UNIVERSIDAD LA SALLE

Escuela de Ingeniería
Incorporada a la U.N.A.M.

"DISEÑO E IMPLEMENTACION DE UN FILTRO DIGITAL"

Tesis Profesional

Que para obtener el título de:

INGENIERO MECANICO ELECTRICISTA

P r e s e n t a :

Mónica María Corlay Trujillo

Director de Tesis: Ing. Guillermo Aranda Pérez

México, D. F., Abril de 1991.

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

INDICE

A mis padres

Lic. Lino Corlay Chiao y Carmen L. Trujillo de Corlay por todo su amor, comprensión, ejemplo y apoyo en todos los momentos de mi vida. Gracias por transmitirme sus experiencias y por darme la oportunidad de estudiar y superarme como persona en todos los aspectos. A ustedes les debo todo lo que soy y lo que tengo.

Con cariño para mi hermano Lino

Gracias por tu apoyo y las alegrías compartidas.

A mis abuelos: Manuel Corlay Ley (q.e.p.d)

Pan Kuey Chiao de Corlay (q.e.p.d)

Javier Trujillo Solís (q.e.p.d)

Margarita Alvarez García

Dedico esta tesis a todas las personas que siempre me han apoyado y que de alguna u otra forma siempre han estado conmigo, en especial a:

A mi tío Jesús Corlay Martínez

A mi tía Lic. Flor de Ma. Esponda Argüello

Familia Kato Aguilar

Familia Guzmán Godoy

Familia Enríquez Habib

Con cariño para:

Mi madrina Cesarea Chamlati T.

Mis primos Roberto y Aída, Lucy, Eduardo, Rubén, Martín, Fanny y Gladys.

Mis tíos Eduardo, Ma. Elena, Bety, Sayra, Jaime, Tere, Guillermo, Daniel, Ismael, Sode y Cory.

A Reyna Corlay.

A Gloria Romero González

Gracias por tu amistad y ejemplo de tenacidad.

A mis maestros de la Universidad la Salle en especial:

Ing. José Antonio Torres Hernández

M. en C. Ricardo Valenzuela G.

Ing. Mario Ibarra Pereyra.

Lic. Jose Sanfuentes Theurel.

A las personas que laboran en Talleres de Ingeniería en especial a Moisés Miranda y Emilio.

A mis amigos y compañeros de generación, en especial a uno de ellos.

A las Cedimitas: gracias por todos los gratos momentos que pasamos juntas.

Mis más sinceros agradecimientos a:

M. en C. Antonio Ramírez Treviño, por toda la ayuda que me facilitó para la realización de este trabajo y por su amistad.

M. en C. Guillermo Aranda Pérez, por su ayuda en la elaboración y correción de esta tesis.

INTRODUCCION

INTRODUCCION: EL MUNDO Y LA INFORMACION.

El ser humano siempre ha tenido la necesidad de comunicarse con sus semejantes. El hecho de establecer una comunicación con otro ser implica enviar y recibir un mensaje. Los mensajes son la representación física de la información.

La tecnología ha desarrollado medios eficientes para transmitir la información. Estos avances incluyen el envío y recepción de mensajes desde cualquier parte del mundo. Los medios de comunicación actuales como son el radio, la televisión y la telefonía trabajan en base a señales eléctricas.

PROCESAMIENTO DE SEÑALES.

Durante la transmisión de la información se pueden introducir señales ajenas a ella. A este tipo de señales se les conoce como ruido.

Como la señal de información puede contener ruido, se desarrollaron sistemas para procesar las señales. Se le llama procesamiento de señales al proceso de extraer la información útil de una señal y suprimir el resto.

Existen dos formas de procesar las señales: el procesamiento analógico y el procesamiento digital de señales.

PROCESAMIENTO ANALOGICO DE SEÑALES.

El procesamiento analógico de señales procesa señales analógicas. Un mensaje es una señal analógica, ya que puede tomar un infinito número de formas de onda. Un ejemplo es la información transmitida por la televisión y el radio.

PROCESAMIENTO DIGITAL DE SEÑALES.

En el procesamiento digital de señales se utilizan señales digitales. En los sistemas digitales el proceso más importantes es el proceso de muestreo, ya que al muestrear una señal analógica se puede obtener una señal digital. De esta forma se pueden procesar señales analógicas mediante procesos digitales.

IMPORTANCIA DEL PROCESAMIENTO DIGITAL DE SEÑALES

En la actualidad el procesamiento de señales se lleva a cabo por medio de sistemas digitales. El interés en el uso de sistemas digitales ha aumentado por las siguientes razones:

1) El uso de canales de comunicación con mayor ancho de banda.
2) Facilidad de acceso a una computadora.
3) Flexibilidad en su implementación.
4) El incremento de la necesidad de transmitir datos digitales.
5) Desarrollo de circuitos digitales con un alto grado de integración y

6) El costo decreciente de los circuitos digitales. Un ejemplo de este tipo de circuitos es el microprocesador.

El microprocesador es uno de los adelantos tecnologicos más importante de los últimos años. Una de sus aplicaciones más importantes es la automatización de procesos, y permite una gran flexibilidad en el diseño de sistemas digitales.

Se han encontrado numerosas aplicaciones importantes al procesamiento digital de señales. Algunas de ellas se encuentran en el campo de las comunicaciones, como son los sistemas de radar y de sonar. Recientemente se ha empezado a aplicar a campos como el audio y la acústica, puesto que son campos muy comerciales se están desarrollado una gran cantidad de sistemas digitales mas compactos y eficientes.

OBJETIVOS DE LA TESIS.

- El objetivo principal de este trabajo es el de analizar el funcionamiento de los filtros digitales a fin de establecer sus caracteristicas y la posibilidad de utilizarlo en un sistema digital.

- Diseñar un filtro digital.

- Desarrollar un sistema fisico para implementar el filtro digital diseñado.

ORGANIZACION DE LA TESIS.

En el capítulo uno, se establece el problema del filtrado de señales desde un punto de vista matemático. Se introducen los conceptos de señal y filtro, así como sus clasificaciones respectivas. Asimismo, se hace un breve análisis de los filtros analógicos y el tipo de respuesta de los filtros Butterworth. Posteriormente se da una introducción a lo que es el filtrado de señales digitales, el proceso de digitalización de una señal y el modelado de sistemas discretos mediante la transformada Z. Para finalizar el capítulo, se realiza una comparación entre los filtros analógicos y digitales.

El capítulo dos presenta un estudio de los filtros digitales de respuesta al impulso infinita (IIR: Infinite Impulse Response) y de respuesta al impulso finita (FIR: Finite Impulse Response). Se analizan sus características y las formas de programación existentes. También se realiza una comparación entre ambos tipos de filtros. El capítulo finaliza con la explicación de algunos métodos de discretización para filtros IIR.

La configuración del sistema mínimo se establece en el capítulo tres. Contiene una breve descripción de los componentes utilizados en el sistema, como son el microprocesador y los convertidores analógico-digitales, así como su funcionamiento. También se muestran los diagramas eléctricos parciales del sistema realizado.

Las herramientas utilizadas para el diseño del filtro y su implementación se explican en el capítulo cuatro. Se da una breve explicación de todos los paquetes computacionales utilizados y de los pasos a seguir para la programación del filtro. También se diseña el filtro que se va a implementar en el sistema.

En el último capítulo se detallan las subrutinas y algoritmos utilizados para la programación del filtro. El uso de algoritmos como los de la búsqueda binaria y el de Booth's, permiten realizar las operaciones en el menor tiempo posible. Además, el uso del formato Q8 facilita el manejo de números con punto decimal. Al final del capítulo se presentan los resultados del desempeño del filtro obtenidos experimentalmente.

Finalmente, en el apéndice A se encuentra el listado del programa del filtro realizado.

CAPITULO 1

FILTRADO DE SEÑALES

1 FILTRADO DE SEÑALES

Primero daremos una definición de señal y su clasificación. Posteriormente se introducirá el concepto de filtro de señales y su clasificación.

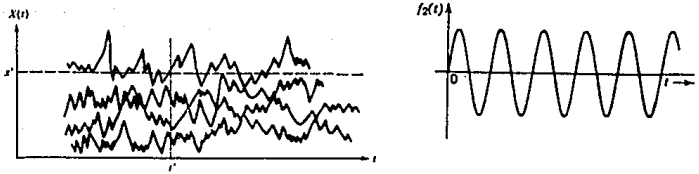
1.1 DEFINICION DE SEÑAL

Para nuestro caso, una señal es una representación eléctrica de una cantidad física en función del tiempo.

1.2 CLASIFICACION DE LAS SEÑALES

Las señales se clasifican de acuerdo a ciertas características; si las señales pueden escribirse o no mediante una ecuación se clasifican en **señales aleatorias** o **señales deterministicas**. Una **señal aleatoria** es la que presenta algún grado de incertidumbre en sus valores. Los valores futuros de la señal no son predecibles, aún conociendo algunos valores anteriores. Una **señal determinística** es aquella señal cuyos valores futuros son predecibles. Se puede escribir una expresión matemática que describe su comportamiento (fig 1.1).

Una señal determinística que se repite a intervalos regulares se clasifica como **señal periódica** o en caso de no repetirse a intervalos regulares en **señal aperiódica**. Una **señal** periódica es una función que tiene el mismo valor cada vez que su



a) Señal aleatoria. b) Señal determinística.

Figura 1.1

variable t aumenta en una cantidad fija llamada periodo, o en un múltiplo de éste. El periodo T es el tiempo en que tarda una señal en cumplir un ciclo.

$$f(t) = f(t + nT) \qquad .....(1.1)$$

donde : T = número real y positivo

n = número entero

y una **señal aperiódica** no cumple con las características de las señales periódicas.

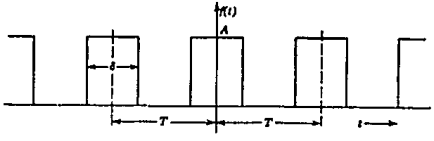
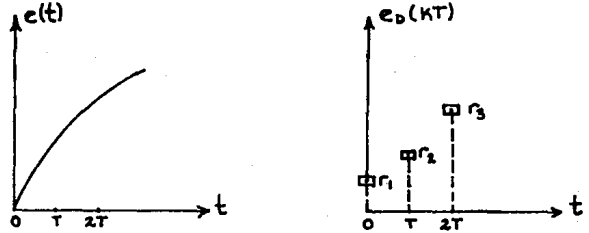


Figura 1.2 Señal periódica.

Una última clasificación se encuentra por el dominio de la imagen de la señal. Una **señal analógica** es una función continua en el tiempo cuyos valores se encuentran definidos dentro del conjunto de números reales. Las **señales discretas** son funciones cuyos valores se encuentran definidos en un conjunto numerable. También recibe el nombre de señal digital.



a) Señal analógica. b) Señal digital.

Figura 1.3

1.3 CARACTERISTICAS DE LAS SEÑALES

Las características principales de todas las señales son **frecuencia, amplitud y fase.**

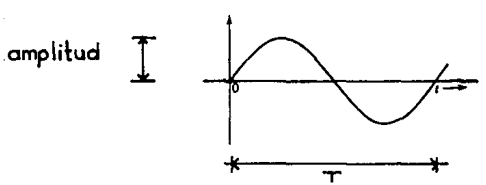


Figura 1.4

La **frecuencia** se define como el número de periodos por segundo de una señal periódica. La siguiente ecuacion describe la relación que guardan el periodo (T) y la frecuencia (f):

$$f = \frac{1}{T} \; ; \; [Hz] \qquad \ldots\ldots 1.2)$$

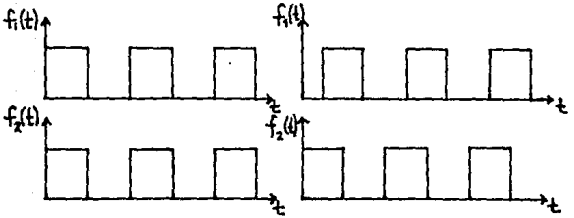
También existe la frecuencia angular w, que se mide en radianes/segundo y cuya magnitud está dada por :

$$w = 2\pi f \; ; \; [s^{-1}] \qquad \ldots\ldots (1.3)$$

El intervalo de frecuencias audibles se extiende desde 20 Hz hasta 20 kHz. El sonido se compone de señales cuyas frecuencias se encuentran dentro de este intervalo. A los sonidos con frecuencias menores a 20 Hz se les denomina infrasonidos, mientras que a los que presentan una frecuencia mayor a 20 kHz se les denomina ultrasonidos.

La **amplitud de una señal** es su máximo valor alcanzado durante un periodo.

La **fase** es el valor angular de una señal en un instante dado. Dos señales de igual frecuencia se hallan en fase cuando adquieren sus amplitudes máxima y mínima al mismo tiempo. Se hallan defasadas cuando sus valores máximos y mínimos no corresponden simultáneamente.



a) Señales en fase. b) Señales defasadas.

Figura 1.5

1.4 PROCESAMIENTO DE SEÑALES

El procesamiento de señales elimina algunas de las características no deseadas de la señal. Para procesar el sonido se utilizan los filtros.

1.5 FILTROS

Un filtro es un sistema que elimina, de una señal, ciertas frecuencias que se encuentren dentro los límites pre-establecidos.

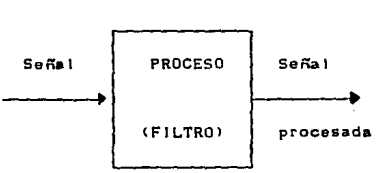
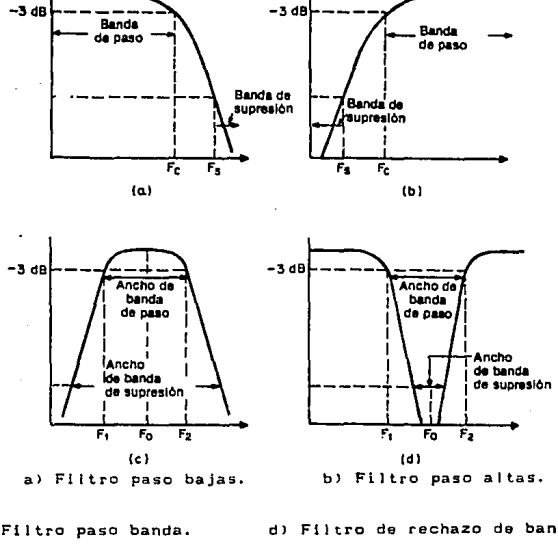


Figura 1.6

Los filtros se clasifican en cuatro categorías, de acuerdo con la función que desempeñan:

- Filtros paso bajas.
- Filtros paso altas.
- Filtros paso banda.
- Filtros de rechazo de banda.



a) Filtro paso bajas. b) Filtro paso altas.

c) Filtro paso banda. d) Filtro de rechazo de banda.

Figura 1.7

Los filtros paso bajas son los que permiten el paso a los componentes de frecuencia cero hasta una frecuencia específica.

A esta frecuencia se le conoce como frecuencia de corte (fc), y es donde la señal disminuye su ganancia en 3 dB.

Por el contrario, los filtros paso altas son los que atenuan las frecuencias menores a la frecuencia de corte fc, permitiendo el paso de frecuencias mayores a fc.

Un filtro paso banda es aquel que deja pasar las frecuencias que están dentro de una banda especifica. Tiene una frecuencia central fc, una frecuencia de corte inferior (fi) y una frecuencia de corte superior (fs).

El filtro de rechazo de banda suprime las frecuencias comprendidas dentro de los límites, fi y fs, dejando pasar los componentes que no están en el rango establecido.

Existe a su vez una clasificación de los filtros de acuerdo con el tipo que señal que procesan: los filtros analógicos y los digitales, procesando señales analógicas y digitales respectivamente.

1.6 FILTRADO DE SEÑALES ANALOGICAS

Una señal analógica es una función continua en el tiempo cuyos valores se encuentran definidos dentro del conjunto de numeros reales [§ 1.2] Un filtro analógico procesa señales analogicas.

Figura 1.8

Definición de función de transferencia H(s): Es la relacion matemática entre la salida y entrada en el dominio de 's'[1] de un sistema lineal e invariante en el tiempo, cuyas condiciones iniciales son nulas. Se representa con la siguiente ecuacion :

$$H(s) = \frac{Y(s)}{X(s)} \qquad .....(1.4)$$

donde X(s) y Y(s) son las transformadas de Laplace de las señales de entrada y la salida respectivamente. Las raices del polinomio Y(s) reciben el nombre de ceros, y las raices de X(s) son los polos.

Como 's' es una variable compleja ($s=\sigma+jw$), H(S) es una cantidad compleja. H(S) tiene una parte real y una parte imaginaria. Haciendo $\sigma = 0$ tenemos

---

[1] 's'es la variable de la transformada de Laplace.

$$H(jw) = Re[H(jw)] + Im[H(jw)] \qquad .....(1.5)$$

En notacion polar:

$$H(jw) = |H(jw)|e^{\angle H(jw)} \qquad .....(1.5)$$

donde $|H(jw)|$ y $\angle H(jw)$ denotan la magnitud y la fase de H(jw) respectivamente.

Para mayor comprensión de estos términos, estudiemos un filtro paso bajas con la siguiente configuración:

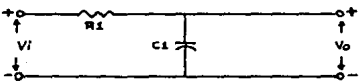


Figura 1.9

La tensión de salida Vo es :

$$Vo = \frac{1/sC_i}{R_i + 1/sC_i} Vi = \frac{1}{1+sR_iC_i} Vi \qquad ...(1.7)$$

Obteniendo la función de transferencia H(s)=Vo/Vi:

$$H(s) = \frac{1}{1+sR_1C_1} \qquad .....(1.8)$$

Llamaremos A_L a la función de transferencia H(s). Para obtener A_L en términos de la frecuencia f se hacen las siguientes transformaciones:

$$s=\sigma+jw \qquad .....(1.9)$$

De la ecuación (1.2)

$$w=2\pi f$$

y sustituyendo en (1.9)

$$s=\sigma+j2\pi f$$

Despejando f

$$f=(s-\sigma)/2\pi j \qquad .....(1.10)$$

Para todo análisis en frecuencia, $\sigma=0$, entonces $f=s/2\pi j$. Sustituyendo en (1.8) se obtiene:

$$A_L = \frac{1}{1-j(f/fc)} \qquad .....(1.11)$$

donde la frecuencia de corte fc equivale a:

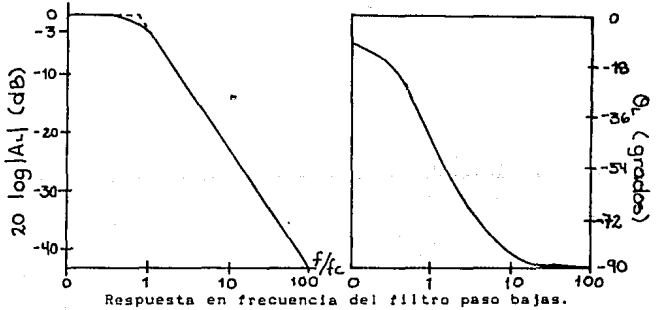
$$fc = \frac{1}{2\pi R1C1} \; ; \; [Hz] \qquad .....(1.12)$$

La magnitud de la ganancia será:

$$|A_L| = \frac{1}{\sqrt{1+(f/fc)^{2n}}} \qquad .....(1.13)$$

y su fase : $\theta_L = - \arctan f/fc$ .....(1.14)

En la siguiente gráfica se observa que cuando f=fc, A_L=0.707 (-3 dB) [2]; y cuando f<fc H(f) tiende al valor de 1 (0 dB).



Respuesta en frecuencia del filtro paso bajas.

Figura 1.10

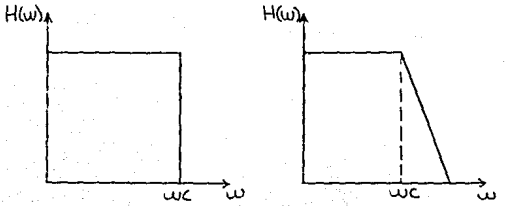
[2] La fórmula para convertir una cantidad en decibeles es: (dB)=20 log A_L.

A estos dos tipos de gráficas se les conoce como diagramas de Bode. En ellas se representan la magnitud y ángulo de fase en función de la frecuencia. Estos diagramas nos sirven para determinar la respuesta a la frecuencia de los sistemas.

Definición. La respuesta a la frecuencia de un sistema se toma en estado estacionario, con una señal de entrada periódica (generalmente senoidal). Esto para todos los valores de frecuencia.

1.8.1 EL FILTRO IDEAL

Comparando la respuesta del filtro paso bajas obtenida anteriormente, con la de un filtro paso bajas ideal, se observa que la respuesta real difiere mucho de la ideal.

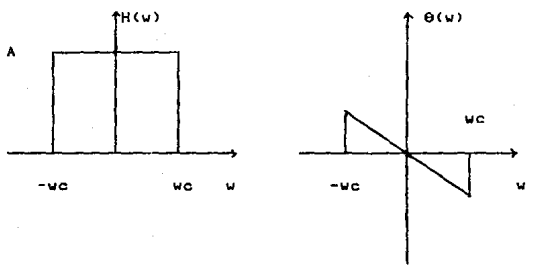


a) Filtro paso bajas ideal. b) Filtro paso bajas real.

Figura 1.11

Un filtro ideal es fisicamente irrealizable debido a que es un sistema no causal. El sistema produce una respuesta antes de que le sea aplicada una señal de entrada.

En las figuras 1.12a y 1.12b se muestra el espectro de un filtro ideal y su fase. Fuera del rango (-wc,wc) no existe ninguna señal. El ángulo de fase del filtro es $-wt_o$.



a) Espectro del filtro ideal. b) Fase del filtro ideal.

Figura 1.12

Se analizará la respuesta del filtro ideal cuando su entrada es un impulso ideal (fig. 1.12c).

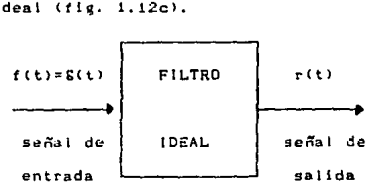


Fig. 1.12c

H(w) es la transformada de Fourier de la función del filtro ideal, y de la gráfica 1.12a se obtiene que:

$$H(w) = A\,G\left[\frac{2wc}{w}\right] e^{-j w t_0} \qquad .....(1.15)$$

Sabemos que H(w) = R(w)/F(w), donde F(w) y R(w) son las transformadas de Fourier de las señales de entrada y salida del filtro respectivamente.

Como F(w)=1, R(w)=H(w)

$$R(w) = A\,G\left[\frac{2wc}{w}\right] e^{-j w t_0} \qquad .....(1.16)$$

Para obtener la respuesta en tiempo, r(t) se usa la transformada inversa de Fourier.

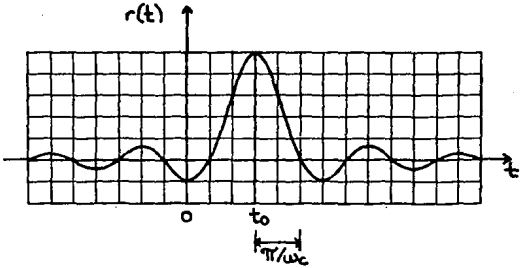
$$r(t) = \mathcal{F}^{-1}\, R(w) \qquad .....(1.17)$$

Entonces:

$$r(t) = \mathcal{F}^{-1}\, A\, G\left[\frac{2w_c}{w}\right] e^{-j\omega t_0} = sa\,(t-t_0) \quad \ldots\ 1.18$$

La gráfica de r(t) se observa en la fig. 1.13. La señal de entrada fué en t=0, y la señal de salida ocurre antes de t=0. Esto quiere decir que el filtro ideal genera una respuesta antes de que se aplique la excitación.

Por esta razón se trabaja con aproximaciones. Una aproximación es una función de transferencia realizable, de tal manera que su gráfica de magnitud se asemeje a las características del filtro ideal.



Respuesta al impulso de un filtro ideal.

Figura 1.13

Las aproximaciones más conocidas son los filtros Elipticos, Chebyshev, Butterworth y Bessel. Se seleccionó trabajar con los filtros Butterworth porque su utilización es sencilla.

1.6.2 FILTROS BUTTERWORTH.

Una aproximación de la función de transferencia de un filtro paso bajas ideal es de la siguiente forma:

$$H(s) = \frac{1}{Pn(s)} \qquad .....(1.19)$$

siendo Pn(s) un polinomio en s.

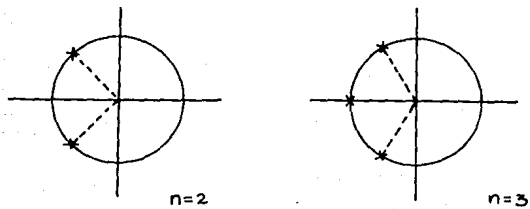
Los poliomios de Butterworth Bn(s) se sustituyen en Pn(s). Sustituyendo s=σ+jw, obtenemos H(w), siendo su magnitud:

$$|H(jw)| = \frac{1}{\sqrt{1+(w/wc)^{2n}}} \qquad .....(1.20)$$

donde n es el número de polos de H(w), y wc la frecuencia de corte del filtro. El número de polos indica el orden del filtro.

Una característica de este tipo de filtro es que sus polos se encuentran equidistantes angularmente alrededor de un círculo con centro en el origen del plano 's'. El espacio angular entre

los polos es π/n radianes. Cuando n es par, no se presentarán polos en el eje real, mientras que si n es impar, un polo se localizará en dicho eje. Nunca se localizarán polos en el eje imaginario [1.11].



Localización de los polos de un filtro Butterworth para n=2 y n=3

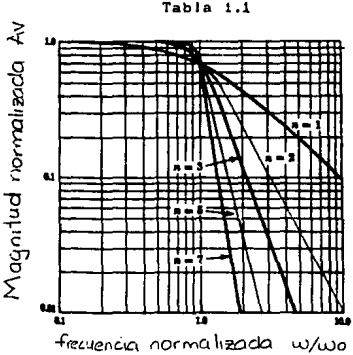
Figura 1.14

Para facilitar el análisis de los filtros paso bajas se normaliza la frecuencia de corte. En este caso, todos los filtros tienen como frecuencia de corte 1 rad/s, en el cual la señal presenta una atenuación de -3 dB de magnitud. La tabla 1.1 presenta los factores de los polinomios de Butterworth normalizados hasta n=7 para filtros paso bajas.

La gráfica 1.15 nos presenta la respuesta a la frecuencia de los filtros paso bajas para varios valores de n. Al aumentar el orden n del filtro la respuesta se aproximará a la de un filtro ideal.

Orden n	Factores del Polinomio Bn(s)
1	$(s + 1)$
2	$(s^2 + 1.414s + 1)$
3	$(s + 1)(s^2 + s + 1)$
4	$(s^2 + 0.765s + 1)(s^2 + 1.848s + 1)$
5	$(s + 1)(s^2 + 0.0618 + 1)(s^2 + 1.618s + 1)$
6	$(s^2 + 0.518s + 1)(s^2 + 1.414s + 1)(s^2 + 1.932s + 1)$
7	$(s + 1)(s^2 + 0.445s + 1)(s^2 + 1.247s + 1)(s^2 + 1.802s + 1)$

Tabla 1.1



Respuesta en frecuencia de los filtros Butterworth paso bajas normalizados

Figura 1.15

1.6.3 PROPIEDADES BASICAS DE LOS FILTROS BUTTERWORTH PASO BAJAS NORMALIZADOS

Las principales propiedades de los filtros Butterworth se mencionan a continuación [6]:

1. Para cualquier orden se tiene que :

$$|H(j0)|^2=1$$
$$|H(j1)|^2=0.5$$
$$|H(j\infty)|^2=0$$

Esto implica que la ganancia en dc (el valor de la magnitud en w=0) es 1 y que en la frecuencia de corte (1 rad/s) es -3 dB.

2. La ganancia de los filtros Butterworth decrece monotónicamente conforme $w\to\infty$. Entonces $|H(jw)|$ tiene su valor máximo en w=0.

3. Las primeras 2n-1 derivadas de un filtro de n-ésimo orden son cero en w=0. Por esta razón también se les conoce como filtros de máxima respuesta plana en la región de paso banda.

4. La pendiente de un filtro de orden n es de 20n dB/década.

La forma típica de una función de transferencia para un filtro paso bajas de orden 1 es:

$$H(s) = \frac{1}{s/wc + 1} \qquad .....(1.21)$$

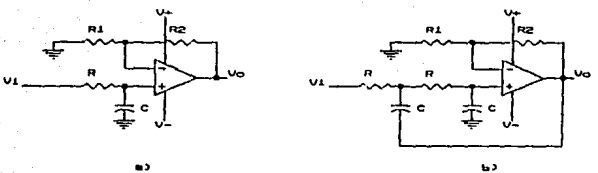
donde la frecuencia de corte wc equivale a $2\pi f$.

Reacomodando términos :

$$H(s) = \frac{wc}{s + wc} \qquad .....(1.22)$$

La configuración de un filtro de primer orden se observa en la fig. 1.16a. La combinación del valor del capacitor con el valor de la resistencia nos da la frecuencia de corte wc(=1/RC).

La fig. 1.16b presenta la configuración para un filtro de segundo orden. Para construir un filtro de orden impar, se utilizan elementos de segundo y primer orden en cascada, mientras que para los de orden par se usan los de segundo orden exclusivamente.



a) FILTRO PASO BAJAS DE PRIMER ORDEN.
b) FILTRO PASO BAJAS DE SEGUNDO ORDEN.

Figura 1.16

1.7 FILTRADO DE SEÑALES DIGITALES.

En el inciso anterior, se describió el funcionamiento de los filtros analógicos. Los filtros digitales realizan la misma función que los analógicos, solo que de una manera diferente.

Un filtro digital es un sistema procesador de señales digitales que utiliza la precisión aritmética finita. También se describen como una ecuación en diferencias lineal y de coeficientes constantes. Pueden implementarse como "software", en subrutina en una computadora digital: como "hardware" en un circuito que contenga registros, multiplicadores y sumadores.

Generalmente, las señales discretas se obtienen a partir del proceso de digitalización de una señal analógica, realizando una cuantificación de sus valores. La forma de digitalizar una señal

analógica es usando muestreadores y retenedores. La función de un muestreador es convertir una señal continua en un tren de pulsos modulados en amplitud. El retenedor mantiene el valor del pulso entregado por el muestreador.En diagramas de bloques, el muestreador se representa con un interruptor, y el retenedor por un bloque. La frecuencia de muestreo debe cumplir con el teorema de Shannon [6,10].

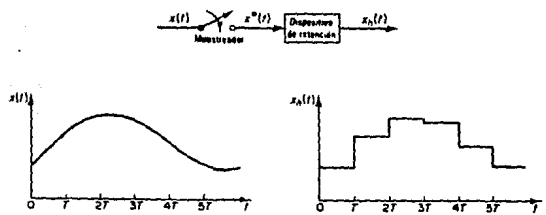


Figura 1.17

1.7.1 DIGITALIZACION DE UNA SEÑAL ANALOGICA

En general, el proceso de muestreo consiste en tomar valores instantáneos de una señal continua f(t) a intervalos regulares de duracion T . [3]

---

[3] Solamente trataremos el caso del muestreo uniforme (a intervalos de tiempo constantes)

Teorema de muestreo de Shannon : Sea una función f(t) que no contiene componentes de frecuencia mayores a wc rad/s (señal de banda limitada). Entonces la información contenida en f(t) puede ser reconstruida con muestras de f(t) que estén espaciadas entre sí por

$$T \leq \pi/wc \qquad .....(1.23)$$

es decir, que se debe muestrear a una frecuencia de por lo menos el doble de la máxima frecuencia de la señal a digitalizar [10].

Este teorema nos da el valor máximo de T permisible para poder reconstruir la señal original y evitar la superposición de espectros, como veremos posteriormente.

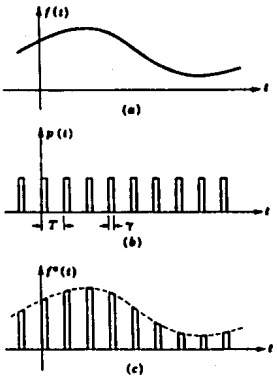
El proceso de muestreo de una señal se puede considerar un proceso de modulación en el cual un tren de pulsos unitarios p(t) multiplica a una función continua en el tiempo f(t) y se produce una función muestreada f*(t). Esto se representa por:

$$f^*(t) = p(t)f(t) \qquad .....(1.24)$$

Si se expande la función p(t) por series de Fourier:

$$p(t) = \sum_{n=-\infty}^{+\infty} C_n e^{jn\omega_s t} \qquad .....(1.25)$$

donde ws es la frecuencia de muestreo, y ws=2π/T.



a) Función continua f(t), b) Tren de pulsos de muestreo p(t).

c) Función muestreada f*(t).

Figura 1.18

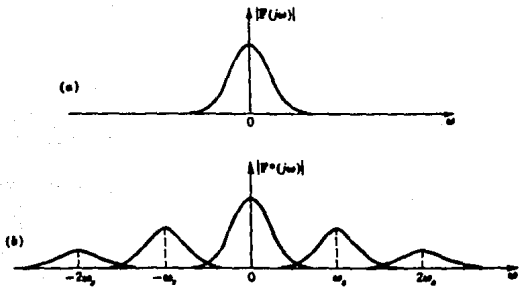
La función muestreada queda:

$$f^*(t) = \sum_{n=-\infty}^{+\infty} C_n f(t) e^{jn\omega_s t} \qquad .....(1.26)$$

Sea F(jw) la transformada de Fourier de la función continua f(t). Entonces, la transformada de Fourier de la función f*(t) es:

$$F^*(jw) = \sum_{n=-\infty}^{+\infty} C_n F(jw+jnws) \qquad .....(1.27)$$

Una comparación del espectro de Fourier de la señal continua y la muestreada se muestra en la figura 1.18. Se observa que el proceso de muestreo produce un espectro fundamental similar en forma a la de la función continua.



Espectro a) de una señal continua f(t), b) una función muestreada por pulsos f*(t).

Figura 1.18

Tambien se produce una sucesión de espectros complementarios que están separados por una frecuencia nws.

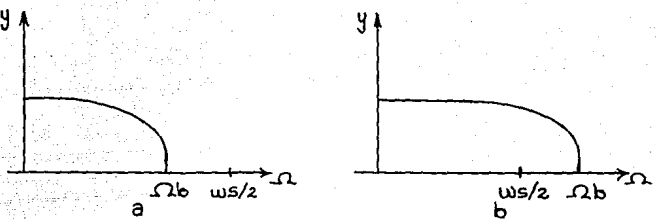
Si la frecuencia de muestreo es suficientemente alta, solo existirá un pequeño traslape entre el espectro fundamental y el complementario. En caso contrario, se presentará el fenómeno de la superposición de espectros.

1.7.2 SUPERPOSICION DE ESPECTROS DE FRECUENCIA

La superposición de los espectros de frecuencia ("aliasing") es un fenómeno relacionado con la frecuencia de muestreo de señales periódicas. Este fenómeno no permite diferenciar entre dos señales periódicas, cuyas frecuencias difieren en múltiplos enteros de la frecuencia de muestreo. El espectro de la señal de mayor frecuencia se repetirá en el mismo lugar que el espectro de la señal de menor frecuencia, lo que impide diferenciar entre un espectro de una señal y el otro. Se explicará este fenómeno con un ejemplo.

Las figuras 1.20a. y 1.20b muestran un espectro de frecuencia típica de una señal continua limitada por la frecuencia Ω_s. La frecuencia de muestreo de ambas señales es ws/2. En el primer caso, Ω_s tiene un valor mayor que ws/2, mientras que en el segundo caso, Ω_s es menor que ws/2.

Sabemos que el espectro de una señal se repite a intervalos de ws/2. En el primer caso, $ws/2>\Omega_s$, la frecuencia de muestreo es al menos el doble de la frecuencia más alta de la señal (Ω_s).



a)Frecuencia de muestreo $ws/2>\Omega_b$.

b)Frecuencia de muestreo $ws/2<\Omega_b$.

Figura 1.20

Entonces el espectro se repetirá periódicamente sin que exista algún traslape entre los espectros.

En cambio, para el segundo caso $\Omega_b>ws/2$, y los espectros se traslapan unos con otros. Se demostrará que en este caso será imposible diferenciar entre dos señales periódicas cuya suma o diferencia de frecuencias es un múltipio entero de ws ($\Omega 1, \Omega 1+ws, \Omega 1+2ws...$).

Sea la función $y(t) = sen(\Omega t)$ ....(1.28) en donde $0<\Omega<\Omega_b$

y $\Omega = \Omega_1 + wsk$ ....(1.29) donde $k = 0,1,2...$

Sustituyendo (1.29) en (1.28) :

$$y(t) = \text{sen } (\Omega_i + wsk)t \qquad .....(1.30)$$

La ecuación (1.28) para cada instante i (i=0,1,2,...) es :

$$yi = \text{sen } (\Omega iT) \qquad .....(1.31)$$

Sustituyendo (1.29) en (1.30)

$$yi = \text{sen } (\Omega_i + wsk)iT \qquad .....(1.32)$$

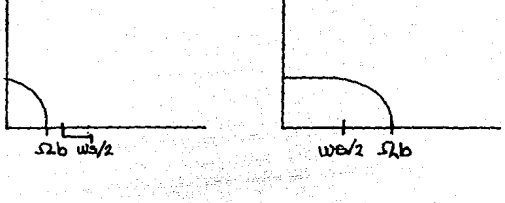
Como ws = 2π/T, entonces

$$yi = \text{sen } (\Omega_i iT + 2\pi ik) \quad (ik=0,1,2,....) \quad ..(1.33)$$

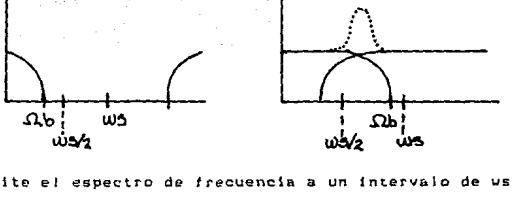
Pero

$$\text{sen } (\Omega_i iT + 2\pi ik) = \text{sen } (\Omega_i iT) \qquad .....(1.34)$$

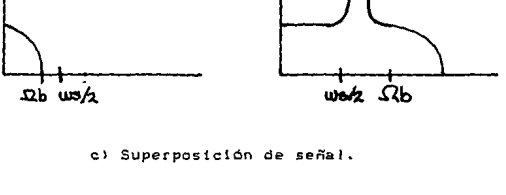
Se observa que las ecuaciones 1.34 y 1.31 son idénticas. Se concluye que todas aquellas señales cuya frecuencia es mayor a ws/2 se ven como señales de una frecuencia entre 0 y ws/2, o sea, las señales de frecuencias mayores son repetidas en el intervalo $0 < \Omega < ws/2$ [7].



a) Espectro de la señal y frecuencia de muestreo.



b) Se repite el espectro de frecuencia a un intervalo de ws/2.



c) Superposición de señal.

Figura 1.21

Generalmente, los sistemas analógicos se analizan en el dominio complejo de Laplace, siempre y cuando sean lineales. Para los sistemas discretos se utiliza la transformada Z, que presenta ventajas como la reducción de complejidad en el análisis al utilizar ecuaciones en diferencias.

1.7.3 TRANSFORMADA Z

La transformada z sirve para modelar, analizar y sintetizar sistemas lineales e invariantes en el tiempo en el dominio discreto.

Antes de la definición formal de la transformada z, se describe el concepto de muestreo ideal.

Tenemos una función continua f(t) que deseamos muestrear. Llamaremos f*(t) a la función muestreada f(t) a intervalos periódicos de duración T.

Si la duración del pulso de muestreo es mucho menor al valor de T, y tiende a ser cero, la salida del muestreador puede aproximarse a un tren de impulsos ideal. La señal f*(t) se puede representar como:

$$f^*(t) = \sum_{k=0}^{\infty} f(t)\,\delta(t-k) \qquad \ldots\ldots 1.35)$$

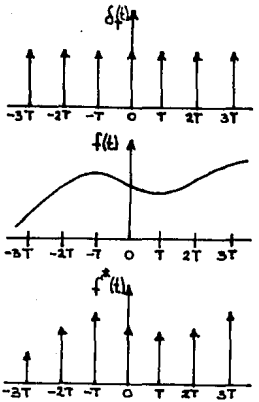


Figura 1.22

Si obtenemos la transformada de Laplace en ambos lados de la ecuación nos queda:

$$L\{ f^*(t) \} = F^*(s) = \sum_{k=0}^{\infty} f(kT)e^{-kTs} \qquad .....(1.36)$$

y definiendo el cambio de variable

$$z = e^{Ts} \qquad .....(1.37)$$

en (1.36) se obtiene:

$$F(z) = F^*(s)\Big|_{s=\ln z/T} \equiv \sum_{k=0}^{\infty} f(kT)z^{-k} \qquad .....(1.38)$$

A F(z) se la conoce como la transformada Z de f(t) [10].

Para que la transformada Z exista, debe de cumplirse la propiedad de convergencia:

$$Z\{f_i\} \text{ existe si } \lim_{k\to\infty} \sum_{i=0}^{k} f_i z^{-i} \text{ existe} \qquad .....(1.39)$$

Otra forma de evaluar la transformada Z de cualquier señal f(t) a partir de F(s) es usando el teorema de los residuos :

$$F(z) = \sum \left[\text{residuos de } \frac{F(\alpha)}{1-e^{T(\alpha-\alpha)}} \right] \qquad ....(1.40)$$

donde $F(\alpha)$ es F(s) evaluado en el polo α.

Desarrollando la ecuación anterior para polos múltiples, y substituyendo $z=e^{Ts}$, nos queda:

$$F(z)=\sum_{i=0}^{n} \frac{1}{(n-1)!} \frac{d^{n-1}}{d\alpha^{n-1}} \left[(\alpha-s_i)^n F(\alpha) \frac{1}{1-e^{T\alpha}z^{-1}} \right]_{\alpha=s_i} \qquad ..(1.41)$$

En el caso que F(s) contenga solamente polos simples, el número de residuos será igual al número de los polos [10].

Existen tablas en las que se relaciona una función en 's' con una en 'z', las cuales permiten un ahorro en tiempo al desarrollar una transformación.

1.7.4 MAPEO ENTRE LOS PLANOS 'S'-'Z'

La localización de los polos y ceros en el plano 's' ayuda a entender el comportamiento dinámico de un sistema continuo. Este mismo análisis es útil para los sistemas discretos en el plano 'z', por lo que es necesario establecer una relación entre las localizaciones de los polos y ceros en ambos planos.

La variable z^{-1} representa un corrimiento en el plano 'z', o un retraso puro e^{-sT} en el plano 's':

$$z^{-1} = e^{-sT} \qquad .....(1.42)$$

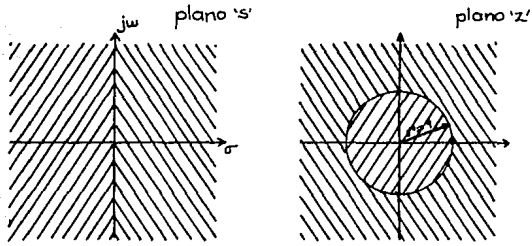
Esta es la relación de transformación entre las localizaciones de los polos del plano 's' y su localización en el plano 'z'.

Como la variable compleja 's' puede escribirse como $s = \sigma + jw$ entonces:

$$z = e^{(\sigma + jw)T} = e^{\sigma T} e^{jwT} \qquad .....(1.43)$$

$$|z| = e^{\sigma T} \ , \ \angle z = \omega T \qquad \ldots\ldots(1.44)$$

Sabemos que un sistema lineal analógico es estable si todos los polos de la función de transferencia están localizados en el semiplano izquierdo de 's'. En el plano 'z', este semiplano corresponde al interior de un círculo, de radio 1, al que se conoce con el nombre de círculo unitario.

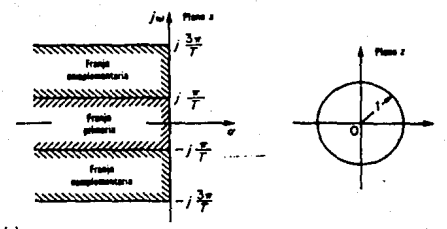


Mapeo entre los planos 's'-'z'.

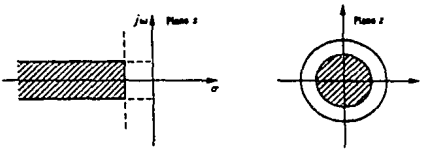
Figura 1.23

Comprobación: en el semiplano izquierdo, $\sigma<0$. Sustituyendo en la ecuación (1.44) la magnitud de z varía entre 0 y 1. El eje imaginario, $\sigma=0$, corresponde al círculo unitario en el plano 'z'. El interior del círculo corresponde al semiplano izquierdo 's'.

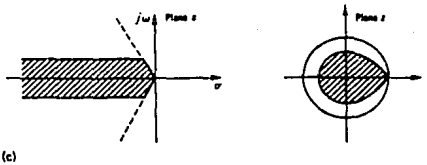
Sea un punto imaginario en el eje jw. Si lo desplazamos desde $-\pi/T$ hasta π/T a lo largo de este eje, tenemos que $|z|=1$ y que $\angle z$ varía desde $-\pi$ hasta π, en dirección contraria a las



(a)



(b)



(c)

Figura 1.24

manecillas del reloj. Si ahora lo desplazamos desde $-\pi/T$ hasta $3\pi/T$, el punto traza un círculo unitario. Cuando el punto en el plano 's' se desplaza desde $-\infty$ a $+\infty$, el círculo unitario se trazará infinito número de veces.

Se concluye que cada franja de ancho $2\pi/T$ en la mitad izquierda del plano 's' se proyecta a la parte interior del círculo unitario en el plano 'z' (fig. 1.24).

Por ser la señal muestreada una señal periódica, y tomando como referencia la correspondencia entre s y z tenemos que:

$$z = e^{sT} = e^{\sigma T}e^{j(\omega T \pm 2\pi k)} \qquad .....(1.45)$$

donde observamos que los polos en el plano 's', cuyas frecuencias sean múltiplos enteros de la frecuencia de muestreo, se localizan en la misma posición en el plano 'z'. Esto es un fenómeno que resulta en la superposición de los espectros.

En conclusión, la diferencia entre la localización de los polos en el plano 's' y 'z', es que en este último plano la localización depende de T [6,7,10].

1.7.5 PROPIEDADES DE LA TRANSFORMADA Z

Dadas la funciones $f_1(t)$ y $f_2(t)$ que poseen transformadas de Laplace y $F_1(z)=Z[f_1^*(t)]$ y $F_2(z)=Z[f_2^*(t)]$ existen, entonces

podemos enumerar las siguientes propiedades de la transformada Z [7] :

1. **Linealidad** : La transformada Z de una suma de funciones f_1 y f_2 es igual a la suma de sus respectivas transformadas.

$$Z\,[f_1^*(t) + f_2^*(t)] = F_1(z) + F_2(z) \qquad .....(1.46)$$

2. **Multiplicación por una constante** : Si A es una constante, entonces la transformada de una función multiplicada por una constante, es igual a la constante multiplicada por la transformada de la función:

$$Z\,[Af_1^*(t)] = A\,Z\,[f_1^*(t)] = A\,F_1(z) \qquad .....(1.47)$$

3. **Traslación temporal** : La traslación temporal (retardo) de una función es:

$$Z\,[f_1^*(t-pT)] = z^{-p}\,F_1(z) \qquad .....(1.48)$$

donde z^{-p} representa un retraso de p periodos, y para una traslación hacia la izquierda (adelanto) :

$$Z\,[f_1^*(t+pT)] = z^p\,F_1(z) - \sum_{i=0}^{p-1} f(iT)z^{p-i} \qquad .....(1.49)$$

donde z^p representa un delanto de p periodos, siendo p un entero.

4. **Traslación compleja** : Siendo A una constante, y $f_A(t)=e^{\pm At}f(t)$ tenemos que:

$$Z\,[f_A^*(t)] = Z\,[F^*(s\mp A)]$$

$$=[F(z)]\text{con } z \text{ reemplazado por } ze^{\pm AT}$$

$$=F(ze^{\pm AT}) = F_A(z) \qquad .....(1.50)$$

5. **Teorema del valor inicial** : Dado que tanto el $\lim_{z\to\infty}F(z)$ existe, como también existe

$$F(z) = Z\,[f^*(t)] = \sum_{k=0}^{\infty} f(kT)z^{-k} \qquad .....(1.51)$$

entonces :

$$f(t) = \lim_{t\to 0} f^*(t) = \lim_{z\to\infty} F(z) \qquad .....(1.52)$$

6. **Teorema del valor final** : Si la función $(1-z^{-1})F(z)$ no tiene polos sobre o dentro del círculo unitario, entonces:

$$\lim_{t\to\infty} f^*(t) = \lim_{t\to\infty} f(kt) = \lim_{z\to 1}[(1-z^{-1})F(z)] \quad ...(1.53)$$

determinará el valor final de $f^*(t)$.

Observación :

Si los polos se ubican fuera del círculo unitario, y ninguno está en el eje real positivo para $|z|>1$. $f^*(t)$ tendrá una

respuesta inestable, por lo que no se podrá aplicar el teorema del valor final.

7. Convolución Real : Si multiplicamos dos funciones $F_1(z)$ y $F_2(z)$ tendremos:

$$F_1(z)F_2(z) = Z\left[\sum_{i=0}^{k} f_1(iT)f_2(kT-iT)\right]$$

$$= Z\left[\sum_{i=0}^{k} f_1(kT-iT)f_2(iT)\right] \quad .....(1.54)$$

1.7.6 RESPUESTA EN FRECUENCIA

La transformada z de sistemas discretos define la respuesta del sistema a un impulso [9].

La repuesta en frecuencia de un sistema discreto tiene dos características: amplitud y fase.

Sea un sistema discreto F(z) excitado por una señal periódica v(z) y cuya salida es u(z):

$$F(z) = \frac{u(z)}{v(z)} \quad .....(1.55)$$

Para una señal periodica $e_i=sen(wiT)$ donde i=0,1,2,... la salida u_i estará dada por:

$$u_i = M\,sen(wiT + \theta) \qquad .....(1.56)$$

Por definición $z= e^{sT}$, y reemplazando $s=jw$, se tiene a la salida:

$$u(e^{jwT}) = F(e^{jwT})v(e^{jwT}) \qquad .....(1.57)$$

En base a la ecuación (1.30) se define:

a) Respuesta a la amplitud :

$$|M| = |F((e^{jwT})| \qquad .....(1.58)$$

b) Respuesta a la fase :

$$\theta = \angle\,(F(e^{jwT}))$$

$$\theta = \tan^{-1}\left(\frac{Im|F(z)|}{Re|F(z)|}\right)_{z=e^{jwT}}$$

$$\theta = \frac{1}{2j}\ln\frac{F(z)}{F(z^{-1})}_{z=e^{jwT}} \qquad .....(1.59)$$

1.8 COMPARACION ENTRE FILTROS DIGITALES Y ANALOGICOS

Después de presentar las características principales de los filtros digitales, se procede a realizar una comparación entre los filtros digitales y los analógicos.

La siguiente tabla presenta las principales características de los dos tipos de filtros [4]:

FILTROS DIGITALES	FILTROS ANALOGICOS
Análisis en tiempo discreto	Análisis en tiempo continuo
Utilizan ecuaciones en diferencias.	Utilizan ecuaciones diferenciales
Uso de la transformada Z y el plano Z.	Uso de la transformada de Laplace y el plano s.

Tabla 1.2

1.8.1 VENTAJAS Y DESVENTAJAS DEL USO DE FILTROS DIGITALES

Las principales ventajas de los filtros digitales son [4]:

1) **Estabilidad térmica** : Los cambios de temperatura que afectan a resistencias y capacitores se eliminan, ya que los filtros digitales se implementan con sumadores, multiplicadores y registros de corrimiento, o como un programa.

2) **Adaptabilidad** : Se pueden cambiar las especificaciones del filtro leyendo un nuevo juego de coeficientes y reemplazando los registros. El diseño puede ser programable y proveer implementación de cualquier orden.

Los filtros digitales también presentan desventajas, como son [4]:

1) **Procesar señales con ancho de banda limitada** : Como resultado del proceso de muestreo del convertidor analógico-digital, el ancho de banda para señales discretas está limitado a la mitad de la frecuencia de muestreo.

2) **Efecto de la longitud finita en los registros** : La implementación en "hardware" de un sistema en tiempo discreto puede resultar en una degradación del desempeño. Este efecto se debe al uso de registros para datos y coeficientes con un número finito de bits.

CAPITULO 2

FILTROS DIGITALES

2 FILTROS DIGITALES

En este capitulo se estudian dos tipos de filtros digitales: los filtros de respuesta al impulso infinita (IIR) y los filtros de respuesta al impulso finita (FIR). Primero se presentan las caracteristicas de cada uno de ellos, su forma de programación, asi como una comparación entre los dos tipos de filtros. Para finalizar el capitulo se especifican los métodos de discretización de los filtros IIR.

2.1 FILTROS DE RESPUESTA AL IMPULSO INFINITA (IIR)

La respuesta de un filtro IIR es una función que depende de tres factores: a) de la muestra de la señal de entrada en el instante n. b) de muestras de la señal de entrada de instantes anteriores a n y c) de muestras de la señal de salida de instantes anteriores a n. De ahí que reciban también el nombre de filtros recursivos.

La ecuación de un filtro IIR tiene la siguiente forma :

$$y(n) = \sum_{l=0} A_l x(n-l) - \sum_{k=0} B_k y(n-k) \qquad .....(2.1)$$

donde x(n) y y(n) son las señales de entrada y salida del filtro, y A_i, B_i son los coeficientes reales de las muestras de las señales de entrada y salida respectivamente.

La dependencia de la señal de salida en muestras anteriores de esta misma señal, da lugar a una duración infinita en la respuesta del filtro cuando los valores de la señal de entrada han cesado. Una desventaja de este tipo de filtros es que presentan ruido, debido a términos creados por la retroalimentación de errores aritméticos que pudieran presentarse en el sistema.

2.2 FILTROS DE RESPUESTA AL IMPULSO FINITA (FIR)

Se denominan filtros FIR a los filtros cuya señal de salida depende de a) la muestra de la señal de entrada en el instante n y b) de un número finito de muestras de la misma señal de entrada anteriores al instante n. También reciben el nombre de filtros no recursivos.

La ecuación de un filtro FIR es de la siguiente forma:

$$y(n) = \sum_{i=0}^{L} A_i x(n-i) \qquad .....(2.2)$$

donde x(n) y y(n) son las señales de entrada y salida del filtro, y A_i son los coeficientes de las muestras de la señal de entrada del filtro.

Este tipo de filtro tendrá una respuesta finita de duración L. Los coeficientes del filtro generalmente son simétricos, y como su respuesta no depende de muestras de la señal de salida, no se retroalimenta el ruido.

El método de diseño más común es obtener una respuesta al impulso de longitud finita truncando una secuencia de respuesta al impulso de infinita duración. A este método se le conoce como método de las ventanas [4].

2.3 PROGRAMACION DE FILTROS IIR

A continuación se explicará la manera de programar los filtros tipo IIR. Para la programación de los filtros FIR se siguen los mismos esquemas que para los filtros IIR, ya que al hacer los coeficientes $B_i=0$ en la ecuación (2.1) se obtiene una ecuación de un filtro FIR [4].

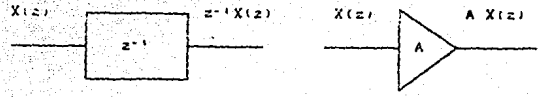
De la ecuación (2.1), se obtiene la función de transferencia de un filtro IIR, que es del tipo :

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{i=0}^{M} A_i z^{-i}}{1 + \sum_{i=1}^{N} B_i z^{-i}} \qquad M \leq N \quad ...(2.3)$$

Desarrollando esta ecuación para un filtro de segundo orden, y considerando $A_0 = 1$, se tiene :

$$H(z) = \frac{1 + A_1 z^{-1} + A_2 z^{-2}}{1 + B_1 z^{-1} + B_2 z^{-2}} \qquad .....(2.4)$$

Para la programación de los filtros se utilizan esquemas conocidos con el nombre de redes. La simbología de este tipo de esquemas se muestra en la figura 2.1, y es similar a la utilizada en los diagramas de bloques. En este caso, un bloque representa un retraso de la señal, y un triángulo significa que la señal se multiplica por un factor.



a) Retraso b) Multiplicador.

Figura 2.1

Existen dos tipos de redes. A continuación, explicamos cada una de ellas, tomando como ejemplo de programación a la ecuación (2.4).

2.3.1 FORMA DIRECTA 1

Este tipo de esquema (fig. 2.2) tiene como característica que la parte de la función de transferencia que representa una prealimentación (el numerador) se realiza al principio del programa, seguida posteriormente de la parte que representa a la retroalimentación negativa (el denominador).

2.3.2 FORMA CANONICA

En la figura 2.3 se observa que en este tipo de red el número de retardos es igual al orden del filtro. Esto es debido a que se utilizan elementos de retardo comunes entre las señales. A diferencia del esquema anterior, la retroalimentación negativa se implementa al principio, seguida de la prealimentación. Para comprobar que ambas figuras representan a la ecuación (2.4) se realiza un desarrollo a partir de la misma ecuación:

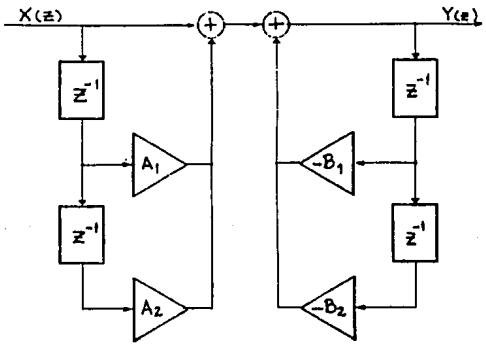


Figura 2.2 Forma Directa 1.

$$H(z) = \frac{Y(z)}{X(z)} \qquad .....(2.5)$$

Multiplicando y dividiendo por W(z) se obtiene:

$$H(z) = \frac{Y(z)}{X(z)} \frac{W(z)}{W(z)} \qquad .....(2.6)$$

Reacomodando términos:

$$H(z) = \frac{Y(z)}{W(z)} \frac{W(z)}{X(z)} \qquad .....(2.7)$$

Igualando las ecuaciones (2.4) y (2.7):

$$\frac{Y(z)}{W(z)}\,\frac{W(z)}{X(z)} = \frac{1 + A_1 z^{-1} + A_2 z^{-2}}{1 + B_1 z^{-1} + B_2 z^{-2}} \qquad .....(2.8)$$

$$\frac{Y(z)}{W(z)}\,\frac{W(z)}{X(z)} = 1 + A_1 z^{-1} + A_2 z^{-2} \left[\frac{1}{1 + B_1 z^{-1} + B_2 z^{-2}}\right] \qquad ..(2.9)$$

Separando el producto en dos factores:

$$\frac{Y(z)}{W(z)} = 1 + A_1 z^{-1} + A_2 z^{-2} \qquad .....(2.10)$$

$$\frac{W(z)}{X(z)} = \frac{1}{1 + B_1 z^{-1} + B_2 z^{-2}} \qquad .....(2.11)$$

Observando la fig. 2.3. se comprueba que la representación gráfica corresponde a la programación del desarrollo de la ecuación (2.4). Para filtros de un orden N, el esquema general se observa en la figura 2.4.

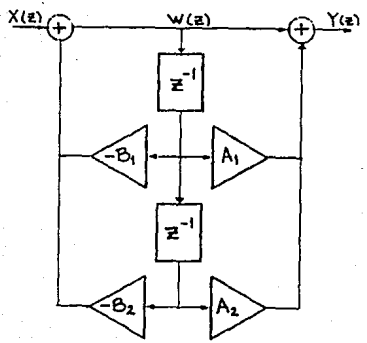


Figura 2.3 Forma canónica.

Se ha demostrado que cuando N tiene un valor grande, la exactitud requerida para la realización del sistema aumenta considerablemente, por lo que también aumenta en complejidad el "hardware" necesario para su implementación.

Leland y Jackson analizaron varias formas alternativas de programación de filtros digitales, y llegaron a la conclusión que las realizaciones de filtros en cascada y paralelo requieren menor precisión aritmética que la forma directa. En el siguiente inciso se explica como se realiza el desarrollo de un filtro en cascada.

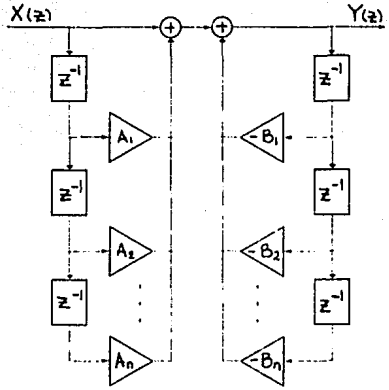


Figura 2.4a Forma directa i para un orden N.

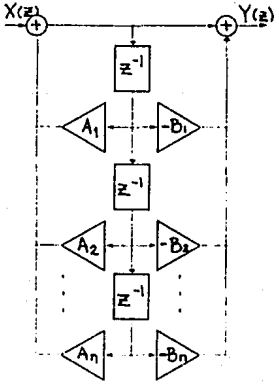


Figura 2.4b Forma Canónica para un orden N.

2.3.3 REALIZACION DE FILTROS EN CASCADA

De la figura anterior (2.4a) se observa que H(z) fue factorizada en dos funciones de transferencia :

$$H(z) = H_0(z)H_w(z) = \left[\frac{1}{\sum_{i=0}^{N} B_i z^{-i}}\right]\left[\sum_{i=0}^{M} A_i z^{-i}\right] \quad \ldots(2.12)$$

Esta ecuación nos presenta a H(z) como el producto de una función de transferencia $H_0(z)$, que corresponde al denominador, multiplicada por otra función de transferencia $H_w(z)$, formada con la porción del numerador de H(z). A esto se le llama factorización en cascada de la función de transferencia, que puede dividirse en dos partes: la porción de prealimentación, resultado de los ceros (determinada por el numerador), y la porción de retoalimentación negativa resultado de los polos (determinados por el denominador).

Se puede invertir el orden de la operación en cascada, sin que esto altere la respuesta del sistema.

A continuación se realiza una comparación entre los dos tipos de filtros digitales.

2.4 COMPARACION ENTRE FILTROS IIR Y FILTROS FIR

Para hacer la elección del tipo de filtro digital a estudiar, se tomaron en cuenta las siguientes características de cada tipo de filtro [4,11]:

1) Los filtros IIR tienen como ventaja que se puede diseñar una gran variedad de filtros a partir de las funciones de filtros analógicos. Se escoge un tipo de respuesta especifica (Butterworth, Chebyshev,etc.) y la ecuación del filtro digital se obtiene por sustitución directa. Un ejemplo del tipo de sustituciones que se realizan se observa en la tabla 2.1 al final de este capitulo.

2) El filtro IIR presenta una excelente respuesta a la amplitud. Desafortunadamente, su respuesta a la fase no es lineal.

3) Otra desventaja que presenta el filtro IIR es la posible retroalimentación del ruido debido a imprecisiones aritméticas en los cálculos.

4) Los filtros FIR se diseñan con fase lineal exacta. Esta característica es muy importante para aplicaciones en donde la distorsión en fase pueda degradar el desempeño del proceso.

5) La existencia de ruido debido a la precisión aritmética finita puede ser despreciable para realizaciones no recursivas, como en este tipo de filtro FIR.

6) Los problemas de exactitud en los coeficientes de los filtros IIR con una pendiente muy pronunciada son menos severos para filtros FIR con las mismas características.

7) No existen ecuaciones de sustitución directa para la realización de filtros FIR. Los metodos que utiliza son iterativos y requieren del uso de dispositivos más sofisticados.

8) La implementación de un filtro FIR de orden alto, utiliza más memoria y el costo de su realización física es mayor que en el caso de los filtros IIR.

9) Se recomienda el uso de los filtros IIR cuando existen limitaciones computacionales, o se van a diseñar pocos filtros.

Se escogió trabajar con los filtros IIR debido a que el desarrollo de un filtro FIR es muy costoso y de gran complejidad computacional.

2.5 FILTROS IIR

En esta sección se explica detalladamente como se obtienen

los filtros IIR a través de los métodos de discretización de filtros analógicos.

2.5.1 METODOS DE DISCRETIZACION DE FILTROS ANALOGICOS

Existen varios métodos para realizar la discretización de filtros analógicos [§ 2.1]. Los métodos que se explican a continuación son :

- Transformada z
- Transformación Bilineal
- Transformada Bilineal y pre-localización de polos.

2.5.2 METODO DE LA TRANSFORMADA Z

También se le conoce con el nombre de Transformación invariante al impulso. Consiste en transformar una función de transferencia f(s) en una función F(z) usando cualquier método propuesto en la sección 1.6.2. Para facilitar el trabajo, la función F(s) se descompone en fracciones parciales y posteriormente se transforma cada una de ellas al dominio z [9].

Es decir:

$$F(s) = \frac{A_1}{s + a_1} + \frac{A_2}{s + a_2} \qquad .....(2.13)$$

$$F(z) = Z[F(z)] = Z\left\{\frac{A_1 z}{s + a_1}\right\} + Z\left\{\frac{A_2 z}{s + a_2}\right\} \quad ..(2.14)$$

Algunas de las características de este método son:

a) Si F(s) es estable, también lo es F(z).
b) F(z) tiene la misma respuesta al impulso que F(s).
c) F(z) no presenta la misma respuesta en frecuencia de F(s).
d) F(z) transforma frecuencias múltiplos de w_s a la misma frecuencia en el plano z, por lo que se puede producir el fenómeno de superposición de frecuencias ("aliasing").

El mayor problema que puede presentarse en el dominio discreto es la repetición de los espectros de frecuencia. Para resolver este problema se suele poner en cascada con el filtro digital un filtro analógico paso bajas, y aumentar la frecuencia de muestreo.

2.5.3 TRANSFORMADA BILINEAL (TRANSFORMADA DE TUSTIN)

Es una forma de síntesis indirecta de un filtro digital, ya que permite obtener una función de transferencia de un filtro digital a partir de la función de transferencia de un filtro analógico, mediante un mapeo diferente entre las variables s y z.

Se basa en el proceso de integración trapezoidal [1,7].

Consideremos un integrador analógico representado por la ecuación (2.15) :

$$H(s) = \frac{1}{s} \qquad .....(2.15)$$

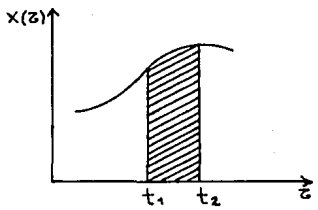


Figura 2.5 Integrador analógico.

La respuesta a una entrada impulso es, en el dominio del tiempo:

$$h(t) = \begin{cases} 1 & \text{para } t \geq 0 \\ 0 & \text{en cualquier otro caso.} \end{cases} \qquad .....(2.16)$$

La respuesta a una entrada cualquiera x(t) se da por la ecuación:

$$y(t) = \int_0^t x(\tau)h(t-\tau)\,d\tau \qquad .....(2.17)$$

Para $0 < t_1 < t_2$ se puede escribir:

$$y(t_2)-y(t_1) = \int_0^{t_2} x(\tau)h(t_2-\tau)\,d\tau - \int_0^{t_1} x(\tau)h(t_1-\tau)\,d\tau \qquad \ldots(2.18)$$

Si $h(t_2-\tau) = h(t_1-\tau) = 1$ para $0<\tau\leq t_1, t_2$, la ecuación puede simplificarse a:

$$y(t_2)-y(t_1) = \int_{t_1}^{t_2} x(\tau)\,d\tau \qquad \ldots(2.19)$$

y si t_1 tiende al valor de t_2

$$y(t_2)-y(t_1) \approx \frac{t_2-t_1}{2}\,[x(t_1)+x(t_2)] \qquad \ldots(2.20)$$

Reemplazando $t_1=nT-T$ y $t_2=nT$ podemos formar una ecuación en diferencias:

$$y(nT) - y(nT-T) = \frac{T}{2}\,[x(nT-T)+x(nT)] \qquad \ldots(2.21)$$

Esta ecuación representa un integrador digital cuya respuesta en el tiempo se aproxima a la respuesta de un integrador analógico. Utilizando la transformada z y la propiedad de retardo en la ecuación anterior nos queda:

$$Y(z) - z^{-1}Y(z) = \frac{T}{2}[z^{-1}X(z)+X(z)] \qquad .....(2.22)$$

y la ecuación de transferencia del integrador digital es:

$$H(z) = \frac{Y(z)}{X(z)} = \frac{T}{2}\,\frac{z+1}{z-1} \qquad .....(2.23)$$

La ecuación anterior nos da la transformación bilineal. Una función de transferencia analógica se puede transformar en una función de transferencia digital sustituyendo s por la relación:

$$s = \frac{2}{T}\,\frac{z-1}{z+1} \qquad .....(2.24)$$

Algunas de sus características son :

a) Facilidad de aplicacion.

b) Transforma todo el plano izquierdo de s en un círculo unitario en el plano z, por lo que no se presenta la superposición de frecuencias.

c) Si F(s) es estable, también lo es F(z).

d) No preserva la respuesta al impulso ni a la frecuencia.

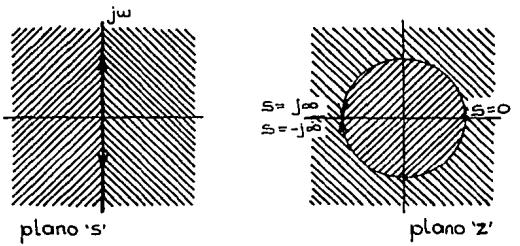


Figura 2.6 Mapeo de la transformación de Tustin.

Debido a que este tipo de transformación no preserva la respuesta a la frecuencia, se le agrego un procedimiento de pre-localización de polos ("prewarping") para compensar este efecto.

2.5.4 TRANSFORMADA BILINEAL Y PRE-LOCALIZACION DE POLOS

Comparemos la respuesta a la frecuencia de H(z) con respecto a H(s). Denominaremos H(jΩ) la respuesta a la frecuencia del sistema continuo, y H($e^{j\omega T}$) la respuesta a la frecuencia del sistema discreto [1,7].

Reemplazando s por jΩ y z por $e^{j\omega T}$ en la ecuación (2.24) nos queda:

$$j\Omega = \frac{2}{T}\,\frac{1-e^{-j\omega T}}{1+e^{-j\omega T}} \qquad \ldots\ldots(2.25)$$

Usando el teorema de De Moivre:

$$\Omega = \frac{2}{T} \tan\frac{wT}{2} \qquad .....(2.26)$$

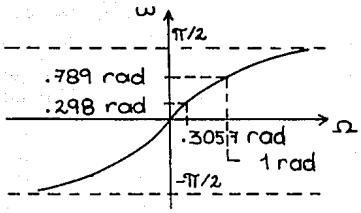


Figura 2.7 Relación entre Ω y w.

Esta ecuación refleja la distorsión entre Ω y w. Usando las ecuaciones (2.25) y (2.26) se obtiene que:

$$|F(s)|_{s=jw} = |F(z)|_{z=ejwT} \quad \text{siempre que } \Omega = \frac{2}{T} \tan\frac{wT}{2} \qquad ...(2.27)$$

La amplitud de F(s) en la frecuencia $f=\Omega_1$ es igual a la amplitud de F(z) en la frecuencia $f=w_1$. La transformación bilineal distorsiona la respuesta en frecuencia. Esto se observa en la fig. 2.8.

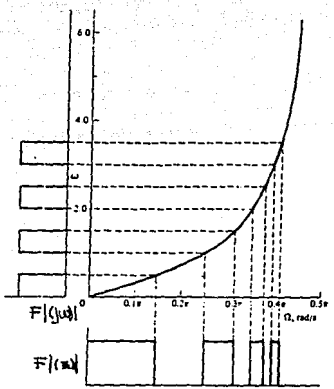


Figura 2.8 Distorsion de la respuesta en frecuencia.

En esta figura se observa que se comprimen las frecuencias entre $0<\Omega<\infty$ a un rango limitado digital $0<\omega T<\pi$.

Un procedimiento general para transformar una función F(s) a F(z) sería:

1. Para todos los polos y ceros deseados (s+a) hacer a=a' (procedimiento de pre-localización de polos), en donde:

$$a' = \frac{2}{T} \tan\frac{aT}{2} \qquad .....(2.28)$$

2. Transformar F(s,a') en F(z,a), sustituyendo s por :

$$s = \frac{2}{T}\frac{z-1}{z+1} \qquad .....(2.29)$$

Algunas de las características de este tipo de transformación son:

a) Mapea el lado izquierdo del plano s dentro del circulo unitario en el plano z.

b) Un sistema estable F(s) se transfoma en un sistema F(z) estable.

c) No hay superposición de espectros de frecuencia.

d) La respuesta en frecuencia es igual en los puntos de corte y cuando la frecuencia es cero.

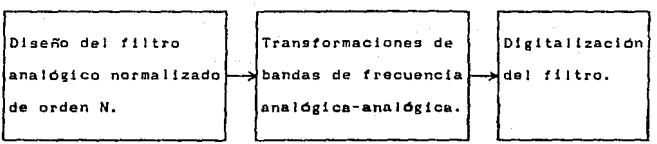
e) La respuesta al impulso y en frecuencia no se conservan.

Para finalizar el capítulo, se presentan los métodos que se utilizan para diseñar filtros IIR.

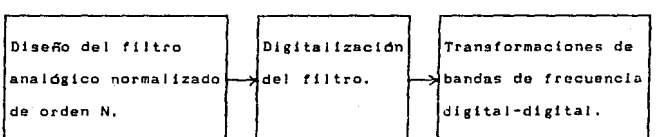
2.8 METODOS DE DISEÑO DE FILTROS DIGITALES

Existen dos formas de diseñar los filtros IIR [4]:

1a. forma



2a. forma



Durante el diseño se utilizó la primera forma para diseñar los filtros. Para la realización de las transformaciones de bandas de frecuencia analógica-analógica se utilizó la tabla 2.1, desarrollada por Constantinides. Con esta tabla se obtienen filtros paso altas, pasobandas, y de rechazo de banda a partir de la función de transferencia en 's' de un filtro paso bajas normalizado [2,7].

TIPO DE FILTRO	FRECUENCIA DE CORTE	SUSTITUIR CADA 's' POR	EN DONDE
PASO BAJAS	w_c	$K\,\frac{1-z^{-1}}{1+z^{-1}}$	$K = \Omega_a \cot \frac{w_c T}{2}$
PASO ALTAS	w_c	$K\,\frac{1+z^{-1}}{1-z^{-1}}$	$K = \Omega_a \tan \frac{w_c T}{2}$
PASO BANDA	w_1 = Frec. de corte inferior. w_2 = Frec. de corte superior. w_o = Frec. central.	$K\,\frac{z^{-2}-2\alpha z^{-1}+1}{1+z^{-2}}$	$\alpha = \frac{\cos \frac{(w_2+w_1)}{2} T}{\cos \frac{(w_2-w_1)}{2} T}$ $\alpha = \cos w_o T$ $K = \Omega_a \cot \frac{(w_2-w_1)}{2} T$

TABLA 2.1

CAPITULO 3

EQUIPO PARA EL DESARROLLO DEL FILTRO DIGITAL

3. EQUIPO PARA DESARROLLO DEL FILTRO DIGITAL

En este capítulo se establece la configuración del sistema mínimo realizado. Un sistema mínimo es aquel construido únicamente con los componentes necesarios para que realice las funciones requeridas por el usuario.

El sistema se divide en tres partes [7]:

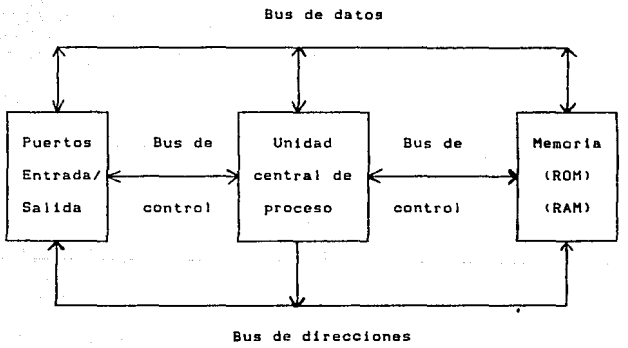


Fig. 3.1

La unidad central de proceso, o CPU, es la parte que procesa la información del sistema. Esta función se realiza mediante un microprocesador.

La sección de memoria tiene como función el almacenar datos o intrucciones. Se divide en dos partes: la memoria RAM, que es una memoria en la que se pueden escribir y leer datos, y memoria ROM, de la cual solo se pueden leer datos.

Los puertos de entrada y salida son dispositivos que permiten al usario obtener o introducir datos en el sistema. También se les conoce con el nombre de periféricos.

Estos tres bloques se comunican entres sí mediante los buses. Un bus es un canal de comunicación que utiliza el sistema para enviar datos, señales de control o alguna dirección. Existen tres tipos de buses:

- **BUS DE DIRECCIONES** : EL CPU introduce en este bus la dirección de una localidad de memoria en la que se va a escribir o leer un dato. También se usa para direccionar a los puertos de entrada o salida.

- **BUS DE DATOS** : Bus bidireccional en el que un dato puede venir de o ir hacia la memoria, CPU o puertos. Los dispositivos de entrada de datos que estén conectados a este bus deben ser de tres estados: el dispositivo debe estar en estado de alta impedancia, excepto cuando se le direccione para leer un dato.

- **BUS DE CONTROL** : Está compuesto por las señales de control del sistema, como son las señales de lectura en memoria o puerto.

A continuación se explica más detalladamente cada bloque del sistema realizado.

3.1 EL MICROPROCESADOR

Se decidió utilizar el microprocesador Z80 porque es muy fácil de utilizar. Además, por ser un sistema sencillo, no se requería de un microprocesador de mayor capacidad. El Z80 es un microprocesador de 8 bits de registros generales, contiene dos juegos de 6 registros de propósito general principal y alterno, que pueden ser usados como registros individuales o en pares. Además, posee dos juegos de acumuladores y registros de banderas [12].

En la fig. 3.2 se muestra el diagrama a bloques del Z80, en el que se observa sus funciones principales. A continuación se explica la función de cada uno de los bloques:

- **ALU** : Es la unidad lógica y aritmética, en donde se realizan operaciones aritméticas como la suma o la resta, y operaciones lógicas como son AND, OR, etc.

Dentro de los registros asociados a la ALU se encuentra el acumulador y el registro de banderas.

El acumulador es un registro de 8 bits que puede contener un dato o el resultado de la operación realizada por la ALU.

El registro de banderas se modifica después de cada operación lógica o aritmética. Se compone de cinco banderas que pueden tomar el valor de "1" ó "0" (set o reset respectivamente), indicando condiciones como son un acarreo producido en la operación. Las cinco banderas son la bandera de signo, de cero. de paridad o desborde, acarreo auxiliar, de suma o resta, y de acarreo.

- **REGISTRO DE INSTRUCCIONES** : Mantiene la siguiente instrucción a realizar antes de ser decodificada.

- **DECODIFICADOR DE INSTRUCCIONES** : Tiene como función decodificar el dato entregado por el registro de instrucciones, interpretar la instrucción y generar las señales necesarias para llevar a cabo la instrucción.

- **CONTROL DEL MICROPROCESADOR Y SINCRONIZACION** : Genera y recibe las señales de control del sistema.

- **REGISTROS** : Existen dos tipos de registros : los de uso específico y los de uso general. Los registros de uso específico son de uso exclusivo del microprocesador y son el contador de progarama (PC) y el apuntador de pila (SP). Ambos son registros de 16 bits.

Los registros de uso general se pueden usar para almacenamiento temporal de datos y pueden ser modificados en

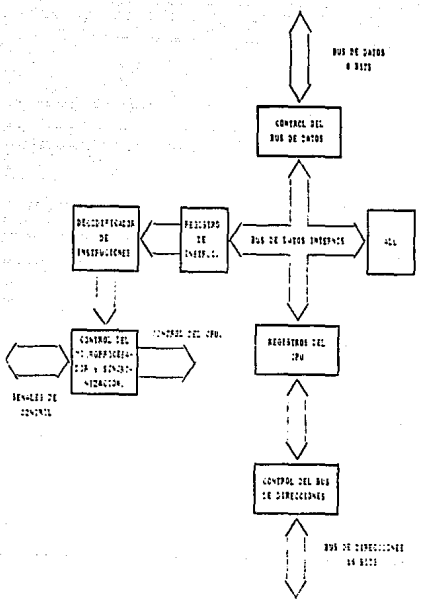


Figura 3.2 Diagrama a bloques del Z80.

cualquier momento que lo requiera el programa. Consiste en dos juegos de seis registros de 8 bits (B,C,D,E,H,L) que pueden ser utilizados para formar 3 registros pares (BD,DE,HL) de 16 bits cada uno.

Existen otros tres registros de 16 bits, como son el registro de interrupciones (I), el registro de regeneración o refresco de memoria (R), y los registros de índice (IX,IY).

REGISTROS PRINCIPALES		REGISTROS ALTERNOS	
A Acumulador	F Banderas	A' Acumulador	F' Banderas
B Propósito General	C Propósito General	B' Propósito General	C' Propósito General
D Propósito General	E Propósito General	D' Propósito General	E' Propósito General
H Propósito General	L Propósito General	H' Propósito General	L' Propósito General

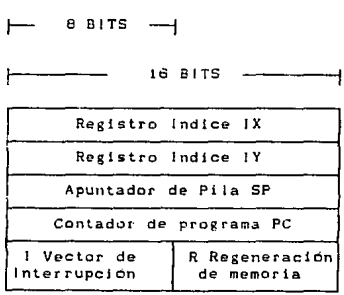


Fig. 3.3

El Z80 cuenta además con dos tipos de interrupciones: la interrupción no-mascarable ($\overline{NMI}$), y la interrupción mascarable ($\overline{INT}$).

El juego de instrucciones del Z80 es muy extenso, y se agrupan dentro de las siguientes categorías:

- Carga e intercambio.
- Aritméticas y lógicas.
- Búsqueda y transferencia de bloques.
- Rotación y desplazamiento.
- Manipulación de bits.
- Salto, llamada y retorno.
- Entrada y salida.
- Control del CPU.

3.2 EL BUS DE DIRECCIONES

La función de bus de direcciones es, como ya se mencionó anteriormente, indicar en que dirección se va a leer o escribir un dato. Se compone de 16 líneas binarias (A0-A15). A0 es el bit menos significativo, y A15 el bit más significativo. El Z80 utiliza únicamente los 8 bits menos significativos (A0-A7) para activar los puertos de entrada-salida. Para direccionar a localidades de memoria, utiliza los 16 bits. De esto se deduce que la capacidad de direccionamiento del Z80 es de 64k para memoria, y 256 para puertos [3].

Para compensar la corriente que demandan los puertos y la memoria se añade un buffer que aumenta la potencia de salida del bus, y de esta manera no se sobrecargue al microprocesador.

Fig. 3.4 Controlador de bus de direcciones

Para esta función se utilizaron tres circuitos 74367, con salida triestado. El bus se controla con la señal de $\overline{BUSAK}$ del microprocesador. Esta señal conmuta el control del bus de direcciones a un dispositivo externo. Mientras esta señal esté en un nivel alto, el bus permitirá la salida de todas las direcciones procedentes del Z80. Al momento de pasar $\overline{BUSAK}$ a un nivel bajo, el bus entrará en un estado de alta impedancia. Esto es útil cuando se emplea el acceso directo a memoria (DMA), que permite que la memoria sea escrita o leída por un dispositivo externo.

3.3 EL BUS DE DATOS

Este bus, al igual que el bus de direcciones, también está provisto de buffers. La única diferencia es que el bus de datos es bidireccional, ya que los datos fluyen en ambas direcciones. Cuando el Z80 escribe un dato en memoria, los datos van desde el procesador hacia la memoria, y cuando el procesador lee un dato de memoria, los datos se dirigen de la memoria al Z80 (3).

Con la ayuda de dos circuitos 8212 se realizó un controlador bidireccional. Uno de ellos permite el flujo de datos del microprocesador a otros dispositivos, mientras que el otro canaliza los datos hacia el Z80. El control del bus se realiza a través de una línea conectada a la señal $\overline{RD}$ del microprocesador. La señal de $\overline{RD}$ se encuentra generalmente en un nivel bajo, a

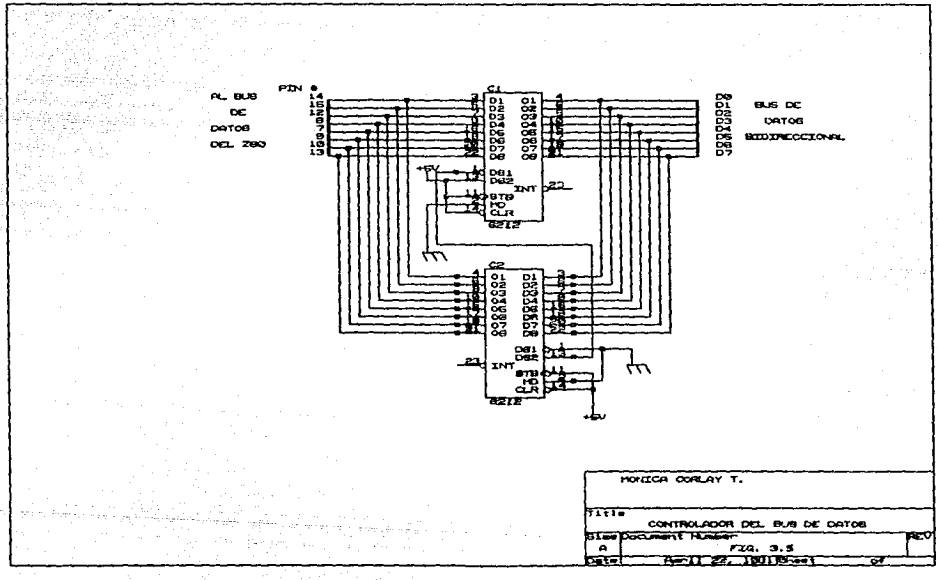


Fig. 3.5 Controlador de bus de datos

menos que se realize una operación de escritura. De esta manera el circuito de lectura de datos está normalmente activo, mientras que el circuito de escritura de datos se encuentra en un estado de alta impedancia. Cuando se desactiva la señal de $\overline{RD}$, se invierte el proceso.

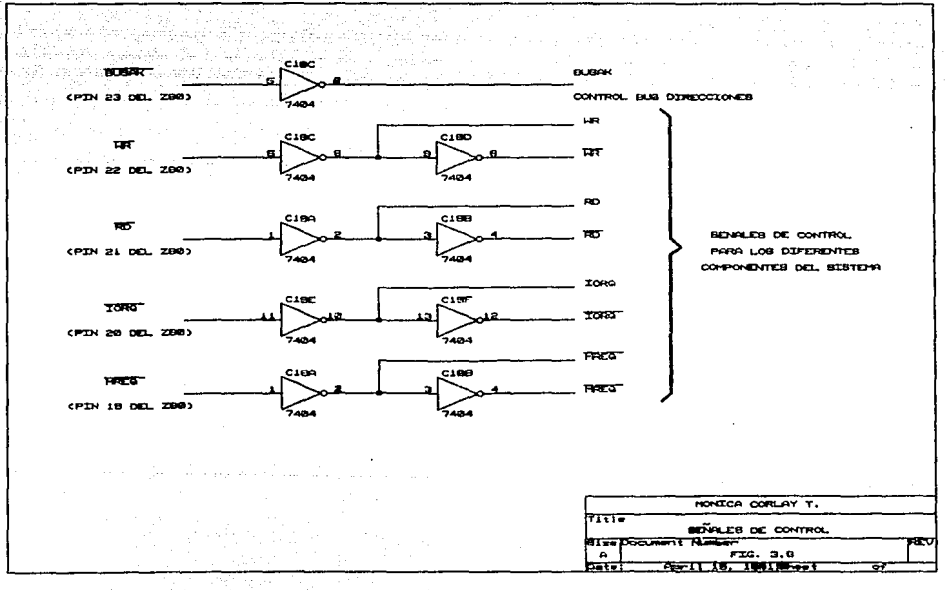
3.4 EL BUS DE CONTROL

Las señales del bus de control coordinan a los periféricos y los datos, y direcciona el flujo de datos en los tiempos adecuados. De las señales de control con que cuenta el Z80, cuatro son las principales, que rigen las operaciones básicas de memoria y de entrada/salida de puertos, y son [3]:

$\overline{MREQ}$.- Petición de memoria. Siempre que se necesite hacer un intercambio de datos entre el procesador central y la memoria, $\overline{MREQ}$ pasará a un nivel lógico "0".

$\overline{IORQ}$.- Petición de entrada/salida por puerto. Al igual que MREQ, pasará a un nivel lógico "0" cuando se requiera hacer una operación entre el procesador central y los puertos de entrada/salida.

$\overline{RD}$.- Petición de lectura. Estará en un nivel lógico "0" cuando se requiera leer algún dato de entrada, ya sea que provenga de un puerto o de memoria.


BUSAK
(PIN 23 DEL Z80)
C18C
7404
BUSAK
CONTROL BUS DIRECCIONES
WR
(PIN 22 DEL Z80)
C18C
C18D
7404
7404
WR
RD
(PIN 21 DEL Z80)
C19A
C19B
7404
7404
RD
SEÑALES DE CONTROL
PARA LOS DIFERENTES
COMPONENTES DEL SISTEMA
IORQ
(PIN 20 DEL Z80)
C18E
C18F
7404
7404
IORQ
MREQ
(PIN 19 DEL Z80)
C19A
C19B
7404
7404
MREQ
MONICA CORLAY T.
Title
SEÑALES DE CONTROL
Size
A
Document Number
FIG. 3.8
REV
Date:
Sheet

$\overline{WR}$.- Petición de escritura. Al realizar una escritura en puerto o en memoria, $\overline{WR}$ tendrá un nivel lógico "0".

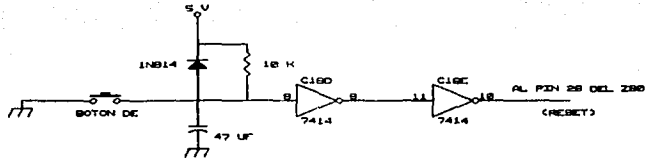
Al combinar las señales de $\overline{IORQ}$, $\overline{RD}$ y $\overline{WR}$ se generan pulsos para la lectura o escritura de un puerto. Esto se hace para diferenciar las señales de control de los puertos de entrada y salida durante las instrucciones de entrada/salida de datos a puertos. De la misma forma se combinan $\overline{MREQ}$, $\overline{RD}$ y $\overline{WR}$ para la lectura/escritura en memoria.

El bus de control está provisto de buffers, con los que también se obtienen las señales complementadas de $\overline{RD}$, $\overline{WR}$, $\overline{IORQ}$ y $\overline{MREQ}$.

La señal de $\overline{RESET}$ reinicializa el sistema. Provoca que el contador de programa se posicione en la dirección 0000H, reiniciando el programa. El sistema ha sido provisto de un reset manual, cuyo esquema vemos en la fig. 3.7.

3.6 MEMORIA

Se utilizó una EPROM 2716 para la sección de memoria ROM. En la ROM se guardan las instrucciones o programas que va a ejecutar el Z80. Su capacidad de almacenamiento es de 16384 bits, es decir, 2048 datos de 8 bits cada uno (2k). Una característica de este tipo de memorias es que puede ser borrada y reprogramada cada vez que sea necesario.



	MONICA CORLAY T.	
Title	RESET DEL SISTEMA.	
Size: A	Document Number: FIG. 3.7	REV: 1
Date: April 18, 1991	Sheet of	

Fig. 3.7 Reset del sistema

Para la parte de memoria RAM se utilizó una memoria estática (6116) con capacidad de 2k. Una ventaja de las memorias estáticas es que no necesita refrescarse continuamente, como las memorias dinámicas.

3.6 PERIFERICOS DEL SISTEMA

Los periféricos comunican al microprocesador y el medio externo. Los periféricos del sistema son: una entrada y una salida de señal de audio. La entrada de la señal de audio es a través de un muestreador-retenedor, y la salida es a través de un convertidor digital-analógico.

3.6.1 EL MUESTREADOR-RETENEDOR Y LOS CONVERTIDORES DIGITAL-ANALOGICO

La función de un muestreador-retenedor [§ 1.1.6] es convertir una señal contínua en un tren de pulsos, y mantener el valor del pulso entre intervalos de muestreo.

El muestreador-retenedor consiste básicamente en un interruptor, un capacitor y dos amplificadores operacionales, configurados como seguidor de voltaje. Al cerrar el interruptor, el capacitor se carga al valor de la señal que recibe del primer amplificador operacional. Al abrir el interruptor, el capacitor

permanecerá cargado y el segundo amplificador entregará la señal que tiene el capacitor. El LF 398 es un circuito que realiza esta función.

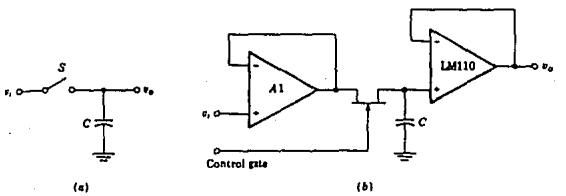


Fig. 3.8

El muestreador-retenedor junto con un convertidor digital analógico (DAC) realizan la conversión de una señal analógica en una señal digital. Un DAC convierte un número binario en un voltaje o corriente analógica. No se utilizó un convertidor analógico-digital en circuito integrado porque se quería que el Z80 realizara este proceso.

El sistema cuenta con dos DAC, uno para la conversión analogica-digital de la señal a procesar, y otro para convertir la señal digital procesada en una señal analógica.

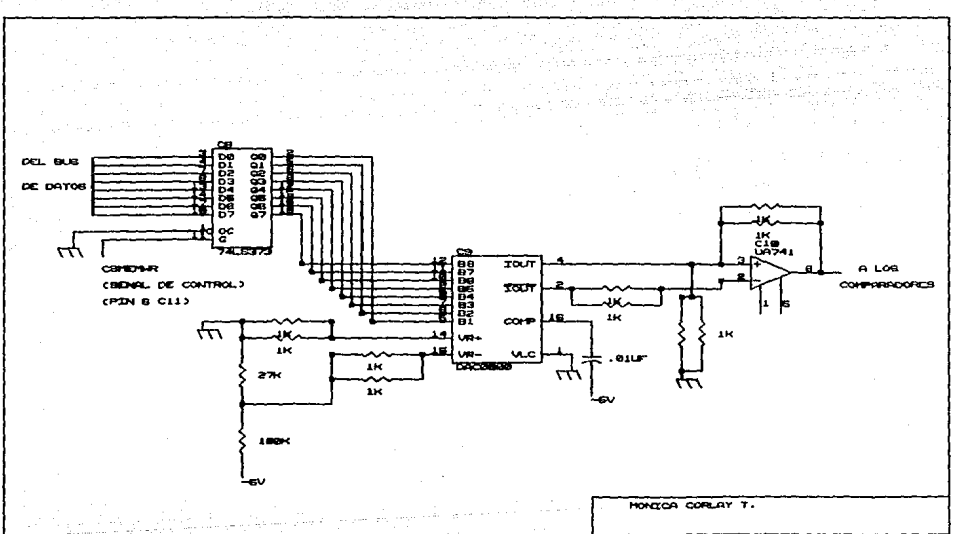


Fig. 3.9 Convertidor D/A para la conversión A/D

3.7 MAPEO A MEMORIA

El mapeo a memoria hace posible que los dispositivos periféricos de entrada/salida sean decodificados como una memoria [5].

La ventaja más importante es que se pueden utilizar las instrucciones que se refieren a la memoria para leer o escribir datos en el dispositivo. Esto hace posible que la escritura y la lectura de un dato sean más rápidos, agilizando cualquier proceso que involucre a los dispositivos.

Como la frecuencia de muestreo del sistema tiene que ser lo más alta posible, se utilizó el mapeo a memoria para direccionar a los convertidores digital-analógicos y al muestreador-retenedor.

3.8 MAPA DE MEMORIA

El mapa de memoria indica los rangos de direcciones que ocupa cada dispositivo en el sistema. El mapa de memoria del sistema realizado se observa en la figura 3.10:

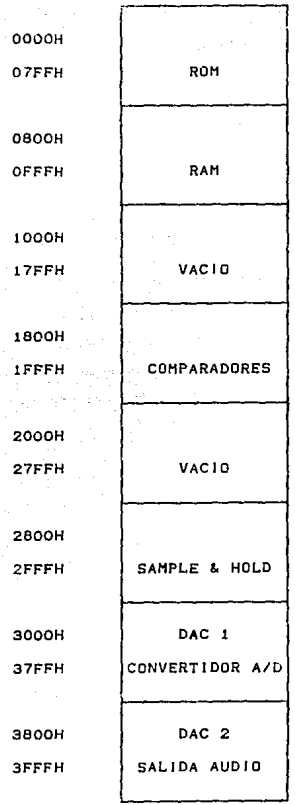

0000H
07FFH
ROM
0800H
0FFFH
RAM
1000H
17FFH
VACIO
1800H
1FFFH
COMPARADORES
2000H
27FFH
VACIO
2800H
2FFFH
SAMPLE & HOLD
3000H
37FFH
DAC 1
CONVERTIDOR A/D
3800H
3FFFH
DAC 2
SALIDA AUDIO

Fig. 3.10

CAPITULO 4

FASES DE DISEÑO

4 FASES DE DISEÑO

Este capítulo versa sobre los pasos que se siguen para el desarrrollo del software para sistema propuesto.

Actualmente existe una infraestructura que permite integrar las fases del diseño, en particular la fase concerniente al diseño y pruebas del filtro.

4.1 FASES

Las fases de diseño del software del sistema son :

* Planeación y diseño.
* Estructuración.
* Realización.
* Integración y pruebas.

- **Planeación y diseño:** En esta fase se definen las operaciones y procesos a realizar.

- **Estructuración:** Consiste en la ordenación de los procesos para obtener un funcionamiento óptimo.

- **Realización:** Cada proceso se diseña y prueba por separado para verificar su funcionamiento.

- Integración y pruebas: Finalmente se integran todos los procesos y se procede a realizar las pruebas necesarias en el sistema existente.

La parte más difícil es el diseño y la programación del filtro. Durante estas fases se contó con un gran número de herramientas. A continuación describimos brevemente como se realizó este proceso paso a paso, así como los programas utilizadas para este fin.

4.2 PAQUETE DISEÑADOR DE FILTROS DIGITALES (DFDP)

El primer paso es realizar un filtro digital. El programa DFDP (Digital Filter Design Package) [Atlanta Signal Processors Inc., 1984] diseña filtros digitales en base a características de respuesta en frecuencia seleccionadas por el usuario. Las opciones de diseño incluyen filtros IIR y FIR.

Este paquete fue diseñado para utilizarse conjuntamente con el procesador TMS320.

4.2.1 DISEÑO DE FILTROS IIR

El paquete diseña los filtros IIR usando la transformada bilineal [ƒ 2.5.3] (desplegado 4.1). En el desplegado 4.2 se observa el 1er. menú de opciones del paquete.

```
                ** Digital Filter Design Package **

                IIR BILINEAR TRANSFORM DESIGN PROGRAM

(C) COPYRIGHT,1984: ATLANTA SIGNAL PROCESSORS INC., VERSION 1.02
                    *          ASPI INTERNAL          *

THIS FILTER DESIGN PROGRAM DESIGNS RECURSIVE DIGITAL
FILTERS FROM BUTTERWORTH, CHEBYSHEV, AND ELLIPTIC
ANALOG PROTOTYPES.




 CR TO CONT
```

Despliegado 4.1

```
    *** IIR BILINEAR TRANSFORM MAIN MENU ***

ENTER THE NUMBER CORRESPONDING TO THE FILTER TYPE DESIRED

    1. LOWPASS
    2. HIGHPASS
    3. BANDPASS
    4. BANDSTOP

OR TAKE THE FOLLOWING ACTION

    5. READ SAVED FILE
    6. RETURN TO PROGRAM SELECTION MENU
    7. QUIT (RETURN TO DOS)

 OPTION DESIRED =
```

Despliegado 4.2

Para cada tipo de filtro se deben proporcionar los siguientes datos :

- Frecuencia de muestreo.
- Frecuencia(s) de corte de la región de paso banda.
- Frecuencia(s) de corte de la región de rechazo de banda.
- Rizo (en dB) en la región de paso banda.
- Rizo (en dB) en la región de rechazo de banda.

Como ejemplo se diseñó un filtro paso bajas, cuyas características se observan en el desplegado 4.3.

```
ALL FREQUENCIES MUST BE ENTERED IN KILOHERTZ

 ENTER SAMPLING FREQUENCY (KHZ) = 1.2

ENTER  LOWPASS FILTER CUTOFF FREQUENCIES

 PASSBAND CUTOFF FREQUENCY (KHZ) = .06
 STOPBAND CUTOFF FREQUENCY (KHZ) = .55
 PASSBAND RIPPLE =    .05
 STOPBAND RIPPLE =    .05
```

Desplegado 4.3

El programa calcula el orden del filtro para cada tipo de respuesta (Butterworth, Chevyshev, etc.). Si el orden del filtro no es satisfactorio, se puede regresar al menú inicial. En caso contrario, se selecciona el tipo de respuesta deseado (desplegado 4.4).

```
TO MEET YOUR SPECIFICATIONS WOULD REQUIRE FILTERS
OF THE FOLLOWING ORDERS:

      BUTTERWORTH:    1
      CHEBYSHEV I:    1
      CHEBYSHEV II:   1
      ELLIPTIC:       1

IF ONE OF THESE IS SATISFACTORY, ENTER THE
CORRESPONDING NUMBER:

            1. BUTTERWORTH
            2. CHEBYSHEV I
            3. CHEBYSHEV II
            4. ELLIPTIC
OTHERWISE
            5. RETURN TO IIR BILINEAR TRANSFORM MAIN MENU
            6. RETURN TO PROGRAM SELECTION MENU
            7. QUIT (RETURN TO DOS)

 OPTION DESIRED =
```

Desplegado 4.4

La siguiente pantalla del programa muestra los parámetros de diseño del filtro :

*** CHARACTERISTICS OF DESIGNED FILTER ***

	BAND 1	BAND 2
LOWER BAND EDGE	.00000	.55000
UPPER BAND EDGE	.06000	.60000
NOMINAL GAIN	1.00000	.00000
NOMINAL RIPPLE	.05000	.05000
MAXIMUM RIPPLE	.04682	.04803
RIPPLE IN DB	.39740	-26.36985

Desplegado 4.5

Nota: La opción de coeficientes cuantizados sólo es utilizable para el procesador TMS320.

Posteriormente aparece el menú del desplegado 4.6. Para visualizar las gráficas de respuesta del filtro, se utiliza la opción 2. Algunas de las graficas se observan en las figuras 4.1, 4.2, 4.3 y 4.4.

```
ENTER CORRESPONDING NUMBER FOR
 INSTRUCTION DESIRED

   1. AUTOMATICALLY INCREMENT FILTER ORDER
   2. PLOT RESPONSES
   3. DISPLAY FILTER COEFFICIENTS
   4. OUTPUT FILTER COEFFICIENTS
   5. QUANTIZE COEFFICIENTS

   6. RETURN TO IIR BILINEAR TRANSFORM MAIN MENU
   7. RETURN TO PROGRAM SELECTION MENU
   8. QUIT (RETURN TO DOS)

 OPTION DESIRED =
```

Desplegado 4.3

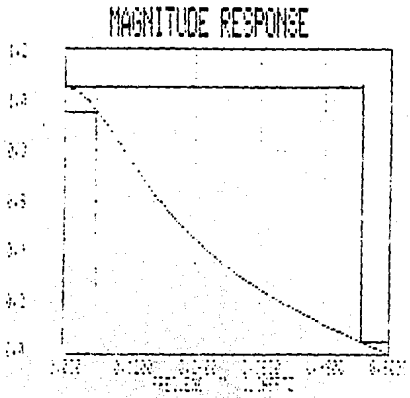


Figura 4.1

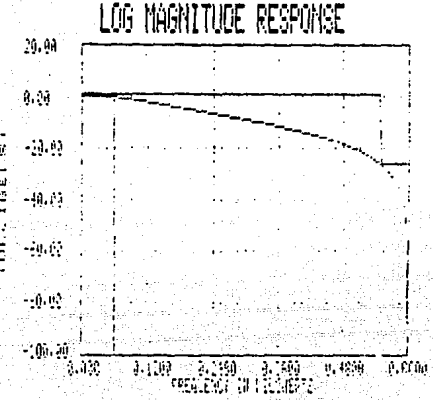


Figura 4.2

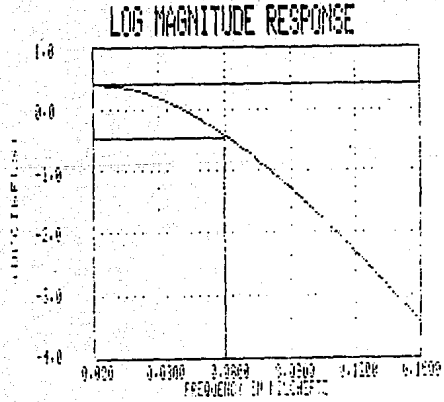


Figura 4.3

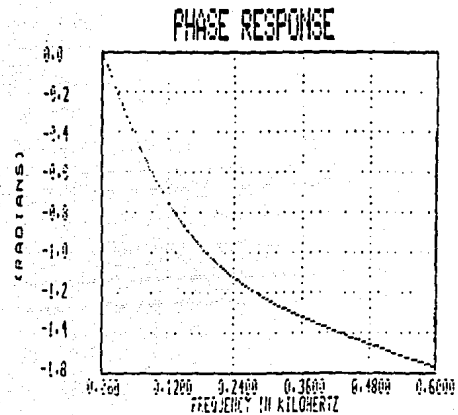


Figura 4.4

La opción 3 nos muestra los coeficientes del filtro. Se sustituyen en funciones del tipo :

$$F(Z)_i = \frac{B_{i,0} + B_{i,1}z^{-1} + B_{i,2}z^{-2}}{1 + A_{i,1}z^{-1} + A_{i,2}z^{-2}} \quad \ldots\ldots(4.1)$$

donde $F(Z)_i$ es una función de transferencia de segundo orden.

Los coeficientes del filtro se pueden guardar en algún archivo con la opción 5.

En el siguiente capitulo se muestra como se implementa este filtro en un programa.

INFINITE IMPULSE RESPONSE (IIR)
BUTTERWORTH LOWPASS FILTER
UNQUANTIZED COEFFICIENTS

FILTER ORDER = 1
SAMPLING FREQUENCY = 1.200 KILOHERTZ

I	A(I,1)	A(I,2)	B(I,0)	B(I,1)	B(I,2)
1	-.482721	.000000	.270748	.270748	.000000

Desplegado 4.11 Coeficientes del filtro diseñado.

4.3 PROGRAMACION DEL FILTRO

Cuando ya se obtuvo el diseño del filtro, se procede a la programacion de las operaciones y procesos necesarios para su ejecución.

Para realizar la programación de un proceso se siguen los siguientes pasos:

1.- Escritura del programa.

2.- Ensamblado.

3.- Simulacion.

4.- Programacion en EPROM.

Con un ejemplo sencillo se describen los pasos que sigue en un programa. El programa ejemplo SUMA.ASM realiza la suma entre dos números. El resultado se almacena en una localidad de memoria. El programa es escrito en un editor.

```
ETIQUETA   INSTRUCCION          COMENTARIOS

           *=$0000              ;Inicio de ROM.
           LD B,$04             ;B=$04
           ADD A,B              ;A=A+B
           LD ($0800),A         ;Carga en 0800H el resultado.
SUMA       JP SUMA              ;Alto.
```

4.3.1 ENSAMBLADO

Durante el ensamblado se realiza la conversión de los nemónicos al código hexadecimal que usa el microprocesador. El ASM80 es un programa ensamblador que realiza esta operación para el Z80.

El ASM80 se alimenta con el archivo que contiene el programa escrito con nemónicos. Se generan dos archivos : un archivo con el código Intel hexadecimal y otro con un listado que señala los errores de sintaxis que contiene el programa.

ASM Z-80

Por Eduardo Virueña Silva
E. S. F. M.

Para el Proyecto:

H. P. R. E. S. I. D. I. U. M.

Universidad Autonoma Metropolitana

¿ Nombre del archivo fuente ? SUMA.ASM
¿ Nombre del archivo de salida ? SUMA.PRN
¿ Nombre del archivo de codigo ? SUMA.HEX

Despiegado 4.12

El formato Intel hexadecimal contiene la dirección de la memoria, la cantidad de datos, los datos y el "cheksum" de estos elementos.

En caso de existir errores se repiten los pasos 1 y 2 hasta que el listado no contenga errores.

4.3.2 SIMULACION

La fase de simulación sirve para depurar un programa, ya que permite su ejecucion paso a paso. Durante esta fase se cuenta con

el programa AVSIMZ80: un paquete que permite observar la ejecución de un programa paso a paso. Es sumamente útil para detectar errores lógicos en la programación.

Con ayuda de la primer pantalla se programa la configuración del sistema hardware que se desea simular. La configuración que se utilizó para simular los programas es la A.

```
AVSIM Z80  Simulator/Debugger
Serial #  01022  Licensed by Avocet Systems, Inc.

Copyright (C) 1984,1985,1986,1987 by Ken Anderson Software
All Rights Reserved

Zilog Z80 Microcomputer Configurations

   A:   Z80
   B:   Z80 + CTC
   C:   Z80 + PIO
   D:   Z80 + PIO + CTC

Choose a CPU for simulation:
```

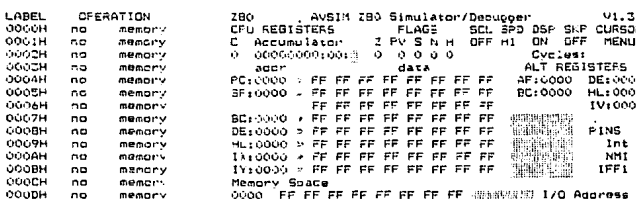
Desplegado 4.13

Con la segunda pantalla se carga el programa y se inicia la simulacion.

En este nivel se observan todos los registros del Z80 y presenta una serie de comandos propios de la simulación.

En la parte media se observan dos espacios de memoria ("Memory space") llamados ventanas. Permiten visualizar y modificar el contenido de las localidades de memoria que se encuentren dentro de un rango de direcciones pre-establecido.

La parte inferior de la pantalla muestra los comandos que puede realizar el simulador. Todos los comandos se activan escribiendo la primer letra del comando, o utilizando las flehas y la tecla < ENTER >.



Desplegado 4.14

A continuación se explican los comandos más importantes del simulador y su uso. La notacion utilizada es la siguiente :

- <C>OMANDO: la letra entre < > es la tecla a oprimir para ejecutar el comando seleccionado.

- <L>OAD : Carga el programa a simular. Al oprimir la tecla <L> aparece un sub-menú. Para cargar el programa usa la opción <P>ROGRAM. El archivo debe estar codificado en formato Intel.

Para el programa ejemplo se escribiría :

<L> <P> <SUMA.HEX> < ENTER >

- <S>ET : Configura el mapa de memoria con la sub-opción <M>emory map. Para configurar la memoria RAM se utiliza random-<A>ccess, y para la memoria ROM se usa read-<O>nly. Posteriormente se introduce la dirección más baja y después la dirección más alta.

Ejemplo: la secuencia

<S> <M> <A> <0800H> <0FFFH> < ENTER >

crea una memoria RAM de la dirección 0800H a la 0FFFH.

- <D>UMP : Configura las ventanas o espacios de memoria. Se selecciona entre la ventana superior <1> o la inferior <2>. La forma de visualización de los registros puede ser <A>bsoluta o <I>ndirecta. Por lo general se utiliza la absoluta. A continuación se introduce la dirección de las localidades de

memoria que nos interesa visualizar.

<D> <1> <A> <0800H> < ENTER >

- <R>ESET : Realiza el "reset" de la <C>PU.

<R>,<C>

- <H>ELP : Esta opción proporciona mayor información sobre el uso de los comandos y funciones el simulador.

Con la tecla <Esc> se tiene acceso a la pantalla o a los comandos. Los registros pueden ser modificados durante la ejecución del programa. Para alterar el contenido de algún registro o localidad de memoria, basta con posicionarse con los cursores en el lugar deseado y teclear el valor.

Existen tres velocidades de simulación del programa : baja, mediana y alta. La tecla <F5> establece la velocidad, la cual se muestra en la parte superior de la pantalla.

La tecla <F1> se ejecuta o detiene la simulación del programa.

Si se desea ejecutar una sola instrucción, se puede realizar con la tecla <F10>. Para retroceder una instrucción, se oprime <F9>.

4.3.3 SIMULACION DE INTERRUPCIONES

Debajo de los registros alternos del Z80 se encuentran las interrupciones $\overline{INT}$ y $\overline{NMI}$ del microprocesador. Con ellas se puede simular un programa que trabaje con interrupciones de este tipo.

Realizando las modificaciones necesarias en el programa anterior :

```
ETIQUETA   INSTRUCCION          COMENTARIOS

           *=$0000              ;Inicio de ROM.
           LD SP,$OFFF          ;SP=OFFFH.
LOOP       EI                   ;Activa interrupción INT
           IM 1                 ;en modo 1
           JP LOOP              ;Regresa a activar la interrupción

           *=$0038
           CALL SUMA            ;Llama a la subrutina SUMA
           RETI                 ;Regresa

SUMA       LD B,$04             ;B=$04
           ADD A,B              ;A=A+B
           LD ($0800),A         ;Carga en 0800H el resultado
           RET                  ;Regresa
```

Al activar el modo de interrupción 1, el programa ejecutara lo que haya en la dirección 0038H. De esta forma, la suma solo se realizará cada vez que el microprocesador reciba una señal de interrupción $\overline{INT}$.

Cada vez que la señal $\overline{INT}$ cambie su valor a cero, se activará la interrupción.

Cuando el funcionamiento del programa ha sido satisfactorio, se procede a programar la EPROM. Si el programa presenta errores de logica se repiten los pasos 1-3 hasta que el funcionamiento sea adecuado.

Previo a la programación, es necesario convertir el programa hexadecimal a un programa objeto, con lo que se obtiene el programa en lenguaje máquina.

4.3.4 GRABADO

Una memoria EPROM en blanco tiene todos los bits en "1". Los "0" se programan en las localidades deseadas. Para programar una memoria EPROM se aplica una carga a un pin de la memoria (la carga y el pin al que se aplicar la carga varian de acuerdo con el tipo de memoria). El pin correspondiente a la señal de OE debe estar en estado alto. Se escoge la dirección del dato. El dato se introduce en las terminales de salida mientras se aplica un pulso de 50 ms a la entrada $\overline{CE}/\overline{PGM}$. Cuando $\overline{OE}$ pasa a un estado bajo, la

memoria se puede leer. En caso de permanecer en estado alto, se puede seguir programando más datos en el dispositivo [6].

El grabador controla todos estos pasos haciendo este procedimiento transparente para el usuario que desea programar una EPROM.

El programa que controla al programador muestra un menú que se observa en el desplegado 4.15:

A continuación se explica el funcionamiento de algunos comandos :

- <E> : "Eprom type /Vpp" .- Tipo de EPROM a programar. Se despliegan los tipos de memorias que pueden ser programadas.

- <B> : "Blank check" .- Verifica que la memoria esté en blanco.

- <L> : "Load disk object file in buffer" .- Carga el programa en memoria.

- <C> : "Copy" .- Copia el programa en memoria a la EPROM.

- <V> : "Verify" .- Compara el contenido de la EPROM con el del programa en memoria.

- <Q> : "Quit" .- Salida del programa.

```
 Modular Circuit Technologies  Eprom Writer  V-1.0
         model : MCT-EPROM      (C) 1987
*****************************************************
Select from the following :
  <E>: Eprom type / Vpp ---- (2764 /21V)
  <Q>: Quit.
  <L>: Load disk object file in buffer.
  <S>: Save buffer on disk.
  <D>: Display, modify, checksum or print buffer.
  <B>: Blank check.
  <R>: Read ------------  in  buffer address 0000.
  <V>: Verify ---------- with buffer address 0000.
  <C>: Copy ------------ from buffer address 0000.
  <1>: Read(A) --------  in  buffer  any address.
  <2>: Verify(A) ------ with buffer  any address.
  <3>: Copy(A) -------- from buffer  any address.
  <4>: Copy(B) --------  Blank  check  and  Copy.
  <5>: Verify(B) ------ Verify  &  Display error.

Result:
```

Descripção 4.15

Cada vez que se desee reprogramar una memoria es necesario borrarla. El procedimiento de borrado se realiza mediante la exposición de la ventana de cuarzo de la EPROM a rayos ultravioleta de longitud de onda corta. La carga almacenada se drena, borrando totalmente la información. El tiempo de exposición varía de acuerdo con el tipo de memoria, la potencia de la lámpara y la distancia entre la EPROM y la lámpara. El tiempo promedio de borrado es de 5 minutos.

CAPITULO 5

IMPLEMENTACION DE UN FILTRO DIGITAL

5. IMPLEMENTACION DE UN FILTRO DIGITAL

En este capitulo se realiza la implementación de un filtro digital. Primero, se hace una explicacion de lo que es el complemento a 2 de un número binario. Posteriormente, se desarrollan los algoritmos utilizados, como son la multiplicación y la búsqueda binaria. También se hace mención al procedimiento de conversion de números con punto decimal a numeros binarios. Finalmente, se efectua la programación del filtro.

5.1 ALGORITMOS DE MULTIPLICACION Y BUSQUEDA BINARIA

Antes de explicar los algoritmos, daremos un repaso de lo que es el complemento a 2.

5.1.1 EL COMPLEMENTO A 2

Definición. El complemento de un número N es una función de la longitud especifica de una palabra (número de bits). Es la diferencia positiva entre el número r^M y el número N. M es un entero positivo que indica el número de bits utilizados para representar N. Para el caso de un número de 8 bits, es la diferencia entre 2^8 y el número N.

El complemento a 2 de un número binario se obtiene al invertir cada bit del número y sumándole un "1" al resultado. Ejemplo:

-45 = -0101101

se invierte cada bit	1010010
se suma 1	+ 1
complemento a 2	1010011

La ventaja del uso del complemento a 2 es que dos números se pueden restar sumando al minuendo el complemento a 2 del sustraendo. Esto se utiliza en el algoritmo de la multiplicación.

A continuación, se explica el procedimiento a seguir con un ejemplo. Si se genera un acarreo, esto indica que el resultado de la operación es positivo. Si no existe acarreo, el resultado es negativo.

90	1011010		1011010	
- 45	-0101101	complemento a 2	+1010011	
45	.		1 | 0101101	= 45
		acarreo		

El resultado es positivo.

```
  45    0101101                          0101101
- 90   -1011010    complemento a 2      +0100110
-----  ---------                        ---------
- 45                                    0 1010011
                                        |
                  no hay acarreo        -
```

El resultado es negativo y está en la forma de complemento a 2, por lo que :

```
se invierte cada bit    0101100
se suma 1             +       1
                      ---------

complemento a 2       - 0101101 = - 45
```

Para regresar al numero de la forma del complemento a 2, se invierte cada bit y se suma un "1" al resultado.

Los números positivos y negativos se identifican utilizando el bit mas significativo como bit de signo. Los números positivos se representan con un 0 en el bit de signo, y los negativos se representan con su complemento a 2 y el bit de signo en "1".

El complemento a 2 con signo-magnitud de un numero negativo se obtiene de la siguiente manera:

1.- Escribir el positivo del número, incluyendo el bit de signo.

2.- Invertir cada bit, incluyendo el bit de signo.

3.- Añadir 1 al resultado.

Cuando se utiliza la notación signo-magnitud, se representan valores entre -128 y +127.

Bit de signo.

↓

+ 8	0	0001000
+ 80	0	1010000
+ 120	0	1111000
- 115	1	0001101
- 50	1	1001110
- 8	1	1111000

A continuación se describe el algoritmo utilizado para la multiplicación.

5.1.2 ALGORITMO DE BOOTH'S

En un sistema dependiente de la rapidez con que se efectúan las operaciones, es necesario utilizar procedimientos ágiles. El algoritmo de Booth's se utiliza por la rapidez con que realiza la

multiplicacion. Este algoritmo realiza la multiplicacion de 2 números en la forma de signo-magnitud.

Se basa en el hecho de que una hilera de 0's en el multiplicador no requiere de suma de productos parciales, sino simplemente un desplazamiento. A su vez, una hilera de 1's en el multiplicador desde el bit 2^n hasta el bit 2^m se puede expresar como :

$$2^{n+1} - 2^m \qquad n>m \qquad .....(5.1)$$

Por ejemplo, si el multiplicador es 0011110 (30), n=4 y m=1. Por lo que $2^{4+1} - 2^1 = 30$.

El algoritmo requiere del examen de los bits del multiplicador y el desplazamiento del producto parcial. Previo al desplazamiento, se debe sumar o restar el multiplicando al producto parcial de acuerdo las siguientes reglas :

1.- El multiplicando se resta del producto parcial después de encontrar el primer 1 en una hilera de 1's en el multiplicador.

2.- El multiplicando se suma al producto parcial después de encontrar el primer 0 (siempre y cuando hubiera un 1 previo) en una hilera de 0's en el multiplicador.

3.- El producto parcial no cambia cuando el bit es idéntico al bit previo del multiplicador.

Sean L y H los registros que contienen al multiplicador y al multiplicando respectivamente.

Se designa como Ln al bit menos significativo del registro L, y L_{n-1} representa al bit menos significativo anterior. La bandera de carry guarda el valor de L_{n-1}.

El diagrama de flujo se muestra en la fig. 5.1. El registro A y L_{n-1} están inicialmente en 0. El contador c se carga con un número igual al número de bits del multiplicador.

Se inspecciona los bits L_nL_{n-1} del multiplicador. Si son iguales a 10 significa que se ha encontrado el primer 1 en una hilera de 1's. Entonces se resta el multiplicando del producto parcial.

Si los dos bits son iguales a 01, quiere decir que el primer 0 en una hilera de 0's ha sido encontrado. Entonces se suma el multiplicando al producto parcial.

Cuando los dos bits son iguales, el producto parcial no cambia.

El siguiente paso es desplazar a la derecha el producto parcial y el multiplicador, incluyendo al bit L_{n-1}. En el producto parcial se ejecuta un desplazamiento aritmético a la derecha (ashr) con el que no se altera el signo.

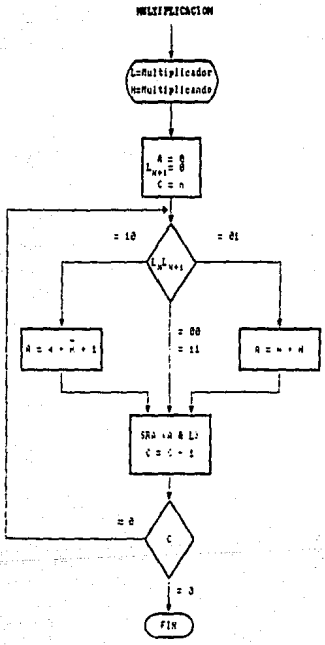

MULTIPLICACION
L=Multiplicador
M=Multiplicando
A = 0
L-1 = 0
C = n
= 10
= 01
= 00
= 11
A = A + M + 1
A = A + M
SRA (A & L)
C = C - 1
= 0
C
= 1
FIN
ALGORITMO DE BOOTH'S
Multiplicacion de numeros en complemento a 2 con signo.

Figura 5.1

Se decrementa el contador y se repite el proceso n veces. Al finalizar, n=0, el resultado lo forman los registros AL. A continuación se presenta un ejemplo numérico del algoritmo:

$$-26 \times -12 = 312$$

	signo-magnitud	signo-magnitud en complemento a 2
-26 =	10011010 =	11100110
-12 =	10001100 =	11110100

L = multiplicador = -26 = 11100110

H = multiplicando = -12 = 11110100

Complemento a 2 de H = 00001100

$L_n L_{n+1}$		A	L	L_n	L_{n+1}	C
	Inicial	00000000	11100110	0		8
00	Rotar	00000000	01110011	0		7
10	Restar H	00001100				
		00001100				
	Rotar	00000110	00111001	1		6
11	Rotar	00000011	00011100	1		5
01	Sumar H	11110100				
		11110111				
	Rotar	11111011	10001110	0		4
00	Rotar	11111101	11000111	0		3
10	Restar H	00001100				
		00001001				
	Rotar	00000100	11100011	1		2
11	Rotar	00000010	01110001	1		1
11	Rotar	00000001	00111000	1		0

Resultado = AL = 0000000100111000 = 312.

5.1.3 ALGORITMO DE BUSQUEDA BINARIA

El algoritmo de búsqueda binaria se utiliza por la rapidez con que ejecuta la conversión analógica-digital. Para un número de n bits se requiere únicamente de n pasos de búsqueda para completar la conversión. Este algoritmo llega a incrementar hasta 10 veces la velocidad del proceso de conversión realizada por cualquier otro procedimiento.

El algoritmo de búsqueda binaria funciona como un convertidor analógico-digital de aproximaciones sucesivas. Un registro de aproximaciones sucesivas (SAR) trabaja de la siguiente forma:

En el primer ciclo de reloj prende el bit más significativo del DAC. El SAR recibe una señal de un comparador, que indica si el voltaje del DAC es mayor o menor que el voltaje de entrada.

Si el comparador indica que el voltaje del DAC es menor que el de entrada, el SAR mantiene el bit más significativo en estado alto. En caso contrario, si el voltaje del DAC es mayor que el de entrada, el SAR pone un cero en el bit más significativo.

En el siguiente pulso, prende el siguiente bit más significativo. El SAR mantendrá el estado del bit dependiendo de la salida del comparador.

El procedimiento se repite hasta el bit menos significativo, terminando la conversion.

El problema de este algoritmo para un número de 8 bits, es que sólo termina cuando han pasado 8 pulsos de reloj. Con ayuda de un tercer comparador que indica si el voltaje del DAC es igual al de entrada, se agiliza la conversión. De este modo la conversion puede terminarse en menos de ocho pasos. La rutina de búsqueda binaria trabaja igual que un SAR.

Se tiene una tabla que contiene los valores desde 00H hasta FFH. El número 00H representa para el DAC 1 V, y le llamararemos el límite máximo. El valor de FFH representa -1 V y es el límite mínimo. El registro LMAX contiene la dirección en donde se encuentra el límite máximo, y el registro LMIN contiene la dirección del límite mínimo.

El código que se envía al DAC se obtiene de la siguiente fórmula:

$$INDICE = \frac{LMAX - LMIN}{2} + LMIN \qquad .....(5.2)$$

Reduciendo la fórmula :

$$INDICE = \frac{LMAX + LMIN}{2} \qquad .....(5.3)$$

El registro INDICE tiene el valor de una dirección, y el dato contenido en esa dirección es enviado al DAC. A continuación se lee el estado de los comparadores:

a) Si V DAC > V Entrada (D0=1), entonces el LMAX adquiere el valor del INDICE.

b) Si V DAC < V Entrada (D1=1), LMIN tendrá el valor del INDICE.

c) En caso de que V DAC = V Entrada (D3=1), la conversión ha terminado.

Para los casos a y b, se repite nuevamente el cálculo del indice. Se procede de esta manera hasta que :

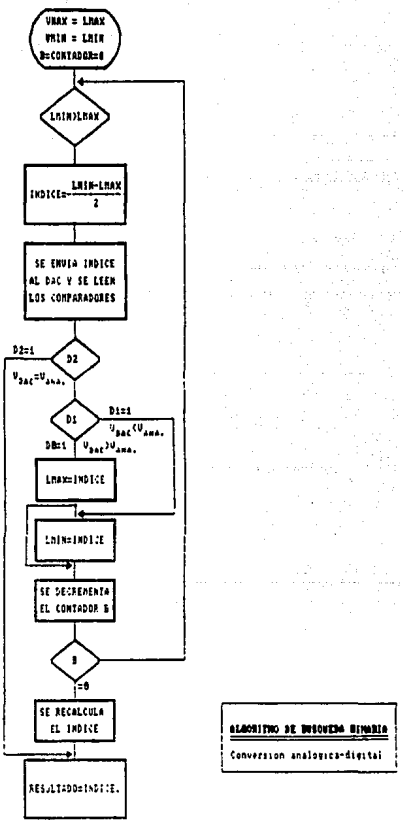

VMAX = LMAX
VMIN = LMIN
B=CONTADOR=8
LMIN>LMAX
INDICE=(LMIN-LMAX)/2
SE ENVIA INDICE AL DAC Y SE LEEN LOS COMPARADORES
D2
D2=1
$U_{DAC}=U_{ANA.}$
D1
D1=1
$U_{DAC}<U_{ANA.}$
D1=1
$U_{DAC}>U_{ANA.}$
LMAX=INDICE
LMIN=INDICE
SE DECREMENTA EL CONTADOR B
B
=0
SE RECALCULA EL INDICE
RESULTADO=INDICE.
ALGORITMO DE BUSQUEDA BINARIA
Conversion analogica-digital

Figura 5.2

a) El cálculo del índice se haya realizado 8 veces, en cuyo caso se procede a calcular el índice nuevamente. Con resultado obtenido, se da fin a la conversión.

b) Si el LMIN > LMAX, entonces también se da por terminada la conversión. Se toma como resultado el índice anterior.

El diagrama de flujo se observa en la fig. 5.2. y el listado de la rutina se encuentra en el apéndice A.

5.2 FORMATO Q8

El formato Q8 consiste en una transformación del conjunto de los números reales en el conjunto de los números naturales que se pueden representar con 8 bits. Esquemáticamente se representa en la fig 5.3.

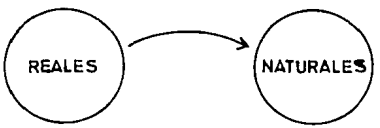


Fig. 5.3

Esta transformación es fácil de encontrar, dado que en la práctica se trabaja con un subconjunto de los números reales, caracterizado por los valores permitidos para las señales analógicas; además esta transformación mapea al cero en el número cero; es decir, el cero de los números reales debe coincidir con

el cero de los números naturales. Llamando T a esta transformacion tenemos :

$$T: \mathbb{R} \rightarrow \mathbb{N} \qquad .....(5.4)$$

es decir, es una transformación que manda a los números reales a los naturales y cumple con :

$$T(0) = 0;\ T(max) = 127 \text{ y } T(min) = T(-max) = -127$$

Con este requisito y por ser lineal :

$$T(x) = Round \left[\frac{127\ x}{max} \right] = round\ (\Gamma x) \qquad .....(5.5)$$

en donde Γ es un factor de corrección. La función Round es para regresar un entero simple.

Deben observarse las siguientes propiedades:

1) $T(T(x)) \neq T(x)$.

2) $T(x)T(y) \neq T(x-y)$.

3) $T(x)T(y) = \Gamma^2 xy$, donde Γ es un factor de corrección.

4) $T(x+y) = T(x)+T(y)$. .....(5.6)

Esto es muy importante, por que al aplicar esta transformación a factores, estos deben de corregirse por separado (propiedad 3).

Ejemplo : Suponer una señal analógica de 5 Volts; y se quiere programar $y = 2.5x + 1$.

Por lo tanto se tiene que:

$$T(x) = \text{Round}\left[\frac{127\,x}{5}\right] \qquad .....(5.7)$$

Aplicando las transformaciones:

$$T(y) = T(2.5x + 1) \qquad .....(5.8)$$

$$T(y) = T(2.5x) + T(1) \qquad .....(5.9)$$

Como lo que se quiere evitar es la multiplicación de dos números reales (2.5x), se separa el producto y se tiene:

$$T(y) = T(2.5)T(x) + T(1) \qquad .....(5.10)$$

Debido a la propiedad 3 tenemos una doble aplicación de la transformación en T(2.5)T(x), es decir:

$$\text{Round } (\Gamma y) = \text{Round } (\Gamma 2.5)\text{Round } (\Gamma x) + \text{round } (\Gamma 1) \quad \ldots.(5.11)$$

$$\text{Round } (\Gamma y) = \text{Round } (\Gamma^2\ 2.5x) + \text{Round } (\Gamma 1) \quad \ldots..(5.12)$$

Esto es un error y para corregirlo, después de cada aplicación de la transformación a un producto de dos factores se divide el resultado entre Γ :

$$\text{Round } (\Gamma y) = \text{Round } \left[\frac{\Gamma^2}{\Gamma} 2.5x \right] + \text{Round } (\Gamma 1) \quad \ldots(5.13)$$

Entonces:

$$\text{Round } (\Gamma y) = \text{Round } (\Gamma\ 2.5x) + \text{Round } (\Gamma 1) \quad \ldots..(5.14)$$

y el resultado es el esperado.

En programación el dividir entre Γ equivale a multiplicar en este ejemplo por 5/127, o aproximadamente multiplicar por 10 y hacer 8 corrimientos a la derecha. Para el caso de señales de 1 volt, el factor de correción Γ equivale a multiplicar por 1/127, o multiplicar por 2, que es lo mismo que realizar un corrimiento

a la izquierda. Este es un factor de corrección que siempre se debe de utilizar, debido al tipo de transformaciones utilizadas.

A continuación se procede a explicar como se programó el filtro y la utilización de la rutinas antes descritas.

5.3 PROGRAMACION DE UN FILTRO DIGITAL

El filtro que se programó es el filtro paso bajas diseñado en el capítulo 4 [§ 4.2]. El orden de programación de procesos fue el siguiente:

1.- **Adquisición de datos.**- Contiene el proceso de muestreo y conversión de una señal analógica.

2.- **Procesamiento de datos.**- El programa del filtro procesa a la señal digitalizada.

3.- **Salida de datos.**- La señal digital procesada se convierte en una señal analógica.

4.- **Actualización de registros.**- Los registros que contienen las muestras de las señales de entrada y salida se actualizan.

A continuación se explica como se realizó cada proceso

5.3.1 PROCESO DE ADQUISICION DE DATOS

El proceso de muestreo de la señal tiene que llevarse a cabo a una frecuencia fija de 1.2 kHz. Con un oscilador a esta frecuencia conectado a la terminal de $\overline{INT}$ del microprocesador, se asegura de que el proceso será realizado a esa frecuencia. Esto quiere decir que la duración de todos los procesos restantes no debe ser mayor a 1/f. En caso de que la duración del proceso sea menor, el sistema tomará otra muestra hasta que se genere el pulso de interrupción.

La interrupción $\overline{INT}$ funcionará en modo 1. Esto quiere decir que cada vez que se active la interrupción $\overline{INT}$, el microprocesador se posicionará en la dirección 0038H, y empezará a ejecutar las intrucciones que se encuentren a partir de esa localidad. En la dirección 0038H, se encuentra la rutina de muestreo y conversión de la señal analógica en digital (§ 5.1.2).

Antes de procesar la señal digitalizada, es necesario realizar algunas transformaciones. Esto se debe a que el formato de datos que utiliza el DAC y el formato Q8 no coinciden. Sea n el dato adquirido. Entonces :

a) Si n=80H, esto quiere decir que es un cero, por lo que se le asigna un valor de 00H.

b) Si 00H<n<80H, un número positivo, el dato se resta de 80H, es decir n=80H-n.

c) Si 80H<n<FFH, un número negativo, entonces se obtiene el complemento a 2 con signo del dato.

Esta transformación también es necesaria aplicarla previa a la salida de datos para que siempre coincida con el formato que utiliza el DAC.

5.3.2 PROCESAMIENTO DE LA SEÑAL

Una vez que se cuenta con la señal digitalizada en el formato adecuado, se continúa con el siguiente proceso, que es el filtrado de la señal.

Para la programación del filtro se siguieron los siguientes pasos:

1.- Conversión de los coeficientes del filtro al formato Q8 y la forma de complemento a 2 con signo.

2.- Aplicación del algoritmo de Booth's para cada multiplicación.

3.- Suma de productos parciales.

Conversión de los coeficientes del filtro al formato Q8. y la forma de complemento a 2 con signo. Los coeficientes del filtro se transformaron al formato Q8 [§ 5.2]. Los coeficientes quedan de la siguiente forma:

COEFICIENTES	EN FORMATO Q6	EN COMPLEMENTO A DOS CON SIGNO
A(0,1)= 1	7F	
A(1,1)= -.482721	-3D	C3
A(2,1)= 0	0	
B(0,1)= .270748	22	
B(1,1)= .270748	22	
B(2,1)= 0	0	

TABLA 5.1

De la función de transferencia del filtro [§ 4.2] se obtiene la ecuación que se programa:

$$\frac{Y(z)}{X(z)} = \frac{B_{0,1} + B_{1,1}Z^{-1} + B_{2,1}Z^{-2}}{A_{0,1} + A_{1,1}Z^{-1} + A_{2,1}Z^{-2}} \quad .....(5.15)$$

Sustituyendo los valores de $A_{0,1}=1$ y $A_{2,1}=B_{2,1}=0$ en la función de transferencia se tiene que:

$$\frac{Y(z)}{X(z)} = \frac{B_{0,1} + B_{1,1}Z^{-1}}{1 + A_{1,1}Z^{-1}} \quad \ldots\ldots(5.16)$$

Reacomodando términos:

$$(1 + A_{1,1}Z^{-1})\,Y(z) = B_{0,1}\,X(z) + B_{1,1}Z^{-1}\,X(z) \quad \ldots\ldots(5.17)$$

Despejando Y(z):

$$Y(z) = B_{0,1}\,X(z) + B_{1,1}Z^{-1}\,X(z) - A_{1,1}Z^{-1}\,Y(z) \quad \ldots(5.18)$$

Obteniendo la ecuación recursiva:

$$Y(k) = B_{0,1}\,X(k) + B_{1,1}\,X(k-1) - A_{1,1}\,Y(k-1) \quad \ldots\ldots(5.19)$$

Sustituyendo los valores de los coeficientes en formato Q8 en la ecuación anterior:

$$Y(k) = 22\,X(k) + 22\,X(k-1) - (-3D)\,Y(k-1) \quad \ldots\ldots(5.20)$$

De esta ecuacion se observa que el coeficiente A_1, que se convierte en un número positivo, por lo que no será necesario el formato complemento a 2 con signo. La ecuación que se programó queda de la siguiente forma:

$$Y(k) = 22\ X(k) + 22\ X(k-1) + 3D\ Y(k-1) \quad .....(5.21)$$

Aplicación del algoritmo de Booth's para cada multiplicación. Es muy importante realizar las operaciones en el menor tiempo posible. Por esta razón, se realizó una rutina de multiplicación para cada coeficiente. De esta manera se optimizó el algoritmo de Booth's. Algunas de las modificaciones realizadas con este objeto son:

a) Si el multiplicando es cero, se asigna al resultado un cero y el programa sigue con el siguiente procedimiento.

b) Si el primer bit L_n del multiplicador es un cero, se omite el procedimiento de desplazamiento del acumulador, ya que es cero al principio. Esta consideración evita realizar pasos innecesarios y agiliza la rutina. El contador se decrementa una unidad cada vez que se omite este desplazamiento. Se puede continuar omitiendo los desplazamientos del acumulador hasta encontrar el primer 1 en el multiplicador. Desde ese momento, se debe seguir la rutina como está estructurada, solo que el contador tendrá el valor de 8-d, donde d es el número de desplazamientos omitidos.

c) Se omite el último corrimiento a la derecha (SRA A) para dejar el resultado ya dividido entre el factor Γ [§ 5.2].

Suma de productos parciales. Después de realizar cada producto, se le asigna una localidad de memoria en donde se guarda el resultado. Al terminar el proceso de multiplicación para cada coeficiente, se procede a sumar los productos parciales. Con esto se finaliza el proceso de datos.

5.3.3 SALIDA DE DATOS

El dato obtenido se convierte en una señal analógica, recordando que previamente se le aplica una transformación [§5.3.1] para que coincidan los formatos utilizados por el DAC y el formato Q8.

5.3.4 ACTUALIZACION DE REGISTROS

Las muestras de cada señal se actualizan. Las localidades de memoria en que se encuentran las muestras de las señales son:

MUESTRAS DE SEÑAL	LOCALIDAD DE MEMORIA
x(k)	$0A02
x(k-1)	$0A01
y(k)	$0902
y(k-1)	$0901

TABLA 5.2

Después de actualizar los registros, el sistema está listo para repetir todo el proceso.

5.4 PRUEBA DEL FILTRO

Para la prueba final del filtro, se le aplicó una señal senoidal de 1 volt de pico. Los rangos de frecuencia que se le aplicaron variaron entre 10 y 400 Hz.

Se observó que las señales con una frecuencia menor a la frecuencia de corte del filtro, no sufrieron atenuación. Es importante mencionar que la frecuencia de corte fc se movió a de un valor de 138 Hz a 118 Hz. Cuando el valor de la frecuencia de la señal de entrada se acercó al valor de la frecuencia de corte, la amplitud de la señal de salida fué decreciendo. Después del valor de fc, la señal se atenuó notablemente.

Para obtener la respuesta en frecuencia, se midieron las amplitudes de las señales de entrada y salida, se realizó el cociente y el resultado se convirtió a dB. La fráfica del filtro obtenida experimentalmente se muestra en la fig. 5.4. La curva A muestra la respuesta del filtro calculada con el programa diseñador de filtros [§ 4.2.1]. La curva B muestra la curva obtenida experimentalmente con el sistema realizado.

En cuanto a la fase, se observó que la señal de salida se defasaba conforme la señal de entrada se acercaba al valor de la frecuencia de corte. Para una frecuencia de 200 Hz aproximadamente, las señales de entrada y salida estaban defasadas (90 grados).

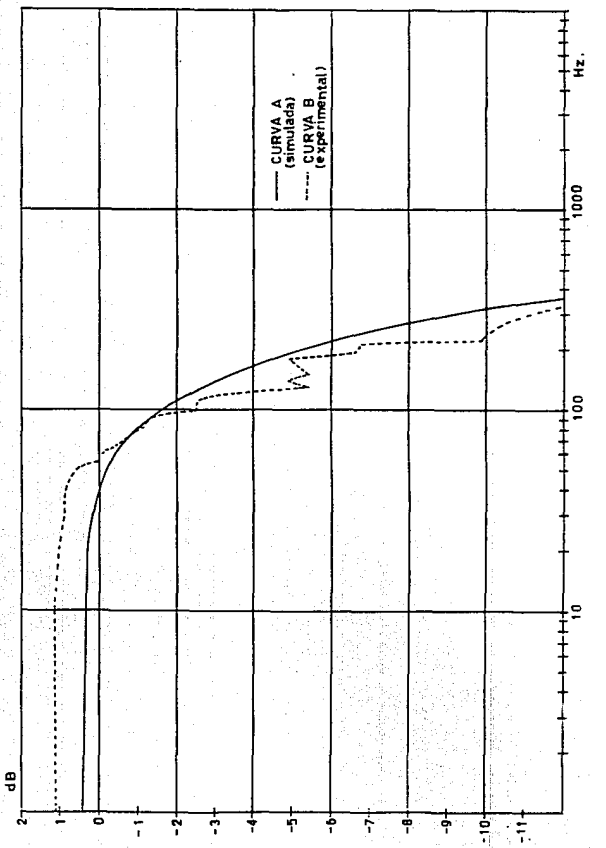


Figura 5.4 Respuesta en frecuencia del filtro IIR de 1er. orden.

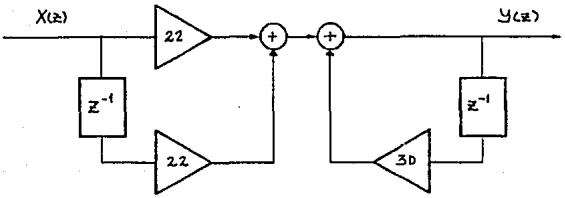


Figura 5.5 Configuración no optimizada del filtro IIR de 1er. orden

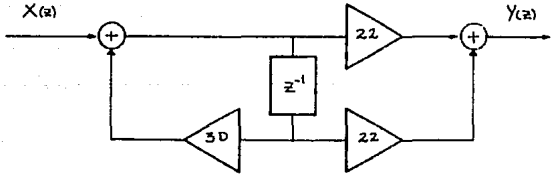


Figura 5.6 Configuración canónica del filtro IIR de 1er. orden

CONCLUSIONES

CONCLUSIONES.

Con este trabajo hemos llegado a conocer el funcionamiento de los filtros digitales tipo IIR. En base a las pruebas realizadas se obtuvieron las siguientes conclusiones:

La banda de paso del filtro de respuesta de amplitud se puede considerar plana (no produce distorsión), puesto que la variación máxima es de .7 dB y lo tolerable es 3 dB.

Los filtros digitales del tipo IIR pueden implementarse en sistemas digitales bajo ciertas restricciones:

Para un mejor desempeño del sistema es necesario un microprocesador con mayor velocidad. Esto es importante para que los procesos como la multiplicación de datos se realicen en el menor tiempo posible.

Hemos podido comprobar como es que la velocidad de muestreo es determinante en un sistema digital. Un sistema con una velocidad de muestreo muy baja no obtendrá una señal de información confiable. Si se tiene una velocidad de muestreo

altas, la señal no perderá sus caracteristicas.

El uso de algoritmos como el algoritmo de Booth's y el de búsqueda binaria mejoran la velocidad de los procesos de multiplicación y conversión analógica-digital respectivamente. Cualquier otro método utilizado para estos procesos degradaria el desempeño del sistema.

El número de bits que utiliza el microprocesador es determinante para la precision del sistema. Entre mayor sea el número de bits que utilice, el sistema tendrá un mejor desempeño.

Se comprobó la utilidad del formato Q8. Con esto se evita el uso de rutinas especiales para manejar números con punto flotante. También es importante mencionar que al programar un sistema en cascada se cometen menos errores de precisión aritmética.

Con el uso de paquetes computacionales se facilita la comprobación del funcionamiento de los algoritmos. Los programas de ensamblado y de simulación permiten la verificación de errores en un programa previo a su programación en la memoria.

La parte más difícil de realizar es la programación del filtro. Es muy importante tratar de realizar únicamente las operaciones necesarias para efectuar cada proceso en el menor tiempo posible. Con esto podemos obtener una mayor velocidad de muestreo, que permite un mejor desempeño del sistema.

Una forma de optimizar el sistema es utilizando un microprocesador adicional, cuya función sería la adquisición, conversión y salida de datos. El otro microprocesador solamente realizaría el procesamiento de los datos.

Otra solución puede ser el uso de microprocesadores como el TMS320 , que son especiales para procesar señales de audio, video, etc.

Algunas aplicaciones prácticas para filtros tipo IIR son en sistemas donde la fase de la señal a procesar no sea un factor determinante del desempeño del sistema. Los filtros digitales pueden aplicarse en sistemas que utilicen señales codificadas en forma binaria. Un ejemplo más, en el campo de la acústica, puede ser un ecualizador o un analizador de espectros digital.

APENDICE A

APENDICE A : PROGRAMA DEL FILTRO

IMPORTANTE : LA FRECUENCIA DE MUESTREO ES DE 1.2K.

EL FILTRO FUE DISEÑADO PARA TRABAJAR UNICAMENTE A ESTA FRECUENCIA DE MUESTREO.

```
1. 0000:                        *= $0000        ;Inicio de ROM.
2. 0000: 31 FF 0F               LD SP,$0FFF     ;SP=0FFFH.
3. 0003: D9                     EXX             ;Cambia al regis
                                                ;tro alterno.
4. 0004: 21 40 00               LD HL,$0040     ;HL=0040H. Con L
                                                ;se manda un
                                                ;pulso al mues-
                                                ;treador. Con H
                                                ;se desactiva.
5. 0007: 01 00 28               LD BC,$2800     ;BC=2800H. Es la
                                                ;direccion del
                                                ;muestreador.
6. 000A: D9                     EXX             ;Regresa al regis
                                                ;tro principal.
7. 000B: F3                     DI              ;Desactiva INT.
8. 000C: CD 3D 00               CALL FILTAB     ;Llama subrutina
                                                ;del filtro.
9. 000F: FB             LOOP    EI              ;Activa INT.
```

```
10. 0010: ED 56          IM 1          ;Int en modo 1.
11. 0012: C3 0F 00       JP LOOP       ;Regresa a acti-
                                       ;var INT.

12. 0015:                ;=============================

13. 0038:                *= $0038      ;Dirección 0038H
14. 0038: CD 3D 00       CALL FILTAB   ;Llama subrutina
                                       ;del filtro.
15. 003B: ED 4D          RETI          ;Regreso.

16. 003D:                ;=============================

                                       ;Inicio de la sub
                                       ;rutina del fil-
                                       ;tro.
```

PROCESO DE MUESTREO.

```
17. 003D: D9       FILTAB  EXX         ;Cambio a regis-
                                       ;tros alternos.
18. 003E: 7D               LD A,L      ;A=L.
19. 003F: 02               LD (BC),A   ;Activa muestrea
                                       ;dor.
20. 0040: 7C               LD A,H      ;A=H.
21. 0041: 02               LD (BC),A   ;Desactiva al
```

```
                                                ;muestreador.
22. 0042: D9                    EXX             ;Regreso a regis-
                                                ;tros principales
```

PROCESO DE CONVERSION ANALOGIA-DIGITAL.

(ALGORITMO DE BUSQUEDA BINARIA).

```
23. 0043:                       ;================================
24. 0043: 11 00 06              LD DE,$0600     ;DE=$LMAX
25. 0046: 21 00 07              LD HL,$0700     ;HL=$LMIN
26. 0049: 06 08                 LD B,$08        ;B=Contador.
27. 004B: 22 10 0C              LD ($0C10),HL   ;Guarda el conte
                                                ;nido de HL en
                                                ;0C10H.
28. 004E: 37            COND    SCF             ;Carry=1
29. 004F: 3F                    CCF             ;Carry=0
30. 0050: ED 52                 SBC HL,DE       ;HL=LMIN-LMAX
31. 0052: 38 2A                 JR C,FINO       ;Si LMAX>LMIN
                                                ;va a FINO.
32. 0054: 2A 10 0C              LD HL,($0C10)   ;HL=0C10H.
33. 0057: 19                    ADD HL,DE       ;HL=HL+DE.
34. 0058: CB 1C                 RR H            ;HL=HL/2
35. 005A: CB 1D                 RR L            ;
36. 005C: 7E                    LD A,(HL)       ;Se carga lo que
                                                ;hay en la direc-
```

```
                                        ;ción de HL (IN-
                                        ;DICE) en A.
37. 005D: 32 00 30      LD ($3000),A    ;Se envía el con-
                                        ;tenido de A al
                                        ;DAC.
38. 0060: 3A 00 18      LD A,($1800)    ;Se leen los
                                        ;comparadores.
39. 0063: 4F            LD C,A          ;C=A.
40. 0064: E6 04         AND $04         ;Se prueba el bit
                                        ;3. Si es uno, la
                                        ;conversión ha
41. 0066: 20 1E         JR NZ,FIN       ;terminado, en
                                        ;caso contrario
42. 0068: 79            LD A,C          ;se carga el dato
                                        ;de los compara-
43. 0069: E6 02         AND $02         ;dores en A y se
                                        ;prueba el bit 2.
44. 006B: 20 0B         JR NZ,FUN1      ;Si es igual a 1
                                        ;el programa con-
                                        ;tinúa en la li-
                                        ;nea 50.
45. 006D: 54            LD D,H          ;Si es igual a 0,
                                        ;el programa asig
                                        ;na el índice al
46. 006E: 5D            LD E,L          ;límita máximo.
                                        ;DE=HL.
47. 006F: 2A 10 0C      LD HL,($0C10)   ;Se lee el límite
                                        ;mínimo.
```

```
48. 0072: 22 10 0C          LD (#0C10),HL  ;Se guarda el va-
                                           ;lor de HL.
49. 0075: C3 7B 00          JP COUNTER     ;Salta a la linea
                                           ;51.
50. 0078: 22 10 0C  FUN1    LD (#0C10),HL  ;Se guarda el in-
                                           ;dice como el li-
                                           ;mite minimo.
51. 007B: 05        COUNTER DEC B          ;Decrementa el
                                           ;contador B.
52. 007C:  20 D0             JR NZ,COND    ;Si no es cero,
                                           ;regresa.
53. 007E: 2A 10 0C  FINO    LD HL,(#0C10)  ;Si B=0, se carga
                                           ;el valor del li-
                                           ;mite  minimo en
                                           ;HL.
54. 0081: 19                ADD HL,DE      ;Se recalcula el
                                           ;indice.
55. 0082: CB 1C             RR H           ;
56. 0084: CB 1D             RR L           ;
57. 0086: 7E        FIN     LD A,(HL)      ;Se carga en A el
                                           ;resultado de la
                                           ;conversión en A.
                                           ;Fin de la conver
                                           ;sión.
```

FORMATO DE DATOS

```
58. 0087:                ;=============================================
59. 0087: 01 00 80               LD BC,$8000     ;BC=8000H.
60. 008A: 81                     ADD A,C         ;Se suman A y C
                                                 ;para verificar
                                                 ;si el dato lei-
                                                 ;do es cero.
61. 008B: 20 05                  JR NZ,NEXT      ;Si no es cero,
                                                 ;continúa en la
                                                 ;línea 64
62. 008D:  3E 7F                  LD A,$7F       ;Si es cero, se
                                                 ;e asigna el va-
                                                 ;or de 7FH.
63. 008F: C3 A1 00               JP FINC         ;Salta a línea 71
64. 0092: B8             NEXT    CP B            ;Se compara el da
                                                 ;to con 80H.
65. 0093: 20 05                  JR NZ,X         ;Si no es 80H,sal
                                                 ;ta a la línea 68
66. 0095: 3E 00                  LD A,$00        ;Si es 80H, se le
                                                 ;asigna el número
                                                 ;00H.
67. 0097: C3 A1 00               JP FINC         ;Continúa en 71.
68. 009A:  FA A7 00      X        JP M,FINCO     ;Si el signo es
```

```
                                          ;positivo, conti-
                                          ;núa en 73.
69. 009D:  CB BF            RES 7,A       ;Si es negativo,
                                          ;se obtiene  su
                                          ;complemento a 2.
70. 009F: ED 44             NEG A         ;
71. 00A1: 32 02 0A    FINC  LD ($0A02),A  ;El dato se guar-
                                          ;da en la locali-
                                          ;dad 0A02H.
72. 00A4: C3 AD 00          JP XPOK       ;Salta a linea 78
73. 00A7: 4F          FINCO LD C,A        ;C=A.
74. 00A8: 78                LD A,B        ;A=B.
75. 00A9:  91               SUB C         ;A=A-C. El dato
                                          ;e resta de 80H.
76. 00AA: C3 A1 00          JP FINC       ;Salta a linea 71
                                          ;Los  datos están
                                          ;en formato  de
                                          ;complemento  a 2
                                          ;con signo.
```

PROGRAMA DEL FILTRO.

```
77. 00AD:                 ;
```

PARA TODAS LAS MULTIPLICACIONES REALIZADAS :

A=ACUMULADOR.
B=MULTIPLICANDO. (DATO A MULTIPLICAR).
L=MULTIPLICADOR. (COEFICIENTE DEL DATO).
C=COMPLEMENTO A 2 DEL MULTIPLICANDO.
HL=RESULTADO DE LA MULTIPLICACION.

MULTIPLICACION DE (22H)(X(K))

```
78. 00AD: 2E 11        XPOK    LD L,$11        ;L=11H. (Se utili
                                               ;za 11H en vez de
                                               ;22H porque se
                                               ;omite 1 corri-
                                               ;miento).
79. 00AF: 3A 02 0A             LD A,($0A02)    ;A=(0A02H)=X(K)
80. 00B2: 47                   LD B,A          ;B=A.
81. 00B3: ED 44                NEG             ;Se obtiene su
                                               ;complemento a 2.
82. 00B5:  20 06                JR NZ,OK1      ;Se prueba si el
                                               ;dato es cero.
83. 00B7: 21 00 00             LD HL,$0000     ;En caso afirma-
                                               ;tivo, se asigna
```

```
                                        ;cero al resulta-
                                        ;do.
 84. 00BA: C3 DA 00          JP CONT1    ;Salta a la linea
                                        ;102.
 85. 00BD: 4F          OK1   LD C,A      ;C=A.
 86. 00BE: CB 2F             SRA A       ;Rutina de multi,
 87. 00C0: CB 1D             RR L        ;plicación.
 88. 00C2: 80                ADD A,B     ;
 89. 00C3: CB 2F             SRA A       ;
 90. 00C5: CB 1D             RR L        ;
 91. 00C7: CB 2F             SRA A       ;
 92. 00C9: CB 1D             RR L        ;
 93. 00CB: CB 2F             SRA A       ;
 94. 00CD: CB 1D             RR L        ;
 95. 00CF: 81                ADD A,C     ;
 96. 00D0: CB 2F             SRA A       ;
 97. 00D2: CB 1D             RR L        ;
 98. 00D4: 80                ADD A,B     ;
 99. 00D5: CB 2F             SRA A       ;
100. 00D7: CB 1D             RR L        ;
101. 00D9: 67                LD H,A      ;
102. 00DA: 22 10 08   CONT1   LD ($0810),HL ;Guarda el resul-
                                        ;tado en 0810H.
```

MULTIPLICACION DE (22H)(X(K))

```
103. 00DD:              ;=============================================
104. 00DD: 2E 11                LD L,$11        ;Se sigue el mis-
                                                ;mo procedimiento
                                                ;que  en la ruti-
                                                ;na anterior, só
105. 00DF: 3A 01 0A             LD A,($0A01)    ;lo que el dato
106. 00E2: 47                   LD B,A          ;es X(K-1).
107. 00E3: ED 44                NEG             ;
108. 00E5: 20 06                JR NZ,OK2       ;
109. 00E7: 21 00 00             LD HL,$0000     ;
110. 00EA: C3 0A 01             JP CONT2        ;
111. 00ED: 4F           OK2     LD C,A          ;
112. 00EE: CB 2F                SRA A           ;
113. 00F0: CB 1D                RR L            ;
114. 00F2: 80                   ADD A,B         ;
115. 00F3: CB 2F                SRA A           ;
116. 00F5: CB 1D                RR L            ;
117. 00F7: CB 2F                SRA A           ;
118. 00F9: CB 1D                RR L            ;
119. 00FB: CB 2F                SRA A           ;
120. 00FD: CB 1D                RR L            ;
121. 00FF: 81                   ADD A,C         ;
122. 0100: CB 2F                SRA A           ;
123. 0102: CB 1D                RR L            ;
124. 0104: 80                   ADD A,B         ;
125. 0105: CB 2F                SRA A           ;
126. 0107: CB 1D                RR L            ;
```

```
127. 0109: 67                    LD H,A          ;
128. 010A: 22 12 08    CONT2     LD ($0812),HL   ;Guarda el resul-
                                                 ;tado en 0812H.
```

MULTIPLICACION DE (3DH)(Y(K-1))

```
129. 010D:              ;=========================================
130. 010D: 2E 3D                 LD L,$3D        ;L=3DH.
131. 010F: 3A 01 09              LD A,($0901)    ;A=(0901H)=Y(K-1)
132. 0112: 47                    LD B,A          ;Se sigue el mis-
                                                 ;mo procedimiento
                                                 ;que en las ruti-
                                                 ;nas anteriores.
133. 0113: ED 44                 NEG A           ;
134. 0115: 20 06                 JR NZ,OK3       ;
135. 0117: 21 00 00              LD HL,$0000     ;
136. 011A: C3 3E 01              JP CONT3        ;
137. 011D: 4F           OK3      LD C,A          ;
138. 011E: CB 2F                 SRA A           ;
139. 0120: CB 1D                 RR L            ;
140. 0122: 80                    ADD A,B         ;
141. 0123: CB 2F                 SRA A           ;
142. 0125: CB 1D                 RR L            ;
143. 0127: 81                    ADD A,C         ;
144. 0128: CB 2F                 SRA A           ;
```

```
145. 012A: CB 1D          RR L          ;
146. 012C: CB 2F          SRA A         ;
147. 012E: CB 1D          RR L          ;
148. 0130: CB 2F          SRA A         ;
149. 0132: CB 1D          RR L          ;
150. 0134: CB 2F          SRA A         ;
151. 0136: CB 1D          RR L          ;
152. 0138: 80             ADD A,B       ;
153. 0139: CB 2F          SRA A         ;
154. 013B: CB 1D          RR L          ;
155. 013D: 67             LD H,A        ;El resultado
                                        ;permanece en HL.
```

SUMA DE RESULTADOS PARCIALES.

```
156. 013E: ED 4B 10 08 CONT3   LD BC,($0810)  ;BC=(0810H)
157. 0142: 09                  ADD HL,BC      ;HL=HL+BC.
158. 0143: ED 4B 12 08         LD BC,($0812)  ;BC=(0812H)
159. 0147: 09                  ADD HL,BC      ;HL=HL+BC.
                                              ;Fin  del proceso
                                              ;de datos.
```

FORMATO DE SALIDA DE DATOS.

```
160. 0148:                ;=============================================
161. 0148:  7C                      LD A,H          ;El dato a salir
                                                    ;es H.
162. 0149: 32 02 09                 LD ($0902),A    ;Se guarda  Y(K)
                                                    ;en 0902H.
163. 014C: 01 00 80                 LD BC,$8000     ;Se sigue el mis
164. 014F: 81                       ADD A,C         ;mo procedimiento
165. 0150: 20 05                    JR NZ,NEXTC     ;utilizado en la
166. 0152: 3E 80                    LD A,$80        ;adquisición de
167. 0154: C3 66 01                 JP FINCC        ;datos.
168. 0157: B8             NEXTC     CP B            ;
169. 0158: 20 05                    JR NZ,XC        ;
170. 015A: 3E 80                    LD A,$80        ;
171. 015C: C3 66 01                 JP FINCC        ;
172. 015F: FA 6C 01       XC        JP M,FINCOC     ;
173. 0162: CB BF                    RES 7,A         ;
174. 0164: ED 44                    NEG A           ;
175. 0166: 32 00 38       FINCC     LD ($3800),A    ;Salida del dato
176. 0169: C3 7C 01                 JP RFFF         ;
177. 016C: FE 7F          FINCOC    CP $7F          ;
178. 016E: CA 77 01                 JP Z,FINC1C     ;
179. 0171: 4F                       LD C,A          ;
180. 0172: 78                       LD A,B          ;
181. 0173: 91                       SUB C           ;
182. 0174: C3 66 01                 JP FINCC        ;
183. 0177: 3E 00          FINC1C    LD A,$00        ;
184. 0179: C3 66 01                 JP FINCC        ;
```

ACTUALIZACION DE REGISTROS

X(K) → X(K-1)

Y(K) → Y(K-1)

```
185. 017C:                ;=================================================
                                                    ;X(K)→Y(K-1)
186. 017C: 21 02 0A  RFFF  LD HL,$0A02              ;HL=0A02H
187. 017F: 7E              LD A,(HL)                ;A=(HL)
188. 0180: 2D              DEC L                    ;HL=HL-1
189. 0181: 77              LD (HL),A                ;(HL)=A
                                                    ;Y(K)→Y(K-1)
190. 0182: 21 02 09        LD HL,$0902              ;HL=0902H
191. 0185: 7E              LD A,(HL)                ;A=(HL)
192. 0186: 2D              DEC L                    ;HL=HL-1
193. 0187: 77              LD (HL),A                ;(HL)=A
194. 0188: C9              RET                      ;Regreso de sub-
                                                    ;rutina.  Fin del
                                                    ;programa del
                                                    ;filtro.
```

TABLA PARA CONVERSION.

```
195. 0189:          ;=============================================================
196. 0600:                    *= $0600          ;En esta tabla
197. 0600: 00       LMAX      .BYTE $00         ;están los datos
198. 0601: 01                 .BYTE $01         ;para la búsqued
199. 0602: 02                 .BYTE $02         ;binaria (conver
200. 0603: 03                 .BYTE $03         ;sión analógica-
201. 0604: 04                 .BYTE $04         ;digital.
202. 0605: 05                 .BYTE $05
203. 0606: 06                 .BYTE $06
204. 0607: 07                 .BYTE $07
205. 0608: 08                 .BYTE $08
206. 0609: 09                 .BYTE $09
207. 060A: 0A                 .BYTE $0A
208. 060B: 0B                 .BYTE $0B
209. 060C: 0C                 .BYTE $0C
210. 060D: 0D                 .BYTE $0D
211. 060E: 0E                 .BYTE $0E
212. 060F: 0F                 .BYTE $0F
213. 0610: 10                 .BYTE $10
214. 0611: 11                 .BYTE $11
215. 0612: 12                 .BYTE $12
216. 0613: 13                 .BYTE $13
217. 0614: 14                 .BYTE $14
218. 0615: 15                 .BYTE $15
219. 0616: 16                 .BYTE $16
220. 0617: 17                 .BYTE $17
221. 0618: 18                 .BYTE $18
222. 0619: 19                 .BYTE $19
```

```
223. 061A: 1A                .BYTE $1A
224. 061B: 1B                .BYTE $1B
225. 061C: 1C                .BYTE $1C
226. 061D: 1D                .BYTE $1D
227. 061E: 1E                .BYTE $1E
228. 061F: 1F                .BYTE $1F
229. 0620: 20                .BYTE $20
230. 0621: 21                .BYTE $21
231. 0622: 22                .BYTE $22
232. 0623: 23                .BYTE $23
233. 0624: 24                .BYTE $24
234. 0625: 25                .BYTE $25
235. 0626: 26                .BYTE $26
236. 0627: 27                .BYTE $27
237. 0628: 28                .BYTE $28
238. 0629: 29                .BYTE $29
239. 062A: 2A                .BYTE $2A
240. 062B: 2B                .BYTE $2B
241. 062C: 2C                .BYTE $2C
242. 062D: 2D                .BYTE $2D
243. 062E: 2E                .BYTE $2E
244. 062F: 2F                .BYTE $2F
245. 0630: 30                .BYTE $30
246. 0631: 31                .BYTE $31
247. 0632: 32                .BYTE $32
248. 0633: 33                .BYTE $33
249. 0634: 34                .BYTE $34
250. 0635: 35                .BYTE $35
```

```
251. 0636: 36                .BYTE $36
252. 0637: 37                .BYTE $37
253. 0638: 38                .BYTE $38
254. 0639: 39                .BYTE $39
255. 063A: 3A                .BYTE $3A
256. 063B: 3B                .BYTE $3B
257. 063C: 3C                .BYTE $3C
258. 063D: 3D                .BYTE $3D
259. 063E: 3E                .BYTE $3E
260. 063F: 3F                .BYTE $3F
261. 0640: 40                .BYTE $40
262. 0641: 41                .BYTE $41
263. 0642: 42                .BYTE $42
264. 0643: 43                .BYTE $43
265. 0644: 44                .BYTE $44
266. 0645: 45                .BYTE $45
267. 0646: 46                .BYTE $46
268. 0647: 47                .BYTE $47
269. 0648: 48                .BYTE $48
270. 0649: 49                .BYTE $49
271. 064A: 4A                .BYTE $4A
272. 064B: 4B                .BYTE $4B
273. 064C: 4C                .BYTE $4C
274. 064D: 4D                .BYTE $4D
275. 064E: 4E                .BYTE $4E
276. 064F: 4F                .BYTE $4F
277. 0650: 50                .BYTE $50
278. 0651: 51                .BYTE $51
```

```
279. 0652: 52                .BYTE $52
280. 0653: 53                .BYTE $53
281. 0654: 54                .BYTE $54
282. 0655: 55                .BYTE $55
283. 0656: 56                .BYTE $56
284. 0657: 57                .BYTE $57
285. 0658: 58                .BYTE $58
286. 0659: 59                .BYTE $59
287. 065A: 5A                .BYTE $5A
288. 065B: 5B                .BYTE $5B
289. 065C: 5C                .BYTE $5C
290. 065D: 5D                .BYTE $5D
291. 065E: 5E                .BYTE $5E
292. 065F: 5F                .BYTE $5F
293. 0660: 60                .BYTE $60
294. 0661: 61                .BYTE $61
295. 0662: 62                .BYTE $62
296. 0663: 63                .BYTE $63
297. 0664: 64                .BYTE $64
298. 0665: 65                .BYTE $65
299. 0666: 66                .BYTE $66
300. 0667: 67                .BYTE $67
301. 0668: 68                .BYTE $68
302. 0669: 69                .BYTE $69
303. 066A: 6A                .BYTE $6A
304. 066B: 6B                .BYTE $6B
305. 066C: 6C                .BYTE $6C
306. 066D: 6D                .BYTE $6D
```

```
307. 066E: 6E                .BYTE $6E
308. 066F: 6F                .BYTE $6F
309. 0670: 70                .BYTE $70
310. 0671: 71                .BYTE $71
311. 0672: 72                .BYTE $72
312. 0673: 73                .BYTE $73
313. 0674: 74                .BYTE $74
314. 0675: 75                .BYTE $75
315. 0676: 76                .BYTE $76
316. 0677: 77                .BYTE $77
317. 0678: 78                .BYTE $78
318. 0679: 79                .BYTE $79
319. 067A: 7A                .BYTE $7A
320. 067B: 7B                .BYTE $7B
321. 067C: 7C                .BYTE $7C
322. 067D: 7D                .BYTE $7D
323. 067E: 7E                .BYTE $7E
324. 067F: 7F                .BYTE $7F
325. 0680: 80                .BYTE $80
326. 0681: 81                .BYTE $81
327. 0682: 82                .BYTE $82
328. 0683: 83                .BYTE $83
329. 0684: 84                .BYTE $84
330. 0685: 85                .BYTE $85
331. 0686: 86                .BYTE $86
332. 0687: 87                .BYTE $87
333. 0688: 88                .BYTE $88
334. 0689: 89                .BYTE $89
```

```
335. 068A: 8A                .BYTE $8A
336. 068B: 8B                .BYTE $8B
337. 068C: 8C                .BYTE $8C
338. 068D: 8D                .BYTE $8D
339. 068E: 8E                .BYTE $8E
340. 068F: 8F                .BYTE $8F
341. 0690: 90                .BYTE $90
342. 0691: 91                .BYTE $91
343. 0692: 92                .BYTE $92
344. 0693: 93                .BYTE $93
345. 0694: 94                .BYTE $94
346. 0695: 95                .BYTE $95
347. 0696: 96                .BYTE $96
348. 0697: 97                .BYTE $97
349. 0698: 98                .BYTE $98
350. 0699: 99                .BYTE $99
351. 069A: 9A                .BYTE $9A
352. 069B: 9B                .BYTE $9B
353. 069C: 9C                .BYTE $9C
354. 069D: 9D                .BYTE $9D
355. 069E: 9E                .BYTE $9E
356. 069F: 9F                .BYTE $9F
357. 06A0: A0                .BYTE $A0
358. 06A1: A1                .BYTE $A1
359. 06A2: A2                .BYTE $A2
360. 06A3: A3                .BYTE $A3
361. 06A4: A4                .BYTE $A4
362. 06A5: A5                .BYTE $A5
```

```
363. 06A6: A6                .BYTE $A6
364. 06A7: A7                .BYTE $A7
365. 06A8: A8                .BYTE $A8
366. 06A9: A9                .BYTE $A9
367. 06AA: AA                .BYTE $AA
368. 06AB: AB                .BYTE $AB
369. 06AC: AC                .BYTE $AC
370. 06AD: AD                .BYTE $AD
371. 06AE: AE                .BYTE $AE
372. 06AF: AF                .BYTE $AF
373. 06B0: B0                .BYTE $B0
374. 06B1: B1                .BYTE $B1
375. 06B2: B2                .BYTE $B2
376. 06B3: B3                .BYTE $B3
377. 06B4: B4                .BYTE $B4
378. 06B5: B5                .BYTE $B5
379. 06B6: B6                .BYTE $B6
380. 06B7: B7                .BYTE $B7
381. 06B8: B8                .BYTE $B8
382. 06B9: B9                .BYTE $B9
383. 06BA: BA                .BYTE $BA
384. 06BB: BB                .BYTE $BB
385. 06BC: BC                .BYTE $BC
386. 06BD: BD                .BYTE $BD
387. 06BE: BE                .BYTE $BE
388. 06BF: BF                .BYTE $BF
389. 06C0: C0                .BYTE $C0
390. 06C1: C1                .BYTE $C1
```

```
391. 06C2: C2                .BYTE $C2
392. 06C3: C3                .BYTE $C3
393. 06C4: C4                .BYTE $C4
394. 06C5: C5                .BYTE $C5
395. 06C6: C6                .BYTE $C6
396. 06C7: C7                .BYTE $C7
397. 06C8: C8                .BYTE $C8
398. 06C9: C9                .BYTE $C9
399. 06CA: CA                .BYTE $CA
400. 06CB: CB                .BYTE $CB
401. 06CC: CC                .BYTE $CC
402. 06CD: CD                .BYTE $CD
403. 06CE: CE                .BYTE $CE
404. 06CF: CF                .BYTE $CF
405. 06D0: D0                .BYTE $D0
406. 06D1: D1                .BYTE $D1
407. 06D2: D2                .BYTE $D2
408. 06D3: D3                .BYTE $D3
409. 06D4: D4                .BYTE $D4
410. 06D5: D5                .BYTE $D5
411. 06D6: D6                .BYTE $D6
412. 06D7: D7                .BYTE $D7
413. 06D8: D8                .BYTE $D8
414. 06D9: D9                .BYTE $D9
415. 06DA: DA                .BYTE $DA
416. 06DB: DB                .BYTE $DB
417. 06DC: DC                .BYTE $DC
418. 06DD: DD                .BYTE $DD
```

```
419. 06DE: DE                .BYTE $DE
420. 06DF: DF                .BYTE $DF
421. 06E0: E0                .BYTE $E0
422. 06E1: E1                .BYTE $E1
423. 06E2: E2                .BYTE $E2
424. 06E3: E3                .BYTE $E3
425. 06E4: E4                .BYTE $E4
426. 06E5: E5                .BYTE $E5
427. 06E6: E6                .BYTE $E6
428. 06E7: E7                .BYTE $E7
429. 06E8: E8                .BYTE $E8
430. 06E9: E9                .BYTE $E9
431. 06EA: EA                .BYTE $EA
432. 06EB: EB                .BYTE $EB
433. 06EC: EC                .BYTE $EC
434. 06ED: ED                .BYTE $ED
435. 06EE: EE                .BYTE $EE
436. 06EF: EF                .BYTE $EF
437. 06F0: F0                .BYTE $F0
438. 06F1: F1                .BYTE $F1
439. 06F2: F2                .BYTE $F2
440. 06F3: F3                .BYTE $F3
441. 06F4: F4                .BYTE $F4
442. 06F5: F5                .BYTE $F5
443. 06F6: F6                .BYTE $F6
444. 06F7: F7                .BYTE $F7
445. 06F8: F8                .BYTE $F8
446. 06F9: F9                .BYTE $F9
```

```
447. 06FA: FA                  .BYTE $FA
448. 06FB: FB                  .BYTE $FB
449. 06FC: FC                  .BYTE $FC
450. 06FD: FD                  .BYTE $FD
451. 06FE: FE                  .BYTE $FE
452. 06FF: FF           LMIN   .BYTE $FF
453. 0700:
```

BIBLIOGRAFIA.

[1] **Antoniou, Andreas.** "Digital Filters: Analysis and Design". Mc Graw Hill Book Company, 1979.

[2] **Bogner R.E and Constantinides A.G.,** "Introduction to Digital Filtering", John Wiley and Sons, 1975.

[3] **Ciarcia, S.** "Construya una Microcomputadora Basada en el Z80: Guía de diseño y Funcionamiento". Mc Graw Hill Book Company,1984.

[4] **De Fatta, David J. Lucas, Joseph G. and Hodgkiss William.** "Digital Signal Processing: A System Design Approach". John Wiley and Sons, 1988.

[5] **Hall, D.V.** "Microprocessors and Digital Systems". Mc Graw Hill Book Company, U.S.A. 1983.

[6] **Houpis C.H. and Lamont G.B.** "Digital Control Systems: Theory, Hardware, Software". Mc Graw Hill Book Company, U.S.A.,1985.

[7] **Katz, Paul.** "Digital Control Using Microprocessors".Prentice Hall International, Englewood Cliffs N.J. 1981.

[8] **Mano, Morris.** "Arquitectura de Computadoras". Ed. Prentice Hall Internacional, Colombia. 1983.

[9] Nichols, Elizabeth. Nichols, Joseph. Rony, Peter. "Z80 Microprocessor Programming & Interfacing". Book 1. Howard W. Sams & Co. Inc. Indiana, USA. 1980.

[10] Ogata,K. "Ingenieria de Control Moderna". Prentice Hall Hispanoamericana, 1985.

[11] Oppenheim, A.V. and Schaffer, R.W. "Discrete Time Signal Processing". Prentice Hall International, Englewood Cliffs N.J. 1989.

[12] "Z80 Microprocessor Family Databook". SGS-Thomson Microelectronics.