81

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

RECONOCIMIENTO DE COMANDOS VERBALES EN DSP's

T E S I S

QUE PARA OBTENER EL TÍTULO DE
INGENIERO EN COMPUTACIÓN

P R E S E N T A N :

OMAR NIETO CRISÓSTOMO
Y
VICTOR LÓPEZ MIRANDA

DIRECTOR DE TESIS Dr JOSÉ ABEL HERRERA CAMACHO

CIUDAD UNIVERSITARIA MEXICO D. F. OCUBRE 2002

A mis padres
Alvaro Nieto Mondragón y
Dominga Crisóstomo Blas
Que me han apoyado incondicionalmente
durante todo el transcurso de mi carrera y
que se han desvelado todas las noches para
cerciorarse que he llegado con bien

A mis hermanos
Carmen Martina Octavio
Marisol y **Yazmin**
Por lo que me han brindado en todo este
tiempo además por su apoyo y ayuda
incondicional que me han ofrecido

Agradecimientos

A DIOS por haberme dado la oportunidad de obtener una carrera

A mi *alma mater* UNAM por haberme formado como persona y haberme guiado a un sendero de conocimiento rico en su virtud

A la Facultad de Ingeniería que me ha ofrecido todas sus instalaciones durante mi formación

A mis profesores por haberme ofrendado su conocimiento experiencia y paciencia

Al Dr. Jose Abel Herrera por darme la oportunidad de realizar este trabajo, por todo su apoyo durante este

A M.I. Larr, Escobar y al Dr. Buhonil Psenicka que me impulsaron en todo este trayecto

A Daniel Vera por ayudarme en demasía en todos los problemas que se presentaron

A Victor Lopez que junto con él logre integrar un equipo formidable y lograr un excelente trabajo como tesis. Ademas por toda su paciencia y serenidad en los problemas que se presentaron

A mi padre **Juan José López Rodríguez** por enseñarme plenamente el significado de la palabra PERSEVERANCIA

A mi madre **Ángela Miranda Briseño** por su fe incondicional en mí, su abnegado apoyo y su perenne capacidad de sacrificio

A **Juan Roman** por su inegable estoicismo en los momentos difíciles y por su limpida visión del futuro

A mis hermanas **Rosa Angélica, Rosalba, Cecilia y María Guadalupe** porque en ellas encontré el hombro que me hizo fuerte en los momentos de flaqueza, porque saben dar todo sin esperar nada a cambio

A **Hector** porque su compañía fué un paliativo constante a mis nostalgias y su fortaleza un soporte en mis tristezas

Al resto de mis hermanos **Juan José, Alejandro, Arturo, Alfredo y José Guadalupe** porque con ellos encontré la contagiosa alegría por vivir

A **Dios** gracias por colocarme al lado de mis padres y hermanos y permitirme compartir con todos ellos tan gratos recuerdos

A **Lorena** por su amor sincero y su paciencia

A **Alicia** por tender el puente que me permitió ingresar en este templo del saber

AGRADECIMIENTOS

A **Dios** por permitirme alcanzar un sueño tan arduo y por acompañarme en los momentos de desesperación.

A mi *alma mater* la **UNAM** por sembrar en mi el germen de su espíritu.

A la **Facultad de Ingeniería** por acogerme durante todo este tiempo en sus aulas y permitirme beber de su fuente de conocimientos.

A todos mis **maestros de la carrera** por abrir mi juicio a ideas novedosas y por cultivar en mi el deseo de mantener un aprendizaje continuo.

A **Abel Herrera** por permitirme trabajar desarrollando mis inquietudes sin exigencias ni sobresaltos.

A **Daniel Vera** por el apoyo desinteresado ofrecido en los momentos críticos y por ampliar el panorama de mis ideas.

A **Omar Nieto** por la grata satisfacción de descubrir a su lado el trabajo en equipo. Por las incontables horas de sostenido esfuerzo, sacrificio. Por la vehemente defensa de sus ideas y por la fortuna de compartir juntos un destino común durante quince meses continuos con pequeños triunfos y grandes desatinos.

A todos mis amigos y compañeros de carrera de quienes aprendí y con quienes disfrute mi estancia en la Facultad.

A todas aquellas personas que contribuyeran con su granito de arena en mi formación y por quienes tengo la satisfacción de estar en este lugar.

A todos ustedes ***GRACIAS*** por permitirme caminar a su lado.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

El procesamiento digital de voz es un área de la ciencia y la ingeniería que se ha desarrollado durante los últimos treinta años, como consecuencia de los avances tecnológicos alcanzados en el área de la computación y de las enormes ventajas que presenta la manipulación digital de las señales.

Entre las personas, la voz representa la forma mas natural de intercambiar información. Debido a esto, el hombre trata de crear maquinas con interfaces cuya entrada sea la voz. El propósito del reconocimiento de voz es obtener una maquina capaz de "entender" un mensaje hablado y al mismo tiempo ejecutar la acción que se le ha indicado. Para superar éste propósito se requiere el apoyo de las técnicas del procesamiento digital de voz y de algunas ramas de la inteligencia artificial.

La meta de este trabajo es desarrollar una aplicación del reconocimiento de voz, capaz de funcionar adecuadamente en tiempo real, implantado en un procesador digital de señales (DSP). Además, esta tesis fue desarrollada de manera independiente, aunque ya se han entablado en la Facultad de Ingeniería varios esfuerzos tendientes a alcanzar la misma meta propuesta anteriormente. En consecuencia, no se trata de la continuación de ninguna investigación.

El objetivo logrado de esta tesis es, sin lugar a dudas, efectuar el reconocimiento de ocho palabras aisladas, para dos locutores distintos, en tiempo real, por medio de un DSP. El DSP elegido para cumplir este objetivo es el TMS320C6711 de *Texas Instruments*. Las técnicas del procesamiento seleccionadas para alcanzar el reconocimiento son las siguientes: análisis LPC (por el método de la autocorrelación), cuantización vectorial (con *k-means*), y distorsión de *Itakura* (como medida comparación).

El alcance de la tesis fue lograr reconocimiento de palabras aisladas, mediante el empleo de las técnicas mas simples, debido a las limitaciones en tiempo y recursos existentes en el área. Por consiguiente, se desarrollaron las técnicas mencionadas en el párrafo anterior, sin pretender ahondar en otras técnicas mas poderosas, pero que consumen mayores recursos. Se entrenaron dos tipos de patrones para reconocimiento: con 16 centroides, y con 32 centroides; 1 segmento. El numero de segmentos se mantuvo fijo en cuatro, debido a que su funcionamiento es el que ha logrado mejor comportamiento, durante pruebas realizadas en otros trabajos.

Los resultados obtenidos por el presente trabajo se dividieron de dos formas: por cada locutor y por numero de centroides. Con 16 centroides, el locutor Victor obtuvo una tasa de 198 aciertos sobre 200 intentos. Mientras tanto, el locutor Omar, alcanzo una tasa de 195 aciertos de 200 intentos. Por otro lado, para 32 centroides, el locutor Victor logro una tasa de 193 aciertos de 200 intentos y el locutor Omar obtuvo una tasa de 194 aciertos de 200 intentos.

A continuación se expone de forma breve el contenido de cada uno de los capítulos involucrados en este trabajo.

El capítulo 1, *Fundamentos del procesamiento de voz*, proporciona los principios teóricos en los que se basa el sistema de reconocimiento desarrollado. Contiene una descripción somera de las características de la producción y percepción de la voz. Incluye fundamentos propios del procesamiento de voz. Además, menciona la técnica de cuantización vectorial para el reconocimiento de patrones específicos de voz.

El capítulo 2, *Descripción general del DSP*, explica de manera breve las características de la arquitectura del DSP, sus periféricos y sus herramientas de desarrollo. Incluye un panorama general de sus rasgos predominantes, una explicación de la estructura del código ensamblador

la forma como se relaciona el lenguaje ensamblador con el lenguaje C. Además describe el entorno de desarrollo (*Code Composer Studio*) y ejemplifica la manera de ligar una aplicación con el intercambio de datos en tiempo real.

El capítulo 3, *análisis y diseño*, describe utilizando diagramas de bloques, la estructura general de las etapas involucradas en el proyecto. Durante este capítulo se plasman las ideas preconcebidas para definir la forma como se implementará el sistema de reconocimiento. A partir de estas ideas se definen los parámetros a utilizar y los alcances de cada módulo involucrado dentro de las diferentes etapas.

El capítulo 4, *Desarrollo*, contiene la explicación de la forma como funcionan los diversos algoritmos creados para el sistema. Incluye una descripción general del sistema utilizando diagramas de estados y una exposición de los procesos involucrados mediante diagramas de flujo.

El capítulo 5, *Resultados y conclusiones*, proporciona un panorama de los resultados alcanzados para los dos diferentes locutores y para las ocho distintas palabras. La forma como se exponen estos resultados es mediante tablas comparativas y gráficas. Además menciona algunas mejoras posibles y aplicaciones en donde sea factible utilizar el sistema. Al final del capítulo viene una recopilación de las conclusiones que se extrajeron de todo el proceso de desarrollo del sistema.

Por último, se proporciona un apéndice donde se mencionan las características del convertidor A/D que provee el *starter kit*.

CAPÍTULO 1

FUNDAMENTOS DEL PROCESAMIENTO DE VOZ

1.1 Producción y percepción de la voz

Aspectos básicos del sistema generador de voz

Los órganos productores de sonidos se pueden dividir en tres regiones: el tracto pulmonar o respiratorio formado por los pulmones y la traquea, la laringe que es una área situada superiormente a la traquea e inferiormente a la laringe y el tracto vocal formado básicamente por la laringe y las cavidades vocal y nasal. El primero es el generador de chorros de aire, posteriormente en la laringe se generan los sonidos y finalmente en el tracto vocal se modulan estos para producir los sonidos resultantes.

Después de la presión de aire generada por los pulmones, la siguiente función es realizada por la laringe y es llamada excitación. Esta adquiere las siguientes formas: fonación, susurreo, fricación, compresión y vibración [Herrera 1993]:

- *Fonación*: Se refiere a la oscilación de las cuerdas vocales por los movimientos de los cartílagos. La apertura y cierre de las cuerdas secciona el pulso de aire en pulsos cuasi periódicos llamados *pulsos glotales*, con una frecuencia fundamental llamada tono (pitch).

- *Susurreo*: Es generado en la laringe. Las cuerdas vocales están juntas, pero en lugar de sellar completamente la glotis existe una pequeña abertura triangular entre estos cartílagos. El aire que corre a través de esta abertura genera turbulencias que ocasionan ruido de banda ancha, el cual sirve como señal excitadora.

- *Fricación*: Es similar al susurreo en cuanto al aire turbulento que genera ruido de banda ancha, pero existe un lugar de articulación adicional en el tracto vocal. Dado que el lugar de articulación es cerca de los labios, solo una pequeña parte del tracto vocal esta entre las fuentes de excitación y el aire de salida. Esto significa que la modulación producida por el tracto vocal esta limitada en extensión y complejidad. La fricación se da con las letras f y s.

- *Compresión*: Cuando el tracto vocal esta prácticamente cerrado y una persona sigue exhalando, la presión aumenta y resulta un pequeño transitorio. La combinación de un silencio pequeño seguido por una ráfaga de ruido crea una excitación aperiódica, la onda de presión es una función escalón con un aspecto inverso a la frecuencia. Si el transitorio es abrupto y limpio, el sonido es una *oclusiva* o una *plosiva*. La compresión se da con las letras p y j.

- *Vibración*: Es cuasi periódica y puede ocurrir en muchos lugares del tracto vocal, por ejemplo, la vibrante r involucra la vibración de la lengua contra el paladar. Esta vibración puede ocurrir con o sin fonación y su efecto principal es la interrupción o una modulación semejante a la fonación, que sea rápida y repetitiva.

El término *tracto vocal* engloba a los órganos productores de voz situados arriba de las cuerdas vocales. Consiste de cinco elementos: faringe laringeal, faringe oral, faringe nasal, cavidad oral y cavidad nasal. En el adulto, el tracto vocal es de aproximadamente 17 cm de longitud. Dado que

la onda acústica pasa por él, su comportamiento espectral es modificado por sus resonancias las cuales dependen de las formas que adopte el tracto. Moviendo la lengua se puede modificar la estructura de la cavidad oral, de la faringe oral. Se puede desacoplar la cavidad nasal del sistema levantando el velo de manera que selle la cavidad.

La función del tracto vocal para la producción de voz es la de articular, articulación de la voz principalmente por la lengua, los labios y la mandíbula baja. También contienen los puntos principales desde los cuales los sonidos son radiados. Esta función es también la modulación ya que el sonido es modulado por el tracto vocal con el efecto de modificar la calidad del sonido e interponer sonidos adicionales e interrupciones a los sonidos.

Modelo del tracto vocal

El tracto vocal una vez que es modelado se manifiesta como un filtro variable en el tiempo cuyos parámetros cambian en función de la acción consiente que se realiza al pronunciar una palabra. El filtro variable en el tiempo tiene dos posibles señales de entrada que dependerán del tipo de señal, sonora o no sonora. Para las señales sonoras, la excitación sería un tren de impulsos de frecuencia controlada, mientras que para las señales no sonoras, la excitación sería un ruido aleatorio. La combinación de estas dos señales modela el funcionamiento de la glotis.

El espectro de frecuencias para la señal vocal se puede obtener a partir del producto del espectro de la excitación con la respuesta en frecuencia del filtro. El tracto vocal manifiesta un número muy grande de resonancias, sin embargo se consideran solo las tres o cuatro primeras mismas que toman el nombre de frecuencias formantes, cubren un rango de frecuencias que oscila entre los 100 y 3500 Hz. Esto se debe a que las resonancias de alta frecuencia son atenuadas por la característica frecuencial del tracto que tiende a actuar como un filtro paso bajas con una caída de aproximadamente -12 dB por octava. Este modelo es una simplificación del proceso del habla [Flores, 1996].

Los sonidos fricativos no se forman por el tracto con la misma extensión que lo hacen las señales sonoras, por lo que el modelo no es muy preciso para este tipo de señales. Además el modelo supone que las dos señales pueden separarse sin considerar que haya ninguna interacción entre ellas, lo cual no es cierto, ya que la vibración de las cuerdas vocales se ve afectada por las ondas de presión dentro del tracto. Sin embargo, estas consideraciones pueden ser ignoradas resultando un modelo lo suficientemente adecuado. Ver la figura 1.1.

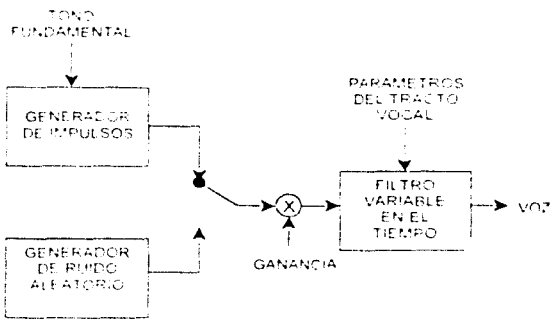

Figura 1.1 Modelo del tracto vocal como filtro variable en el tiempo.

Caracteristicas de la señal de voz

Los sonidos se pueden clasificar, de forma genérica, en sonoros y no sonoros o sordos. En los primeros se abren y cierran las cuerdas vocales, modificando el área de la traquea y produciendo un tren de impulsos casi periódicos. El periodo o frecuencia fundamental de este tren de impulsos se conoce con el nombre de pitch y su valor está comprendido entre los 50 y 400 Hz para los hombres y es superior en mujeres y niños. En los sonidos no sonoros, el aire fuye libremente hasta alcanzar el tracto vocal, al permanecer abiertas las cuerdas vocales. Finalmente, la variación voluntaria del tracto vocal, junto con el estado variante de las cuerdas produce la voz.

El tracto vocal actúa como una cavidad resonante para los sonidos sonoros, estando centradas las frecuencias de resonancia para la mayoría de la gente en 500 Hz y sus armónicas pares. Esta resonancia produce grandes picos en el espectro resultante, a los cuales se les llama formantes. Además la señal tiene la naturaleza de un filtro paso bajas y a partir de unos 4000 Hz comienza a predominar el ruido.

En cambio, el segmento de voz no sonoro presenta una estructura ruidosa tanto en el dominio del tiempo como en el de la frecuencia, no teniendose formantes. Además la energía de la señal es mucho menor que la de los sonidos sonoros [Lopez, 1999].

Representación de la voz en los dominios del tiempo y la frecuencia

La señal de voz tiene una variación de sus características estadísticas muy pequeña, cuando se examina para periodos de tiempo muy cortos (entre los 5 y 100 ms). A esta característica se le conoce como estacionariedad en el tiempo, dentro de esos periodos. Sin embargo, para periodos más grandes (del orden de los 200 ms o mayores), las características de la señal cambian, para reflejar los diferentes sonidos del habla [Rabiner, 1993].

Comúnmente se acepta la convención de una representación en tres estados, para la clasificación de producciones de voz, en el tiempo: 1) silencios, cuando no se produce o genera voz, 2) sonidos no sonoros o sordos y 3) sonidos sonoros. Debe quedar claro, sin embargo, que la clasificación anterior no genera segmentos con regiones bien definidas. Esto representa una dificultad cuando se quieren distinguir regiones dentro de una palabra. Ver la figura 1.2.

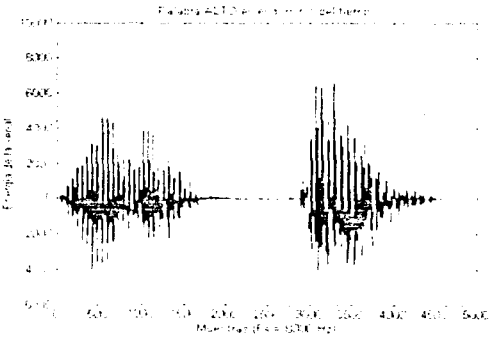

Figura 1.2 Representación de una señal de voz en el dominio del tiempo, con una frecuencia de muestreo de 8 KHz.

Otra alternativa para representar la señal de voz y caracterizar su información con una representación de la información asociada a la señal es utilizando la representación espectral. La forma más común de este tipo de representaciones son los espectrogramas.

Los espectrogramas representan en tres dimensiones las siguientes características: (1) intensidad de la voz, (2) diferentes bandas de frecuencias y (3) descripción del comportamiento de las frecuencias en el tiempo.

Esta representación de la variación de la señal en el tiempo y la frecuencia es la base para la parametrización del modelo de voz.

Sin embargo, el mayor problema que se presenta con este tipo de representaciones es la dificultad para establecer estimaciones confiables de las frecuencias formantes para sonidos sonoros de bajo nivel, así como la dificultad para encontrar las frecuencias formantes de los sonidos sordos y los silencios. Ver la figura 1.3.

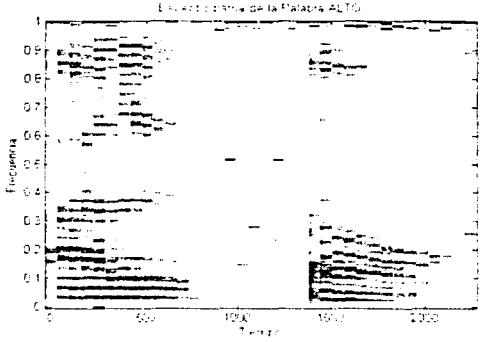

Figura 1.3 Espectrograma de la palabra ALTO. Se muestran las tres dimensiones: horizontal, tiempo; vertical, frecuencia; y colores, la intensidad de la voz (la gradación).

Percepción general de la voz

La percepción de la voz es el proceso cerebral mediante el cual se le da una interpretación a la voz. No se conoce con seguridad cual es el proceso que se sigue en la percepción de la voz. Existen varios trabajos de investigación donde se experimenta sobre la forma en que percibimos la voz [Parsons, 1987].

El proceso de percepción no es totalmente utilizado en un sistema de reconocimiento automático de voz, pero puede ser de gran ayuda para desarrollar algunas partes del sistema, fundamentándose en las funciones realizadas durante el proceso de percepción de voz por los seres humanos.

Parece ser que el cerebro hace una distinción fundamental entre sonidos de voz y sonidos que no lo son. Aparentemente, el cerebro procesa en forma diferente a los sonidos de voz. Es creíble que exista una predisposición innata en el sistema nervioso humano para decodificar entradas de voz [Parsons, 1987].

La importancia del contexto en el que se genera la voz influye en la facilidad y precisión de la percepción. Nosotros percibimos la voz por medio de la modulación interna del parlante. Duplicamos mentalmente la voz siguiendo y anticipándonos a la misma. Tal modelo requiere coherencia y continuidad de la voz. Esta teoría de la modulación interna es de gran influencia para desarrollar sistemas reconocedores de voz [Parsons, 1987].

La forma de procesar la información se basa principalmente en la habilidad del ser humano para reconocer distintas características de los sonidos recibidos, sin tener en cuenta las características físicas y acústicas de la voz, al momento de realizar el proceso de detección de voz.

Algunas de las teorías más simples que intentan explicar el proceso de percepción de voz son las siguientes [Parsons, 1987]:

1. La voz humana no es entendida solo por las características acústicas o análisis puro de señales, sino que involucra el conocimiento del lenguaje y el medio.

2. La voz entrante es procesada palabra por palabra, en el orden en que estas son recibidas. La identificación de cada palabra ayuda al reconocimiento de las palabras posteriores, limitando las posibilidades de las palabras subsecuentes.

3. En el reconocimiento de voz, el oyente pone más interés en detectar y reconocer la primera parte del sonido, para eliminar así las posibilidades del resto de la palabra y hacer más fácil el reconocimiento.

Saber con exactitud cuál es el proceso que realiza el cerebro para la percepción de la voz, es muy difícil. Solo se tienen varias teorías que sugieren algunas formas. En un sistema reconocedor se pueden tomar esas ideas y aplicarlas. Tal vez no sea la forma correcta como realiza el proceso de detección el cerebro, pero para aplicaciones prácticas son bastante útiles.

1.2 Fundamentos del procesamiento de voz

La señal de voz necesita cumplir ciertas características para facilitar el proceso de extracción de la información. En esta sección se describen los fundamentos que permiten cumplir estas características. Incluyen un filtro paso bajas, un filtro de preénfasis, tasa de cruces por cero, energía y magnitud promedio, detección de inicio y fin de la palabra, ventaneo y autocorrelación.

Es indispensable hacer notar que estos fundamentos no forman parte del método mismo de reconocimiento de voz, pero se deben utilizar como preámbulo para transformar la señal acústica, de forma que se facilite la obtención de sus características y se mejoren los resultados finales.

Filtro paso bajas

Los *filtros digitales IIR* se pueden obtener fácilmente empezando con un filtro analógico y después usando una correspondencia para transformar del plano s al plano z. Así el diseño de un filtro digital se reduce a diseñar un filtro analógico apropiado y después realizar la conversión de *H(s)* a *H(z)*, de tal manera que se preserven lo mejor posible las características deseadas del filtro analógico [Proakis, 1998].

Para el diseño e implementación de filtros digitales existen varios tipos de estructuras, por ejemplo: en forma directa, en forma de cascada, en forma paralela, en forma de retículo (de celda),

etc. De particular importancia son las estructuras en cascada, en paralelo y en lattice, ya que muestran robustez para la implementación con valores de longitud finita (presentan pocos errores de cuantización) [Proakis, 1996].

Considerando un sistema discreto, lineal e invariante en el tiempo, que se caracteriza por su función de transferencia, dada por

$$H(z) = \frac{\sum_{k=0}^{q} b_k z^{-k}}{1 + \sum_{k=1}^{p} a_k z^{-k}} \qquad \text{Ec. 1.1}$$

De esta representación, se obtienen los ceros y los polos de la función de transferencia, los cuales dependen de la elección de los parámetros del sistema $\{b_k\}$ y $\{a_k\}$ y además determinan las características de la respuesta en frecuencia del sistema. Sin pérdida de generalidad, se asume que $p \geq q$. El sistema se puede factorizar por medio de la ecuación 1.1 en sistemas de segundo orden en cascada, esto es

$$H(z) = \prod_{k=1}^{K} H_k(z) \qquad \text{Ec. 1.2}$$

Donde K es la parte entera de $(p + 1) / 2$, p representan el número de polos del filtro y $H_k(z)$ es un sistema de segundo orden que tiene la forma general

$$H_k(z) = \frac{b_{0k} + b_{1k} z^{-1} + b_{2k} z^{-2}}{1 + a_{1k} z^{-1} + a_{2k} z^{-2}} \qquad \text{Ec. 1.3}$$

Los coeficientes $\{b_k\}$ y $\{a_k\}$ en los subsistemas de segundo orden, son reales. Esto implica que para formar estos subsistemas de segundo orden se debe agrupar un par de polos complejos conjugados y un par de ceros complejos conjugados. Sin embargo, el emparejamiento de dos polos complejos conjugados, con un par de ceros complejos conjugados o ceros reales, se puede realizar de forma arbitraria. Además, cualquier par de ceros reales se puede agrupar para formar un factor cuadrático, y de la misma forma, cualquier par de polos reales se puede agrupar para formar un factor cuadrático. Consecuentemente, el factor cuadrático en el numerador (o denominador) de la ecuación 1.3, puede consistir de un par de raíces reales o de un par de raíces complejas conjugadas [Proakis, 1996].

Si $p > q$, algunos de los subsistemas de segundo orden tienen coeficientes del numerador igual a cero, es decir, $b_{2k} = 0$ o $b_{1k} = 0$ o ambos, $b_{1k} = b_{2k} = 0$ para algún k. Además, si p es impar, uno de los subsistemas, digamos $H_1(z)$, debe tener $a_{2k} = 0$, de tal forma que el subsistema sea de primer orden. Para preservar la modularidad en la implementación de $H(z)$, a menudo es preferible utilizar los subsistemas básicos de segundo orden en la estructura en cascada y tener algunos coeficientes igual a cero en alguno de los subsistemas [Proakis, 1996].

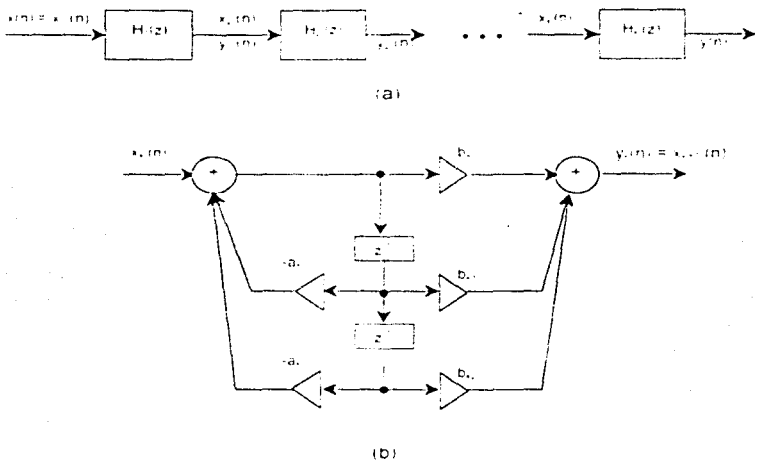

Figura 1.4 (a) Estructura en cascada de sistemas de un sistema de segundo orden (b) Implementación de cada sección de segundo orden

La forma general de la estructura en cascada se muestra en la figura 1.4 (a). Si utilizamos la estructura en forma directa II como se observa en la figura 1.4 (b) para cada uno de los sub sistemas el algoritmo computacional para realizar al sistema IIR con función de transferencia H(z) se describe mediante el siguiente conjunto de ecuaciones:

$$y_0(n) = x(n)$$

$$w_k(n) = -a_{1k} w_k(n-1) - a_{2k} w_k(n-2) + y_{k-1}(n) \quad k = 1, 2, \ldots, K$$

$$y_k(n) = b_{0k} w_k(n) + b_{1k} w_k(n-1) + b_{2k} w_k(n-2) \quad k = 1, 2, \ldots, K$$

$$y(n) = y_K(n)$$

Así este conjunto de ecuaciones proporciona una descripción completa de la estructura en cascada basada en la forma directa II [Proakis 1998].

Preénfasis

En el espectro de la voz existe una caída de -6 dB/octava conforme la frecuencia aumenta. Esto se debe a la combinación de una caída de -12 dB/octava ocasionada por la fuente de excitación de la voz y un incremento de +6 dB/octava ocasionado por la radiación de la boca. Esto significa que cada vez que la frecuencia aumenta al doble la amplitud de la señal se reduce en un factor de 1/2. Por lo que se desea compensar esta caída de -6 dB/octava con una manipulación de la

señal de voz que de un incremento de +6 dB/octava en el rango apropiado de manera que la medición del espectro tenga un rango dinámico similar a lo largo de todo su ancho de banda [Deller 1987]

A esto se le conoce como preenfasis. En un sistema de procesamiento digital de voz el preenfasis puede ser implementado de dos formas: ya sea por un filtro analógico paso altas de primer orden con una frecuencia de corte de 3 dB en algún punto entre los 100 Hz y 1 kHz (la posición exacta no es crítica), el cual precede al filtrado antialiasing y al convertidor A/D, o con un filtro digital paso altas que procese a la señal de voz digitalizada. Este filtro digital puede ser implementado con la siguiente ecuación en diferencias:

$$y[n] = x[n] - a\,x[n-1] \qquad \text{Ec. 1.4}$$

Donde cada muestra leída de la señal se almacena en x[n]. Además, x[n – 1] representa una muestra inmediatamente anterior en el tiempo. Por otro lado, y[n] es la salida del filtro de preenfasis en el tiempo n. Y finalmente ***a*** es una constante, para nuestro caso ***a*** = 0.95, de acuerdo a las recomendaciones de [Ohsens 1993]. Las figura 1.5 y 1.6 muestran el efecto que tiene el filtro de preenfasis sobre una señal de voz.

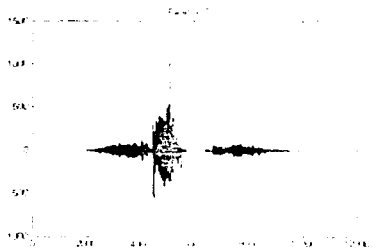

Figura 1.5 Se utiliza la palabra SIX sin filtro de preenfasis.

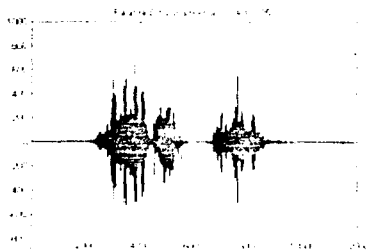

Figura 1.6 Se utiliza la palabra SIX para mostrar el efecto del filtro de preenfasis con a = 0.95.

⌟ Tasa de cruces por cero

En el estudio de señales discretas un cruce por cero ocurre si dos muestras sucesivas tienen signos distintos. La tasa de cruces por cero es una forma muy simple de medir el contenido frecuencial de una señal lo cual resulta cierto para señales de banda angosta [Gardida 1998]

Aunque las señales de voz son señales de banda ancha y la interpretación de la tasa de cruces por cero en promedio es mucho menos precisa que la energía y magnitud promedio se pueden estimar las propiedades espectrales de la señal de voz utilizando esta técnica. La tasa de cruces por cero en promedio se define por

$$Zn = \sum_{m} \left| sgn[x(m)] - sgn[x(m-1)] \right| w(n-m) \qquad \text{Ec 1.5}$$

donde

$$sgn[x(n)] = \begin{cases} 1 & x(n) \geq 0 \\ -1 & x(n) < 0 \end{cases} \qquad \text{Ec 1.6}$$

y

$$w(n) = \begin{cases} \frac{1}{2N} & 0 \leq n \leq N-1 \\ 0 & \text{otro} \end{cases} \qquad \text{Ec 1.7}$$

Expresiones que nos indican que las muestras se comparan de dos en dos se suman aquellas que tienen signos distintos y el resultado se promedia sobre N muestras consecutivas (donde este último promedio es opcional)

En el procesamiento de voz la tasa de cruces por cero se utiliza considerando que el modelo de voz sugiere que la energía de la voz sonora se concentra por debajo de los 3000 Hz mientras que la energía de la voz sorda se encuentra en su mayoría en altas frecuencias. Debido a que las altas frecuencias involucran una alta tasa de cruces por cero y las bajas frecuencias involucran una baja tasa de cruces por cero existe una íntima relación entre la tasa de cruces por cero y la distribución de la energía con la frecuencia [Gardida 1998]

De acuerdo a la anterior relación se puede hacer una razonable generalización cuando la tasa de cruces por cero es baja existe voz sonora y viceversa cuando la tasa de cruces por cero es alta existe voz sorda. Tengamos cuidado la aseveración anterior no es del todo correcta por lo que es necesario hacer otros tipos de consideraciones [Gardida 1998]

⌟ Energía y magnitud promedio

Al observar cualquier gráfica de señales de voz por ejemplo ver figura 1.5 se puede notar que su amplitud varía considerablemente con el tiempo. En general la amplitud de los sonidos sordos es mucho menor que la amplitud de los sonidos sonoros. La energía en tiempo corto de la señal de voz proporciona una representación conveniente que refleja estas variaciones. La energía de una señal discreta se define como

$$E_n = \sum_m x^2(m) \qquad \text{Ec 1 8}$$

Pero esta expresión nos resulta de poca utilidad debido a que la información que presenta es de poca utilidad para conocer las propiedades que dependen del tiempo de la señal. Por lo que es mejor utilizar la definición de la energía en tiempo corto

$$E_n = \sum_m x^2(m)h(n-m) \qquad \text{Ec 1 9}$$

donde

$$h(n) = w^2(n) \qquad \text{Ec 1 10}$$

La interpretación de la ecuación 1 9 es que la señal $x^2(m)$ se pasa a través de un filtro con respuesta al impulso $h(n)$. Pero existe el problema de seleccionar la ventana que mejor se acople a las necesidades específicas

En nuestro caso una aplicación de la función de energía es que nos permite distinguir entre los segmentos de voz y los segmentos de silencio

Una dificultad que se tiene al utilizar la función de energía en tiempo corto es que resulta muy sensible a valores grandes de la señal debido a que en su cálculo la señal se multiplica por sí misma, enfatizándose las diferencias de los valores que existen entre muestra , muestra. Para disminuir estas diferencias es mas recomendable utilizar la función de magnitud promedio en vez de la función de energía

$$M_n = \sum_m |x(m)|w(n-m) \qquad \text{Ec 1 11}$$

en la cual se suman los valores absolutos de la señal en vez de sumar los cuadrados

La representación generada por la Ec 1 11 resulta para fines prácticos de igual utilidad que la representación de la función de energía (Ec 1 9) aunque los rangos de la función varían en una raíz cuadrada y las diferencias entre voz sonora y voz sorda no son tan pronunciadas

Ventanas

En aplicaciones prácticas del procesamiento de señales es necesario trabajar con pequeñas partes de la señal a las que se denomina *tramas* a menos que la señal sea de corta duración. Esto es especialmente cierto cuando se utilizan técnicas de análisis convencional sobre señales con características dinámicas no estacionarias (como la señal de voz). En este caso es necesario elegir una parte de la señal que pueda asumirse razonablemente como estacionaria [Deller 1987]

Esto se renombra *ventana* (en el dominio del tiempo discreto) se denota por $x(n)$ es una secuencia real de tamaño finito utilizada para seleccionar la trama deseada de la señal original denotada por $x(m)$ por un proceso de multiplicación [Deller 1987]

Con la anterior consideración, asumimos que la señal es igual a cero fuera del intervalo de interés (ver Ec. 1.12)

$$x_w(n) = x(n)w(n) \qquad 0 \le n \le N-1 \qquad \text{Ec. 1.12}$$

Existen los siguientes tipos de ventanas: rectangular, Bartlett, triangular, Hamming, Hanning y Blackman, cada una de ellas con características particulares. Enseguida se muestran las ecuaciones que definen a cada una [Deller, 1987]

- *Rectangular*

$$w(n) = \begin{cases} 1, & n = 0,1,\ldots,N-1 \\ 0, & \text{e.o.c.} \end{cases} \qquad \text{Ec. 1.13}$$

- *Triangular o Bartlett.*

$$w(n) = \begin{cases} \dfrac{2}{N-1}n, & 0 \le n \le \dfrac{N-1}{2} \\ 2 - \dfrac{2}{N-1}n, & \dfrac{N-1}{2} \le n \le N-1 \\ 0, & \text{e.o.c.} \end{cases} \qquad \text{Ec. 1.14}$$

- *Hamming*

$$w(n) = \begin{cases} 0.54 - 0.46\cos\left(\dfrac{2\pi n}{N-1}\right), & 0 \le n \le N-1 \\ 0, & \text{e.o.c} \end{cases} \qquad \text{Ec. 1.15}$$

- *Hanning*

$$w(n) = \begin{cases} 0.5 - 0.5\cos\left(\dfrac{2\pi n}{N-1}\right), & 0 \le n \le N-1 \\ 0, & \text{e.o.c} \end{cases} \qquad \text{Ec. 1.16}$$

- *Blackman*

$$w(n) = \begin{cases} 0.42 - 0.5\cos\left(\frac{2\pi n}{N-1}\right) + 0.08\cos\left(\frac{4\pi n}{N-1}\right), & 0 \le n \le N-1 \\ 0, & \text{en otro caso} \end{cases} \qquad \text{Ec. 1.16}$$

La figura 1.7 muestra un ejemplo que compara cada una de estas ventanas

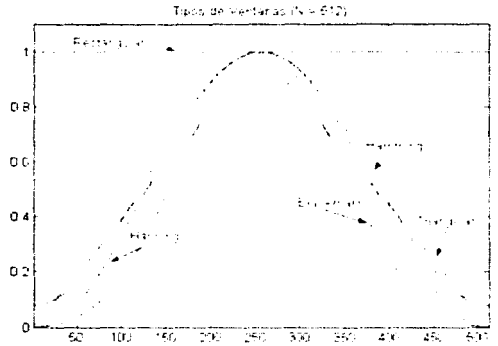

Figura 1.7 Tipos de ventanas con N = 512 muestras

Detección de inicio y fin de la palabra

El problema de localizar donde inicia y donde termina una palabra resulta importante en muchas áreas del procesamiento de voz y es de particular importancia en el reconocimiento de palabras aisladas. Tiene la finalidad de permitirnos trabajar solo con las muestras que en realidad sean únicamente representativas de la señal de voz eliminando el ruido que se presenta al inicio y al final de la grabación [Garrido, 1998].

El algoritmo generalmente utilizado para tal fin se conoce como *Rabiner-Sambur*, esta basado en la combinación de dos mediciones realizadas en el dominio del tiempo (magnitud promedio y cruces por cero).

Primeramente se definen tres tipos de umbrales para la detección del inicio y fin de las palabras aisladas de la siguiente forma:

- ITU es el umbral superior de energía y se obtiene con la siguiente ecuación

$$0.5\max(M(n)) \qquad \text{Ec. 1.17}$$

- ITL es el umbral inferior de energía y se obtiene con la ecuación

$$m_M + 2\sigma_M \qquad \text{Ec 1 18}$$

- ITZC es el umbral de cruces por cero y se obtiene con la siguiente ecuación

$$m_z + 2\sigma_z \qquad \text{Ec 1 19}$$

Enseguida se obtienen por cada trama de n muestras la magnitud promedio de la señal y su tasa de cruces por cero. Posteriormente se asume que las primeras 10 tramas no contienen voz, esto se hace con el fin de obtener una caracterización estadística del ruido de fondo. Utilizando esta estadística del ruido se calculan los umbrales que nos servirán para detectar el inicio y fin. Se define un primer umbral muy conservador ITU para obtener el intervalo en el cual la energía promedio de la señal siempre es excedida y se asume que el inicio y el final se encuentran fuera de este intervalo [Gardida 1998]

Enseguida se verifican las tramas desde el punto donde se rebasó por primera vez el umbral ITU hacia el inicio (o final) de la grabación hasta un punto N1 (o N2) donde la magnitud promedio queda por debajo de un umbral menor ITL y que es seleccionado tentativamente como el inicio (o final) de la palabra [Gardida 1998]

Resulta valido suponer que el inicio y fin no se encuentran dentro de estos puntos por lo que el siguiente paso es verificar hacia el inicio (o final) de la grabación desde N1 (o N2) la tasa de cruces por cero y compararla con el umbral ITZC. Esto se realiza para las 25 tramas que preceden a N1 (o siguen a N2). Y si la tasa de cruces por cero excede el umbral ITZC 3 o mas veces el inicio N1 es recorrido hasta el punto en donde el umbral fue excedido por primera vez. En caso contrario el inicio se escoge en el primer punto N1. Se realiza un procedimiento similar para determinar el final de la palabra. Un ejemplo típico de este método se muestra en la figura 1.8

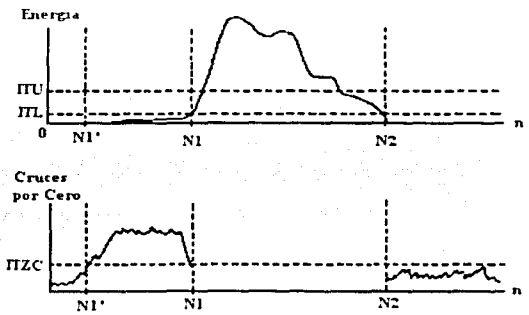

Figura 1.8 Graficas Típicas de la Magnitud Promedio y los Cruces por Cero

Función de autocorrelación

La función de autocorrelación de una señal determinística y discreta se define por

$$\phi(k) = \sum_{m} x(m)\,x(m+k) \qquad \text{Ec. 1.20}$$

La importancia que tiene la función de autocorrelación dentro de las señales de voz radica en el despliegue conveniente que hace de ciertas características de la señal, por ejemplo:

1. Si la señal es periódica con período P, entonces $\phi(k) = \phi(k + P)$. Esto significa que la función de autocorrelación resulta igualmente periódica y con un período P.
2. Es una función par: $\phi(k) = \phi(-k)$.
3. Tiene su valor máximo en $k = 0$, o sea $|\phi(k)| \le \phi(0)$ para toda k.
4. La cantidad $\phi(0)$ es igual a la energía para señales determinísticas o es igual a la potencia promedio si las señales son periódicas o aleatorias.

Con la ayuda de estas propiedades, podemos observar que para señales periódicas, la función de autocorrelación presenta máximos en los puntos $0, \pm P, \pm 2P, \ldots$. Esto es, si se ignora el punto de origen, el período de la señal puede estimarse al localizar el primer máximo de la función, lo que resulta aun más importante ya que permite estimar la periodicidad en segmentos de señales, inclusive en las señales de voz.

Al igual que en el caso de la energía, conviene obtener una representación en tiempo corto, definiéndose la función de autocorrelación en tiempo corto por medio de la ecuación 1.21:

$$R_n(k) = \sum_{m} x(m)w(n-m)\,x(m+k)w(n-k-m) \qquad \text{Ec. 1.21}$$

Reescribiendo la expresión de la siguiente forma:

$$R_n(k) = \sum_{m=n-N+1}^{n} x_n(m)\,x_n(m+k) \qquad m = 0,1,2,\ldots,p \qquad \text{Ec. 1.22}$$

Como la señal de voz se puede considerar *ergódica*, la Ec. 1.22 se emplea para la estimación de la autocorrelación en señales de voz en tiempo corto. Un proceso aleatorio es *ergódico* si los promedios temporales obtenidos para un solo evento son iguales a los promedios estadísticos de una colección de eventos [Rabiner, 1993].

1.3 Parametrización de la voz

La suposición básica con que se trabaja es que la voz se puede modelar con un sistema lineal e invariante en el tiempo (LIT), debido a que sus características estadísticas varían muy lentamente (para períodos muy cortos de tiempo, entre 5 y 100 ms). Lo que nos conduce a que la voz pueda ser analizada en segmentos de corta duración (tramas), y a que cada trama pueda ser representada como respuesta a una excitación de entrada para un sistema LIT. Las dos entradas que suponemos para generar la voz pueden ser un tren de impulsos casi periódicos (voz sonora) o un ruido aleatorio (voz sorda). Por lo tanto, el problema principal en el análisis de voz es estimar los parámetros del modelo de la voz y medir sus variaciones en el tiempo. A

continuación se presenta el tipo de parametrización con el que se modela la señal de voz [Gardida 1998]

❑ Análisis LPC

Una de las técnicas mas poderosas en el analisis de voz es el metodo de Analisis de Prediccion Lineal. El modelo de prediccion lineal es ampliamente utilizado para modelar el tracto vocal, ya que los modelos comunes del proceso de produccion de voz usualmente tratan por separado al tracto vocal y al aire que entra a este (excitación).

En el analisis LPC el modelo de produccion de voz del tracto vocal se representa por medio de un filtro digital lineal variante con el tiempo. Este filtro debe representar los efectos de la radiacion de los labios, la forma del pulso glotal y el acoplamiento de la cavidad nasal cada vez que se requiera. Ademas, para representar la voz, utiliza una baja tasa de transmisión o almacenamiento de bits. La importancia de esta técnica radica en la capacidad de proveer parámetros estimadores de la voz y la relativa velocidad de calculo.

La idea basica detras del modelo LPC es que dada una muestra de voz en un tiempo n, $s(n)$, esta puede ser aproximada por medio de una combinacion lineal de las p muestras de voz precedentes. Al minimizar la suma de las diferencias de los cuadrados (sobre un intervalo finito) entre las muestras de voz actuales y las predecidas linealmente, un conjunto único de coeficientes de prediccion puede ser determinado, esto es

$$s(n) \approx a_1 s(n-1) + a_2 s(n-2) + \ldots + a_p s(n-p). \qquad \text{Ec 1.23}$$

donde los coeficientes a_1, a_2, ... a_p se suponen constantes sobre la trama de voz analizada. La ecuacion anterior se convierte en igualdad al incluir un termino de excitación $Gu(n)$, quedando

$$s(n) = \sum_{i=1}^{p} a_i s(n-i) + Gu(n). \qquad \text{Ec 1.24}$$

donde $u(n)$ es la excitación normalizada y G es su ganancia. Al expresar esta ecuación en el dominio de z se obtiene la relación

$$S(z) = \sum_{i=1}^{p} a_i z^{-i} S(z) + GU(z). \qquad \text{Ec.1.25}$$

conduciéndonos a la función de transferencia

$$H(z) = \frac{S(z)}{GU(z)} = \frac{1}{1 - \sum_{i=1}^{p} a_i z^{-i}} = \frac{1}{A(z)} \qquad \text{Ec. 1.26}$$

La interpretación de esta última ecuación esta dada en la figura 1.9, donde se muestra la fuente de excitación normalizada, $u(n)$, siendo escalada por la ganancia G y actuando como entrada al sistema todo polos, $H(z) = 1/A(z)$, para producir la señal de voz estimada, $s(n)$.

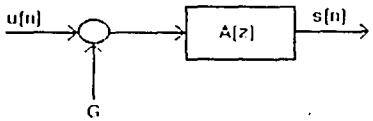

Figura 1.9 Sistema todo polos para producir la señal de voz

Además, la filosofía del modelo LPC se encuentra estrechamente relacionada con el conocimiento de que la fuente de excitación para la señal de voz es esencialmente un tren de pulsos casi periódicos (para la voz sonora) o una fuente de ruido aleatorio (para la voz sorda) que sirve de excitación a un sistema lineal variante con el tiempo. Precisamente el modelo LPC nos proporciona un método robusto, realizable y preciso, para estimar los parámetros que caracterizan a este sistema lineal e invariante con el tiempo [Rabiner, 1993].

Análisis de ecuaciones LPC

De acuerdo al modelo de la figura 1.7, la relación exacta entre $s(n)$ y $u(n)$ es

$$s(n) = \sum_{k=1}^{p} a_k s(n-k) + Gu(n) \qquad \text{Ec. 1.27}$$

Consideramos la combinación lineal de las muestras de voz pasadas como la estimación $\hat{s}(n)$ definida como

$$\hat{s}(n) = \sum_{k=1}^{p} a_k s(n-k) \qquad \text{Ec. 1.28}$$

Formamos el error de predicción, $e(n)$, de la siguiente forma

$$e(n) = s(n) - \hat{s}(n) = s(n) - \sum_{k=1}^{p} a_k s(n-k) \qquad \text{Ec. 1.29}$$

con función de transferencia del error

$$A(z) = \frac{E(z)}{S(z)} = 1 - \sum_{k=1}^{p} a_k z^{-k} \qquad \text{Ec. 1.30}$$

Ahora el problema básico en el análisis de predicción lineal es determinar el conjunto de coeficientes de predicción $\{a_k\}$ directamente de la señal de voz de modo que las propiedades espectrales de la voz se mantengan constantes. Y ya que las características espectrales de la voz varían con el tiempo, los coeficientes de predicción en un tiempo dado n deben estimarse en un segmento corto de la señal de voz alrededor de este tiempo. Entonces, la idea fundamental es encontrar un conjunto de coeficientes de predicción que minimicen el error de predicción cuadrático medio (sobre un segmento corto).

Para empezar, se define el segmento de voz y el segmento de error en un tiempo n como

$$s_n(m) = s(n+m)$$
$$e_n(m) = e(n+m) \qquad \text{Ec. 1.31}$$

Se busca minimizar la señal del error cuadrático medio en el tiempo n

$$E_n = \sum_m e_n^2(m) \qquad \text{Ec. 1.32}$$

Al usar la definición de $E_n(m)$ en términos de $s(m)$, se puede escribir como

$$E_n(m) = \sum_m \left[s_n(m) - \sum_{k=1}^{p} a_k s_n(m-k) \right]^2 \qquad \text{Ec. 1.33}$$

Para resolver esta ecuación, se deriva parcialmente E_n con respecto a cada a_K y el resultado se iguala a cero

$$\frac{\partial E_n}{\partial a_k} = 0, \quad k = 1,2,\ldots,p \qquad \text{Ec. 1.34}$$

Resultando

$$\sum_m s_n(m-i)s_n(m) = \sum_{k=1}^{p} \hat{a}_k \sum_m s_n(m-i)s_n(m-k) \qquad \text{Ec. 1.35}$$

Observando que los términos de la forma $\sum_m s_n(m-i)s_n(m-k)$ representan las *covarianzas* de los segmentos $s_n(m)$, se pueden escribir como

$$\phi(i,k) = \sum_m s_n(m-i)s_n(m-k) \qquad \text{Ec. 1.36}$$

Por lo que la ecuación 1.36 se puede escribir en forma más compacta como

$$\phi_n(i,0) = \sum_{k=1}^{p} \hat{a}_k \, \phi_n(i,k) \qquad \text{Ec. 1.37}$$

Esta última ecuación describe un conjunto de p ecuaciones con p incognitas. Y para obtener los coeficientes de predicción óptimos, se tiene que calcular $\phi_n(i,k)$ para $1 \leq i \leq p$ y $0 \leq k \leq p$ y resolver el conjunto de p ecuaciones simultaneas.

Existen varios métodos para calcular los coeficientes de predicción, como *covarianzas, autocorrelación, enrejado, filtro inverso, estimación espectral, máxima probabilidad* y *el de producto interno*. En el reconocimiento de voz (y particularmente en este trabajo), se utiliza el método de autocorrelación, debido a su eficiencia computacional y a su estabilidad inherente. Este método siempre produce un filtro de predicción cuyos ceros se encuentran dentro del círculo unitario en el plano Z [Rabiner, 1993].

❑ Método de la autocorrelación

Una forma de determinar los límites de las sumatorias de las ecuaciones anteriores, es asumir que el segmento $s_n(m)$ es igual a cero fuera del intervalo $0 \leq m \leq N-1$. Esto puede escribirse de la forma

$$s_n(m) = s(m+n)w(m) \qquad \text{Ec. 1.38}$$

en donde, $w(m)$ es una ventana de longitud finita (usualmente ventana de Hamming), que es igual a cero fuera del intervalo $0 \leq m \leq N-1$.

El efecto de la suposición anterior para las expresiones que contienen a E_n, puede verse al considerar la ecuación 1.38. Claramente, si $s_n(m)$ es diferente de cero solo para $0 \leq m \leq N-1$, entonces su error de predicción correspondiente, $e_n(m)$, para un predictor de orden p, será diferente de cero en el intervalo $0 \leq m \leq N-1+p$. Entonces, en este caso, E_n se expresa aproximadamente como

$$E_n = \sum_{m=0}^{N-1+p} e_n^2(m) \qquad \text{Ec. 1.39}$$

y $\phi_n(i,k)$ se puede expresar

$$\phi_n(i,k) = \sum_{m=0}^{N-1+p} s_n(m-i)s_n(m-k), \quad \begin{matrix} 1 \leq i \leq p \\ 0 \leq k \leq p \end{matrix} \qquad \text{Ec. 1.40}$$

Como esta última ecuación está sólo en función de $(i-k)$ (en lugar de ser función de dos variables i y k), la función de covarianza, $\phi_n(i,k)$, se reduce a una simple función de autocorrelación

$$\phi_n(i,k) = r_n(i-k) = \sum_{m=0}^{N-1-(i-k)} s_n(m)s_n(m+i-k), \quad \begin{matrix} 1 \leq i \leq p \\ 0 \leq k \leq p \end{matrix} \qquad \text{Ec. 1.41}$$

Ademas como la funcion de autocorrelacion es simétrica $r_n(-k) = r_n(k)$ las ecuaciones LPC pueden expresarse como

$$\sum_{k=1}^{p} r_n(|i-k|)\hat{a}_k = r_n(i) \qquad 1 \le i \le p \qquad \text{Ec 1.42}$$

o en forma matricial

$$\begin{bmatrix} r_n(0) & r_n(1) & r_n(2) & \cdots & r_n(p-1) \\ r_n(1) & r_n(0) & r_n(1) & \cdots & r_n(p-2) \\ r_n(2) & r_n(1) & r_n(0) & \cdots & r_n(p-3) \\ \vdots & \vdots & \vdots & & \vdots \\ r_n(p-1) & r_n(p-2) & r_n(p-3) & \cdots & r_n(0) \end{bmatrix} \begin{bmatrix} \hat{a}_1 \\ \hat{a}_2 \\ \hat{a}_3 \\ \vdots \\ \hat{a}_p \end{bmatrix} = \begin{bmatrix} r_n(1) \\ r_n(2) \\ r_n(3) \\ \vdots \\ r_n(p) \end{bmatrix} \qquad \text{Ec 1.43}$$

Esta matriz de orden $p \times p$ con los valores de autocorrelacion es una matriz *Toeplitz* (simétrica con los elementos de la diagonal principal iguales) que puede resolverse eficientemente con el uso de varios procedimientos numéricos. Uno de ellos es el algoritmo de *Levinson Durbin* [Rabiner 1993]

Algoritmo de Levinson-Durbin

Inicialización

$$E^{(0)} = r(0)$$

Para $1 \le i \le p$

$$k_i = \frac{r(i) - \sum_{j=1}^{i-1} a_j^{(i-1)} r(i-j)}{E^{(i-1)}}$$

$$a_i^{(i)} = k_i$$

para $1 \le j \le i-1$

$$a_j^{(i)} = a_j^{(i-1)} - k_i a_{i-j}^{(i-1)}$$

$$E^{(i)} = (1 - k_i^2) E^{(i-1)}$$

La solución final esta dada por

$$a_m = coeficientes\ LPC = a_m^{(p)}$$

1.4 Distancias y medidas de distorsión

Un componente clave en la mayoría de los algoritmos de comparación de patrones es formular una medida de distorsión entre dos vectores característicos. Esta medida de distorsión puede ser manejada con rigor matemático si los patrones son visualizados en un espacio vectorial.

Suponer que se tienen dos vectores característicos x e y definidos en un espacio vectorial. Se define una *métrica* o *función de distancia* d en el espacio vectorial como una función de valor real sobre el producto cartesiano, que cumpla las siguientes condiciones:

1. $0 \le d(x, y) < \infty$ para $x, y \in$ y $d(x, y) = 0$ si y solo si $x = y$
2. $d(x, y) = d(y, x)$ para $x, y \in$
3. $d(x, y) \le d(x, z) + d(z, y)$ para $x, y, z \in$

Además, una función de distancia se denomina invariante si:

4. $d(x + z, y + z) = d(x, y)$

Las primeras tres propiedades comúnmente son conocidas como *positividad* (no negatividad), *simetría* y *desigualdad del triángulo* respectivamente. Una métrica que contenga estas propiedades permite un alto grado de manejo matemático. Si una medida de distancia d satisface solo la propiedad de positividad, se le denomina *medida de distorsión*, particularmente cuando los vectores son representaciones del espectro de la señal.

Para el procesamiento de voz es importante considerar que la definición (o elección) de la medida de distancia es significativamente subjetiva. Una medida matemática de la distancia para ser utilizada en el procesamiento de voz debe tener una alta correlación entre su valor numérico y su distancia subjetiva aproximada, para evaluar una señal real de voz. Para el reconocimiento de voz, la consistencia *psicofísica* de diferentes matices que se le pueden imprimir a una misma palabra o frase que se desea medir con la distancia, obliga a que se encuentre una medida matemática ajustada por necesidad a las características lingüísticas percibidas. Estos requisitos tan subjetivos no pueden ser satisfechos con medidas de distancia que proporcionen manejo matemático. Un ejemplo es la tradicional medida de Error Cuadrático $d(x, y) = (x - y)^2$.

Dado que existe una enorme dificultad al querer cumplir simultáneamente con ambos objetivos (subjetividad y manejo matemático), afrontar algún compromiso es inevitable. Por consiguiente y dado que se necesita manipular matemáticamente las propiedades de esa medida de distancia, se requiere probar que éstas propiedades subjetivas son lo suficientemente buenas como para lograr el reconocimiento de voz.

Por otra parte, se hablará de "medidas de distorsión" en vez de "métricas", debido a que se relajan las condiciones de simetría y desigualdad del triángulo. No se debe utilizar el término *distancia* en sentido estricto, acorde a la definición de arriba. Además, se debe mantener la costumbre de la literatura de voz, donde el término *distorsión* es análogo a las medidas de distorsión [Rabiner, 1993].

El hecho de medir las diferencias entre dos patrones de voz en términos de promedios o distorsión espectral acumulada, parece ser un camino muy razonable de comparación de patrones en términos del manejo matemático y su eficiencia computacional. Debido a que muchos estudios psicoacústicos sobre las diferencias percibidas en el sonido pueden ser interpretadas en términos de diferencias en sus características espectrales, la medida de distorsión espectral puede ser a la vez un argumento razonablemente matemático y subjetivo.

Existen varios tipos de medidas de distorsión cada una con sus características especiales Entre ellas tenemos *Distancia Euclidiana Cuadrática, Distorsión del Error Cuadrático Medio, Distorsión del Error Cuadrático Ponderado, Distancia de Itakura* etc

- *Distancia Euclidiana Cuadrática* La medida mas conveniente y ampliamente usada para calcular distancias es el Error Cuadrático o Distancia Euclidiana Cuadrática entre dos vectores definida como

$$d(X_1, X_2) = \|X_1 - X_2\|^2 = (X_1 - X_2)^T (X_1 - X_2) \qquad \text{Ec 1.44}$$

- *Distorsión del Error Cuadrático Medio* La distorsión del Error Cuadrático Medio (MSE) es otra de las medidas mas utilizadas y se define como

$$d(X_1, X_2) = \frac{1}{N}(X_1 - X_2)^T (X_1 - X_2) = \frac{1}{N}\sum_{i=1}^{N}(X_{1i} - X_{2i})^2 \qquad \text{Ec 1.45}$$

en la cual la distorsión está definida por cada dimensión La popularidad del MSE se basa en su simplicidad y seguimiento matemático

- *Distorsión del Error Cuadrático Medio Ponderado* Otra medida de distorsión es el Error Cuadrático Medio Ponderado En el MSE la medida asume que las distorsiones contribuyen cuantizando los diferentes parametros $\{X_{ij}\}$ de igual forma Y de manera general se pueden introducir pesos diferentes con el fin de aportar ciertas contribuciones a la distorsión dependiendo del parametro El MSE ponderado general se define como

$$d(X_1, X_2) = (X_1 - X_2)^T W (X_1 - X_2) \qquad \text{Ec 1.46}$$

Donde W es una matriz de ponderación definida simétrica y positiva y los vectores X_1 y X_2 son tratados como vectores columna

Cada una de las medidas de distorsión mencionadas anteriormente resultan simétricas en sus argumentos X_1 y X_2 y pueden ser aplicadas a las características derivadas del análisis de producción lineal de la voz el uso de algunas presenta ciertas desventajas tal es el caso de la distancia euclidiana que aunque resulte fácil de calcular no todas sus características tienen el mismo significado perceptible

Por lo anterior en ciertos casos resulta conveniente y efectivo escoger una matriz de ponderación $W(X_1)$ que dependa explícitamente del vector X_1 para así obtener una medida de distorsión perceptiblemente motivada En este caso la distorsión

$$d(X_1, X_2) = (X_1 - X_2)^T W(X_1)(X_1 - X_2) \qquad \text{Ec 1.47}$$

es asimétrica

- *Distancia de Itakura*. En muchos casos del procesamiento de voz es necesario tener otra medida de la distancia que existe entre dos vectores LPC. La distancia Euclidiana no es apropiada para medir los parámetros de dos LPC's individuales en vectores que estén relacionados. Esto es debido a que los vectores LPC generan un peso de la matriz de autocorrelación correspondiente a cada LPC.

  La medida de distancia más comúnmente utilizada para este propósito es la propuesta por *Itakura*. Esta distancia se deriva utilizando una interpretación de la ML del rango de predicción en el error de la energía. Fue obtenida originalmente utilizando la máxima probabilidad existente entre argumentos similares. La distancia de *Itakura* es probablemente la medida de distorsión más empleada para encontrar la similitud entre dos vectores LPC [Deller 1987].

  Esta distancia se define de la siguiente forma: sean R_x y R_y las matrices de autocorrelación de las señales de voz de entrada y de comparación respectivamente. Sean asimismo a_x, R_x la energía de salida del filtro inverso tomado como referencia con la entrada y a_y, R_y la energía mínima posible de salida del filtro LPC con respecto a la entrada de la voz. Entonces tenemos que la distancia de *Itakura* se obtiene mediante la Ec. 1.48:

$$d(X_x, X_y) = \log(a_y R_y a_y^T / a_x R_x a_x^T) \qquad \text{Ec. 1.48}$$

  en donde tenemos que la autocorrelación para una señal dada se consigue mediante la doble sumatoria mostrada en la ecuación 1.49:

$$R_x^i = \sum_{n=0}^{p} a_n \sum_{m=0}^{p} x_m x_{m+|i-n|} a_m \qquad \text{Ec. 1.49}$$

1.5 Cuantización vectorial

La cuantización vectorial (VQ) resulta una generalización de la cuantización escalar, pero ahora aplicada a todo un vector. El cambio de una a varias dimensiones trae consigo un gran número de ideas, conceptos, técnicas y aplicaciones nuevas. Mientras la cuantización escalar se utiliza principalmente en la conversión analógico/digital, la cuantización vectorial se enfrenta a las sofisticadas técnicas del procesamiento digital de señales. En la mayoría de los casos las características más relevantes de las señales de entrada tiene representación digital, por eso la cuantización vectorial se utiliza usualmente en la compresión de datos. Sin embargo, existen ciertos paralelismos entre ambas cuantizaciones, lo que permite la utilización de varios métodos en la cuantización vectorial, como una generalización [Gersho 1997].

Un vector se puede utilizar para describir prácticamente cualquier tipo de patrón, como puede ser un segmento de una señal de voz o de una imagen, simplemente al formar un vector con las muestras de la señal de voz o de la imagen. La cuantización vectorial puede aplicarse al reconocimiento de patrones, ya que un patrón de entrada es comparado y aproximado a alguno de los patrones de referencia almacenados. El reconocimiento permite encontrar el patrón de referencia que más se acopla al patrón de entrada. Por lo tanto, la cuantización vectorial es más que una generalización de la cuantización escalar. En fechas recientes, se ha convertido en la principal herramienta del reconocimiento de voz, además de que se sigue utilizando en la compresión de señales de voz e imágenes [Gersho 1997].

Definición

Partimos de un conjunto de vectores que pertenecen a un espacio *K*-dimensional asumiendo que ***x*** es un vector perteneciente a ese conjunto, cuyos componentes [**x** $_i$ $1 \le i \le K$] son variables aleatorias reales y de amplitud continua. Un cuantizador vectorial *Q* de dimensión *K* y tamaño *N* es una transformación de un vector **x**, del espacio euclidiano de dimensión R^k, en un conjunto finito *C* que contiene *N* salidas o puntos de reproducción llamados *code vectors* (vectores de código)

$$Q : R^k \rightarrow C \quad C = \{y_1, y_2, \ldots, y_N\} \quad y_i \in R^k \quad \forall \; i \in I = \{1, 2, \ldots, N\} \qquad \text{Ec 1.50}$$

El conjunto *C* es llamado *code book* (libro de códigos) y tiene un tamaño *N*, esto significa que tiene *N* distintos elementos, cada uno de ellos dentro del espacio R^k

Asociado a cada cuantizador vectorial de *N* puntos, existe una *partición* de R^k en *N* regiones o *celdas*, R_i para $i = 1$. La *i*-esima celda esta definida por

$$R_i = \{\mathbf{x} \in R^k : Q(\mathbf{x}) = y_i\} \qquad \text{Ec 1.51}$$

algunas veces llamada *imagen inversa* o *pre-imagen* de y_i dentro del mapeo *Q* y denotada de forma más consistente por $R_i = Q^{-1}(y_i)$

De la definición de celdas tenemos que

$$\bigcup_{i=1}^{N} R_i = R^k \quad \text{y} \quad R_i \cap R_j = \varnothing \quad \text{para} \quad i \neq j \qquad \text{Ec 1.52}$$

o sea las celdas forman una partición de R^k

Una propiedad importante de un conjunto en R^k es ser convexo. Para el caso de dos o tres dimensiones, un conjunto se dice convexo si, dados dos puntos cualesquiera dentro de él, existe una línea recta que los une y pertenece al mismo conjunto. Esta idea se extrapola a R^k de la siguiente manera: un conjunto $\mathbf{S} \subset R^k$ es convexo si *a* y *b* $\in$ **S** implica que $\alpha a + (1 - \alpha)b \in S$ $\forall$ $0 \le \alpha \le 1$

Un cuantizador vectorial se denomina *regular* si

(a) Cada celda R_i es un conjunto convexo

(b) Para cada i $\mathbf{y}_i \in R_i$

La tarea de codificación de un cuantizador vectorial es examinar cada vector de entrada **x** e identificar a qué celda *k*-dimensional del espacio R^k pertenece. El codificador vectorial simplemente identifica el índice *i* de la región y el decodificador vectorial genera el vector del código $\mathbf{y}_i$ que representa a esta región [Gersho 1997]

El conjunto **y** es conocido como *diccionario de reconstrucción* o simplemente *diccionario* donde *N* es el tamaño del diccionario y $\{\mathbf{y}_i\}$ es el conjunto de vectores del código. Los vectores $\mathbf{y}_i$ son conocidos también en la literatura de reconocimiento de patrones como los *patrones de*

referencia o *plantillas*. El tamaño *N* del diccionario también se conoce como *número de niveles*, término proveniente de la cuantización escalar. De esta forma, se puede hablar de un diccionario de *N* niveles. Al proceso de creación del diccionario también se le conoce como *entrenamiento* o *población del diccionario*.

El modelo de operación de este codificador, se define de forma similar al caso escalar, la *función de selección* $S_i(x)$, como *indicador* o *función miembro* $1_R(x)$ para la celda R_i de la partición, esto es:

$$S_i(x) = \begin{cases} 1 & \text{si } x \in R_i \\ 0 & \text{en otro caso} \end{cases} \qquad \text{Ec. 1.53}$$

La operación de un cuantizador vectorial puede ser representada como

$$Q(x) = \sum_{i=1}^{N} y_i S_i(x) \qquad \text{Ec. 1.54}$$

Una descomposición estructural (como la mostrada anteriormente), es particularmente evaluable para encontrar un algoritmo efectivo, durante la implementación de la cuantización vectorial [Gersho, 1997].

❑ Agrupamiento

El agrupamiento es la forma en que se realiza la cuantización vectorial, consiste en lo siguiente: a partir de un conjunto de *N* muestras $x = \{X_1, X_2, X_3, \ldots, X_N\}$ se intentan separar en *K* subconjuntos disjuntos $x_1, x_2, x_3, \ldots, x_K$. En donde cada subconjunto representa a un grupo G_i, y en el cual las muestras pertenecientes tienen una mayor similitud entre sí, en comparación a las muestras de cualquier otro grupo.

Existen varios algoritmos de agrupamiento, entre los que tenemos: *simple, distancia máxima, K-Medias, ISODATA y LBG*, estos dos últimos son variantes del agrupamiento *K-Medias*, pero con mayor complejidad.

❑ Definición del algoritmo de K-medias

Se describe a continuación el algoritmo de agrupamiento *K-Medias*. Su criterio de función es:

$$J_e = \sum_{j=1}^{K} \sum_{x \in x_j} \| x - z_j \|^2 \qquad \text{Ec. 1.55}$$

donde:

K = número de grupos
$\mathbf{z}_j$ = centro del grupo (centroide) para el grupo j
x_j = subconjunto de muestras asignadas al grupo j

Algoritmo de K-medias:

1) Escoger K centroides iniciales $\mathbf{z}_1(1), \mathbf{z}_2(1), \ldots, \mathbf{z}_K(1)$

2) En la iteración l, asignar las muestras a los grupos
Asignar

$$\mathbf{x} \text{ a } \chi_j(l) \text{ si } \|\mathbf{x} - \mathbf{z}_j(l)\| \leq \|\mathbf{x} - \mathbf{z}_i(l)\| \quad j = 1,2, \ldots, K, \quad j \neq i$$

3) Calcular los nuevos centros de grupo

$$\mathbf{z}_i(l+1) = \frac{1}{N_i} \sum_{\mathbf{x} \in \chi_i} \mathbf{x} \qquad i = 1,2, \ldots, K$$

donde N es el número de muestras asignadas a $\chi_i(l)$

4) Si $\mathbf{z}_i(l+1) = \mathbf{z}_i(l)$ para $i = 1, 2, \ldots, K$, el algoritmo ha convergido y debe terminarse. En caso contrario, regresar al paso 2.

Una característica de este algoritmo es que los centroides o cuantizadores obtenidos no son los óptimos globales, es decir, se obtienen cuantizadores óptimos locales. Estos dependen de varios factores, como son: asignación inicial de centroides (principalmente), orden de la toma de muestras, propiedades geométricas de los datos, tipo de distorsión empleada, etc. Una forma de poder alcanzar los óptimos globales, es probar con una gran variedad de centroides iniciales y seleccionar los cuantizadores finales que tengan la menor distorsión con respecto a los vectores a los cuales representan. Solución poco factible porque existe un gran número de combinaciones para los cuantizadores iniciales. Otra forma es elegir los centroides iniciales de forma aleatoria para buscar una distribución homogénea [Deller, 1987].

Hasta aquí se han descrito los fundamentos teóricos más importantes que se usan en este trabajo. El siguiente capítulo describe, de forma general, la herramienta de desarrollo utilizada para el reconocimiento de palabras aisladas (DSP TMS320C6711 de *Texas Instruments*).

CAPÍTULO 2

DESCRIPCIÓN GENERAL DEL DSP

2.1 Características y opciones del TMS320C62x/C67x

Las familias de dispositivos TMS320C62x / C67x ejecutan un máximo de 8 instrucciones de 32 bits por ciclo. El CPU contiene 32 registros de proposito general de 32 bits y 8 unidades funcionales. Estos dispositivos tienen un conjunto completo de herramientas de desarrollo y optimizacion que incluyen un compilador C eficiente, un optimizador de ensamble para simplificar la planificación y programacion del lenguaje ensamblador, y un depurador con interface grafica basada en *Windows* para visualizar las caracteristicas de ejecución en el codigo fuente.

Ademas, contiene una tarjeta de emulacion de hardware compatible con la interface del emulador TI XDS510. Estas herramientas cumplen con los estandares 1149.1-1990, revision de acceso a puerto y arquitectura de verificacion de limites de la IEEE.

Las caracteristicas de los dispositivos C62x / C67x incluyen:

- Un CPU avanzado VLIW (*very long instruction word*) con 8 unidades funcionales, que incluyen 2 multiplicadores y 6 ALU's (unidades logico aritmeticas)
  - Ejecuta un maximo de 8 instrucciones por ciclo, 10 veces mas que los DSP's tipicos
  - Permite rapido tiempo de desarrollo en diseños con codigo RISC (reduced instruction set computer) altamente efectivos
- Empaquetado de instruccion
- Obtiene el tamaño del codigo equivalente a las 8 instrucciones ejecutadas serialmente o en paralelo
- Reduce el tamaño del codigo, el consumo de energia y el *fetch* del programa
- Ejecucion condicional de todas las instrucciones
  - Reduce los saltos costosos
  - Incrementa el paralelismo para mantener un alto desempeño
- Ejecuta el codigo programado en unidades funcionales independientes
- Proporciona soporte eficiente de memoria para una variedad de aplicaciones de 8, 16 y 32 bits de datos
- Maneja operaciones aritmeticas de 40 bits, adicionando precisión extra a *vocoders* y otras aplicaciones computacionalmente intensivas
- Proporciona soporte de normalizacion y saturacion en operaciones aritmeticas claves
- Soporta operaciones comunes, halladas en aplicaciones de control y manipulación de datos como: manipulacion de campos, extraccion de instruccion, activacion, desactivación y conteo de bits

Ademas, el C67x tiene las siguientes caracteristicas:

- Un maximo de 1336 MIPS (millones de instrucciones por segundo), a 167 MHz

- Un máximo de 1 G FLOPS (operaciones en punto flotante por segundo) a 167 MHz, para operaciones de precisión simple
- Un máximo de 250 M FLOPS a 167 MHz para operaciones de doble precisión
- Un máximo de 688 M FLOPS a 167 MHz para operaciones de multiplicación y acumulación
- Soporte de hardware para operaciones de punto flotante de simple y doble precisión (con formato IEEE)
- Multiplicación entera de 32 x 32 bits con resultado de 32 o 64 bits

Los dispositivos C62x / C67x tienen la siguiente variedad de opciones en memoria y periféricos

- Amplia memoria RAM para ejecución rápida de algoritmos
- Soporta interfaces para memoria externa de 32 bits (SDRAM, SBSRAM, SRAM y otras memorias asíncronas), para aumentar el intervalo de memoria externa y maximizar el desempeño del sistema
- Acceso a la memoria y periféricos de los dispositivos C62x / C67x a través del puerto *host*
- Controlador del multicanal DMA
- Puerto (s) serie multicanal
- *Timer* (s) de 32 bits

2.2 Arquitectura de los dispositivos TMS320C62x/c67x

Los dispositivos C6211, C6711, C6701, C6201 y C6202 operan a 150, 150, 167, 200 y 250 MHz respectivamente. Todos estos DSP's ejecutan un máximo de 8 instrucciones por ciclo. El DSP C6211 es de punto fijo mientras que el C6711 es de punto flotante.

Los procesadores C62x/C67x consisten de tres partes: el CPU, los periféricos y la memoria. El CPU ocho unidades funcionales operan en paralelo (seis ALU's y dos multiplicadores), con dos conjuntos similares de cuatro unidades funcionales básicas. Las unidades se comunican usando una trayectoria cruzada entre dos clasificaciones de registros, cada una de los cuales contienen 16 registros de 32bits. La figura 2.1 muestra el diagrama de bloque de los dispositivos TMS320C62x/C67x.

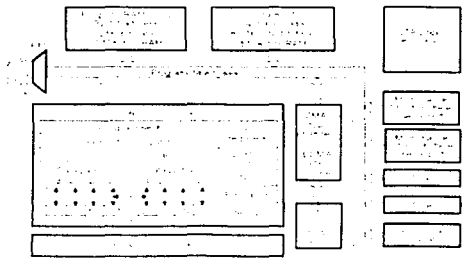

Figura 2.1 Diagrama de bloques del TMS320C62x/C67x

Unidad de procesamiento central (CPU)

El CPU contiene

- Unidad *fetch* de programa
- Unidad de despacho de instrucción
- Unidad de decodificación de instrucción
- 32 registros de 32bits
- Dos trayectorias de datos (*paths*) cada uno con cuatro unidades funcionales
- Registros de control
- Lógica de control
- Lógica de interrupción, emulación y prueba

El CPU tiene dos trayectorias de datos (A y B), cada trayectoria tiene cuatro unidades funcionales (L, S, M y D) y un archivo que contiene 16 registros de 32 bits (*register file*). Las unidades funcionales ejecutan operaciones de lógica, corrimiento, multiplicación y direccionamiento de datos. Todas las instrucciones aceptan operaciones de carga y almacenamiento sobre los registros. Las dos unidades de direccionamiento de datos (D1 y D2) son exclusivamente responsables de toda la transferencia de datos entre los archivos de registros y la memoria.

Trayectorias de datos del CPU

Las trayectorias de datos del CPU consisten de: dos archivos de registros de propósito general (A y B), ocho unidades funcionales (L1, L2, S1, S2, M1, M2, D1 y D2), dos trayectorias de lectura de memoria (LD1 y LD2), dos trayectorias de almacenamiento en memoria (ST1 y ST2), dos trayectorias cruzadas entre los archivos de registros (1X y 2X) y dos trayectorias de direccionamiento de datos (DA1 y DA2). La figura 2.2 muestra la trayectoria de los datos del CPU (CP).

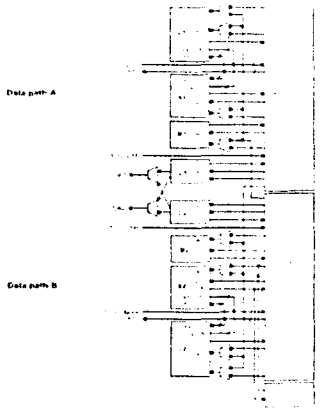

Figura 2.2 Trayectoria de datos del TMS320C67x

❖ *Archivos de registros de propósito general (register files)*

Hay dos archivos de registros de propósito general (A y B) en las trayectorias de datos del 'C62x/'C67x. Cada uno de esos archivos contiene 16 registros de 32 bits (A0-A15) para el archivo A y (B0-B15) para el archivo B. Los registros de propósito general pueden ser usados para manejar datos o punteros de direccionamiento de éstos. Los registros A1, A2, B0, B1 y B2 pueden ser empleados como registros de condición. Los registros A4-A7 y B4-B7 pueden ser utilizados para el direccionamiento circular.

Los archivos de registros de propósito general soportan datos de 32 y 40 bits de punto fijo. Los datos de 32 bits pueden estar contenidos en cualquier registro de propósito general. Los datos de 40 bits están contenidos en dos registros: los 32 bits menos significativos del dato (LSB) son colocados en un registro par y los restantes ocho bits mas significativos (MSB) son colocados en los ocho bits menos significativos del registro próximo superior (que es siempre un registro impar). El 'C67x también usa ese par de registros para colocar valores de punto flotante, de doble precisión, de 64 bits.

❖ *Unidades funcionales*

Las ocho unidades funcionales en las trayectorias de datos del C62x/C67x pueden ser divididas en dos grupos de cuatro: cada unidad funcional en una trayectoria de datos es casi idéntica a la unidad correspondiente en la otra trayectoria de datos (p. ej. L1 es muy similar a L2). Las unidades funcionales son descritas en la tabla 2.1.

Tabla 2.1 Unidades funcionales y las operaciones que realizan

Unidad Funcional	Operaciones en punto fijo	Operaciones en punto flotante
Unidad L (L1, L2)	Operaciones aritméticas y comparación de 32 y 40 bits Cuenta de 0's o 1's mas a la izquierda para 32 bits Normalización de 32 y 40 bits Operaciones lógicas de 32 bits	Operaciones aritméticas Operaciones de conversión: DP → SP, INT → DP, INT → SP
Unidad S (S1, S2)	Operaciones aritméticas de 32 bits Corrimientos de 32 y 40 bits. Operaciones *campos de bits* en 32 bits Operaciones lógicas de 32 bits Saltos Generación de constantes Transferencia de registros de hacia registros (solamente S2)	Comparación recíproca Operaciones de raíz cuadrada Operaciones de valor absoluto Operaciones de conversión SP a DP
Unidad M (M1, M2)	Operaciones de multiplicación de 16x16 bits	Operaciones de multiplicación de 32x32 bits Operaciones de multiplicación de punto flotante
Unidad D (D1, D2)	Sumas, restas y cálculos de direccionamiento circular de 32 bits Carga y almacenamiento con offset constante de 5 bits Carga y almacenamiento con offset constante de 15 bits (solo D2)	lectura de palabras dobles con offset constante de 5 bits

La mayoría de las trayectorias en el CPU soportan operaciones de 32 bits y algunas soportan operaciones largas (40 bits). Cada unidad funcional tiene su propio puerto de escritura de 32 bits en un archivo de registros de propósito general. Todas las unidades terminan en 1 (p. Ej. L1) cuando se refiere al archivo de registros A y en 2 cuando se refiere al archivo de registros B. Cada unidad funcional tiene dos puertos de lectura de 32 bits para cada operando *src1* y *src2*. Cuatro unidades (L1, L2, S1 y S2) tienen un puerto extra de 8 bits para ejecutar

operaciones de 40 bits. Debido a que cada unidad tiene su propio puerto de escritura de 32 bits, las ocho unidades pueden ser usadas en paralelo en cada ciclo.

❖ *Archivos de registros de control de TMS320C62x/C67x*

Una unidad (.S2) puede leer *de* y escribir *hacia* los registros de control. Mostrados en la figura 2.2. La tabla 2.2 menciona y describe los registros de control contenidos en el archivo de registros de control. Cada registro es accesado con la instrucción MVC.

Tabla 2.2 Registros de control

Abreviatura	Nombre	Descripción
AMR	Registro de modo de direccionamiento	Especifica si se utiliza direccionamiento lineal o circular para cada uno de los ocho registros, también contiene el tamaño para el direccionamiento circular
CSR	Registro de control de estado	Contiene el bit de interrupción global, los bits de control del caché y otros bits de control de estado misceláneos
IFR	Registro de bandera de interrupción	Despliega el estado de las interrupciones
ISR	Registros para activar interrupción	Permite activar interrupciones manualmente
ICR	Registro para limpiar interrupción	Permite limpiar interrupciones pendientes manualmente
IER	Registro para habilitar interrupción	Permite habilitar / deshabilitar interrupciones individuales
NRP	Puntero de retorno de interrupción no mascarable	Contiene la dirección de retorno para el regreso de una interrupción no mascarable
PCE1	Contador de programa fase E1	Contiene la dirección del paquete fetch (contiene el paquete de ejecución del pipeline) en la etapa E1

El C67x posee tres registros de configuración adicionales para soportar operaciones de punto flotante (vea la tabla 2.2.3). Los registros especifican los modos de redondeo de punto flotante, para las unidades M y L. También contienen campos de bit para advertir si *src1* y *src2* son NaN (no es un número) o son números desnormalizados. Además si resulta *overflow* o *underflow*, es inexacto, infinito o inválido. Existen campos que advierten si una división por cero fue ejecutada o si una comparación fue ejecutada con un NaN.

Tabla 2.3 Registros de control adicionales para el TMS320C67x

Abreviatura	Nombre	Descripción
FADCR	Registro de configuración para sumador de punto flotante	Especifica modo underflow, modo de redondeo, NaN y otras excepciones para la unidad L
FAUCR	Registro de configuración auxiliar de punto flotante	Especifica modo underflow, modo de redondeo, NaN y otras excepciones para la unidad S
FMCR	Registro de configuración para multiplicador de punto flotante	Especifica modo underflow, modo de redondeo, NaN y otras excepciones para la unidad M

❖ *Trayectorias entre archivos de registros (Register File Cross Paths)*

Cada unidad funcional lee directamente *de* y escribe directamente *hacia* el archivo de registros dentro de su propia trayectoria de datos. Esto es, las unidades L1, S1, D1 y M1, escriben en el archivo de registros A y las unidades L2, S2, D2 y M2, escriben en el archivo de registros B. Los archivos de registros son conectados a las unidades funcionales del archivo de registros opuesto a través de las trayectorias cruzadas 1X y 2X. Esas trayectorias cruzadas permiten a las unidades funcionales de una trayectoria de datos acceder a operandos de 32 bits del lado opuesto. La trayectoria cruzada 1X permite a las unidades funcionales de la trayectoria de datos A leer su operando fuente del archivo de registros B. La trayectoria cruzada 2X permite a las unidades funcionales de la trayectoria de datos B leer su operando fuente del archivo de registros A.

❖ *Trayectorias de Memoria, Cargas y Almacenamiento*

Hay dos trayectorias de 32 bits para leer los datos de memoria entre los registros de almacenamiento. LD1 para el archivo de registros A y LD2 para el archivo de registros B. El C67x también tiene una segunda trayectoria de carga de 32 bits para ambos archivos de registros (A y B). Esta segunda trayectoria permite que la instrucción LDDW lea dos registros de 32 bits simultaneamente en los lados A y B. Existen además dos trayectorias de 32 bits ST1 y ST2 para almacenar valores de los registros a la memoria para cada archivo de registros. Las trayectorias de lectura largas L y S son compartidas con las trayectorias de almacenamiento.

❖ *Trayectorias de direccionamiento de datos*

Las trayectorias de direccionamiento de datos (DA1 y DA2) mostrados en la figura 2.2.2 colocados fuera de las unidades D permiten generar direcciones de datos de un archivo de registros. Con esto se sostienen cargas y almacenamientos en memoria desde el otro archivo de registros. Sin embargo, las cargas y almacenamientos ejecutados en paralelo deben cargar a y del mismo archivo de registro. Aunque también existe la alternativa de que ambos usen una trayectoria cruzada al registro opuesto.

Mapeo entre instrucciones y unidades funcionales

La tabla 2.4 muestra el mapeo existente entre las instrucciones y las unidades funcionales para instrucciones de punto fijo del TMS320C62x/C67x.

Tabla 2.4 Mapeo de las instrucciones de punto fijo y las unidades funcionales

Unidad .L	Unidad .M	Unidad .S		Unidad .D	
ABS	MPY	ADD	SET	ADD	STB (15-bit offset) ‡
ADD	MPYU	ADDK	SHL	ADDAB	STH (15-bit offset) ‡
ADDU	MPYUS	ADD2	SHR	ADDAH	STW (15-bit offset) ‡
AND	MPYSU	AND	SHRU	ADDAW	SUB
CMPEQ	MPYH	B disp	SHRL	LDB	SUBAB
CMPGT	MPYHU	B IRP †	SUB	LDBU	SUBAH
CMPGTU	MPYHUS	B NRP †	SUBU	LDH	SUBAW
CMPLT	MPYHSU	B reg	SUB2	LDHU	ZERO
CMPLTU	MPYHL	CLR	XOR	LDW	
LMBD	MPYHLU	EXT	ZERO	LDB (15-bit offset) ‡	
MV	MPYHULS	EXTU		LDBU (15 bit offset) ‡	
NEG	MPYHSLU	MV		LDH (15-bit offset) ‡	

Tabla 2.4 (Continuación) Mapeo de las instrucciones de punto fijo y las unidades funcionales

Unidad .L	Unidad .M	Unidad .S	Unidad .D
NORM	MPYLH	MVC †	LDHU (15-bit offset) ‡
NOT	MPYLHU	MVK	LDW (15-bit offset) ‡
OR	MPYLUHS	MVKH	MV
SADD	MPYLSHU	MVKLH	STB
SAT	SMPY	NEG	STH
SSUB	SMPYHL	NOT	STW
SUB	SMPYLH	OR	
SUBU	SMPYH		
SUBC			
XOR			
ZERO			

* Solo S2
‡ Solo D2

La tabla 2.5 muestra el mapeo entre las instrucciones y las unidades funcionales para las instrucciones de punto flotante del TMS320C62x/C67x

Tabla 2.5 Mapeo de instrucciones de punto flotante y unidades funcionales

Unidad .L	Unidad .M	Unidad .S	Unidad .D
ADDDP	MPYDP	ABSDP	ADDAD
ADDSP	MPYI	ABSSP	LDDW
DPINT	MPYID	CMPEQDP	
DPSP	MPYSP	CMPEQSP	
INTDP		CMPGTDP	
INTSP	CMPGTSP		
RCPSP		CMPLTDP	
RSQRSP	CMPLTSP		
SPDP		RCPDP	
SPINT		RCPSP	
SPTRUNC		RSQRDP	
SUBDP		RSQRSP	
SUBSP			

Modos de direccionamiento

Los modos de direccionamiento son lineales por *default* para los C62x y C67x, aunque también existe el modo de direccionamiento circular. El modo de direccionamiento se especifica con el registro modo de direccionamiento (AMR).

Con todos los registros se puede efectuar el direccionamiento lineal. Solo en ocho de ellos se puede realizar direccionamiento circular, del A4 al A7 (que son usados por la unidad D1) y del B4 al B7 (que son utilizados por la unidad D2). Ninguna otra unidad puede efectuar direccionamiento circular. Las instrucciones *LDB/LDH/LDW, STB/STH/STW, ADDAB/ADDAH, ADDAW y SUBAB/SUBAH/SUBAW*, se apoyan en el registro AMR para determinar que tipo de calculo del direccionamiento es realizado por esos registros.

Los CPU's 'C62x/'C67x tienen arquitectura de carga/almacenamiento lo que significa que la única forma de accesar datos en memoria es con la instrucción de carga o almacenamiento. La tabla 2.6 muestra la sintaxis de un direccionamiento indirecto para una localización en memoria.

Tabla 2.6 Generación de direccionamiento indirecto para Load /Store

Tipo de direccionamiento	Ninguna modificación del registro de dirección	Preincremento o Predecremento del registro de dirección	Postincremento o Postdecremento del registro de dirección
Registro indirecto	*R	*++R *--R	*R++ *R--
Registro relativo	*+R[ucst5] *-R[ucst5]	*++R[ucst5] *--R[ucst5]	*R++[ucst5] *R--[ucst5]
Base + índice	*+R[offsetR] *-R[offsetR]	*++R[offsetR] *--R[offsetR]	*R++[offsetR] *R--[offsetR]

Interrupciones

Los CPUs 'C62x/'C67x manejan 14 interrupciones de hardware. Estas son: reset, interrupción no mascarable (NMI) e interrupciones de la 4 a la 15. Estas interrupciones corresponden a las señales RESET, NMI e INT4-INT15 respectivamente, sobre los límites del CPU. En los mismos dispositivos 'C62x/'C67x estas señales pueden estar ligadas directamente a lo pines del dispositivo, conectando periféricos al chip, o pueden estar desactivadas permanentemente, cuando están ligadas e inactivas en el chip. Generalmente RESET y NMI son conectadas directamente a los pines del dispositivo. Las características del servicio de interrupción incluyen:

- El pin IACK del CPU es usado para confirmar la recepción de una petición de interrupción.
- Los pines INUM0—INUM3 indican el vector de interrupción que esta siendo utilizado.
- Los vectores de interrupción son reubicables.
- Los vectores de interrupción consisten de un paquete *fetch*. Con los paquetes se proporciona un rápido servicio.

Para obtener mas información sobre la arquitectura del TMS320C62x/C67x revisar el manual *"TMS320C62x/C67x CPU and Instruction Set Reference Guide"* de *Texas Instruments*.

2.3 Periféricos

Los periféricos disponibles en los dispositivos TMS320C6000 se muestran en la tabla 2.7.

Tabla 2.7 Periféricos de los dispositivos TMS320C6000

Periféricos	C6201	C6202	C6211	C6701	C6711
Controlador de Acceso Directo a Memoria (EDMA)	Y	Y	N	Y	N
Controlador de Acceso Directo a Memoria Mejorado (EDMA)	N	N	Y	N	Y
Interface al Puerto Host (HPI)	Y	N	Y	Y	Y
Bus de Expansión	N	Y	N	N	N
Interface de Memoria Externa (EMIF)	Y	Y	Y	Y	Y
Configuración del Boot	Y	Y	Y	Y	Y
Puertos Seriales de Multicanal (McBSPs)	2	3	2	2	2
Selector de Interrupción	Y	Y	Y	Y	Y
Timers de 32-bits	2	2	2	2	2
Lógica de Energía Baja	Y	Y	Y	Y	Y

Los periféricos que son accesibles al usuario se configuran con un conjunto de registros de control mapeados en memoria. El controlador del bus de periféricos realiza el arbitraje para permitir el acceso a los periféricos. La lógica de configuración del *host* está conectada por señales externas y la lógica de baja energía es accesible directamente por el CPU.

La figura 2.3 muestra un diagrama de bloques con los periféricos de los dispositivos TMS320C6211/C6711.

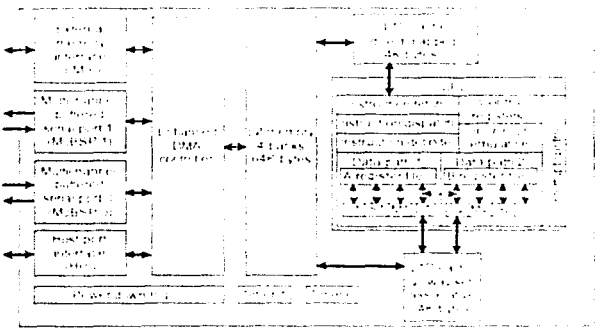

Figura 2.3 Diagrama de bloques para los dispositivos TMS320C6211/C6711

- **Controlador DMA**. El controlador DMA transfiere datos entre intervalos de direccionamiento en el mapa de memoria sin intervención del CPU. El controlador DMA tiene cuatro canales programables y cinco canales auxiliares.

- **Controlador EDMA**. El controlador EDMA mejora las mismas funciones del controlador DMA. El EDMA tiene dieciséis canales programables así como un espacio de RAM para soportar múltiples configuraciones de futuras transferencias.

- **HPI**. El HPI es un puerto paralelo por medio del cual un procesador *host* puede accesar directamente al espacio de memoria del CPU. El dispositivo *host* tiene facilidad de acceso debido a que es el maestro de la interface. El host y el CPU pueden intercambiar información a través de la memoria interna o externa. En suma, el *host* tiene acceso directo a los periféricos de memoria mapeada.

- **Bus de expansión**. El bus de expansión es un reemplazo para el HPI así como una expansión del EMIF. La expansión proporciona dos áreas distintas de funcionalidad (puerto *host* y puertos de Entrada - Salida) que pueden coexistir en un sistema. El puerto *host* del bus de expansión puede operar en modo asíncrono (esclavo) de forma similar al HPI o en modo síncrono (maestro - esclavo). Estos modos permiten la interface de dispositivos para una variedad de protocolos del bus del *host*. Los FIFOs síncronos y los dispositivos periféricos de Entrada - Salida asíncronos pueden conectarse al bus de expansión.

- **EMIF**. El EMIF soporta una interfaz de baja adherencia (*glueless*) para varios dispositivos externos. Incluye:

  - SRAM de ráfaga síncrona (SBSRAM)

- DRAM síncrona (SDRAM)
- Dispositivos asíncronos incluyendo SRAM ROM y FIFOs
- Dispositivo externo de memoria compartida

- **Configuración del *boot*** Los dispositivos TMS320C62x C67x proporcionan una variedad de configuraciones del *boot* que determinan las acciones de inicialización que ejecuta el DSP, después del *reset* del dispositivo Estas incluyen cargas de código en un espacio externo de ROM sobre el EMIF y cargas de código a través del HPI bus de expansión de un *host* externo

- **McBSP** El puerto serial multicanal con *buffer* (McBSP) esta basado en las interfaces estándar del puerto serie encontradas en los dispositivos con plataformas TMS320C2000 y C5000 Resumiendo, el puerto puede almacenar muestras seriales en un *buffer* de memoria automáticamente con la ayuda del controlador DMA/EDMA También tiene capacidad de multicanal compatible con los estándares de conexión de redes T1 E1 SCSA y MVIP Proporciona

  - Comunicación *full-duplex*
  - Registros de datos de doble *buffer* para flujo continuo de datos
  - Tramado independiente y temporización para dispositivos y transmisión
  - Interface directa a *codecs* estándar chips de interface analógica (AICs) y otros dispositivos A/D y D/A conectados serialmente

Tiene las siguientes capacidades

- Interface directa a

  Tramas T1/E1
  Dispositivos conforme a ST-BUS™
  Dispositivos conforme a IOM-2
  Dispositivos conforme a AC97
  Dispositivos conforme a IIS
  Dispositivos SPI™

- Transmisión y recepción multicanal de 128 canales
- Un selector del ancho del tamaño del dato que incluye 8 12 16 20 24 y 32 bits
- Ley-μ y Ley-A de compansión
- Transferencia inicial de 8 bits con LSB (bit menos significativo) o MSB (bit más significativo)
- Polaridad programable para ambas tramas de sincronización y relojes de datos
- Reloj interno altamente programable y generación de trama

- **TIMER** Los dispositivos C6000 tienen dos timer de propósito general que son usados para

  - Eventos del *timer*
  - Eventos del contador
  - Generador de pulsos
  - Interrupción del CPU
  - Enviar eventos de sincronización al controlador DMA/EDMA

- **Selector de interrupción** El conjunto de periféricos del C6000, genera de 14 a 16 fuentes de interrupción El CPU tiene 12 interrupciones disponibles El selector de interrupción permite elegir entre las 12 interrupciones la que necesita su sistema El selector de interrupción también permite cambiar la polaridad de entrada para la interrupción externa

- **Lógica de bajo consumo de energía** La lógica de bajo consumo de energía permite reducir el reloj para disminuir el consumo de energía. La mayoría de la potencia de operación de la lógica CMOS es disipada durante la conmutación del circuito de un estado lógico a otro. Para prevenir algo o toda la lógica del chip de conmutación se pueden realizar ahorros significativos de energía sin perder datos ni contexto operacional.

Para obtener más información sobre los periféricos del TMS320C6711 revisar el manual TMS320C6000 *Peripherals Reference Guide* de *Texas Instruments*.

2.4 Code Composer Studio (CCS)

El *Code Composer Studio* mejora y acelera el proceso de desarrollo para aplicaciones del procesamiento de señales incrustadas *(embeddedo)* en beneficio de los programadores que crean y prueban en tiempo real. El CCS proporciona herramientas para la configuración, construcción, depuración, mensajes (*tracing*) y análisis del programa.

El *Code Composer Studio* incluye los siguientes componentes:

- Herramientas de generación de código del TMS320C6000
- Entorno de Desarrollo Integrado (IDE) del *Code Composer Studio*
- DSP/BIOS *plug-ins* y API s
- RTDX *plug-in*, interface *host* y API s

Estos componentes, trabajando en conjunto, se muestran en la figura 2.4

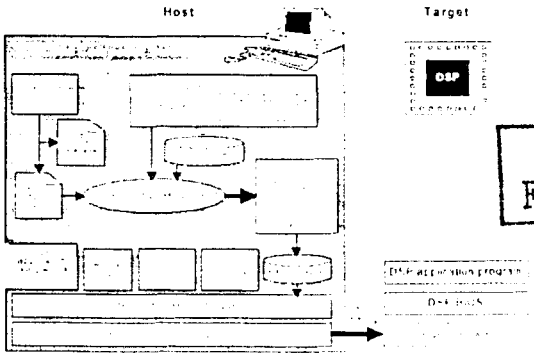

Figura 2.4 Diagrama de bloques del entorno de desarrollo del CCS

Herramientas de desarrollo para la generación del código

Los TMS320C6000 soportan un conjunto de herramientas para el desarrollo del software, que incluyen compilador C/C++ optimizado, ensamblador optimizado, ensamblador, ligador y las herramientas asociadas a ellos.

Ademas, el TMS320C6000 soporta las siguientes herramientas, para el desarrollo del lenguaje ensamblador

- Ensamblador (*Assembler*)
- Archivador (*Archiver*)
- Ligador (*Linker*)
- Listado Absoluto (*Absolute lister*)
- Listado de referencias cruzadas (*Cross-reference lister*)
- Utileria de conversion hexadecimal (*Hex conversion utility*)

La figura 2 5 muestra el flujo para desarrollo de software, con el TMS320C6000 La parte sombreada, representa la ruta mas común del desarrollo las demas partes son opcionales Estas otras partes, representan funciones periféricas que enlazan el proceso de desarrollo

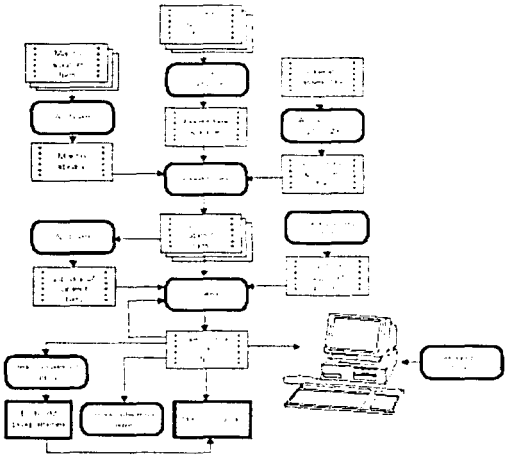

Figura 2 5 Flujo para el desarrollo de programas con el TMS320C6000

- El **optimizador de ensamble** (*assembly optimizer*), permite escribir código en lenguaje ensamblador lineal o secuencial sin importar la estructura del pipeline o la asignación de registros Asigna registros y usa optimización de ciclos para convertir el codigo de lenguaje ensamblador lineal a lenguaje ensamblador en paralelo

- **Compilador C/C++.** Acepta codigo fuente en lenguaje C/C++ y produce codigo fuente en lenguaje ensamblador para el TMS320C6000

  Para invocar al *shell* del compilador

  cl6x [options] [filenames] **[–z** [linker options] [object files]]

- **El ensamblador** (*assembler*) Traduce los archivos de lenguaje ensamblador fuente en lenguaje de maquina

  Para llamar ensamblador utilizar la siguiente linea de comandos

  asm6x [input file [object file [listing file]]] [options]

- El **ligador** (*linker*) Combina los archivos objeto en un solo archivo ejecutable para el DSP También acepta archivos de librerias y modulos de salidas creados por una ejecución previa del mismo

  La sintaxis general para invocar al ligador es la siguiente

  lnk6x [options] filename 1 filename n

- El **archivador** (*archiver*) Permite juntar un grupo de archivos dentro de un solo archivo, llamado libreria Por ejemplo se puede juntar distintas macros dentro de una libreria de macros

  Llamar al archivador de la siguiente forma

  ar6x [–] command [options] libname [filename 1 filename n]

  Tambien se puede usar la utileria de **construcción de librerias** (*library-build utility*), para construir su propia libreria de soporte en tiempo real

- El **listado absoluto** (*absolute lister*) es una herramienta para depuración de programas. Aceptan como entrada archivos objeto ligados y crea archivos con extension abs que son ensamblados para producir una lista que muestra las direcciones absolutas del codigo objeto

  La sintaxis para activar el listado absoluto es

  abs6x [–options] input file

- La utileria de **conversión hexadecimal** (*hex conversion utility*) Convierte un archivo objeto de tipo COFF (*common object file format*) en un archivo cuyo formato objeto se puede seleccionar entre los siguientes *Ti-Tagged* ASCII hexadecimal, *Intel Motorola-S* o *Tektronix* El archivo convertido puede ser transmitido a una memoria EPROM

  La invocacion de la utileria de conversion hexadecimal es de la siguiente forma

  hex6x [options] filename

- El **listado de referencias cruzadas** (*cross-reference lister*) Usa archivos objeto para producir un listado de referencias cruzadas mostrando simbolos definiciones y sus referencias en el archivo fuente ligado

  Activar el listado de referencias cruzadas de la siguiente forma

  xref6x [options] [input filename [output filename]]

Estructura del código ensamblador

Un programa en lenguaje ensamblador debe ser un archivo de texto en código ASCII. Cualquier línea del código ensamblador puede incluir cualquiera de los siguientes elementos:

- Etiqueta
- Barras paralelas
- Condiciones
- Instrucciones
- Unidad Funcional
- Operandos
- Comentarios

- **Etiquetas** (*label*). Una etiqueta identifica una línea de código o una variable y representa una dirección de memoria que contiene cualquier instrucción o dato. A continuación se muestra la posición de la etiqueta en una línea de código ensamblador. Los dos puntos posteriores a la etiqueta son opcionales.

  label: [condition] instruction unit operands comments parallel bars

  Las etiquetas deben reunir las siguientes condiciones:

  - El primer caracter de la etiqueta debe ser una letra o un guión bajo (_) seguido por una letra
  - La etiqueta debe situarse en la primera columna del archivo de texto
  - La etiqueta puede incluir hasta 32 caracteres alfanuméricos

- **Barras Paralelas** (*parallel bars*). Para indicar que una instrucción se ejecuta en paralelo con la **instrucción** previa se indica con barras paralelas (||). Este campo es un espacio en blanco para una instrucción que no se ejecuta en paralelo con la instrucción previa.

  label **parallel bars** [condition] instruction unit operands comments

- **Condiciones** (*condition*). El C6000 tiene cinco registros disponibles para las condiciones A1, A2, B0, B1 y B2. Enseguida se muestra la posición de una condición en una línea de código ensamblador.

  label parallel bars **[condition]** instruction unit operands comments

  Todas las instrucciones de los dispositivos C6000 son condicionales:

  - Si no se especifica ninguna condición, la instrucción siempre será ejecutada.
  - Si se especifica y esa condición es verdadera, la instrucción se ejecuta. Por ejemplo:

Con esta condición ...	La instrucción se ejecuta si ...
[A1]	A1 = 0
[!A1]	A1 = 0

  - Si se especifica una condición y es falsa, la instrucción no se ejecuta.

Con esta condicion...	La instruccion se ejecuta si ...
[A1]	A1 = 0
[!A1]	A1 = 0

- **Instrucciones** (*instruction*) Las instrucciones en codigo ensamblador son de dos tipos directivas y mnemonicos

  - *Directivas* Son comandos para el ensamblador (asm6x) que controlan el proceso de ensamblado o que definen la estructura de los datos (constantes o variables) en el programa de lenguaje ensamblador Todas las directivas de ensamblador comienzan con un punto como se muestra en el listado parcial de la tabla 2 8

  - *Mnemónicos* Son las verdaderas instrucciones del microprocesador que se encuentran en rutinas y ejecutan la operacion del programa Los mnemónicos comienzan a partir de la columna 2

A continuacion se muestra la posicion de la instruccion en una linea de codigo en ensamblador

label parallel bars [condition] **instruction** unit operands comments

Tabla 2 8 Directivas del TMS320C6x

Directivas	Descripcion
sect "name"	Crea seccion de información (datos o codigo)
double value	Reserva dos palabras consecutivas de 32 bits (64 bits) en memoria y las llena con la representacion en punto flotante IEEE de doble precision (64-bit) del valor especificado
float value	Reserva 32 bits en memoria y los llena con la representacion de punto flotante IEEE de precision simple del valor especificado
[illegible]	Reserva 32 bits en memoria y los llena con el especificado valor
[illegible]	
short value	Reserva 16 bits en memoria y los llena con el especificado valor
[illegible]	
byte value	Reserva 8 bits en memoria y los llena con el valor especificado

- **Unidades Funcionales** (*unit*) El CPU C6000 contiene ocho unidades funcionales, mismas que se muestran en la figura 2 6 y se describieron en la tabla 2 1

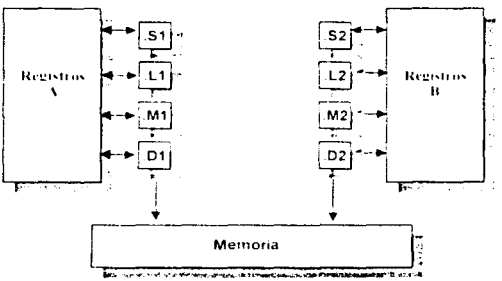

Figura 2 6 Unidades funcionales del TMS320C6x

Es opcional especificar la unidad funcional en el código. La especificación puede ser usada para documentar que recurso (s) utiliza cada instrucción.

label parallel bars [condition] instruction **unit** operands comments

- **Operandos** (*operands*). Las instrucciones tienen los siguientes requerimientos, para manejar los operandos del código ensamblador:

  - Todas las instrucciones requieren un operando destino.
  - La mayoría de las instrucciones requieren uno o dos operandos fuente.
  - El operando destino debe estar en la misma unidad de registros que el operando fuente.
  - Un operando fuente de cada archivo de registros por paquete de ejecución puede llegar del archivo de registros opuesto del otro operando fuente.

Cuando un operando llega de otro archivo de registro, la unidad incluye una X como se muestra a continuación, indicando que la instrucción utiliza una de las trayectorias cruzadas.

```
ADD     L1      A0,A1,A3
ADD     L1X     A0,B1,A2
```

Todos los registros excepto B1 son de misma lado del CPU

Las instrucciones del C6000 usan tres tipos de operandos para el acceso a datos:

- *Operandos de registro*. Señalan al registro que contiene el dato.
- *Operando constante*. Especifica el dato dentro del código ensamblador.
- *Operando puntero*. Contiene la dirección del valor de datos.

Unicamente las instrucciones de carga y almacenamiento requieren y usan operandos puntero para mover valores de datos entre memoria y un registro. Enseguida se muestra la posición de los operandos en una línea de código ensamblador.

label parallel bars [condition] instruction unit **operands** comments

- **Comentarios** (*comments*). Como ocurre con todos los lenguajes de programación, los comentarios proporcionan la documentación del código. A continuación se muestra la posición del comentario en una línea de código ensamblador.

label parallel bars [condition] instruction unit operands **; comments**

Las siguientes son directrices para usar comentarios en código ensamblador:

- Un comentario puede comenzar en cualquier columna, cuando se precede por un punto y coma (;).
- Un comentario debe comenzar en la primera columna, cuando se precede por un asterisco (*).
- Los comentarios no son indispensables, pero se recomienda su uso.

Ejemplo Código en ensamblador en paralelo del producto punto en aritmética de punto fijo

```
MVK .S1 100, A1 ; carga contador del ciclo
||    ZERO .L1 A7 ; limpia acumulador
LOOP:
LDH .D1 *A4++,A2 ; carga los A1 en memoria
||    LDH .D2 *B4++,B2 ; carga los B1 en memoria
SUB .S1 A1,1,A1 ; decrementa contador de ciclo
   [A1]    B .S1 LOOP ; retorna a LOOP
NOP 2 ; dos tiempos de retardo para LDH
MPY .M1X A2,B2,A6 ; A1 * B1
NOP ; dos tiempos de retardo para MPY
ADD .L1 A6,A7,A7 ; sum += (A1 * B1)
```

❑ Código en C/C++

El compilador C/C++ del 'C6000 traduce programas de C ANSI estándar a lenguaje ensamblador del 'C6000

Soporta todas las funciones de las librerías que conforman el estándar ANSI C. Las librerías incluyen funciones para entrada y salida estándar, manipulación de cadenas, manipulación de asignación dinámica de memoria, conversión de datos, temporización, funciones trigonométricas, exponenciales e hiperbólicas. Las funciones de manejo de señales no están incluidas porque son específicas para cada sistema

❖ *Tipos de datos*

La tabla 2.10 muestra los diferentes tipos de datos que se pueden manejar con los dispositivos TMS320C6x, su tamaño respectivo, su representación y el rango de valores que alcanza cada tipo de datos

Tabla 2.9 Diferentes tipos de datos que manejan los dispositivos TMS320C6x

Tipo	Tamaño	Representación	Rangos	
			Mínimo valor	Máximo valor
char, signed char	8 bits	ASCII	−128	127
unsigned char	8 bits	ASCII	0	255
short	16 bits	complemento a 2	−32 768	32 767
unsigned short	16 bits	Binario	0	65 535
int, signed int	32 bits	complemento a 2	−2 147 483 648	2 147 483 647
unsigned int	32 bits	Binario	0	4 294 967 295
long, signed long	40 bits	complemento a 2	−549 755 813 888	549 755 813 887
unsigned long	40 bits	Binario	0	1 099 511 627 775
enum	32 bits	complemento a 2	−2 147 483 648	2 147 483 647
float	32 bits	IEEE 32-bit	1.175 494e−38 †	3.40 282 346e+38
Double	64 bits	IEEE 64-bit	2.22 507 385e−308 †	1.79 769 313e+308
long double	64 bits	IEEE 64-bit	2.22 507 385e−308 †	1.79 769 313e+308
pointers, references, pointer to data members	32 bits	Binario	0	0xFFFFFFFF

† Para cifra mínima precisión

❖ *Ejemplos*

- Llamando funciones en ensamblador en el codigo C/C++

El siguiente ejemplo muestra una funcion en C *main*, que llama a otra en ensamblador *asmfun*. La funcion *asmfun* toma un solo argumento, añade este a la variable global llamada *gvar* y regresa el resultado

(a) C program

```
extern " C"  {
extern int asmfunc(int a);   /* declaración de la función como
                                externa */
int gvar = 4;                /* define la variable global */
}
void main(){
      int i = 5;
      i = asmfunc(i);        /* llamada de función normal */
}
```

(b) Assembly language program

```
      .global _asmfunc
      .global _gvar
_asmfunc:
      LDW *+b14(_gvar),A3
      NOP 4
      ADD a3,a4,a3
      STW a3,*+b14(_gvar)
      MV a3,a4
      B b3
      NOP 5
```

- Producto punto en aritmética de punto flotante

```
float dotp1(const float a[], const float b[])
{
int i;
float sum = 0;
      for (i=0; i<512; i++)
            sum += a[i] * b[i];
return sum;
}
```

Entorno de desarrollo integrado del Code Composer Studio (IDE)

El IDE del *Code Composer Studio* esta diseñado para permitir editar, construir y depurar programas del DSP

❖ *Caracteristicas del editor de codigo de programas*

El *Code Composer Studio* permite editar código en C y en ensamblador. Puede verse el código fuente C o con las instrucciones correspondientes en ensamblador, mostrando las sentencias de C despues (utilizando del menu *View ->Mixed Source/ASM*), como muestra la figura 2.7

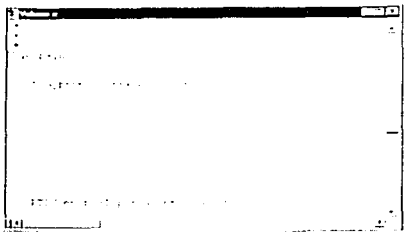

Figura 2.7 Codigo fuente en C mezclado con codigo ensamblador

El editor integrado, proporciona soporte a las siguientes actividades

- Resalta palabras clave comentarios y cadenas en color
- Marca bloques de C en parentesis y corchetes encontrando el par o próximo parentesis o corchete
- Niveles de sangrado
- Busqueda y reemplazo en uno o mas archivos
- Deshaciendo y rehaciendo multiples acciones
- Obtencion de ayuda sensible al contexto

❖ *Caracteristicas de construccion de aplicaciones*

Con el *Code Composer Studio* se puede crear un proyecto de trabajo que es usado para construir la aplicacion. Los archivos en el proyecto, pueden ser archivos de codigo fuente en C, archivos en ensamblador archivos objeto librerias, archivos de comando del ligador y archivos de declaración *include*. En la figura 2.8, se muestra la forma que adquiere un proyecto de trabajo con el CCS.

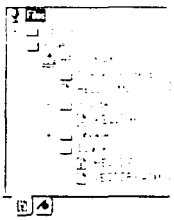

Figura 2.8 Proyecto de trabajo

❖ *Caracteristicas de depuración de aplicaciones*

El *Code Composer Studio* provee soporte para las siguientes actividades de depuración.

- Establecer puntos de ruptura
- Actualizar automaticamente las ventanas en los puntos de ruptura
- Visualizar el valor de las variables

- Ver y editar registros y memoria
- Ver el stack de las llamadas a funciones
- Usar herramientas punto de prueba para flujo de datos de y a la tarjeta
- Graficar las señales de la tarjeta
- Generar estadísticas de ejecución
- Ver instrucciones desensambladas e instrucciones en C ejecutandose sobre la tarjeta
- Proporciona un lenguaje GEL. Este lenguaje permite añadir funciones al menú para optimizan las tareas comunes

Para obtener una mejor perspectiva de las herramientas de edición y depuración además del entorno de desarrollo del *Code Composer Studio* consultar en la documentación *Code Composer Studio User's Guide* de *Texas Instruments*.

DSP/BIOS *plug-ins*

Los *plug-ins* del *Code Composer Studio* proporcionan con el DSP/BIOS soporte para el análisis en tiempo real. Se pueden usar para visualmente probar, señalar y monitorear una aplicación DSP con el mínimo impacto en el *performance*.

Las API's DSP/BIOS proporcionan las siguientes capacidades en tiempo real:

- *Mensajes del programa* (*program tracing*). Despliega los eventos escritos en registros designados y refleja dinamicamente el control del flujo durante la ejecución del programa.
- *Monitoreo del Performance* (*performance monitoring*). Rastrea las estadísticas que reflejan el uso de los recursos como la carga del procesador y los tiempos de procesos.
- *Archivos de Flujo* (*flow file*). Liga archivos de datos en la PC a objetos de Entrada/Salida en el programa del DSP.

❖ *Configuración del DSP/BIOS*

Se pueden crear archivos de configuración utilizando el entorno del *Code Composer Studio*. Con ellos se definen objetos que son usados por las API's del DSP/BIOS. Estos archivos también simplifican el mapeo de memoria y el mapeo de los vectores en las rutinas de atención de interrupción.

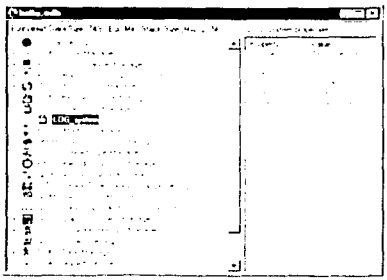

Figura 2.9 Ventana de configuración de objetos del DSP/BIOS

Cuando se abre un archivo de configuración del DSP/BIOS el *Code Composer Studio*, muestra un editor visual que permite crear y establecer propiedades para objetos de tiempo real. Estos objetos son usados en las llamadas a las API's del DSP/BIOS. También incluyen interrupciones por software, tuberías de I/O, mensajes de eventos (*logs*) etc. La figura 2.9, muestra el editor visual del DSP/BIOS.

❖ *Modulos del DSP/BIOS*

Las API's del DSP/BIOS están divididas en los siguientes modulos:

- **ATM** Funciones atómicas que pueden ser usadas para manipular datos compartidos.
- **C62** Este modulo proporciona funciones específicas del DSP para manejo interrupciones.
- **CLK** Este modulo controla el timer interno del DSP y proporciona un reloj lógico de 32 bits en tiempo real con alta resolución.
- **DEV** Este módulo permite crear y usar sus propios controladores de dispositivos.
- **HST** El módulo *host* manejas este tipo de objetos de canal que permiten a una aplicación transmitir flujo de datos entre al DSP y un *host*. Los canales *host* son configurados estáticamente para entrada o salida.
- **HWI** El modulo de interrupciones por hardware, proporciona soporte para rutinas que atienden este tipo de interrupciones interrupciones.
- **IDL** Este módulo maneja funciones *idle* que corren cuando no existe ninguna función de mayor prioridad para ejecutarse.
- **LCK** El modulo *lock* maneja recursos globales compartidos y es usado para controlar el acceso a estos recursos entre varias tareas que los disputen.
- **LOG** Este modulo maneja objetos tipo *LOG* que capturan eventos en tiempo real mientras el programa objeto se ejecuta. Se puede usar *logs* de sistema o se puede definir *logs* propios. Con el *Code Composer Studio* se puede visualizar estos mensajes *log*s en tiempo real.
- **MBX** El modulo *mailbox* maneja objetos que pasan mensajes de una tarea a otra.
- **MEM** El modulo de memoria permite especificar los segmentos de memoria requeridos.
- **PIP** Este modulo maneja tuberías de datos (*pipe*), que son usadas como flujos de *buffers* para entrada y salida de datos. Estas tuberías de datos proporcionan una estructura de datos consistente para manejar entradas/salidas entre el DSP y otros dispositivos periféricos, en tiempo real.
- **PRD** El modulo de manejo de funciones periodicas, administra objetos de este tipo. Permite activar ejecuciones cíclicas de una función. La velocidad de ejecución para estos objetos puede ser controlada por la frecuencia de reloj mantenida por el modulo CLK ó por llamadas regulares a la función *PRD_tick*.
- **QUE** Este módulo maneja estructuras de colas de datos.
- **RTDX** Permite el intercambio de datos en tiempo real, entre la PC y el DSP y además, analizar y desplegar los datos en la PC usando una automatización *OLE cliente* (esta puede estar programada en *Visual C++, Visual Basic, Excel, Matlab, LabView*, etc.)
- **SEM** El modulo de semáforos permite sincronizar tareas y realizar exclusión mutua.
- **SIO** El modulo *stream* maneja objetos que proveen eficiencia en tiempo real de dispositivos de I/O.

- **STS**. Modulo de estadísticas: administra acumulación de estadísticas clave en tiempo real, mientras el programa se ejecuta
- **SWI**. Este módulo administra las interrupciones por software. Estas interrupciones son procesos que tienen menor prioridad que las interrupciones por hardware y mayor prioridad que el modulo de tareas. Cuando una función anuncia a un objeto SWI con una llamada de API, el modulo SWI programa para ejecución la función correspondiente
- **SYS**. Modulo de dispositivos de sistema: proporcionan funciones de propósito general que realizan servicios de sistema básicos, como parar la ejecución de programa e imprimir texto con formato
- **TRC**. El modulo *trace* envía mensajes a la ventana de depuración en tiempo real
- **TSK**. Modulo de tareas (las tareas son procesos con prioridad mas baja que las interrupciones por software)

Emulación de hardware e intercambio de datos en tiempo real (RTDX)

Los DSP de *Texas Instruments* proporcionan emulación en el chip habilitadas por el *Code Composer Studio*, para la ejecución de programas de control y monitoreo de la actividad del programa en tiempo real. La comunicación con este soporte de emulación ocurre a través de un enlace mejorado JTAG. Este enlace es una vía de baja intromisión de conexión en cualquier sistema DSP. Una interface de emulación como TI XDS510 proporciona la conexión JTAG al lado del *host*.

El hardware de emulación proporciona una variedad de capacidades:

- Iniciación, detención o *reset* del DSP
- Carga de código o datos dentro del DSP
- Examina registros o memoria del DSP
- Puntos de ruptura de instrucciones de hardware o dependencia de datos
- Soporte de conteo, incluyendo perfiles exactos de ciclos
- Intercambio de datos en tiempo real (RTDX) entre el *host* y el DSP

El RTDX proporciona, en tiempo real, visibilidad continua en el trayecto de la operación de aplicaciones. El RTDX permite desarrollar sistemas para transferir datos entre una computadora *host* y el DSP, sin parar la aplicación designada. Los datos pueden ser analizados y visualizados sobre el *host* usando cualquier automatización OLE. Esto aporta el tiempo de desarrollo, dando al diseñador una representación realista del trayecto de la operación que realmente sigue el sistema.

El RTDX consta de ambos componentes, *host* y DSP. Una pequeña librería de software RTDX corre sobre el DSP. El diseñador de aplicaciones DSP realiza llamadas de funciones a estas librerías APIs, para pasar datos a o del DSP. Las librerías usan emulación de hardware en el chip, para mover datos *a o de* la plataforma *host* a través de una interface JTAG. La transferencia de datos al *host* ocurre en tiempo real, mientras la aplicación del DSP esta corriendo. La figura 2.10 muestra un diagrama de bloques con los componentes que intervienen en un intercambio de datos en tiempo real.

FALTA
PAGINA

48

CAPÍTULO 3

ANÁLISIS Y DISEÑO

3.1 Descripción general

El sistema para el reconocimiento de palabras aisladas con un DSP, que se desarrollará será capaz de identificar ocho palabras distintas con dos locutores diferentes en tiempo real.

Ambos locutores pueden ser modificados de acuerdo a las palabras de entrenamiento que se proporcionen. Estas palabras deben ser capturadas con el DSP y procesadas en la PC para obtener los patrones de referencia. La elección del locutor se hace dependiendo de la aplicación del sistema.

Se utilizarán las técnicas del procesamiento de voz más comunes. Se hará análisis de la señal en tiempo corto, aprovechando las propiedades de la voz en pequeños intervalos, utilizando secuencias de 128 muestras de la señal. Para cada secuencia se calculan sus coeficientes LPC representativos, mismos que serán empleados tanto en el entrenamiento como en el reconocimiento. Por tanto, la obtención de los LPC's constituye la parte medular del sistema.

Por cierto, tomando en consideración que la frecuencia de muestreo utilizada por el DSP para señales de voz es de 8 kHz, se tiene que el tamaño en tiempo de cada trama de 128 muestras es de 16 mseg.

Los diagramas de bloques que se muestran en las figuras 3.1 y 3.2 ilustran, de forma general, los procesos que se realizan en el entrenamiento y reconocimiento de palabras aisladas respectivamente.

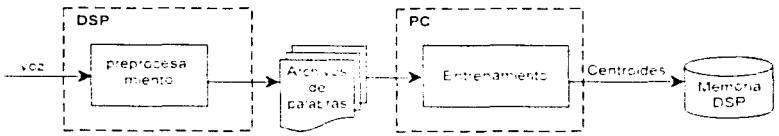

Figura 3.1 Etapa de Entrenamiento

Como se observa en los diagramas, existe un módulo común en ambas etapas, el preprocesamiento. Por esta razón, el preprocesamiento se manejará como un módulo independiente, para efecto del análisis.

Como se muestra en la figura 3.1, durante la etapa de entrenamiento se capturan diferentes señales de voz, a la que denominaremos *repeticiones de palabras*. A cada repetición se le aplica preprocesamiento para obtener una palabra delimitada. Con esta palabra se obtienen los coeficientes LPC's de cada trama de 128 muestras. Posteriormente, se agrupan todos los coeficientes LPC's y con ellos se obtienen sus centroides correspondientes. Estos centroides se guardan en la memoria del DSP.

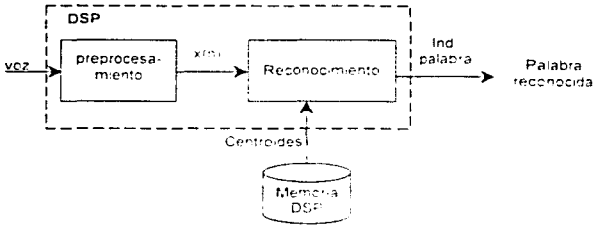

Figura 3.2 Etapa de Reconocimiento

En la etapa de reconocimiento, como se observa en la figura 3.2, se captura la palabra que se desea reconocer. A esta palabra también se le aplica preprocesamiento, para recortarla y obtener sus coeficientes LPC's. Utilizando los centroides calculados durante el entrenamiento y almacenados en la memoria del DSP, se realizan las comparaciones necesarias para efectuar el reconocimiento. El éxito del evento dependerá de ciertos parámetros estadísticos obtenidos mediante el análisis de una población de resultados.

3.2 Preprocesamiento

En el preprocesamiento se recibe una señal de voz y se determinan sus límites. La señal entra a un convertidor A/D donde es cuantizada. Posteriormente se le aplican dos tipos de filtros: paso bajas y de preénfasis para eliminar ruidos de altas frecuencias y realzar las frecuencias altas presentes en la voz, respectivamente. Enseguida se divide la señal en tramas de 128 muestras y a cada trama se le aplica una ventana de tipo Hamming, para suavizar el espectro.

Es necesario aclarar, que cuando se inicializa el sistema, se determinan ciertos parámetros umbrales, necesarios para la detección de inicio y fin de la palabra. Estos parámetros se calculan recolectando muestras del ruido presente en el ambiente circundante. El módulo encargado de realizar este proceso, se denomina ruido ambiental. Finalmente, con las tramas provenientes del ventaneo y con los umbrales obtenidos para el ruido ambiental, se determina el inicio y fin de cada palabra.

El preprocesamiento se ejecuta completo en el DSP. La figura 3.3 muestra el diagrama de bloques para este módulo.

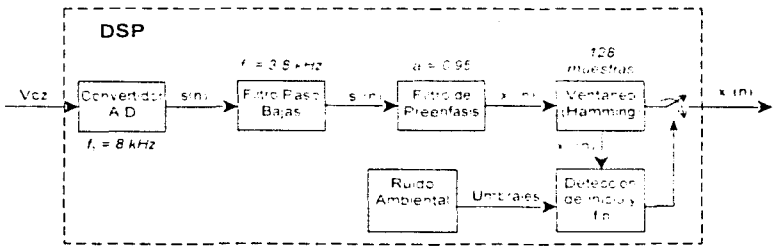

Figura 3.3 Diagrama de bloques para el módulo de preprocesamiento

Como ya se mencionó, el preprocesamiento es un módulo común en el entrenamiento y en el reconocimiento. Sin embargo, existe una diferencia sutil: durante el entrenamiento, cada palabra recortada se almacena en un archivo independiente que posteriormente se utilizará para determinar los centroides representativos. Mientras que en el reconocimiento, la palabra recortada se utiliza directamente para hacer las comparaciones con los centroides.

El preprocesamiento consta de los siguientes módulos: filtro paso bajas, filtro de preénfasis, ventaneo, detección de ruido ambiental[1], y detección de inicio y fin de la palabra. Estos módulos se describen a continuación:

- ***Convertidor A/D.*** Para la obtención de los datos, se captura una señal de voz previamente amplificada[1]. Esta señal se hace pasar por un convertidor A/D (con frecuencia de muestreo de 8 kHz), que se comunica con el DSP a través de la interface serial. El convertidor A/D es un dispositivo integrado al *starter kit* del DSP, que convierte una señal analógica en una señal digital discreta, con un formato de salida en complemento a 2. Cada valor de la señal digital se representa con 16 bits, correspondientes a diferentes niveles de cuantización para la señal analógica.
- ***Filtro paso bajas.*** La señal de voz que se captura por medio del DSP contiene, además de la información de interés, ruido y distorsiones que presentan altas frecuencias. El filtro paso bajas elimina gran parte de estas frecuencias indeseables.
- ***Filtro de preénfasis.*** La señal de voz original presenta poca influencia de las frecuencias mas altas (aproximadamente de 3000 Hz en adelante). Por esta razón se debe dar mayor amplitud a estas frecuencias. Este filtro logra homogeneizar las amplitudes de todo el intervalo de frecuencias.
- ***Ventaneo.*** La señal de voz es un proceso aleatorio no estacionario, pero en intervalos muy pequeños, sus propiedades estadísticas no varían, por lo que se puede suponer un proceso estacionario para este intervalo de tiempo. Esto conduce a métodos de procesamiento en tiempo corto, en los cuales solo una parte de la señal es aislada y procesada en pequeños segmentos, a los que se denomina *tramas*. Además se pueden utilizar ventanas de tipo Hamming, que permite suavizar los cortes de la señal entre una trama y la siguiente.
- ***Detección de ruido ambiental.*** Para determinar los parámetros que se utilizan para la detección del inicio y fin de una palabra, se deben capturar algunas tramas que identifiquen el ruido del medio ambiente circundante.
- ***Detección de inicio y fin.*** Este módulo permite recortar la señal de voz a límites bien definidos, en donde se tiene la certeza de que existe una palabra. Como se ve en la figura 3.3, la detección de inicio y fin se emplea como interruptor a la salida del ventaneo. La razón fundamental del recorte es asegurar que existe una señal de voz. Además se evita realizar cálculos para obtener la autocorrelación y los coeficientes de predicción lineal, en aquellas tramas que no pertenecen a una palabra.

3.3 Entrenamiento

Para el entrenamiento, se deben capturar diversas palabras recortadas, por medio del módulo de preprocesamiento, que representan las diferentes *repeticiones*. A partir de estas repeticiones, se obtiene un conjunto de centroides que son almacenados en la memoria del DSP. El entrenamiento se ejecuta utilizando los recursos de una PC.

[1] Debido a las características del micrófono, es necesario amplificar la señal de voz, para adecuarla a las necesidades del convertidor A/D. Esta etapa no se describe en el proyecto.

Los archivos de palabras previamente almacenados, son utilizados para aplicarles varios procesos a cada uno. Primero se determinan sus vectores de autocorrelación para obtener los coeficientes de predicción lineal. Enseguida estos coeficientes son segmentados de forma lineal, en cuatro grupos. Se sigue el mismo procedimiento para todas las repeticiones de una misma palabra. Posteriormente una vez que se tienen segmentadas todas las repeticiones, se agrupan en conjuntos del mismo segmento. Para cada conjunto se obtienen centroides que representaran al segmento. Finalmente estos centroides son almacenados en memoria. La figura 3.4 esquematiza el proceso completo del entrenamiento.

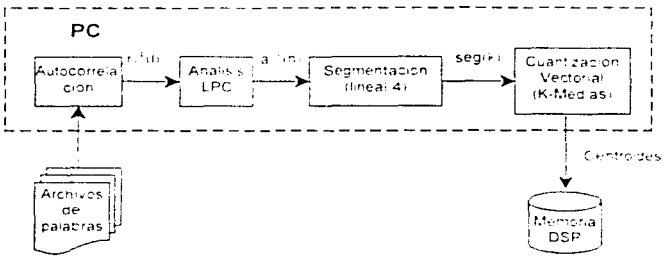

Figura 3.4 Modulo de Entrenamiento

El entrenamiento es la etapa que nos permite obtener los patrones de comparación, para las señales de voz. Consta de los siguientes modulos: autocorrelacion, analisis LPC, segmentacion y cuantizacion vectorial.

- ***Autocorrelación*** En este modulo se obtienen los primeros ***p*** valores del vector de autocorrelacion (donde ***p*** representa el orden del *predictor*) que corresponden a cada trama.
- ***Análisis LPC*** Este modulo calcula los coeficientes de prediccion lineal. Con estos coeficientes se obtiene una aproximacion al modelo del tracto vocal, como ya se menciono anteriormente.
- ***Segmentacion*** En este modulo se realiza una separacion de los vectores LPC's de la señal, en cuatro regiones representativas. De este modo se obtienen cuatro grupos (divididos linealmente) con las repeticiones de cada palabra.
- ***Cuantización Vectorial*** Este proceso obtiene los centroides representativos de cada segmento. Esto es, obtiene un patron especifico para cada palabra, mismo que sera utilizado durante la etapa de reconocimiento.

3.4 Reconocimiento

El reconocimiento recibe las tramas de una señal de voz, proveniente del preprocesamiento, para efectuar comparaciones con los centroides y obtener un indicador a la palabra reconocida.

Cada trama recibida es utilizada para calcular su autocorrelacion y sus coeficientes de prediccion lineal. Este proceso se repite hasta que el preprocesamiento detecta el fin de la palabra. Una vez detectado el final, se segmentan los vectores LPC y los vectores de autocorrelacion correspondientes, de forma lineal. Con cada segmento se realiza una comparacion, utilizando la distancia de *Itakura*, con los centroides que correspondan al mismo segmento [Herrera, 1999].

El resultado de esta comparación, aunado con ciertos parámetros estadísticos, determina el éxito o fracaso del reconocimiento. La figura 3.5 muestra el diagrama de bloques del reconocimiento.

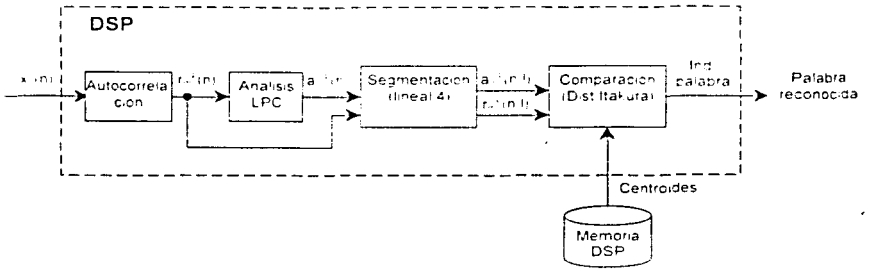

Figura 3.5 Módulo de Reconocimiento

El reconocimiento permite identificar una señal de voz similar a ciertos patrones definidos previamente. Consta de los siguientes módulos: autocorrelación, análisis LPC, segmentación y comparación.

- ***Autocorrelación*** En este módulo se obtienen los primeros ***p*** valores del vector de autocorrelación (donde ***p*** representa el orden del *predictor*), que corresponden a cada trama.
- ***Análisis LPC*** Este módulo calcula los coeficientes de predicción lineal. Con estos coeficientes se obtiene una aproximación al modelo del tracto vocal, como ya se mencionó anteriormente.
- ***Segmentación*** En este módulo se realiza una separación de los vectores LPC's de la señal y de los vectores de autocorrelación correspondientes, en cuatro regiones representativas. De este modo se obtienen cuatro grupos (divididos linealmente), tanto de los vectores LPC's como de los vectores de autocorrelación.
- ***Comparación*** Es el proceso que permite cotejar los patrones obtenidos durante el entrenamiento y almacenados en memoria, con los vectores LPC's previamente segmentados. El resultado, aunado con ciertos parámetros estadísticos, determina el éxito o fracaso del reconocimiento de una palabra.

El siguiente capítulo describe la forma como se implantó el sistema reconocedor de palabras aisladas utilizando el DSP TMS320C6711 de *Texas Instruments*, utilizando para ello diagramas de estados.

CAPÍTULO 4

DESARROLLO

4.1 Introducción

Para hacer reconocimiento de palabras aisladas en tiempo real se necesita utilizar un tipo de arquitectura computacional adecuada. En general se pueden encontrar dos tipos de arquitecturas la *Von Newman* y la *Harvard*. La primera de ellas se caracteriza por tener solo un bus para datos y direcciones. En contraste, la arquitectura tipo *Harvard* maneja un bus de datos y otro de direcciones con lo que se aumenta la velocidad de ejecución. A este ultimo tipo de arquitecturas pertenecen los DSP's.

Las caracteristicas mas importantes de los DSP's son las siguientes: permiten hacer procesamiento en paralelo por medio del *pipeline*, cuenta con una arquitectura tipo *Harvard*, contiene multiplicadores por hardware, maneja memorias internas rapidas, ademas incluyen un amplio conjunto de instrucciones especificas para el procesamiento digital de señales.

En particular, el DSP TMS320C6711 de *Texas Instruments* puede ejecutar hasta ocho instrucciones al mismo tiempo. Cuenta con ocho unidades funcionales (seis unidades ALU's y dos Multiplicadores) y dos memorias *cache* de 64 K bytes cada una (para datos y programa), entre otras ventajas. Este DSP se conecta a través del puerto paralelo a una PC.

4.2 Descripción general del sistema

Para la realización del sistema reconocedor de palabras aisladas con el DSP TMS320C6711 de *Texas Instruments*, se dividio el proyecto en dos fases: entrenamiento y reconocimiento.

Durante el entrenamiento se capturaron veinte repeticiones de cada palabra a reconocer. Para que el sistema sea capaz de identificar a dos locutores distintos se utilizaron diez repeticiones de un locutor para cada palabra. Las lecturas se realizaron con el DSP, con la finalidad de tener las mismas condiciones tanto en el entrenamiento y como en el reconocimiento.

En el análisis se describieron las caracteristicas del sistema mediante diagramas de bloques funcionales, tratando de explicar con mayor claridad la secuencia de procesos que se llevan a cabo. Ahora se describira el sistema mediante diagramas de estados, para hacer mas clara la visualización del comportamiento de los algoritmos desarrollados.

En la figura 4.1 se muestra el diagrama de estados con los modulos que se ejecutan en el DSP. Estos estados corresponden a la etapa del entrenamiento.

Como se observa en el diagrama de la figura 4.1, la fase de entrenamiento comienza con el estado de iniciación y calculo de umbrales. En este estado se ejecutan los siguientes procesos: configuración del convertidor A/D y calculo de umbrales del ruido ambiental. Una vez terminados los procesos anteriores, se activa el siguiente estado, por medio de la interrupcion del puerto serie.

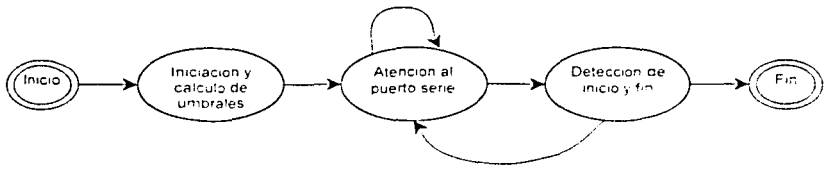

Figura 4.1 Diagrama de estados para los procesos del DSP durante el entrenamiento

El estado de atención al puerto serie realiza estos procesos: lectura de cada muestra, filtrado paso bajas y filtrado de preénfasis. Se mantiene en este estado hasta que completa una trama de 128 muestras. Al completar la trama, se activa el siguiente estado, por medio de una interrupción por software.

El último estado, detección de inicio y fin, efectúa los procesos mencionados a continuación: ventaneo (Hamming), cálculo de la magnitud promedio, tasa de cruces por cero y comparación con los umbrales del ruido. Mientras que no se detecte el fin de la palabra, regresa al estado anterior a solicitar otra trama; de lo contrario, finaliza el programa. Al terminar, el programa obtiene una palabra que es almacenada en la memoria del DSP.

Una vez que se tiene la palabra en memoria, se almacena en un archivo con formato ASCII. La palabra será procesada posteriormente con el paquete *Matlab* en una PC.

Para completar la fase de entrenamiento, es necesario seguir los siguientes procesos: división en tramas de la palabra, autocorrelación, análisis de los coeficientes LPC's, segmentación y cuantización vectorial. La figura 4.2 muestra el diagrama de bloque con los procesos anteriores, ejecutados en la PC.

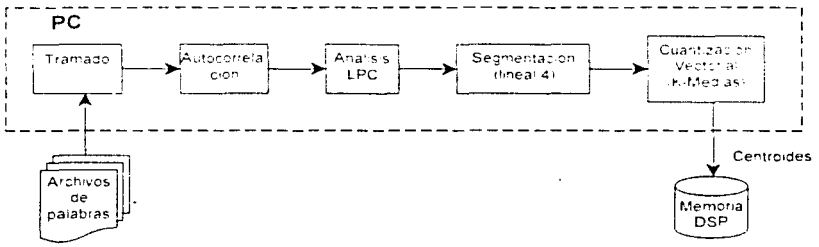

Figura 4.2 Diagrama de bloques de procesos en PC

Al finalizar el entrenamiento se obtienen los centroides representativos de cada palabra. Por otro lado, la fase de reconocimiento utiliza los centroides anteriores como patrón para la comparación con la señal de voz entrante en el DSP. Si existe similitud con respecto a los patrones y cae dentro de los umbrales estadísticos preestablecidos, se consigue su reconocimiento. En caso contrario, se emite un mensaje para solicitar la repetición de la palabra.

La figura 4.3 muestra el diagrama de estados para los módulos que constituyen la etapa de reconocimiento en el DSP.

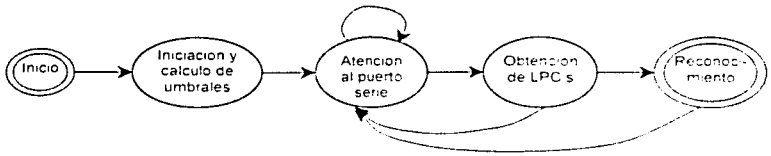

Figura 4.3 Diagrama de estados para las actividades del DSP en el reconocimiento

Como se ve en el diagrama de estados de la figura 4.3, los primeros dos estados son idénticos a los estados iniciales de la fase de entrenamiento. Sin embargo, ahora el estado de atencion al puerto serie, activa al estado de obtención de LPC s, por medio de una interrupción de software.

En el estado de obtencion de los coeficientes LPC s se realizan los procesos siguientes: ventaneo (Hamming), calculo de la magnitud promedio, tasa de cruces por cero, comparación con los umbrales del ruido, autocorrelacion y analisis LPC. Mientras que no se detecta el fin de la palabra, regresa al estado anterior, atencion al puerto serie, a solicitar otra trama. En caso contrario, se tiene una palabra completa y delimitada. Cuando se completa la palabra, se activa el estado de reconocimiento.

Este ultimo estado ejecuta dos procesos, segmentacion y comparacion. Al terminar la comparacion de una palabra, se obtiene un mensaje con el resultado alcanzado e inmediatamente, regresa al estado de atencion al puerto serie, para esperar una nueva palabra.

4.3 Fase de entrenamiento

El objetivo del entrenamiento es obtener los patrones de comparacion, de cada una de las palabras a reconocer (*diccionario de palabras*).

Se definio el tamaño del diccionario con ocho palabras, en español. Estas son: *arriba, abajo, adelante, atrás, alto, sigue, izquierda* y *derecha*. La eleccion de estas palabras, no tiene mayor relevancia. Sin embargo, se eligieron asi, porque representan ordenes claras y precisas para manipular un movil.

El entrenamiento utilizó veinte repeticiones, con dos locutores distintos, por cada palabra. Esta fase, se desarrollo en dos plataformas, el DSP y la PC.

A continuacion, se esquematizan con detalle, los procesos de cada estado involucrado en la fase de entrenamiento, mediante diagramas de flujo.

- ***Iniciacion y calculo de umbrales.*** La figura 4.4 esquematiza el diagrama de flujo para el estado de iniciacion y calculo de umbrales, del ruido ambiental.

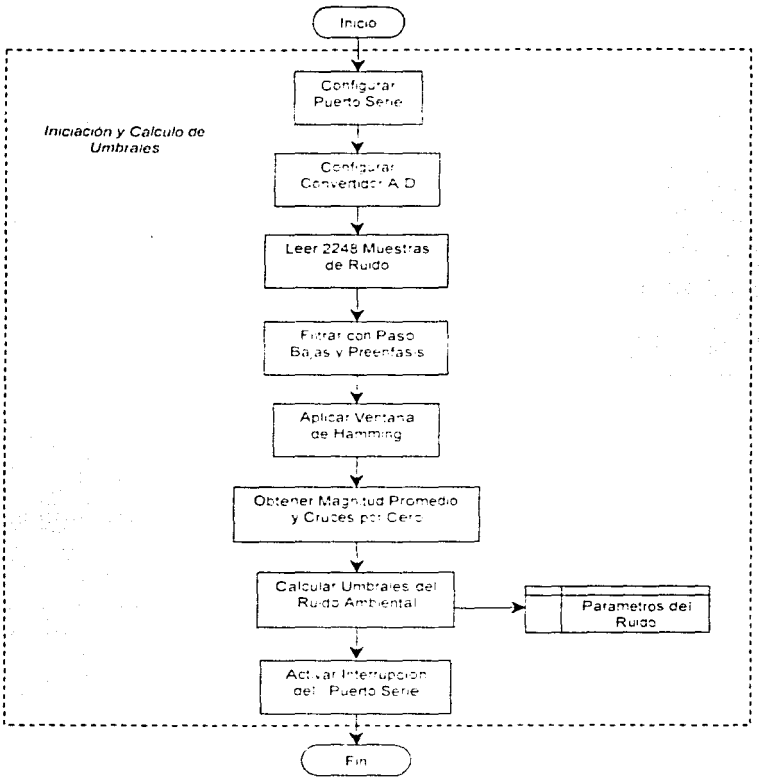

Figura 4.4 Diagrama de flujo para el estado de iniciacion y calculo de umbrales

Este estado solo se ejecuta una vez y se encarga de configurar tanto el puerto serie como el convertidor A/D. Ademas determina los umbrales del ruido ambiental. Enseguida se describen los procesos que lo constituyen:

- *Configurar puerto serie*. Este proceso habilita los registros del puerto serie para establecer la comunicacion entre el convertidor y el DSP.
- *Configurar convertidor A/D*. Es el proceso que activa los registros del convertidor para obtener una tasa de transferencia de 8 KHz. El convertidor utilizado, es el TLC320AD535C/I que proporciona el *starter kit* del DSP TMS320C6711 de *Texas Instruments*. Para obtener mas informacion al respecto, consultar el apendice.

- *Leer 2248 Muestras de Ruido* En este proceso se leen las primeras 200 muestras de la señal y se eliminan para desechar cualquier valor presente en el convertidor A/D. A continuación se capturan y almacenan las 2048 muestras siguientes (16 tramas). Se considera que estas 16 tramas representan el ruido ambiental.
- *Filtrar con paso bajas y preénfasis* Es el proceso que realiza dos tipos de filtrado a la señal de ruido. Las características de los filtros se mantienen invariables durante el entrenamiento y el reconocimiento. A continuación se describen los filtros paso bajas y de preénfasis.

  - Filtro paso bajas El filtro paso bajas utiliza un filtro digital de orden seis en cascada con frecuencia normalizada de corte de 0.95 * (2 π). Es decir, como la frecuencia de muestreo empleada con el DSP es de 8 kHz, la frecuencia analógica de corte es de 3.8 kHz. La figura 4.5 muestra la estructura del filtro digital de sexto orden en cascada.

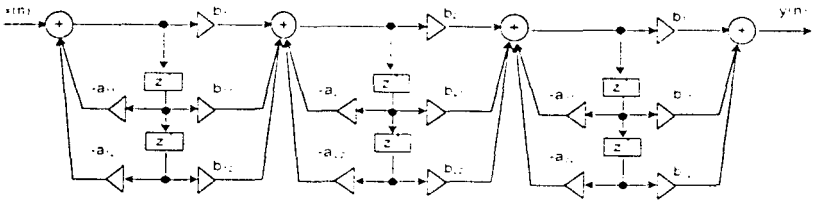

Figura 4.5 Filtro paso bajas de sexto orden

  - Filtro de Preénfasis En aplicaciones prácticas del análisis LPC, se debe utilizar un filtrado de preénfasis antes de calcular los parametros LPC. Esto se debe a que la aplicación del este filtro incrementa la energía en el espectro de altas frecuencias. La función de transferencia del filtro es

$$H(z) = 1 - a\,z^{-1}$$

Normalmente se utiliza un parametro a cercano a 1. En este caso $a = 0.95$. La figura 4.6 muestra la estructura del filtro.

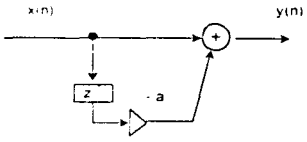

Figura 4.6 Filtro de preénfasis

- *Aplicar ventana de Hamming* Proceso que divide las muestras en tramas de 128 muestras y que ademas aplica una ventana de tipo Hamming a cada trama.
- *Obtener magnitud promedio y cruces por cero* Este proceso realiza el calculo de la magnitud promedio y la tasa de cruces por cero para cada trama.

- *Calcular umbrales del ruido ambiental* Proceso que obtiene una estimación de los umbrales que representan el nivel del ruido ambiental Los parametros estimados, se almacenan en memoria interna del DSP
- *Activar interrupción del puerto serie* Este proceso habilita la bandera que activa la *interrupcion* del puerto serie en el DSP Esta interrupción de hardware permite pasar al estado siguiente

- ***Atención al puerto serie*** La figura 4 7 muestra el diagrama de flujo para el estado de atencion al puerto serie

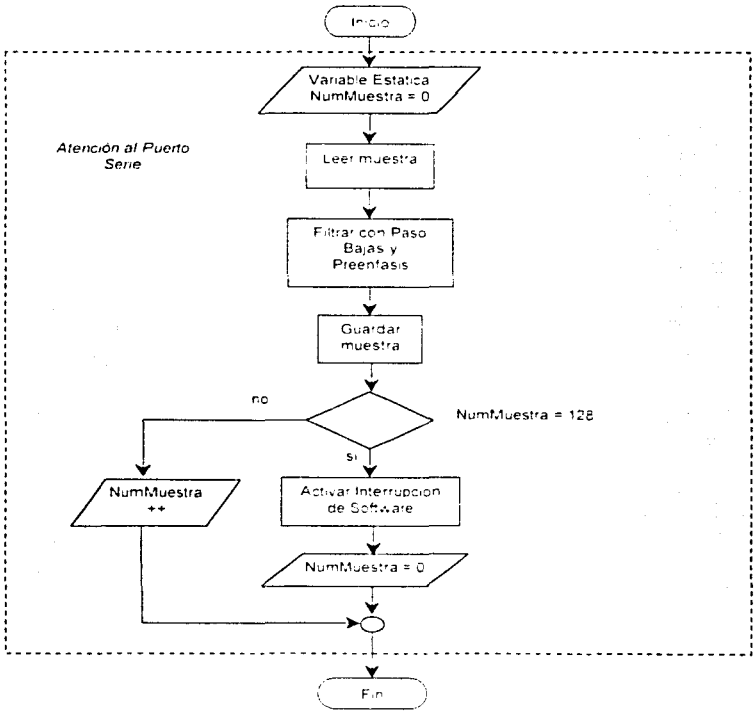

Figura 4 7 Diagrama de flujo para el estado de atención al puerto serie

Este estado se activa por medio de la interrupcion de hardware del puerto serie una vez que el convertidor A/D envia una muestra El estado lee la muestra y le aplica dos procesos de filtrado paso bajas y de preenfasis Almacena la muestra y finalmente, una vez que reúne 128 muestras activa una interrupción de software A continuación se describen los procesos del estado

- *Leer muestra*. Cada vez que el convertidor A/D envía una muestra se activa una interrupción de hardware. La interrupción avisa al DSP que existe un nuevo dato en el registro del puerto serie. Este proceso toma el dato proveniente del convertidor y lo envía al proceso de filtrado.
- *Filtrar con paso bajas y preénfasis*. El dato recopilado se pasa por un filtro paso bajas de orden seis y frecuencia de corte analógico de 3.8 KHz. Después se le aplica un filtro de preénfasis con parámetro $a = 0.95$. La estructura de ambos filtros se muestra en las figuras 4.5 y 4.6.
- *Guardar muestra*. Una vez filtradas las muestras, este proceso se encarga de almacenarlas consecutivamente en un buffer.
- *Activar interrupción de software*. *Una vez que se han almacenado consecutivamente 128* muestras (tamaño de una trama), este proceso habilita una interrupción de software para activar el estado de detección de inicio y fin de la palabra.

- ***Detección de inicio y fin***. El diagrama de flujo de la figura 4.8 muestra la secuencia de procesos que se efectúan durante el estado de detección de inicio y fin de la palabra.

  El estado de detección de inicio y fin determina si la trama recibida del estado anterior pertenece a una palabra. Cuando las tramas recibidas pertenecen a una palabra son guardadas en memoria. Primeramente se calcula la magnitud y los cruces por ceros de la trama. Estos valores se comparan con los umbrales obtenidos para el ruido ambiental. Si los umbrales son mayores se trata de una trama de silencio. Cuando los umbrales son menores es una trama de voz. Existen dos registros donde se lleve el recuento del número de tramas de voz y de silencio.

  Cuando la trama recibida es de voz se incrementa su registro y se inicializa el registro de las tramas de silencio. A la trama se le aplica una ventana del tipo Hamming y se guarda en memoria.

  Si la trama es de silencio y los registros que llevan el recuento se encuentran debajo de ciertos límites preestablecidos, la trama se toma como parte de la palabra y en consecuencia también se le aplica una ventana Hamming y se almacena. Los límites preestablecidos, fueron obtenidos de manera experimental, considerando las características de los fonemas y pequeñas perturbaciones del ruido ambiental.

  Cuando existen más de diez tramas de silencio consecutivas (más de 160 ms) y además hay más de quince tramas de voz (más de 240 ms), se considera que se ha llegado al fin de la palabra. Los parámetros anteriores dependen de la duración de las palabras y del tipo de fonemas que las constituyen. Como antes de detectar el fin de la palabra se almacenaron diez tramas de silencio, es necesario eliminar estas tramas.

  Si existen por lo menos diez tramas de silencio consecutivas y hay menos de quince tramas de voz, se considera que la señal almacenada hasta ese momento es ruido, por lo que se elimina de la memoria.

  - *Calcular magnitud y cruces por cero*. Este proceso recibe la trama del estado anterior y le determina la magnitud y los cruces por cero.
  - *Aplicar ventana de Hamming*. Si la trama pertenece a una palabra, se le aplica una ventana de tipo Hamming.
  - *Guardar trama*. Con este proceso, cada trama que pertenece a una palabra, se almacena consecutivamente en memoria.

- *Limpiar buffer* Cuando se detecta que una trama no pertenece a la palabra, utilizando el criterio descrito este proceso se encarga de eliminar las tramas indeseables de la memoria
- *Deteccion de fin de palabra* Si una trama se identifica como la trama final de una palabra este proceso se encarga de detener el programa

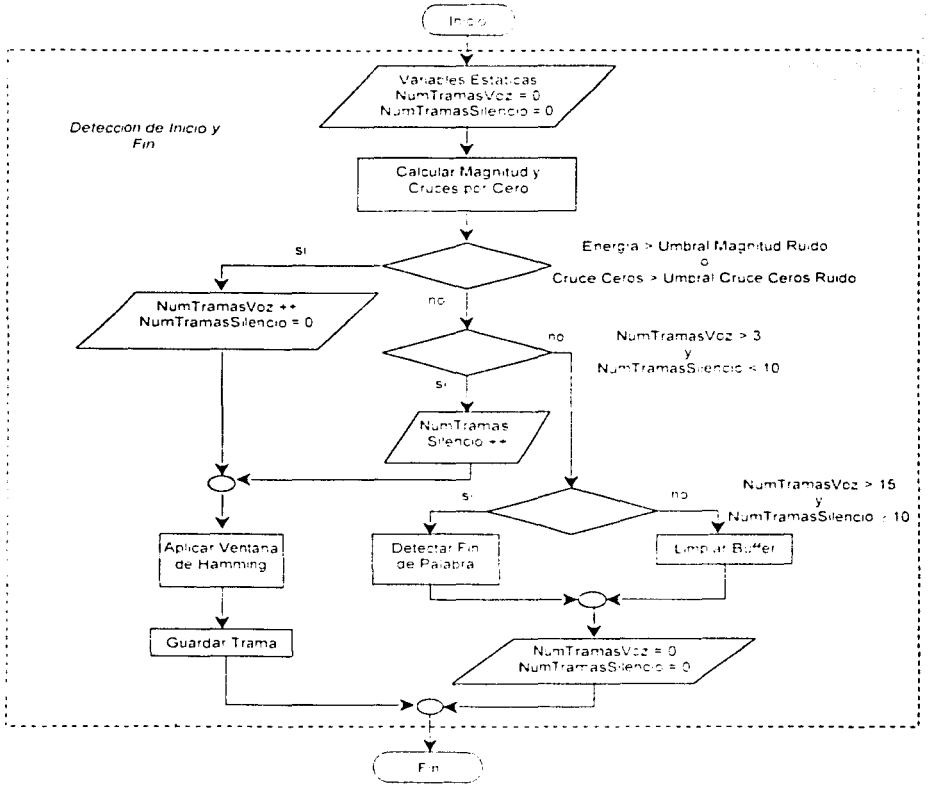

Figura 4.8 Diagrama de flujo para el estado deteccion de inicio y fin

Mientras no se detecta el final de la palabra el proceso cambia al estado anterior, para solicitar otra trama En caso contrario termina el programa En este momento, se puede leer la palabra de entrenamiento, en la memoria del DSP A partir de aqui el entrenamiento continua utilizando programas elaborados con el paquete *Matlab*

La palabra almacenada en el DSP se guarda en un archivo con formato ASCII. Lo mismo se hace con todas las repeticiones obteniendo veinte archivos con las señales de voz recortadas (delimitada por la aplicación de la variante del algoritmo de *Rabiner – Sambur* descrito previamente).

Una vez obtenidas las palabras del entrenamiento, los siguientes procesos se ejecutan en la PC. Estos procesos son listados enseguida: *tramado, autocorrelación, análisis LPC, segmentación y cuantización vectorial*, como se muestran en la figura 4.2. Para ejecutarlos se desarrollaron funciones independientes en *Matlab*, se llamaron mediante un programa principal. Abajo se describen con mas detalle estos procesos.

- ***Tramado*** En este proceso, cada archivo se divide en tramas de 128 valores, sin *traslape*. El traslape consiste en tomar las últimas muestras de una trama, como las primeras muestras de la trama siguiente, es decir, se hace una intersección de tramas.

- ***Autocorrelación*** Este proceso calcula la autocorrelación para cada trama. Almacena sólo los primeros $P + 1$ valores (donde $P = 8$ e indica el orden del *predictor*).

- ***Análisis LPC*** Con los valores de la autocorrelación, se calculan los coeficientes LPC's mediante el algoritmo de *Levinson – Durbin*. Este algoritmo se describió en la sección 1.3 Parametrización de la voz.

- ***Segmentación*** Con los vectores LPC's de cada repetición (veinte repeticiones por palabra), se realizó la segmentación. Se dividió cada repetición en cuatro segmentos en el tiempo, de forma lineal. La figura 4.9, muestra la forma como se realiza la segmentación lineal, para los vectores LPC's obtenidos con el análisis de los coeficientes de predicción lineal.

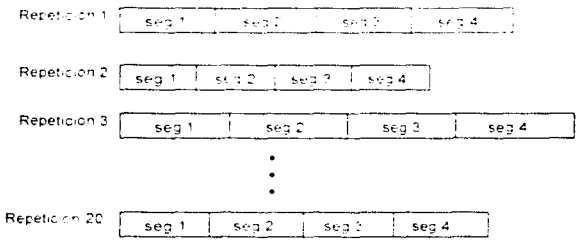

Figura 4.9 Segmentación lineal para las 20 repeticiones por palabra.

- ***Cuantización vectorial*** Cada segmento obtenido anteriormente, por cada repetición, se concatena con el número de segmento correspondiente, de las demas repeticiones. De cada uno de los cuatro grupos de segmentos concatenados, se obtienen sus centroides representativos (*libro de codigos*), utilizando el algoritmo de *K-medias* (descrito en la sección 1.5 Cuantización vectorial). Estos centroides se utilizan durante la fase del reconocimiento. La figura 4.10 esquematiza la forma de obtener los libros de codigos (*codebooks*).

  Se utilizaron los siguientes parametros para el algoritmo de *K-medias*: K = 16 y 32 (donde K es el numero de centroides obtenidos por cada libro de codigos), y obtención de centroides iniciales de forma aleatoria.

Los centroides se almacenan en archivos independientes (un archivo por cada una de las ocho palabras que reconocera el sistema). Estos centroides forman el patron de comparación necesario para lograr la identificacion de las palabras durante el reconocimiento. Con el almacenamiento de los centroides, termina la fase de entrenamiento.

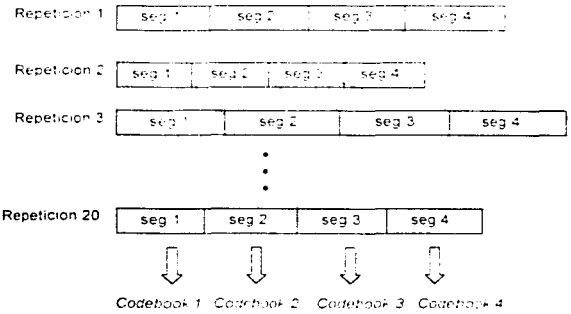

Figura 4.10 Obtencion de los libros de codigos utilizando el algoritmo de K-medias

4.4 Fase de reconocimiento

La fase de reconocimiento, se encarga manipular una señal de voz en tiempo real, confrontándola con los patrones obtenidos durante la fase de entrenamiento (*codebooks*). El objetivo de esta fase es obtener los patrones que mas se asemejen a la señal de voz para identificar una palabra.

El diagrama que se muestra en la figura 4.3 describe la secuencia de estados que se ejecutan durante el reconocimiento. Toda el reconocimiento utiliza como plataforma de ejecución el DSP TMS320C6711 de *Texas Instruments*.

Los primeros dos estados ejecutan casi los mismos procesos que se describieron durante la fase de entrenamiento, por lo que ya no es necesario detallarlos. Cada estado presenta solo una variación. En primer lugar, durante el estado de iniciación y calculo de umbrales, se almacenan en la memoria del DSP los valores correspondientes a los libros de codigos de todas las palabras del diccionario. Y en segundo lugar, el estado de atención al puerto serie, en vez de cambiar al estado de inicio y fin, ahora cambia al estado de obtención de los coeficientes LPC's, aunque en realidad, ejecuta los mismos procesos.

A continuación se describen los estados restantes del diagrama mostrado en la figura 4.3.

- ***Obtencion de LPC's.*** La figura 4.11 esquematiza el diagrama de flujo que se ejecuta en el estado de obtención de LPC's.

  Comparando las figuras 4.7 y 4.11, se observa que los diagramas de flujo son muy similares. En la figura 4.11, se agregan dos nuevos procesos y se modifican otros dos. Debido a esta similitud, solo se enfatiza en los cambios, por lo que se describen los nuevos procesos y las modificaciones, entendiendose que los procesos inalterados, son identicos.

Los nuevos procesos son: *autocorrelación* y *análisis LPC*. Las modificaciones son: el proceso de *guardar trama* cambia a *almacenar* y el proceso *detectar fin de palabra*, ahora es *activar interrupción de software*. Enseguida se describen sólo estos cuatro procesos.

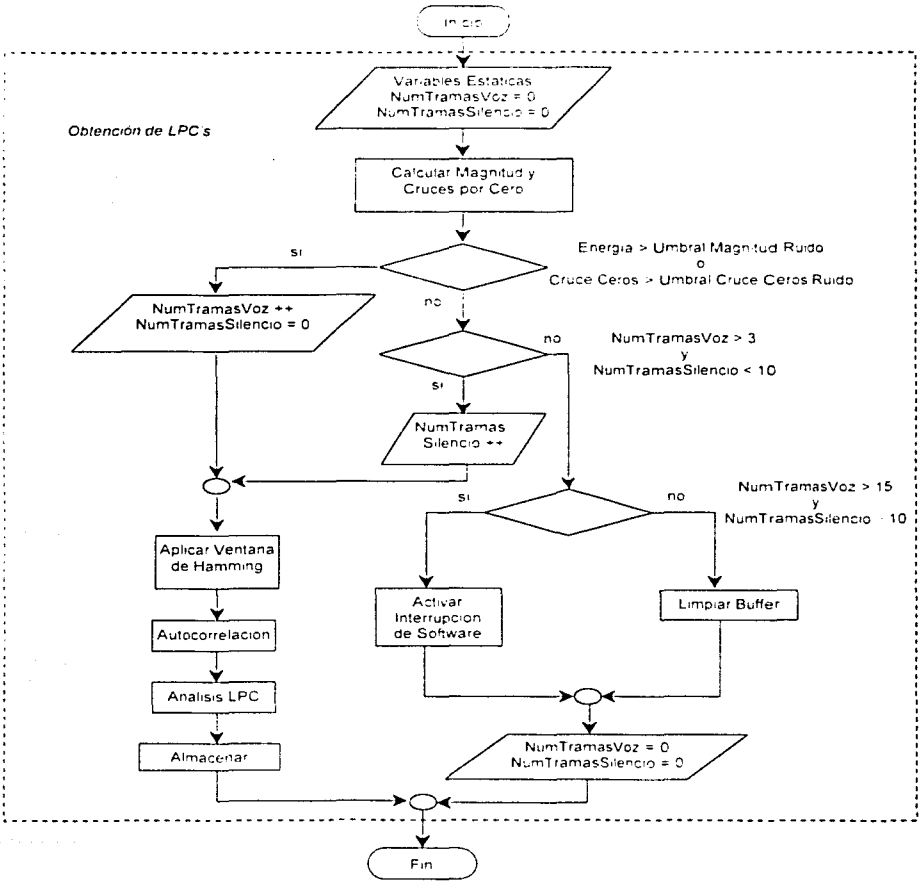

figura 4.11 Diagrama de flujo para el estado de obtencion de LPC's

- *Autocorrelación*. Una vez que el algoritmo detecta la trama como parte de una palabra y que le aplica el ventaneo a dicha trama, se obtiene su autocorrelación. Los primeros nueve valores calculados son utilizados por el siguiente proceso.

- *Análisis LPC* Con los valores provenientes de la autocorrelación se obtienen ocho coeficientes de predicción lineal para la trama utilizando el algoritmo *Levinson – Durbin*, de orden ocho
- *Almacenar* Los primeros nueve valores de la autocorrelación y los ocho coeficientes de predicción lineal se almacenan en memoria secuencialmente para ser utilizados posteriormente
- *Activar interrupción de software* Si la trama se identifica como el fin de una palabra este proceso se encarga de activar al siguiente estado *reconocimiento* mediante una interrupción de software

Cuando el estado *obtención de LPC s* detecta el fin de una palabra pasa inmediatamente al estado *reconocimiento* En caso contrario regresa al estado de *atención al puerto serie*

- ***Reconocimiento*** La figura 4.12 muestra el diagrama de flujo para los procesos que se ejecutan durante el estado de reconocimiento Se emplean ciclos *for* anidados El primero se realiza para cada una de las ocho palabras manejadas en el diccionario El segundo ciclo se ejecuta por cada uno de los cuatro segmentos en que se divide cada palabra El tercer ciclo se realiza por cada uno de los vectores LPC que se obtienen del segmento El último ciclo calcula todas las distancias mínimas existentes entre los coeficientes LPC s de la palabra a reconocer y los centroides de comparación (16 o 32 dependiendo del algoritmo)

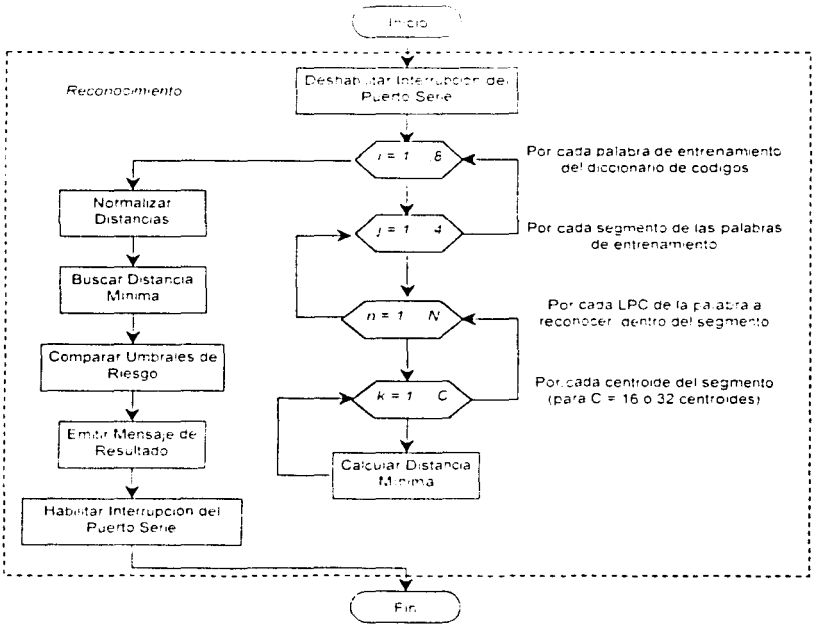

Figura 4.12 Diagrama de flujo para el estado de reconocimiento

Al detectarse el final de una palabra, se activa el estado de reconocimiento por medio de una de una interrupción de software. En este momento, en memoria se encuentran los coeficientes LPC's y los nueve primeros valores por trama, de la autocorrelación, para la palabra a reconocer.

Además, se tiene los conjuntos de centroides de comparacion como valores constantes. Este algoritmo, encuentra la distancia mínima existente entre una palabra capturada, en tiempo real, y un grupo de patrones de reconocimiento. Por medio de esta distancia determina el éxito o fracaso del reconocimiento.

El algoritmo comienza deshabilitando la interrupcion del puerto serie. Realiza la comparacion de todas las distancias (utilizando distancia de Itakura), entre los coeficientes LPC s de la palabra a reconocer y los centroides del diccionario de codigos. Cada distancia mínima, con respecto a cada palabra de entrenamiento, se normaliza. Con las distancias anteriores, se encuentra la distancia mínima global. Posteriormente, se compara esta distancia con ciertos umbrales de riesgo y se emite un mensaje del reconocimiento. Finalmente, se habilita la interrupcion de hardware para regresar al estado de atencion al puerto serie. Con esto ultimo procedimiento, se permite reconocer una nueva palabra.

En consecuencia, los procesos que intervienen en el estado de reconocimiento, son los siguientes: deshabilitar interrupcion del puerto serie, calcular distancia mínima, normalizar distancias, buscar distancia mínima, comparar umbrales de riesgo, emitir mensaje de resultado y habilitar interrupcion del puerto serie. A continuacion se incluye una descripcion de estos procesos.

- *Deshabilitar interrupción del puerto serie.* Este proceso desactiva la interrupcion de hardware del puerto serie, con el fin de evitar interrupciones innecesarias.
- *Calcular distancia mínima.* Con la ejecucion reiterada de este proceso, se encuentran las menores distancias de Itakura, entre los coeficientes LPC s y los centroides de un segmento. Estas distancias se acumulan. Se obtiene una distancia total sumando las distancias de los segmentos. Al final, existe un conjunto de distancias mínimas totales, que corresponden a la distancia existente entre la palabra a reconocer y cada palabra del diccionario de codigos. La distancia mínima total, representa la distorsion existente entre dos palabras.
- *Normalizar distancias.* La distancia mínima total calculada, depende del numero de coeficientes LPC s que tiene la palabra a reconocer. Este hecho, afecta la comparacion con los límites de tolerancia, por lo que es necesario normalizar. El proceso de normalizacion, ejecuta la division entre la distancia mínima total y el numero de coeficientes LPC s de la palabra.
- *Buscar distancia mínima.* Este proceso localiza la distancia menor entre las diferentes distancias mínimas totales normalizadas e identifica la palabra a la cual pertenece.
- *Comparar umbrales de riesgo.* Los umbrales de riesgo, son fronteras determinadas de manera empírica, mediante el empleo de ciertas caracteristicas estadísticas, de las palabras. Para ello, se utilizaron veinte repeticiones por locutor y por palabra, es decir, se emplearon (para ocho palabras y dos locutores), 320 repeticiones totales.

  De cada repeticion, se obtuvo la distancia mínima de la palabra reconocida y la distancia próxima, que pertenece a la palabra que mas se asemeja a la anterior. A las cuarenta repeticiones (para ambos locutores y por cada palabra), se les calculo la distancia mínima promedio, la distancia próxima promedio, la diferencia entre ambas distancias y sus desviaciones estandar. Con estos valores y por observacion directa de las graficas correspondientes, se determinaron tres criterios de discriminacion. Estos criterios son:

- Frontera máxima Especifica el valor más alto permitido para una distancia minima Se obtiene por observacion y comparacion de las distancias minimas en todas las palabras

  En el conjunto de repeticiones se detecto que las minimas no sobrepasan cierto valor Para 16 centroides el valor de la frontera es de 0 43 Para 32 centroides la frontera tiene un valor de 0 38 Cuando una distancia minima cualquiera exceda esta frontera, sera etiquetada automaticamente como *palabra desconocida*

- Frontera superior Determina un valor por cada palabra Este valor delimita las distancias minimas permisibles en cada caso Se obtiene sumando dos veces la desviación estandar con la distancia promedio minima en cada palabra Este criterio se considera como no suficiente para establecer el reconocimiento de una palabra por lo que se apoya en el tercer criterio

- Frontera minima Debido a que en ciertos casos existe ambiguedad en el reconocimiento (cuando la distancia minima esta muy cercana a la distancia próxima), se creo un tercer criterio de discriminacion en apoyo al anterior Este nuevo criterio, representa el valor minimo permisible entre la distancia minima y la distancia próxima Se obtuvo mediante la observacion y comparacion de las diferencias entre las distancias minimas y proximas

  La forma como se combina con el criterio de la frontera superior es la siguiente cuando la diferencia entre las distancias minima y proxima es menor a la frontera minima, el algoritmo de reconocimiento detecta ambiguedad dado que no existe la certeza de que la distancia minima pertenezca a la palabra correcta Si la diferencia entre ambas distancias es mayor a la frontera minima y ademas la distancia minima no excede a la frontera superior el algoritmo etiqueta la palabra como reconocida

  En el caso de que la frontera minima exceda a la frontera superior el algoritmo calcula la diferencia entre la distancia minima y el valor promedio de la palabra correspondiente Ademas calcula la diferencia entre la distancia proxima y el valor promedio de su palabra respectiva Cuando la primer diferencia es menor se etiqueta la palabra como reconocida En cualquier otro caso se le solicita al locutor la repeticion de la palabra

En la figura 4 13 se muestran las ubicaciones de las fronteras las distancias minimas y las distancias proximas para 40 repeticiones de la palabra "adelante" utilizando 16 centroides

- *Emitir mensaje de resultado* Este proceso envia un mensaje al usuario con el tipo de resultado que se obtuvo del reconocimiento
- *Habilitar interrupcion del puerto serie* Una vez que se emite el mensaje de resultado este proceso activa de nueva cuenta la interrupción por hardware del puerto serie que se desactivo al inicio del reconocimiento Con lo anterior, el sistema regresa a esperar una nueva palabra para reconocer

El codigo fuente desarrollado durante la implantación del sistema reconocedor de palabras aisladas con un DSP se encuentra en la siguiente direccion electronica

http://grupos.msn.com/Procesamientodevoz

Dicho codigo consiste de dos partes en lenguaje C para el DSP TMS320C6711 de *Texas Instruments* y en lenguaje propio del paquete *Matlab* para la simulacion del reconocimiento

En el capítulo final se muestran algunas gráficas y estadísticas con la presentación de los resultados obtenidos. Además, se describen las conclusiones extraídas de este trabajo.

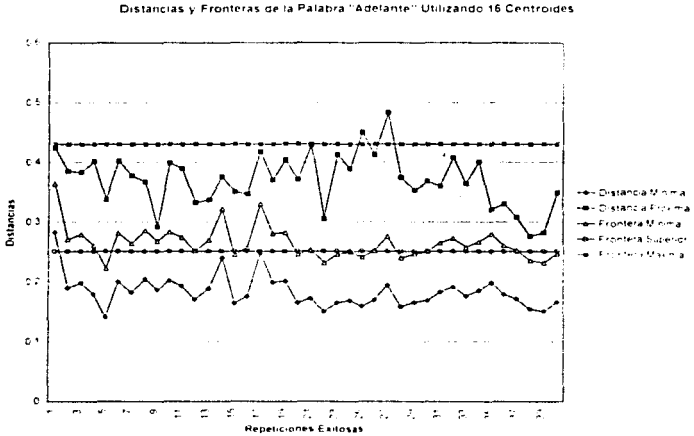

Figura 4.13 Representación de las distancias y fronteras para la palabra ADELANTE

CAPÍTULO 5

RESULTADOS Y CONCLUSIONES

5.1 Reconocimiento

El sistema de reconocimiento de palabras aisladas utilizando un DSP, se diseñó particularmente, para el reconocimiento de 8 palabras distintas con dos locutores diferentes

Se emplearon, durante la etapa de entrenamiento 20 repeticiones de cada palabra (10 repeticiones por locutor) Con estas repeticiones se obtuvieron dos diferentes grupos de patrones de comparación El primero de ellos utiliza 16 centroides por segmento y mientras que el segundo, emplea 32 centroides La segmentación es de tipo lineal y divide la duración de cada palabra (recortada) en cuatro partes iguales Los centroides se obtuvieron mediante el empleo del algoritmo de *K – medias* Se utilizo como medida de distorsión la distancia de Itakura

Durante el reconocimiento se realizaron dos tipos de pruebas La primera utilizando patrones de 16 centroides y la segunda con los patrones de 32 centroides, por segmento Para cada tipo de prueba se ejecutaron 25 repeticiones por palabra y por locutor

Los resultados obtenidos se presentan a continuación

- **Con patrones de 16 centroides por segmento**

En las siguientes tablas se hace una presentación de los resultados finales utilizando 16 centroides por segmento Se ejecutaron 25 repeticiones para cada palabra Los datos incluidos en estas tablas se explican a continuación

La columna *Palabras* hace referencia a las palabras del diccionario de palabras, pronunciadas por cada locutor La columna *Reconocidas*, indica los eventos exitosos del reconocimiento

La columna *Reconocidas error* contiene los eventos en los cuales el reconocimiento se alcanza de manera errónea esto es cuando se reconoce una palabra diferente a la que se ha pronunciado

Finalmente la columna *No reconocida* indica aquellos eventos cuyo reconocimiento no ha sido posible en otras palabras cuando existen distancias muy alejadas de los núcleos aceptados para las palabras o cuando se presentan estados en los cuales no se sabe con certeza si realmente la palabra reconocida es la correcta (ambigüedad)

También se hace una presentación gráfica de los mismos valores para proporcionar una mejor visualización de los resultados En las gráficas se indican los porcentajes de reconocimiento alcanzados para cada palabra

En la tabla 5 1 se presentan los resultados obtenidos para el locutor Víctor

Tabla 5 1 Resultados obtenidos por el locutor Victor para 16 centroides

Palabras	*Reconocidas*	*Reconocida error*	*No reconocida*
Alto	25	0	0
Sigue	25	0	0
Izquierda	25	0	0
Derecha	25	0	0
Arriba	25	0	0
Abajo	25	0	0
Adelante	24	0	1
Atrás	24	0	1
Totales	**198**	**0**	**2**
Porcentajes	**99%**	**0%**	**1%**

La gráfica 5 1 ilustra los porcentajes de reconocimiento alcanzados para cada una de las ocho palabras del diccionario por el locutor Victor

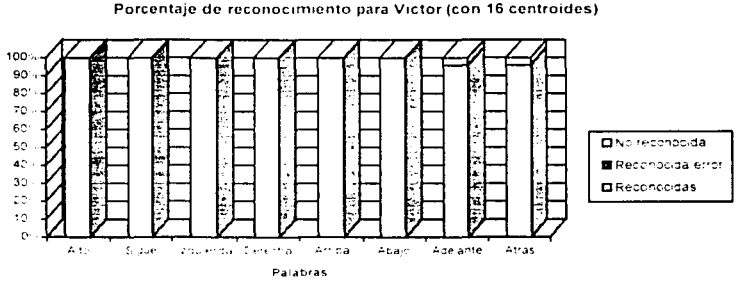

Figura 5 1 Porcentajes de reconocimiento por palabras para el locutor Victor

En la tabla 5 2 se muestran los resultados alcanzados por el locutor Omar

Tabla 5 2 Resultados obtenidos por el locutor Omar para 16 centroides

Palabras	*Reconocidas*	*Reconocida error*	*No reconocida*
Alto	25	0	0
Sigue	20	0	5
Izquierda	25	0	0
Derecha	25	0	0
Arriba	25	0	0
Abajo	25	0	0
Adelante	25	0	0
Atras	25	0	0
Totales	**195**	**0**	**5**
Porcentajes	**97.5%**	**0%**	**2.5%**

La gráfica 5 2 muestra los porcentajes de reconocimiento por cada palabra, obtenidos por el locutor Omar.

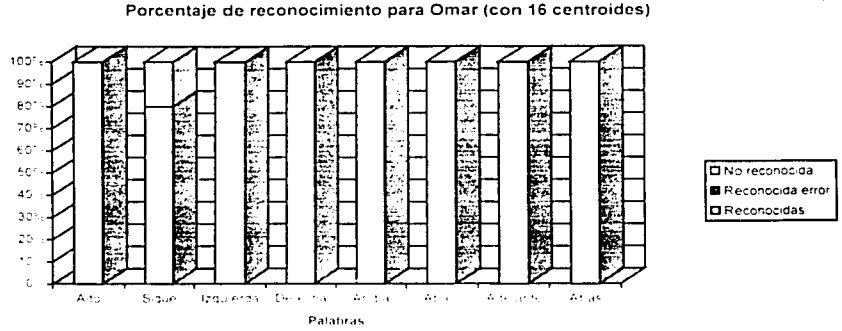

Figura 5 2 Porcentajes de reconocimiento por palabras para el locutor Omar

En la tabla 5 3 se reúnen los resultados obtenidos por ambos locutores, para presentar el resultado final del reconocimiento, utilizando 16 centroides por segmento

Tabla 5 3 Resultados obtenidos por ambos locutores con 50 repeticiones de cada palabra y 16 centroides

Palabras	*Reconocidas*	*Reconocida error*	*No reconocida*
Alto	50	0	0
Sigue	45	0	5
Izquierda	50	0	0
Derecha	50	0	0
Arriba	50	0	0
Abajo	50	0	0
Adelante	49	0	1
Atras	49	0	1
Totales	**393**	**0**	**7**
Porcentajes	**98.25%**	**0%**	**1.75%**

Enseguida, la gráfica 5 3 ilustra los porcentajes de reconocimiento, obtenidos por ambos locutores, para cada palabra

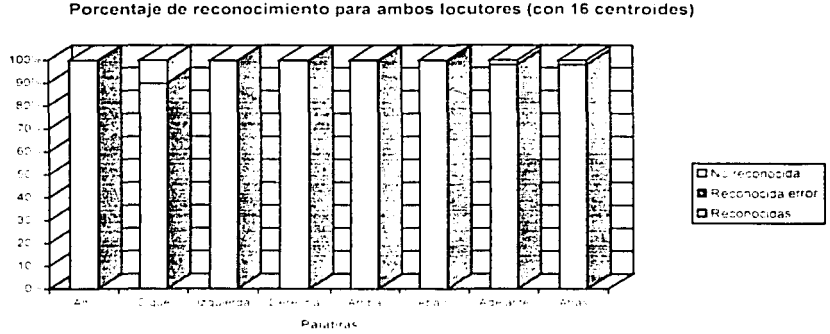

Figura 5 3 Porcentajes de reconocimiento por palabras para ambos locutores , con 16 centroides

La gráfica 5 4 presenta el porcentaje global de reconocimiento, para todas las palabras del diccionario de codigos Como se puede observar, se alcanzo un 98% de reconocimiento global Ademas se obtuvo un 2% de palabras no reconocidas y ningún caso de palabras que se reconocieran erroneamente Estos resultados, enfatizando, utilizando 16 centroides por segmento El número total de eventos ejecutados fue de 400 repeticiones

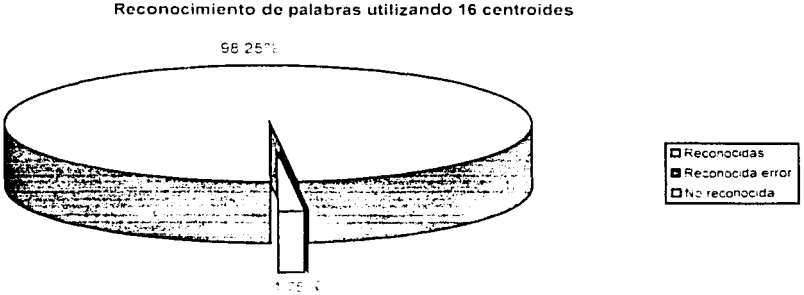

Figura 5 4 Porcentaje global de reconocimiento para ambos locutores

Reconocimiento utilizando patrones de 32 centroides por segmento

Las tablas 5.4, 5.5 y 5.6, presentan los resultados obtenidos, ahora para 32 centroides por segmento. De la misma forma, las gráficas 5.5, 5.6, 5.7, 5.8, muestran los porcentajes de reconocimiento alcanzados en cada caso.

La tabla 5.4, muestra los resultados obtenidos por el locutor Víctor.

Tabla 5.4 Resultados obtenidos por el locutor Víctor para 32 centroides

Palabras	*Reconocidas*	*Reconocida error*	*No reconocida*
Alto	24	0	1
Sigue	25	0	0
Izquierda	24	0	1
Derecha	24	0	1
Arriba	23	0	2
Abajo	25	0	0
Adelante	24	0	1
Atrás	24	1*	0
Totales	**193**	**1**	**6**
Porcentajes	**96.5%**	**0.5%**	**3%**

* La palabra que reconoció erróneamente para este caso fue *abajo* en vez de *atrás*

La gráfica 5.5, ilustra los porcentajes de reconocimiento alcanzados por el locutor Víctor.

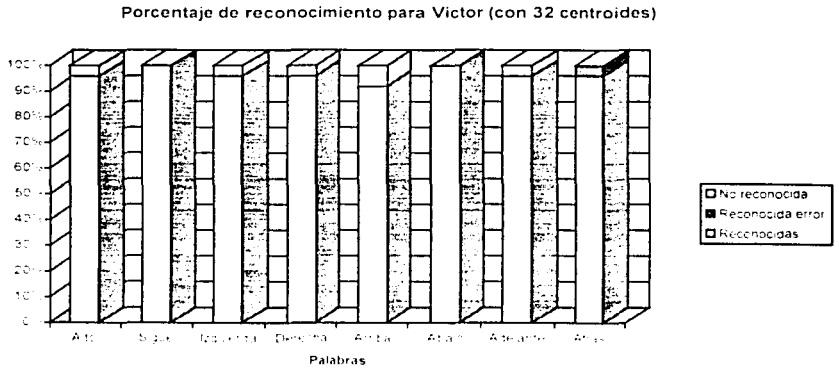

Figura 5.5 Porcentajes de reconocimiento por palabras para el locutor Víctor

En la tabla 5.5, se muestran los resultados conseguidos por el locutor Omar

Tabla 5 5 Resultados obtenidos por el locutor Omar para 32 centroides

Palabras	*Reconocidas*	*Reconocida error*	*No reconocida*
Alto	25	0	0
Sigue	23	1	1
Izquierda	24	0	1
Derecha	25	0	0
Arriba	25	0	0
Abajo	24	0	1
Adelante	23	0	2
Atrás	25	0	0
Totales	**194**	**1**	**5**
Porcentajes	**97%**	**0.5%**	**2.5%**

La gráfica 5 6 muestra el porcentaje de reconocimiento por cada palabra, que obtuvo el locutor Omar

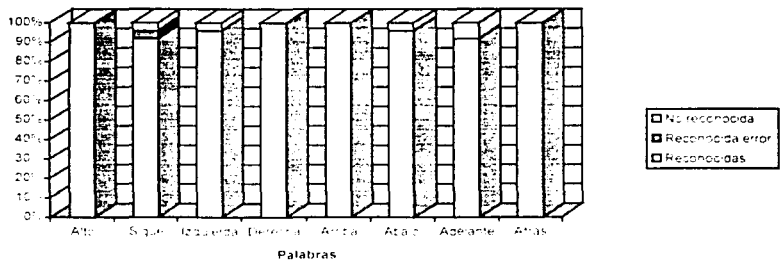

Figura 5 6 Porcentajes de reconocimiento por palabras para el locutor Omar

En la tabla 5 6 se reunen los resultados obtenidos para ambos locutores

Tabla 5 6 Resultados obtenidos por ambos locutores con 32 centroides

Palabras	*Reconocidas*	*Reconocida error*	*No reconocida*
Alto	49	0	1
Sigue	48	1	1
Izquierda	48	0	2
Derecha	49	0	1
Arriba	48	0	2
Abajo	49	0	1
Adelante	47	0	3
Atrás	49	1	0
Totales	**387**	**2**	**11**
Porcentajes	**96.75%**	**0.5%**	**2.75%**

La gráfica 5.7, proporciona los porcentajes de reconocimiento logrados por ambos locutores, utilizando 32 centroides por segmento

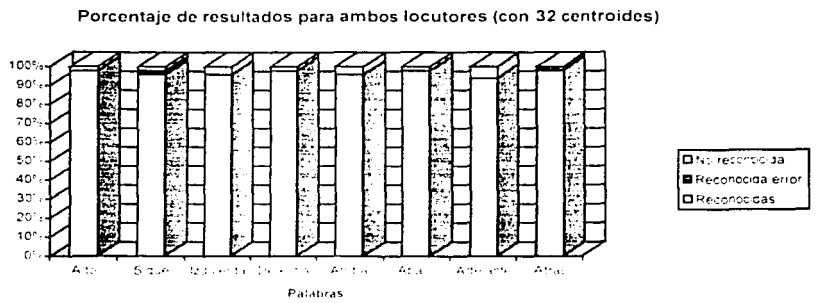

Figura 5.7 Porcentajes de reconocimiento por palabras para ambos locutores

Finalmente, la gráfica 5.8 muestra el resultado global del reconocimiento, para ambos locutores, con 32 centroides por segmento. Como se observa en la gráfica, se alcanzó un 97% de reconocimiento exitoso global. Ademas, se obtuvo un 3% de palabras no reconocidas y en este caso existió un 1% de palabras que se reconocieron erróneamente. De forma similar al caso de 16 centroides, se realizaron 400 repeticiones

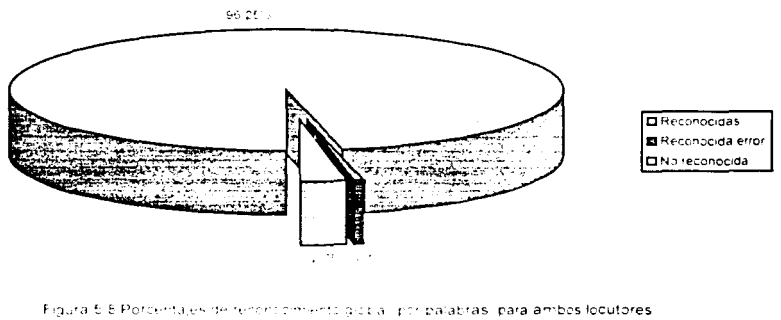

Figura 5.8 Porcentajes de reconocimiento global por palabras para ambos locutores

Tiempos de ejecución

Una caracteristica importante en los sistemas de reconocimiento, es determinar el tiempo que se tardan en efectuar el reconocimiento. El *tiempo de reconocimiento* es el intervalo transcurrido desde el termino de la pronunciacion de una palabra hasta la obtencion de un resultado. Dado lo anterior, para este sistema en particular, el tiempo de reconocimiento es equivalente al tiempo transcurrido en las comparaciones (patrones *vs* LPC s de la palabra

pronunciada). En consecuencia, este tiempo no engloba la pronunciación de la palabra, ni la emisión del mensaje de resultado.
Para el sistema de reconocimiento de palabras aisladas, en tiempo real, desarrollado en el DSP TMS320C6711 de *Texas Instruments*, se obtuvieron los ciclos de reloj, especificados para cada uno de los principales procesos involucrados. Estos procesos son: *Captura de la señal*, *Cálculo de la magnitud promedio y cruces por cero*, *Obtención de coeficientes LPC's por tramas*, *Reconocimiento con 16 centroides* y *Reconocimiento con 32 centroides*. Para los últimos dos procesos, se omitió la parte de emisión de mensajes, dado que es una actividad muy relativa, en otras palabras, el mensaje emitido dependerá de la aplicación específica en la que se involucre el sistema.

A continuación, se muestran las tablas con los tiempos de ejecución. El *tiempo aproximado* se obtiene considerando la frecuencia a la que trabaja el DSP y los ciclos de reloj que consume cada proceso. En particular, el DSP TMS320C6711 de *Texas Instruments*, trabaja a una frecuencia de 150 MHz, esto es, en un segundo se ejecutan 150 millones de ciclos de reloj. Además, es importante tener en cuenta que debido a la arquitectura del DSP, durante un ciclo determinado de reloj, se pueden estar ejecutando hasta ocho instrucciones en paralelo.

La tabla 5.7 muestra la duración aproximada (para cada trama) de los procesos más importantes involucrados, desde la captura de la señal hasta el cálculo de los coeficientes LPC's. Contiene las siguientes columnas: *Nombre del proceso*, *Ciclos de reloj* y *Tiempo aproximado* para la duración del proceso. Los valores obtenidos son calculados para una sola trama. Hay que tener en cuenta que para una palabra, existe un número variable de tramas, dependiendo de su duración.

Tabla 5.7 Duración aproximada de los procesos más importantes

Nombre del proceso	***Ciclos de reloj*** *(por trama)*	***Tiempo aproximado*** *(por trama)*
Captura de la señal	150 272	1.002 mseg
Cálculo de la magnitud promedio y cruces por ceros	1 178	7.8533 μseg
Obtención de LPC s por tramas	7,862	52.4133 μseg

Si la frecuencia de muestreo del convertidor A/D es de 8 KHz y se definió cada trama como un conjunto de 128 muestras, entonces:

 # tramas en un segundo = 8000 [muestras/seg] ÷ 128 [muestras/trama]
 # tramas en un segundo = 62.5 [tramas/seg]

Esto es, en un segundo se leen 62.5 tramas. Mediante observación directa, se detectó que la mayor parte de las veces, la duración de las palabras del diccionario de códigos, es menor a un segundo.

Tomando en consideración los valores de la tabla 5.7, se tiene que el tiempo que tarda el DSP, en obtener los coeficientes LPC s, para una palabra que dura un segundo, es:

Tiempo LPC = (1.002 + 0.0078533 + 0.0524133)[mseg/tramas] x (62.5)[tramas]
Tiempo LPC = 66.392 [mseg]

Por consiguiente, el DSP tarda aproximadamente 0 066392 segundos en ejecutar los cálculos necesarios para obtener los coeficientes LPC's de una palabra que dura un segundo Estos cálculos los ejecuta al mismo tiempo que realiza la captura de la señal de voz

Por otra parte, para el proceso de comparacion se detiene la captura de datos En consecuencia, el DSP se dedica exclusivamente a realizar este proceso

La tabla 5 8 permite obtener un valor promedio de los ciclos de reloj que consume el proceso de reconocimiento de una sola trama para el caso de 16 centroides Muestra el numero de tramas que existio en la pronunciacion de cada palabra el numero de ciclos de reloj que tarda el DSP en realizar la comparacion y el numero de ciclos de reloj que se ejecutan en cada trama (este dato se obtuvo, dividiendo el número de total de ciclos entre la cantidad de tramas presentes en la palabra) para ambos locutores, utilizando 16 centroides Se ejecutaron 16 eventos para obtener un promedio general del numero de ciclos de reloj involucrados en la comparacion de una trama

Tabla 5 8 Calculo de ciclos promedio por trama para todas las palabras del diccionario de datos ambos locutores y 16 centroides

	Palabra	Tramas	Ciclos	Ciclos/Tramas
Locutor Victor	alto	30	4569484	152316
	sigue	31	4721542	152308
	izquierda	40	6090140	152254
	derecha	37	5633965	152269
	arriba	43	6546326	152240
	abajo	53	8067247	152212
	adelante	88	13389773	152157
	atrás	56	8523704	152209
	Promedio			**152246**

	Palabra	Tramas	Ciclos	Ciclos/Tramas
Locutor Omar	alto	29	4417432	152325
	sigue	26	3961185	152353
	izquierda	37	5633989	152270
	derecha	42	6394305	152245
	arriba	41	6242209	152249
	abajo	31	4721485	152306
	adelante	42	6394272	152245
	atrás	41	6242218	152249
	Promedio			**152280**

Promedio global de ciclos para cada trama y 16 centroides	**152262**

Utilizando el promedio global de ciclos para cada trama de la tabla 5 8 y considerando las 62 5 tramas por segundo tenemos el siguiente tiempo estimado de reconocimiento para una palabra que dura un segundo

Tiempo reconocimiento estimado = (152262[ciclos/trama] x 62 5 [tramas]) ÷ 150[MHz]
Tiempo reconocimiento estimado = 0 0634425 [seg]

Esto es, el DSP tarda aproximadamente 0 0634425 segundos, adicionales a la captura de la señal, en realizar la comparación de una palabra que dura un segundo, con 16 centroides de comparacion

La tabla 5 9 muestra los datos que se ocupan para obtener el promedio global de ciclos de reloj por trama, para el caso de 32 centroides de comparación

Tabla 5 9 Calculo de ciclos promedio por trama para todas las palabras del diccionario de datos ambos locutores y 32 centroides

	Palabra	Tramas	Ciclos	Ciclos/Tramas
Locutor Victor	alto	34	10248034	301413
	sigue	30	9043930	301464
	izquierda	57	17171079	301247
	derecha	44	13257780	301313
	arriba	43	12957016	301326
	abajo	43	12956878	301323
	adelante	58	17472328	301247
	atrás	43	12957040	301327
	Promedio			**301332**

	Palabra	Tramas	Ciclos	Ciclos/Tramas
Locutor Omar	alto	28	8441955	301498
	sigue	23	6937031	301610
	izquierda	39	11753035	301360
	derecha	36	10849993	301389
	arriba	40	12054077	301352
	abajo	36	10850080	301391
	adelante	41	12354895	301339
	atrás	33	9946819	301419
	Promedio			**301420**

Promedio global de ciclos para cada trama y 16 centroides	**301374**

Con el promedio global de ciclos de reloj por trama obtenido en la tabla 5 9 tenemos el siguiente tiempo estimado de reconocimiento para una palabra que dura un segundo

Tiempo reconocimiento estimado = (301374[ciclos/trama] x 62 5 [tramas]) ÷ 150[MHz]
Tiempo reconocimiento estimado = 0 1255725 [seg]

Es decir el DSP tarda aproximadamente 0 1255725 segundos, en realizar la comparación de una palabra que dura un segundo utilizando 32 centroides de comparacion

Comparando los tiempos de reconocimiento estimados para ambos casos, se observa que el tiempo para el caso de 32 centroides es casi el doble del tiempo obtenido para 16 centroides.

Es necesario enfatizar que los tiempos aproximados no son estrictamente exactos, dado que no se consideran algunas funciones que utiliza el DSP/BIOS (por ejemplo, almacenamiento en la pila y llamadas a funciones API's para manejo de interrupciones) ni la emisión de mensajes de resultados.

- **Memoria de datos utilizada**

Cada dato se almacena en localidades de 32 bits (4 bytes). Para calcular la memoria ocupada por los patrones se deben tener en cuenta las siguientes consideraciones: orden de los coeficientes LPC's (o centroides), número de segmentos por palabra, número de centroides por segmento y número de palabras del diccionario de códigos. En este trabajo, el orden de los coeficientes LPC's es de 8, el número de centroides es de 16 ó 32, el numero de segmentos es de 4 y el número de palabras es de 8.

Memoria patrones = 4 [bytes] x (orden LPC) x (# centroides) x (# segmentos) x (# palabras)

Memoria patrones 16 = 4 [bytes] x 8 x 16 x 4 x 8 = 16384 [bytes]

Memoria patrones 32 = 4 [bytes] x 8 x 32 x 4 x 8 = 32768 [bytes]

Para guardar los parámetros de la señal de entrada se utilizan dos matrices de dimensiones fijas (una para almacenar coeficientes LPC's y otra para la autocorrelación). En ambas matrices se pueden salvar hasta dos segundos. La memoria ocupada por estas matrices es de:

Memoria parametros =4 [bytes] x (orden LPC) x (128 tramas) x (2 matrices)

Memoria parametros = 4 x 8 x 128 x 2 = 8192 [bytes]

Además, se utilizan 4096 bytes de memoria para el almacenamiento temporal de la señal.

La memoria de datos total es aproximadamente la siguiente:

memoria de datos total = memoria patrones + memoria parámetros + memoria temporal

memoria de datos total 16 = 16384 + 8192 + 4096 = 28672 [bytes]

memoria de datos total 32 = 32768 + 8192 + 4096 = 45056 [bytes]

Es necesario aclarar que no se considera la memoria utilizada por el *stack* (memoria que se usa para manejo de variables locales, llamadas a funciones, etc.). Tampoco se toma en cuenta la memoria ocupada por el código de los algoritmos.

5.2 Mejoras

Los algoritmos desarrollados para el sistema fueron creados en lenguaje C. Estos algoritmos, pueden ser optimizados (disminuyendo el tiempo de ejecución), utilizando directamente el lenguaje ensamblador del DSP. Con esto se aumenta el tiempo de programación y la complejidad de los algoritmos.

Otra posible mejora, es agregar traslape en las tramas. En este caso, no se utilizo traslape debido a que se trato de mantener siempre la velocidad del sistema desarrollado. El empleo del traslape disminuye la velocidad de respuesta debido a que aumenta el número de cálculos y la cantidad de memoria utilizada.

Por otra parte, se puede realizar un filtrado del ruido ambiental, para eliminarlo con más eficacia, utilizando dos microfonos. Con el primer micrófono, se detecta el ruido ambiental y con el segundo se captura la señal de voz mas el ruido. El filtrado consiste en la eliminación del ruido ambiental, por medio de la comparación de ambas señales. Sin embargo, se presenta un inconveniente para el desarrollo de este filtrado. Se necesita otro convertidor A/D adicional, para el micrófono extra.

De igual forma, los umbrales de ruido pueden ser calculados de manera dinámica, utilizando los periodos de tiempo en que el DSP, permanece ocioso. Esto es, se puede activar un proceso capaz de detectar esta situación y de efectuar un nuevo cálculo de umbrales del ruido.

Asimismo, es posible mejorar los umbrales de riesgo, aumentando el numero de eventos para el calculo estadístico de cada palabra. Estos calculos estadísticos quedan fuera de los alcances del objetivo fijado para la tesis.

Otra mejora posible, es obtener las palabras de entrenamiento de un solo locutor, para hacer al sistema más específico para una persona.

Además, se puede enriquecer el diccionario de palabras, siempre teniendo como limitante, la capacidad de memoria del DSP. Aumentar las palabras de reconocimiento aumenta el número de comparaciones necesarias y por tanto, disminuye la velocidad de respuesta.

Finalmente, unas mejoras mas radicales consisten en cambiar la técnica del reconocimiento empleada. Una es DTW (alineamiento dinamico en el tiempo), lo que implica disminuir la velocidad de respuesta. Otro posible cambio es utilizar modelos ocultos de *Markov*, teniendo la desventaja de hacer al sistema mas complejo.

5.3 Posibles aplicaciones

Se puede utilizar el sistema de reconocimiento de palabras aisladas, con un DSP, en el manejo de un móvil (por ejemplo, un robot, un carro, una silla de ruedas, etc.). Para abrir una puerta de seguridad, en combinación con algunas otras técnicas, que aumenten el nivel de seguridad. En telefonia móvil. En sistemas interactivos, en los cuales se pueda solicitar información o ejecutar una acción (por ejemplo, el conductor de un vehículo puede comunicarse con el tablero de controles del vehículo, para solicitar el nivel de gasolina o abrir una ventana). En autómatas diseñados para la venta de productos. En el manejo de un elevador. En edificios inteligentes. Y en general, en cualquier aplicación limitada por la imaginación del posible usuario.

5.4 Conclusiones

Los resultados obtenidos, desde un punto de vista particular, fueron excelentes. Se alcanzó un 98.25 % de aciertos en los eventos ejecutados para reconocimiento con 16 centroides por segmento. Además, se obtuvo un 96.25 % de éxito en el reconocimiento con 32 centroides. En consecuencia, no se mejora el nivel de reconocimiento, cuando se aumenta el número de centroides y en cambio, se tiene la desventaja de que aumenta el tiempo de ejecución, a casi el doble de ciclos de reloj.

Durante el desarrollo del sistema se trabajó de forma simultánea con una simulación en el paquete *Matlab*. Los resultados parciales y finales en la simulación, fueron muy parecidos a los alcanzados en la aplicación con el DSP. En este último, se manejaron datos en punto flotante de 32 bits, que no se vieron afectados en demasía por efecto del redondeo, en contraste con el tipo de datos manejados por el paquete Matlab. Por consiguiente, para la aplicación es suficiente con el manejo de datos en punto flotante de 32 bits.

Se mejoró el tiempo de cálculo, mediante el desarrollo de algoritmos capaces de efectuar operaciones conforme se captura la palabra. Gracias a lo cual, al término de la pronunciación de la palabra, solo se ejecutan las comparaciones con los patrones existentes. Por tanto, el tiempo total que tarda el DSP en efectuar el reconocimiento es muy corto y exclusivo.

Los tipos de discriminación implantados para la comparación de patrones (umbrales de riesgo), a pesar de ser obtenidos empíricamente (apoyados en estadísticas de las diversas palabras), son aceptables. Se puede trabajar más con ellos para mejorarlos en un futuro. El trabajo con las estadísticas de las palabras, queda fuera del alcance de esta tesis y por ello no se le invirtió demasiado tiempo a este aspecto.

Por otro lado, existe el inconveniente siguiente: para el caso de palabras distintas a las contenidas en el diccionario de palabras, existe el riesgo latente de reconocimiento erróneo. Esto se debe a que el método utilizado no realiza un seguimiento de la secuencia de fonemas que pertenecen a la palabra. En consecuencia, se debe restringir el número de palabras pronunciadas al vocabulario del diccionario de palabras definido.

Además, los umbrales de ruido son calculados al inicio de la sesión y permanecen constantes durante todo el tiempo que se efectúa el reconocimiento. Debido a esto, un aumento en los niveles de ruido, puede afectar la precisión en el reconocimiento.

APÉNDICE A

DESCRIPCIÓN GENERAL DEL CONVERTIDOR A/D

A.1 Características

El convertidor A/D TLC320AD535 está constituido por dos canales *codec* y un circuito analógico híbrido, con dos puertos seriales independientes para la comunicación con el procesador *host* y con un juego de resistencias y capacitores externos que sirven para determinar la ganancia y los polos del filtro.

Este dispositivo también tiene alimentación de micrófono y amplificación, capacidad mixta de audio en el canal de voz, control programable de ganancia y dos controles para el micrófono.

El dispositivo trabaja con fuentes de poder analógicas de 5 ó 3.3 volts, digitales de 5 ó 3.3 volts y del amplificador monitor de 5 o 3.3 volts. Además, el dispositivo viene empaquetado en un circuito integrado con 64 pines. Posee las siguientes características:

- Fuentes de poder analógicas, digitales y del amplificador monitor, que suministran 5 y 3.3 volts.
- Software independiente, para trabajo en modo de bajo consumo de energía, con los canales de datos y voz.
- Velocidades de muestreo hasta de 11.025 kHz para los canales independientes de voz y datos.
- 16 bits para el procesamiento de señales.
- Intervalos dinámicos de 80 dB, en los canales de voz y datos.
- El total de la relación señal a ruido, más la distorsión, es de 77 dB, para los ADCs y de 74 dB para los DACs.
- Amplificadores de ganancia programables.
- Maneja 600 Ω de resistencia interna para la tarjeta de sonido y el canal de datos.
- Maneja 60 Ω de resistencia interna para los audífonos y los amplificadores de ganancia programable.
- Maneja 8 Ω de resistencia interna para bocinas diferenciales AT41 con amplificador de ganancia programable.
- La máxima alimentación del micrófono es de 5 mA con 2.5 V / 1.5 V.
- La corriente máxima de referencia para el microteléfono es de 2.5 mA a 2.5 V / 1.5 V.
- La corriente máxima de referencia para el canal de datos es de 10 mA a 2.5 V / 1.5 V.
- El voltaje para los circuitos de encendido y reiniciar (*reset*) MV_{DD} es de 5 V / 3.3 V.
- Un circuito habilitador de escritura de destello, para escribir en el dispositivo de memoria de destello.
- Un circuito integrado de 64 pines, PM (QFP), que trabaja dentro de un intervalo de temperaturas comprendido entre los – 40 °C y los 85 °C.

La figura A-1 muestra el diagrama funcional de bloques, para el caso de una aplicación típica utilizando el convertidor.

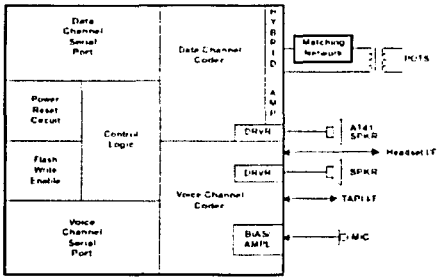

Figura A-1 Diagrama funcional de bloques del convertidor A/D

A.2 Descripción funcional

❑ Requisitos del dispositivo y vista general del sistema

El dispositivo TLC320AD535 contiene dos canales *codecs*, un circuito híbrido con resistencias y capacitores externos que sirven para establecer la ganancia y los polos del filtro, dos puertos serie independientes y otras funciones lógicas misceláneas

❑ Funciones del codec

El *codec* optimiza las funciones necesarias para los dos canales del convertidor (analógico –digital y digital – analógico), el filtrado paso bajas, el control de las ganancias analógicas de entrada y salida, el sobremuestreo interno (conectado con decimación interna e interpolación), y dos interfaces de puerto serial de 16-bits para el procesador *host*. Los puertos seriales son independientes entre sí y son capaces de trabajar con diferentes frecuencias de muestreo. La máxima frecuencia de muestreo para ambos canales es de 11 025KHz

❑ Funciones híbridas

El circuito híbrido en el canal de datos incluye amplificadores integrados cuya ganancia y polos en la frecuencia del filtro se establecen por medio de resistencias y capacitores externos. Esto permite una gran flexibilidad para ajustarse tanto a diversas tarjetas y como a diferentes estándares internacionales, proporcionando integración de funciones. Los estados de amplificación para el filtro en el canal de datos van seguidos de una ganancia amplificada programable que alimenta a los controladores de las bocinas diferenciales de 8 Ω para las llamadas al proceso del monitor de bocinas AT41

El controlador del monitor de bocinas además puede ser programado para que tenga una ganancia de 0 dB o para que tenga silencio, utilizando el registro de control 2. La fuente para la entrada del monitor de bocinas puede tomarse de cualquier salida existente en la salida amplificada del DAC (DATA_OUT PGA) o en la señal de entrada del ADC, utilizando el registro de control 1. Se proporciona un voltaje de referencia de 2.5V / 1.5V (DT_REF), para la transformación. Es necesario que exista este voltaje de referencia por encima de la tierra física, ya que los amplificadores se apagan utilizando un suministro único de energía

❑ Canal analógico de voz

El circuito analógico en el canal de voz, incluye alimentación de micrófono con una fuente máxima de 5mA y 2.5V / 1.5V. Además, tiene un preamplificador de micrófono que permite seleccionar ganancias de 0 dB o 20 dB.
El dispositivo también tiene una interface microtelefónica, con amplificadores de recepción y transmisión (ver la figura A-1). Estas tres entradas se pueden sumar en cualquier combinación y el resultado se envía a un amplificador de ganancia programable PGA *line_in*. Durante esta etapa existe un intervalo de ganancia que va desde 12 dB hasta –36 dB y con 1.5 dB del silencio. Con esta ganancia se alimenta al canal de voz ADC. En el trayecto DAC, la salida del DAC se envía al PGA *line-out* con un intervalo de ganancia que va desde 12 dB hasta –36 dB y con 1.5 dB del silencio. Esta alimentación sirve para ambos controladores de salida TAPI de 600 Ω y para el controlador de bocina de 60 Ω (la bocina puede ser programada con silencio o con ganancia de 0 dB).

El *time-out* para la ganancia del silencio cambia cuando se cumple el tiempo máximo que el sistema puede esperar para que ocurra un cruce por cero de la señal, antes de que el sistema solicite un cambio de ganancia. Este tiempo es de aproximadamente 9 ms. Las figuras A-2 y A-3, muestran las configuraciones típicas del canal de voz y de datos, respectivamente, del *codec*.

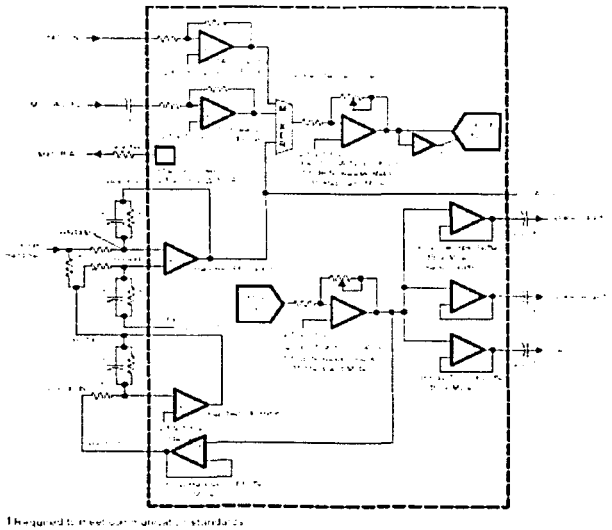

Figura A-2. Configuración de una aplicación típica del canal de voz del *codec*.

❑ Lógica miscelánea y otros circuitos

Las funciones lógicas incluyen los circuitos necesarios para implantar dos puertos seriales independientes y registros de control programables, gracias a una comunicación secundaria existente sobre dichos puertos.

Se tienen cinco registros de control que son programados durante la comunicación secundaria, tanto para canal de datos como para el canal de voz del puerto serie. Esos registros de control establecen la ganancia del amplificador de las siguientes formas: eligiendo diversas entradas multiplexadas, seleccionando funciones *loopback* y escogiendo lecturas de banderas de *overflow* del ADC.

El dispositivo también incluye un circuito *power-on reset* (POR) para la fuente de poder del monitor 5V / 3.3V MV_{DD} en el sistema y proporciona una señal de *reset* cuando el voltaje suministrado, MV_{DD}, cae por debajo del umbral del voltaje. Recapitulando, existe un circuito *flash write enable* (FWE), que toma una entrada lógica externa y proporciona 40 mA de corriente, para la energía del *write enable circuit* (WEC), desde un dispositivo de memoria externa. El *flash write enable circuit* se alimenta por medio de la fuente de poder digital.

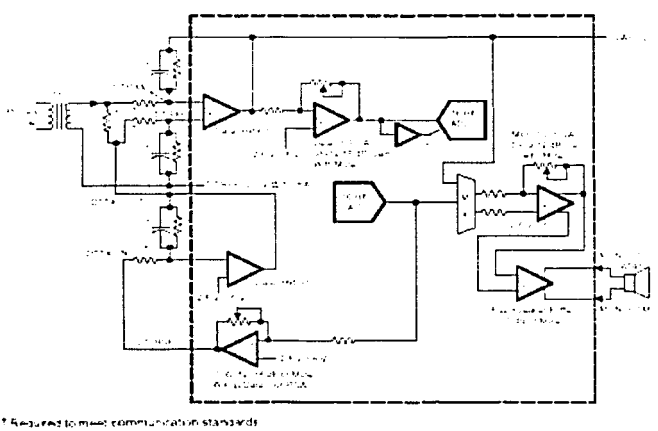

Figura A-3. Configuración de una aplicación típica del canal de datos del *codec*

A.3 Descripción del funcionamiento del codec

❑ Frecuencias de operación

El TLC320AD535 es capaz de manejar una frecuencia máxima de muestreo de 11.025KHz, en cualquier canal de datos o de voz. La frecuencia de muestreo se establece con la ayuda de la frecuencia del reloj maestro del *codec* (representada por una entrada al puerto serie para ese canal). La frecuencia de muestreo (o de conversión), se obtiene a partir de la frecuencia generada internamente en el circuito divisor del reloj maestro del *codec*, por medio de la siguiente ecuación:

$$f_s = \text{frecuencia de muestreo} = (MCLK_x / 512)$$

En donde, xMCLK se refiere a cualquier canal de voz o de datos del *codec* (VC_MCLK ó DT_MCLK) y alimenta externamente al *codec* por medio del circuito divisor de frecuencias. Este circuito divide el reloj maestro del sistema, hasta obtener la frecuencia necesaria para alimentar el *codec*.

El inverso de la frecuencia de muestreo, es el periodo de muestreo (o conversión). La frecuencia de muestreo de los canales de voz y datos, se puede establecer de manera independiente, utilizando sus relojes maestros respectivos. Ambos canales, pueden ser muestreados simultaneamente a diferentes frecuencias.

⌟ Señales del canal ADC

Las señales de entrada son amplificadas y filtradas por medio de *buffers* antes de ser colocadas en su respectiva entrada ADC. Para el caso del canal de voz, las entradas del micrófono y del microteléfono, pueden sumarse antes de amplificarse y atenuarse, por medio de la línea PGA del ADC.
El ADC convierte la señal de salida en palabras digitales discretas, con un formato en complemento a 2 que corresponde al valor de la señal analógica en el tiempo de muestreo. Esas palabras digitales de 16 bits, representan el valor muestreado de la señal analogica de entrada. Se envían al *host* a través de la interface del puesto serie, para sus canales respectivos.

Si el ADC alcanza su máximo valor, se activa una bandera del registro de control. Este bit de *overflow*, esta presente en el registro de control 2 del canal de datos (en DC), o en el registro de control 5, del canal de voz. Este bit únicamente puede ser leído por su respectivo puerto serie y la bandera de *overflow* se desactivara solo cuando se realice la lectura a través del canal de voz de su mismo puerto serie, de igual forma ocurre para el canal de datos. Dos pines externos (CAP_V y CAP_D), agregan un capacitor en serie antes de la entrada ADC, utilizado para realizar el acoplamiento de la corriente alterna con la entrada del ADC, eliminando la corriente directa de *offset*. Las conversiones ADC y DAC, son síncronas y de fase cerrada.

⌟ Señales del canal DAC

El DAC recibe los datos en palabras de 16 bits (con complemento a 2), desde el *host* y a través de la interface del puerto serie, para cada canal. El dato recibido se convierte en voltajes analógicos, por su respectiva sigma - delta DAC, que incluye un filtro de interpolación digital y un modulador digital.

Las salidas de los DACs son enviadas al filtro paso bajas interno, para completar la reconstrucción de la señal, lo que da como resultando una señal analógica. Estas señales analógicas son almacenadas y amplificadas con un controlador de salida, capaz de manejar la carga requerida. La ganancia de esta salida amplificada es programada por los registros de control del *codec*.

⌟ Sigma - Delta ADC

Cada ADC es un modulador de sobremuestreo sigma - delta. El ADC proporciona alta resolución y bajo desempeño del ruido, utilizando técnicas de sobremuestreo y las ventajas de la forma del ruido en las modulaciones sigma – delta.

⌟ Filtro de decimación

Cada filtro de decimación reduce la frecuencia de los datos digitales en la frecuencia de muestreo. Esto se logra decimando con una relación igual a la frecuencia de muestreo. La salida de este filtro es una palabra de 16 bits en complemento a 2, sincronizada con la frecuencia de muestreo seleccionada.

- **Sigma - Delta DAC**

Cada DAC es un modulador de sobremuestreo sigma - delta. El DAC, posee alta resolución y bajo ruido en la conversión digital – analógica, usando técnicas de sobremuestreo sigma - delta

- **Filtro de interpolación**

Cada filtro de interpolación vuelve a muestrear el dato digital, utilizando una frecuencia de N veces la nueva frecuencia de muestreo, donde N es la relación del sobremuestreo. La salida de datos de alta velocidad de este filtro, es utilizada por el sigma - delta DAC

- **Loopbacks analógico y digital**

Las capacidades de prueba incluyen *loopbacks* analogico y digital. Los *loopbacks* proporcionan una forma de revisar los canales ADC / DAC y se usan para efectuar pruebas dentro del circuito, a nivel de sistema. Los *loopbacks* se alimentan de la salida ADC a la entrada DAC, en el circuito integrado, para cada canal individual.

Las funciones del *loopback* analogico, solamente prueban las partes del dispositivo *codec* y no incluyen los amplificadores hibridos. Los ciclos *loopbacks* analogicos de la salida DAC, regresan a la entrada del ADC del mismo canal. Los ciclos de la salida ADC, regresan a la entrada del DAC, en el canal respectivo. El *loopback* analogico es habilitado mediante el bit D4 del registro de control 1, para el canal de datos o en el registro de control 3 para el canal de voz. El *loopback* digital, es activado utilizando el bit D5 del registro de control 1, para el canal de datos o por el registro de control 3, para el canal de voz.

- **Software de bajo consumo de energia**

El software de bajo consumo de energía, restablece todos los contadores internos, pero mantiene el contenido de los registro de control programables, sin cambiar la seleccion del canal. Este dispositivo contiene bits separados e independientes de software de bajo consumo de energia para los canales de canales de voz y datos. Las caracteristicas de este software son llamadas activando el bit D5 dentro del registro de control 1, para el canal de datos o el mismo bit dentro del registro de control 3, para el canal de voz. No existe ninguna funcion de hardware de bajo consumo de energía, en el TLC320AD535.

A.4 Comunicaciones seriales

DT_DOUT, DT_DIN, DT_SCLK y DT_FS son señales de comunicación serial para el canal de datos del puerto serie, mientras que VC_DOUT, VC_DIN, VC_SCLK y VC_FS son señales de comunicación serial para el canal de voz del puerto serie. Los datos de salida digital del ADC, se toman de DT_DOUT (o VC_DOUT). Los datos de entrada digital del DAC, se colocan en DT_DIN (o VC_DIN). El reloj de sincronización para la comunicación serial de datos y el pulso de sincronización de trama, se obtienen de DT_SCLK para el canal de datos y de VC_SCLK para el canal de voz. El pulso de sincronización de trama, que señala el comienzo del intervalo de transferencia de datos del ADC y del DAC, se toma de DT_FS y VC_FS para los canales de voz y datos, respectivamente.

Para indicar la transferencia de datos del ADC o hacia el DAC, se utiliza una comunicación primaria. En una segunda comunicación, se leen o escriben palabras en los registros de control, que manejan las opciones y la configuración del circuito del dispositivo. El proposito de ambas comunicaciones, es permitir que un dato de muestreo y un dato de control, sean transmitidos a traves del mismo puerto serie.

La primer transferencia se dedica siempre a la conversión de datos. La segunda transferencia se usa para establecer o leer los valores de los registros de control. La primer transferencia ocurre en

cada periodo de muestreo. La segunda transferencia ocurre únicamente cuando es requerida. La segunda comunicación serial es solicitada por medio de software, utilizando el bit D0 de la primer entrada de datos en DT_DIN, si se trata del canal de datos del puerto serie o en VC_DIN para el caso del canal de voz del puerto serie.

La segunda petición puede ser realizada por el canal de voz, sin necesidad de hacer una segunda petición para el canal de datos o viceversa. Los registros de control 1 y 2, únicamente pueden ser establecidos o leídos, desde el canal de datos del puerto serie. Los registros de control del 3 al 6, pueden ser establecidos o leídos únicamente desde el canal de voz del puerto serie.

- **Primera comunicación serial**

La primer comunicación serial, transmite y recibe conversión de datos de la señal. Después de encender o *resetear*, los dispositivos entran al modo de 15 bits DAC. La longitud de la palabra DAC es de 15 bits y siendo el último bit de la primer palabra de 16 bits de la comunicación serial, un bit de control usado para solicitar una segunda comunicación.

En todas las comunicaciones seriales, el bit mas significativo es el que se transfiere primero. Para la palabra de 16 bits ADC, D15 es el bit mas significativo y D0 es el bit menos significativo. Para la palabra de datos de 15 bits DAC en una primer comunicación, D15 es el bit mas significativo, D1 es el bit menos significativo y D0 es usado para solicitar una segunda comunicación de control. Todos los valores de los datos digitales estan en un formato de complemento a 2. La figura A-4 muestra el formato de esta primera comunicación.

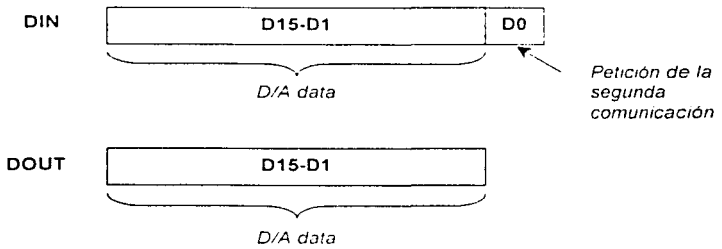

Figura A-4. Formato de la primera comunicación DIN y DOUT

- **Segunda comunicación serial**

La segunda comunicación serial lee o escribe palabras de 16 bits, que programan las opciones y la configuración del circuito del dispositivo, para cualquier canal de voz o datos. La programación de los registros ocurre siempre durante la segunda comunicación, para ese canal. Los registros de control 1 y 2 sólo pueden ser escritos o leídos desde el canal de datos del puerto serie. Los registros de control del 3 al 6, unicamente pueden ser escritos o leídos, desde el canal de voz del puerto serie.

Se requieren cuatro ciclos de comunicación primaria y secundaria, para programar los cuatro registros de los canales de voz. Análogamente, se requieren dos ciclos de comunicación, primarios y secundarios, para programar el registro de control del canal de datos. Si se desea el valor por omisión, para un registro en particular, entonces el registro de direccionamiento, puede ser suprimido durante la segunda comunicación. El comando *NOP* (no operación), direcciona un pseudo registro, denominado registro 0 y ningun otro registro programado toma su lugar durante

esa comunicación secundaria. Esto puede ser usado por cualquier canal de datos o voz del puerto serie

Durante una segunda comunicación, un registro es escrito a o leído de Cuando se establece un valor en el registro, las líneas DT_DIN o VC_DIN contienen el valor que será escrito El dato que regresa en DT_DOUT (o VC_DOUT) es 00h El método para solicitar una segunda comunicación, es activando el bit menos significativo (D0) de DT_DIN (o VC_DIN) La tabla A-1 describe las funciones del bit D0

Tabla A-1 Funciones de control del bit menos significativo

Bit de control D0	Función del bit de control
0	No necesita pedir segunda comunicacion
1	Pedir segunda comunicacion

La figura A-5, muestra el formato de los datos para la segunda comunicación

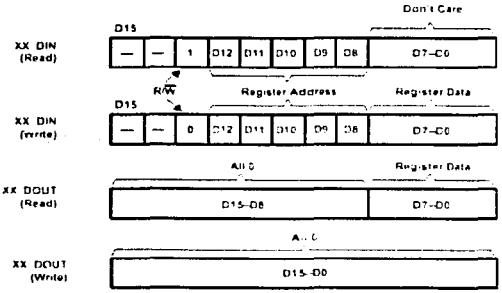

Figura A-5 Muestra el formato de datos XX_DIN y XX_DOUT durante la segunda comunicación

Para obtener más información sobre el funcionamiento del convertidor A/D, ver el manual TLC320AD535C/I *Data Manual*, de *Texas Instruments*

BIBLIOGRAFÍA

Proakis J y Manolakis. D *Tratamiento Digital de Señales* Tercera Edición Editorial Prentice Hall Madrid España. 1998

Deller. Proakis & Hansen *Discrete-Time Processing of Speech Signals* Tercera Edición. Editorial Prentice Hall New Jersey. U S A . 1987

Rabiner L & Juang B H *Fundamentals of Speech Recognition* Segunda Edición Editorial Prentice Hall New Jersey. U S A . 1993

Gersho. A & Gray. R M *Vector Quantization and Signal Compression* Sexta Edición Editorial Kluwer Academic Publishers Norwell. Massachusetts U S A . 1997

Parsons, Thomas W Voice and Speech Processing Primera Edición Editorial McGraw Hill. México. 1987

Owens Frank J Signal Processing of Speech Primera Edición Editorial McGraw Hill México. 1987

Herrera Camacho. José Abel Apuntes de Procesamiento Digital de Voz UNAM 1999

Gardida Degollado. Arturo *Reconocimiento de Palabras Aisladas Utilizando Cuantización Vectorial* Tesis. Ingeniero en Telecomunicaciones FI UNAM. 1998

Ibarra Salinas. Ricardo *Reconocimiento Automático de Voz Usando LPC y Transformada HLT, Aplicando Técnicas de Ajuste Dinámico de Tiempo* Tesis Ingeniero en Telecomunicaciones FI. UNAM 1998

Sánchez Cacique Oscar *Segmentación Acústica de Voz* Tesis Ingeniero en Computacion FI. UNAM. 1997

Flores Espinoza Andres Reconocimiento de palabras aisladas Pontificia Universidad Católica del Perú *http://www.alek.pucp.edu.pe/~dflores/tesis/modelo.html* 1998

Lopez Alonso. Luis Alberto Implementación y simulación del procesamiento de la señal de voz en la interfaz de radio del sistema de telefonía movil GSM Universidad de Granada. España *http://ceres.ugr.es/~alumnos/alonso/index.html* 1999

Code Composer Studio User's Guide Texas Instrumets

TMS320C6000 CPU and Instruction Set Reference Guide. de Texas Instrumets

TMS320C6000 Peripherals Reference Guide de Texas Instrumets

TMS320C6000 Assembly Language Tools User's Guide. de Texas Instrumets

TMS320C6000 DSP/BIOS User's Guide de Texas Instrumets

TMS320C6000 DSP/BIOS Application Programming Interface (API) Reference Guide, de Texas Instrumets

TMS320C6000 Programmer's Guide de Texas Instrumets.

TLC320AD535C/I Data Manual. de Texas Instrumets

ÍNDICE ALFABÉTICO