despliega todo el arreglo en escala 1:1*/

case full  for(i=0,i<series_size,i++)
           putpixel(i) lseries[i],color);
           break;
/*Desplegar en un espacio menor*/

case twice  for(i=0;i<(series_size/2),i++)
            putpixel(i lseries[i]<<1 color),
            break;

case quad  for(i=o i<(series_size/4,i++)
           putpixel(i lseries[i<<2],color)
           break
}
  free(lseries),/*libera la copia local del arreglo*/
}
```

Figura 8-16. Continuación

```
display.h
/* Declaraciones de funciones para reportar despliegue
gráfico de la señal*/
#include <graphics.h>  /*turbo C BGI*/
/* Una función similar a printf pero en modo gráfico*/
void gprint(char * format, ...);

void display_viewsetting(struct viewporttype view);
/* tipos de escalas para "display_series"*/
typedef enum{quarter, half,full, twice,quad}scale_factor;
/* Despliegue de una serie de números como puntos en la
pantalla*/
void display_series(unsigned *series,int series_size,int offset,
                    scale_factor vertical_scale_factor,
                    scale_factor horizontal_scale_factor,int color);
```

Figura 8-17. Declaración de funciones para despliegue.

voz.c

```
/*Despliega  un  patrón de voz usando funciones de  Borland
BGI*/
#include <conio.h> /*kbhit*/
#include <stdlib.h>/*calloc()*/
#include "captura.h"
#include "dispaly.h"/*incluye graphics.h*/
main () {
int graphdriver= DETECT; /* usara autodetección*/
struct viewporttpye view;
int i,graphmode, color;
unsigned int * points;
/*  inicializa gráficas.  la cadena en el  tercer  argumento
dice a la función donde buscar los drivers gráficos*/
initgraph (&graphdriver, &graphmode, "c:\\turboc");
color = getmaxcolor();
getviewsettings(&view);
/*display_viewsettings(view)*//*para debug*/
/*  crear un arreglo de acuerdo al tamaño de los pixels  del
monitor*/
points= (unsigned int *)calloc (view.right, sizeofint));
MUX(0); /*select el canal a leer*/
while (!kbit)){
captura(points, view.right, 1000,0);
for (i=0;i<view.right;i++)
   points[i] >>3;/*escalamiento básico*/
display_series(points, view.right  20, half, full, colors);
clearviewport();
}
  closegraph();
}
```

Figura 8-18. Programa de muestra para probar el sistema de adquisicion de datos.

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>

#define N 128
#define PI2 6.28


XREAL[N], XIMAG[N];
int N,N2,l,k,i;
double NU,NU1;
 N2=N/2;
NU1=NU-1;
K=0;
for(l=0;l<=NU;l++){
label1: for(i=1;i<=N2,i++)
{
p=ibitr(k/pow(2.0,NU1),NU)
arg=PI2*P/N;
c=cos(arg);
s=sin(arg);
K1=K+1;
K1N2=K1+N2
TREAL=XREAL[k1N2]*c+Ximag[K1N2]*s;
TIMAG=Ximag[K1N2]*c-Xreal[K1N2]*s
XREAL[K1N2]=XREAL[K1]-TREAL;
XIMAG[K1N2]=XIMAG[K1]-TIMAG;
XREAL[K1]=XREAL[K1]+TREAL;
XIMAG[K1]=XIMAG[K1]+TIMAG;
K=K+1;
}
K=K+N2;
if(K<N) goto label1
K=0;
NU1=NU-1;
label2: N2=N2/2;
for(K=1,k<N;K++)
{
i=ibitr(k-1,NU)+1;
if(i<k) got label3
TREAL=XREAL[K],
TIMAG=XIMAG[k];
XREAL[k]=XREAL[i];
XIMAG[k]=XIMAG[i];
XREAL[i]=TREAL=TREAL;
XIMAG[i]TIMAG;
int ibitr(int j, int NU);
j1=j;
ibitr=0;
for(i=0;i<=NU;i++)
{
j2=j1/2;
ibitr=ibitr*2+(j1-2*j2);
}
```

Figura 8-19. Programa para el calculo de TRF

El primer problema en la adquisición de datos es realmente, la forma de adquirirlos.
El acceso directo a memoria (DMA) ofrece una solución con relativa simplicidad en el hardware.
Con el uso de DMA, los datos producidos por un dispositivo externo (convertidor A/D y su respectiva lógica) se inyectan directamente a la memoria de la PC, sin intervención directa del CPU. También el DMA puede operar en la dirección inversa : desde la memoria de la PC a un dispositivo externo o inclusive de un área de memoria a otra.
Mientras se efectua la transferencia de datos, el CPU está libre para efectuar otras tareas. Su velocidad de procesamiento se degrada a causa de los "robos de ciclos" que efectua el DMA. Esta modalidad de liberar al procesador nos permite asociar esta idea con el concepto de multitarea, esto es, la transferencia se efectua como un proceso en el fondo o segundo plano y el análisis y despliegue de datos en primer plano. Obviamente la eficiencia y complejidad de estos procesos está en relación directa con el tipo de procesador y del sistema operativo.

ARQUITECTURA DEL DMA EN MICROCOMPUTADORAS

En una microcomputadora IBM o compatible tenemos disponibles la mayoría de las señales directamente del bus para llevar a cabo transferencias por DMA. Solo tenemos que diseñar una circuiteria extra para el manejo DMA y usar los datos de una fuente externa o también incluir la adquisición de datos de la misma tarjeta.

En una microcomputadora compatible IBM, hay un controlador DMA 8237A (Intel). Un circuito integrado 8237A controla 4 canales, del 0 al 3, cada uno puede programarse en forma individual y los cuatro canales activarse simultáneamente.
En una microcomputadora, el canal 0 se usa para refrescar la memoria dinámica, y sus señales de control no están presentes en el bus. El canal 2 lo usa el floppy (controlador de unidad de disco flexible) y el canal 3 lo utiliza el controlador de disco duro. Esto nos deja solamente el cana 1 disponible, aunque debemos considerar que algunos dispositivos como son unidades de cinta, tarjetas de red, etc, utilizan este canal.
En una microcomputadora compatible AT (procesador 80286), tenemos dos integrados 8237 disponibles, de esta manera se liberan algunos "cuellos de botella".
Las señales de control están disponibles solo en el conector extra del bus AT. La mayoría de las tarjetas comerciales (controladores de cinta, tarjetas de red, etc) tienen micro-interruptores que permiten la selección de un canal libre.
En una AT, el canal 2 se usa para llevar la unidad de disco flexible, y el canal 4 para enlazar las dos controladoras. Tenemos disponibles los canales 0,1,3 y 5-7 para su uso.
El 8237 es un integrado complejo que ocupa un bloque de 16

direcciones en el espacio de E/S. La direccion base del controlador primario es cero, de tal manera que se utilizan las direcciones de 0 a FF (hex). La dirección secundaria (AT) para el segundo controlador es C0 (hex), pero involucra palabras en lugar de bytes, y ocupa las direcciones C0-DE (hex). La figura 8-20 muestra la asignacion de puertos en el 8237.

PC only address	AT only address	Description of register contents
0	C0	Channel 0 base address (write 2 bytes, LSB first)
1	C2	Channel 0 count (write 2 bytes, LSB first)
2	C4	Channel 1 base address
3	C6	Channel 1 count
4	C8	Channel 2 base address
5	CA	Channel 2 count
6	CC	Channel 3 base address
7	CE	Channel 3 count
8	D0	Command status register (not used in this application)
9	D2	Request register (not used)
A	D4	Single mask register (write only) 7 6 5 4 3 2 1 0 channel select (0-3) mask bit value (0-1)
B	D6	Mode register (write only) 7 6 5 4 3 2 1 0 channel select (0-3) 01=write, 10=read 0=disable autoinitialization, 1=enable 0=address increment, 1=decrement 01=single transfer mode
C	D8	Clear byte pointer flip-flop (write only)
D	DA	Master clear (not used)
E	DC	Clear mask register (not used)
F	DE	Write all mask registers (not used)

Figura 8-20. Asignacion de puertos del 8237 DMA.

La siguiente es la secuencia de operaciones involucradas en una transferencia de DMA :
El software programa el modo de transferencia (lectura o escritura), la direccion de inicio del area de memoria desde o hacia la cual se efectua la transferencia, el número de bytes (palabras, en el caso del segundo controlador) a ser transferidos, y luego limpia el bit de mascara del canal, habilitando al hardware para transferir datos.
El hardware hace una peticion de un ciclo de DMA, levantando la señal (DMA request line) DRQ.
Para el canal en un alto nivel. El controlador DMA responde poniendo en estado bajo (0) la señal DACK (DMA acknowledge line), normalmente en estado alto para ese canal. Esta señal puede usarse en el siguiente ciclo del bus, por medio del

hardware para transferir un byte o palabra. Si el controlador se programa para escritura, cuando la señal DACK se pone en estado bajo el hardware puede colocar el siguiente valor en memoria. El controlador efectua lo necesario como son: incrementar la localidad de memoria, manejo de lineas de direccion, y manejar las lineas de control al CPU.
La situación es más difícil si el programa desea notificación cuando se ha concluido la transferencia del DMA. El DMA levanta ("1") una señal de "cuenta total" (total count) TC junto con la ultima señal DACK. La señal TC esta disponible en el bus, y se puede alambrar con alguna de las interrupciones IRQ, causando una interrupción cuando la transferencia se completa. La rutina de interrupción por software puede programar el DMA para arreglar la siguiente transferencia, creando un lazo o proceso iterativo y haciendo una tarea que opera en forma indefinida.

DISEÑO DE UN CIRCUITO INTERFAZ DMA

La figura 8-21 muestra un circuito que efectua todo el enlace de señales (handshake) descrito anteriormente.
Junto con el software descrito,conforma una alternativa para adquisición de datos; funciona cuando los datos incluyen una señal de reloj byte a byte.
El corazón del circuito es un integrado FIFO o stack (U1 en figura 8-21), un circuito que implementa en hardware el concepto de primera-entrada primera-salida.
Hay 8 lineas de datos paralelos que van hacia el FIFO y 8 señales saliendo del mismo. Los bytes entrantes se escriben al FIFO por medio de la patilla de escritura "W" y los bytes se leen de la misma manera que fueron escritos por medio de la patilla de lectura "R". Ambas patillas son activas en estado bajo. El integrado FIFO (U1) actua como enlace o buffer entre la fuente de información y la computadora.
El FIFO provee una bandera de "vacio", que es una señal que está en estado bajo cuando el FIFO esta vacio y en alto cuando ("1") cuando el FIFO tiene datos.
Esta señal se usa directamente para manejar la señal DRQ del canal 1. Esto significa que mientras exista un dato en el FIFO, habrá peticiones para transferir datos. Suponiendo que el controlador de DMA ha sido programado en forma correcta por software, habrá señales de reconocimiento por medio de DACK1. Cada señal DACK (activa en estado bajo), habilita la señal de lectura FIFO. La señal de lectura en el FIFO, lo cual obliga a que aparezca el siguiente byte en las lineas de salida del FIFO. La senal DACK tambien activa el siguiente integrado (U2), que es un buffer "tres estados" cuya salida se conecta al bus de datos de la microcomputadora. Cuando el buffer no está habilitado (la señal DACK en estado alto), sus lineas de salida están inactivas y no interfieren con la operación del bus. Cuando DACK está en estado bajo, el buffer tiene sus 8 lineas activas sobre el bus, donde el DMA recibe dichas lineas en la memoria.

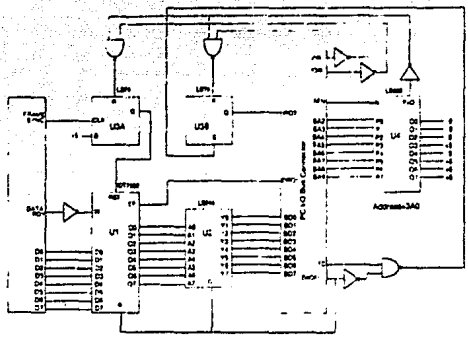


figura 8-21. Circuito para el manejo de DMA.

El numero de bytes que se transfieren de esta manera es dependiente de la cuenta programada en el controlador de DMA.
En la ultima transferencia el controlador de DMA pone la señal TC en estado alto. Hay solamente una señal TC para todos los canales de DMA, asi que para distinguirlos de otras operaciones propias del DMA, es alambrado lógicamente como una AND con la señal DACK negada. Esto habilita un Flip-Flop (U3B), cuya salida va directamente a IRQ3 del bus de la PC, causando así una interrupción al CPU.
Esta es la misma interrupcion que usa COM2, por lo tanto debemos tener cuidado con los conflictos de interrupciones.
Para asegurarnos que reconozca la linea de interrupción, la linea de interrupción debe mantenerse en estado alto hasta que el CPU transfiere el control al servicio o rutina de interrupcion. Desafortunadamente, no hay una manera para determinar lo que ISR está ejecutando, así que el software debe notificar en forma explicita al hardware cuando es tiempo de poner en estado bajo la señal IRQ. Esto se efectua por medio de la creación de un puerto de E/S.
Cuando el software lee o escribe desde o hacia un puerto de E/S, el CPU coloca la direccion del puerto de E/S en el bus de direcciones del BUS, junto con la señal AEN además de una señal ya sea IOR o IOW, dependiendo de si el puerto de E/S es leido o escrito. Normalmente, cualquier dispositivo que reconoce su propia dirección de E/S usa el siguiente ciclo del bus para transferir un byte de datos, ya sea desde o hacia el CPU. Sin embargo, no es necesario que todos los datos sean transferidos.

El circuito aprovecha esto por el hecho de que ha ocurrido un ciclo de lectura en su propia dirección para limpiar la petición de interrupción.
El comparador de direcciones en U4 levanta su línea de salida cuando su dirección alambrada aparece en el bus junto con AEN. Por claridad, se decodifica la dirección en forma parcial. Esto es, cuando los bits 9-2 de la dirección son "11101000" si tomar en cuenta el resto de los bits, el decodificador se pone en estado alto. La palabra binaria 11101000XX es el hexadecimal 3A0-3A3, así que cualquiera de estas direcciones funcionara (también 7A0-7A3, BA0-BA3, y FA0-FA3). Normalmente, estas direcciones no se usan en una PC, pero podemos lograr otras direcciones. Para finalmente apagar el flip-flop que maneja el IRQ, la salida del decodificador es alambrado como compuerta AND con IOR. El resultado es que el programa puede leer el puerto de E/S 3A0 para limpiar la petición de interrupción.
Usamos un bit para restablecer al FIFO a su estado inicial. Este 'reset' se efectua al principio de la adquisición de datos para borrar los datos anteriores o reiniciar el llenado del FIFO con nuevos datos, lo que marca el principio de un nuevo bloque de datos. Esta vez usamos la misma decodificación del puerto de E/S junto con IOW, para habilitar el Flip-Flop en U3A, poniendo en estado bajo la señal de 'reset' en el FIFO. El software puede efectuar este 'reset' escribiendo en el puerto 3A0. El flip-flop se borra al siguiente pulso de sincronía, restableciendo el FIFO y permitiendo la entrada de datos.

El software debe programar de inmediato el controlador del DMA para empezar una transferencia, antes que el FIFO pierda datos.

PROGRAMACION DE LA INTERFAZ DMA

El software que hace funcionar al sistema es programa dma.c donde están todas las subrutinas para configurar y arrancar el hardware, y el segundo es un programa de muestra para ilustrar el harware en operación.

El software utiliza dos buffers (revisar capitulos VI y VII). Mientras el DMA esta llenando un buffer, el contenido del otro buffer está disponible al programa para su procesamiento. Cuando esta completa la transferencia, los buffers se intercambian y se inicia una nueva transferencia.
El tamaño de los buffers se escoge de tal manera que haya tiempo suficiente para que el programa procese mientras se intercambian los buffers. A una tasa de 64 Kbits por segundo hay 8 segundos de datos en un buffer de 64 KBytes.

Aunque se puede hacer mucho procesamiento en 8 segundos, no es un tiempo muy grande. Si un programa necesita hacer rompimiento de números(calculos numéricos), su primer acción, cuando los buffers se intercambian, es copiar algunos o todos los nuevos buffers en un área de memoria y trabajar en dicha área. La adquisición de datos es totalmente manejada por interrupción. Puede pensarse en una tarea de mayor prioridad en primer plano, con los datos procesadonse en segundo plano. El programa principal llama a alloc_dma_buf para colocar los buffers. El tamaño del buffer en bytes, se especifica en la variable global buf_size. Por la forma en que un controlador de DMA funciona, no permite que ningun buffer ocupe la frontera de una página de 64KB, en otras palabras, cada buffer del DMA debe estar comprendido en el rango XOOOO-XFFFF (Hex) donde X indentifica una de las 10 paginas de 64 KB en el rango de 640 KB de la memoria base de la PC. Este requisito es difícil de cumplir y lo efectua alloc_dma_buf() usando las funciones propias de DOS para alojamiento de memoria para reservar toda la memoria disponible.
El programa llama despues a dma_conf() para preparar el controlador DMA y habilitar la rutina de interrupción. El número de canal se asigna con dma_chan() y el número de interrupción en dma_irq.
La rutina dma_conf localiza un cierto numero de registros (direcciones de puertos) que se usaran después. Un controlador de DMA pude manejar paquetes o bloques de 64KB. Para tener acceso a los 640 KB de espacio, los 4 bits de la parte alta de la dirección de transferencia se colocan en el registro de pagina. Hay un registro de pagina por cada canal del DMA, como se muestra en la figura 8-22. La dirección correcta del registro de pagina se selecciona por medio de dma_conf(), asi como las direcciones de la base, de la

cuenta, de la mascara, y del registro de modo de la figura 8-20.

Channel Number	Page Register Address (hex)
0	87
1	83
2	81
3	82
4	8F
5	8B
6	89
7	8A

Figura 8-17. Asignación de registros de página.

El siguiente paso usa una caracteristica no documentada en el sistema operativo DOS. Ya que el DOS no es reentrante, debemos tener cuidado al llamar las funciones de DOS desde una rutina de interrupción. La función de DOS no documentada 34 (hex) regresa el par de registros ES:BX. la direccion de la bandera de sección crítica. Cuando el byte en esta dirección es cero (0), es seguro llamar una función de DOS. Cuando el byte es diferente de cero, el dos está ejecutando una sección crítica del codigo. En la rutina dma_conf(), la dirección de esta bandera se encuentra y se guarda para su uso posterior.
El paso final en dma_conf es colocar la dirección de la rutina de interrupción dma_isr(), en el vector especificado, y habilitar la interrupción limpiando el correspondiente bit de mascara en el controlador de interrupciones (revisar capitulos VI y VII).

INICIO DEL PROCESO DE ADQUISICION DE DATOS.

Para iniciar la adquisicion de datos, el programa llama a la rutina start_dma(). Requiere de dos argumentos : la dirección del buffer y el numero de bytes a transferir.
La rutina start_dma() programa al controlador del DMA con la dirección del buffer y la cuenta, usando las direcciones de registros calculados en dma_conf(). El 8237 se programa byte por byte, con el byte menos significativo primero. Para asegurar un correcto orden, un registro en el 8237 restablece el flip_flop a su estado inicial, pero es seguro asumir que en una PC este Flip-Flop está limpio asi que puede escribirse el byte menos significativo primero.
Primero, el bit de mascara en el 8237 el canal elegido se

pone en "1", deshabilitando el DMA en este canal mientras el 8237 está siendo programado.
A continuación la dirección del buffer se divide en 3 bytes por medio de algunos de los 2 metodos mostrados en la figura 8-23. El metodo a usar depende de si estamos manejando bytes o palabras.

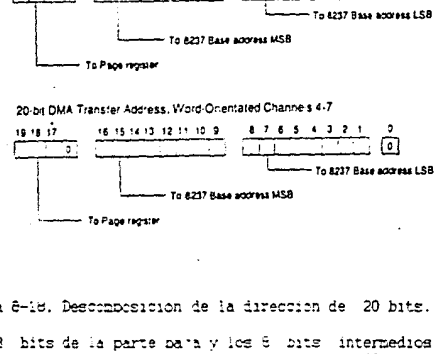


Figura 8-18. Descomposicion de la direccion de 20 bits.

Los 8 bits de la parte baja y los 8 bits intermedios se cambian al registro de la direccion base del 8237 en dos operaciones sucesivas. Los bits de la parte alta se escriben al registro de página. A continuación, la cuenta de bytes se divide en dos para canales del tipo palabra, entonces se decrementa y se envia al registro de cuenta del 8237 en dos escrituras.
La razon por la que se decrementa es que la cuenta total (TC) se alcanza cuando la cuenta remanente va de 0 a FFFF. Finalmente, el bit de mascara de canal se borra, permitiendo al DMA empezar.
Cuando el ultimo byte o palabra se transfiere, la tarjeta de interfaz produce una interrupcion al CPU.
Ya que la direccion de dma_isr() se colocó en el vector de interrupción correspondiente, esta rutina entra tan pronto como ocurre la interrupcion. Declarando la función con el tipo de interrupcion, el codigo de entrada usual en C se reemplaza por el codigo de la interrupcion entrante: los registros se guardan y se cargan los segmentos de datos correctos. La funcion regresa al restablecer los registros y emitir una instruccion IRET. El segmento de stack puede no ser el mismo que el segmento de datos que sería lo usual en el codigo del modelo pequeño, así los apuntadores a variables locales no son permitidos en un contexto de

interrupciones.
En dma_isr() la variable curr_buff apunta al buffer que acaba de ser llenado. El programa principal puede usar esta dirección para accesar los datos. El índice de buffers se activa para intercambiar buffers y se llama a start_dma(), pasando la dirección al siguiente buffer. Así, dma_isr(), propaga su propia ejecución ya que cuando el siguiente buffer se llena, la interrupción ocurrirá y el dma_isr() se llama otra vez. Antes de salir, la rutina de interrupción borra su propia petición de interrupción y saca la señal de fin-de-interrupción al controlador de interrupciones. Se ha declarado la función reset_irq() en el programa principal en lugar de aquí porque es dependiente del hardware, la función en dma.c trabaja para cualquier configuración del harware.
Si la variable file_handle es diferente de cero, se asume que file_handle se asignó a un nuevo archivo por medio de open().
A continuación llamamos a write_buf() para escribir el buffer que se acaba de llenar al archivo en el disco representado por file_handle. Aquí es donde la bandera de la sección crítica del DOS se pone en uso. Si la rutina de interrupción interrumpe al DOS en una localidad crítica, la escritura a disco es omitida y el buffer se pierde.

```
/* dma.c -- subrutina para adquisicion de datos*/

/*compilador Microsoft versiòn 5.0*/
/* poner el interruptor /Gs para manejo de stack*/

#pragma check_stack(off)
#include <dos.h>
/*definiciòn de registros en DMA*/
#define DMA0_BASE  0x00  /*direcciòn del controlador de DMA*/
#define DMA1_BASE  0xC0  /*direcciòn del controlador (canal 4-7)*/
/*Definiciòn del controlador de interrupciòn*/

#define   INTA00      0x20  /*Direcciòn base del controlador de
                             interrupciòn*/

#define   INTA01      0x21  /*direcciòn base del segundo registro*/

#define EOI           0x20  /*còdigo para el fin de interupciòn*/

/* variables externas */
extern   char   fat  *dma_buffers[];   /*arreglo con buffers de
                                       direcciones*/
extern int  buf_index;                  /*indice del buffer actual*/
extern char fat *curr_buf;              /*apuntador al buffer recien
                                        llenado*/
extern  usigned buf_size;               /* tamaÑano del buffer  en
                                        bytes*/
extern   int  lost_buffers;              /*cuenta de buffers que no
                                         han sido escritos*/

extern int dma_irq;                      /*interrupciòn h/w con DMA
                                         completo*/

extern int dma_chan;                     /*número de canal*/
extern int file_handle;                  /*manejo de archivo*/
extern int  reset_irq();                 /*función para borrar la
                                         petición de interrupción*/

/* variables locales*/
/* para evitar uso excesivo de stack*/

static union REGS r;      /* registros generales*/
static struct SREGS s;    /* registros de segmentos*/
static int sel;           /* bits de selecciòn de canales*/
static int basereg;       /* registro de direcciòn base*/
static int cntreg;        /* registro de cuenta*/
static int maskreg;       /* registro de mascara*/
static int pagereg;       /* registro de pàgina*/
static int modereg;       /* registro de modo*/
static int page_tbl[] = /* tabla de direcciones del
                           registro de pàgina*/

     {0x87, 0x83, 0x81, 0x82
      0x8f, 0x8b, 0x89, 0x8a};
char far *dos_crit_addr;     /*Direcciòn de la seccion critica*/
```

```
static void
       (interrupt far *dma_int_save)();
/*macros para extraer bytes de las direcciones de 20 bits*/

#define LSB(x) *((unsigned char *) &x)
#define MSB(x) *(((unsigned char *) &x)+1)
#define PAGE(x) *(((unsigned char *) &x)+2)

/* prototipo de funciones*/

void dma_conf(void);
void dma_finish(void);
int alloc_dma_buf(void);
void start_dma(char far *, unsigned);
void interrupt far dma_isr(void);
int write_buf();
/*****************************************/
int alloc_dma_buf() /*coloca un par de buffers al DMA*/
{
   unsigned buf;    /* variable temporal*/
   unsigned max;    /* direccion de parrafo*/
   unsigned seg;
   unsigned size;   /*tamaño de buffer */

/*  Esta rutina coloca un par de buffers que pueden ser llenados por
el DMA*/

size = (buf_size ==0) ? /*convierte bytes a parrafos*/
          0x1000: buf_size >>4; /*divide por 16*/
_dos_allocmem(0xffff,&max);  /*el maximo de parrafos de DOS*/
_dos_allocmem(max,&seg);  /* todos */

buf=seg;       /*intento inicial al segmento de buffer*/

if ( ((buf + size -1) & 0xf000)   /* si el buffer cruza la */
        != (buf & 0xf000)         /* la frontera fisica de página*/
       buf=  (buf  & 0xf000) + 0x1000;  /* ajusta  a  la  siguiente
                                          página*/
dma_buffers(0)= (char far *)   /* convierte segmentos de buffers */
     ((long) buf <<16);        /* a apuntadores para retorno*/

buf+=size;                       /*intento   inicial   al   siguiente
                                 buffer*/

if( ((buf + size -1 ) & 0xf000)
          != (buf & 0xf000) )
        buf  =  (buf  & 0xf000) +  0x1000;  /*ajusta  si  cruza  la
                                            frontera de página*/

dma_buffers[1] = char far * )          /* regresa como apuntador*/
          ((long) buf <<16);

size = buf + size -seg;             /*calcula el tamaño necesario*/
return                              /*libera la memoria sin ocupar*/
```

```
_dos_setblock(size,seg,&max);/*regresa un error si no es
                              suficiente*/
}
*-------------------------------------------*
void dma_conf()
int intmsk;

sel = dma_chan & 3;      /*aisla los bits de selecion de canal*/
pagereg = page_tbl(dma_chan);   /* coloca registro de página*/

if (dma_chan<4)

      basereg= DMA0_BASE +sel +2;   /*controlador de DAM estándar*/
      cntreg= basereg +1;
      maskreg= DMA0_BASE +10;
      modereg= DMA0_BASE + 11;
}
 else
  {
/*
   basereg =DMA1+BASE + sel +4;  /*solo controlador alterno (AT)*/

   cntreg= basereg +2;

    maskreg= DMA0_BASE + 20;

    modereg= DMA1_BASE + 22;
*/
  }

r.h.ah = 0x34;      /*función críticas de DOS*/

intdosx(&r,&r,&s);
dos_crit_addr = (char far *)   /* guarda su dirección para prueba*/
    (((long) s.es << 16) | r.x.bx); /* como un apuntador*/

if (dma_irq <0 || dma_irq >7) /*validar número de interrupción*/
       return;
dma_int_save =                /*guarda los contenidos en uso del
                                vector de interrupción*/

  _dos_getvect(dma_irq + 8);
  _dos_setvect(dma_irq+8,  dma_isr);   /*configura  nueva rutina  de
                                        interrupción*/
 int_msk  =  inp(INTA01);               /*obtener  la  mascara  de
                                        habilitación  de  interrup-
                                        ción*/
 int_msk &=~(1 << dma_irq);             /*borra el bit de mascara en
                                        la interrupción del DMA*/
outp(INTA01,intmsk);                    /*sacar  la nueva  mascara,
                                         habilitando   la interrup-
                                         ción*/

atexit(dma_finish);
}
/*-----------------------------*/
```

```
static void dma_finish()    /*llamada via atexit()*/
{
    int intmsk;

    intmsk= inp (INTA01);    /*obtener la mascara de habilitación
                              de interrupción en uso*/
    intmsk  |=  (1 << dma_irq);  /*poner la mascara de bits  para  la
                                  interrupción del DMA*/
    outp(INTA01, intmsk);       /*coloca nueva mascara y
                                 deshabilita interrupción*/
    _dos_setvect(dma_irq+8,dma_int_save); /*restablece el contenido
                                           del vector anterior*/
}
/* ---------------------------------*/
void start_dma(buf,count)     /*empieza operación del DMA*/
char far *buf;                /*dirección del buffer a ser llenado*/
unsigned count;
{
   int page;
   unsigned long addr=     /*dirección de 20 bits del buffer del
                           DMA*/
        FP_OFF(buf) +
        ((long) FP_SEG(buf) << 4;

 page= PAGE(addr);   /*extrae los bits superiores de la dirección*/

  if(dma_chan >=4)
  {
      count >>= 1;       /*convierte la cuenta a palabras*/
      addr >>=1;         /*convierte dirección a palabras*/
      page &= 0x7e;      /*el bit 16 de dirección está en 'addr'*/
   }
  count--;              /*calcula count-1 (xfr se detiene en ffff)*/
  outp(maskreg,  sel | 0x04); /*coloca el bit de mascara para
                              deshabilitar el dma*/
  outp(modereg,  sel | 0x44); /* xfr mode (sngl,inc,noint,write)*/
  outp(basereg,  LSB(addr));  /*dirección base lsb)*/
  outp(basereg,  MSB(addr));
  outp(pagereg,  page);
  outp(cntreg,   LSB(count));/*manda la cuenta de lsb*/
  outp(cntreg,   MSB(count));
  outp(maskreg, sel         );/*limpia el bit de mascara, habilitando
                               el DMA*/
}
/*-------------------------*/
static void interrupt far dma_isr()
cur_buf = dma_buffers[buf_index];       /*dirección de buf lleno*/
buf_index^=1;                           /*índice del buffer sig.
start_dma  (dma_buffers[buf_index],buf_size);  /*empieza dma en sig.
                                                 buffer*/
if(file_handle)                        /*si el disco es habilitado*/
   write_buf();                        /*escribe al disco*/
reset_irq();                           /*reset especifico*/
outp(INTA)),EOI);                      /*señal de fin-de-interrup.*/
}
```

```
/****-------------------------------*/
static int write_buf()      /*escribe buffer al disco*/
{
if (*dos_crit_addr)         /*primero checa la sección crítica*/
{
    lost_buffer++;            /*si  está  colocado.  se evita  la
                               escritura*/
      return 0;
}
    r.x.dx =FP_OFF(curr_buf);    /*se escribe ....coloca direcc.  en
*/
    s.ds    =FP_SEG(curr_buf);     /*los  registros  adecuados  para
                                   llamadas al sistema operativo*/
    r.x.bx = file_handle;         /*coloca file_handle para  escribir*/
    r.x.cx = buf_size;            /*coloca la cuenta de bytes */
    r.h.ah = 0x40;                /* función de escritura en dos*/
    if(intdos(&r, &r, &s) == buf_size   /*checa valor de retorno*/
        && r.x.cflag == 0         /*bandera de acarreo*/
        return 0;
  else
  {
    lost_buffers++;             /*no logró escribir*/
 return 1;
  }
}
```

```
/***-------------------------------------------*/
/*prueba.c*/
/* compilador microsoft C versión 5.0
/* compilar con la opción /Gs  ya  que
  reset_irq() se llama desde una interupción*/
# include <bios.h>
int far *dma_buffers[2];   /* apuntadores a dos buffers*/
int far *curr_buf;         /*apuntador al buffer en curso*/
int buf_size;              /* tamaño del buffer*/
int buf_index;              /*índice del buffer en uso*/
int dma_irq=3;
int dma_chan=1;
int file_handle=0;
int lost_buffer=0;

#define FRSIZE 64;   /*palabras por cuadro*/
#define NUMFR  8    /* Cuadros por buffer de DMA*/

/*-----------------------------*/
main()
{
int temp;
int i;
unsigned char frame;
int far *cp;

   reset_irq();   /* borra interrupción*/
   buf_size  =  FRSIZE  *  NUMFR  *  sizeof(int);   /*Averigua
                                                     tamaño de
                                                     buffer*/
   alloc_dma_buf();                  /*aloja buffers*/
   printf("buf1= %p buf2= %p\n",
            dma_buffers[0],dma_buffers[1]);
     dma_conf();

     outp(0x3a0,0);              /* reset a FIFO en la interfaz*/
     start_dma(dma_buffers[0],buf_size);     /*  empieza  la
                                          adquisición*/
     file_handle = create("tmp.dat");
       temp= buf_index;

     while  (!_bios_keybrd(KEYBRD_READY))     /*Detenerse  en
                                          con   la   sig.
                                           tecla*/

   {
       if(temp  != buf_index)        /* espera a que  el  DMA
                                     termine*/
         {
           printf("%d:",temp);       /*imprime   el   num.   de
                                      índice*/

  for (i=0; cp =curr_buf+1; i<NUMFR;i++,cp +=FRSIZE)
   {
```

```
    if(frame != *cp) printf("*"); /*la cuenta  de cuadros
                                    es incorrecta*/

        else
            printf(" ");            /*es correcta*/
            printf("%04x",*cp);     /*imprime el contador
                                    de cuadros*/
             frame= *cp+1;
       }
     }
    close(file_handle);
   exit(0);
   }
#pragma check_stack(off)
/*-----------------*/
reset_irq()     /*limpia la interrupción*/
{
    inp(0x3A0);
}
```

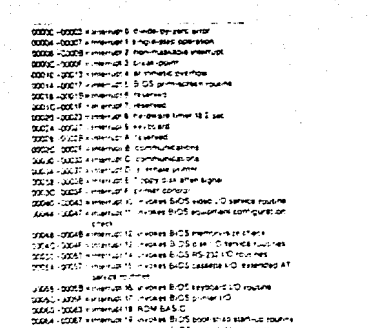


Figura 8-1A. Vectores de Interrupción.

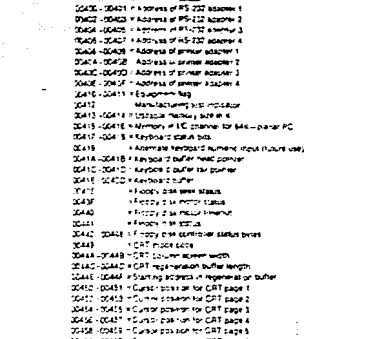


Figura 8-1B. Area de BIOS.

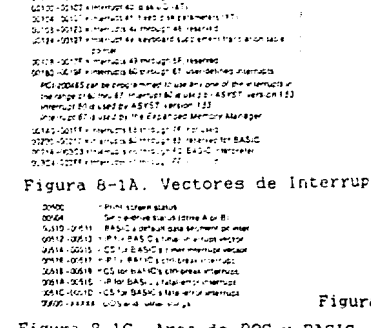


Figura 8-1C. Area de DOS y BASIC

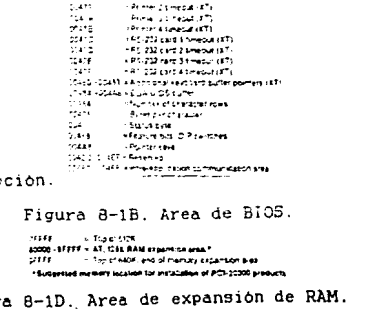


Figura 8-1D. Area de expansión de RAM.

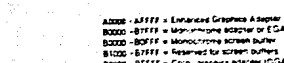


Figura 8-1E. Buffers

para manejo de video.

Figura 8-1F. Area de usuario.

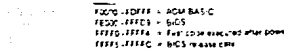


Figura 8-1G. Area de ROM.

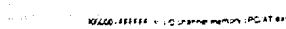


Figura 8-1H.

Area de memoria extendida.

solo para IBM PC AT.

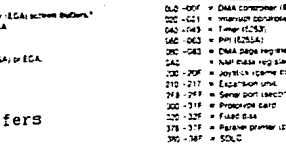


Figura 8-1I. Mapa de memoria

para IBM XT

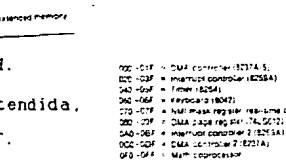


Figura 8-1H Area de memoria

PC AT.

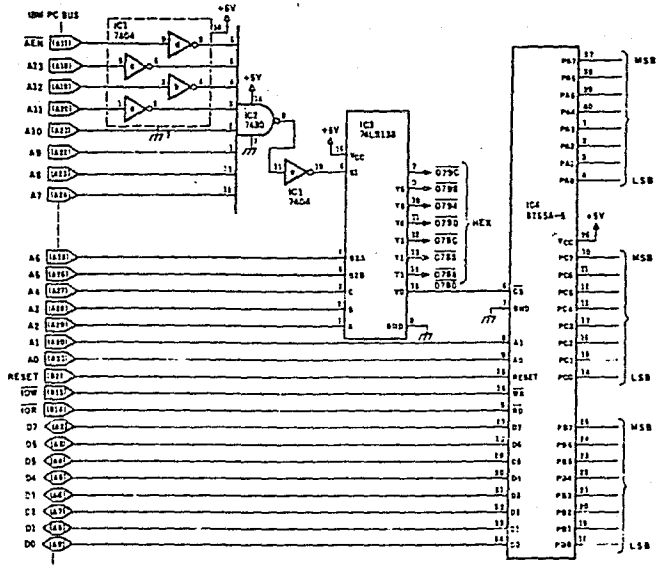


Figura 8-2. Diagrama de E/S para la tarjeta de adquisición de datos

IX- CONCLUSIONES

IX-1. EVALUACION DEL TRABAJO REALIZADO.

El objetivo de este trabajo consistió en establecer las bases para el diseño de hardware específico para microcomputadores compatibles, para usos tales como instrumentación y control, y algunas aplicaciones especiales como son el monitoreo remoto de variables atmosféricas, para la medición y control de la contaminación o fines meteorológicos.
El hardware y software descritos en este trabajo son básicos, es decir, todavía quedan muchas horas de labor para realizar una versión completa, con la posibilidad abierta de ofrecer un producto en el mercado nacional.
El hardware y software descritos en el capítulo VIII, podrían servir, a primera instancia, como prototipos didácticos en alguna materia o laboratorio donde sea necesario mostrar el proceso de diseño de interfaces a microcomputadoras, así como el desarrollo de software para el manejo de una interfaz.

En el capítulo 2, se expone lo fundamental de convertidores A/D y D/A, con el fin de tenerlos presentes en el desarrollo de aplicaciones.
El capítulo III expone lo básico para la transformada rápida de Fourier y la transformada discreta de Fourier, omitiendo las series de Fourier y la transformada de Fourier, ya que existen varios libros de referencia al respecto. Este capítulo involucra el manejo de la transformada rápida de Fourier de una manera práctica, marcando las diferencias y ventajas entre TDF y TRF.
El capítulo V está dedicado a mostrar un panorama general sobre procesadores programables de señales.
Los capítulos VI y VII se refieren al manejo de E/S en el diseño de interfaces a microcomputadoras.
El capítulo VIII es el diseño y descripción del circuito de interfaz; también expone el software para el manejo de la adquisición de datos y de la TRF.

PERSPECTIVAS DEL DISEÑO DE INTERFACES

La interfaz descrita en este trabajo podemos mejorarla tanto en hardware como en software.

HARDWARE

Se puede incluir el uso del DMA propio del microcomputador (8237) para poder transferir datos con mayor velocidad usando técnicas de programación de buffers múltiples.
También podemos incluir en al interfaz su propio procesador para el manejo de TRF sin necesidad de interrumpir al CPU central. Los procesadores a usar podrían ser :

1) Z-80 a 8 MHZ o 12 MHZ
2) 8088/8086 y 8087 (coprocesador) a 10 MHZ
3) 80188/80186 (microcontroladores de Intel) a 12 MHZ
4) 8051/8751 a 12 MHZ (o similar de Siemens)
5 8096 (microcontrolador 16 bits de Intel) a 12 MHZ

También podemos incluir en la lista la nueva generación de procesadores digitales de señales (DSP).

6) TMS320C10 (Texas Instruments), TMS320C20 y 320C30.

7) 56000 y 96000 de Motorola.
8) AMD 29000 con tecnología RISC (Reduced Instructions small computer)

Todos los procesadores presentan ventajas y desventajas, algunos son muy veloces (hasta 27 MIPS), pero de costo relativamente elevado; tal vez presentando una mala relación costo-beneficio para el propósito buscado.

SOFTWARE

Podemos desarrollar una aplicación completa para el manejo de señales tanto en el dominio del tiempo como en el dominio de la frecuencia por medio del lenguaje 'C' y ensamblador. En el caso de usar un procesador propio en la interfaz, tendríamos que desarrollar programación en el lenguaje propio del procesador, e incluirlo en un ROM, de tal manera que se encargue de controlar las señales para entenderse con el CPU del microcomputador. La ventaja al tener un procesador independiente es liberar al CPU central de las tareas iterativas y de E/S. Una de las razones para justificar un CPU independiente es el manejo en tiempo real. En la medida que aumentamos la frecuencia de operación de la señal inspeccionada complicamos el diseño y aumentamos el costo. Una opción de diseño es, en el caso de requerirse, un enfoque hacia la instrumentación y control.

El diseño de la interfaz esta basado en la decodificación del bus de direcciones de la PC compatible, mediante el espacio disponible en la memoria, y algunas de las líneas del bus de direcciones. Este diseño es el hardware básico para la adquisición de datos, mediante un convertidor A/D y un paquete de software básico para la captura y despliegue de información por medio de transformada rápida de Fourier.
Podemos encontrar varias fuentes de información para el diseño de hardware en la bibliografía de este trabajo.
Las fuentes de información para este trabajo son numerosas: ya que comprende varias áreas dentro de la ingeniería. Por lo tanto, reunir y clasificar la información puede traer como consecuencia errores u omisiones. El hardware diseñado es una versión simplificada, la cual nos podría permitir efectuar análisis espectral de señales con frecuencias hasta de 20 KHz (revisar capítulo II), con una microcomputadora compatible con procesador 8088-1 a 10 MHz, pero para frecuencias mayores requerimos un computadora AT 80286 a 12 MHz ó 16 MHz, y posiblemente un cambio en el diseño que incluya manejo de DMA (revisar capítulo VII y VIII).
La frecuencia máxima de la señal bajo análisis depende del tipo de convertidor A/D, del tipo de procesador usado por la microcomputadora (8088,80286, 80386 y 80486), la complejidad de hardware en la interfaz, y por último, del software para el manejo de la interfaz.

AUTOEVALUACION

El trabajo realizado presenta carencias tales como :

1) La terminacion de un paquete de software completo programado en Turbo C (borland) o Microsoft C Compiler.

2) El diseño de hardware más elaborado :

a) La utilización de un procesador propio de la interfaz.

b) El manejo de acceso directo a memoria (DMA). Ya sea uno de los canales libres de la microcomputadora o la utilización de un integrado DMA propio.

Por otro lado, es posible usar un convertidor con mayor velocidad de conversión, pero esto implica mayor inversión. Los programas del sistema son programas de prueba y no presentan la ventajas que ofrecería un versión detallada con mayor manejo de gráficas.

El sistema diseñado tiene un enfoque didáctico para su utilización en alguna materia relacionada con procesamiento digital de señales, lo que se verá incluido en materias de los planes de estudios futuros. Lo que permitiría utilizar sistemas parecidos para enseñanza.
El programa o subrutina para el calculo de la transformada rápida de Fourier no es un programa optimizado, lo que significa que tendríamos que reprogramar la rutina para dicho fin.

Desgraciadamente no incluí suficiente información, en detalle, sobre algunos DSP's que hay en el mercado de tal manera que se hubiera podido hacer una comparación entre algunos de los más reconocidos.

El trabajo contiene mucha información, algunas veces en detalle, pero ofrece desde mi punto de vista, el material necesario para fomentar la busqueda en este campo del análisis espectral dentro del procesamiento digital de señales.

1-Ramirez W. Robert. The FFT fundamentals and concepts. Prentice Hall 1985.

2-Yuen C.K. Microprocessors Systems and their applications to signal processing. Academic Press. 1982

3-Uffenbeck John. Microcomputers and Microprocessors Prentice Hall 1985.

4-Oppenheim Alan V. Schaffer W. Ronald. Discrete Time signal processing. Prentice Hall 1989.

5-Stanley D. William. Digital Signal Processing. Reston Publishing Company Inc. 1975.

6-Singh Avtar/Triebel A. Walter. The 8088 microprocessor, programming, interfacing, software, hardware, and applications. Prentice Hall. 1989.

7-Liu Yu-Cheng, Gibson A. Glenn Microcomputer Systems. The 8086/8088 Family. Prentice Hall 1986.

8-Kassab Vincent. Technical C Programming. Prentice Hall. 1989.

9-Artwick V. Douglas Microprocessors and digital Systems. Mc Graw Hill 1985.

10-Schildt Herbert. C made easy. Mc-Graw Hill 1985.

11-Ezzel Ben Graphics Programming in Turbo C 2.0. Addison Wesley. 1989.

12-Stevens T. Roger. Graphics Programming in C. M&T Books 1989.

13-Barkakati. The Waite Groups. Quick C Bible. Howard W. Sams & Company. 1989.

14-Stan Kelly- Bootle. Mastering Quick C. Sybex 1989

Libros Especializados

15-Lim S. JAE. Oppenheim V. Allan. Advanced Topics in signal processing. Prentice Hall 1988.

16-Stearns D. Samuels. David A. Ruth. Signal Processing Algorithms. Prentice Hall 1988.

17-Kay M. Steven. Modern Espectral Estimation: Theory & Applications. Prentice Hall 1988.

18-Brigham Oran E. Fast Fourier Transform. Prentice Hall. 1974