UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

SISTEMA GENERADOR DE VOZ MEDIANTE TECNICAS DE COMPACTACION DE FONEMAS

T E S I S

QUE PARA OBTENER EL TITULO DE:

INGENIERO EN COMPUTACION

P R E S E N T A N :

ANA ELENA SASTRIAS BORDES
VIRGINIA OLIVIA LOZANO ROBLEDO
JOSE IVAN EDGARDO MOYA MEZA

Director de Tesis:

ING. ROBERTO MANDUJANO WILD

MEXICO, D. F. 1989

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

INDICE

PROLOGO

PROLOGO

El ser humano, desde el inicio de su existencia, se ha esforzado continuamente por comprender y controlar la naturaleza, como consecuencia de esto, ha tratado de explicarse los diferentes fenómenos que ocurren a su alrededor. Primero buscó una explicación mediante una *mística religiosa* que le permitió describir de alguna forma ciertos fenómenos, con lo cual logró apaciguar su inquieto espíritu. Pero a medida de que sus conocimientos se incrementaron, gracias al legado de varias generaciones de pensadores, tuvo que sustituir mucha de su mística religiosa por explicaciones más concretas basadas en el mundo observable del cual tomaba sus experiencias. A medida de que sus conocimientos se incrementaban, tuvo la oportunidad de tomar elementos de una y otra de las "*proto-ciencias*" que lo rodeaban, las organizó y complementó hasta que fue capaz de predecir y controlar en forma casi exacta múltiples fenómenos de su medio ambiente, incluso fue capaz de detectar ciertos fenómenos que no permitían una observación directa mediante métodos inductivos y deductivos, lo cual le abrió un universo aún más grande, que nunca pensó que existiece.

Uno de los aspectos más importantes de la Ciencia, es que nos permite ampliar y extender muchas de las capacidades naturales del hombre. Por ejemplo, máquinas, que manejadas por un operador pueden levantar varias toneladas con sólo mover una palanca o apretando un botón. Más recientemente, las computadoras, que son una herramienta que extiende la capacidad mental del ser humano dejando para éstas las tareas más monótonas y rutinarias, dan al hombre la oportunidad de aplicar su mente al desarrollo creativo, que es a final de cuentas, lo que lo ha hecho tan grande como especie.

En sus inicios las computadoras fueron consideradas como máquinas muy complicadas, las cuales sólo podían ser operadas por especialistas en estos sistemas, siendo un misterio y algo casi místico para los no conocedores. Sin embargo, es un hecho que cualquier descubrimiento o invención que sea útil para unos pocos, al pasar del tiempo se convirtiera en indispensable para el resto de los humanos; la computadora no podía ser la excepción. Gracias al desarrollo tecnológico que tuvo la Electrónica y otras diferentes disciplinas científicas y tecnológicas, el costo y el tamaño de la computadora se redujo hasta convertirse en una necesidad a nivel industrial y, poco a poco, lo será a nivel doméstico.

Mucho del misticismo de la computación ha desaparecido pero aun existe cierto repudio de algunas personas porque piensan que las computadoras van a desplazar a nivel laboral al hombre, pero esto es falso, ya que se considera a la computadora como una ayuda, y que sin la intervención del hombre, que le dé instrucciones, ésta no podría realizar ningún proceso.

Por otro, lado algunas personas rechazan a las computadoras por el hecho de tener que aprender a comunicarse en un lenguaje conveniente para ellas, pues esto a veces dista de los deseos del usuario promedio. Por estas razones, se ha intentado hacer sistemas más "*amigables*" tratando de que la computadora pueda *interpretar y responder en forma más natural los deseos del usuario*, tomando en cuenta que no existe nada mejor para el hombre que la comunicación verbal, se han diseñado sistemas de *Generación, Síntesis y Reconocimiento de Voz*.

Cada vez más, la idea de hablar con una computadora y obtener de ella respuestas verbales, se va convirtiendo en una necesidad que puede tener un sinnúmero de aplicaciones prácticas que quizá aún no hemos imaginado.

Las razones anteriores, entre otras, son las que nos motivaron a tomar este tema de tesis, en el cual sólo se ven algunos de los múltiples temas y subtemas que intervienen en los sistemas de síntesis y reconocimiento de voz. El campo de investigación a este respecto es tan amplio y variado que llevaría muchos años de investigación profundizar en forma detallada en cada uno de los aspectos que comprende. Por el momento, una de sus aplicaciones más interesantes es utilizar nuestro proyecto como **Diagnóstico en Anomalías en el Tracto Vocal**, o en **Parálisis Psicomotriz**. Para el desarrollo del proyecto ***FONETICS*** se consideraron conceptos de **Síntesis de Voz**,(como periodo de pitch, parámetros del tracto vocal y parámetros de excitación), así como **Formas de Representación de Señales de Voz**, **Tipos de Modulación** y **Técnicas de Compactación de la Señal**.

Se agrega una reseña histórica sobre otros proyectos que han utilizado los conceptos que hemos mencionado anteriormente además de comprender conceptos de **Lingüística y Fonética**, al hacerse una basta clasificación de los fonemas del Español hablado en México.

CAPITULO I

I. INTRODUCCION

El hombre nunca apartó de su mente el crear algún día un prototipo de máquina que contestara con voz a una orden en forma escrita o en forma hablada.

En aquel entonces se realizaban novelas y películas de ciencia ficción en donde la computadora hablaba y razonaba tomando sus propias decisiones, controlando una serie de situaciones. Se pensó pues, que eso era una *fantasía* sin saber que, poco a poco, sería posible, gracias al avance tecnológico, surgiendo la *Síntesis y Reconocimiento de Voz* así como otros logros trascendentes.

La razón por la cual se eligió este tema fue la inquietud por conocer y entender los diversos métodos de síntesis y generación de voz, y tomar posteriormente, estos principios para una aplicación práctica, flexible y de bajo costo, y de ser posible, simplificar algunos procesos ya existentes en la comunicación *hombre - máquina*.

Para el desarrollo de este trabajo se realizó una investigación del tracto vocal, del comportamiento del lenguaje y de proyectos anteriores que facilitaron el desarrollo de la tesis.

El primer elemento a analizar, fue la *palabra hablada*, la cual está constituida por unidades elementales sonoras llamadas ***fonemas***, cuyo orden está derterminado por las reglas propias de cada lenguaje. Al estudio de estas reglas se le llama *Lingüística*, al estudio y clasificación de los sonidos del habla se le llama *Fonética*. Ambas, *Lingüística* y *Fonética* son dos disciplinas fundamentales para el desarrollo de este proyecto, ya que una dará la estructura correcta para ensamblar los ***fonemas*** y producir palabras y frases, mientras que la otra permitirá conocer el número de ***fonemas*** requeridos y las características de cada uno para generar un vocabulario adecuado a las necesidades de cada aplicación, así como la cantidad aproximada de información que se necesita almacenar en la computadora.

I.1 CONTENIDO GENERAL DE LA TESIS

La tesis se ha dividido en ocho capítulos y dos apéndices, en el **Capítulo I** se incluye una breve introducción que describe las inquietudes iniciales que motivaron el desarrollo de este proyecto, así como los objetivos del mismo. El **Capítulo II** está dedicado a una breve reseña de algunos antecedentes históricos que pueden ser de interés, tanto por su forma de abordar los diferentes problemas, como por sus aplicaciones.

Los **Capítulos III**, **IV** y **V** están dedicados a conocer los elementos básicos de la producción de voz, su análisis y otros aspectos teóricos que aún cuando no son aplicados, son útiles como parámetros de comparación, en los **Capítulos VI** y **VII** se describe, analiza y justifica el desarrollo de la tesis, además se incluye el Manual del Sistema ***FONETICS***, el cual será de gran utilidad para que cualquier persona pueda aprender a manejarlo de forma rápida y sencilla.

El **Capítulo VIII** está dedicado a proponer algunas aplicaciones prácticas del sistema ***FONETICS***.

En el **Apéndice A** se puede profundizar con más detalle en la arquitectura de las computadoras PC.

El **Apéndice B** contiene el Diccionario de Datos del Programa del Sistema ***FONETICS*** y se muestra el listado del programa del sistema ***FONETICS***.

Por último hay una sección de **Referencias Bibliográficas** que permitirá tener otras bases de consulta.

I.2 OBJETIVO

El objetivo de esta tesis no es la creación de modelos matemáticos de manera exhaustiva de los sistemas de generación de voz, si no más bien la de *crear un sistema práctico y de bajo costo* que permita la generación de voz mediante ***fonemas*** pregrabados y almacenados en la computadora en forma compacta para, posteriormente, reproducirla y reducir los requerimientos de memoria. Los ***fonemas***, una vez almacenados, pueden ser reproducidos convirtiéndolos nuevamente en sonido. Los ***fonemas*** serán concatenados en forma ordenada para articular palabras, estas palabras podrán ser generadas a partir de datos leídos desde el teclado, los cuales serán procesados para obtener palabras y frases completas.

Aún cuando a primera vista el problema pudiera parecer simple, realmente implica mucho más que la conversión analógica-digital y digital-analógica, ya que algunos de los puntos que se deben de tomar en cuenta son:

a) Un análisis completo de la estructura fonética del lenguaje,especialmente en español.

b) Determinación exacta del inicio y del fin del ***fonema*** para poder almacenar sólo la información deseada de un ***fonema*** en particular.

c) Técnicas de compactación de la información, las cuales deben permitir un rápido almacenamiento de la señal en el menor espacio de memoria posible y un rápido acceso a la misma en el momento en que se requiera generar el ***fonema***.

d) Optimación de la frecuencia de muestreo para no tener que manejar una gran cantidad de muestras y poder recuperar a la salida del sistema, voz de alta calidad.

e) Conocer la teoría elemental de los fenómenos acústicos relacionados con la generación de voz.

f) Conocer los factores que generan el ***ruido eléctrico*** para así poderlo evitar y reproducir fielmente la señal de voz.

CAPITULO II

II. ANTECEDENTES

II.1 SISTEMAS DE RESPUESTA HABLADA:

Los sistemas de respuesta hablada se han usado en gran variedad de aplicaciones comerciales. La forma menos costosa y más natural de recibir respuestas detalladas de una computadora por vía telefónica, consiste en recibirlas en forma hablada, de modo que un receptor telefónico convencional sea el mecanismo de salida. Por ejemplo, un sistema de respuesta hablada podría servir para verificar el crédito de un cliente en un almacén de departamentos mediante una línea telefónica conectada a una computadora, que enviaría la respuesta en forma verbal, dando así el crédito disponible.

Actualmente, existen sistemas de transmisión y almacenamiento digital de la señal de voz (***VOCODER***), sistemas de síntesis de voz, sistemas de verificación e identificación de la señal de voz y sistemas de reconocimiento de la señal de voz.

II.2 ALGUNOS MECANISMOS DE RESPUESTA HABLADA

Desde los inicios de los *70's*, se pensó en dar servicio a los usuarios en su propia casa o centro de trabajo. La terminal menos costosa en que se pensó fue el teléfono, por lo que se han diseñado muchos sistemas con respuesta hablada. Uno de los primeros fue el de la Bolsa de Valores de Nueva York, que en los años *60's* propocionaba cotizaciones actualizadas de los precios de los valores. El mecanismo que usa se asemeja a un teléfono ***Touch-Tone*** en su teclado que está conectado a un sistema **IBM.** Tiene doce teclas en vez de las diez de los teléfonos convencionales (Hay que notar que cada tecla tiene dos significados posibles. Por ejemplo, la tecla "8" puede utilizarse para pedir a la computadora que repita su respuesta). El usuario anota con las teclas los detalles de los cálculos que quiera efectuar, y la computadora contesta hablando.

Luego anota con las teclas los operadores de los cálculos. Cada operador que se marca (excepto el signo "+") llevará en seguida un asterisco "*", para que la computadora sepa que es un operador. Se termina cada operación con el signo "**".

Los primeros esfuerzos relacionados con el procesamiento digital de la señal de voz, análisis y síntesis de la señal, fue la construcción de "***VOCODERS***" para la reducción de ancho de banda de frecuencia telefónica, inventado por Homer Dudley en 1928. En muchas ocasiones era incrementado el ancho de banda a causa de los satélites, microondas y sistemas ópticos de comunicación. Por otro lado, la necesidad aumentó para los sistemas digitales de voz, ya que se necesitaba un menor número de bits como fuera posible y a menor costo para un futuro. Después, la realización de sistemas de respuesta acústica con mensajes predefinidos y almacenados digitalmente aumentó la complejidad en la codificación y parametrización digital.

En relación a la síntesis de voz, uno de los problemas es la conversión de cualquier texto ortográfico a voz, partiendo ya no de un análisis de un mensaje particular sino de un conocimiento exhaustivo de las reglas de producción de la voz tanto a nivel fisiológico y fonológico como prosódico y psicológico.

Hay básicamente dos formas distintas de que la computadora transmita señales de voz. Una consiste en registrar los sonidos en forma analógica, como se registran en una grabadora en forma magnética.

En un sistema típico, que consta de un cilindro, la superficie se divide en rayas de tiempo de longitud fija y cada una de ellas contiene una palabra humana o una porción de palabra. Se seleccionan una o varias palabras las cuales, primero son captadas por las cabezas magnéticas y posteriormente la señal correspondiente a la palabra es amplificada para conectarse a la línea telefónica. Cuando una palabra ocupa más de una raya de cilindro, la computadora transmitirá las rayas apropiadas en secuencia. La conmutación entre las cabezas del cilindro se hace con suficiente rapidez para que no haya interrupción perceptible entre las rayas que se transmiten.

En la Bolsa de Valores de Nueva York se usa un mecanismo de esa índole para que los corredores coticen los precios actuales de los valores. Los detalles del último precio del mismo se almacenan en los archivos de la computadora de la Bolsa de Valores, que tiene mecanismos de transmisión de datos en el piso en los que los corredores anotan todos los cambios.

Al contestar las preguntas, la computadora escoge las palabras apropiadas o las fracciones de palabra del cilindro y a medida que este gira, frente a las cabezas, las palabras se transmiten por la línea hasta el teléfono del corredor, que recibe una descripción clara y audible, del precio del valor requerido.

El método del registro analógico tiene dos limitaciones:

Primera, el vocabulario queda limitado por el tamaño del tambor; en un sistema típico el número máximo de palabras que pueden usarse es de 128.

Segunda, si el sistema usa rayas de tiempo de longitud fija, la respuesta tendrá que consistir de bloques cortos de sonidos de longitud fija que pueden oirse en forma desarticulada; por ejemplo, el sistema de la Bolsa de Valores de Nueva York se oye más interrumpido que un registro de voz que anuncie un número telefónico incorrecto.

La otra técnica de respuesta hablada no utiliza registros analógicos, sino más bien un registro digital.

Una unidad de disco de computadora que pueda almacenar, por ejemplo 100 millones de bits, podría retener de ese modo, un gran vocabulario de voz. El programa puede manejar las palabras (*y las pausas*) como se desee, componer frases, almacenarlas en cinta, etc. Pueden almacenarse palabras o frases de longitud completamente variable y de ese modo evitar el ajuste desarticulado de los sonidos de longitud fija.

La ventaja de este segundo método es que puede construirse un mecanismo para que el usuario registre o ponga en dígitos, nuevas palabras o frases con toda facilidad.

El empleo de un sistema de respuesta hablada requiere de una forma poco costosa que permita la transmisión de pequeñas cantidades de datos a la computadora a fin de que el usuario pueda hacer preguntas y anotar datos. En algunos sistemas como en el de la Bolsa de Valores de Nueva York, esto se logra con el mismo disco del teléfono y a veces se usan las diez teclas de un teléfono ***Touch-Tone***.

En el sistema bancario ***Touch-Tone***, la cajera de banco inserta una tarjeta de datos en su teléfono de disco que la conecta automáticamente con la computadora del banco. La máquina contesta la llamada y da una señal de tono para indicar que espera la transacción. La cajera inserta luego la tarjeta de la cuenta personal del cliente en su teléfono para que la computadora sepa quién es el cliente y se prepara a recibir información adicional. A continuación la cajera marca en el disco en la computadora la cantidad depositada o retirada. La computadora al recibir la información actualiza el nuevo saldo en la cuenta del cliente.

En un sistema típico en el que se use un disco telefónico, el teléfono se conecta primero con la computadora, marcando su número de línea en la forma acostumbrada. El usuario sabe cuándo se establece la conexión, porque la computadora envía un "tono de datos", que por lo general es una nota aguda de unos cuantos segundos de duración. El usuario marca luego un dígito, número de cuenta u otra información que almacena el programa de la computadora el cual es usado para para responder a la solicitud.

Otro de los desarrollos orientados a la generación de voz es el de Ray Kurzweil, experto en Inteligencia Artificial(*).

Algunas de estas máquinas también hacen de los minusválidos seres humanos menos dependientes. En la década de 1970, Kurzweil desarrolló una máquina de lectura que podría considerarse como el sistema más importante para ayudar a los ciegos desde el *código Braille*. La misma máquina puede identificar palabras en una página impresa y luego leer ésta en voz alta mediante un procedimiento de voz sintetizada. Es en efecto, una máquina que lee, con vocabulario de 20,000 palabras.

La mente humana funciona a través de un amplio espectro, mientras que las computadoras pueden programarse para que funcionen de modo experto en pequeñas áreas de especialización sin fatigarse. Kurzweil sugiere combinar los sistemas expertos, con una inteligencia bien definida, dentro de ciertos parámetros, como el reconocimiento de patrones gráficos o de voz. Como ejemplo, el ***Voice RAD*** que ha aprendido a reconocer "*frases activadoras*", es decir, "*frases comando*". La máquina de lectura ***Kurzweil*** identifica las formas comunes de las letras del alfabeto (**A**, **B**,...,**Z**). Las máquinas ***Voice RAD*** y ***Voice WORKS*** reconocen voces individuales, sujetas a un programa de adiestramiento por las voces que lo usen normalmente; se le proporciona al ***Voice WORKS*** un vocabulario básico de 1,200 palabras, incluyendo órdenes necesarias.

A la máquina se le pueden enseñar otras palabras, esto se puede hacer grabando estas palabras en disco. De esta forma, ***Voice WORKS*** se adapta al vocabulario especial de una compañía o industria.

Cuando el usuario habla ante el micrófono, se amplifica primero la corriente del sonido, la que enseguida se analiza a 16,000 veces por segundo y codificada en forma binaria, este proceso se parece al proceso de grabación de los discos compactos. Luego se inicia el proceso de reconocimiento de formas, su función es identificar sonidos básicos, ***fonemas***, para distinguir la A de la O. Después identifica secuencia de ***fonemas***: Los sonidos de palabras completas, y otro sistema que diferencíe un homófono de otro (las palabras que tienen un sonido igual pero un significado diferente, ejemplo haya y aya), para esto, se requiere el manejo de las reglas gramaticales, para determinar cuál es la palabra correcta.

(*) El término Inteligencia Artificial (IA) se inventó en una Conferencia que tuvo lugar en la Universidad de Dartmon en 1965.

Al hablar con el ***Voice WORKS***, el operador debe dejar espacios de un décimo a dos décimos de segundo entre las palabras. Esto evita que la máquina divida una palabra en dos. En caso de que el ***Voice WORKS*** no capte una palabra, el usuario dice: *"Palabra de adiestramiento"*, y escribe en la máquina la palabra y luego la pronuncia en voz alta. Las correcciones y los errores cometidos son asimilados por las memorias de los sistemas expertos para ser más inteligentes en un futuro.

Actualmente en México se están desarrollando sistemas de reconocimiento de voz, pero aún siguen en su etapa de prueba y análisis debido a la complejidad que conllevan la etapa de identificación y verificación, utilizándose métodos paramétricos de cuantización y de predicción lineal, que en los siguientes capítulos se discutirán.

II.3 MICROVOZ, EJEMPLO DE UN PROYECTO PRACTICO DE SINTESIS DE VOZ

El sistema ***MICROVOZ*** es una calculadora diseñada como ayuda a personas invidentes, la cual utiliza la síntesis de voz mediante ***Microfonemas*** y permite oir los dígitos y operadores marcados en la ***Calculadora***, así como también los resultados de los cálculos realizados.

II.3.1 CARACTERISTICAS DEL MICROVOZ

Microvoz, es una calculadora aritmética con la siguiente capacidad de calculo:

a) Operaciones con 9 dígitos.

b) Cálculo de porcentaje.

c) Un registro de memoria.

d) Operaciones con punto flotante.

La presentación de resultados es doble, es decir, *en forma escrita*, en displays de 7 segmentos y 9 dígitos; y *en forma hablada*, destinada precisamente a personas invidentes.

El sistema acústico de ***Microvoz*** tiene dos modos de funcionamiento, seleccionables por la computadora. En el modo "***con eco***", siempre que se pulsa una tecla, se genera el eco acústico correspondiente a la función de dicha tecla; y en modo "***sin eco***", sólo se genera el mensaje acústico al pulsar la tecla "=" o "%". Además la lectura del display siempre es posible, independientemente de las operaciones aritméticas, mediante una tecla que actúa directamente sobre el *"microprocesador"*. La calculadora dispone de conexión para un potenciómetro que regulará el volumen. El vocabulario de ***Microvoz*** consta de 6 palabras, 10 dígitos, signo de adición (+), signo de sustracción (-), signo de producto (x) y signo de división (/). La reconstrucción de estas palabras se efectúa a partir de la representación codificada de 16 ***microfonemas***.

Las dimensiones de la calculadora son 3 x 9 x 7 **cm**. La bocina es de 2.5 **pulgadas**, colocada en posición inferior el consumo se eleva a 1.5 W, por lo que funciona con un adaptador a la línea; ultimamente se ha desarrollado una batería con una autonomía de una hora. La **figura II.1** muestra la vista exterior del ***Microvoz*** y la **figura II.2** muestra el diagrama de bloques del ***Microvoz***.

II.3.2 VOCABULARIO Y METODO DE SINTESIS

Las 16 palabras que constituyen el vocabulario de ***Microvoz*** suman en total 8 **s.** de voz aproximadamente.

Con una codificación **PCM** a 10 **KHz.** y 8 **bits** se hubiera necesitado 740 **Kbits** de memoria. Con las técnicas de compactación que a continuación se discuten se logra descender a 9.6 **Kbits** (1200 **Bytes**).

En primer lugar se ha tenido en cuenta la naturaleza articulada de la voz para hacer una descomposición de las palabras en sus ***fonemas*** constituyentes, que son luego concatenados por el programa de reconstrucción acústica. Aún así, (se distinguen 16 ***fonemas*** en ***Microvoz*** con una duración media de 120 **ms.**), la codificación directa de los ***fonemas*** aislados hubiera conducido a 160 **bits**. La idea del proyecto ***Microvoz*** era desarrollar una calculadora portátil con salida acústica, de dimensiones similares a las calculadoras comerciales existentes, por tanto el límite máximo de memoria para programa y datos se estableció en 2 **KBytes**. El estudio de la compactación de los datos necesarios para la síntesis condujo al ***Método de Síntesis mediante Microfonemas***, descrito en el **Capítulo** IV (sección IV.2). Los *patrones de los microfonemas* se obtuvieron de una voz femenina con un tono de 5 ms, codificándose en **PCM** de 8 **bits**, por lo que el conjunto de los 16 micro***fonemas*** ocupaba unos 800 **Bytes** de memoria.

Por otra parte, las 16 palabras del vocabulario de la calculadora contienen unos 8 ***fonemas*** en promedio; especificando la evolución temporal de la forma de onda para cada ***fonema*** mediante 5 **Bytes** resultan unos 350 **Bytes**.

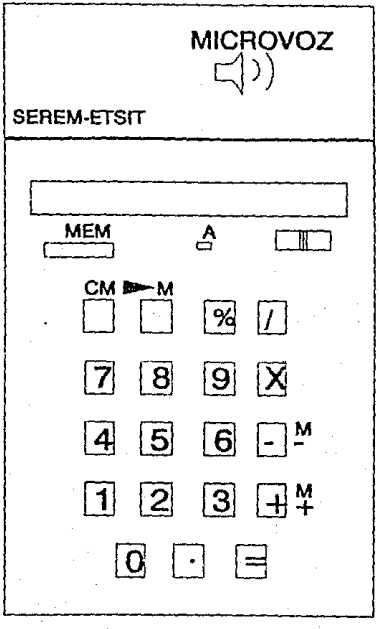


FIGURA II.1

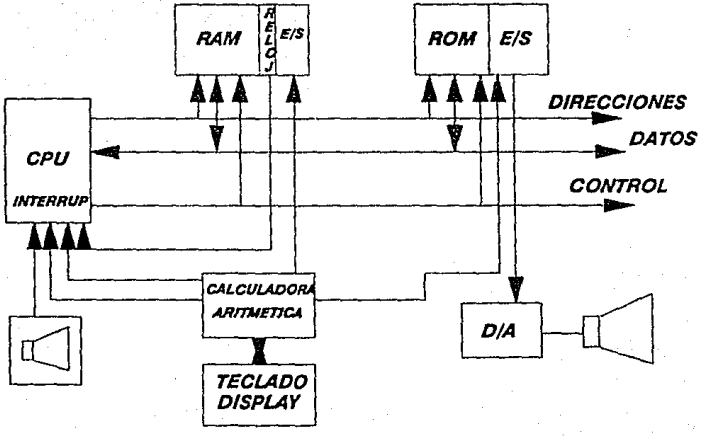

RAM
RELOJ
E/S
ROM
E/S
DIRECCIONES
DATOS
CONTROL
CPU
INTERRUP
CALCULADORA
ARITMETICA
D/A
TECLADO
DISPLAY

FIGURA II.2

En total el conjunto de datos para la reconstrucción acústica es de unos 1150 **Bytes**, estando ocupada el resto de la memoria por los *programas de comunicación* con el *microprocesador aritmético* y por el ***programa de reconstrucción acústica***. La reducción de datos mediante el ***Método de Síntesis por Microfonemas*** con respecto a una codificación **PCM directa** de la forma de onda *temporal* se sitúa alrededor de la relación **1:500** para esta aplicación concreta.

II.3.3 ARQUITECTURA DEL MICROVOZ

La ***calculadora Microvoz*** puede dividirse en tres subsistemas: *Calculadora Aritmética, Microcomputadora y Audio*. En la **figura II.2** se muestra un diagrama de bloques del conjunto de las intercomunicaciones entre sus partes.

III. FISIOLOGIA DEL TRACTO VOCAL EN LA GENERACION DE LA VOZ Y TEORIA ACUSTICA DE LA PRODUCCION DE VOZ

Es un hecho, que la mejor forma de enfrentar un problema es estudiar sus causas básicas, para posteriormente poder proponer soluciones basadas en realidades concretas. Por esta razón, se considera fundamental el estudio de la Fisiología del Tracto Vocal y la Teoría Acústica de la Producción de Voz, para la Síntesis de Voz.

III.1 ORIGEN DEL LENGUAJE HABLADO

El lenguaje surge por imitación, en la que intervienen no sólo factores audio-motores si no igualmente factores óptico motores, a través de los cuales se va estructurando el lenguaje infantil. Este proceso no sólo abarca la articulación, como imitación de movimientos y sonidos, si no también la comprensión de lo escuchado y expresado.

Para entender cómo se genera voz, es necesario recurrir a la ***fisiología del tracto vocal***, para así hacer un modelo matemático que simule su funcionamiento y genere, en forma aproximada, la señal de voz con un mínimo de error. En este capítulo se describen en forma somera los Mecanismos de Producción de Voz, y la Fisiología de la Articulación de ***Fonemas Básicos***.

El tracto vocal empieza en la apertura entre las cuerdas vocales o glotis y la terminación de los labios. Este tracto esta constituído básicamente por la faringe y la boca o cavidad oral. En promedio el largo del tracto vocal es de aproximadamente 17 **cm** y la sección transversal varía desde cero hasta 20 **cm** cuadrados. El tracto nasal comienza en la úvula (vélum) y termina en la cavidad nasal (nostril).

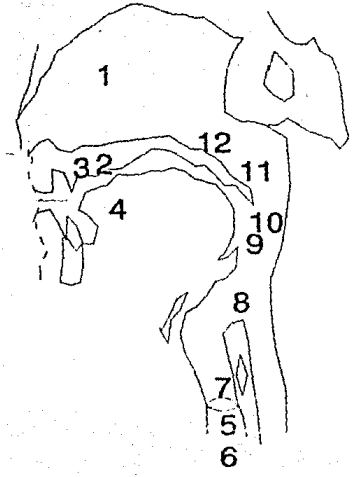
Cuando se baja el vélum el tracto nasal se acopla acústicamente con el tracto vocal para producir los sonidos nasales del habla. En la **figura III.1** se muestra un corte transversal del tracto vocal.

1.- Organos que intervienen en la articulación de los fonemas :

1.1. - Organos de la respiración: Pulmones, bronquios y la tráquea.

1.2. - Organos de la producción de voz: Faringe, laringe y cuerdas vocales

ORGANOS DEL TRACTO VOCAL



1. cavidad nasal.
2. paladar duro.
3. alvéolos.
4. lengua.
5. laringe.
6. traquea.
7. cuerdas vocales.
8. faringe laringe.
9. epliglotis.
10. laringe oral.
11. velum.
12. faringe nasal.

FIGURA III.1

2.- Función fonatoria de la laringe:

La **intensidad, tono** y **timbre** tienen su origen en la laringe. La **intensidad** depende de la presión del aire respirado, esto determina una mayor o menor amplitud en la vibración de las cuerdas vocales. El **tono** está determinado por la frecuencia de las vibraciones o el número de ellas, que se dan por segundo. El sonido es agudo, cuando se produce un gran número de vibraciones por segundo, y es grave cuando los pliegues vocales se mueven con más lentitud. El **timbre** permite distinguir unos sonidos de otros aunque del mismo tono y la misma intensidad. *La frecuencia fundamental, tono fundamental o primera armónica*, va acompañada de una serie de armónicas filtradas por los *resonadores*, que constituyen el timbre del sonido. La variación del timbre, su constitución y sus alteraciones, dependen de la caja de resonancia. En la **figura III.2** nos muestra la posición de las cuerdas vocales al producir voz.

3.- Organos de la articulación:

La corriente de aire productora del sonido pasa de la ***zona laríngea*** a la ***región laringo-faríngea y laringo-oral***, donde va a realizar toda la división del material fónico.

3.1.- Organos activos de la articulación: labios y lengua La lengua actúa directamente en la articulación de los ***fonemas linguodentales o dentales:*** */t/,/d/*; ***linguointerdentales:*** */z/*; ***linguoalveolares o alveolares:*** */s/,/n/,/ll/,/r/* y */rr/*; ***linguopalatales o palatales:*** */y/,/ch/,/lll/,/ñ/*; y ***linguovelares:*** */k/,/g/,/j/*.

3.2.- Organos pasivos de la articulación: paladar. La mayor elevación del ***velo palatino*** se da con */g/* y la menor elevación con */f/* y */v/*; ***alveolos, dientes, fosas nasales.***

En la **figura III.1** se pueden apreciar las zonas bucales.

III.2 FORMAS DE CLASIFICACION DE FONEMAS

Los ***fonemas*** se pueden clasificar en distintos aspectos:

1) **Vocales y consonantes:** Las vocales son sonidos producidos por la vibración de las cuerdas vocales, con resonancia en la cavidad faringeo-vocal, sin que haya contacto de la lengua con la bóveda palatina y sin participación activa de la punta de la lengua, siendo los sonidos que representan mayor abertura de los órganos articulatorios. En la **figura III.2** se muestra la posición, en su parte anterior, de las cuerdas vocales.

POSICION DE LAS CUERDAS VOCALES
(PARTE ANTERIOR)

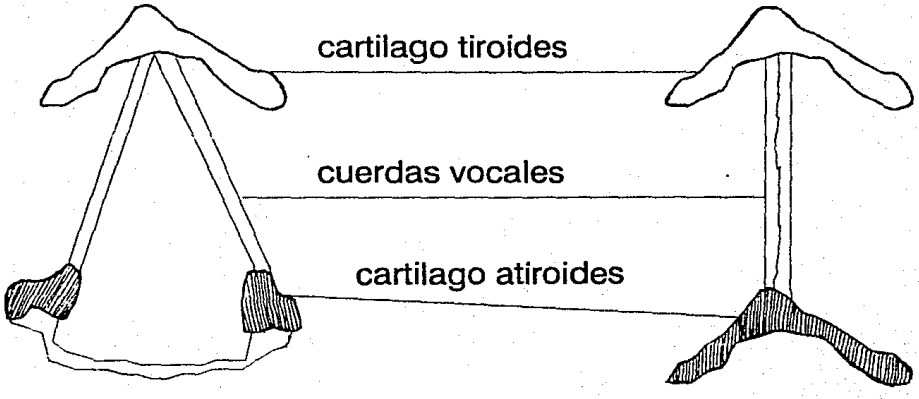


FIGURA III.2

Las consonantes se caracterizan por el **ruido** ocasionado por la aparición de un obstáculo en uno u otro punto, que se opone a la corriente de aire que fluye en la laringe.

2) **Por la acción de las cuerdas:** Si en la emisión de aire, las cuerdas vocales se aproximan y comienzan a vibrar, se origina el sonido articulado "sonoro", entre los que se encuentran las vocales y muchas consonantes. Si por el contrario, se acercan pero no vibran, dan lugar a un sonido articulado "sordo".

3) **Por la acción del velo del paladar:** Cuando el aire sale solamente por la cavidad vocal, por hayarse el velo del paladar adherido a la pared faríngea, los sonidos emitidos son "orales" o "vocales", cuando está abierto el conducto nasal y cerrado el vocal por el descenso del velo del paladar, son producidos los sonidos nasales.

4) **Por el modo de articulación:** Las vocales se clasifican en:

1.- **Cerradas** o altas como /i/,/u/.

2.- **Medias** como /e/,/o/.

3.- **Abiertas o Bajas** como /a/.

Las consonantes se clasifican en:

1.- **Oclusivas:** Hay un cierre completo de los órganos de articulación. El aire aspirado empuja el obstáculo que cierra su salida.

2.- **Fricativas:** Si el sonido se forma por un estrechamiento de los órganos articulatorios, se produce un **ruido** de fricción.

3.- **Africadas:** Combinación de oclusivas y fricativas, dándose un cierre completo de los órganos articulatorios seguido de una pequeña abertura donde se desliza el aire contenido.

4.- **Laterales:** La corriente fonatoria se escapa por uno de los dos lados de la lengua, ocupando ésta la línea media del canal bucal en contacto con el paladar.

5.- **Vibrantes:** Producidas por interrupciones intermitentes del aire sonoro por una serie de vibraciones de la punta de la lengua.

5) **Por el lugar de la articulación:** (de gran importancia en la corrección de las dislalias).

Las vocales se dividen en:

1.- **Anteriores** como /i/,/e/.

2.- **Posteriores** como /u/, /o/.

3.- **Centrales** como /a/.

Las consonantes se dividen en:

1.- **Bilabiales:** Cuando los labios se ponen en contacto.

2.- **Labiodentales:** Entre dientes y labio inferior.

3.- **Linguodentales o dentales:** Apoyando la punta de la lengua contra la parte interna de los incisivos superiores e inferiores.

4.- **Linguointerdentales o interdentales:** Cuando la punta de la lengua se localiza entre los incisivos superiores e inferiores.

5.- **Linguoalveolares o alveolares:** Cuando la punta de la lengua se apoya en los alvéolos.

6.- **Linguopalatales o palatales:** Cuando la lengua se adhiere a la parte media y anterior del paladar duro, dejando en medio un pequeño canal por donde pasa el aire.

7.- **Linguovelares o velares:** Cuando se acerca el postdorso de la lengua al paladar blando o velo del paladar.

La **figura III.3** nos muestra la clasificación de las consonantes en español.

FONEMAS DE CONSONANTES EN ESPANOL

oclusiva	linguodental			linguovelar			bilabial												
	sonoro		sordo	sonoro		sordo	sonoro		sordo										
	/d/		/t/	/g/		/k/	/b/		/p/										
fricativa	labiodental			linguovelar			linguopalatal			ling.intr.dental			ling.alveolar						
	sonoro		sordo	sonoro		sordo	sonoro		sordo	sonoro		sordo	sonoro		sordo				
		/f/			/j/		/y/				/z/			/s/					
africada	linguopalatal																		
	sonoro		sordo																
		/ch/																	
nasal	linguopalatal			bilabial			ling.alveolar												
	sonoro		sordo	sonoro		sordo	sonoro		sordo										
	/n/			/m/			/n/												
lateral	linguopalatal			ling.alveolar															
	sonoro		sordo	sonoro		sordo													
	/ll/			/l/															
vibrante simple	ling.alveolar																		
	sonoro		sordo																
	/r/																		
vibrante multiple	ling.alveolar																		
	sonoro		sordo																
	/rr/																		

FIGURA III.3

CLASIFICACION DE FONEMAS CONSONANTICOS

	SONOROS								SORDOS	
	ORAL					NASAL	CONTINUA	INTERRUPTO	CONTINUA	INTERRUPTO
		LIQ.			NLIQ.					
		LAT.	CENTR.							
			cort.	larg.						
LABIALES	b					m	f	p		
DENTO-ALVEOLAR		l	r	rr	d	n			s	t
PALATAL	y					ñ			ch	ch
VELAR	g	g	g	g	g	g	g	g	j	k

FIGURA III.4

III.3 CLASIFICACION DE CONSONANTES DE ACUERDO A SUS PUNTOS DE ARTICULACION

1.- **Labial** /b/,/p/,/h/,/m/.

2.- **Dentoalveolar** /n/,/d/,/t/,/s/,/r/,/rr/,/l/.

3.- **Palatal** /y/,/ch/,/ñ/.

4.- **Velar** /g/,/k/,/j/.

Otra clasificación de las consonantes se muestra en las **figuras : III.4, III.4.a. y III.4.b.**

Se puede observar que en el idioma Español, la letra "**h**" no le corresponde ningún fonema, pues no representa ningún sonido. Aún cuando la letra "**w**" se aplica a palabras extranjeras, el fonema correspondiente a esta letra, en el idioma Español, es igual al fonema */u/* de la letra "**u**".

A la letra "**x**", en México, le corresponden tres fonemas:

1.- */j/*, como en la palabra "**México**" la pronunciación es */Méjico/*.

2.- */ks/*, como en la palabra "**éxito**" la pronunciación es */éksito/*.

3.- */sh/*, como en la palabra "**Xolotl**" que se pronuncia */Sholotl/*.

También se puede observar que en las combinaciones "**gu**" al ir seguidas de las letras "**e**" o "**i**", la "**u**" de las sílabas "**gue**","**gui**" es muda, es decir, la construcción fonética corresponde a */ge/,/gi/*, respectivamente. En cambio, si la "**u**" tiene diéresis como "**gü**", y si va seguida de "**e**" o de "**i**", las sílabas "**güe**","**güi**" se pronuncian como */gue/,/gui/*,respectivamente.

III.4 MECANISMOS DE PRODUCCION DE VOZ

En esta sección se explica el funcionamiento de los mecanismos de producción de voz para el *Análisis y la Síntesis de la Señal de Voz*.

La producción de voz se realiza a través de los siguientes mecanismos: las **cavidades supraglóticas**, las **cuerdas vocales** y el aire expelido por los **pulmones**.

CLASIFICACION DE FONEMAS VOCALICOS

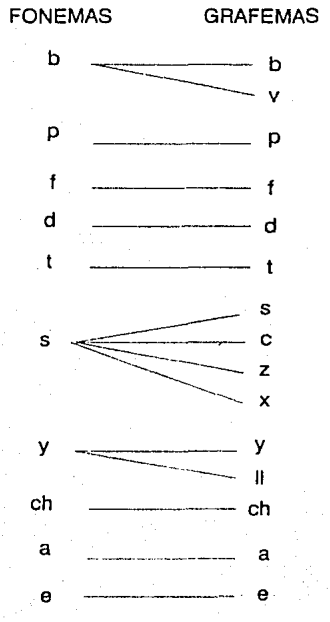
	anteriores	centrales	posteriores
cerrados	/i/		/u/
medios	/e/		/o/
abiertos		/a/	

labializados ——— /o/ , /u/

no labializados ——— /e/, /i/

neutro ——— /a/

FIGURA III.4.A



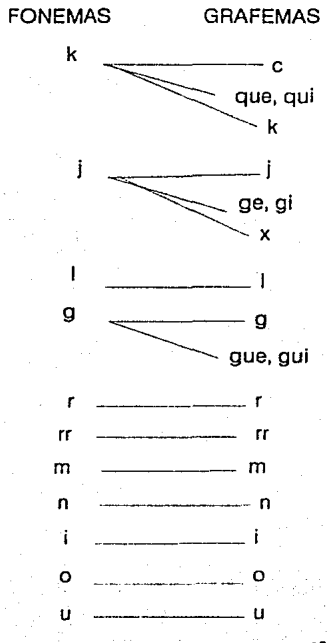


FIGURA III.4.B

Como modelo más sencillo del aparato fonador se puede tomar un *tubo de longitud y sección uniforme sin pérdidas, totalmente abierto en el final*. En esta situación los ***polos*** corresponden a aquellas frecuencias que producen ondas estacionarias, con amplitud de presión máxima en la **glotis** y nula en los **labios**, es decir aquellas para las cuales ***L*** es *1/4, 3/4, 5/4, etc.*. de la **longitud de onda**.

$$L = \frac{2n + 1}{4} \times \text{lambda}$$

n : Número de polos.

lambda : longitud de onda.

Para un varón adulto la longitud del aparato fonador es de unos 17 **cm** lo que siguiendo este modelo daría unos formantes situados en 800, 1500, 2500 ... **Hz**, que corresponden bastante bien con los observados en la pronunciación de la vocal (sonido intermedio entre */e/* y */a/* que no existe en español), para la cual el aparato fonador adopta la forma más *tubular* posible.

Estos ***formantes*** son alterados modificando, mediante estrechamientos y ensanchamientos, la forma de las **cavidades supraglóticas**, utilizando la **mandíbula inferior**, el cuerpo y la punta de la **lengua**, los **dientes** y los **labios**. Los ***formantes*** varían de posición sin guardar relación fija entre ellos. El primero se suele situar entre 200 y 1000 **Hz**., el segundo entre 500 y 2500 y el tercero entre 1500 y 3500 Hz.

En los **sonidos nasales** (como */m/*, o */n/*) la **boca** permanece cerrada pero el **velo del paladar** permite la transmisión hacia el **orificio nasal** por donde se produce la ***radiación***. En estos casos aparecen también los ***formantes (polos de transmisión)***, pero además ***antiformantes (ceros de transmisión)***, debidos a que la cavidad bucal, aunque cerrada, está acoplada lateralmente a las cavidades en juego introduciendo atenuaciones importantes a ciertas frecuencias. Algún ***antiformante*** puede coincidir con un ***formante*** anulándose los efectos.

La **excitación fricativa** se produce generalmente en algún *punto intermedio* del **aparato fonador**. En estas condiciones las cavidades posteriores introducen también antiformantes en la transferencia de la fuente a la apertura que, de nuevo pueden anular o atenuar el efecto de los formantes, o introducir atenuaciones fuertes en ciertas frecuencias o en los extremos de la banda (por ejemplo, atenuación relativa de las bajas frecuencias, debida a un cero de segundo orden por debajo del primer formante).

La **figura III.5** muestra el tracto vocal y el tracto nasal como tubos no uniformes en una área de corte esquematizada.

El conjunto de las cavidades supraglóticas constituye un *resonador o un filtro acústico* que conforma el espectro de la excitación aplicada que puede ser, por ejemplo, un tren de impulsos generados por la vibración de las cuerdas vocales. La energía del sistema es suministrada por los pulmones al tratar de expeler aire.

Existen básicamente tres tipos de excitación, que además pueden aparecer simultáneamente.

En primer lugar, se considera la vibración de las cuerdas vocales, ya citada. Esta vibración se produce al tratar de expirar y mantener la glotis cerrada. La presión subglotal llega a ser suficiente para separar las cuerdas vocales lo que provoca la salida de un pulso de aire al mismo tiempo que se reduce la presión y permite que los ligamentos vocales se vuelvan a cerrar, repitiéndose así el ciclo. La onda generada por esta oscilación tiene una forma aproximadamente triangular, de frecuencia entre 100 y 200 **Hz**, y ciclo de trabajo entre 0.3 y 0.7 **Hz** según la tensión de las cuerdas. El *espectro* se compone de la *fundamental y las demás armónicas con amplitud decreciente* unos 12 **dB/octava**. Este período fundamental caracteriza la *altura tonal* de los sonidos articulados llamados **sonoros** (frente a los **no sonoros o sordos** en que no se produce vibración de las cuerdas) y recibe en la literatura inglesa el nombre de "**pitch**" siendo éste uno de los parámetros más importantes en *Análisis / Síntesis de señales de voz*.

Un segundo tipo de excitación vocal consiste en la generación de una *turbulencia de aire* en algún estrechamiento del **aparato fonador** (en la pronunciación de la "j", por ejemplo) produciéndose un *ruido acústico* aproximadamente *blanco* (espectro plano en el margen de audio). Este es el origen de los **sonidos fricativos**.

Un tercer tipo de excitación corresponde a los **sonidos oclusivos**: el aparato se mantiene cerrado en algún punto mientras se crea una *presión de aire por detrás* (en la "p", por ejemplo). La *apertura brusca de la oclusión* produce un efecto semejante a una **excitación en escalón** de presión con un espectro que cae inversamente con la frecuencia.

La excitación vocal es aplicada a las **cavidades supraglóticas** que filtran su espectro y conforman la señal. La transmisión desde la **glotis** a la apertura labial puede ser modelada por una serie de *cavidades resonantes*, obteniéndose una ***Función de Transferencia*** producto de *términos resonantes de segundo orden* del tipo:

$$H(s) = \frac{A(s)}{\prod_{n} (s - s_o)(s - s_o)}$$

es decir, caracterizada por una serie de ***polos complejos conjugados*** correspondientes a los *modos normales de transmisión*. En los espectros de los sonidos articulados se observan en efecto una serie de **picos** a ciertas frecuencias que se denominan ***formantes***.

MODELO TUBULAR DEL TRACTO VOCAL

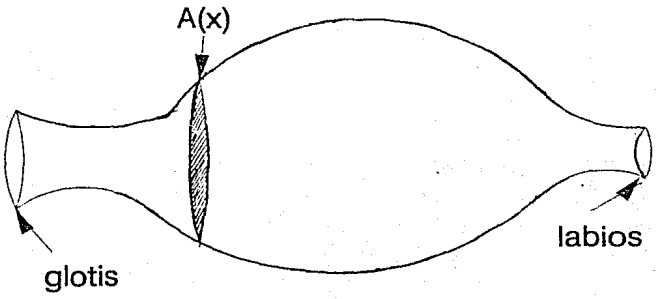


FIGURA III.5

Las vocales son producidas por pulsos casi periódicos. En la **figura III.6** se puede apreciar la forma de onda de las vocales. Como se puede observar la vocal */i/*, por ejemplo, muestra una oscilación de baja frecuencia atenuada, sobre la cual se sobrepone una oscilación de alta frecuencia relativamente fuerte. Esto consiste de un primer formante bajo y de un segundo y tercer formantes altos. En contraste con la vocal */u/* que muestra una energía relativamente baja de alta frecuencia, como una secuencia del primer y segundo formante de baja frecuencia.

En la **figura III.7** se muestra un ejemplo de forma de onda de dos consonantes nasales combinadas de la forma vocal-consonante nasal-vocal.

En la **figura III.8** nos muestra un ejemplo de forma de onda de las consonantes fricativas */f/, /s/* y */sh/*.

III.5 TEORIA ACUSTICA DE LA PRODUCCION DE VOZ

En este aspecto, el punto más importante es la obtención de modelos discretos en el tiempo para representar señales muestreadas a partir de la señal de voz. Para ésto, es necesario considerar los diferentes tipos de sonidos producidos por la voz y adecuar estas consideraciones a la fonética particular de cada idioma.

Si se desea profundizar en este tema se tendría que considerar la propagación del sonido en el tracto vocal, su analogía con una línea de transmisión y un análisis del estado estable del sistema de articulación de los fonemas o sonidos. De esta forma se puede obtener un modelo lineal, invariante en el tiempo, excitado por una fuente generadora de ruido o por secuencias de pulsos casi periódicos, a partir de ésto se pueden hacer modelos matemáticos, discretos en el tiempo, de la señal de voz.

Las ondas sonoras son creadas por vibración y son propagadas en el aire o por medio de vibraciones de las partículas del medio. Las leyes de la física son la base para describir la generación y propagación del sonido en el sistema vocal. Las leyes físicas, como un conjunto de ecuaciones diferenciales parciales, pueden ser obtenidas para describir el movimiento del aire en el sistema vocal. La teoría acústica, en detalle, debe considerar los siguientes efectos:

1.- Variación en el tiempo de la forma del tracto vocal.

2.- Pérdidas debido a la conducción de calor y a la fricción viscosa en las paredes del tracto vocal.

3.- La suavidad de las paredes del tracto vocal.

4.- La radiación del sonido en los labios.

5.- Acoplamiento nasal.

6.- Excitación del sonido en el tracto vocal.

FORMA DE ONDA ACUSTICA DE LAS VOCALES

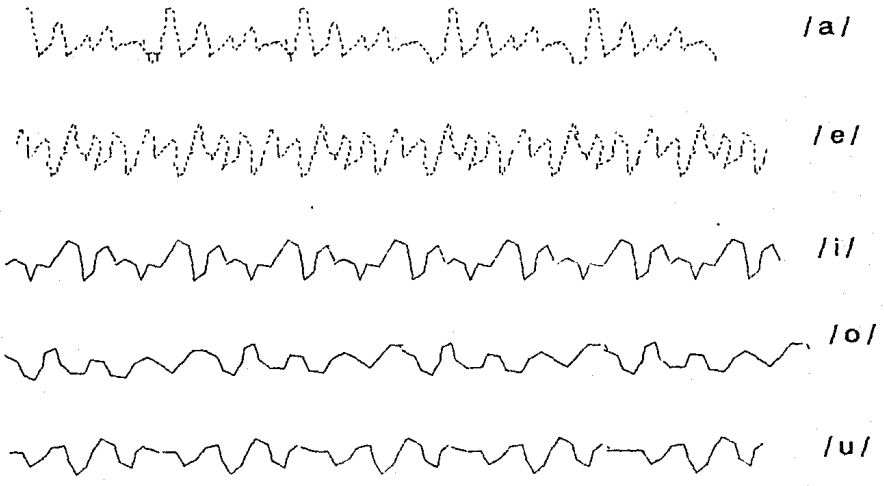


FIGURA III.6

FORMA DE ONDA ACUSTICA NASAL

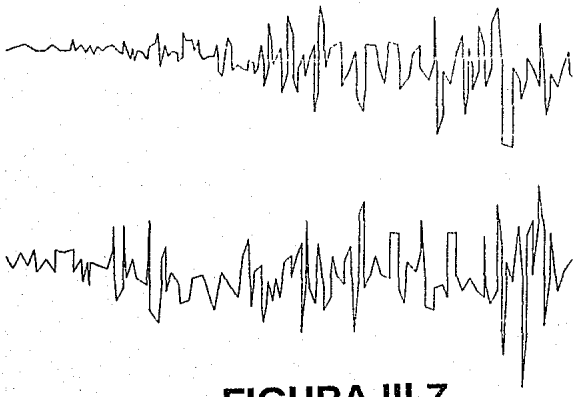


FIGURA III.7

FORMA DE ONDA ACUSTICA FRICATIVA

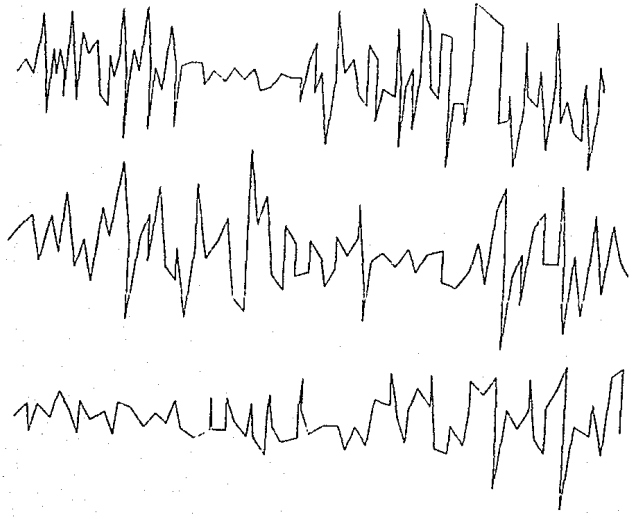


FIGURA III.8

La configuración física más simple, que tiene una interpretación muy útil en términos del proceso de la producción de la voz, se describe en la **figura III.5**, cuya referencia se hizo en la página número 17. El tracto vocal es modelado como un tubo no uniforme, variante en el tiempo y de sección transversal. Para frecuencias que corresponden a las longitudes de onda menores a 4000 Hz., es razonable asumir la propagación de la onda plana a través del tubo. No existen pérdidas debido a la viscosidad o a la conducción térmica ni en las paredes del tubo.

Con estas consideraciones, mas las leyes de la conservación de la energía, la masa y el momentum, se ha demostrado que las ondas sonoras en el modelo del tracto vocal satisfacen el siguiente par de ecuaciones:

$$\frac{\partial p}{\partial x} = \frac{\partial (u/A)}{\partial t}$$

$$\frac{\partial u}{\partial x} = \frac{1}{c^2} * \frac{(rho)*A}{t} + \frac{A}{t}$$

donde:

p = p(*x*,*t*) es la variación en la presión del sonido en el tubo en una posición *x* y un tiempo *t*.

u = u(*x*,*t*) es la variación en volumen de la velocidad del fluido en una posición *x* y un tiempo *t*.

(***rho***) es la densidad del aire en el tubo.

c es la velocidad del sonido.

A = A(*x*,*t*) es la función de área del tubo, es decir, el valor del área de la sección transversal normal a los ejes del tubo, a lo largo del tubo y como una función del tiempo.

Las soluciones numéricas pueden ser obtenidas de cualquier forma. La solución completa de las ecuaciones diferenciales requiere que la velocidad de la presión y del volumen sean obtenidas mediante los valores de *x* y de *t* en la región que se encuentra entre la **glotis** y los **labios**.

En la **figura III.5**, cuya referencia se da en la página número 17, muestra el área del tubo en función del tiempo, a un tiempo en particular, para un sonido contínuo, es lógico asumir que A(*x*,*t*) no cambia en el tiempo.

III.5.1 PRODUCCION Y PROCESAMIENTO DE LA SEÑAL DE VOZ

En los sistemas de comunicación hablada, la señal de voz es transmitida, almacenada y procesada en varias formas. Se deben tomar en cuenta dos factores importantes para cualquier sistema:

1.- Preservación del mensaje contenido en la señal de voz.

2.- Representación de la señal de voz en forma conveniente para la transmisión y almacenamiento de la misma , de tal forma que sea flexible en caso de modificaciones.

Existen varios métodos de representación de la señal de voz:

2.1.- Representación de forma de onda.

La forma de onda de la señal analógica de la voz se obtiene a través del muestreo y de un proceso de cuantización.

2.2.- Representación Paramétrica.

Es aquella basada en el análisis de la onda y del espectro de la señal de voz, obteniéndose los siguientes tipos de parámetros.

2.2.1.- Parámetros de excitación.

2.2.2.- Parámetros del tracto vocal referidos a fonemas.

III.5.2 PROCESAMIENTO DE LA SEÑAL DE VOZ EN FORMA DIGITAL

Las técnicas de procesamiento de señales digitales se aplican a problemas de procesamiento de voz como simulaciones de sistemas analógicos complejos. El punto de vista inicial es que estos sistemas analógicos pueden ser simulados en una computadora para evitar la necesidad de construir los sistemas experimentales con parámetros y otras consideraciones de diseño. Los sistemas digitales son tan confiables y compactos que han logrado incrementar el avance tecnológico.

Existen muchas otras razones para usar técnicas digitales en sistemas de comunicación de señal de voz, por ejemplo: Si es usado un código ajustable, la voz en forma digital puede ser transmitida en cualquier canal con ruido. Por otra parte teniendo una transmisión de la señal de voz algunas veces se requiere seguridad, es por eso que la representación digital toma ventajas sobre los sistemas analógicos, ya que permite encriptar la informacion de la señal de voz.

Las técnicas de procesamiento digital de la señal aplicadas a problemas de comunicación, comprenden tres tópicos principales que son:

1) La representación de la señal hablada en forma digital.

2) La implementación de técnicas de procesamiento sofisticado.

3) Las aplicaciones que se pueden obtener.

En la **figura III.9** se muestra la manipulación y proceso de la información.

III.5.3 ESPECTRO DE LA SEÑAL DE VOZ.

El oído humano puede captar sonidos a frecuencias que varían aproximadamente entre 20 **ciclos** por **segundo** hasta 20000 **ciclos** por **segundo**, aunque la mayor parte de la gente tiene una menor percepción que ésta.

Cuando se habla de un tono de cierta frecuencia, se quiere decir que el aire está vibrando a ese número de oscilaciones por segundo. Para poder transmitir ese sonido, el transductor de un teléfono lo convierte a una cantidad de oscilaciones eléctricas de una gama equivalente a las frecuencias de la voz humana, aunque a menudo se cambian esas frecuencias para fines de transmisión.

Tomando en cuenta el rango de 300 **Hz** a 3000 **Hz** de la transmisión de la señal de voz por vía telefónica, se puede diseñar cualquier sistema de comunicación hablada que interactúe con el usuario, bajo esos rangos.

MANIPULACION Y PROCESAMIENTO DE LA SEÑAL

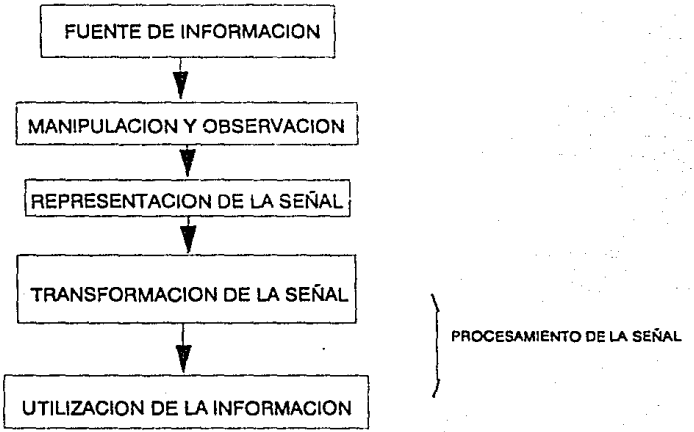


FIGURA III.9

CAPITULO IV

IV TECNICAS DE PROCESAMIENTO DE LA SEÑAL DE VOZ EN FORMA DIGITAL

En este capítulo se expondrán algunas de las técnicas usadas para procesar la señal de voz en forma digital, algunas de ellas, a pesar de estar basadas en sólidos conceptos matemáticos, representan un tiempo de procesamiento muy grande para un microprocesador, como el microprocesador **8086**, otros más "empíricos" poseen características que permiten ser procesados en menos tiempo. A continuación se expondrán las *Propiedades Básicas de la Señal de Voz* y las diferentes *Técnicas de Síntesis de Voz*, mediante *Microfonemas* y diferentes *Métodos Vectoriales de Cuantización*, así como los *Conceptos Básicos de Representación de la Señal en el Tiempo y Frecuencia, Muestreo y Cantidad de Información*.

IV.1 PROPIEDADES BASICAS DE LA SEÑAL DE VOZ

La señal de voz contiene una considerable *redundancia de información*, a causa de los mecanismos físicos del tracto vocal y de la estructura del lenguaje. El hombre al producir sonidos puede emitir o percibir un rango dinámico relativamente amplio de sonidos. La ventaja de esto es que al tener esa redundancia de información es posible compactar tomando sólo los bits necesarios para codificar y transmitir señales de voz.

La voz es generada por sonidos provocados por una excitación, en donde se puede apreciar una serie de pulsos en las regiones propias de la señal de voz (***regiones sonoras***), que es cuando las cuerdas vocales están vibrando, mientras en las ***regiones no sonoras***, existe una contracción en el tracto vocal.

El tracto vocal actúa como un filtro acústico variable en el tiempo en donde la excitación se representará espectralmente. Cuando se presenta un largo período de tiempo (1/2 **s.**) la voz llega a ser un proceso altamente *no estacionario* debido a la variación de amplitud, resultado de las formas sonoras y no sonoras, silencios y de la variación en el tiempo del tracto vocal.

Sin embargo, en períodos cortos de tiempo (de 20 **ms** a 40 ms.), la voz es *localmente estacionaria* y tiene muy bien definido su espectro de tiempo corto. Las **figuras IV.1, IV.2.a y IV.2.b** muestran modelos espectrales de tiempo corto para segmentos sonoros y no sonoros de voz.

EJEMPLOS DE SEGMENTO DE ONDA SONORA EN TIEMPOS CORTOS Y LARGOS

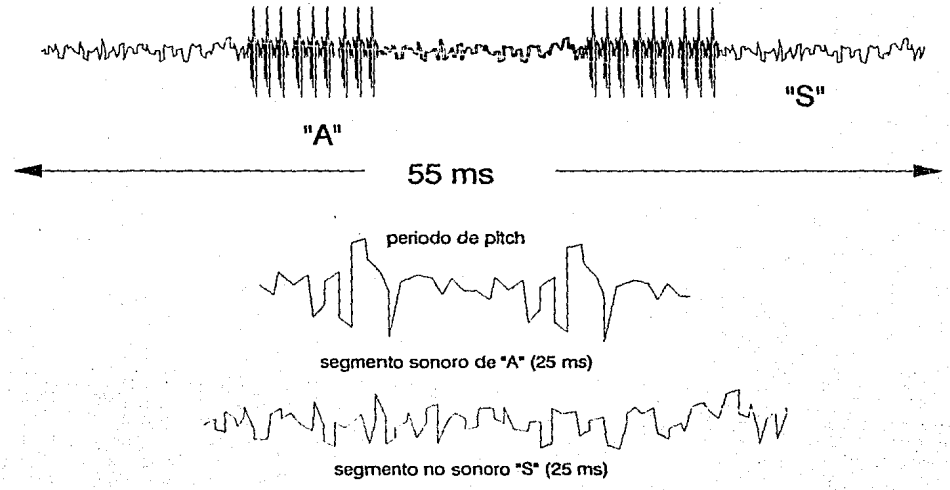


FIGURA IV.1

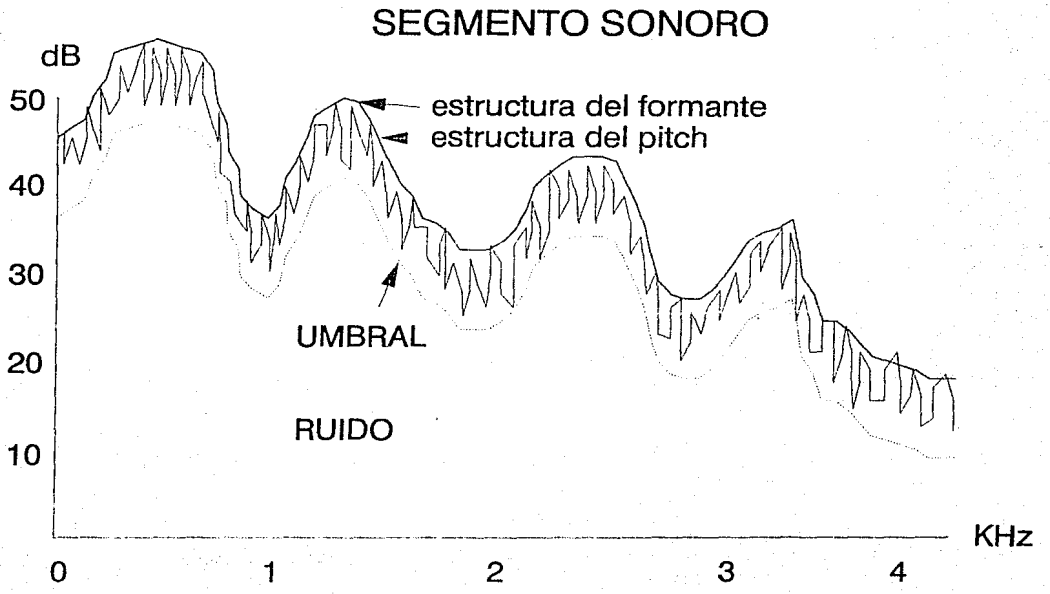

SEGMENTO SONORO
dB
50
40
30
20
10
estructura del formante
estructura del pitch
UMBRAL
RUIDO
KHz
0
1
2
3
4

FIGURA IV.2.A

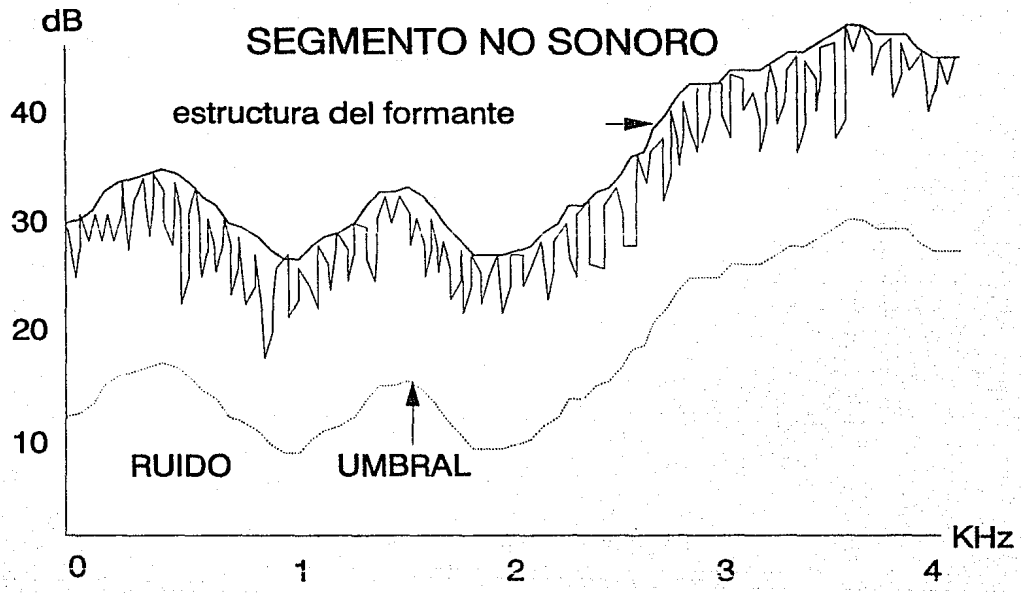

dB
SEGMENTO NO SONORO
estructura del formante
40
30
20
10
RUIDO
UMBRAL
KHz
0
1
2
3
4

FIGURA IV.2.B

La ***envolvente del espectro*** es determinada primordialmente por la frecuencia que es la respuesta del tracto vocal (*filtro*). La ***resonancia*** de este filtro se observa en forma de *picos* bajo la envolvente espectral, los cuales reciben el nombre de ***formantes***. Básicamente se observan cerca de 4 tipos de resonancias en el espectro de voz de 0 **Hz** a 4 **KHz**. En las ***regiones sonoras***, se puede observar que la forma es más o menos *periódica en el tiempo*, y esto también se refleja en el espectro como una *estructura armónica en frecuencia*. En las ***regiones no sonoras***, la estructura pura tiene una apariencia de *ruido en tiempo y frecuencia*.

Cualquier ruido o distorsión que sea añadido a la señal, no es percibido si éste cae bajo el umbral.

Las propiedades más importantes de la producción y percepción de voz son:

1) En Producción de voz :

1.- Tiempo-Largo no estacionario.

1.1. - Amplitud no estacionaria.
1.2. - Regiones sonoras / no sonoras / silencios.
1.3. - Estructura fonética y lenguaje.

2.- Tiempo-Corto estacionario.

2.1. - Estructura de tipo formante (correlación en tiempo corto).
2.2. - Estructura de voz (**pitch**) sonora.
2.3. - Estructura de ruido (no sonora)

2) Percepción de la voz :

1. - Rango espectral dinámico local.

2. - Máscara auditiva (efecto de enmascaramiento auditivo).

Algunas de las propiedades de la señal de voz antes mencionadas hacen posible la ***Compactación de Información***. Existe una gran variedad de técnicas que pueden emplearse para codificar la señal de voz, tomando en cuenta las ventajas de sus propiedades.

En la **figura IV.3** se puede observar el resultado obtenido de la comparación de diferentes técnicas de compactación elevando la calidad de la voz a razón del número de bits.

La **escala vertical** representa una *hipotética medida* de la **calidad de voz**, donde el valor "**1**" implica una calidad que es muy semejante a la señal de entrada para un ancho de banda de (**200 Hz - 3200 Hz**) en una señal telefónica. El valor "**0**" implica una calidad sumamente pobre e inentendible.

CALIDAD VS CANTIDAD DE BITS CODIFICADOS

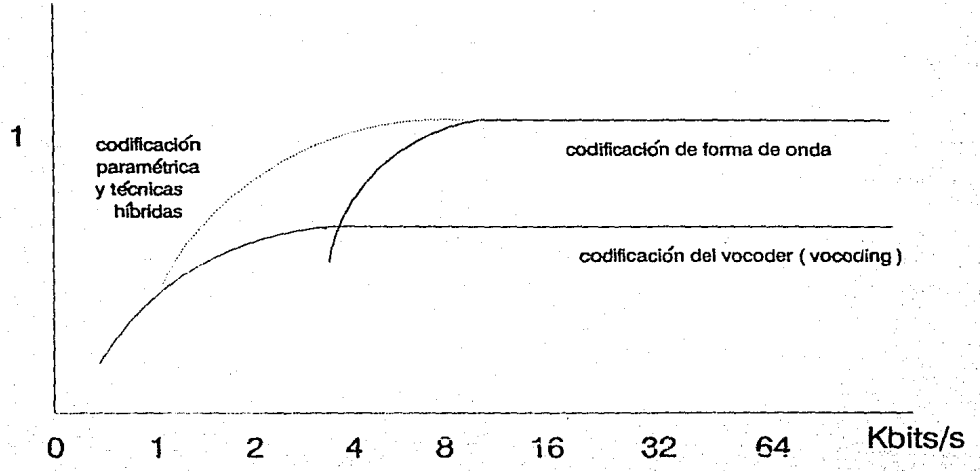


FIGURA IV.3

La **calidad** es un *atributo multidimensional* y el definirla es mucho más complejo que representarla en una simple escala.

Una simple codificación de forma de onda a razón de 64 **Kb/s** puede lograr una buena calidad, si se toma en cuenta la ventaja que puede tener la *amplitud no estacionaria* (con una ganancia variable en el tiempo). Si se tienen las características de la *forma espectral dinámica* (tiempo de correlación local), o *estática*, se puede conseguir buena calidad en la voz y una razón de bits en el rango de 24 **Kb/s** a 32 **Kb/s**.

IV.2 SINTESIS DE FONEMAS MEDIANTE ARTICULACION DE MICROFONEMAS

Desde el punto de vista fonético la unidad más elemental del lenguaje es el fonema. Sin embargo, examinando detalladamente la forma de onda correspondiente a un fonema puede llegarse a la definición de un unidad más elemental, el microfonema, el cual es apropiado para la síntesis del fonema y la compactación de la información.

En la definición del microfonema distinguimos tres casos, según el tipo de articulación del fonema correspondiente:

1.-**Fonemas sonoros:** La forma de onda es casi periódica (como se ve en la **figura IV.4**) en consonancia con la vibración de las cuerdas vocales. En este caso el microfonema será un *periodo básico*, o intervalo entre dos aperturas consecutivas de las cuerdas vocales.

2.-**Fonemas oclusivos:** Estos sonidos se originan por una excitación del aparato fonador y son de corta duración. El microfonema y el fonema coinciden.

3.-**Fonemas fricativos sordos:** Por un lado estos sonidos son de larga duración y por otro su estructura es muy aleatoria. La solución adoptada ha sido la de tomar un segmento de la forma de onda como microfonema representativo. La longitud de este segmento requiere un compromiso entre calidad y cantidad de información a almacenar (ver "s" en la **figura IV.5**).

El número de bits necesarios para una codificación PCM de los microfonemas depende de la frecuencia de muestreo y la *frecuencia fundamental* o "**tono**".

Para una frecuencia de muestreo de 10 **KHz** y un rango de tonalidades de 5 **ms** a 10 **ms** son necesarios de 50 a 100 muestras por microfonema.

La síntesis de palabras aisladas mediante microfonemas se efectúa mediante concatenación de éstos con las amplitudes adecuadas.

SEGMENTO DE ONDA DE LA PALABRA "DOS"
CORRESPONDIENTE A LA /O/

MICROFONEMA CORRESPONDIENTE A LA /O/

FIGURA IV.4

En la **figura IV.5** se presenta la gráfica de la forma de onda de la palabra "**TRES**". Pueden distinguirse los cuatro segmentos correspondientes a los fonemas /t/,/r/,/e/ y /s/, así como cierta modulación de amplitud.

Por tanto, a partir de la especificación fonética de una palabra y de su envolvente de amplitud, puede ésta sintetizarse, según se muestra en la **figura IV.5**. En la misma figura, pueden verse las formas de onda de los microfonemas a partir de los cuales se sintetiza "**TRES**" así como su palabra de origen.

En resumen, la síntesis por articulación de microfonemas se efectúa a partir de la codificación de los microfonemas y de una especificación que comprende tipos de fonemas, duración y envolvente de amplitud para cada palabra del vocabulario a sintetizar. La aplicación de este método de síntesis conduce a una reducción drástica de la cantidad de información a almacenar para la generación de un vocabulario finito.

IV.3 FORMAS DE REPRESENTACION DE UNA SEÑAL EN EL TIEMPO

En síntesis, las diferentes formas de representación de una señal en el tiempo hacen uso de una o más de las siguientes características:

1.- Forma de Onda:

1.1 Razón de cruces por cero: Es la cantidad de veces en que el valor de la señal coincide con el eje de las abscisas durante el muestreo de la señal en el dominio del tiempo.

1.2 Energía: Es el valor de la minimización del Error Cuadrático Medio de Aproximación que se entiende como la mínima diferencia entre dos muestras consecutivas.

1.3 Función de autocorrelación: Es la relación de afinidad o proximidad que debe existir entre las muestras para que se pueda tomar un criterio de discriminación, el cual es aplicable en reconocimiento de señales de voz.

2.- Parámetros : Al hacer un análisis, utilizando cualquiera de los criterios de análisis de forma de onda, se utilizan los siguientes parámetros :

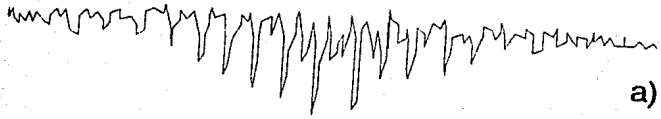
2.1 Razón sonidos sonoros / sordos.

2.2 Tono o "Pitch".

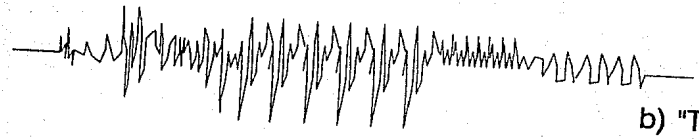
2.3 Intensidad.

2.4 Modo de excitación: glotal, fricción, radiación, etc.

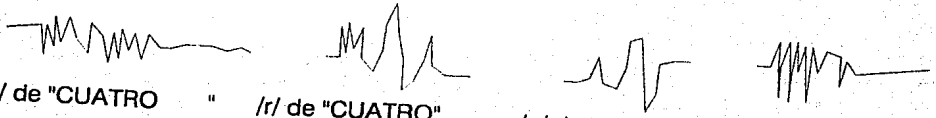
2.5 Parámetros del tracto vocal: cambian lentamente en el tiempo.



a) "TRES" natural



b) "TRES" sintético



/t/ de "CUATRO " /r/ de "CUATRO" /e/ de "CERO" /s/ de "SEIS"

FIGURA IV.5

IV.4 REPRESENTACIONES DIGITALES DE LA FORMA DE ONDA DE LA SEÑAL DE VOZ

Para sistemas digitales, la forma de onda puede ser representada por una secuencia de números, los cuales especifican los modelos de fluctuación de amplitud.

La forma de onda de la señal de voz puede ser muestreadaperiódicamente en el tiempo, y para reducir una secuencia de muestras **Xa**(*nT*) (donde "**X**" es la muestra; "**a**" el número de la muestra; "*n*" el número de período y; "*T*" es el período), es necesario cuantizarlas en un conjunto *finito* de valores para poder obtener una representación digital.

La forma de onda puede procesarse en 2 pasos :

1. - **Muestreo.**

2. - **Cuantización.**

IV.4.1 MUESTREO DE LA SEÑAL DE VOZ

El muestreo de la señal analógica será limitado por un ancho de banda, y la razón de muestras será de por lo menos dos veces la frecuencia máxima (***frecuencia de Nyquist***). Además, para poder realizar una **conversión analógica/digital**, deben considerarse las propiedades espectrales de la señal.

Para poder representar el sonido de la señal de voz se puede muestrear a razón de más de 20 **KHz**. En el proceso de muestreo se tomarán en cuenta las frecuencias de los primeros 3 *formantes* de la señal de voz, y sólo interesa la porción espectral hasta de los 3.5 **KHz** aproximadamente. Después de esto, si la señal es filtrada por un filtro analógico antes de ser muestreada la ***frecuencia de Nyquist*** es de 4 **KHz**, y así se podrá tener un muestreo a razón de 8 KHz.

Aún cuando la forma de onda de la señal pueda tener un espectro de banda limitada, la señal puede ser afectada por un **ruido** de ancho de banda aleatorio. En este caso la señal será procesada por un **filtro paso-bajas** y así cortará el **ruido** de alta frecuencia.

IV.4.1.1 CANTIDAD DE INFORMACION DE LA SEÑAL DE VOZ

La capacidad humana de producir informacion mediante emisión de sonidos está acoplada en cierta medida con el ancho de banda de los canales de percepción. Se han desarrollado diversos experimentos para determinar el caudal de información que puede procesar un individuo encontrándose un límite superior en los 50 **bits/s**.

En el caso de pronunciación rápida de fonemas equiprobables casi toda la capacidad de percepción se gasta en recibir especificación fonética correcta. En cambio, en una conversación social con restricciones y redundancias se dispone de un margen amplio para atender a características personales y de articulación.

La relación entre **información** y **redundancia** en la señal de voz muestra las alternativas de codificación de la señal, tomando en cuenta la *cantidad de información no redundante*, así como los parámetros que la definen, dando como resultado mejores técnicas de codificación y compactación de la señal de voz.

Desde la invención del **VOCODER** en 1928 hasta fines de los 60's todas las investigaciones sobre la naturaleza de la voz tenían como finalidad reducir el ancho de banda en los sistemas de transmisión telefónicos. A partir de mediados de los 60's cambió el panorama con la introducción del tema de la *comunicación hombre-máquina* mediante voz. Las ***Técnicas Digitales de Análisis / Síntesis de Voz*** se desarrollaron en el campo de la ***Transmisión Telefónica*** y más concretamente en el tema de la ***Compactación de Banda***.

Como preámbulo al estudio de los métodos de compactación de banda en la transmisión de voz, según **Duddley**, el inventor del **VOCODER**, para el idioma inglés, considerando 42 fonemas, se verifica que el promedio de información por fonema es de 5.4 bits en el caso de fonemas equiprobables, y de 4.9 **bits** si se consideran sus frecuencias positivas de aparición. Añadiendo restricciones secuenciales en la selección de los fonemas impuestos por el contexto, el promedio de información por fonema puede bajar hasta 3 **bits**. Suponiendo que un locutor emite 10 fonemas por segundo el ***caudal de información*** de una conversación es de unos 50 **bits**.

Desde el punto de vista teórico, pueden transmitirse por el canal de ancho de banda de 5 **Hz**, 50 **bits/s** con una relación **señal/ruido** de 30 **dB**.

El ancho de banda utilizado en telefonía resulta de dos a tres veces el valor teórico. En realidad, considerando la transmisión fonética de la voz, es evidente que en la voz exista más información relativa a características personales del locutor y a rasgos prosódicos del mensaje que se emite.

Históricamente el canal **VOCODER** está entre los primeros tipos de técnicas de compactación de voz. El principio básico del canal **VOCODER** está enfocado al modelo de producción de voz. Los parámetros que se han definido en el filtro del tracto vocal, como son: sonidos sonoros/sordos, silencios y **pitch**, son analizados y codificados por un módulo transmisor para después ser transmitidos. Este principio fue creado desde hace 50 años y los sintetizadores modernos lo utilizan.

Investigaciones anteriores del **VOCODER** han sido principalmente enfocadas a técnicas de análisis y representación de los parámetros básicos para lograr una voz sintetizada más natural. Así mismo, nuevos métodos de codificación de la excitación de la señal para **VOCODER** prometen la obtención de una excelente calidad de voz.

Existen varios métodos de codificación de la señal para su análisis y su compactación como:

1.- Las modulaciones **PCM, DPCM, ADPCM, CVSD**, etc.

2.- Técnicas de parametrización.

3.- Técnicas de predicción lineal.

4.- Técnicas de autocorrelación.

5.- Detección del **Pitch**.

El método más simple de codificación digital consiste en la representación directa de la forma de onda temporal mediante modulación por impulsos codificados por **PCM**.

Suponiendo un ancho de banda de "W" **Hz** y una codificación de las muestras mediante palabras binarias de "B" **bits** resulta un caudal de información de "**2*B*W**" **bits/s**. Los tres primeros *formantes* de los fonemas sonoros caen por debajo de 3 **KHz** mientras que los fonemas fricativos tienen un ancho de banda de unos 10 **KHz**.

Resulta, por tanto, una frecuencia de muestreo entre 6 **KHz** y 20 **KHz**. La relación **señal/ruido** en decibeles depende del número "B" de **bits** por muestra según la relación **Señal/Ruido** "**SNR**" descrita a continuación:

$$SNR = 6*B*7.2$$

Para tener una "**SNR**" de 60 **dB** se necesita que "B" = 11, por lo que una representación adecuada **PCM** requiere de 70 **Kb/s**.

No existe flexibilidad en la reducción de la frecuencia de muestreo, por lo que las técnicas de reducción del caudal de información en la representación **PCM** se orienta a la reducción del número de **bits** por muestra. El hecho fundamental en que se basa esta reducción está en el gran margen dinámico de la señal de voz. Las técnicas **LOG PCM** se basan en la utilización de un cuantizador logarítmico reduciéndose **B** a 7 **bits/muestra**. Otras técnicas se basan en la utilización de un cuantizador que se adapta al nivel de energía de la señal.

Se puede obtener una mayor reducción basándose en el hecho de que incluso a la ***frecuencia de muestreo de Nyquist***, la correlación entre muestras sucesivas es grande.

Esta correlación se puede incrementar aumentando la frecuencia de muestreo; estas técnicas reciben el nombre de **PCM** diferencial, y la reducción en el número de **bits/muestra** viene acompañada por una mayor complejidad de los sistemas de proceso de la señal. La **figura IV.6** muestra una comparación entre las diferentes técnicas de representación de la relación **señal/ruido** contra el número de **bits** por segundo.

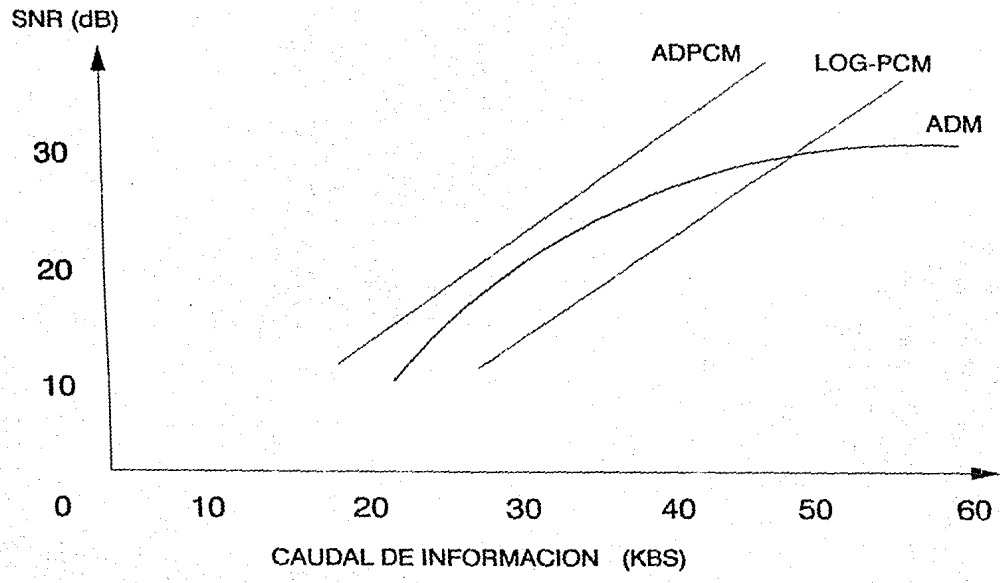

SNR (dB)
ADPCM
LOG-PCM
ADM
30
20
10
0
10
20
30
40
50
60
CAUDAL DE INFORMACION (KBS)

FIGURA IV.6

De esta figura se deduce que para una relación **señal/ruido** de 30 **dB** el caudal de información no puede reducirse por debajo de 40 **Kbs.** Para conseguir una reducción del mismo orden de magnitud o más, hay que recurrir a técnicas más complejas que tengan en cuenta el proceso de producción de la voz y la naturaleza de la señal de voz. La clave de estas técnicas será la deconvolución de la señal vocal en una excitación al aparato fonador y la respuesta en frecuencias de éste último.

IV.4.2 METODOS VECTORIALES DE CUANTIZACION

Los métodos vectoriales de cuantización permiten obtener una señal de voz con una calidad aceptable y con una razón de muestreo de 16 **Kbits/s,** además de ser métodos más sencillos que otros que pueden dar una calidad de voz semejante.

En estos métodos se utilizan vectores de cuantización, como un agrupamiento de muestras en una entidad, y además "***codebooks***" que son entidades que almacenan "***patrones***" de referencia.

Las técnicas de Cuantización Vectorial de Complejidad Reducida reducen la cantidad de código, pero requieren de más memoria y tienen una menor razón de **Señal/Ruido** "**SNR**", comparados con una búsqueda completa en un libro de códigos óptimos (***Codebooks*** **(*)**).

IV.4.2.1 CUANTIZACION VECTORIAL

Para lograr esta simplicidad, este método agrupa varias muestras codificadas en **PCM** dentro de un vector **PCM (VPCM)**. Estas muestras consecutivas de la señal son tratadas como una "entidad". A cada vector se le asigna una palabra binaria que lo identifica en forma única.

Cada vector se codifica mediante la comparación con un conjunto de vectores de referencia, a los cuales se les llama ***Vectores Código*** o ***Patrones***. Comparando los vectores de entrada con los **patrones** del libro de código se busca la mejor aproximación y así se asigna una palabra de código a cada vector.

Un **Codebook** para la señal de voz podría consistir de 64 **patrones** de seis muestras cada uno. Los **patrones** se optiman para reducir el ***error cuadrático medio de aproximación***, a las formas de onda de la voz. El tamaño del ***Codebook*** es crítico para la flexibilidad y complejidad del código.

(*)*Codebook* es un conjunto de **patrones** de referencia almacenados.

La frecuencia de transmisión puede ser calculada con la siguiente fórmula :

$$r = \frac{LOG_2\ N}{K}$$

Donde :

r : Rapidez de transmisión de código en **bits/muestra.**

N : Número de **patrones** en el libro de código.

K : Dimensión del Vector.

Nota : Se asume una frecuencia de muestreo de 8 **KHz.**

Para la suposición anterior **K** = 6, **N** = 64, entonces r = 1 **bit/muestra.**

Para el **VPCM** la razón **Señal/Ruido SNR** es :

$$SNR = 6 * \frac{LOG_2\ N}{K} + C_K$$

donde :
C_K : Es una constante dependiente de K.

SNR y C_K se valúan en **dB.**

De la ecuación anterior la **SNR** para **VPCM** se incrementa **6/K dB** cada vez que se duplica el tamaño del ***Codebook***. Pasa algo muy parecido en el **PCM** que se incrementa su **SNR** en 6 **dB** cada vez que se aumenta un bit por muestra, pero **VPCM** lo aventaja ya que **C(*K*)** es alto si se aprovecha adecuadamente la correlación entre las componentes del vector.

Ya que las multiplicaciones son, por lo general, las operaciones que demandan mayor tiempo de proceso comparadas con las sumas, el número de multiplicaciones puede ser un parámetro útil en la estimación de la complejidad del sistema. Otro parámetro a considerar es la dimensión "K" de los vectores, ya que un cuantizador de vector de dimensión "**K**" a "**r**" **bits/muestra** requiere de 2^(K*r) , ("^" indica elevación de potencia), cálculos por muestra, y una memoria de K*2^(K*r) palabras, por esta razón el uso directo de "**VQ**" (**Vector de Cuantización**) debe de tomar un factor **Kr** <= 8.

Se puede asumir dentro de un marco de duración de 10 **ms** a 20 **ms** que la forma de onda de la señal de voz es localmente estacionaria. A través de la Cuantización Vectorial Adaptiva se puede acoplar el "***codebook***" a cambios de la forma de onda por medio de una rutina de aprendizaje de la secuencia en particular que se acaba de registrar en la forma de onda y renovar los modelos del "***codebook***" si es necesario.

El ***código de sub-banda*** está provisto de una técnica poderosa para la codificación de voz. Esta técnica está basada en la estrategia de alojar **bits** de diferente ancho de banda teniendo cierto criterio. El **VQ** ha logrado cortar a la mitad la cantidad de **bits**, conservando los necesarios para una buena información y sin perder la calidad.

Los "***codebooks***" pueden estructurarse en *forma de árbol*. Se utilizan para la ***Codificación Lineal Predictiva***, reducen favorablemente la complejidad en la codificación a expensas de lo que se necesite en memoria. Nesecita un algoritmo de búsqueda óptimo. Es algo así como el ***método de aproximaciones sucesivas*** en la **conversión analógica/digital** que se utiliza en los **codificadores PCM**.

IV.4.2.2 CUANTIZACION POR FIGURA DE GANANCIA

Otra forma de aprovechar la memoria sería normalizando el nivel de energía (**ganancia**) de un vector a codificar y codificarlo separadamente con un ***Cuantizador Escalar***. La *energía normalizada* se conoce como "**figura**".

Esta técnica se llama ***Cuantización Vectorial de la Figura de la Ganancia*** que fué estudiada por **Sabin** y ***Gray*** que encontraron la forma óptima para diseñar conjuntamente la "**ganancia**" y la "**figura**" del **cuantizador**. Para una transmisión de un **bit/muestra** (8 **Kb/s**) la dimensión estandar es de K=8 y **SNR**=9.7, y con una dimensión de K=12 se logra 0.7**dB**.

La única variante es la normalización del nivel de energía (**Ganancia**), para así comparar sólo vectores normalizados.

IV.5 TECNICAS DE CODIFICACION DE LA SEÑAL DE VOZ

Existen varias técnicas de codificación de la señal de voz como:

1.- Codificación de Forma de Onda en el Dominio del Tiempo.

2.- Codificación de Forma de Onda en el Dominio de la Frecuencia.

3.- Métodos de Codificación Paramétrica e Híbridos.

IV.5.1 METODOS DE CODIFICACION DE FORMA DE ONDA EN EL DOMINIO DEL TIEMPO

Quizá la codificación de forma de onda más simple es la llamada **PCM** (*Modulación por Codificación de Pulsos*) donde la señal analógica es *uniformemente cuantizada* en forma de celdas rectangulares en tiempo y amplitud. Este principio es utilizado en métodos de conversión de señales analógicas a digitales.

Existen dos variaciones del **PCM** que son muy usadas para teléfonos comerciales (denotadas como ***Ley-mu y Ley-A PCM***) y se basan en una *cuantización no uniforme de la amplitud de la señal* de acuerdo a una ***escala logarítmica*** en lugar de una ***escala lineal.*** Dichos códigos utilizan características de **amplitud no estacionaria** en la señal de voz y logran una buena calidad a razón de 56 **Kb/s** y 64 **Kb/s**. como se ve en la **figura IV.6**, cuya referencia se hizo en la página 28.

Las propiedades de correlación de muestra a muestra en la señal de voz, debida a las características paso bajas del espectro de voz y de las características espectrales dinámicas, propias de los *formantes*, son utilizadas en métodos de codificación diferencial, los cuales se basan en registrar las *pequeñas variaciones* que hay desde un tiempo "*t*" hasta un tiempo "*t*+1".

En la codificación diferencial la eficiencia es obtenida por la cuantización de la diferencia entre muestra y muestra de la señal de voz más que de ésta misma, de tal forma, que el promedio de correlación de muestra a muestra de la voz es grande, *el nivel* **RMS** *de la señal diferencial es mucho más bajo que el nivel* **RMS** *de la señal original.* De manera que un intervalo más pequeño de cuantización permitira un nivel más pequeño de **ruido.** Utilizando en combinación con métodos de intervalo de **tamaño adaptivo**, obtendremos una mejor aproximación al método **ADM** (*Modulacion Delta Adaptiva*) y el **ADPCM** (**PCM *Adaptivo Diferencial***).

El **ADPCM** en particular, ofrece una buena calidad de voz para lineas telefónicas a razón de 32 **Kb/s.**

El concepto de codificación diferencial puede ser aplicado al caso de la *correlación dinámica en la señal de voz*, esto es, una ***estructura de formantes variante en el tiempo***. En este caso se usarán diferencias de *alto orden* o ***predicción adaptiva***. De esta forma el nivel **RMS** de la señal diferencial y del tamaño del intervalo pueden ser relativamente minimizados en comparación con la señal original. Si la predicción está basada de las 4 a las 10 muestras anteriores, generalmente esto se refiere a una ***predicción de tiempo corto***. Por otro lado, la ***predicción de tiempo largo*** se debe a las **correlaciones** hechas de **período de pitch** en **período de pitch**, que pueden ser utilizadas para reducir esta diferencia de una manera *dinámica*.

Los métodos de gran complejidad nos ofrecen una razón de 16 **Kb/s** aunque las variaciones de estas técnicas siguen siendo exploradas y analizadas.

IV.5.2 METODOS DE CODIFICACION DE FORMA DE ONDA EN EL DOMINIO DE LA FRECUENCIA

Una segunda categoría de técnicas de codificación de onda están basadas en subdividir la señal de voz dentro de un conjunto de frecuencias dominadas por las componentes, las cuales son codificadas independientemente . Existe una forma simple de hacerlo mediante la ***Codificación de Sub-Banda*** (**SBC**). que se basa en la división de la banda de voz en 4 o 5 **sub-bandas** como si fuera un *banco de filtros*. La salidas de este banco se reducen al muestrear la señal utilizando el **APCM**. Esta técnica puede adaptarse dinámicamente a las variaciones del espectro, es decir, la estructura de los *formantes* de la señal de voz, y nos da una
representación del **ruido** para así tener una mayor percepción de la señal de voz. Esta aproximación nos da una buena calidad de voz a razón de 16 **Kb/s**. Se puede mejorar la aproximación si aumentamos el número de sub-bandas y además variamos dinámicamente el número de **bits** por muestra.

IV.5.3 METODOS DE CODIFICACION PARAMETRICOS E HIBRIDOS

En contraste con los métodos de codificación de la señal de voz antes estudiados que manejan directamente la onda temporal de la voz, existen otros métodos para representar la voz, llamados métodos de parametrización. Los parámetros característicos de una señal de voz que la definen univocamente, pueden ser la altura tonal, la intensidad, los *formantes*, etc.

Las técnicas de proceso digital de señales han instrumentado el análisis y la síntesis paramétrica de la voz basándose en un modelo de producción de ésta. El modelo consiste en un filtro variable en el tiempo excitado por un tren de pulsos casi periódicos (sonidos sonoros) o un **ruido** blanco (sonidos sordos). La analogía física de este modelo es el aparato fonador. La señal de excitación la producen las cuerdas vocales, y el filtro equivale a todas las cavidades acústicas que la señal atravieza desde las cuerdas hasta el exterior.

El análisis de la voz para obtener éstos parámetros exige considerables esfuerzos de cálculo e introduce cierta degradación en la señal vocal, pero en cambio, conduce a una reducción drástica del caudal de información a transmitir (hasta 600 **bits/s**) los parámetros son casi estacionarios, es decir se consideran constantes durante un cierto tiempo y se actualizan, transcurrido ese tiempo. El tiempo que se considera es del orden de 20 **ms**. La casi estacionariedad de la señal se basa en que físicamente los órganos que componen el aparato fonador no pueden variar su posición instantaneamente.

Entre los ***métodos paramétricos de Análisis / Síntesis*** de la voz más utilizados están el ***Método Homomórfico***, que utiliza *parámetros relacionados con la fisiología del aparato fonador*; y el ***Método de Predicción Lineal***, que *utiliza parámetros propios de la señal*. A continuación se hará una exposición somera de ambos métodos.

IV.5.3.1 SINTESIS PARAMETRICA ANALOGICA DE LA VOZ

La descripción de los mecanismos de producción de voz, que se hizo en el **Capítulo III Sección Número 5**, sugiere la posibilidad de una simulación electrónica en la cual las fuentes vocales de sonoridad y fricación, son representadas por generadores de pulsos y **ruidos**, mientras que los *formantes* son representados por filtros resonantes de segundo orden organizados en paralelo o en serie.

Los parámetros de control de este modelo serán los que indiquen frecuencia y amplitud de los pulsos de sonoridad, amplitud de la excitación de **ruido**, frecuencia, ancho de banda y amplitud de los *formantes* vocales, así como de los *formantes* y *antiformantes* que aparecen en presencia de **ruido** o nasalización. La alteración de estos parámetros, hechos por el hombre durante el proceso de produccion de voz, obedece a acciones mecánicas y por lo tanto, de variación lenta. Se estima, de hecho, que el sistema fonador es casi estacionario dentro de intervalos de 10 **ms** a 30 **ms**, y este ritmo relativamente lento es el que debe seguir la actualización de los parámetros de dicho modelo.

El ancho de banda necesario para la transmisión de estos parámetros se reduce a unas decenas o centenas de **Hz.**, frente a los 3400 **Hz**. que exige la transmisión directa de la señal vocal. *La reducción del ancho de banda depende del número de parámetros y de los niveles de cuantización utilizados para codificarlos*. Es suficiente tomar en promedio 16 niveles (4 **bits**).

La ***parametrización*** de la señal de voz encuentra de esta forma una primera aplicación en la reducción del ancho de banda de la señal vocal. Asociado con la reducción del ancho de banda, se obtiene una mejora práctica del *flujo de información* "**C**" (**bits/s**) necesario y, por tanto, también de *la memoria ocupada por un tiempo* "**T**", de mensaje a emitir ("**C*T**" **bits**) en sistemas de respuesta de voz.

Así se pasa de 80,000 **bits/s** con una codificación **PCM** a 10 **KHz** y 8 **bits/muestra**, a menos de 600 **bits/s** en la parametrización por *formantes* o métodos digitales que en la actualidad se proponen.

Los sintetizadores analógicos actuales parten del esquema básico descrito anteriormente y son controlados por parámetros digitales. Es decir, constituyen un ***Sistema híbrido analógico/digital*** que recibe la información codificada digitalmente por *canales discretos* y elabora la señal de voz sobre *filtros activos analógicos controlados numéricamente*.

Existen varios modelos en el mercado pero uno de los más perfectos es el **OVE-IIId**, diseñado por **G. Fant** en el ***Instituto Real de Tecnología de Estocolmo***. En él se observan las fuentes de sonoridad (de frecuencia controlada por **F0**) y de **ruido blanco**, y tres ***canales de transmisión y filtrado***. En la **figura IV.7** se muestra es diagrama de bloques del sintetizador **OVE-IIId**.

El canal central corresponde al conducto bucal y está controlado por 5 *formantes* de segundo orden, siendo los tres primeros controlables en **frecuencia** y **ancho de banda**. El **ancho de banda** está relacionado con la **parte real** de los **polos (complejos conjugados)** y se suele aceptar (siguiendo determinaciones experimentales y razonamientos sobre las *pérdidas y rozamiento en las paredes de las cavidades fonadoras*) que dicha parte real o es aproximadamente constante para los primeros *formantes* o se pueden relacionar con la frecuencia.

Por ello se puede obtener una calidad de voz aceptable aún evitando el manejo de los parámetros extras "**B0**", "**B1**", "**B2**", así como "**F4**" y "**F5**" están fijos en 3.5 **Hz** y 4.0 **Hz** respectivamente. "**KH**" cumple con la doble función de mezclar las señales de voz y **ruido** de aspiración (*fricación en las zonas posteriores del aparato*) según niveles ajustados por los controles de amplitud "**AV**" y "**AH**" y de compensar mediante *preénfasis* en el espectro de la influencia de los *formantes* altos no tomados en cuenta.

El canal superior representa al **conducto nasal** controlado por los parámetros de amplitud "**AN**" y de frecuencia "**N1**" correspondiente a un formante nasal.

El canal inferior se utiliza para generar los **sonidos fricativos** y consta de dos *resonadores* de frecuencia "**K2**" y "**K1**" y dos *amplificadores* "**AK**" y "**AC**" que es además *sumador*, con lo que se consigue una *conformación del espectro* según las siguientes reglas:

1.- **AC** establece el nivel de referencia.

2.- **K1** y **K2** introducen **picos**.

3.- **AK** establece la amplitud relativa en bajas frecuencias, por encima, por debajo o al mismo nivel de **AC**.

4.- Debido a la forma en que se suman las 3 señales en el último amplificador se introduce un ***antiformante*** que puede estar por debajo o por encima de **K1** o coincidir con él.

En total se dispone de 15 **parámetros de control** *codificados logarítmicamente* con 6 **bits**, si bien es este conjunto redundante para la obtención de una calidad normal de voz, aunque se aprovecha la flexibilidad del instrumento para estudios fonológicos y establecimientos de reglas de síntesis.

IV.5.3.2 ANALISIS / SINTESIS HOMOMORFICA

Según lo dicho anteriormente, y basándose en el modelo de producción de voz expuesto, la señal de voz, matemáticamente hablando, es la convolución de una señal excitadora con la respuesta a impulso del filtro correspondiente.

El análisis homomórfico realiza la deconvolución de la señal de voz y obtiene por una parte los parámetros correspondientes a la señal de excitación y por otra los parámetros correspondientes al filtro, en este caso su respuesta a impulso. Esta deconvolución se hace obteniendo primero el ***cepstrum*** de la señal.

El ***cepstrum*** de una señal muestreada es a la vez otra señal muestreada cuya ***transformada discreta de Fourier*** es el *logaritmo del módulo de la transformada discreta de Fourier de la señal original.*

Es decir, dada una señal x(n), su ***cepstrum*** es $\hat{X}(n)$ donde $\hat{X}(z)=\log x(z)$. El ***cepstrum*** transforma un ***producto de convolución*** en una ***suma***, ya que si por ejemplo:

Producto de Convolución de la Señal:

$$x(n) = x_1(n) * x_2(n)$$

Entonces:

Suma de Cepstrums:

$$\hat{X}(z) = \hat{X}_1(z) + \hat{X}_2(z)$$

Dos propiedades del cepstrum hacen que esta técnica se aplique con éxito en la ***deconvolución*** en la señal de voz.

1.- El ***cepstrum*** de una sucesión cuya ***Transformada Z*** (equivalente a la ***Transformada de Fourier*** para $z=e^{jw}$) sea una función racional de **z**, tiende a estar concentrado en **n** = 0.

2.- El ***cepstrum*** de un tren de pulsos es también un tren de pulsos.

Debido a que la señal vocal es una ***convolución*** de la ***respuesta a impulso*** del filtro y de la ***excitación***, estas dos cumplen las propiedades anteriores; el ***cepstrum*** de la señal de voz es la suma de la sucesión que tiende a estar concentrada en n=0 con otra que es periódica.

DIAGRAMA DE BLOQUES DEL SINTETIZADOR OVE-III d

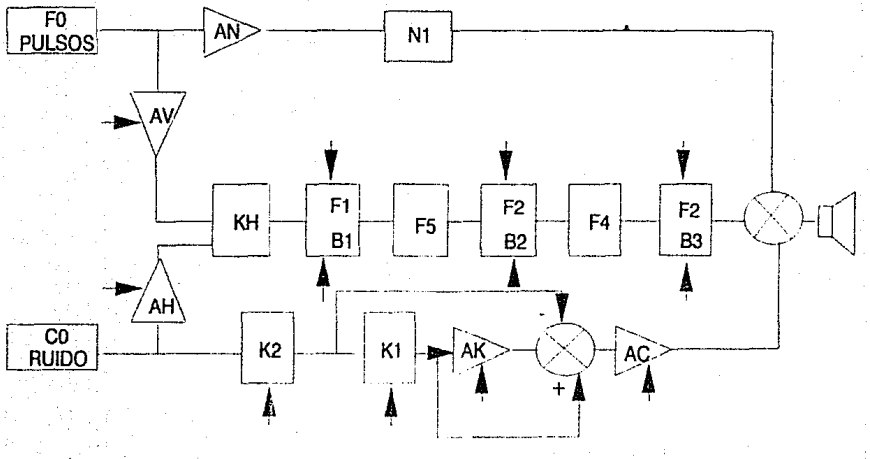


FIGURA IV.7

DIAGRAMA DE BLOQUES DEL SINTETIZADOR OVE-III d

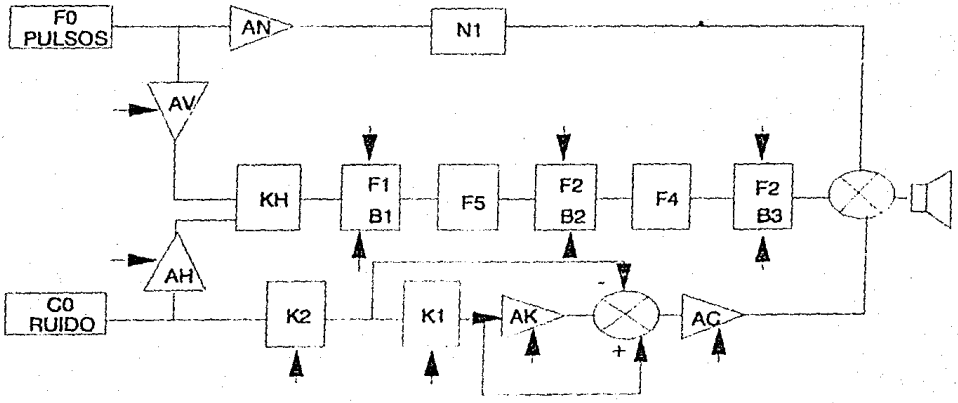


FIGURA IV.7

La primera se recupera multiplicando la señal por "1" para valores pequeños de "n" y por "0" el resto. Con la obtención de estos **parámetros de excitación** y los **parámetros propios del filtro** se muestra cómo se puede recuperar la voz. En la **figura IV.8** se presenta el diagrama de bloques del ***Analizador Homomórfico***.

IV.5.3.3 METODO DE ANALISIS / SINTESIS DE PREDICCION LINEAL

La señal de voz se genera a través de un **tren de pulsos** o un **generador de ruido blanco** que alimentan a un filtro, digitalizando todos los **polos**. Para poder reconstruir una señal vocal con este modelo se necesitan varios parámetros que son los llamados ***parámetros*** **LPC** (*Linear Predictive Coding*).

Estos parámetros son los siguientes:

1.- Razón sonido sonoro/sordo.

2.- Periodo del tren de pulsos para sonidos sonoros.

3.- Ganancia del amplificador.

4.- Coeficientes del filtro digital.

Los tres primeros parámetros corresponden a la señal de excitación del filtro de la que hablamos anteriormente.

La clave del método **LPC** está en la elección de un filtro todo polos que conduce a un tratamiento lineal de la señal. Esta elección se justifica en gran medida debido a que en los sonidos sonoros no nasales, el modelo del aparato fonador no tiene ceros. En los sonidos nasales y no sonoros, el modelo tiene polos y ceros pero éstos últimos están dentro del círculo unitario y pueden aproximarse por los múltiples polos. En la **figura IV.9** se muestra el diagrama de bloques del ***Método de Análisis / Síntesis de Predicción Lineal***.

La extracción de los parámetros **LPC** de una señal se hace por medio de un análisis por tramos aproximadamente de **20 ms** de señal.

La **parametrización LPC** se aplica a la *reducción de banda en transmisión, almacenamiento eficiente de información, cálculo de las trayectorias de* ***formantes,*** *determinación de la altura tonal, reconocimiento de voz, etc.*

La parametrización de la señal de voz encuentra de esta forma una primera aplicación en la reducción del ancho de banda de la señal de voz.

Los ***sintetizadores analógicos*** actuales parten del esquema básico descrito anteriormente y son controlados por parámetros digitales.

DIAGRAMA DE BLOQUES DE UN ANALIZADOR HOMOMORFICO

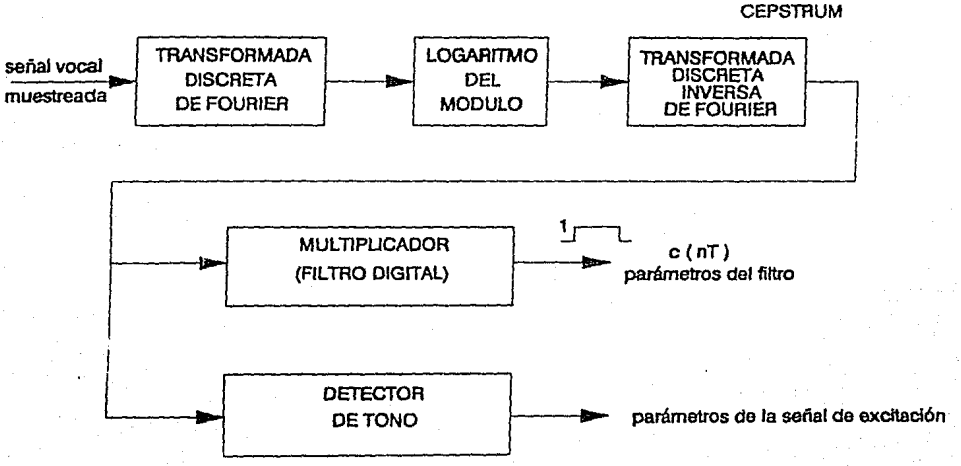


FIGURA IV.8

MODELO DE SINTESIS POR PREDICCION LINEAL

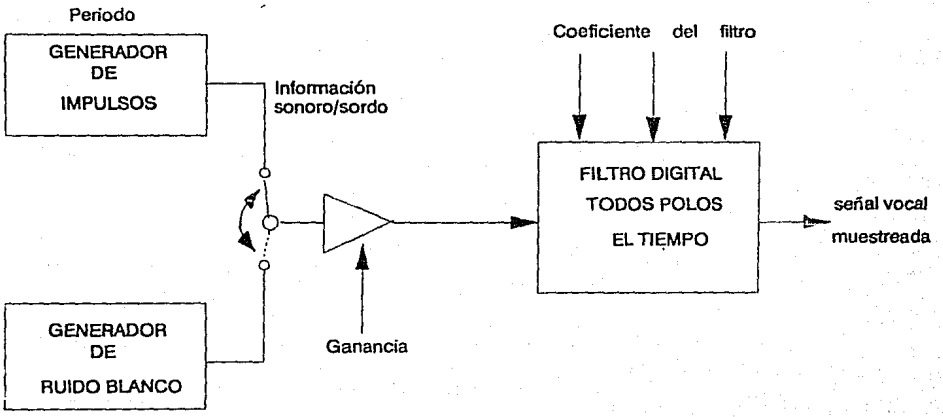


FIGURA IV.9

Como modelo más sencillo del aparato fonador tenemos:

$$L = \frac{2n + 1}{4} \text{ lambda}$$

L: para un varón adulto es de 17 **cm**.
n: número de polos, orden del filtro inverso.
lambda: longitud de onda.

Formantes : 800 **Hz**, 1500 **Hz**, 2500 **Hz**, etc.

1o. formante : entre 200 **Hz** a 1000 **Hz**.

2o. formante : entre 500 **Hz** a 2500 **Hz**.

3o. formante : entre 1500 **Hz** a 3500 **Hz**.

La transmisión desde la glotis a la apertura labial puede ser modelada por una serie de cavidades resonantes obteniendose una función de transferencia producto de terminos resonantes de segundo orden del tipo:

$$\frac{1}{(s-s_o)^n (s-s_o)}$$

Es decir, caracterizada por una serie de polos complejos conjugados correspondientes a los métodos normales de transmisión.

IV.5.3.3.1 METODO DE CODIFICACION POR PREDICCION LINEAL

Actualmente es uno de los métodos más poderosos y más usados en la determinación de ***Parámetros Básicos de la Señal de Voz***, como el **Pitch**, ***Formantes***, **Espectro de Energía, Funciones del Area del Tracto Vocal y Representación de la Señal de Voz** con un nivel relativamente reducido de bits.

Este método se basa en el hecho de que una muestra de la señal de voz puede ser expresada como una *combinación lineal de muestras anteriores, minimizando la suma de los cuadrados de las diferencias, en un intervalo finito entre las muestras actuales y las predichas linealmente, se puede determinar un conjunto único de coeficientes del Predictor, donde los coeficientes de éste son coeficientes* "**ponderados**" *usados en la combinación lineal.*

La ***Codificación por Predicción Lineal*** usa un modelo basado en un sistema lineal variable en el tiempo, excitado por pulsos casi periódicos, o ruido aleatorio para representar la señal de voz, mediante este método, lo que se logra es obtener los *parámetros que caracterizan a dicho sistema*.

Las técnicas **LPC** han sido usadas en diferentes áreas como son el ***Control Adaptivo y Teoría de la Información***, bajo los nombres de ***Estimación de Sistemas***.

En relación al procesamiento de la señal de voz el término **Predicción Lineal** es usado para denominar una gran variedad de métodos de formulación del problema, que son en esencia equivalentes, como son :

1. El Método de Covarianza.

2. La Formulación de la Autocorrelación.

3. El Método de Lattice.

4. La Formulación del Filtro Inverso.

5. La Formulación de la Estimación Espectral.

6. La formulación de Probabilidad Máxima.

7. La Formulación del Producto Interno.

Los tres primeros métodos son básicos, ya que los demás son equivalentes a alguno de ellos.

El modelo básico sobre el que se aplica el Método de Codificación por Predicción Lineal es el de la **figura IV.10**.

MODELO BASICO DEL METODO DE CODIFICACION LPC

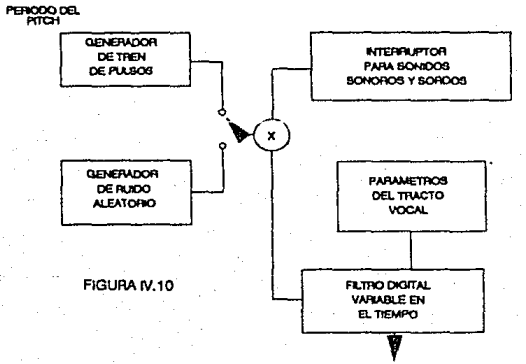


FIGURA IV.10

Asi la señal de voz puede ser representada mediante un *filtro digital variable en el tiempo* cuya función de estado estable es de la forma :

$$H(Z) = G/A(Z) = G/(1 - \sum_{k=1}^{p} akz^{-k})$$

De tal forma que los parámetros para este modelo son :

a) Clasificación de sonidos sonoros/sordos.

b) Periodo del Pitch.

c) Ganancia G.

d) Coeficientes $\{ a_k \}$

Donde estos parámetros varían lentamente en el tiempo.

Con los parámetros antes mencionados se puede reducir en gran medida la información necesaria para representar la señal de voz logrando así una compactación de la información.

IV.6 SINTESIS POR FORMANTES

La señal de voz se ha podido representar de una manera *discreta*, teniendo un ***diccionario de fonemas del lenguaje*** como un *conjunto de símbolos finitos*.

Cada uno tiene un sonido particular y forma diferente, dependiendo de la forma del tracto vocal. Los fonemas y, posteriormente los mensajes, van a poder ser formados a partir de la concatenación de los ***formantes***.

La ***Síntesis por Formantes*** depende principalmente del análisis del proceso del habla para que de ahí se pueda generar un *vocabulario*. Este *vocabulario* se almacenaría por medio de ***formantes o armónicas***. Los ***formantes*** son las *resonancias naturales del tracto vocal, con diferentes frecuencias durante el habla*. Por ejemplo, para **sonidos nasales** el rango de frecuencias varía de 0 a 3 **KHz**. El dato que representa la longitud del ***formante*** es accesado por *Software* y posteriormente *concatenado* para completar una palabra.

Una vez obtenidos los ***formantes*** característicos, se utilizarán para *sintetizar una forma de onda aproximada a la señal original de voz*.

Numerosos **sistemas analógico - digitales** se han diseñado para la ***síntesis de formantes***.

La **eficiencia del almacenamiento** de la representación de voz por ***formantes***, depende de la *presición* con la cual los *parámetros básicos* puedan ser representados o especificados.

La síntesis de voz de alta calidad puede ser obtenida si el **periodo del pitch** es especificado alrededor de 0.1 **ms**, la **ganancia** especificada en una zona de 100 **dB** y la **frecuencia de los *formantes*** de cerca de 1 **KHz**. Así los parámetros son estimados para poder sintetizar 100 veces por segundo. La **razón de información** requerida puede ser muestreada a 4600 **bits/s**.

Existen 2 consideraciones que son importantes:

La **primera** es el **ancho de banda** requerido para preservar las *variaciones escenciales de los parámetros*, obtenido al diseñarse el arreglo de filtros **paso altas** y **paso bajas** que en su conjunto dan como resultado un filtro **paso banda**, con un rango de frecuencias entre 200 **Hz** y 3500 **Hz**.

La **segunda**, es el **grado de cuantización** que se usa en la ***Síntesis de Parámetros***, dependiendo del número de **patrones** contenidos en el ***Code-book*** y el número de bits que se desean cuantizar.

CAPITULO V

V. CONSIDERACIONES PREVIAS AL DESARROLLO DEL PROYECTO

V.1 BREVE DESCRIPCION DEL FUNCIONAMIENTO DEL PROYECTO

Este proyecto genera voz a través de la previa grabación mediante sistemas digitales.

La señal de voz se codifica y decodifica para obtener su reproducción. Si se grabara toda la señal de voz, esto implicaría ocupar un gran espacio de memoria. Para aprovechar mejor ese espacio, se eligió una técnica de compactación de la señal, eliminando partes poco representativas de ésta.

V.2 CONSIDERACIONES GENERALES

El proyecto se ha dividido fundamentalmente en dos etapas:

1.- ***Adquisición y Procesamiento de la señal de voz por medio de Hardware.***

2.- ***Captura y Procesamiento de datos mediante Software.***

La primera etapa, de ***Adquisición y Procesamiento de la señal de voz por Hardware***, comprende:

1.1.- Etapa de amplificación.

1.2.- Etapa de filtrado de la señal.

1.3.- Etapa de conversión de señal analógica a señal digital.

1.4.- Interfase de acoplamiento con el bus de datos y direcciones de la computadora.

1.5.- Sincronía en la comunicación.

1.6.- Convertidor de señales digitales a señales analógicas.

1.7.- Filtro de salida y recuperación.

1.8.- Amplificador de audio.

V.2.1 DESCRIPCION GENERAL DE LA ETAPA DE HARDWARE

Consiste en un ***micrófono dinámico*** que es conectado como fuente de entrada a un amplificador en cascada con un filtro **Chevyshev** paso altas de quinto orden. Dicho amplificador tiene una frecuencia de corte de 200 **Hz** para limitar las componentes de baja frecuencia de 60 **Hz** y otras componentes no propias de la voz, además de otro **filtro paso bajas Chevyshev** de cuarto orden conectado en cascada con una frecuencia de corte de 3 **KHz**. Estos dos filtros nos dan una "ventana" como se muestra en la siguiente figura:

VENTANA DEL FILTRO PASO-BANDA TIPO CHEVYSHEV

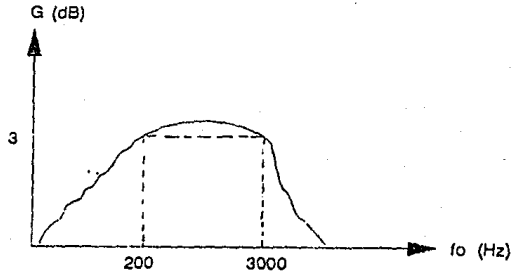


FIGURA V.1

Con esto se reducen las posibilidades de captar señales de otras fuentes diferentes a la de la voz y se reduce la cantidad de ruido proveniente de los circuitos de la computadora y de algunas otras fuentes de alta frecuencia.

La etapa de conversión **A/D** y **D/A** está implementada por los circuitos integrados **ADC 0805** y **DAC 08**, respectivamente. Aunque en un principio el **DAC** se implementó mediante un sistema **(R/2R)** con componentes discretos.

La interfase de acoplamiento con el **bus de datos** está implementada por un decodificador y algunas compuertas lógicas adicionales. El filtro de salida y recuperación consiste en un **filtro paso altas** en cascada con uno **paso bajas** de tipo **Chevyshev**, con idénticas características a los filtros de la primera etapa. En esta primera etapa es menester garantizar la fidelidad de la reproducción y la eficiencia en la comunicación *tarjeta / computadora*. La *Etapa de Hardware* se describe más ampliamente en el capítulo **VI**.

V.2.2 DESCRIPCION GENERAL DE LA ETAPA DE SOFTWARE

La segunda etapa, ***Captura y Procesamiento de Datos mediante Software***, comprende:

2.1.- Software de Control de Interfase de Entrada / Salida.

2.2.- Analizador de Estructuras Fonéticas.

2.3.- Analizador de la Señal Capturada y Despliegue Gráfico de la misma.

2.4.- Compactación de la Señal y Almacenamiento.

2.5.- Reproducción de la Señal Compactada.

V.2.2.1 SOFTWARE DE CONTROL DE INTERFASE ENTRADA / SALIDA.

Este es un programa que inicialmente se realizó en ensamblador **8086 / 8088** utilizando instrucciones de carga / descarga al stack, carga a memoria, llamadas a direcciones de puertos y llamadas a interrupciones.

Actualmente, este programa está implementado en **TurboPascal**, un lenguaje con instrucciones de llamada a puertos *físicos*, entre otras funciones. Por lo que se tomó el criterio de que los datos capturados iban a ser traídos a memoria mediante este tipo de instrucciones.

Este programa además tiene un control sobre la ***frecuencia de muestreo***, de aproximadamente 7 **KHz**.

V.2.2.2 ANALIZADOR DE ESTRUCTURAS FONETICAS.

Este programa analiza un texto almacenado en un archivo de la computadora línea a línea. Y después desglosa la línea por palabras.

Posteriormente cada palabra es analizada por sílabas.

Inicialmente no se tiene almacenada ninguna sílaba, correspondiente a un fonema, por lo tanto, *manualmente* se procederá a dar la *separación fonética / silábica* de las palabras escritas en esa línea. Después se llama a la Rutina de Captura para almacenar los fonemas digitalizados que desconoce en ese momento la computadora.

Ya una vez realizado este procedimiento, la computadora requiere de ***reglas gramaticales*** (*letras esperadas, después de definir una separación silábica*) para hacer la separación silábica correspondiente, identificarla y verificar si existe o no en el diccionario de fonemas, almacenado en disco. De ahí que la computadora poco a poco, irá aprendiendo y aumentando su vocabulario hasta que pueda reproducir, en voz, fielmente lo que se ha escrito.

V.2.2.3 ANALIZADOR DE LA SEÑAL CAPTURADA Y DESPLIEGUE GRAFICO DE LA MISMA

Esta rutina llama a las siguientes rutinas: ***Captura, Compactación*** y ***Reproducción***; que se explicarán más adelante. Una vez capturados los datos, se almacenan en un **vector** que será desplegado por la ***Rutina de Graficación de la Señal***, la cual fija la *escala horizontal y vertical*, realiza el desplazamiento de la señal en el *tiempo* y además traza una rejilla de escala en *tiempo y amplitud*.

Se pueden hacer *cortes* para analizar la señal deseada, *regenerarla* y *desplegar la señal originalmente capturada, comparada con la señal descompactada y reproducida*.

V.2.2.4 COMPACTACION DE LA SEÑAL Y ALMACENAMIENTO

Es el programa encargado de *compactar la señal en memoria*. Existen, actualmente, diversas formas para representar la señal de voz. Una de ellas es, la forma *paramétrica* por el ***Método Homomórfico*** mediante *la simulación del tracto vocal*, manejando *tipos de excitación y el Area Glotal*; y el ***Método de Codificación de Análisis / Síntesis de Predicción Lineal***, del cual se calculan *parámetros* como el **Pitch**, *Error Cuadrático Medio de Aproximación, Parámetros de Excitación, Intensidad y Timbre*.

Otra opción fue crear un método de compactación propio y que se *ajustara exclusivamente a las necesidades que exigía el proyecto*. Por lo que se desarrolló un ***Método de Compactación mediante Tablas de Máxima Proximidad***. A continuación se explicará de una manera más detallada el método de compactación que se utilizó en el desarrollo de este proyecto.

V.2.2.4.1 METODO DE COMPACTACION MEDIANTE EL USO DE TABLAS DE MAXIMA PROXIMIDAD

Primeramente cabe mencionar que este método fue desarrollado tomando en cuenta las siguientes características:

1.- Este método es de ***Codificación de Forma de Onda***.

2.- Requiere **PCM** de 8 **bits**.

3.- Utiliza el concepto de *diferencias*, más no aplica una ***Modulación Delta***, en el estricto sentido del método.

4.- La filosofía no es *predecir sino aproximarse*, por lo que no se maneja ***error de predicción***, como en los métodos de ***Parametrización y Síntesis***, sino ***error de proximidad***.

5.- Crea a partir de un **vector de diferencias** (*tomado de las muestras capturadas*), otro vector que almacena los datos *utilizando menos espacio*, de acuerdo a una ***Tabla Intermedia de Valores Próximos*** a la diferencia original con lo que se logra disminuir la **entropía** de la señal, y pueden ser representados y almacenados en una *menor cantidad de* **bits** a través de una concatenación.

6.- No se utilizan **vectores de cuantización**, sino que se construye una tabla que tenga los *valores más representativos de las diferencias de la señal de voz*. Esta tabla es un *arreglo de ocho elementos*, considerando el elemento **0** como un *cambio de pendiente*, y del elemento **1** al **7** *submúltiplos de la* ***máxima diferencia*** *entre muestras*, si la Ley es **Lineal de Elementos Fijos (*)**.

7.- Existe una *etapa de transformación* llamada ***Concatenación de Indices de la Tabla de Máxima Proximidad***. Cada diferencia es representada en 3 **bits**, por lo que la razón **bit/muestra** se reduce de 8 **bits** a 3 **bits**, más el número de *elementos ceros*, de tal forma de que entre más suave sea la forma de la señal de voz, el número de cambios de pendiente disminuirá, obteniéndose así una compactación mayor.

Una vez capturados los datos desde el puerto de entrada, se representa la señal a través de muestras y se calculan las diferencias entre muestra y muestra. Después se determinará la **máxima diferencia** (tomando en cuenta el valor ***absoluto de las diferencias***) como se puede ver en la **figura V.2**. Para el método lineal se obtendrá un elemento base de la siguiente manera:

$$BASE = \frac{\text{Máxima Diferencia}}{7} = I \quad ....(1)$$

V.2.2.4.1.1 GENERACION DE TABLA POR EL METODO LINEAL DE PROPORCIONES FIJAS.

Se generan múltiplos con la siguiente definición:

$$Vprx(K) = KI \qquad 1 <= K <= 7 \quad ....(2)$$

Donde **7I = Máxima diferencia** y el elemento **K= 0** de la tabla representa el cambio de pendiente **+ / -**.

(*) Esta ley toma en cuenta proporciones fijas, es decir, la tabla de valores generados al codificar la señal va a comprender valores que están incluidos dentro de la ***Tabla de Valores Próximos***, que van a ser valores fijos dentro del proceso de muestreo y compactación.

ELEMENTO K No.	CONTENIDO Vprx(K)
0	+ / -
1	I
2	2I
3	3I
4	4I
5	5I
6	6I
7	7I

Ya creada la tabla, se procede a comparar los valores "**Vprx**" con los datos de entrada originales haciendo referencia al *valor más próximo en tablas*, el cual puede ser representado en 3 **bits** y no en 8 **bits**, estos valores se concatenan de tal forma que ocupen bytes completos.

V.2.2.4.1.2 ALGORITMO DE DETECCION DE MAXIMA PROXIMIDAD

$$Ax = M(i) - Ac \quad ....(3)$$

$$Dif = \mathbf{abs}(Ax) \quad ....(4)$$

$$Ep = \mathbf{min}\ (\mathbf{abs}(Vprx\ (K) - Dif)) \quad ....(5)$$

Donde :

M = Muestras.
I = Num. de muestra.
K = Indice de la tabla de proporciones fijas.
Ac = Valor acumulado.
Ax = Diferencia.
Vprx = Valor en tabla de máxima proximidad correspondiente al índice "**K**".
Ep = Error Parcial.

En la **figura V.3** se describe en forma gráfica el Algoritmo de Detección de Máxima Proximidad.

Con la ecuacion (5) se detecta el índice "**K**" del valor "**Vprx**" más próximo, cuando "**Ep**" tiende a cero. Entonces ese valor de "**K**" se va a expresar como "**Kx**".

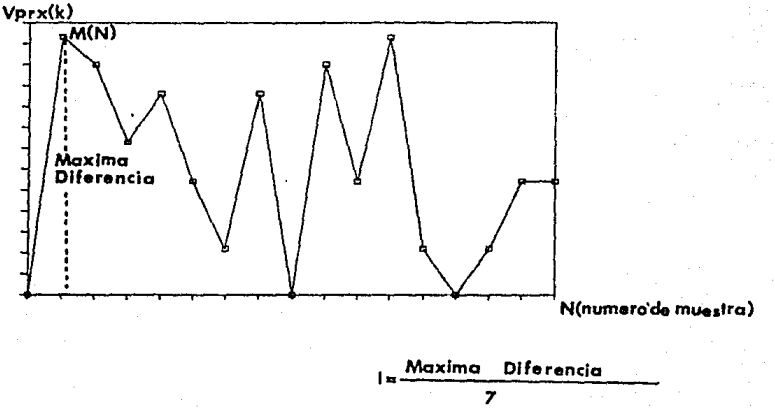
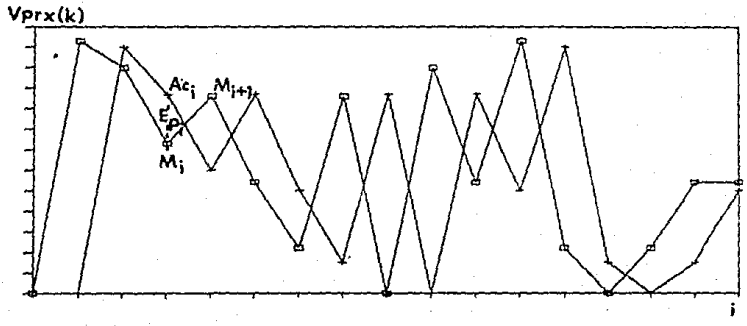

Vprx(k)
M(N)
Maxima
Diferencia
N(numero de muestra)
I = Maxima Diferencia / 7

FIGURA V.2



$$Ac_{i+1} = Ac_i \pm Vprx(k)$$

FIGURA V.3

En las **figuras V.4** y **V.5** se explicarán, a través de gráficas y tablas, el *Proceso de Compactación* y el de *Descompactación*.

Condición:

$$0 <= Ac <= 255 \quad \textbf{Si } Ac < 0 \textbf{ entonces } Ac = 0$$

$$\textbf{Si } Ac > 255 \textbf{ entonces } Ac = 255$$

signo = valor del signo de la pendiente (+ ó -).

$$Ac = Ac_{ant} + (\textbf{signo} * Vprx(K_x)) \quad ....(6)$$

Donde:
Ac $_{ant}$: es el valor anteriormente acumulado.

Partiendo de la ecuación (5) se deduce que :

$$Ep = \textbf{min}(\textbf{abs}[Vprx(K) - \textbf{abs}[M(i) - Ac_{ant} + \text{signo} * Vprx(K_x)]]) \quad .....(7)$$

V.2.2.4.1.3 ALGORITMO DE COMPACTACION.

Vc = vector de valores compactados con índices igual a **(j+b)**

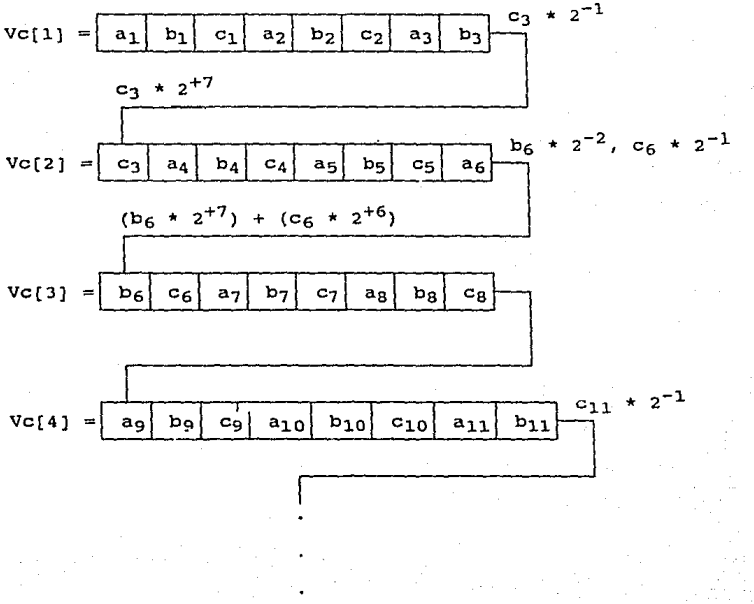
Vi = Vector de índices x=3r de la tabla de M. P.
Donde :
m = 0,1,2

Para:	**Con:**
k=3,	j=3, r=2
ó	
k=4,	j=2, r=1
ó	
k=5,	j=1, r=0

Donde **m:** es el número de secuencia de muestras a concatenar, sólo se pueden concatenar las muestras en secuencias de **tres en tres.**

$$Vc[j+b] \mid , \mid = \sum_{\substack{j=1\ b=0,4,8,12,.. \\ m=0\ k=5\ r=0}}^{3\ \ b+3} \mid , \mid (Vi[3r] \bmod 2)*2 + (Vi[x+1]*2 \mid)_{x=3r}^{3r+3 \le 8} \quad ...(8)$$

Σ : Sumatoria con límites inferior y superior.



Donde a_n, b_n y c_n representan los bits de codificación de Indices de la Tabla, y los vectores **Vc[m]** son palabras de 1 **Byte** cada una.

FIGURA V.4

Partiendo de los Vectores de Compactación **Vc[m]** que se representan en la **figura V.4**, sus Vectores de Descompactación, **Vcx[i]** son los siguientes:

Vcx[1] = | a_1 | b_1 | c_1 | representan el índice de la Tabla de Máxima Proximidad, que va de (000 a 111) en Binario.

Vcx[2] = | a_2 | b_2 | c_2 |

Vcx[3] = | a_3 | b_3 | c_3 |

Donde **Vcx[i]** es un elemento del Vector de Descompactación. Toma el Valor en Tabla cuyo índice se forma con los valores a_n, b_n y c_n.

FIGURA V.5

| : Condición "dado que" con límites inferior y superior.

V.2.2.4.1.4 ALGORITMO DE DESCOMPACTACION

Se vuelven a obtener los vectores de índices a partir de los vectores de compactación.

$$Vcx\ [j+b] = Vc\ [j+b]\ \textbf{donde:}\ \ j=1\ \textbf{cuando}\ k=5\ \ldots.(9)$$

$$\textbf{y}\ \ b= 0,4,8,12,..,etc$$

Vc : es un valor ya compactado localizado en el vector de compactación.

Vcx: es el valor del elemento sujeto a corrimientos para obtener la descompactacion de muestras a través de un valor compactado del vector de compactación (k+3).

$$Vcx\ [j+b] = Vc\ [j+b] - \textbf{\textit{acarreo}}\ [j]*2\ \ldots.(10)$$

Para:
j=2, k=4
ó
j=3, k=3

donde "***acarreo***" : es el **bit** más significativo de "**carry**" que se resta al siguiente elemento a descompactar.

$$\textbf{\textit{acarreo}}[j] = Vc[(j+1)+b] * 2^{-(k+2)}$$

Para:
j=1, k=5
ó
j=2, k=4

$$Vdx\ [3j-2] = Vd\ [3j-2]*2^{k} - Vd\ [3j-1]*2^{(k-3)}\ \ldots(11)$$

Para:
j=1, k=5
ó
j=2, k=4

Vdx : Es el valor de la tercera muestra descompactada a partir de la descompactación de las dos muestras anteriores.

Con estas fórmulas se obtienen todos los índices:

Caso 1 : Obtención de la primera muestra descompactada para una secuencia de compactación de tres muestras compactadas en un valor.

$$\text{I.-} \quad Vd\,[3i\text{-}2] = Vcx\,[(3i\text{-}2) + b] * 2^{-k}$$

Para:
i=1, k=5
ó
i=2, k=4
ó
i=3, k=3

Caso 2 : Obtención de la segunda muestra descompactada para una secuencia de compactación de tres muestras compactadas en un valor.

$$\text{II.-} \quad Vd\,[3i\text{-}1] = Vcx\,[i+b] * 2^{-(k-3)} - Vd\,[3i\text{-}2] * 2^{(k-(i+j))}$$

Para:
j=1, i=1, k=5
ó
j=-1, i=2, k=4
ó
j=-3, i=3, k=3

Caso 3 : Obtención de la tercera muestra descompactada para una secuencia de compactación de tres muestras compactadas en un valor.

$$\text{III.-} \quad Vd\,[3i] = Vcx\,[i+b] - Vdx\,[3i\text{-}2] + \textit{acarreo}\,[i]$$

Para:
i=1, k=5
i=2, k=4

V.2.2.5 REPRODUCCION DE LA SEÑAL DESCOMPACTADA

Esta rutina se encarga de reproducir la señal a partir de la descompactación de los datos almacenados. Esta descompactación consiste en tomar la concatenación y calcular los componentes que la formaron para así obtener los valores próximos a la diferencia original. Una vez hecho esto, se toma el primer valor próximo y se suma al segundo, tomando en cuenta el signo de la

Para desarrollar este proyecto se optó en usar una computadora tipo "**PC**", utilizando para la programación, el lenguaje **TurboPascal**, el cual ha sido de gran utilidad en el manejo de puertos y direcciones, tanto físicas como lógicas, además de tener la facilidad de realizar gráficas de alta resolución.

Una vez instalado el sistema **FONETICS** en la **PC**, y alimentada la memoria con fonemas pregrabados, el sistema llamará a las ***Rutina de Concatenación de Sílabas*** que después serán reproducidas como ***grupos fonéticos***. Además, a través de la ***Rutina de Graficación***, se desplegará la *señal capturada contra la señal descompactada*, así se podrá hacer un análisis de calidad de reproducción de señal de voz, aplicable a diagnóstico y a investigación. Estas rutinas van a ser *transparentes* al usuario final.

CAPITULO VI

VI. ETAPA DE HARDWARE

VI.1 CONSIDERACIONES PREVIAS AL DISEÑO

La presente descripción corresponde a las necesidades básicas del *hardware* que requeriere el proyecto para su funcionamiento.

La señal muestreada es una señal de voz, la cual se concentra para fines prácticos en un espectro de frecuencias que no supera los 3 **KHz**.

Se debe suprimir al máximo toda señal ajena a la señal de voz.

La señal una vez captada, amplificada y filtrada, se convierte a una señal *digitalizada* para que pueda ser almacenada y posteriormente procesada por la computadora.

Se requiere de una lógica adicional para que la computadora lea los datos y decodifique correctamente las direcciones de los sistemas descritos anteriormente.

Es necesario que después de capturada y procesada por la computadora, la señal de voz pueda ser reproducida. Para poder lograr la reproducción de voz, se requiere de una lógica que permita el direccionamiento de los circuitos adecuados en la tarjeta.

Una vez seleccionados los circuitos correctos en la tarjeta, la *señal digital* enviada por la computadora, tendrá que ser convertida a una *señal analógica* para poder ser reproducida en forma de voz. Para reproducir en forma adecuada la señal, tiene que ser filtrada para eliminar al máximo el **ruido** que esté presente.

Como último paso, la señal tendrá que amplificarse para que tenga un nivel adecuado y pueda ser reproducida por medio de una bocina.

El diagrama de la **figura VI.1** contiene la descripción del funcionamiento de la circuitería básica usada en el procesamiento de la señal de voz.

DIAGRAMA DE BLOQUES GENERAL DE LA ETAPA DE HARDWARE

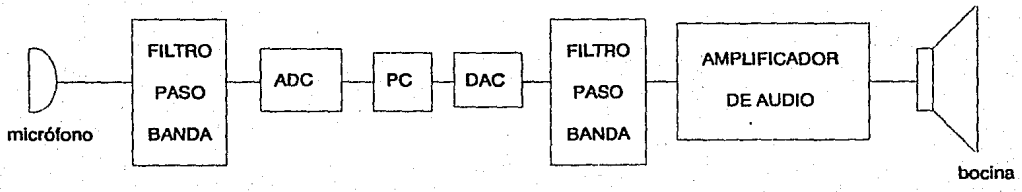


FIGURA VI.1

A continuación se detalla el cálculo y la justificacion del diseño del *hardware*, para esto se ha dividido en dos grandes secciones que son:

1.- Sección de Captura y Adquisición de la Señal de Voz.

2.- Sección de Salida y Reproducción de Voz.

A su vez, las dos secciones anteriores se han subdividido en dos etapas básicas, las cuales son:

1.1.- Etapa de procesamiento de señales analógicas.

2.1.- Etapa de procesamiento de señales digitales.

VI.2 CALCULOS REALIZADOS PARA EL DISEÑO DE LA ETAPA ANALOGICA DE LA SECCION DE CAPTURA Y ADQUISICION DE LA SEÑAL DE VOZ

Para la etapa de *pre-amplificación de la señal de audio* se utiliza un circuito amplificador con acoplamiento de **CA** para reducir al mínimo el nivel de **CD**.

CONSIDERACIONES:

1.- Se utiliza un ***micrófono dinámico***.

2.- El **voltaje máximo** obtenido en el micrófono fue de 3.53 **mVpp**.

3.- Es necesario evitar la saturación de la señal de salida de la etapa de pre-amplificación.

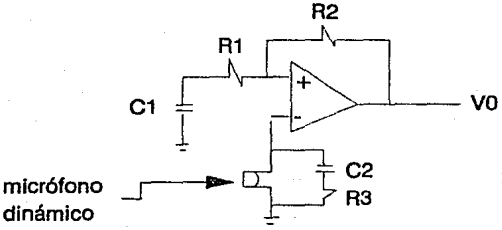
Por lo anterior se concluye que el circuito de la **figura VI.2** es el más apropiado para obtener una señal de una calidad aceptable y aumentar la ***Razón Señal / Ruido*** **SNR**, de acuerdo a los fines que se persiguen en este proyecto.

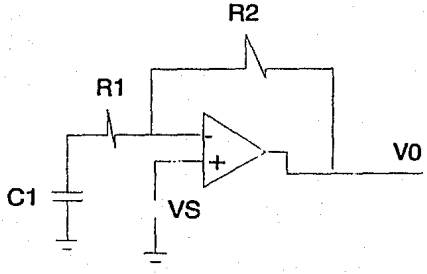
En donde:

$$Vs = \frac{Vo}{Z1 + R2} * Z1 \quad ....(1)$$

Despejando :

ETAPA ANALOGICA DE LA SECCION DE CAPTURA





MODELO DE IMPEDANCIAS

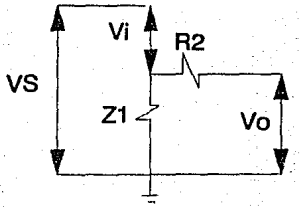


FIGURA VI.2

$$\frac{Vo}{Vs} = \frac{Z1 + R2}{Z1} \quad ....(2)$$

Sustituyendo **Z1** :

$$Vo = \frac{R1 + \frac{1}{SC1} + R2}{R1 + \frac{1}{SC1}} Vs = 1 + \frac{R2}{R1 + \frac{1}{SC1}} Vs$$

$$= 1 + \frac{R2}{R1}\left(\frac{1}{1 + \frac{1}{R1SC1}}\right)Vs$$

$$\frac{Vo}{Vs} = 1 + \frac{R2}{R1} * \frac{S}{S + \frac{1}{R1C1}} \quad .... (3)$$

De la ecuación (3):

$$W0 = \frac{1}{R1C1} \quad \{\textit{frecuencia del sistema}\} \quad .... (4)$$

Entonces:

$$W = 2\,fL \quad \{\textit{frecuencia de resonancia}\} \quad .... (5)$$

Despejando **fL** :

$$fL = \frac{1}{2\,R1C1} \quad .... (6)$$

Donde "**fL**" es la frecuencia de corte más baja con una caída de 3 dB.

Sustituyendo los valores de "**R1**" y "**C1**" :

fL = 21.9 **Hz.**

De acuerdo a lo obtenido se tiene que:

La ***ganancia*** está dada por :

$$G = 1 + \frac{R2}{R1} = 1 + \frac{10^{3}}{330} = 31.3 \ldots . (7)$$

si $V_{s_{max}} = 3.53$ **mVpp**

$$V_o = (31.3)\,(3.53 \times 10^{-3}) = 110.5 \text{ mV}$$

"**Vo**" tiene un voltaje bajo pero es adecuado para manipular la señal sin saturar la salida y permite usar etapas posteriores de amplificación.

Después de esta etapa, se conecta un filtro **paso banda** para limpiar la señal, el cual se describe en la siguiente sección.

VI.3 CALCULO DE FILTROS TIPO CHEBYSHEV

Para la etapa de filtrado del circuito se utilizan ***filtros Chebyshev*** por presentar las siguientes características:

1.- Corte más abrupto de las frecuencias que otro tipo de filtros.

2.- Fácil construcción.

3.- Amplia flexibilidad para la variación de los parámetros de diseño.

A continuación se plantean los parámetros fijados para el cálculo de los filtros, así como los cálculos mismos, para se usó un método gráfico normalizado que simplifica los cálculos.

FILTRO PASO BAJAS:

Fc = 3000 **Hz** (*Frecuencia de Corte* a 3 **dB**).

Fs = 5000 **Hz**. (*Frecuencia de Supresión de Banda* a 30 **dB**, 0.1 **dB** de rizo).

Factor de escarpamiento (Rizo):

$$\mathbf{As} = \frac{5000\ \mathbf{Hz}}{3000\ \mathbf{Hz}} = 1.66 \approx 1.7$$

De la ***gráfica normalizada***, **figura VI.3**, se obtiene que un filtro de **5o.** orden satisface las condiciones:

Si **n** = 5

Factor de Cambio de Frecuencia:

$$\mathbf{FSF} = \frac{2\pi(3000)\ \mathbf{rad}}{1\ \mathbf{rad}} = 18849.56$$

Se toma la estructura normalizada del ***filtro todo-polos*** de la **figura VI.4.**

Los ***valores normalizados*** para **n** = 5 y 0.1 **dB** de rizo son:

	C1	**C2**	**C3**
1a. ETAPA	4.446	2.520	0.3804
2a. ETAPA	6.810	0.158	

$$\mathbf{C1'} = \frac{\mathbf{C1}}{\mathbf{FSF}} = \frac{4.446\ \mathbf{farads}}{18849.56} = 2.3587 \times 10^{-4}\ \mathbf{farads}$$

Factores Z:

Para tener una escala adecuada en los valores de las resistencias y capacitores se obtienen los siguientes factores:

Si $\mathbf{Z} = 10 \times 10^{3}$

$$C1'' = \frac{C1'}{Z} = \frac{2.3587 \times 10^{-4}}{10 \times 10^{3}} = 23.587 \times 10^{-9} \text{ farads}$$

$$C2'' = 13.37 \times 10^{-9} \text{ farads}$$

$$C3'' = 2.018 \times 10^{-9} \text{ farads}$$

R = 10 Kohm

El mismo filtro se colocará en la etapa de salida del **DAC.**

FILTRO PASO ALTAS:

Fc = 200 **Hz.**

Fs = 100 **Hz.** a 30 **dB**

0.1 **dB** de rizo

***Factor de escarpamiento*:**

$$As = \frac{Fc}{Fs} = \frac{200 \text{ Hz}}{100 \text{ Hz}} = 2$$

De la gráfica normalizada, **figura VI.3**, se obtiene que un filtro de 4o. orden sería suficiente.

Si **n** = 4

$$FSF = \frac{2\pi(200) \text{ rad}}{1 \text{ rad}} = 1256.64$$

CARACTERISTICA DE ATENUACION DE UN FILTRO CHEBYSHEV

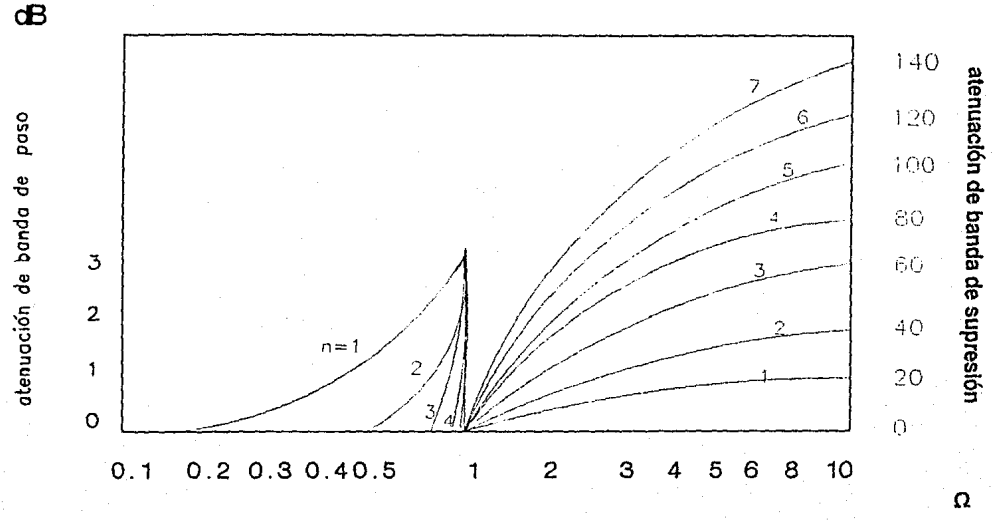


FIGURA VI.3

Para un filtro de **4o. orden** :

C1	C2
1.9	1.241
4.592	0.2410

En la **figura VI.4** se puede observar la estructura del filtro **paso bajas**. Para cambiarlo a **paso altas** se sustituye toda "R" con un capacitor de **1/R** [**farads**] y todo **C** con una resistencia de **1/C** [**ohms**].

Finalmente se le pondrá un factor de escala a los elementos tomando en cuenta las siguientes relaciones :

$$FSF = 2f\,; \quad C = \frac{C\ normalizada}{Z \times FSF}$$

R = **R** ***normalizada*** x Z

FSF = 2 200 = 1256.64

Z = 10000 (*adimensional*).

$$C = \frac{1F}{10^{3} \times 10 \times FSF} = 7.9577 \times 10^{-8}\ \textbf{farads}$$

C = 79.577 **nfarads** = 0.079 **microfarads** ≈ 0.08 **microfarads**

R1 = 5.263 **Kohms** ≈ 5.3 **Kohms**

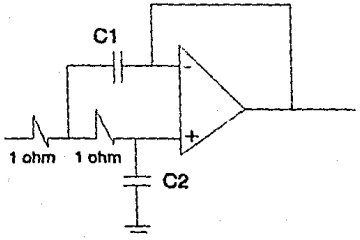
R2 = 8.058 **Kohms** = 8.1 **Kohms**

R1' = 2.177 **Kohms** ≈ 2.2 **Kohms**

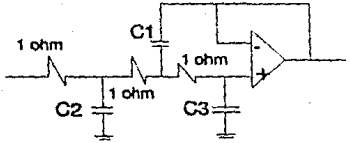
R2' = 41.494 **Kohms** ≈ 41 **Kohms**

FILTRO PASO-BAJAS TIPO CHEVYSHEV

n=2



n=3



ESTRUCTURA TODOS POLOS

FIGURA VI.4

VI.4 DISEÑO DE LA ETAPA DIGITAL DE LA SECCION DE CAPTURA Y ADQUISICION DE LA SEÑAL DE VOZ.

En base a las investigaciones realizadas se encontró que las direcciones que corresponden a **079C** y **0780** son algunas de las disponibles para el puerto de **E/S**. Se codificaron en forma binaria las direcciones para poder observar cuáles eran los bits que variaban en ese rango, y el resultado obtenido fue que había una variación en sólo 3 de los **bits**, los cuales se consideraron como líneas para un decodificador y el resto se conectaron a un arreglo de compuertas lógicas. La **figura VI.5.a**, muestra en forma gráfica las direcciones.

Como se puede observar los únicos **bits** que varían son los correspondientes a los **A2**, **A3** y **A4**. Con estos tres bits se pueden direccionar hasta 8 localidades diferentes, lo cual se podrá hacer fácilmente con un decodificador de **3 X 8** y, en este caso se eligió el circuito **74LS138**.

Las líneas de dirección **A2**, **A3** y **A4** se pueden conectar directamente a las pines **1**, **2** y **3** del **74LS138**, el cual además tiene 3 entradas para habilitar las salidas, la que corresponde a la pin número **6** habilitado con un nivel alto.

Si se toman las líneas de la **A19** a la **A12**, de la **A10** a la **A5** y de la **A1** a la **A0**, la *lógica combinacional* que iría conectada al pin **6** sería la que se muestra en la **figura VI.5.b**.

La línea de dirección **A11** va conectada directamente al pin número **5** del **74LS138**, la cual es usada también para habilitar al circuito. La terminal número **4** es usada para que el direccionamiento se realice sólo cuando haya direccionamiento a puertos, y son conectadas como se muestra en la **figura VI.6**, con lo que se completa la codificación de dirección del circuito.

En la **figura VI.6** se muestra el circuito general de la etapa digital de la sección de captura y adquisición de la señal de voz.

VI.5 DISEÑO DE LA ETAPA ANALOGICA DE LA SECCION DE SALIDA Y REPRODUCCION DE LA SEÑAL DE VOZ.

Para el convertidor analógico digital se utiliza un **DAC 08**. Este circuito tiene internamente un arreglo **R/2R**, el cual tiene la característica de que las resistencias usadas para su construcción no tienen que ser potencias de dos del primer valor, si no que basta con que una resistencia sea el doble de la otra.

En la **figura VI.7** podemos observar la etapa de *conversión analógica digital* de la sección de salida y reproducción de la señal de voz.

Para el caso específico del **DAC 08**, el manual "**Linear Databook**" de **National Semiconductor** da los principales parámetros a calcular y la forma de hacerlo se detalla a en la siguiente sección.

DIRECCION DEL PUERTO DE SALIDA

0	0	7	9	C
0 0 0 0 0 0 0 0	0 1 1 1	1 0 0 1	1 1 0 0	

A19 A0

DIRECCION DEL PUERTO DE ENTRADA

0	0	7	8	0
0000	0000	0111	1000	0000

A19 A0

FIGURA VI.5.A

DIAGRAMA ELECTRICO DE DECODIFICACION

DE DIRECCIONES

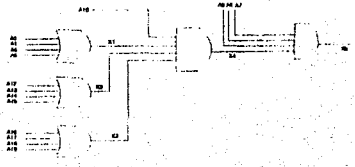


FIGURA VI.5.B

ETAPA DIGITAL DE LA SECCION DE CAPTURA

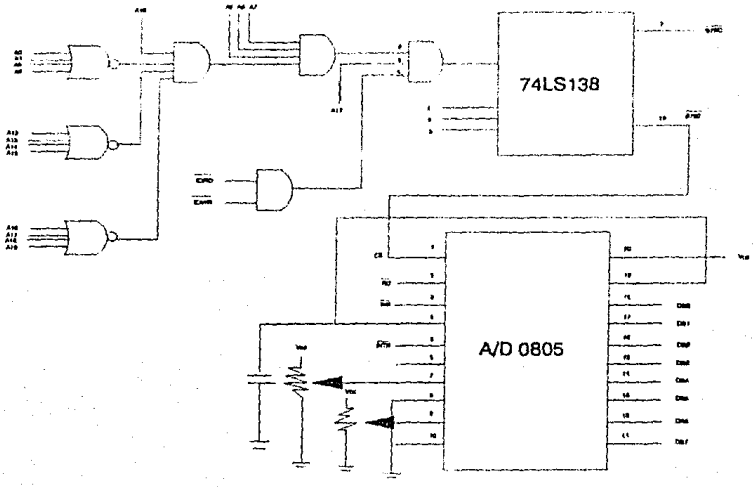


FIGURA VI.6

VI.6 ANALISIS DEL DAC 08

Las **figura VI.7** muestra el diagrama de pines del *convertidor digital analógico* **DAC 08.**

Las fórmulas tomadas del manual **Linear Data Book** son las siguientes:

$2^8 = 256$

$2^8 - 1 = 255$

$$IFS \approx \frac{Vref}{Vref} \times \frac{255}{256} \quad .... (8)$$

$$\overline{Io} + Io = ISF \quad .... (9)$$

$$Iref = \frac{Vref}{Rref} \quad .... (10)$$

Si **Vref = 5 V** y **Rref = 5 Kohm**

$$IFS = \frac{5\ V}{5\ Kohm} = 1\ mA$$

Si las salidas se conectan como lo indica la **figura VI.7** se tiene que:

$$IFS = \frac{5\ V}{5\ Kohm} \times \frac{255}{256} = 0.996\ mA$$

como $\overline{Io} = 0$ **A**

entonces: **Io = IFS = 0.996 mA**

Eo = -Io(5 Kohm) = 4.98 V

DIAGRAMA DE PATAS DEL DAC08

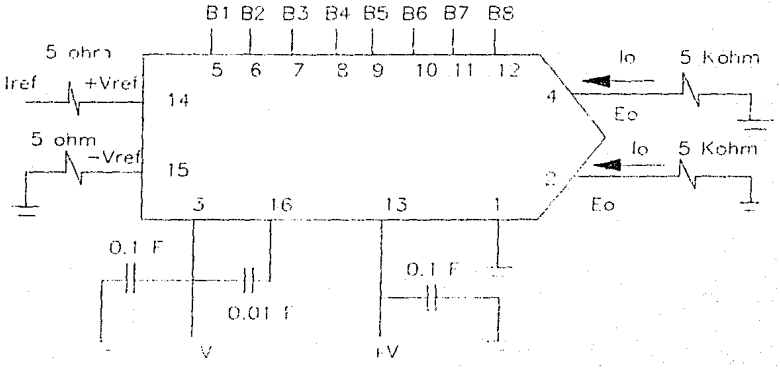


FIGURA VI.7

Si se desea que las salidas queden entre (+/-) 1.5 V para no saturar los amplificadores operacionales se tiene:

$$Eo = -1.5\ V = -Io(RL)$$

$$Io = 0.996 \quad \textit{para escala completa}$$

por lo tanto:

$$RL = \frac{1.5}{Io} = \frac{1.5}{0.996 \times 10^{-3}} = 1506.02\ \textbf{ohms}$$

RL = 1.5 K ohms.

VI.7 DISEÑO DE LA ETAPA DIGITAL DE LA SECCION DE SALIDA Y REPRODUCCION DE VOZ.

Para la etapa de salida, se requieren básicamente 3 bloques que son:

1.- Decodificador de direcciones.

2.- Acoplamiento del bus de datos.

3.- Convertidor digital/analógico.

Que se muestran graficamente en la **figura VI.8.**

DIAGRAMA DE BLOQUES DE LA ETAPA DIGITAL DE LA SECCION DE SALIDA Y REPRODUCCION DE LA SEÑAL

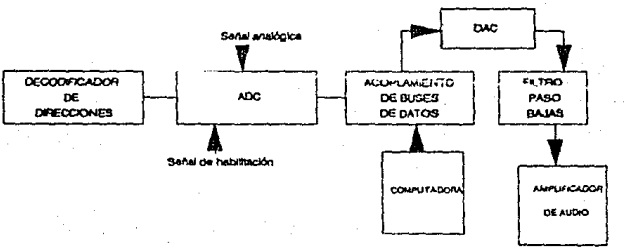


FIGURA VI.8

La etapa digital está formada por el decodificador de direcciones, que es el mismo que aparece en la **figura VI.6**, además del acoplamiento del bus de datos, el cual se muestra con más detalle en la **figura VI.7**, estas figuras aparecen a partir de la página 60.

En la **figura VI.8**, descrita en la página 61, se muestra el diagrama de bloques completo de como van a ir estructuradas las etapas del circuito.

El filtro **paso banda** es el mismo usado en la etapa de entrada y los parámetros, así como los cálculos son idénticos.

El amplificador de audio usado es el **LM386** de **National**, el cual fué seleccionado por su simplicidad, bajo costo y por requerir sólo de una fuente de alimentación, lo cual simplifica el diseño de la tarjeta y reduce su costo.

CAPITULO VII

VII. ETAPA DE SOFTWARE

JUSTIFICACIONES AL DESARROLLO DEL SISTEMA"FONETICS"

Para la lectura de un texto por computadora y su posterior síntesis en forma de voz, existen 2 opciones:

La primera de ellas y la más usada hasta el momento, es desarrollar un sistema que posea suficiente cantidad de reglas gramaticales particulares para cada idioma.

La segunda opción es crear un sistema inteligente capaz de aprender reglas gramaticales. Estas reglas son consultadas por el instructor, y pueden ser actualizadas. Cada actualización es almacenada en una localidad de memoria. El sistema asume las reglas gramaticales que rigen la separación silábica de cada palabra del texto. Si la separación silábica asumida es correcta y además lo es también la regla gramatical que la rige, entonces el instructor no tendrá necesidad de consultar, si es correcta o no, la regla gramatical. Este sistema es el sistema **FONETICS**, nombre del proyecto de tesis, el cual tiene la ventaja de poderse aplicar a cualquier estructura idiomática, hasta puede aplicarse a cuestiones musicales, con la diferencia de que se tendría que hacer una adaptación para que leyera un pentagrama.

Dada la versatilidad de la opción anteriormente expuesta se decidió adaptar este criterio al proyecto de tesis.

El instructor, inicialmente, tiene que alimentar a la memoria de la computadora con separaciones silábicas para las primeras palabras a analizar en el texto. La computadora va a asumir el siguiente caracter, de la palabra, seguido de la separación silábica dada, como una regla gramatical. Por lo que una separación silábica puede estar regida por más de una regla gramatical, es decir, se crea un vector de hasta veinte elementos, que almacene todos los casos en que se puede presentar esa sílaba, guardando en cada caso el caracter que le sigue a esa sílaba.

Por esta razón, el mismo sistema se autoconsulta y va aprendiendo la gramática de las palabras del texto.

A continuación se presenta un ejemplo de cómo es la dinámica del sistema.

En primer lugar se despliega, la primera línea de texto anteriormente creado en memoria en disco.

Por ejemplo si el texto a analizar fuese el siguiente:

" ESTE ES UN TEXTO DE PRUEBA "

El sistema va a separar cada palabra del renglón a analizar. La primera palabra a analizar es "**ESTE**".

Después pregunta cuál es su separación silábica: La primera sílaba es "**ES**" con la regla gramatical "**T**".

Después vuelve a preguntar por la siguiente separación silábica, la cual es "**TE**" con la regla gramatical " ". Por lo que el sistema almacena en disco estas separaciones, junto con las reglas gramaticales correspondientes.

Entonces dada la palabra "**ESTE**", la separación silábica correspondiente es "**ES-TE**".

Ya una vez almacenados estos datos, el sistema se dispones a analizar la siguiente palabra que es "**ES**".

Como la palabra "**ES**" corresponde a la sílaba "**ES**" de la palabra anteriormente analizada, entonces el sistema sólo preguntará si la regla gramatical asumida es correcta, si no es correcta entonces el instructor tiene que definir la separación silábica. Pero, como en este caso, se asume como regla el caracter " ", y es correcto, por lo que ya no hay necesidad de definir la separación silábica, tomando la sílaba de la palabra "**ES**" como "**ES**".

Otro caso de interés se da al analizar la palabra "**TEXTO**", ya que anteriormente, en la palabra "**ESTE**" se analizaron las sílabas "**ES**" y "**TE**",por consiguiente, el sistema **FONETICS**, en un principio, asume para la palabra "**TEXTO**", la siguiente separación:

"TE-XTO"

Como esta separación está regida por la regla " ", que anteriormente tenía la sílaba "**TE**", entonces el sistema pregunta si es correcta esa regla para esa separación. Ya que no es correcta, el sistema nos permite "redefinir" una nueva separación silábica, la cual en este caso es: "TEX" con la regla gramatical "**T**".

Después de definir la separación "**TEX**". El sistema asume la siguiente separación como "**TO**" y el sistema pregunta si es correcta, en el caso de que sea correcta, asume también que la regla gramatical de "**TO**" es " ". Si no fuera correcta, el instructor del sistema tendría que definirla.

Siguiendo con esta secuencia de la dinámica del sistema, el renglón del texto "**ESTE ES UN TEXTO DE PRUEBA**", quedaría analizando gramaticalmente de la siguiente forma:

ES-TE ES UN TEX-TO DE PRUE-BA

La primera sílaba es "**ES**" con la regla "**T**".

La segunda sílaba es "**TE**" con la regla " ".

La tercera sílaba es "**ES**" con la regla " ".
Por lo cual se agrega a la lista de reglas de la sílaba "**ES**" la regla " ".

La cuarta sílaba es "**UN**" con la regla " ".

La quinta sílaba es "**TEX**" con la regla "**T**".

La sexta sílaba es "**TO**" con la regla " ".

La séptima sílaba es "**DE**" con la regla " ".

La octava sílaba es "**PRUE**" con la regla "**B**".

La novena sílaba es "**BA**" con la regla " ".

El análisis seguiría de esta forma en todas las líneas del texto hasta aprender todas las sílabas y sus reglas correspondientes. Se encontró que al almacenar sólo una regla hacia adelante (la letra siguiente a la sílaba) era suficiente para evitar ambigüedades en la mayoría de los casos (excepto con las sílabas que contengan "r" o "rr" que requieren como mínimo una regla hacia atrás, es decir la letra anterior), aún cuando no se descarta la posibilidad de que para algún idioma en particular se requiera un análisis de "n" reglas hacia adelante e inclusive "m" reglas hacia atrás.

Para que el sistema **FONETICS** pueda manejar tanto las sílabas como las reglas gramaticales, cuenta con una tabla de sílabas y reglas gramaticales, como se muestra a continuación en la **figura VII.1**, correspondiente a una tabla.

SILABA	REGLAS
"ES"	" ","T", ...
"UN"	" ", ...

FIGURA VII.1

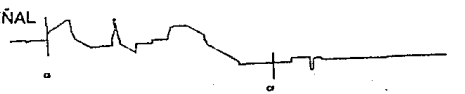
Una vez teniendo la sílaba y sus reglas asociadas, se procede a capturar el fonema correspondiente por medio de la tarjeta digitalizadora de voz. Como el fonema puede estar acompañado de lapsos de silencio tanto al inicio como al final de éste, es necesario "recortarlo" para que contenga sólo la información útil, esto se hace mediante una rutina que grafica la voz digitalizada y posteriormente es cortada por el instructor hasta obtener el fonema deseado.

Se utiliza un ciclo de captura de aproximadamente 6 KHz, para así tener una sincronía en la captura del puerto.

Esto fue posible a través del Lenguaje TurboPascal que hace uso de direcciones físicas y rutinas ya implementadas de manejo de strings (cadena de caracteres), manejo de pantallas, de ventanas (windows)y graficación de la señal en monitores monocromaticos y a colores.

A continuación se muestra una señal recortada correspondiente a un fonema en la **figura VII.2**.

PROCESO DE CORTE DE LA SEÑAL



SEÑAL RECORTADA Y ALMACENADA PARA ANALIZAR

ci : Corte inicial de la señal.
cf : Corte final de la señal.

FIGURA VII.2

Una vez cortado el fonema, se compacta, se almacena en disco y se concatena en un vector de reproducción, el cual contiene todos los fonemas de la palabra que se está analizando, así, en cuanto se termine el análisis de esta palabra, se procederá a su reproducción en forma de voz.

Este proceso se repite para cada sílaba de cada palabra en todo el texto. Es evidente que entre más texto lea el sistema, tendrá menos problemas en el aprendizaje, es decir, en las primeras palabras se tiene que enseñar, tanto las sílabas como los fonemas, pero en palabras posteriores se tendrá que enseñar sólo las reglas y ya no la sílabas ni los fonemas, ya que el sistema los aprendió anteriormente, haciéndose así menos tedioso su entrenamiento.

Algunas mejoras que se pueden hacer a futuro en el mismo sistema son:

1.- Reconocimiento automático de silencios para permitir que la computadora por si misma reconozca y corte el fonema sin intervención del instructor.

2.- Elaboración de rutina de edición de fonemas y sílabas para depuración y corrección.

3.- Elaboración de rutina para lectura de texto con estructura en forma de árbol (se utilizaría ya que el sistema haya aprendido todos los fonemas).

4.- Detección y entonación de símbolos de puntuación.

En el apéndice "B" se muestran :

1.- Diccionario de datos con breve explicación de las rutinas del sistema **FONETICS**.

2.- Listado del programa del sistema **FONETICS**.

VII.1. MANUAL DE OPERACION DEL SISTEMA "FONETICS"

Este manual describe el funcionamiento del sistema **FONETICS**, Generador de Voz por Compactación de Fonemas, a través de las pantallas de comunicación con el usuario de una forma detallada, con un lenguaje sencillo enfocado a los usuarios que tienen muy poco conocimiento de lo que es una computadora.

VII.1.1 BREVE EXPLICACION DEL FUNCIONAMIENTO DEL SISTEMA "FONETICS"

SISTEMA FONETICS :

Es un sistema realizado en **TurboPascal V.3.0**. Este sistema utiliza tres tipos de archivos :

a) Un archivo que contiene el texto escrito.

b) Un archivo que contiene la señal compactada de cada fonema grabado.

c) Un archivo con la cadena de caracteres correspondiente al fonema grabado y sus respectivas reglas gramaticales, así como apuntadores de acceso al archivo anterior.

INTERACCION ENTRE ARCHIVOS.

En un principio pueden estar inicializados, o no creados.

El primero que se crea es el archivo del texto escrito. Luego el de los fonemas que se van a grabar y después el de las tablas de caracteres relacionados con los fonemas grabados. A través de estos archivos se establece una comunicación Usuario- Máquina de la siguiente manera :

1) Ya una vez creado el texto a reproducir en voz, el programa va a analizar línea por línea del texto escrito, y cada línea va a estar separada en palabras, las cuales van a ser analizadas por sílabas. En un principio el instructor tendría que dar las separaciones silábicas correspondientes al fonema que se desee grabar. Pero después, el sistema va aprendiendo a hacer separaciones silábicas.

2) Dado el fonema en forma de sílaba escrita, el usuario pronunciará ante un micrófono, conectado a la tarjeta digitalizadora de voz, el fonema correspondiente. Ya una vez grabado y compactado éste, se almacenará en el archivo de fonemas.

3) Al analizar la sílaba, también se almacenó ésta en el archivo de tablas, al igual que la regla gramatical que la rige, la longitud, en bytes, del fonema y su localización en el archivo de fonemas.

RUTINAS PRINCIPALES DEL SISTEMA FONETICS :

a) ***Proceso de Captura de la Señal de Voz***, haciendo uso de un puerto de entrada.

b) ***Proceso de Compactación de la Señal de Voz***, mediante Métodos Numéricos que representan en forma más compacta la señal.

c) ***Proceso de Descompactación***, haciendo uso de Métodos Numéricos que recuperan la señal capturada a través de la señal compactada con un bajo porcentaje de error.

d) ***Proceso de Análisis del Texto Escrito***, en donde se hacen llamadas a los archivos del texto escrito, de fonemas y de tablas de reglas gramaticales, para crearlos, actualizarlos y dándole inteligencia al sistema para que aprenda a separar por sílabas cada palabra analizada, almacenando las reglas gramaticales y haciendo acceso al archivo de fonemas, para así reproducir en voz, la palabra que se escribió y se analizó silábicamente.

e) ***Reproducción de la Señal de Voz*** utilizando un puerto de salida.

Existen, además, otras dos rutinas alternas, que sirven para que el instructor pueda visualizar la señal en la pantalla, como si la viera en un osciloscopio.

a) ***Rutina de Graficación de la Señal Capturada***.

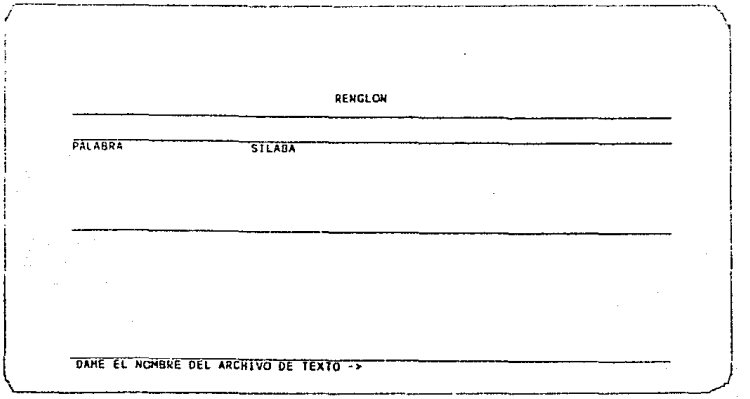
b) ***Rutina de Corte y Análisis de la Señal Graficada***.

Estas rutinas le ayudan al instructor a tener la imagen de lo que ha grabado, y poder analizar la estacionariedad o no estacionariedad de la señal.

En seguida se explicará el funcionamiento del sistema haciendo una presentación de cada pantalla que despliega el sistema como enlace con el ususario.

SECCION DE PANTALLAS

VII.1.2 SECUENCIA DE PANTALLAS DEL MANUAL DEL SISTEMA "FONETICS"



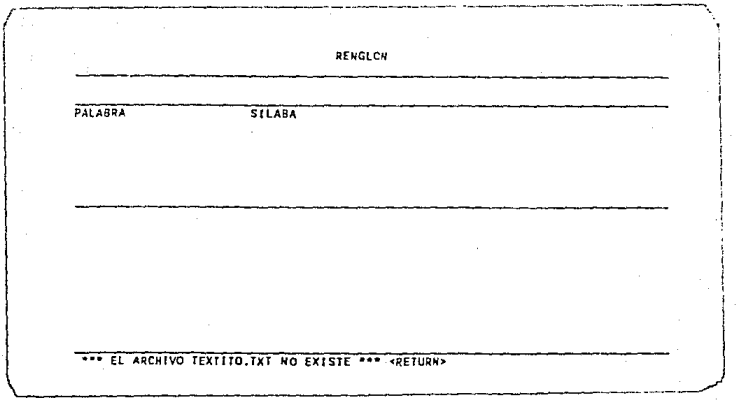
PANTALLA # 1

Esta es la ***Pantala de Inicio***, en la cual se le pide al instructor que le dé un nombre al archivo del texto a crear, o a llamar, en caso de que haya sido creado anteriormente. Se teclea el nombre del archivo del texto con longitud de 8 caracteres, se agrega un punto, y después 3 caracteres del nombre de la extensión.

< 8 caracteres máx. del nombre > "." < 3 caracteres de la extensión >

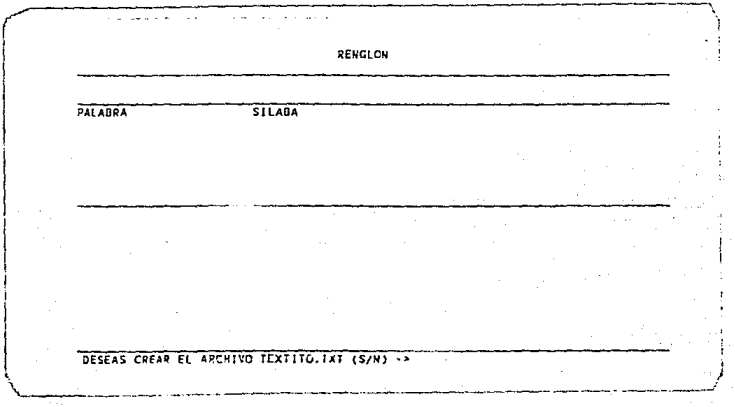
Por ejemplo, si el instructor ya había creado un archivo con el nombre de **TEXTO.TXT** , entonces se puede teclear **TEXTO.TXT** y <*return*> y cargará ese archivo listo para analizar.

Si el usuario teclea **TEXTITO.TXT**, entonces el programa desplegará una pantalla donde marca un mensaje de no existencia de ese archivo.



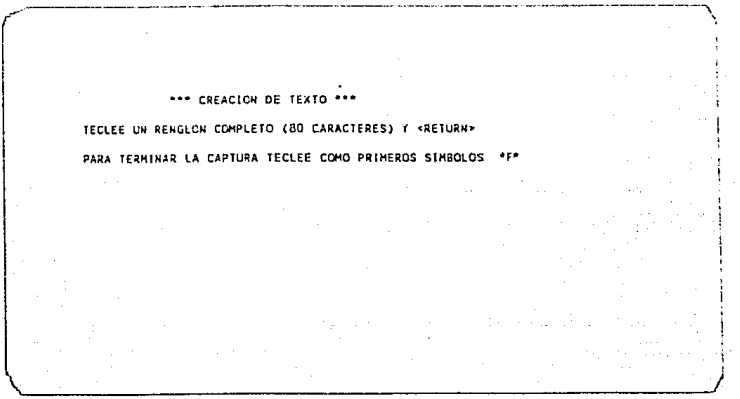
PANTALLA # 2

Esta pantalla se presenta cuando el usuario da un nombre de archivo que no existe. En este caso tiene la opción de crear o no crear el archivo. Basta con oprimir *<la tecla de return>* para pasar a la siguiente pantalla.



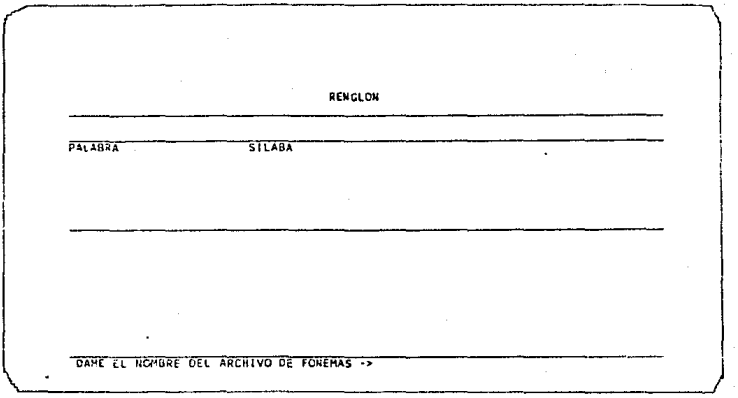
PANTALLA # 3

Esta pantalla le da, al instructor, la opción de crear o no crear el nuevo archivo de texto. Si la respuesta es afirmativa, pasará el usuario a la Pantalla de Creacion de Archivo del Texto Escrito. Si la respuesta es negativa el usuario regresará a la Pantalla de Inicio.



PANTALLA # 4

En esta pantalla se crea el texto que se desea analizar, haciendo uso de un editor de texto. Para concluir la creación del archivo hay que poner en el último renglón del texto escrito la cláusula " *F* ", como primeros caracteres del renglón final.

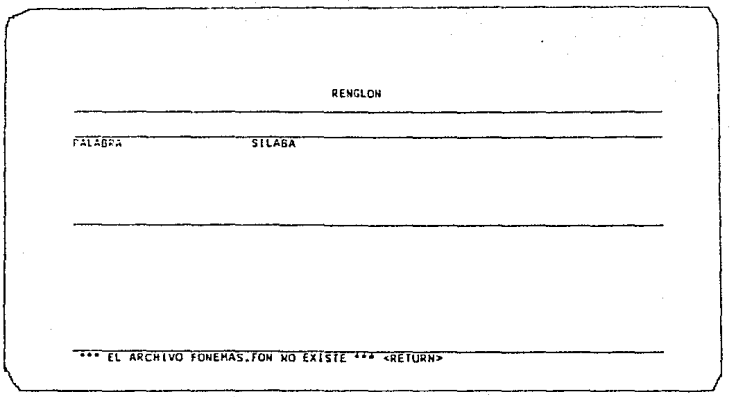


PANTALLA # 5

Se pide que se dé el nombre del archivo de fonemas, que puede o no haber sido creado anteriormente por el instructor. La forma de escribir el nombre del archivo es como ya se ha descrito anteriormente :

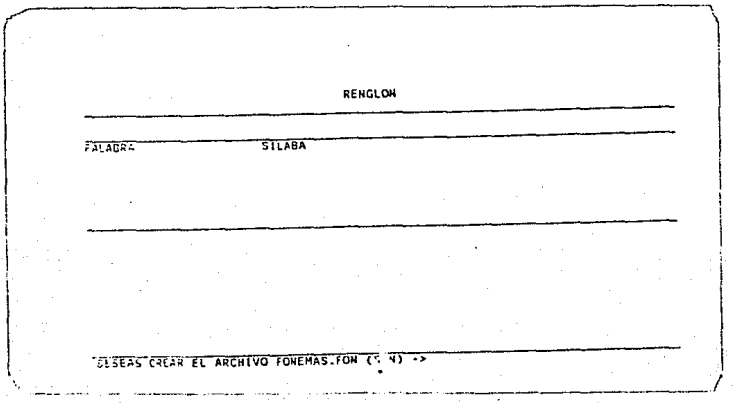
< 8 caracteres máx. del nombre > "." < 3 caracteres de la extensión >

Si se teclea el nombre de un archivo de fonemas que no existe, enseguida el programa desplegará una pantalla que nos indica su no existencia. Para trasladarse de una pantalla a otra, basta con oprimir la tecla de <*return*> . Por ejemplo si el instructor ya había creado un archivo llamado **FONEMAS.FON**; si vuelve a teclearlo estará invocándolo, es decir, se opera con los datos anteriores. Si se teclea un nombre distinto al que se había creado, como FONEM.FON, entonces desplegará un mensaje que anuncia la no existencia de tal archivo.



PANTALLA # 5.1

Indica la no existencia del archivo dado por el usuario .



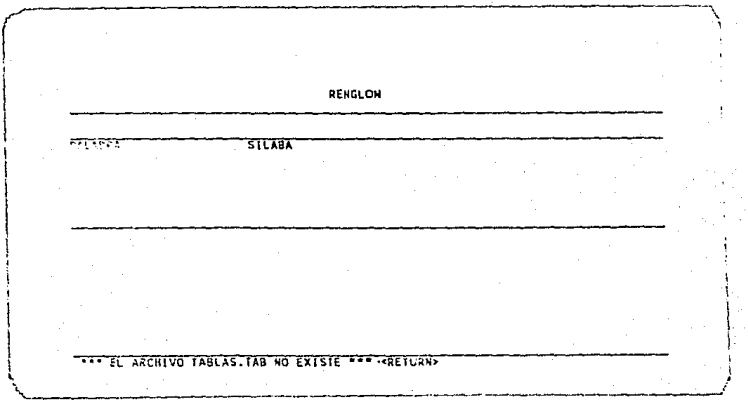
PANTALLA # 5.2

Se presenta la opción de crear ese nuevo archivo. Si se contesta afirmativamente, se crea un nuevo archivo con el nombre que se le dió. Si se contesta en forma negativa, volverá el programa a desplegar la pantalla donde se le pide al instructor nuevamente el nombre del archivo de fonemas.

En esta pantalla se le pide al instructor el nombre del archivo de tablas de fonemas y reglas gramaticales. Se teclea el nombre del archivo de tablas de la siguiente manera :

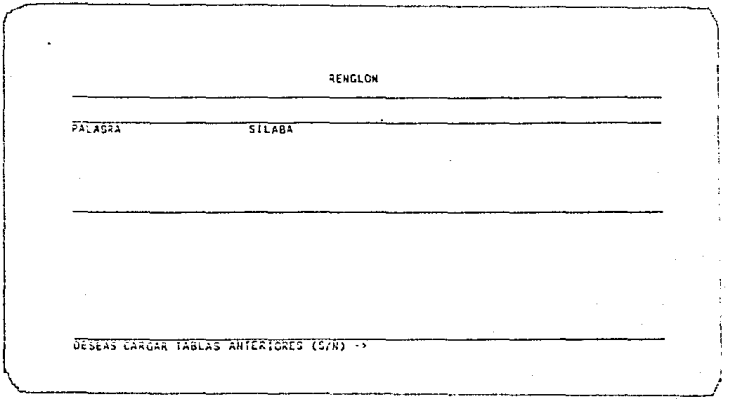
< 8 caracteres máx. del nombre > "." < 3 caracteres de la extensión >

Si se escribe un nombre distinto al que se había creado anteriormente, entonces se identifica su "no existencia" y se solicita su creación. Por ejemplo, si el instructor creó un archivo llamado **TABLAS.TAB** ; al volver a escribir su nombre, lo estará llamando. Si escribe un nombre distinto al ya creado, entonces tendrá la opción de crearlo o no crearlo, como por ejemplo TABLITA.TAB.



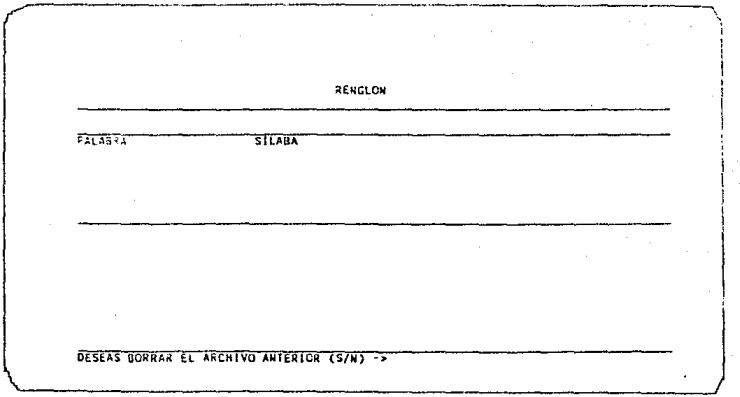
PANTALLA # 6.1

Le indica al instructor la no existencia del archivo escrito anteriormente. Lo cual le da opción al instructor de crear o no crear tal archivo.



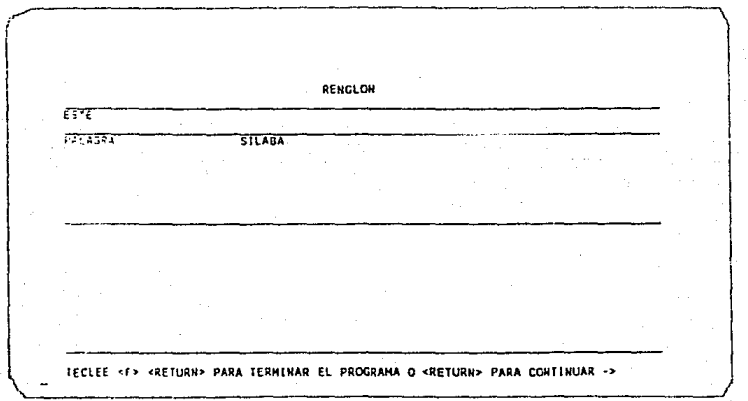
PANTALLA # 7

Pantalla que solicita la carga de tablas anteriores. Es decir, mantener o no mantener la información referente a las separaciones silábicas analizadas anteriormente. Si se contesta afirmativamente, se cargan esas tablas y se le solicita al instructor desplegar o no desplegar en pantalla esas tablas. Se sugiere borrar las versiones anteriores a menos que se desee hacer alguna actualización en las tablas existentes ya creadas.



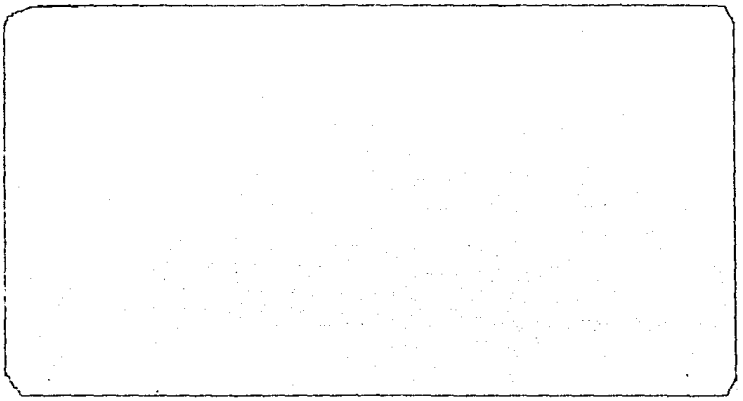
PANTALLA # 8

En esta pantalla se le da al instructor la opción de borrar o no borrar las versiones anteriores.



PANTALLA # 9

Pantalla de Introducción al Análisis del Texto. Se le da al instructor la opción de continuar con el proceso o iniciar el análisis oprimiendo *<return>*, o de terminar el análisis y la ejecución del programa oprimiendo la tecla <F>.



PANTALLA # 9.1

Pantalla de Terminación de Ejecución del Programa.

RENGLON

ESTE

PALABRA ESTE

SILABA ESTE

*** NO ENCONTRE EL FONEMA "ESTE "
DE LA PALABRA "ESTE "
EN LA TABLA, DAME EL FONEMA CORRECTO ->

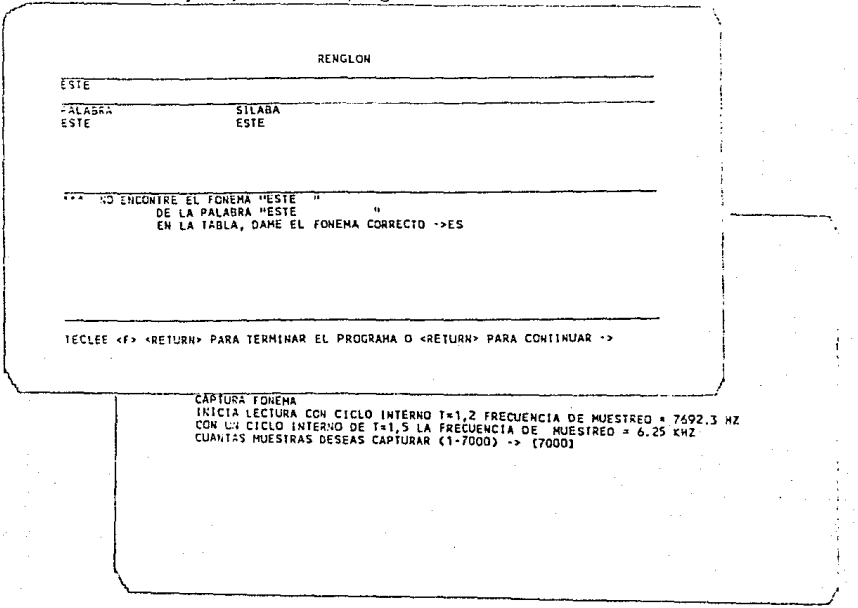
PANTALLA # 10

Esta pantalla le muestra al instructor un renglón del texto escrito a analizar, palabra por palabra, así como las sílabas que la constituyen.

Se le presenta al instructor la palabra de la línea de texto a analizar.

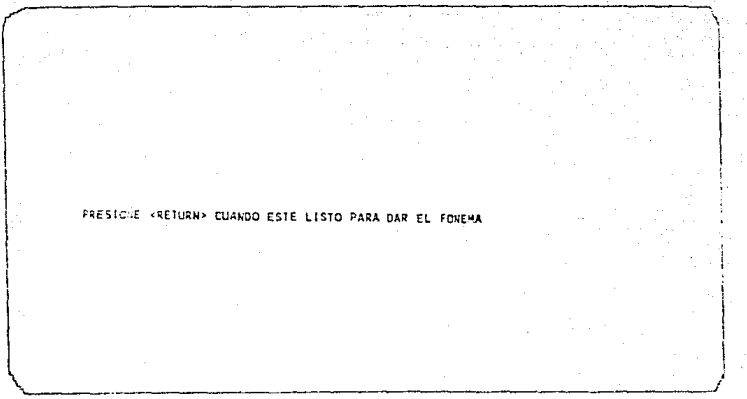
Se escribe la separación silábica adecuada. Oprimir <*return*> para continuar con el proceso y mas adelante capturar el fonema correspondiente a la separación silábica.

Ya una vez analizada la palabra completa, con sus respectivas separaciones silábicas se puede optar por continuar el proceso de análisis del texto, o terminar el análisis y la ejecución del programa.



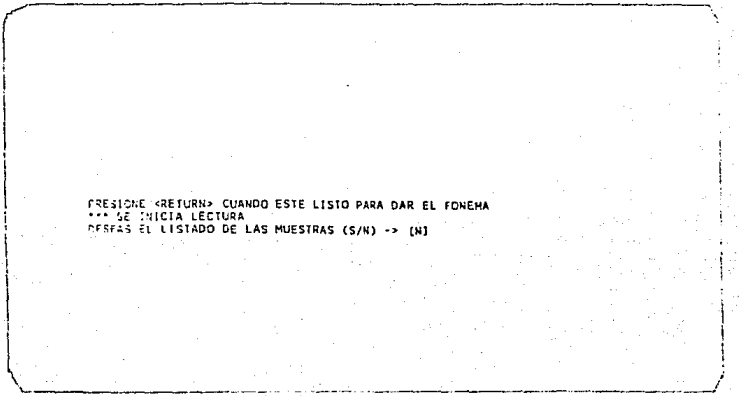
PANTALLA # 11

Esta es la Pantalla de Captura del Fonema. Se puede escribir el número de muestras a capturar, las cuales varian de [1 a 7000] o se puede asumir el número de muestras dado por *default* **[5000 muestras]** tecleando <*return*> .



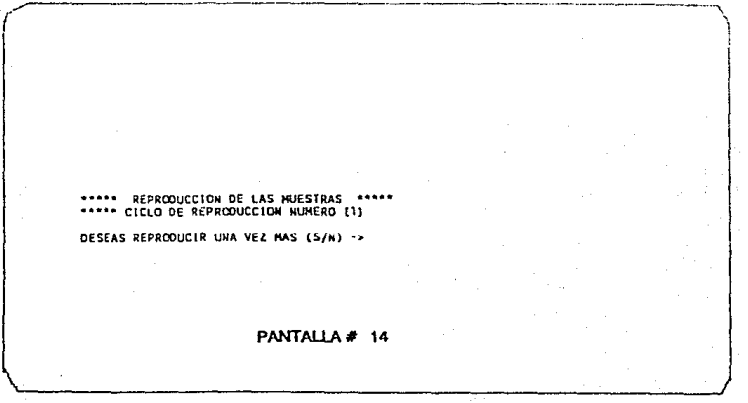
PANTALLA # 12

En esta pantalla se prepara al instructor para pronunciar el fonema correspondiente, ante el micrófono con voz alta y dicción clara, después de oir el sonido del "beep".



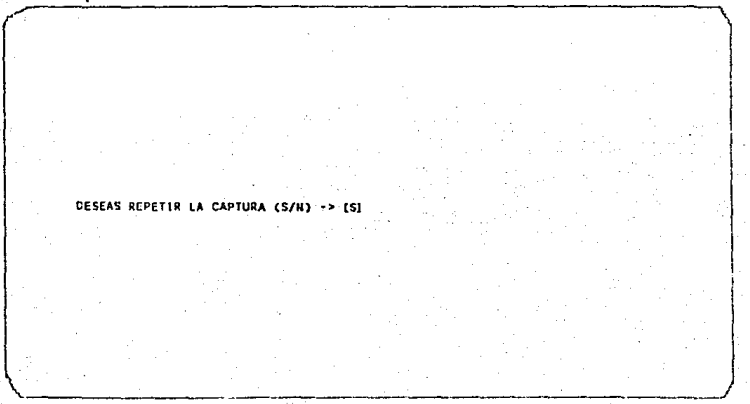
PANTALLA # 13

En esta pantalla se le da al usuario la opción de listar las muestras anteriormente capturadas o no hacer listado de las muestras, sino sólo de los ciclos de reproduccion, que son cinco para que el instructor del sistema esté seguro de que se capturó el fonema completo.



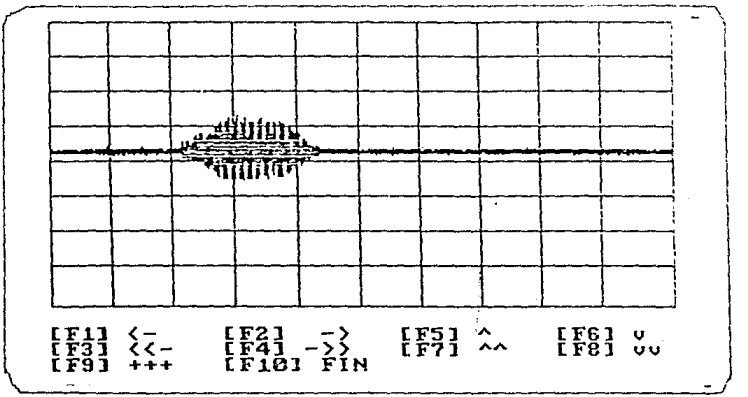
PANTALLA # 14

Se muestran los ciclos de reproducción. Sólo hay que oprimir la tecla de < *return* > para continuar.



PANTALLA # 15

Esta pantalla nos da la opción de volver a repetir la captura en caso de que no haya sido satisfactoria. Si se contesta afirmativamente, quiere decir que el instructor desea repetir el proceso de captura del fonema.

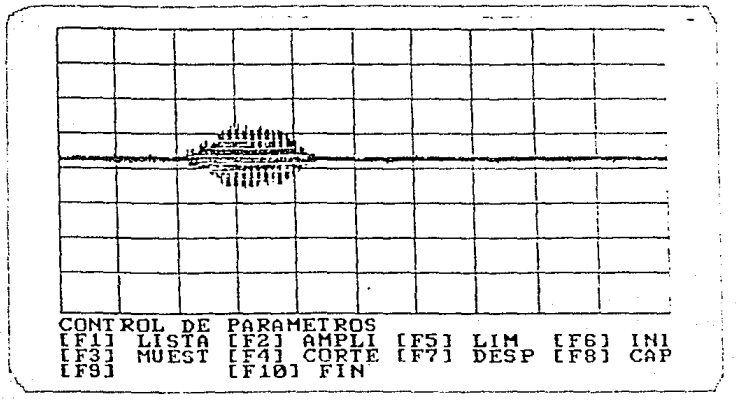


PANTALLA # 16

Esta es la Pantalla Principal de Graficación de la Señal. En la parte inferior se presentan varias funciones :

F1 : " < " desplazamiento mínimo hacia la izquierda.
F2 : " > " desplazamiemto mínimo hacia la derecha.
F3 : " << " desplazamiento máximo hacia la izquierda.
F4 : " >> " desplazamiento máximo hacia la derecha.
F5 : " ^ " desplazamiento mínimo hacia arriba.
F6 : " v " desplazamiento mínimo hacia abajo.
F7 : " ^ ^ " desplazamiento máximo hacia arriba.
F8 : " vv " desplazamiento máximo hacia abajo.
F9 : " +++" función que traslada al instructor a otro menú con más opciones.
F10 : " FIN" significa fin de proceso. Pero éste no puede ser ejecutarse sin antes hacer cortes de análisis y reproducción de la señal.

Si se quiere analizar más detalladamente la señal se selecciona la función F9 para pasar a la siguiente pantalla.



PANTALLA # 17

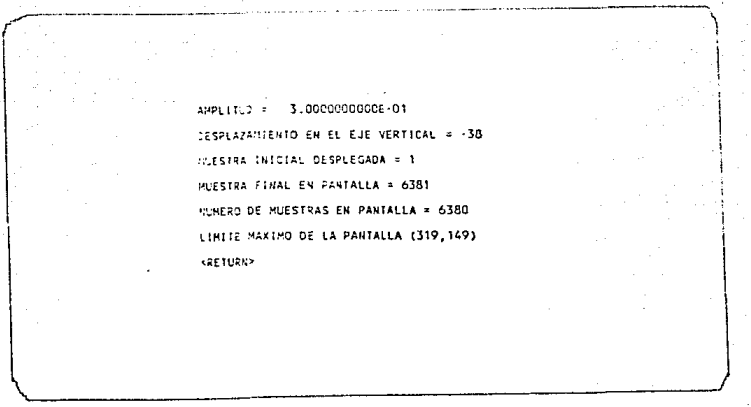
Esta pantalla nos presenta la misma gráfica de la señal, pero con un menú de control de parámetros de la señal. A continuación se presentará la pantalla o pantallas asociadas a las nuevas funciones :

F1 : "**LISTA**" llama a una pantalla que lista los parámetros actuales.
F2 : "**AMPLI**" llama a una pantalla que controla los parámetros de amplitud.
F3 : "**MUEST**" llama a una pantalla que controla el incremento entre muestras.
F4 : "**CORTE**" llama a una pantalla que controla los factores de corte de análisis de la señal.
F5 : " **LIM** " llama a una pantalla que muestra los límites máximos de la pantalla.
F6 : " **INI** " llama a una pantalla de cambio en los parámetros actuales por los iniciales.

F7 : "**DESP** " llama a una pantalla de control de desplazamientos mínimos y máximos tanto en el eje "X" como en el eje "Y".
F8 : {*ninguna función*}
F9 : {*ninguna función*}
F10 : " **FIN** " fin de proceso, pero sólo si se reproduce antes la señal, si no se reproduce, el programa despliega el mensaje "**ANTES DE REPRODUCIR HAY QUE PASAR POR LA RUTINA DE CORTE DE LA SEÑAL**".

Si no se reproduce esa señal antes, el programa muestra un mensaje de proceso incompleto, indicando que primero hay que pasar por la rutina de corte, reproducir la señal y oprimir **<F10>**.

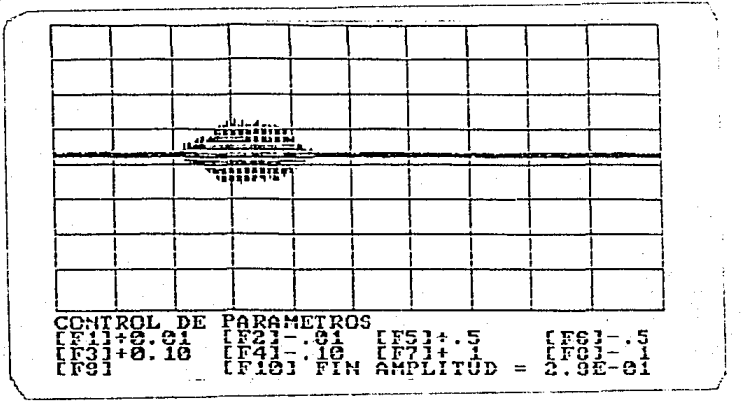
Si el instructor oprime **<F1>** obtendrá un listado de los parámetros de la señal.



PANTALLA # 18

Se presenta el listado de los parámetros. Para pasar a la pantalla principal de graficación sólo hay que oprimir la tecla de *<return>*.

Ya una vez estando el instructor ubicado en la ***Pantalla Principal de Graficación***, oprime F9 y se desplazará a la ***Pantalla de Control de Parámetros Principal***, y si el instructor oprime F2 se trasladará a la siguiente pantalla.

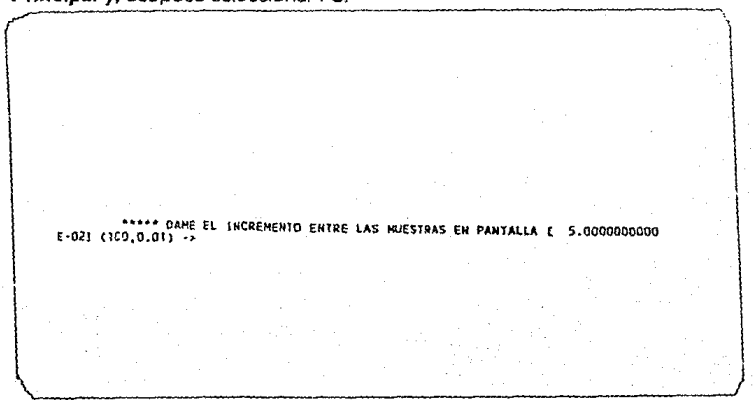


PANTALLA # 19

Es la pantalla que resulta de oprimir F2. Con esta opción se puede modificar el factor de amplificación de la señal, pasando a otro menú con otras opciones que comprenden a :

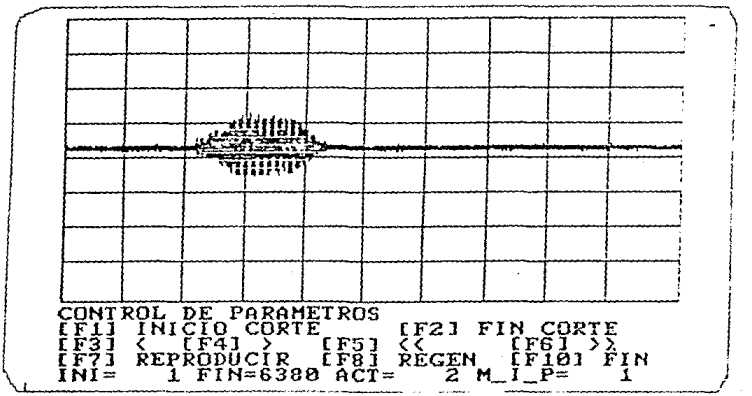
F1 : **+0.01** Aumenta la amplitud en **0.01** unidades.
F2 : **-0.01** Disminuye la amplitud en **0.01** unidades.
F3 : **+0.10** Aumenta la amplitud en **0.10** unidades.
F4 : **-0.10** Disminuye la amplitud en **0.10** unidades.
F5 : **+0.50** Aumenta la amplitud en **0.50** unidades.
F6 : **-0.50** Disminuye la amplitud en **0.50** unidades.
F7 : **+1.00** Aumenta la amplitud en **1.00** unidad.
F8 : **-1.00** Disminuye la amplitud en 1.00 unidad.
F9 : { *No realiza ninguna función* }
F10 : Retorno a la Pantalla Principal de Graficación.

Para seleccionar la función F3 de la ***Pantalla de Control de Parámetros Principal***, se oprime F10 para regresar a la ***Pantalla Principal de Graficación***, luego oprimir **F9** para pasar al menú de la ***Pantalla de Control de Parámetros Principal*** y, después seleccionar **F3**.



PANTALLA # 20

Pantalla que nos pide el incremento entre muestras. El incremento de *default* es **[1.6667 e-4** unidades aproximadamente **]** y se da con sólo teclear *<return>*. Si se desea modificar el incremento, sólo hay que teclear el valor del incremento y *<return>*.



PANTALLA # 21

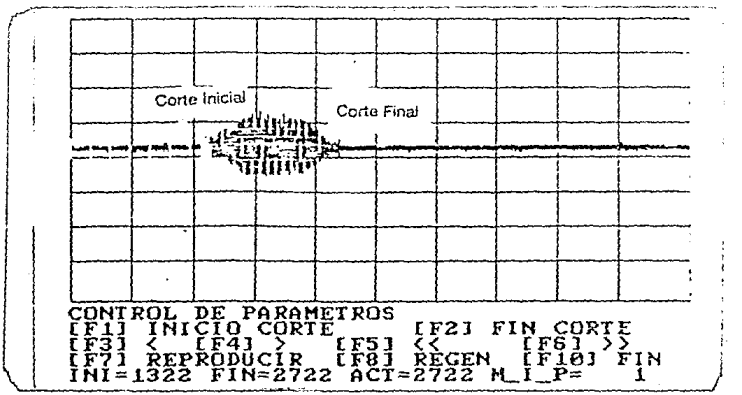
Presenta la ***Pantalla de Control de Parámetros de Corte*** (*). Se presentan las diferentes opciones para hacer cortes de análisis. Los cortes se tienen que hacer para almacenar y reproducir una parte de la señal que suene casi igual o igual a la señal original de voz muestreada.

F1 : " **INICIO CORTE** " se marca el punto de la señal donde se desea hacer el corte inicial de la señal.
F2 : " **FIN CORTE** " se marca el punto de la señal donde se desea hacer el corte final de la señal.
F3 : " **<** " desplazamiento mínimo de la marca de corte hacia la izquierda.
F4 : " **>** " desplazamiento mínimo de la marca de corte hacia la derecha.
F5 : " **<<** " desplazamiento máximo de la marca de corte hacia la izquierda.
F6 : " **>>** " desplazamiento máximo de la marca de corte hacia la derecha.
F7 : " **REPRODUC** " opción de reproducción en forma audible de la señal capturada y descompactada.
F8 : " **REGEN** " regeneración de los puntos marcados de la señal graficada.
F9 : { *No ejecuta ninguna acción* }
F10 : " **FIN** " fin del proceso de corte de análisis.

(*) Para seleccionar las subsecuentes funciones de la ***Pantalla de Control de Parámetros Principal***, el instructor debe oprimir **F10**, después **F9** y después la función que el quiera seleccionar. Si selecciona **F4** pasará a la ***Pantalla de Parámetros de Corte***.

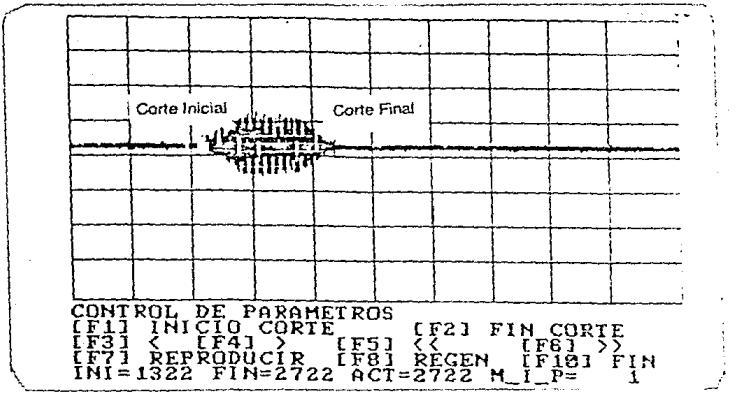
COMO ES QUE SE HACE UN CORTE DE ANALISIS DE LA SEÑAL ?

Primero con F1 se selecciona la zona donde se va a empezar el corte de análisis, pero pueden utilizarse primero las funciones F3, F4, F5 o F6, para moverse a alguna parte intermedia de la señal y después seleccionar F1. Después utilizar cualquiera de las funciones F3, F4, F5 o F6 y ya una vez satisfecho con la región de análisis, oprimir F2 para indicar el fin del corte.



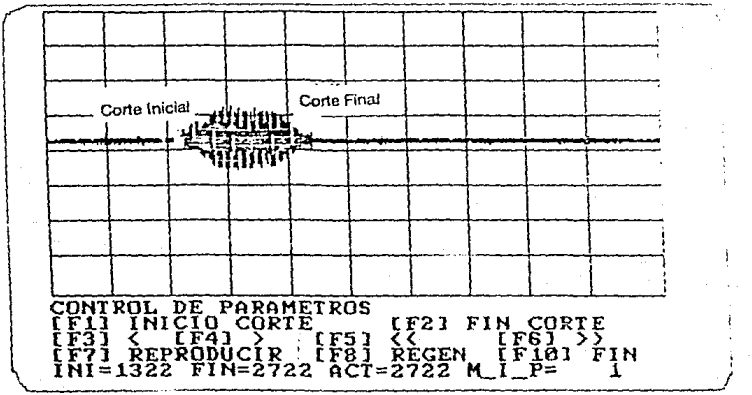
PANTALLA # 22

Muestra el **Corte inicial** y el **Corte final** que se le hizo a la señal graficada. Para regenerar las zonas marcadas por los cortes de la señal se oprime la tecla F8.



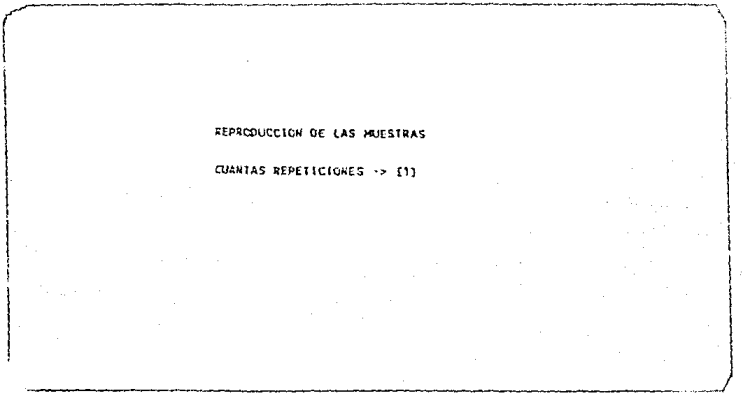
PANTALLA # 23

El instructor podrá observar como el segmento de curva recortado ahora está contínuo. Ahora si se oprime F7, se reproduce la señal, desplegando una pantalla que contiene la gráfica de la señal capturada con el corte de análisis y la gráfica de la señal descompactada con su corte de análisis, se puede observar que existe mucha similitud entre las dos señales. Si no se oprime F7 y en cambio se opirme F10, se despliega un mensaje que dice : "**PROCESO INCOMPLETO.....**".Ya que el proceso no puede terminarse sin antes hacer el corte de análisis y realizar la reproducción.



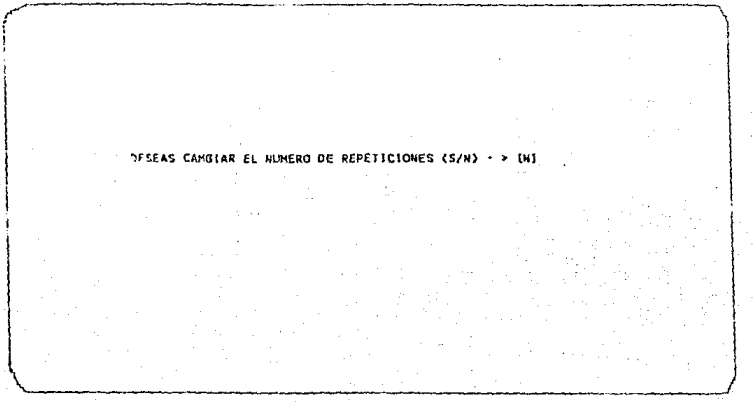
PANTALLA # 23.1

Es la misma pantalla de control de parámetros de corte, que se presenta cuando no se completó el proceso y se desea reproducir la señal para completarlo.



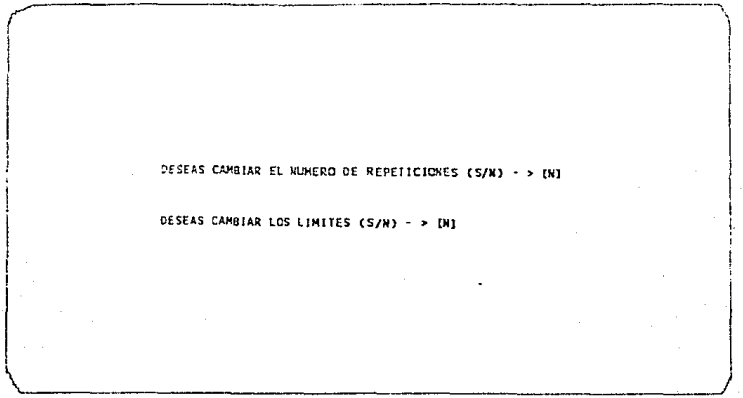
PANTALLA # 24

Pantalla de reproducción de las muestras. Pregunta por el número de repeticiones, es decir, el número de veces que se va a repetir el fonema, para que se oiga continuo y natural. El número de repeticiones por *default* es [1]. Pero se puede cambiar ese número, tecleando el valor del número de repeticiones que se desea.



PANTALLA # 25

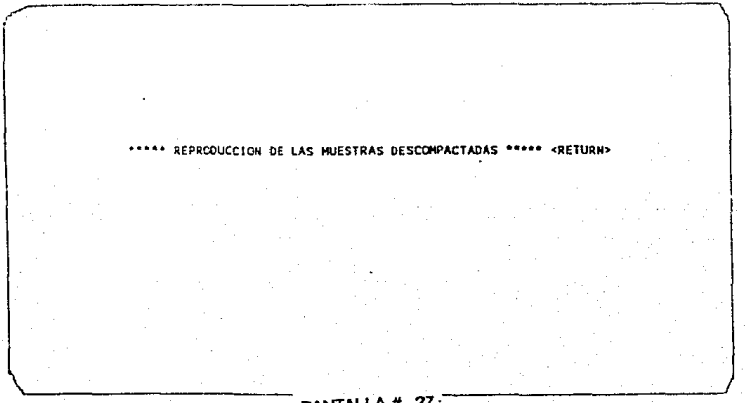
En esta pantalla se puede asumir el valor de *default* **[N]**, oprimiendo la tecla de *<return>* o poner [S] y *<return>*. Si se contesta afirmativamente, presenta la pantalla de número de repeticiones, para dar el número de repeticiones.



PANTALLA # 26

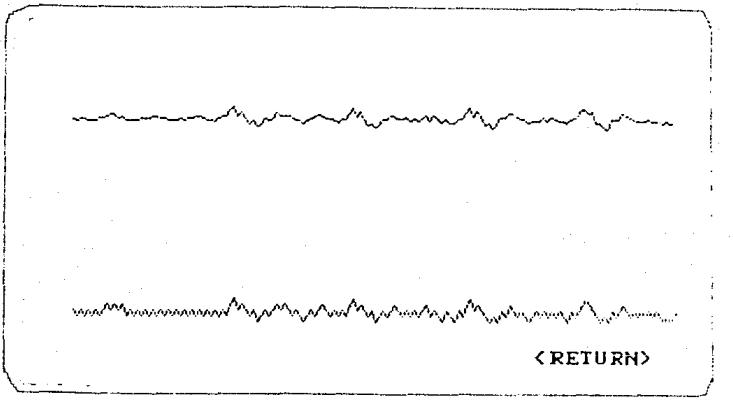
Pantalla que pregunta por el cambio de límites de la señal a reproducir. El valor de *default* es **[N]** y se obtiene oprimiendo *<return>*.

Si se contesta afirmativamente nos presenta una pantalla que nos pregunta por los límites nuevos, límite inferior de la señal y límite superior de la señal.



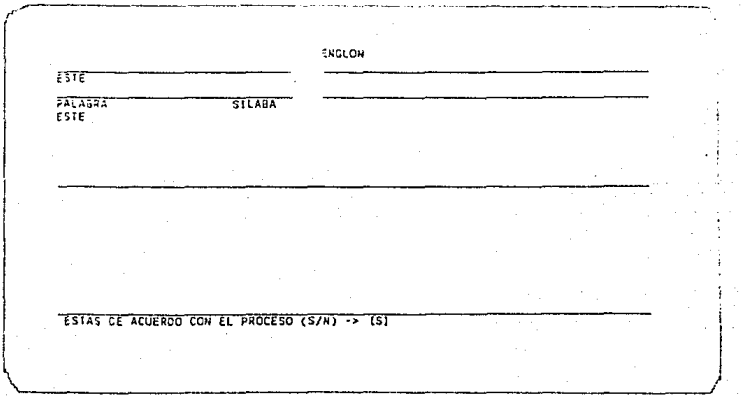
PANTALLA # 27

Pantalla que muestra un mensaje que nos dice que el programa está preparado para reproducir las muestras descompactadas.



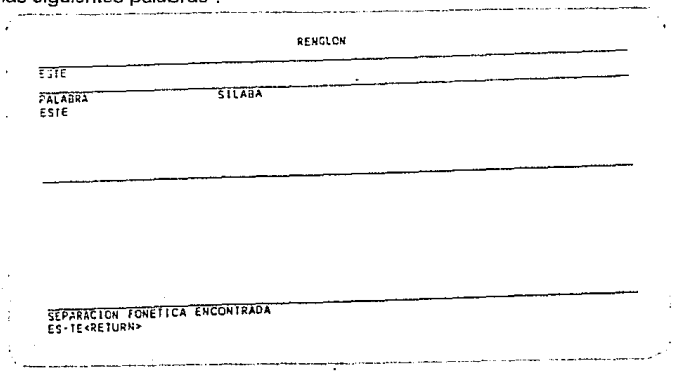
PANTALLA # 28

Pantalla que muestra en la parte superior la gráfica de la señal original capturada y recortada y en la parte inferior la gráfica de la señal descompactada y recortada, lista para ser reproducida en forma de voz.



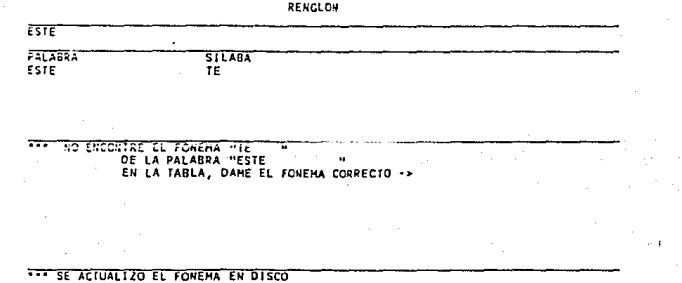
PANTALLA # 29

Pantalla que nos presenta el renglón a analizar y la palabra analizada y reproducida. Se le pregunta al instructor si esta de acuerdo con el proceso, si la respuesta es negativa se vuelve a capturar el fonema y repetir todo el proceso, es decir, pasa a la Pantalla de Captura del Fonema, luego a la Pantalla de Graficación de la Señal del Fonema, después a los menús complementarios de Control de Parámetros de la Señal, hasta Recortar y Reproducir la Señal que se capturo. Si la respuesta es afirmativa se continúa con el proceso de análisis de las siguientes palabras .



PANTALLA # 30

Al oprimir *<return>* se reproduce la palabra analizada sílaba por sílaba, grabada como fonemas asociados.

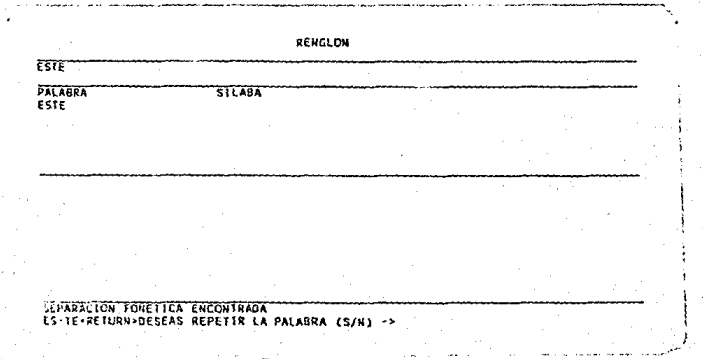


PANTALLA # 31, 32

Si al analizar la palabra en forma silábica, una de la sílabas coincide con alguna sílaba almacenada anteriormente en el archivo de fonemas y de tablas, entonces el programa nos pregunta si la nueva regla de la sílaba a analizar, es decir, el siguiente caracter después de la sílaba, rige esa separacion o no. En caso de que esa regla no corresponda a esa separación silábica, el usuario tendrá que teclear la separación silábica correcta. En caso de que si corresponda, entonces se almacena una nueva regla en el archivo de tablas, para ese fonema.

PANTALLA # 32

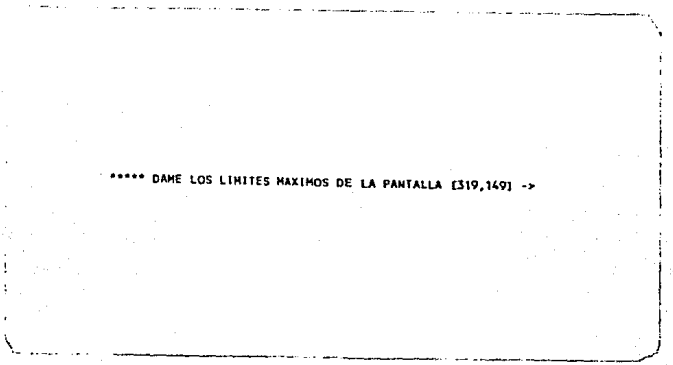
Pantalla de Actualización de la regla gramatical. Es decir, se agrego una nueva regla que rige esa separación silábica.



PANTALLA # 33

Despliega el mensaje de que esa separación silábica asociada a un grupo fonético se ha localizado con su regla correspondiente. En este momento se leen de disco las muestras correspondientes al fonema, se descompactan y se concatenan los fonemas grabados en disco para reproducir la palabra asociada a los fonemas localizados y concatenados.

Cuando el instructor esta en la Pantalla de Control de Parámetros Principal y selecciona F5 " LIM ", se traslada a la siguiente pantalla.



Contiene el mensaje que nos pide los límites máximos de la pantalla.

Los límites por *default* son **[319,149]**. Para dar los límites hay que teclear < límite en X > " " < límite en Y >, o < *return* > si se desean los límites dados por *default*.

Si el instructor oprime F6 ' INI ', pasara a la pantalla donde se pide al instructor cambiar los parámetros actuales por los originales.

Si la respuesta es afirmativa, cambian los parámetros actuales por los iniciales. Si la respuesta es negativa, no hay ninguna alteración y el instructor podra continuar con el proceso.

```
***** DESEAS CAMBIAR LOS PARAMETROS ACTUALES POR LOS ORIGINALES (S/N) ->
```

PANTALLA # 35

Pantalla de Cambio de Parámetros Actuales por los Originales (*).

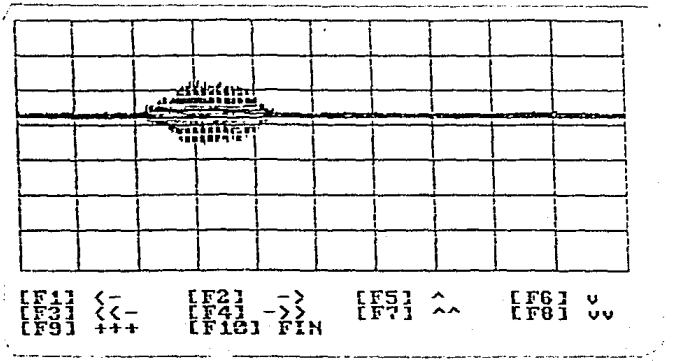
Si seleccionamos **F7 " DESP "**, el instructor pasa a la **Pantalla de Cambio de Parámetros de Desplazamiento**.

```
***** CAMBIO DE LOS PARAMETROS DE DESPLAZAMIENTO *****

 DESPLAZAMIENTO MINIMO EN X -> [100]
100
 DESPLAZAMIENTO MAXIMO EN X -> [1000]
1000
 DESPLAZAMIENTO MINIMO EN Y -> [1]
1
 DESPLAZAMIENTO MAXIMO EN Y -> [10]
```

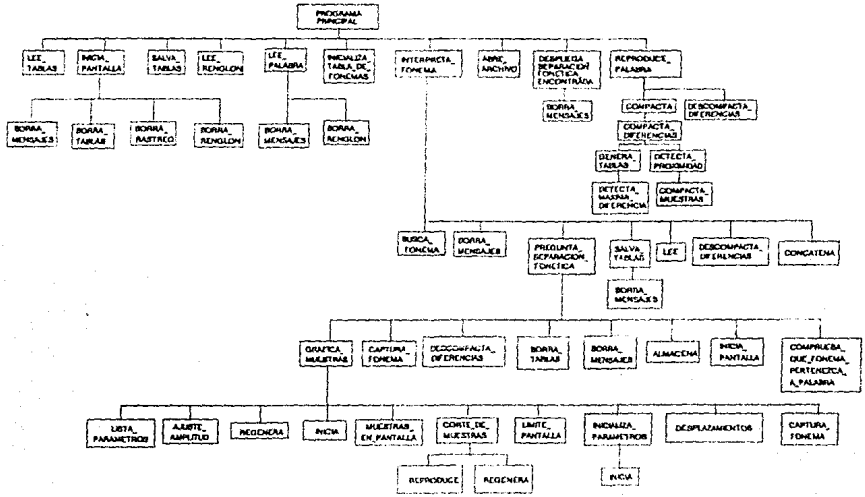
PANTALLA # 36

Pantalla de cambio de parámetros de desplazamiento. El *desplazamiento mínimo* sobre el eje X, por *default* es **[10]**, pero puede modificarse, siempre y cuando el número que se dé sea entero. Después muestra el *desplazamiento máximo* sobre el eje X, por *default* es **[100]**, si se desea modificar solo hay que teclear otro número diferente a 100 y mayor al *desplazamiento mínimo* sobre X. Y después muestra los desplazamientos mínimo y máximo en Y, que por default son **[10]** y **[100]**, respectivamente, los cuales pueden ser modificados también. Si no se desea hacer modificaciones, basta con oprimir *<return>* en los mensajes que no se desea modificar. Si se selecciona F10 " FIN ", se regresa a la ***Pantalla Principal de Graficación***.

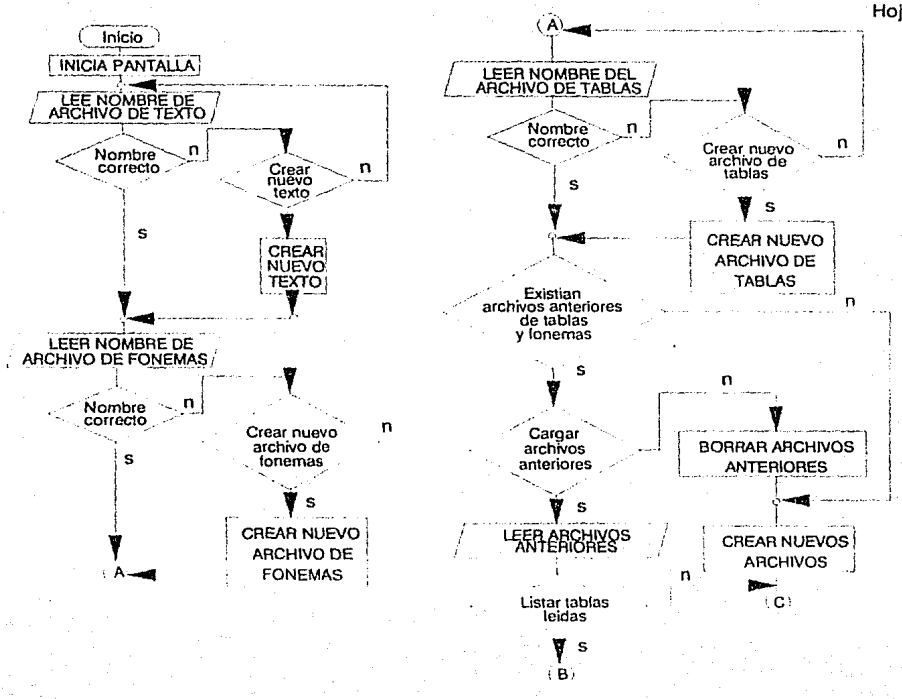


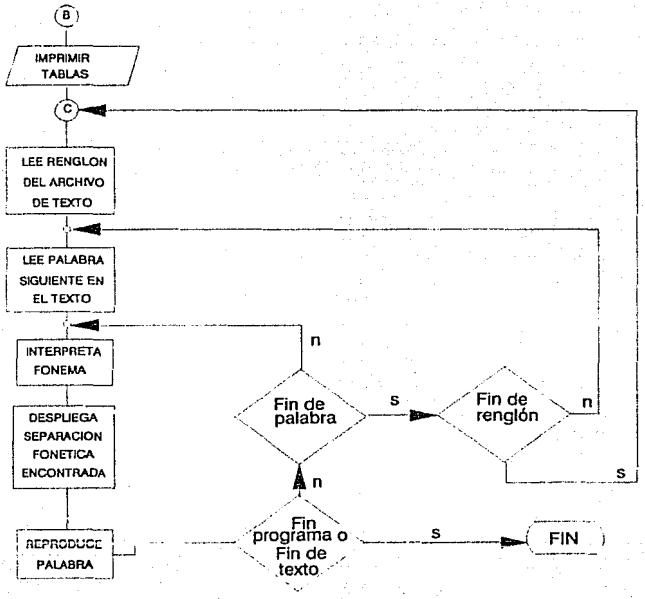
PANTALLA # 36.a

DIAGRAMA DE BLOQUES GENERAL DEL SISTEMA FONETICS



DIAGRAMAS DE FLUJO


Inicio
INICIA PANTALLA
LEE NOMBRE DE ARCHIVO DE TEXTO
Nombre correcto
n
s
Crear nuevo texto
n
CREAR NUEVO TEXTO
LEER NOMBRE DE ARCHIVO DE FONEMAS
Nombre correcto
n
s
Crear nuevo archivo de fonemas
n
s
CREAR NUEVO ARCHIVO DE FONEMAS
A
A
LEER NOMBRE DEL ARCHIVO DE TABLAS
Nombre correcto
n
s
Crear nuevo archivo de tablas
n
s
CREAR NUEVO ARCHIVO DE TABLAS
n
Existian archivos anteriores de tablas y fonemas
s
n
Cargar archivos anteriores
BORRAR ARCHIVOS ANTERIORES
s
LEER ARCHIVOS ANTERIORES
CREAR NUEVOS ARCHIVOS
n
C
Listar tablas leidas
s
B


B
IMPRIMIR TABLAS
C
LEE RENGLON DEL ARCHIVO DE TEXTO
LEE PALABRA SIGUIENTE EN EL TEXTO
INTERPRETA FONEMA
DESPLIEGA SEPARACION FONETICA ENCONTRADA
REPRODUCE PALABRA
Fin de palabra
n
s
Fin de renglón
n
s
n
Fin programa o Fin de texto
s
FIN

CAPITULO VIII

VIII. CONCLUSIONES

El sistema de Adquisición y Reproducción de voz, **FONETICS**, realizado en el presente proyecto, tiene la ventaja de ser un sistema con un objetivo muy general, es decir, es un prototipo aplicable a diversos enfoques en Análisis de Texto y en Análisis de la Señal de Voz. Además puede ser optimado para que en un futuro sea aplicable a : Reconocimiento de Voz para Sistemas Interactivos *Hombre-Máquina*, Análisis Espectral de la Voz, Identificación de Patrones de Voz específicos para Sistemas de Seguridad, Reconocimiento de Figuras, Detección de Acentos Prosódicos y Ortográficos, etc.

El sistema permite que la computadora "*aprenda*" *fonemas* y sus reglas correspondientes. Es una herramienta útil para la creación de una extensa *Biblioteca Fonética* en cualquier idioma ya que el sistema utilizado posee una alta flexibilidad y su única limitante es el espacio disponible de memoria en disco y el tiempo de procesamiento.

Se incluyó, además, un Análisis Fonético del idioma Español que será de gran utilidad para las personas que deseen optimar procesos de Síntesis o Generación de voz específicos para el idioma Español.

A través de este proyecto se obtuvo una buena experiencia en manejo de puertos, direcciones físicas, además de la sincronización con la transmisión y la recepción de la señal digital a procesar.

La Síntesis y el Análisis de la Señal de Voz tienen varias aplicaciones, desde la lectura de un texto hasta el Reconocimiento de Patrones de Voz, como también en relojes, sistemas de control automático, telefonía, asistencia en rehabilitación acústica, en aprendizaje infantil, servicio de control a tripulaciones aéreas, navales y espaciales, Composición Musical y Lectura por Notas, etc.

Todo esto con el fin de poder hacer después un reconocimiento de sonidos, desarrollando más la Inteligencia Artificial que se le ha aplicado.

El desarrollo de este proyecto se dividió en varias etapas:

a) La etapa de Investigación Bibliográfica: En donde se investigaron temas como Técnicas de Simulación del Tracto Vocal, así como Algoritmos de Síntesis Paramétrica, mediante un Análisis de la Señal de Voz. Otros temas de investigación fueron aquellos referentes a los diversos proyectos relativos al desarrollo en Reconocimiento de Voz, Generación y Síntesis.

Lo cual nos dio un panorama general para buscar alternativas en el desarrollo del proyecto y realizarlo con miras al diseño e implementación de varios prototipos con un sinnúmero de aplicaciones en procesamiento de señales.

b) Etapa de Investigación Técnica: Era importante definir cómo se iba a implementar el proyecto **FONETICS**. Para esto se consultaron manuales de arquitectura de computadoras, de microprocesadores y memorias. Primero, se investigaron las direcciones físicas y lógicas de los puertos de comunicación y de memoria, ya que se definió la importancia de la transmisión y recepción de datos digitalizados se concluyó que era conveniente manejar direcciones a puertos de Entrada / Salida, en vez de manejar direcciones a Interrupciones **IRQ's**, ya que el proyecto no requería interrumpir algunos de los dispositivos de Entrada / Salida, de Almacenamiento y demás periféricos, sólo reproducir y grabar voz. Segundo, se diseñaron los filtros Paso Banda con un ancho de banda de 200 **Hz** a 3200 **Hz**. Esta etapa requirió de sumo cuidado al escoger los dispositivos y los distintos arreglos de resistencias y capacitores al hacerse los cálculos que se establecen en el capítulo VI. Una vez calculado esto y probado en una tarjeta "***Protoboard***" (*Experimental*) se avanzó a la siguiente etapa. El arreglo de filtros fue de Cuarto Orden, utilizando combinaciones de Filtros Paso Altas con Filtros Paso Bajas para definir las Frecuencias de Corte que delimitarían el Ancho de Banda adecuado para el buen funcionamiento del Sistema.

c) Etapa de Construcción de la Tarjeta Interfaz de Generación de Voz: La construcción de una tarjeta conectada a un "**slot**" de una **PC**, como una extensión de las aplicaciones que puede tener una computadora, consiste en el diseño de circuitos digitales y analógicos, generación de pistas y nodos de interconexión de componentes, mediante procesos químicos realizados sobre una tarjeta de cobre, pasando por una etapa intermedia en donde las pistas están simuladas por finísimos alambres blindados llamados "***wire wrap***". Y también el proceso de soldadura de componentes y terminaciones de "***wire wrap***". Para revisar el buen funcionamiento de la tarjeta, en sus diferentes etapas, se utilizaron instrumentos de medición como: Osciloscopio y Multímetro, en donde se hace un Análisis de la Forma de Onda, de manera visual, se mide la Cantidad de Ruido, el Factor de Rizo, Niveles de Saturación, Frecuencia de Muestreo entre otras cosas. La tarjeta prototipo que se utilizó genera cierto *ruido* en la transmisión de la señal, debido a que todavía no está en su última etapa de acabado, es decir, no se han generado pistas ni se ha blindado la tarjeta a través de procesos químicos. Pero a pesar de esto, se considera que el sistema reproduce de manera adecuada la señal de voz.

d) Etapa de Implementación del Software del Sistema FONETICS: Ya una vez investigadas las direcciones físicas de los puertos de Entrada y Salida de Información, para configurar lógicamente el sistema de comunicación de puertos, se procedió a desarrollar las rutinas de Captura y Reproducción, de manera aislada. Al hacerse las pruebas, se ajustaron parámetros como la frecuencia de muestreo a 7 **KHz**.

Las rutinas se desarrollaron en **TurboPascal V.3.0**, actualmente se podría implementar en **Lenguaje C**, o **TurboPascal V.5.0**.

Una etapa bastante importante fue el desarrollo de las Rutinas de Compactación, Descompactación, Generación de Pantallas de Comunicación y de Gráficas de la Señal Capturada *vs*. la Señal Descompactada y Reproducida. Debido a la cantidad de rutinas y complejidad del proyecto, se tuvieron problemas en memoria en disco, tanto para la codificación del programa, como para el almacenamiento de fonemas. Se concluyó que al hacerse la grabación de fonemas en Disco Duro, el tiempo de respuesta, al accesarse cada fonema, disminuye considerablemente, que en el caso de que se hiciera el acceso a disco flexible. Otra posibilidad de aumentar la velocidad de acceso de fonemas pregrabados existe al utilizarse **RAMDISK**, pero con la desventaja de que al apagar la computadora desaparecerían de la memoria aquellos fonemas que anteriormente se habían grabado en memoria **RAM**. Otra consideración que se debe de tomar en cuenta es, que la frecuencia de muestreo sincronice con la frecuencia que manejan el **DAC** y el **ADC**, para evitar todo *ruido* generado en las diferentes rutinas.

Ambas etapas, Construcción de Tarjeta e Implementación de Rutinas, se relacionan intrínsecamente, ya que existe una comunicación entre puertos de Entrada y Salida y direcciones de memoria. Por eso, en estos casos, van muy ligadas una con la otra.

Todo proyecto relacionado con Emulaciones o Comunicación de Puertos y Control de Interrupciones, implica un conocimiento profundo de la arquitectura de la computadora, de las direcciones disponibles en memoria, de las direcciones físicas de los puertos y de las interrupciones que se habilitan.

Nuestro proyecto está, basicamente, enfocado a la Generación de Voz, con una calidad aceptable de señal reproducida, y al almacenamiento de las unidades mínimas del lenguaje hablado, llamadas "***fonemas***". Este segundo punto es crucial en el desarrollo del proyecto, ya que sin la utilización de técnicas de Compactación, no hubiera sido posible almacenar palabras y frases en memoria.

El procedimiento que se utilizó para compactar fonemas, almacenados en memoria, fue adaptado a la aplicación del proyecto, utilizándose una codificación de forma de onda con una modulación **PCM** y un Error de Aproximación, que no tiene que ver con el Error de Predicción que se utiliza en los métodos de Predicción Lineal.

Lo interesante de este proyecto es que la computadora aprende una serie de fonemas, partiendo de la situación de que en un principio la memoria de la computadora no estaba alimentada de reglas y fonemas. Por eso, este proyecto es muy aplicable a Detección de Problemas de Aprendizaje y de Lenguaje. Además puede aplicarse como herramienta de trabajo para personas que sufren de ceguera.

El proyecto es aún un prototipo de laboratorio, no es algo totalmente consumado, sujeto a modificaciones y a mejoras, ya que su objetivo es de lo más general.

Se incluyó además una extensa bibliografía con la cual se puede profundizar con más detalle en cualquier aspecto que el lector considere de especial interés.

México, D.F., a 16 de Octubre de 1989.

Ciudad Universitaria

Por mi raza, hablará el espíritu

APENDICE A

APENDICE A

A.1 INTRODUCCION A LA ARQUITECTURA DE LA PC

En este apéndice se examinará la estructura interna de las computadoras tipo **PC** compatibles con **IBM**, las cuales constan principalmente de un microprocesador **8086/8088** y el **8087** que es un coprocesador opcional, así como de otros circuitos integrados especializados en funciones de Entrada/Salida, de Control, de Comunicaciones, etc.

A.2 CARACTERISTICAS DE LA PC

Su procesador es el **8086/8088**, procesa 16 **bits** a 4.77 **MHz.** Maneja más rápidamente palabras de mayor longitud, aumentando significativamente la velocidad global de procesamiento, en un factor cercano a 10 **MHz.** Mejora esta posibilidad a través de un co-procesador aritmético que trabaja en paralelo con el procesador central. Entre las facilidades directamente atribuibles a un mejor sistema operativo cabe mencionar la simplicidad del manejo de las "***ventanas***" (*windows*), y de las gráficas de la pantalla del monitor. También la posibilidad de manejar 80 columnas de caracteres en pantalla, en lugar de 40, mejorando las posibilidades de despliegue de información. Trae integrados puertos serie y paralelo sin la necesidad de incorporar tarjetas suplementarias para este fin. Esto, obviamente abarata y simplifica la comunicación con el exterior. En cuanto a costo, existe una tendencia a la baja, en general en toda la familia de computadoras compatibles con **IBM-PC.**

A.3 INTRODUCCION AL MICROPROCESADOR 8086/8088 (16 BITS)Y SUS CO-PROCESADORES

Los microprocesadores de 16 **bits 8086** y **8088** son una extensión del **8080.** Los dos, **8086** y **8088**, utilizan el concepto de "***colas***" de instrucciones para aumentar la velocidad de proceso, donde guardan los bytes de la instrucción.

El **8086** accesa hasta 1 **MB**, utilizando un esquema de direccionamiento de memoria llamado *segmentación*. Parte de la dirección y todo el bus de datos están multiplexados en 16 terminales (patas). Los cuatro **bits** de dirección restantes coinciden con las otras cuatro terminales o pines de dirección adicionales que también utilizan para el estado. Se requiere un reloj externo y un controlador de bus de direcciones/datos.

El **8088** tiene una estructura de interupciones muy potente, se necesita 1 **MB** para accesar los 256 vectores de interrupciones.

El **8086** realiza operaciones de **Entrada/Salida** en un espacio de 64 **KB** de longitud. Existen programas traductores especiales para pasar de **8080** a **8086**. El **8087**, es un co-procesador paralelo numérico que maneja datos, controla el flujo de instrucciones del **8086** localizando sus propias instrucciones y ejecutándolas sin ayuda del **8086**. El segundo co-procesador es el **8089**, que es un chip diseñado para un funcionamiento eficiente de los movimientos del bloque de datos, tiene dos canales y puede multiplexar entradas y salidas fácilmente. También se conecta con el bus local y tiene su propio juego de instrucciones. Estos chips, junto con el **8086**, son una sólida base para el diseño de computadoras realmente potentes.

A.4 PROCESADOR 8088

Está diseñado con una arquirtectura de 16 **bits** internamente, y 8 **bits** para el bus de datos externamente, hacia la memoria y entrada/salida. El Circuito Integrado consta de 40 patas y opera a un nivel de voltaje de 5 V.

Contiene 8 registros generales de 16 **bits** cada uno y, además estos se subdividen en *registros altos/bajos* y se accesan en forma separada. Esto significa que un registro de datos se maneja internamente como un registro de 16 **bits** o como 2 registros de 8 **bits**. El **8088** utiliza una arquitectura de "***pipeline***", consistiendo en 2 unidades:

a) Unidad de interfase

El **BIU** (*Bus Interface Unit*) hace un "***fetch***" sobre las instrucciones adicionales.

b) Unidad de ejecución

El **EU** (***Execution Unit***) ejecuta previamente aquellas instrucciones adicionales que pasaron por la etapa de "***fetch***". Se crea por lo tanto un "***pipeline***" o una "***cola***" entre las dos unidades.

A.5 CONTROLADORES DEL SISTEMA

1.- Controlador programable de **DMA 8237**.

2.- Controlador programable de interrupciones **8259**.

3.- Controlador de Bus **8288**.

A.5.1. CONTROLADOR DMA 8237 (DIRECT MEMORY ACCESS)

Se encuentra localizado en la tarjeta del sistema, permitiendo altas velocidades de transferencia del programa y bytes de datos entre la memoria y los dispositivos de **Entrada/Salida**. Este controlador tiene cuatro canales independientes permitiendo la transferencia de cuatro bloques, concurrentemente. El **canal 0** es usado para ejecutar el *refrescamiento* de la **RAM dinámica**.

Los **canales 1 y 3** están disponibles. El **canal 2** es utilizado para transferencias de información a **Floppys** mediante el **DMA**. Los **canales 1 y 3** también se pueden usar para expander buses mediante una tarjeta.

A.5.2 CONTROLADOR DE INTERRUPCIONES PROGRAMABLE 8259 A

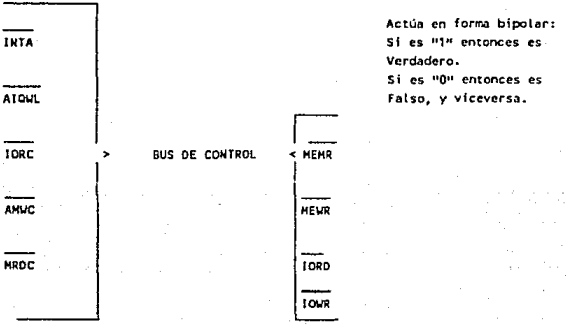
Consta de 28 patas y puede direccionar hasta **ocho vectores** de **interrupción**, en orden de prioridad, hacia el **CPU**. Está diseñado para minimizar el *Software* y el *Overhead* en *tiempo real* al accesar interupciones de diferentes niveles de prioridad. Hace llamadas desde el equipo periférico determinando cual de las llamadas a interrupción tiene la más alta prioridad y acciona una interrupción hacia el **CPU**. Cada dispositivo periférico llama a una "rutina" que está asociada a sus requerimientos de operación. Este chip selecciona el nivel de modos de interrupción, pero puede ser reconfigurado, para que realice ciertos tipos de interrupción, u otros, durante la operación del sistema, aunque no es muy recomendable.

La lista de las llamadas a los vectores de interrupción (**IRQ**) conectadas a dispositivos periféricos es:

IRQ0 SYSTEM TICK (*reloj del sistema*).
IRQ1 KEYBOARD INTERFACE (*teclado*).
IRQ2 Available (*disponible*).
IRQ3 COMMUNICATION INTERFACE, COM2. (*interfase de comunicación serie*).
IRQ4 COMMUNICATION INTERFACE, COM1. (*interfase de comunicación serie*).
IRQ5 Available (*disco duro, en las nuevas* **PC's**).
IRQ6 5 1/4 INCH DISKETTE DRIVE INTERFACE. (*drive del diskette*).
IRQ7 PRINTER INTERFACE (*impresora*).

A.5.3 CONTROLADOR DE BUS 8288

Se utiliza cuando el **8088** se configura para usarse en modo máximo, y genera las siguientes señales:



Tan pronto como el **DMA** toma control sobre el Bus, las salidas del **8288** se van a **TRI-STATE**. El **pin 6 (AEN)** se conecta a **AENBRD** de la lógica de estado de espera para garantizar una correcta transferencia de datos con los dispositivos de **Entrada/Salida** ó a memoria deseada.

A.6 CONTROLADOR DE COMUNICACIONES INS 8250

Elemento de comunicación asíncrona **ACE**. Funciona como interfase de **Entrada/Salida** de datos en serie. Este dispositivo realiza la conversión de datos serie a datos paralelo; en datos o caracteres recibidos desde un dispositivo periférico o desde un modem, y viceversa, en datos y caracteres recibidos desde el **CPU**. La información del **status**, reporta el tipo y las condiciones de la transferencia de operaciones que se están ejecutando por el **8250**, así como cualquier condición de error (*paridad, over-run, framing, break interrup*). Se conecta con el dispositivo en serie con el **DB_25s**. que es un conector de 25 patas.

A.6.1 REGISTRO DE IDENTIFICACION DE LA INTERRUPCION DEL 8250

El **8250** tiene interrupciones por prioridad en cuatro niveles:

1.- Receiver Line Status.

2.- Received Data Ready.

3.- Transmitter Holding Register.

4.- Modem Status.

El tipo de interrupción por prioridad espera a que se almacene en el registro de identificación de interrupción. Cuando se habilita el *chip-select* (**CS**) y durante su habilitación se direcciona el registro de identificación de la interrupción, éste escoge la interrupción de más alta prioridad hasta que ésta entre a las rutinas del **CPU** y pueda el ***Registro de Identificación de Interrupción*** reconocer otra interrupción. El **IIR** consta de **8 bits**: El **bit 0** se utiliza para prioridades de interrupción por *Hardware* y además indica cuando una interrupción está en espera. Cuando vale **0 lógico** significa que la interrupción está en espera, y cuando vale **1 lógico** está en estado de **Polling** y la interrupción no está en espera.

Los **bits** 1 y 2 son utilizados para identificar la interrupción en espera con la prioridad más alta.

Del **bit 3** al **bit 7** permanecen en **0 lógico**.

A.6.2 REGISTRO DE HABILITACION DE LA INTERRUPCION DEL 8250

Sus 8 **bits** son utilizados para habilitar los cuatro tipos de interrupción del **8250** para esperar y activar el chip Interrupt y la señal de salida de interrupción. Es posible deshabilitar el sistema de interrupción para resetear los **bits** 0 al 3 del registro de habilitación de la interrupción. Todos se habilitan con **1 lógico**.

Bit 0 *Received Data Available Interrupt.*
Bit 1 *Transmitter Holding Register Empty Interrupt.*
Bit 2 *Receiver Line Status Interrupt.*
Bit 3 *Modem Status.*

Los demás bits siempre están en ***ceros***.

A.7 MANEJADORES DE BUS Y ROM's

Existen dos tipos de buffers:

1.- Buffer de Direcciones **74LS373**.

2.- Buffer de Datos **74LS245**.

A.7.1 BUFFER DE DIRECCIONES 74LS373

Es un latch tipo "**D**" que selecciona cualquiera de las ocho líneas de direcciones externas (**A0 - A7**) del **8088**.

A.7.2 BUFFER DE DATOS 74LS245

Es un Bus Trasnceptor Octal con salidas **TRI-STATE (D0 - D7)** que es multiplexado con los ocho primeros bits del ***Bus de Direcciones***.

A.8 INTERFASE DEL SISTEMA DE ENTRADA/SALIDA 8259

Las siguientes señales del **8259** son mandadas al controlador de disco duro desde el bus de **Entrada/Salida**.

SEÑAL DEL SISTEMA	NOMBRE DEL PIN	FUNCION
A19-A0	**A12-A31**	Veinte líneas de dirección.

SEÑAL DEL SISTEMA	NOMBRE DEL PIN	FUNCION
D7-D0	**A2-A9**	Bus de 8 **bits** de datos,transmite datos e información de status entre el Host y Controlador.
IR	**B14**	Se habilita en señal baja por el Host cuando se realiza una lectura en el puerto de entrada/salida.
IW	**B13**	Se habilita en señal baja por el Host cuando se realiza una escritura en el puerto de entrada/salida.
AEN	**A11**	Se habilita en positivo por el Host cuando el **DMA** tiene el control del Bus.
Reset	**B2**	Se habilita positivamente cuando se forza a condiciones iniciales de encendido.
IRQ2-IRQ7	**B4,B25-B21**	Se habilitan positivamente y permiten al controlador interrumpir al **CPU** en la tarjeta. La prioridad va de **IRQ2**, la más alta, a **IRQ7**, la más baja.
DRQ1,DRQ2,DRQ3	**B18,B6,B16**	Se habilitan positivamente. Cuando los datos están disponibles para ser transferidos hacia o desde el controlador, se accesa la señal de llamada al **DMA** por controlador. Los tres canales disponibles van en prioridad desde el

A.9 ORGANIZACION FISICA DE LA MEMORIA

En los procesadores **8086/8088**, una dirección física de memoria se guarda como dos cantidades de 16 **bits**. Ambas cantidades, se combinan en una forma especial formando una "*dirección real*" de 20 **bits**. Un apuntador es una palabra doble que almacena esas dos cantidades, el número de segmento y el desplazamiento.

Para un chip de memoria, las líneas de señal forman un bus, con los habituales sub-buses de alimentación, control, dirección y datos. El sub-bus de datos puede tener un solo **bit** habilitado, "*4 binario*" u "*8 binario*", según sea el caso. El sub-bus de direcciones tiene de 10 a 16 líneas de señal.

A.10 CORRESPONDENCIA ENTRE EL ESPACIO DE DIRECCIONES Y LA TARJETA DE MEMORIA REAL

Normalmente hay interruptores en la tarjeta que permiten proyectar, o "*poner en correspondencia*" la memoria de la tarjeta con una parte del espacio de direcciones de la computadora. Puede haber espacios prohibidos. Si a todas las posibles direcciones no se les asigna memoria real, cuando se intente leer alguna de estas direcciones no asignadas, a la computadora, devolverá "todos dígitos ceros", según el convenio tomado.

Puede suceder que el sistema tenga más memoria real de la que puede acomodar su rango de direcciones. Es el caso de los microprocesadores **8080** y **8085**, o el Z80, que pueden direccionar 64 **KB** y se consigue dividiendo la memoria en "***páginas***", cada una de las cuales si tiene una correspondencia directa con la memoria real. Con un conjunto adicional de líneas de direccionamiento se selecciona una de las posibles páginas cada vez.

Este esquema de direccionamiento recibe el nombre de "***paginación por hardware***". Ocurre a veces que el enviar la dirección de la página resulta bastante más lento que enviar una dirección normal. Un esquema de *paginación* frecuentemente utilizado en las computadoras de 8 **bits** consiste en enviar la información de selección de la página a través de un puerto de **Entrada/Salida** a las placas de memoria del sistema. En cada placa, una parte de la lógica se encarga de codificar dicha información y activar o desactivar los chips de memoria de acuerdo con ella.

De esta manera , los procesadores de 8 **bits** pueden utilizar memorias mayores que los 64 **KB** adicionales. Otro método es la "***segmentación***", el cual es un método de acceso a memoria en el que toda dirección se compone de dos cantidades: Un "***identificador de segmento***" y un "***desplazamiento***". El identificador de segmento apunta a un área general de memoria, mientras que el desplazamiento apunta a una dirección dentro del mismo segmento.

El microprocesador **8086/8088** tiene 4 registros especiales, llamados "***registros de segmentación***":(**CS**) ***Code Segment***, (**DS**) ***Data Segment***, (**ES**) ***Extra Segment***, (estos dos últimos son usados para datos); y para almacenamiento temporal el (**SS**) ***Stack Segment***.

Los contenidos de estos registros se multiplican por 16 y se suman a la información sobre la dirección proveniente del **CPU**, es decir, el desplazamiento, para calcular las direcciones de memoria reales. Con el **8086** y **8088**, los segmentos pueden comenzar en cualquier frontera de 16 **bits**, y acabar en cualquier posición, hasta 64 **KB** más adelante. Las versiones futuras del **8086/ 8088**, el **IAPX 286**, permitirán que los segmentos comiencen en cualquier posición de memoria. Hay instrucciones especiales que cargan la información sobre los identificadores de segmento en los registros de *segmentación*. El **8086** y **8088** son microprocesadores de 16 **bits** de propósito general de **Intel** son idénticos, excepto en el tamaño de su bus de datos externo y están siendo fabricados por una gran número de empresas japonesas, incluyendo a **Fujitsu**.

APENDICE B

DICCIONARIO DE DATOS

APENDICE B

DICCIONARIO DE DATOS DEL PROGRAMA DEL SISTEMA "FONETICS"

El programa **FONETICS** es un programa Generador de Voz por Técnicas de Compactación de Fonemas. Lo que se almacena en la Memoria son los Fonemas. Fonema es la Unidad más simple del Lenguaje Hablado. En este programa se almacenan grupos fonéticos correspondientes a las sílabas.

Este programa consiste de ocho rutinas principales:

1) ***Captura.***

2) ***Compactación.***

3) ***Lectura.***

4) ***Escritura.***

5) ***Descompactación.***

6) ***Reproducción.***

7) ***Análisis Silábico Fonético de un Texto Escrito.***

8) ***Graficación de la Señal del Fonema.***

B.1. DEFINICION DE LAS VARIABLES DEL PROGRAMA PRINCIPAL

TIPOS DE VARIABLES

ST6 Es un arreglo de tipo STRING de 6 caracteres.
ST14 Es un arreglo de tipo STRING de 14 caracteres.
ST80 Es un arreglo de tipo STRING de 80 caracteres.

VARIABLES

LONG_BLOQUE : **INTEGER** Es la longitud del bloque a almacenar en disco en el Archivo de Fonemas.

APUNT : **INTEGER** Es un apuntador de Propósito General.

NOMBRE_TABLAS : **ST14** Nombre del Archivo de Tablas.

NOMBRE_ARCHIVO_FONEMAS : **ST14** Nombre del Archivo de Fonemas.

NOMBRE : **ST14** Variable con nombre de Archivos de Propósito General.

TABLAS_ANTERIORES : **STRING[1]** Variable usada para preguntar por el uso de Tablas anteriores.

ALFA : **STRING[1]** Variable Alfa-Numérica de Propósito General.

BORRAR_ARCHIVO : **STRING[1]** Variable usada para preguntar si se desea borrar el archivo.

TABLA_FONEMAS : **ARRAY[1..1000] OF TABFON** Arreglo que contiene hasta 1000 fonemas con sus atributos.

FONEMA : **ST6** Cadena de seis caracteres.

BLOQUE_INICIAL : **INTEGER** Apuntador al bloque inicial.

TOTAL_DE_FONEMAS : **INTEGER** Número total de fonemas a grabar.

REPETICIONES : **INTEGER** Número de repeticiones de un segmento de fonema.

FIN_PROGRAMA : **STRING[1]** Cadena de 1 caracter <return> o <F>.

FIN_RENGLON : **BOOLEAN** Variable que indica si termina el renglon.

RENGLON : **ST80** Renglón que contiene una cadena de 80 caracteres.

VECTOR_FONEMAS : **ARRAY[1..10] OF ST6** Arreglo de 10 grupos fonéticos.

PALABRA : **ST14** Tamaño de la palabra.

B.2. RUTINAS QUE INTERVIENEN EN EL DESARROLLO DE ESTE PROGRAMA

(1) BORRA_MENSAJES :

Borra los mensajes relacionados con la Secuencia de Control del Sistema de cada pantalla .

* Las variables que intervienen son :

I : INTEGER variable que interviene en los ciclos.

(2) BORRA_TABLAS :
Borra la Sección de Pantalla dedicada a la Captura Alfabética de los Fonemas usados para llenar las Tablas de Fonemas .

* Las variables que intervienen son :

I : INTEGER Variable que interviene en los ciclos .

(3) BORRA_RASTREO :
Borra la Pantalla y pone la presentación entre palabra y sílaba.

* Las variables que intervienen son :

I : INTEGER Variable que interviene en los ciclos .

(4) BORRA_RENGLON :
Pone en blancos la Sección de Pantalla dedicada al renglón e imprime el renglón leído del Archivo.

* Las variables que intervienen son :

I : INTEGER Variable que interviene en los ciclos .

(5) INICIA_PANTALLA :
Inicia la Pantalla de Análisis Fonético-Silábico.

* Las variables que intervienen son :

LONG_BLOQUE : INTEGER Especifica la longitud del bloque

LONG_FONEMA : INTEGER Especifica la longitud del fonema indicando el número de muestras de que consta.

FONEMAS : FILE Archivo de fonemas .

(6) CONCATENA :
Concatena las muestras que forman el grupo fonético .

* Parámetros :

LONG_FONEMA : INTEGER Especifica la longitud del fonema indicando el número de muestras de que consta .

* Las variables que intervienen son :

REPETICIONES : INTEGER Número de repeticiones.

LONG_FONEMA : INTEGER Especifica la longitud del fonema indicando el número de muestras de que consta.

APUNT : INTEGER Es un apuntador de Propósito General.

I : INTEGER Variable que interviene en los ciclos.

J : INTEGER Variable que interviene en los ciclos.

VECTOR_DESCOMPACTACION : ARRAY [1..7100] OF BYTE Contiene el fonema descompactado.

(7) ALMACENA :
Almacena el bloque de fonemas.

* Parámetros :

NO_ELEMENTOS : INTEGER número de elementos a almacenar .

* Las variables que intervienen son :

NO_DE_BLOQUES	: INTEGER Variable que indica el número de bloque .
BLOQUE	: ARRAY [1..40] OF BYTE Vector que contiene un bloque del fonema a almacenar .
NO_DE BLOQUE	: INTEGER Contador .
I	: INTEGER Variable de Propósito General.
LONG_BLOQUE	: INTEGER Es la longitud del bloque a almacenar en disco en el Archivo de Fonemas .
FONEMAS	: FILE OF BLO Archivo de 40 bytes por registro.

(8) LEE :
Lee el fonema almacenado por lanumero de bloque .

* Tipos de variables

BLO	: Es un arreglo de hasta 40 elementos de tipo BYTE .

* Parámetros :

LONG_FONEMA	: INTEGER Especifica la longitud del fonema indicando el numero de muestras de que consta.
BLOQUE_INICIAL	: INTEGER Valor del bloque inicial .
REPETICIONES	: INTEGER Número de repeticiones.

* Las variables que intervienen son :

I	: INTEGER Variable de Propósito General .
J	: INTEGER Variable de Propósito General .
FONEMAS	: FILE OF BLO Archivo de 40 bytes .
NO_DE_BLOQUES	: INTEGER Número de bloques .
BLOQUE	: BLO Es el bloque donde se almacenan hasta 40 bytes.

(9) ABRE_ARCHIVO :
Abre el archivo de texto y lo inicializa.

TEXTO.TXT	: TEXT Archivo del texto a analizar .

(10) LEE_TABLAS :
Abre el archivo de Tablas, hace acceso al archivo mediante la longitud del dato en tablas .

* Las variables que intervienen son :

TABLAS	: FILE OF TABFON Archivo que contiene todos los atributos de cada fonema.
NOMBRE_TABLAS	: ST14 Es una cadena de 14 caracteres que contiene el nombre de las tablas .
TABLA_FONEMAS	: ARRAY[1..1000] OF TABFON Arreglo que contiene hasta 1000 fonemas con sus atributos .
TOTAL_DE_FONEMAS	: INTEGER Número total de fonemas a grabar.

(11) SALVA_TABLAS :
Graba en disco los datos almacenados en Tablas.

* Las variables que intervienen son :

TABLAS	: FILE OF TABFON Archivo que contiene todos los atributos de cada fonema.
NOMBRE_TABLAS	: ST14 Es una cadena de 14 caracteres que contiene el nombre de las tablas.
TABLA_FONEMAS	: ARRAY[1..1000] OF TABFON Arreglo que contiene hasta 1000 fonemas con sus atributos .
TOTAL_DE_FONEMAS	: INTEGER Número total de fonemas a grabar .

(12) LEE_RENGLON :
Lee un renglón del texto .

* Las variables que intervienen son :

TEXTO.TXT	: TEXT Archivo de texto.
RENGLON	: ST80 Renglón que contiene una cadena de 80 caracteres.

(13) LEE_PALABRA :
Lee e interpreta la palabra escrita en el archivo de texto.

* Las variables que intervienen son :

P : INTEGER Es un contador de letras analizadas.
RENGLON : ST80 Renglón que contiene una cadena de 80 caracteres .
NO_LETRA : INTEGER Indice del número de letra del renglón .
PALABRA : ST14 Tamaño de la palabra .
FIN_RENGLON : BOOLEAN Variable que indica si termina el renglón.
FIN_PROGRAMA : STRING[1] Cadena de 1 caracter <return> o <F>.

(14) INTERPRETA_FONEMA :
Interpreta la separación silábica de un fonema .

* Llama a las rutinas:
COMPRUEBA_QUE_FONEMA_PERTENEZCA_A_PALABRA y **BUSCA_FONEMA :**

(15) COMPRUEBA_QUE_FONEMA_PERTENEZCA_A_PALABRA :
Es una rutina de validación entre la palabra y la separación fonética escrita.

*Las variables que intervienen en esta rutina son:

FONEMA_PERTENECE_A_PALABRA : BOOLEAN Variable que indica si el fonema pertenece a la separación fonética escrita .
FONEMA : ST6 cadena de seis caracteres que contiene el fonema.
FONEMA_LOCALIZADO : BOOLEAN indica si se localizó la separación fonética en Tablas.
REGLA_LOCALIZADA : BOOLEAN indica si se localizó la regla para esa separación fonética .
F : INTEGER contador de letras que forman una separación fonética.
LONGITUD_DE_FONEMA : BYTE longitud del fonema grabado.
T : INTEGER Variable de Propósito General.
AUX : INTEGER Variable de Propósito General.
LONGITUD_FONEMA : BYTE Número de Caracteres del Fonema.
FONEMA_TEMPORAL : INTEGER Número de Fonema mas Aproximado que se pudo localizar en Tablas.
CAR : STRING[1] Detecta la Regla.
ULTIMA_LETRA_CORRECTA : BYTE Indice de la última letra correcta en la palabra.

(16) BUSCA_FONEMA :
Es una rutina que localiza el fonema en la tabla de fonemas.

* Las variables que intervienen son :

I	: INTEGER Variable que interviene enciclos.
T	: INTEGER Variable que indica el número de fonema localizado.
TABLA_FONEMAS	: ARRAY[1..1000] OF TABFON Es un arreglo de 1000 fonemas con sus atributos.
FONEMA	: STRING[20] Es una cadena de hasta 20 caracteres .
LONGITUD_DE_FONEMA	: BYTE Longitud del fonema grabado.
FONEMA_LOCALIZADO	: BOOLEAN Indica si se localizó la separación fonética en Tablas.

(17) COMPACTA :
Compacta la señal muestreada mediante el Método de Máxima Proximidad . Contiene las rutinas :

DETECTA_MAXIMA_DIFERENCIA ; GENERA_TABLAS ; COMPACTA_MUESTRAS ; DETECTA_PROXIMIDAD ; COMPACTA_DIFERENCIAS .

* Parámetros :

ELEMENTO_INICIAL	: INTEGER Elemento Inicial a Compactar .
ELEMENTO_FINAL	: INTEGER Elemento Final a Compactar .

* Las variables que intervienen son :

TOTAL_MUESTRAS	: INTEGER Total de muestras .
DIFERENCIA_MAXIMA	: INTEGER Máxima Diferencia entre muestras .
TABLA_DIFERENCIAS	: ARRAY[0..7] OF BYTE Tabla donde se almacenan los valores más próximos a los datos capturados.
MULTIPLO	: INTEGER Valor del múltiplo de tablas.
VECTOR_INDICES	: ARRAY[1..8] OF BYTE Vector de índices de compactación.
BASE	: INTEGER Es un valor de conteo de los índices del vector de compactación.
SIGNO	: INTEGER Valor del signo de la pendiente (+ o -).
INDICE	: INTEGER Indice .
NO_INDICES	: INTEGER Es el número del índice .

MUESTRAS_ANALIZADAS : INTEGER Número de muestras analizadas

AUX1 : INTEGER Es un valor calculado por diferencias parciales.
DIFERENCIA : INTEGER Es el valor absoluto de AUX1 .
I : INTEGER Variable de Propósito General.
J : INTEGER Variable de Propósito General .
DIFER_ANTERIOR : STRING[3] Cadena 'NEG' o 'POS' para indicar valor positivo o negativo de la pendiente.
DIFER_ACTUAL : STRING[3] Cadena 'NEG' o 'POS' para indicar valor positivo o negativo de la pendiente .
ERROR_ANTERIOR : INTEGER Valor del error de aproximación anterior.
ERROR_PARCIAL : INTEGER Valor del error de aproximación parcial.
ELEMENTO_MAS_PROXIMO : BYTE El valor del índice correspondiente al valor en tabla más próximo al real.
ACUMULADO : INTEGER Valor acumulado que controla el valor del error parcial de aproximación .
AUXILIAR1 : INTEGER Cálculo auxiliar .
AUXILIAR2 : INTEGER Cálculo auxiliar .
ACARREO : INTEGER Cálculo del acarreo de bits .

(18) DETECTA_MAXIMA_DIFERENCIA :
Rutina que detecta la Máxima Diferencia entre muestras .

(19) GENERA_TABLAS :
Genera las tablas de Máxima Proximidad mediante el Método de Proporciones Fijas.

(20) COMPACTA_MUESTRAS :
Concatena en un byte de Memoria los índices de la Tabla Generada obtenidos a razón de 2.66 **muestras** por 8 **bits** .

(21) DETECTA_PROXIMIDAD :
Es una rutina que detecta la proximidad entre el valor capturado y el valor codificado .

* Las variables que intervienen son :

I : INTEGER Variable de control de ciclo .
J : INTEGER Variable de control de ciclo .

DIFER_ANTERIOR : STRING[3] Cadena 'NEG' o 'POS' para indicar valor positivo o negativo de la pendiente.

DIFER_ACTUAL : STRING[3] Cadena 'NEG' o 'POS' para indicar valor positivo o negativo de la pendiente .

ERROR_ANTERIOR : INTEGER Valor del error de aproximación anterior .

ERROR_PARCIAL : INTEGER Valor del error de aproximación parcial .

ELEMENTO_MAS_PROXIMO : BYTE El valor del índice correspondiente al valor en tabla más próximo al real .

ACUMULADO : INTEGER Valor acumulado que controla el valor del error parcial de aproximación .

AUX1 : INTEGER Es un valor calculado por diferencias finitas .

DIFERENCIA : INTEGER Es el valor absoluto de AUX1 .

NO_INDICES : INTEGER Es el número del indice .

BASE : INTEGER Es un valor de conteo de los índices del vector de compactación .

VECTOR : ARRAY[0..7000] OF BYTE Vector de muestras capturadas.

SIGNO : INTEGER Valor del signo de la pendiente (+ o -) .

ELEMENTO_INICIAL : INTEGER Parámetro de muestra inicial .

TOTAL_MUESTRAS : INTEGER Total de muestras a capturar .

MUESTRAS_ANALIZADAS : INTEGER Número de muestras analizadas .

INDICE : INTEGER Indice .

TABLA_DIFERENCIAS : ARRAY[0..7] OF BYTE Tabla donde se almacenan los valores más próximos a los datos capturados .

(22) COMPACTA_DIFERENCIAS :
Método de Compactación de Memoria por Máxima Proximidad .

* Llama a las rutinas **GENERA_TABLAS y DETECTA_PROXIMIDAD.**

(23) DESCOMPACTA_DIFERENCIAS :
Método que extrae los valores compactados y los descompone en sus elementos .

* Las variables que intervienen son :

I : INTEGER Variable que interviene en ciclos

J : INTEGER Variable que interviene en ciclos

BASE2 : INTEGER Variable que controla el índice de los valores descompactados.
VECTOR_INDICES : ARRAY[1..8] OF BYTE Vector de índices de descompactación.
ACUMULADO : INTEGER Valor acumulado.
AUXILIAR1 : INTEGER Cálculo auxiliar.
AUXILIAR2 : INTEGER Cálculo auxiliar.
AUXILIAR3 : INTEGER Cálculo auxiliar.

ACARREO : INTEGER Cálculo del acarreo de bits.
MULTIPLO : INTEGER Valor del número múltiplo.
TABLA_DIFERENCIAS : ARRAY[0..7] OF BYTE Tabla donde se almacenan los valores más próximos a los datos capturados.
BASE : INTEGER Es un valor de conteo de los índices del vector de compactación.
SIGNO : INTEGER Valor del signo de la pendiente (+ o -).

(24) GRAFICA_MUESTRAS :
Rutina que grafica todas las muestras en un plano XY tomando en cuenta escala, desplazamientos, cortes .

Llama a las rutinas :
REGENERA ; REPRODUCE ; LISTA_PARAMETROS ;
AJUSTE_AMPLITUD ; MUESTRAS_EN_PANTALLA ;
CORTE_DE_MUESTRAS ; LIMITE_PANTALLA ;
INICIA ; INICIALIZA_PARAMETROS ;
DESPLAZAMIENTOS ;

* Las variables que intervienen son :

I : INTEGER Variable de Propósito General.
J : INTEGER Variable de Propósito General.
XM : INTEGER Límite Máximo en X de la Pantalla.
YM : INTEGER Límite Máximo en Y de la Pantalla.
INC : INTEGER Incrementos de Escalamiento.
MUESTRA_INICIAL : INTEGER Muestra Inicial a Graficar.
MUESTRAS_PANTALLA : INTEGER Número de Muestras a Graficar.
DESP_Y : INTEGER Desplazamiento Vertical.
MUESTRA : INTEGER Muestra de Referencia.
M_AUX : INTEGER Variable auxiliar.
AMPLITUD : INTEGER Amplitud de la señal.
COLOR_ESCALAS : INTEGER Codigo de color de las escalas.
COLOR_GRAFICA : INTEGER Codigo de color de la gráfica.
INC_MUESTRAS : INTEGER Incrementos entre Muestras.
A : CHAR Variable de Control.
B : CHAR Variable de Control.

TOTAL_DE_MUESTRAS : INTEGER Total de Muestras.
M_I : INTEGER Variable auxiliar.
D_Y : INTEGER Variable auxiliar.
AMP : INTEGER Amplitudes en la Escala.
DESP_X_MIN : INTEGER Desplazamiento Mínimo en X.
DESP_X_MAX : INTEGER Desplazamiento Máximo en X.
DESP_Y_MAX : INTEGER Desplazamiento Máximo en Y.
DESP_Y_MIN : INTEGER Desplazamiento Mínimo en Y.

(25) REGENERA :
Rutina que regenera la gráfica.

(26) REPRODUCE :
Rutina que reproduce la señal de voz.

(27) LISTA_PARAMETROS :
Lista los parámetros de Graficación de la señal, como Amplitud , Desplazamiento Vertical, Desplazamiento Horizontal , Muestra Inicial Desplegada , Muestra Final en Pantalla , Número de Muestras en Pantalla.

* Las variables que intervienen son :

AMPLITUD : INTEGER Amplitud de la señal.
DESP_Y : INTEGER Desplazamiento Vertical.
MUESTRA_INICIAL : INTEGER Muestra Inicial Desplegada.
MUESTRAS_PANTALLA : INTEGER Número de Muestras en Pantalla.
YM : INTEGER Límite Máximo en Y de la Pantalla.
XM : INTEGER Límite Máximo en X de la Pantalla.

(28) AJUSTE_AMPLITUD :
Fase de Ajuste de Amplitud de la Señal.

* Las variables que intervienen son :

B : CHAR Variable de Control.
I : INTEGER Variable de Propósito General.
YM : INTEGER Límite Máximo en Y de la Pantalla.
VECTOR : ARRAY[0..7000] OF BYTE Contiene las muestras de la señal de voz.

MUESTRA_INICIAL : INTEGER Apunta a la muestra inicial.
M_AUX : INTEGER Apuntador auxiliar.
AMPLITUD : INTEGER Amplitud de la señal.
DESP_Y : INTEGER Desplazamiento Vertical.
TABLA_FONEMAS : ARRAY[1..1000] OF TABFON arreglo que contiene hasta 1000 fonemas con sus atributos.
LONG_FONEMA : INTEGER Especifica la longitud del fonema indicando el número de muestras de que consta.

MUESTRA_FINAL : INTEGER Muestra Final Desplegada.
MUESTRA_INICIO : INTEGER Muestra Inicio.
INICIO_FONEMA : INTEGER Inicio del Fonema.
FIN_FONEMA : INTEGER Fin del Fonema.

(29) MUESTRAS_EN_PANTALLA :
Cambia el número de muestras en pantalla .

(30) CORTE_DE_MUESTRAS :
Realiza graficamente cortes de análisis de la señal muestreada.

(31) LIMITE_PANTALLA :
Es una rutina que nos da los Límites Máximos que se desean de la Pantalla, los cuales pueden ser modificados en esta misma rutina.

* Las variables que intervienen son :

YM : INTEGER Límite Máximo en Y de la Pantalla.
XM : INTEGER Límite Máximo en X de la Pantalla.

(32) INICIA :
Procedimiento de inicialización de Parámetros de Graficación de la Señal.

* Las variables que intervienen son :

AMPLITUD : INTEGER Amplitud de la señal.
DESP_Y : INTEGER Desplazamiento Vertical.
MUESTRA_INICIAL : INTEGER Muestra Inicial Desplegada.
MUESTRAS_PANTALLA : INTEGER Número de Muestras en Pantalla.

YM : INTEGER Límite Máximo en Y de la Pantalla.
XM : INTEGER Límite Máximo en X de la Pantalla.

(33) INICIALIZA_PARAMETROS :
Llama a la rutina INICIA.

* Las variables que intervienen son :

B : CHAR Variable que indica si se desean cambiar los parámetros.

(34) DESPLAZAMIENTOS :
Rutina que procesa los cambios de desplazamientovertical y horizontal.

* Las variables que intervienen son :

DESP_X_MAX : INTEGER Desplazamiento Máximo en X.
DESP_Y_MAX : INTEGER Desplazamiento Máximo en Y.
B : CHAR Variable de Control.

(35) CAPTURA_FONEMA :
Realiza la Captura de los Fonemas Pronunciados desde el Micrófono y Digitalizados en la Tarjeta.

(36) PREGUNTA_SEPARACION_FONETICA :
Rutina que pregunta por la separación fonética.

* Las variables que intervienen son :

AUX : INTEGER Uso auxiliar múltiple.
CAR : STRING[1] Usado para rastrear los caracteres del texto.
PREGUNTA : CHAR Indica cuándo una pregunta se contesta afirmativa o negativamente.
FONEMA_PERTENECE_A_PALABRA : BOOLEAN Variable que indica si el fonema pertenece a la separación fonética escrita.

TOTAL_DE_FONEMAS : INTEGER Número total de fonemas a grabar.
FONEMA : STRING[20] Es una cadena de hasta 20 caracteres.
TABLA_FONEMAS : ARRAY[1..1000] OF TABFON Arreglo que contiene hasta 1000 fonemas con sus atributos.
PALABRA : ST14 Contiene la palabra en análisis.
ULTIMA_LETRA_CORRECTA : BYTE Valor de la última letra correcta.
I : INTEGER Variable de Propósito General.
ULT_LET_ANAL_TEMP : BYTE Valor de la última letra analizada temporalmente.

REGLAS : STRING[20] Conjunto de letras esperadas después de una separación fonética.
TABLAS : FILE OF TABFON Archivo de Fonemas con sus atributos.
LONG_FONEMA : INTEGER Especifica la longitud del fonema indicando el número de muestras de que consta.
BLOQUE_INICIAL : INTEGER Valor del bloque inicial.
REPETICIONES : INTEGER Número de repeticiones.
FONEMA_LOCALIZADO : BOOLEAN Indica si se localizó la separación fonética en Tablas.
ULTIMA_LETRA_ANALIZADA : BYTE Ultima letra analizada.
VECTOR_FONEMAS : ARRAY[1..10] OF ST6 Arreglo de 10 grupos fonéticos.
NUMERO_DE_FONEMAS_ENCONTRADOS: BYTE Número de fonemas encontrados.

(37) INICIALIZA_TABLA_DE_FONEMAS :
Inicializa los atributos de cada fonema .

* Las variables que intervienen son :

T : INTEGER Variable de Propósito General .
I : INTEGER Variable de Propósito General .
TABLA_FONEMAS : ARRAY[1..1000] OF TABFON arreglo que contiene hasta 1000 fonemas con sus atributos.
FONEMA : ST6 Cadena de hata seis caracteres de un fonema analizado.
REGLAS : STRING[20] Conjunto de letras esperadas después de una separación fonética.

(38) DESPLIEGA_SEPARACION_FONETICA_ENCONTRADA :
Despliega en Pantalla la Separación Fonética Encontrada en el Archivo de Tablas .

* Las variables que intervienen son :

I : INTEGER Variable de Propósito General .
NUMERO_DE_FONEMAS_ENCONTRADOS: INTEGER Número de Fonemas Encontrados.
VECTOR_FONEMAS : ARRAY[1..10] OF ST6 Arreglo de 10 grupos fonéticos.

LISTADO DEL PROGRAMA

LISTADO DEL PROGRAMA DEL SISTEMA FONETICS

```
PROGRAM FONETICS(INPUT,OUTPUT,TEXTO);

TYPE

     ST80=STRING[80];
     ST14=STRING[14];
     ST6=STRING[6];
     ST30=STRING[30];

     TABFON=RECORD
          FONEMA:STRING[6]; (*LONG MAXIMA*)
          REGLAS:STRING[20];
          BLOQUE INICIAL:INTEGER;
          LONG FONEMA:INTEGER;
          REPETICIONES:BYTE;
          BLOQUES:BYTE;
     END;

     FE=ARRAY[1..10] OF ST6;
     BLO=ARRAY[1..40] OF BYTE; (* CAMBIAR AQUI LA LONG DEL BLOQUE *)


CONST

     LETRAS:SET OF  CHAR=['A'..'Z','a'..'z'];
     SIMBOLOS:SET OF CHAR=[' ','"','(',')','.',',','-',':',';','?'];
     LONG MAXIMA:BYTE=6;
     INDICE MAXIMO:INTEGER=3000;


VAR

     TEXTO:TEXT;
     TABLAS:FILE OF TABFON;
     RENGLON:ST80;
     NOMBRE:ST14;
     NOMBRE TABLAS:ST14;
     PALABRA:ST14;
     NO LETRA:[1..3]:INTEGER;
     TOTAL DE FONEMAS:INTEGER;
```

```
FIN RENGLON:BOOLEAN;
VECTOR FONEMAS:FE;
TABLA FONEMAS:ARRAY[1..720] OF TABFON;
ULTIMA LETRA ANALIZADA:BYTE;
ULTIMA LETRA CORRECTA:BYTE;
ULT LET ANAL TEMP:BYTE;
ULT LET CORR TEMP:BYTE;
NUMERO DE FONEMAS ENCONTRADOS:BYTE;
SEPARACION OK:STRING[1];
SEPARACION FONETICA CORRECTA:ST30;
ALFA.TABLAS ANTERIORES.BORRAR ARCHIVO:STRING[1];
FONEMA PERTENECE A PALABRA:BOOLEAN;
FIN PROGRAMA:STRING[1];
VECTOR:ARRAY [0 ..7050] OF BYTE;
VECTOR DESCOMPACTACION:ARRAY [1 ..7050] OF BYTE;
VECTOR COMPACTACION:ARRAY [0 ..5050] OF BYTE;
VECTOR REPRODUCCION:ARRAY [1 ..20000] OF BYTE;
NO MUESTRAS:INTEGER; (* NUMERO DE MUESTRAS A CAPTURAR *)
REPETICIONES:INTEGER; (* NUMERO DE REPETICIONES DEL MICROFONEMA *)
NOMBRE ARCHIVO FONEMAS:STRING [14]; (* USADO EN ALMACENA *)
INICIO FONEMA:INTEGER;
FIN FONEMA:INTEGER;
APUNT.NO ELEMENTOS:INTEGER;
LONG BLOQUE:INTEGER;
MUESTRA INICIO:INTEGER;
MUESTRA FINAL:INTEGER;
BLOQUE INICIAL:INTEGER;
PROCESO COMPLETO:BOOLEAN;
FONEMAS:FILE OF BLO;
```

DECLARACION DE TIPOS, CONSTANTES Y VARIABLES QUE INTERVIENEN EN TODAS LAS RUTINAS DEL PROGRAMA FONETICS.

```
{$R+}  (* ACTIVA CHEQUEO DE RANGO *)

PROCEDURE BORRA MENSAJES;

BEGIN
     GOTOXY(1,21);
     FOR I:=1 TO 80 DO
         WRITE(' ');
     GOTOXY(1,22);
     FOR I:=1 TO 3 DO
          BEGIN
               CLREOL;
               WRITELN;
          END;
     GOTOXY(1,22);
END;


PROCEDURE BORRA TABLAS;

BEGIN
     GOTOXY(1,10);
     CLREOL;
     GOTOXY(1,10);
     WRITELN('       ');
     GOTOXY(1,11);
     FOR I:=1 TO 80 DO
          WRITE(' ');
     GOTOXY(1,12);
     FOR I:=1 TO 9 DO
         BEGIN
               CLREOL;
               WRITELN;
          END;
     GOTOXY(1,12);
END;


PROCEDURE BORRA RASTREO;

BEGIN
     GOTOXY(1,4);
     FOR I:=1 TO 80 DO
          WRITE(' ');
     GOTOXY(1,5);
     CLREOL;
     GOTOXY(1,5)
     WRITELN('PALABRA');
```

```
     GOTOXY(25,5);
     WRITELN('SILABA');
     GOTOXY(45,5);
     WRITELN('      ');
     GOTOXY(1,6);
     FOR I:=1 TO LONG MAXIMA DO
     BEGIN
          CLREOL;
          WRITELN;
     END;
     GOTOXY(1,6);
     WRITE('                    ');
     GOTOXY(1,6);
     WRITELN(PALABRA);
END;


PROCEDURE BORRA RENGLON;

BEGIN
     GOTOXY ( 1 , 1 );
     CLREOL;
     GOTOXY(36,1);
     WRITELN('RENGLON');
     GOTOXY(1,2);
     FOR I:=1 TO 80 DO
          WRITE(' ');
     GOTOXY(1,3);
     CLREOL;
     GOTOXY(1,3);
     WRITE(RENGLON);
```

```
PROCEDURE INICIA PANTALLA;

BEGIN
     CLRSCR;
     BORRA RENGLON;
     BORRA RASTREO;
     BORRA TABLAS;
     BORRA MENSAJES;
END;


PROCEDURE CONCATENA ( LONG FONEMA : INTEGER ) ;

BEGIN
IF (REPETICIONES*LONG FONEMA+APUNT) <= 20000  THEN
     BEGIN
     FOR I :=0 TO REPETICIONES-1 DO
     FOR J :=1 TO LONG FONEMA DO
          BEGIN
          VECTOR REPRODUCCION[ I * LONG FONEMA + J + APUNT ]:=
          VECTOR DESCOMPACTACION[J];
          END ;
     APUNT:=(REPETICIONES)*LONG FONEMA+APUNT;
END
ELSE
     BEGIN
     WRITELN (' *** SE EXCEDIO LA CAPACIDAD DEL VECTOR DE REPRODUCCION
     <RETURN>',
     CHAR(7) );
     REPEAT UNTIL KEYPRESSED;
     END;
END;


PROCEDURE ALMACENA;

VAR

     NO DE BLOQUES  :INTEGER;
     BLOQUE         :BLO;
     NO DE BLOQUE. I:INTEGER;

BEGIN
     NO DE BLOQUES:=( NO ELEMENTOS + 1 ) DIV LONG BLOQUE;
     IF ( NO ELEMENTOS MOD LONG BLOQUE )  >  0 THEN
          NO DE BLOQUES:=NO DE BLOQUES + 1;
     ASSIGN ( FONEMAS, NOMBRE ARCHIVO FONEMAS );
     RESET ( FONEMAS );
```

```
     BLOQUE INICIAL:=FILESIZE ( FONEMAS );
     SEEK ( FONEMAS , FILESIZE ( FONEMAS ) );
     NO DE BLOQUE:=0;
     WRITELN ('PORCENTAJE DE COMPACTACION = ',
          100-(NO ELEMENTOS/(MUESTRA FINAL-MUESTRA INICIO))*100.'%');
     REPEAT
          I:=1;
          REPEAT
               BLOQUE[I]:=VECTOR COMPACTACION[I+
                    LONG BLOQUE*NO DE BLOQUE-1];
               I:=I+1;
          UNTIL (I=LONG BLOQUE+1);
          WRITE (FONEMAS,BLOQUE);
          FLUSH(FONEMAS);
          NO DE BLOQUE:=NO DE BLOQUE+1;
     UNTIL (NO DE BLOQUE=NO DE BLOQUES);
     CLOSE(FONEMAS);
     TABLA FONEMAS(TOTAL DE FONEMAS)
          .BLOQUES:=NO DE BLOQUES;
END;



PROCEDURE LEE(NO DE BLOQUES,LONG FONEMA,BLOQUE INICIAL
               ,REPETICIONES:INTEGER);

TYPE

     BLO=ARRAY[1..40] OF BYTE;

VAR

     I,J,LONG VEC CAP:INTEGER;
     FONEMAS:FILE OF BLO;
     BLOQUE:BLO;

BEGIN
     ASSIGN (FONEMAS,NOMBRE ARCHIVO FONEMAS);
     RESET (FONEMAS);
     SEEK(FONEMAS,BLOQUE INICIAL);
     FOR I:=0 TO 5000 DO
          VECTOR COMPACTACION[I]:=0;
```

```
FOR I:=0 TO NO DE BLOQUES-1 DO
BEGIN
     READ(FONEMAS,BLOQUE);
     J:=0;
     REPEAT
          VECTOR COMPACTACION[J+1*LONG BLOQUE]:=BLOQUE[J+1];
          J:=J+1;
     UNTIL ((J+(*LONG BLOQUE)=(LONG FONEMA+1))OR(J=LONG BLOQUE);
END;
CLOSE(FONEMAS);
```

ESTOS PROCEDIMIENTOS INICIAN PANTALLA DE COMUNICACION CON EL USUARIO, ALMACENAMIENTO DE DATOS Y LECTURA

```
 PROCEDURE ABRE ARCHIVO(NOMBRE:ST14);
BEGIN
     ASSIGN (TEXTO,NOMBRE);
     RESET (TEXTO);
END;


PROCEDURE LEE TABLAS;
 BEGIN
     ASSIGN(TABLAS,NOMBRE TABLAS);
     RESET(TABLAS);
     I:=1;
     REPEAT
          READ(TABLAS,TABLA FONEMAS[I]);
          I:=I+1;
     UNTIL EOF(TABLAS);
     TOTAL DE FONEMAS:=I-1;
     CLOSE(TABLAS);
END;


PROCEDURE SALVA TABLAS;
BEGIN
     ASSIGN(TABLAS,NOMBRE TABLAS);
     RESET(TABLAS);
     SEEK(TABLAS,FILESIZE(TABLAS));
     WRITE(TABLAS,TABLA FONEMAS[TOTAL DE FONEMAS]);
     BORRA MENSAJES;
     WRITELN(CHAR(7),'*** SE ACTUALIZO EL FONEMA EN DISCO');
     WRITELN('******* LONGITUD = ',FILESIZE(TABLAS));
     FLUSH(TABLAS);
     CLOSE(TABLAS);
END;


PROCEDURE LEE RENGLON; (*CONTIENE EL RENGLON LEIDO*)

VAR
     NO:STRO;
     I:INTEGER;
BEGIN
     READLN(TEXTO,RENGLON);
END;
```

```
PROCEDURE LEE PALABRA;

VAR
     P:INTEGER;
BEGIN
     P:=1;
     BORRA RENGLON;
     WHILE ((RENGLON[NO LETRA] IN SIMBOLOS)
          AND
          (NOT (NO LETRA=80))) DO
          NO LETRA:=NO LETRA+1;
     WHILE ((RENGLON[NO LETRA] IN LETRAS)
          AND
          (NOT (NO LETRA=80))) DO
     BEGIN
          PALABRA[P]:=RENGLON[NO LETRA];
          P:=P+1;
          NO LETRA:=NO LETRA+1;
     END;
     IF NO LETRA>=80 THEN
          FIN RENGLON:=TRUE;
     BORRA MENSAJES;
     WRITELN;
     WRITE('TECLEE <F> <RETURN> PARA TERMINAR EL PROGRAMA O <RETURN> PARA
          CONTINUAR ->');
     READLN(FIN PROGRAMA);
```

ESTOS PROCEDIMIENTOS ALMACENAN Y LEEN LAS TABLAS DE FONEMAS Y ADEMAS LEEN LOS RENGLONES Y LAS PALABRAS DE CADA RENGLON A ANALIZAR

```
 PROCEDURE INTERPRETA FONEMA;

TYPE

     ST6=STRING[6];(* LONG MAXIMA *)

VAR

     FONEMA:ST6;
     FONEMA LOCALIZADO:BOOLEAN;
     REGLA LOCALIZADA:BOOLEAN;
     F:INTEGER;(* CONTADOR DE CARACTERES DEL FONEMA *)
     LONGITUD DE FONEMA:BYTE;
     T.AUX:INTEGER;
     LONGITUD FONEMA:BYTE;
     FONEMA TEMPORAL:INTEGER;
     CAR:STRING[1];
     NUEVA REGLA:BOOLEAN;


PROCEDURE COMPRUEBA QUE FONEMA PERTENEZCA A PALABRA;

BEGIN
     FONEMA PERTENECE A PALABRA:=TRUE;
     LONGITUD DE FONEMA:=0;
     FOR I:=1 TO LONG MAXIMA DO
          IF (TABLA FONEMAS[TOTAL DE FONEMAS].FONEMA[I]<>CHR(32)) THEN
               LONGITUD DE FONEMA:=LONGITUD DE FONEMA+1;
     FOR I:=ULTIMA LETRA CORRECTA+1 TO ULTIMA LETRA CORRECTA
          +LONGITUD DE FONEMA DO
     BEGIN
          IF (PALABRA[I]<>
        TABLA FONEMAS[TOTAL DE FONEMAS].FONEMA[I-ULTIMA LETRA CORRECTA])
          THEN
               FONEMA PERTENECE A PALABRA:=FALSE;
     END;
     IF (LONGITUD DE FONEMA=0) AND (PALABRA[I]<>CHR(32)) THEN
          FONEMA PERTENECE A PALABRA:=FALSE;
END;


PROCEDURE BUSCA FONEMA;

BEGIN
     GOTOXY(35,6);
     WRITE(' ');
     GOTOXY(25,6);
     FOR I:=1 TO  6 DO
          WRITE(' ');
```

```
T:=1;
REPEAT
     REPEAT
          FONEMA LOCALIZADO:=TRUE;
          FOR I:=1 TO LONG MAXIMA DO
          BEGIN
               IF TABLA FONEMAS[T].FONEMA[I]<>FONEMA[I] THEN
                    FONEMA LOCALIZADO:=FALSE;
          END;
     T:=T+1;
UNTIL ((T>=TOTAL DE FONEMAS+1) OR (FONEMA LOCALIZADO));
GOTOXY(25,6);
FOR I:=1 TO  6 DO
     WRITE(' ');
GOTOXY(25,6);
WRITE(FONEMA);
LONGITUD FONEMA:=0;
REPEAT
     LONGITUD FONEMA:=LONGITUD FONEMA+1;
UNTIL(TABLA FONEMAS[T-1].FONEMA[LONGITUD FONEMA]=CHAR(32))
     OR (LONGITUD FONEMA=LONG MAXIMA);
IF (FONEMA LOCALIZADO) THEN
     BEGIN
          I:=1;
          REGLA LOCALIZADA:=FALSE;
          REPEAT
               IF TABLA FONEMAS[T-1].REGLAS[I]=
               PALABRA[ULTIMA LETRA ANALIZADA+I]
               THEN
                    BEGIN
                         REGLA LOCALIZADA:=TRUE;
                         FONEMA TEMPORAL:=T-1;
                    END;
               I:=I+1;
          UNTIL (I>=21) OR REGLA LOCALIZADA
          OR (TABLA FONEMAS[T-1].REGLAS[I]=CHAR(0));
          IF NOT(REGLA LOCALIZADA) THEN
          BEGIN
               FONEMA TEMPORAL:=T;
               ULT LET ANAL TEMP:=ULTIMA LETRA ANALIZADA;
               ULT LET CORR TEMP:=ULTIMA LETRA CORRECTA;
          END;
     END;
UNTIL (FONEMA LOCALIZADO AND REGLA LOCALIZADA)
OR
(T>=TOTAL DE FONEMAS+1);
```

ESTOS PROCEDIMIENTOS BUSCAN EN LA TABLA DE FONEMAS E INTERPRETAN CADA FONEMA Y LO RELACIONAN CON LA SILABA O PALABRA CORRESPONDIENTE

```
PROCEDURE COMPACTA(ELEMENTO INICIAL:INTEGER;ELEMENTO FINAL:INTEGER);

VAR

     TOTAL MUESTRAS:INTEGER;
     DIFERENCIA MAXIMA:INTEGER;
     TABLA DIFERENCIAS:ARRAY [ 0..7 ] OF BYTE;
     MULTIPLO:INTEGER;
     VECTOR INDICES:ARRAY [ 1..8 ] OF BYTE;
     BASE:INTEGER;
     SIGNO:INTEGER;
     INDICE:INTEGER;
     NO INDICES:BYTE;
     MUESTRAS ANALIZADAS:INTEGER;


     PROCEDURE DETECTA MAXIMA DIFERENCIA;

     VAR

          AUX1:INTEGER;
          DIFERENCIA:INTEGER;

     BEGIN
          VECTOR [ 0 ]:=128;
          DIFERENCIA MAXIMA:=0;
          FOR I:=ELEMENTO INICIAL TO ELEMENTO FINAL DO
          IF ABS(VECTOR[I]-VECTOR[I-1]) > DIFERENCIA MAXIMA THEN
          DIFERENCIA MAXIMA:=ABS(VECTOR[I]-VECTOR[I-1]);
     END;


     PROCEDURE GENERA TABLAS;

     VAR

          I:INTEGER;

     BEGIN
          DETECTA MAXIMA DIFERENCIA;
          MULTIPLO:=DIFERENCIA MAXIMA DIV 7;
          IF MULTIPLO=0 THEN
               MULTIPLO:=1;
               VECTOR COMPACTACION [ 0 ]:=MULTIPLO;
               TABLA DIFERENCIAS[0]:=0;
               FOR I := 1 TO 7 DO
                    TABLA DIFERENCIAS[I]:=I*MULTIPLO;
          END;
```

```
PROCEDURE COMPACTA MUESTRAS;

VAR

     I:INTEGER;

BEGIN
     IF (BASE+3)<5000 THEN
     BEGIN
          NO INDICES:=NO INDICES+1;
          VECTOR INDICES[NO INDICES]:=INDICE;
          IF (NO INDICES=8) OR (MUESTRAS ANALIZADAS=TOTAL MUESTRAS)
          THEN
          BEGIN
               NO INDICES:=0;
               VECTOR COMPACTACION[1+BASE]:=VECTOR INDICES[1]*32;
               VECTOR COMPACTACION[1+BASE]:=VECTOR INDICES[2]*4+
                    VECTOR COMPACTACION[1+BASE];
               VECTOR COMPACTACION[1+BASE]:=(VECTOR INDICES[3] DIV
                    2)+
               VECTOR COMPACTACION[1+BASE];
               VECTOR COMPACTACION[2+BASE]:=(VECTOR INDICES[3]MOD 2)
                    *128+
               (VECTOR INDICES[4]*16);
               VECTOR COMPACTACION[2+BASE]:=(VECTOR INDICES[5]*2)+
                    VECTOR COMPACTACION[2+BASE];
               VECTOR COMPACTACION[2+BASE]:=(VECTOR INDICES[6]
                    DIV 4)+
               VECTOR COMPACTACION[2+BASE];
               VECTOR COMPACTACION[3+BASE]:=(VECTOR INDICES[6]
                    MOD 4)
                    *64;
               VECTOR COMPACTACION[3+BASE]:=VECTOR INDICES[7]*4+
                    VECTOR COMPACTACION[3+BASE];
               VECTOR COMPACTACION[3+BASE]:=VECTOR INDICES[8]+
                    VECTOR COMPACTACION[3+BASE];
               BASE:=BASE+3;
               NO ELEMENTOS:=BASE;
          END;
     END
     ELSE
          WRITELN (' **** SE EXCEDIO LA CAPACIDAD DEL VECTOR DE
               COMPACTACION ',
               ' BASE + 3 = '.BASE+3.CHAR(7));
END;
```

```
PROCEDURE DETECTA PROXIMIDAD;

VAR

     I,J,AUX1:INTEGER;
     DIFER ANTERIOR:STRING[3];
     DIFER ACTUAL:STRING[3];
     ERROR ANTERIOR:INTEGER;
     ERROR PARCIAL:INTEGER;
     ELEMENTO MAS PROXIMO:BYTE;
     ACUMULADO,DIFERENCIA:INTEGER;

BEGIN
     NO INDICES:=0;
     BASE:=0;
     DIFER ANTERIOR:='POS';
     VECTOR[0]:=128;
     SIGNO:=1;
     ACUMULADO:=VECTOR[0];
     MUESTRAS ANALIZADAS:=0;
     FOR I:=ELEMENTO INICIAL TO ELEMENTO INICIAL+TOTAL MUESTRAS DO
     BEGIN
          MUESTRAS ANALIZADAS:=MUESTRAS ANALIZADAS+1;
          ERROR ANTERIOR:=999;
          AUX1:=VECTOR[I]-ACUMULADO;
          IF AUX1>0 THEN
               DIFER ACTUAL:='POS'
          ELSE
               DIFER ACTUAL:='NEG';
          IF DIFER ACTUAL<>DIFER ANTERIOR THEN
          BEGIN
               DIFER ANTERIOR:=DIFER ACTUAL;
               INDICE:=0;
               SIGNO:=SIGNO*(-1);
               COMPACTA MUESTRAS;
          END;
          DIFERENCIA:=ABS(AUX1);
          FOR J:=1 TO 7 DO
          BEGIN
               ERROR PARCIAL:=ABS(TABLA DIFERENCIAS[J]
                    -DIFERENCIA);
               IF ERROR PARCIAL<ERROR ANTERIOR THEN
               BEGIN
                    ELEMENTO MAS PROXIMO:=J;
                    ERROR ANTERIOR:=ERROR PARCIAL;
               END;
          END;
          INDICE:=ELEMENTO MAS PROXIMO;
          COMPACTA MUESTRAS;
```

```
            ACUMULADO:=ACUMULADO+(SIGNO*TABLA DIFERENCIAS
                 [ELEMENTO MAS PROXIMO]);
            IF ACUMULADO<0 THEN
            ACUMULADO:=0;
            IF ACUMULADO>255 THEN
            ACUMULADO:=255;
        END;
    END;


    PROCEDURE COMPACTA DIFERENCIAS;

    BEGIN
        GENERA TABLAS;
        DETECTA PROXIMIDAD;
    END;

AQUI SE ENCUENTRA EL INICIO DEL PROCEDURE COMPACTA

BEGIN
    FOR I:=1 TO INDICE MAXIMO DO
    VECTOR COMPACTACION[I]:=0;
    FOR I :=1 TO 8 DO
    VECTOR INDICES[I]:=0;
    TOTAL MUESTRAS:=ELEMENTO FINAL-ELEMENTO INICIAL;
    IF (TOTAL MUESTRAS MOD 8)<>0 THEN
    TOTAL MUESTRAS:=((TOTAL MUESTRAS DIV 8)+1)*8;
    COMPACTA DIFERENCIAS;
```

ESTOS PROCEDIMIENTOS REALIZAN EL PROCESO DE COMPACTACION DE LA INFORMACION EN MEMORIA

```
 PROCEDURE DESCOMPACTA DIFERENCIAS;

VAR

     I,J:INTEGER;
     BASE2:INTEGER;
     VECTOR INDICES:ARRAY[1 .. 8] OF BYTE;
     ACUMULADO:INTEGER;
     AUXILIAR1:INTEGER;
     AUXILIAR2:INTEGER;
     AUXILIAR3:INTEGER;
     ACARREO:INTEGER;
     MULTIPLO:INTEGER;
     TABLA DIFERENCIAS:ARRAY [0 .. 7] OF BYTE;
     BASE:INTEGER;
     SIGNO:INTEGER;

BEGIN
     MULTIPLO:=VECTOR COMPACTACION [0];
     TABLA DIFERENCIAS[0]:=0;
     (* DEPURADO HASTA AQUI *)
     FOR I := 1 TO 7 DO
     TABLA DIFERENCIAS [I] := I * MULTIPLO ;
     FOR I:= 1 TO 7000 DO
     VECTOR DESCOMPACTACION [I]:=0;
     BASE := 0 ;
     BASE2 := 0 ;
     SIGNO := 1 ;
     ACUMULADO := 128 ;
     REPEAT
          AUXILIAR1 := VECTOR COMPACTACION [ 1 + BASE ] DIV 32 ;
          VECTOR INDICES [ 1 ] := AUXILIAR1 ;
          AUXILIAR2 := VECTOR COMPACTACION [ 1 + BASE ] DIV 4
               - ( AUXILIAR1 * 8 ) ;
          VECTOR INDICES [ 2 ] := AUXILIAR2 ;
          ACARREO := VECTOR COMPACTACION [ 2 + BASE ] DIV 128 ;
          VECTOR INDICES [ 3 ] :=( VECTOR COMPACTACION [ 1 + BASE ] - (
               AUXILIAR1 * 32 )
               - ( AUXILIAR2 * 4 ) ) * 2 + ACARREO ;
          AUXILIAR1 := ( VECTOR COMPACTACION [ 2 + BASE ] - ( ACARREO *
               128 )
               ) DIV 16 ;
          VECTOR INDICES [ 4 ] := AUXILIAR1 ;
          AUXILIAR2 := ( VECTOR COMPACTACION [ 2 + BASE ] - ( ACARREO *
               128 ) )
               DIV 2 - ( AUXILIAR1 * 8 ) ;
          VECTOR INDICES [ 5 ] := AUXILIAR2 ;
          AUXILIAR3:= VECTOR COMPACTACION [2 + BASE ] - ( ACARREO * 128 )
               ;
          ACARREO := VECTOR COMPACTACION [ 3 + BASE ] DIV 64 ;
```

```
VECTOR INDICES [ 6 ] := ( AUXILIAR3- ( AUXILIAR1 * 16 ) - (
     AUXILIAR2 * 2 ) ) * 4 + ACARREO :
AUXILIAR1 := ( VECTOR COMPACTACION [ 3 + BASE ] - ( ACARREO *
     64 ))
     DIV 8 :
VECTOR INDICES [ 7 ] := AUXILIAR1 :
AUXILIAR2 := ( VECTOR COMPACTACION [ 3 + BASE ] - ( ACARREO *
     64 ))
     - ( AUXILIAR1 * 8 ) :
VECTOR INDICES [ 8 ] := AUXILIAR2 :
AUXILIAR1 := 0 :
J := 0 :
FOR I := 1 TO 8 DO
     IF VECTOR INDICES [ I ] = 0 THEN
          SIGNO := SIGNO * ( -1 )
     ELSE
     BEGIN
          J := J + 1 :
          ACUMULADO := ACUMULADO + ( SIGNO * TABLA DIFERENCIAS
               [
               VECTOR INDICES [ I ] ] ) :
          IF ACUMULADO < 0 THEN
          ACUMULADO := 0 :
          IF ACUMULADO > 255 THEN
               ACUMULADO :=  255 :
          VECTOR DESCOMPACTACION [ J + BASE2 ] := ACUMULADO :
          AUXILIAR1 := AUXILIAR1 + 1 :
     END :
     BASE2 := BASE2 + AUXILIAR1 :
     BASE := BASE + 3 :
     UNTIL (( VECTOR INDICES [ 1 ] = 0 ) AND ( VECTOR INDICES [
          2 ] = 0 ))
          OR ((J+BASE2) > 7000) OR ((3+BASE) > 7000) :
```

ESTOS PROCEDIMIENTOS MANIPULAN LAS DIFERENCIAS PARA COMPACTAR Y DESCOMPACTAR LA SEÑAL

```
PROCEDURE CAPTURA FONEMA ;FORWARD;

    PROCEDURE GRAFICA MUESTRAS;

    CONST

        COMANDOS : SET OF CHAR = [';','<','?','@','=','>','A'..'D'] ;

    VAR

        I,J,XM,YM,MUESTRA INICIAL,MUESTRAS PANTALLA,DESP Y:INTEGER;
        MUESTRA,M AUX:INTEGER;
        AMPLITUD:REAL;
        COLOR ESCALAS:INTEGER;
        COLOR GRAFICA:INTEGER;
        INC MUESTRAS,INC:REAL;
        A,B:CHAR ;
        TOTAL DE MUESTRAS : INTEGER ;
        M I,D Y           : INTEGER ;
        AMP               : REAL ;
        CURSOR            : ARRAY [1..10] OF INTEGER ;
        DESP X MIN,DESP X MAX,DESP Y MIN,DESP Y MAX : INTEGER ;


    PROCEDURE REGENERA;

    BEGIN
        INC:=1;
        FOR I:=1 TO 8 DO
        BEGIN
            DRAW(1,ROUND(INC),XM,ROUND(INC),COLOR ESCALAS);
            INC:=INC+(YM DIV 8);
        END;
        DRAW(1,YM-1,XM-1,YM-1,COLOR ESCALAS);
        INC:=1;
        FOR I:=1 TO 10 DO
        BEGIN
            DRAW(ROUND(INC),1,ROUND(INC),YM,COLOR ESCALAS);
            INC:=INC+(XM DIV 10);
        END;
        DRAW(XM-1,1,XM-1,YM-1,COLOR ESCALAS);
        INC:=1;
        I:=MUESTRA INICIAL;
        REPEAT
            DRAW(ROUND(INC),(YM-ROUND(VECTOR[I]*AMPLITUD))+DESP Y),
            ROUND(INC+INC MUESTRAS),
            (YM-ROUND(VECTOR[I+1]*AMPLITUD))+DESP Y),
```

```
      COLOR GRAFICA);
      INC:=INC+INC MUESTRAS;
      I := I + 1 ;
UNTIL (I=MUESTRA INICIAL+MUESTRAS PANTALLA) OR
      (I=TOTAL DE MUESTRAS) ;
```

ESTOS PROCEDIMIENTOS CAPTURAN LA SENAL Y DESPLIEGAN PANTALLAS DE COMUNICACION CON EL USUARIO

```
PROCEDURE REPRODUCE;

VAR

     I,J,K,T1:INTEGER;
     A,C:CHAR;
     R:REAL;
     P MUESTRAS:ARRAY[0 .. 255] OF INTEGER;
     P DIFERENCIAS:ARRAY [-255 .. 255] OF INTEGER;
     P TABLAS:ARRAY[0 .. 7] OF INTEGER;
     H MUESTRAS:REAL;
     H DIFERENCIAS:REAL;
     H TABLAS:REAL;
     MAXIMA DIFERENCIA LOCAL:INTEGER;
     TABLA LOCAL:ARRAY[0 .. 7] OF BYTE;
     ACUMULADO LOCAL:INTEGER;
     SIGNO LOCAL:INTEGER;
     SIGNO ANTERIOR:INTEGER;
     DIFERENCIA LOCAL:INTEGER;
     INDICE:INTEGER;


BEGIN
     CLRSCR;
     IF (MUESTRA FINAL-MUESTRA INICIO) > 0 THEN
     BEGIN
          A:='S';
          WRITELN (' REPRODUCCION DE LAS MUESTRAS '
               ):WRITELN:WRITELN;
          REPEAT
               FOR I:=0 TO 255 DO
                    P MUESTRAS[I]:=0;
               FOR I:=0 TO 7 DO
                    P TABLAS[I]:=0;
               FOR I:=-255 TO 255 DO
                    P DIFERENCIAS[I]:=0;
               IF A='S' THEN
               BEGIN
                    WRITE (' CUANTAS REPETICIONES ->
                         [' REPETICIONES.'1 '):
                    READLN (REPETICIONES);
                    WRITELN (REPETICIONES);
               END;
               WRITELN (' ***** REPRODUCCION DE LAS MUESTRAS ***** ');
```

```
FOR J:=1 TO REPETICIONES DO
     FOR I:=MUESTRA INICIO TO MUESTRA FINAL DO
     BEGIN
          PORTC$79C3:=VECTOR CI3:
          FOR T1:=1 TO 2 DO:
          T1:=T1:
          T1:=T1:
          T1:=T1:
          T1:=T1;
     END:
CLRSCR:
WRITELN (' CALCULO DE LA ENTROPIA DE LAS MUESTRAS'):
FOR I:=MUESTRA INICIO TO MUESTRA FINAL DO
     P MUESTRASCVECTORCI33:=P MUESTRASCVECTORCI3 +1:
J:=0:
FOR I:=0 TO 255 DO
J:=J+P MUESTRASCI3:
H MUESTRAS:=0:
FOR I:=0 TO 255 DO
     IF P MUESTRASCI3<>0 THEN
          H MUESTRAS:=H MUESTRAS+
          P MUESTRASCI3/J *
          LN(1/(P MUESTRASCI3/J))/LN(2) :
          WRITELN (' LA ENTROPIA DE LAS MUESTRAS ES
          :', H MUESTRAS):
     REPEAT UNTIL KEYPRESSED:
     CLRSCR:
     WRITELN (' CALCULO DE LA ENTROPIA DE LAS DIFERENCIAS '):
     FOR I:=MUESTRA INICIO TO MUESTRA FINAL - 1 DO
     P DIFERENCIASCVECTORCI+13-VECTORCI33:=
     P DIFERENCIASCVECTORCI+13-VECTORCI33+1:
     J:=0:
     FOR  I:= -255 TO 255 DO
          J:=J+P DIFERENCIASCI3:
     H DIFERENCIAS:=0:
     FOR I:=-255 TO 255 DO
          IF P DIFERENCIASCI3<>0 THEN
               H DIFERENCIAS:=H DIFERENCIAS+
                    P DIFERENCIASCI3/J*
                    LN(1/(P DIFERENCIASCI3/J))/LN(2):
               WRITELN(' LA ENTROPIA DE LAS
                    DIFERENCIAS ES :',
                    H DIFERENCIAS):
```

```
REPEAT UNTIL KEYPRESSED ;
CLRSCR ;
WRITELN(' CALCULO DE LA ENTROPIA DEL METODO DE
MAXIMA PROXIMIDAD '):
MAXIMA DIFERENCIA LOCAL:=0:
FOR I:=MUESTRA INICIO TO MUESTRA FINAL-1 DO
IF ABS(VECTOR[I+1] - VECTOR[
     I])>MAXIMA DIFERENCIA LOCAL THEN
     MAXIMA DIFERENCIA LOCAL :=
          ABS(VECTOR[I+1]-VECTOR[I]):
TABLA LOCAL[0]:=0:
FOR I:=1 TO 7 DO
     IF ROUND ((MAXIMA DIFERENCIA LOCAL DIV 7)*I)<0
     THEN
          TABLA LOCAL[I]:=1
     ELSE
          TABLA LOCAL[I]:=
               ROUND((MAXIMA DIFERENCIA LOCAL
               DIV 7)
               *I):
     ACUMULADO LOCAL:=0:
     SIGNO LOCAL:=1:
     SIGNO ANTERIOR:=1:
     FOR I:=MUESTRA INICIO TO MUESTRA FINAL DO
     BEGIN
          DIFERENCIA LOCAL:=256:
     FOR J:=1 TO 7 DO
          IF DIFERENCIA LOCAL>ABS(TABLA LOCAL[J]-
               ABS(VECTOR[I]-ACUMULADO LOCAL)) THEN
          BEGIN
               DIFERENCIA LOCAL:=ABS(TABLA LOCAL[J]-
               ABS(VECTOR[I]-ACUMULADO LOCAL)):
               INDICE:=J:
          END:
     ACUMULADO LOCAL:=ACUMULADO LOCAL+
          TABLA LOCAL[INDICE]*
     SIGNO LOCAL:
     IF ACUMULADO LOCAL>255 THEN
          ACUMULADO LOCAL:=255:
     IF ACUMULADO LOCAL<0 THEN
          ACUMULADO LOCAL:=0:
     IF (VECTOR[I]-ACUMULADO LOCAL)>=0 THEN
          SIGNO LOCAL:=1
     ELSE
          SIGNO LOCAL:=-1:
```

```
     IF SIGNO ANTERIOR <> SIGNO LOCAL THEN
     BEGIN
          SIGNO ANTERIOR:=SIGNO LOCAL;
          P TABLAS[0]:=P TABLAS[0]+1;
     END;
     P TABLAS[INDICE]:=P TABLAS[INDICE]+1;
END;
H TABLAS:=0;
J:=0;
FOR I:=0 TO 7 DO
     J:=J+P TABLAS[I];
WRITELN (' J= ',J);
FOR I:=0 TO 7 DO
     WRITELN('I= '.I.' >> '.P TABLAS[I]);
FOR I:=0 TO 7 DO
     IF P TABLAS[I]<>0 THEN
          H TABLAS:=H TABLAS+
               P TABLAS[I]/J*LN(1/(P TABLAS[I]/J)
               )/LN(2);
          WRITELN(' LA ENTROPIA DEL METODO DE
               MAXIMA PROXIMIDAD ES :',H TABLAS);
          REPEAT UNTIL KEYPRESSED;
          CLRSCR;
          REPEAT
               WRITE('DESEAS CAMBIAR EL NUMERO DE
                    REPETICIONES (S/N) - > [N] ');
               READLN(A) ; WRITELN ; WRITELN ;
                    WRITELN ;
               IF A=CHAR(26) THEN A:= 'N' ;
          UNTIL (A='N') OR (A='S') ;
          REPEAT
               WRITE('DESEAS CAMBIAR LOS LIMITES
                    (S/N) - > [N] ');
               READLN(C) ; WRITELN ; WRITELN ;
                    WRITELN ;
               IF C=CHAR(26) THEN C:='N' ;
          UNTIL (C='N') OR (C='S') ;
          IF C = 'S' THEN
          BEGIN
               WRITE ('MUESTRA INICIAL ->
                    [',MUESTRA INICIO,']');
               READLN (MUESTRA INICIO); WRITELN ;
               WRITE ('MUESTRA FINAL ->
                    [',MUESTRA FINAL,']');
               READLN (MUESTRA FINAL); WRITELN ;
          END ;
          CLRSCR ;
```

```
      UNTIL (A='N') AND (C='N');
      COMPACTA ( MUESTRA INICIO , MUESTRA FINAL );
      DESCOMPACTA DIFERENCIAS ;
      WRITE (' ***** REPRODUCCION DE LAS MUESTRAS
            DESCOMPACTADAS ***** ') ;
      K:=MUESTRA FINAL - MUESTRA INICIO + 1 ;
      FOR J := 1 TO REPETICIONES DO
            FOR I:= 1 TO K DO
            BEGIN
                  PORTC$79C]:=VECTOR DESCOMPACTACION
                        CI];
                  FOR T1:=1 TO 2 DO ;
                  T1:=T1;
                  T1:=T1;
                  T1:=T1;
                  T1:=T1;
            END ;
      WRITELN ('<RETURN>') ;
      REPEAT UNTIL KEYPRESSED ;
      PROCESO COMPLETO := TRUE ;
      GRAPHCOLORMODE ;
      GRAPHWINDOW(1,1,319,199) ;
      I := 1 ;  R := 0.35 ;   INC := 1  ;
      REPEAT
            DRAW(ROUND(INC)
            ,(99-ROUND(VECTORCI+
            MUESTRA INICIO-1)*R)),
            ROUND(INC+2),
                  (99-ROUND(VECTORCI
                  +MUESTRA INICIO])*R)),
                  3)];
            DRAW(ROUND(INC),
                  (199-ROUND(VECTOR DESCOMPACTACIONCI]
                  *R)),
            ROUND(INC+2),
                  (199-ROUND(VECTOR DESCOMPACTACION
                  CI+1]
                  *R)),
                  3)];
             INC := INC + 2 ;   I:= I + 1 ;
      UNTIL (I=159) OR (I=K+1) ;
      GOTOXY (21,22) ; WRITELN ('<RETURN>') ;
      REPEAT UNTIL KEYPRESSED ;
END
ELSE
```

```
BEGIN
     WRITELN ('*** ERROR LA LONGITUD NO PUEDE SER
          CERO <RETURN>'.CHAR(7));
     REPEAT UNTIL KEYPRESSED ;
END ;
GRAPHCOLORMODE ;
GRAPHWINDOW(1,1,XM,YM) ;
B := 'D';
```

```
PROCEDURE LISTA PARAMETROS;

BEGIN
     TEXTMODE(BW80) ; WRITELN ; WRITELN ;
     WRITELN('***** LISTADO DE PARAMETROS *****');
     WRITELN ; WRITELN ; WRITELN ;
     WRITELN('AMPLITUD = '.AMPLITUD) ; WRITELN ;
     WRITELN('DESPLAZAMIENTO EN EL EJE VERTICAL =
          '.DESP_Y);WRITELN ;
     WRITELN('MUESTRA INICIAL DESPLEGADA =
          '.MUESTRA_INICIAL);WRITELN ;
     WRITELN('MUESTRA FINAL EN PANTALLA =
          '.MUESTRA_INICIAL+MUESTRAS_PANTALLA);WRITELN ;
     WRITELN('NUMERO DE MUESTRAS EN PANTALLA =
          '.MUESTRAS_PANTALLA) ;
     WRITELN ;
     WRITELN('LIMITE
     MAXIMO DE LA PANTALLA ('.XM.','.YM.')');WRITELN ;
     WRITELN('<RETURN>') ;
     REPEAT UNTIL KEYPRESSED ;
     GRAPHCOLORMODE ;
     B := 'D' ;
END;


PROCEDURE AJUSTE AMPLITUD;

BEGIN
     REPEAT
          REPEAT
               GOTOXY(1,21);
               WRITELN;
               WRITELN;
               WRITELN ('                                                  ');
               GOTOXY(1,21);
               WRITELN('[F1]+0.01  [F2]-.01  [F5]+.5    [F6]-.5 ');
               WRITELN('[F3]+0.10  [F4]-.10  [F7]+ 1    [F8]- 1 ');
               WRITELN('[F9]        [F10] FIN AMPLITUD = '.AMPLITUD:7:2);
               READ(KBD,B,B);
          UNTIL B IN COMANDOS ;
          CASE B OF
               ':' : AMPLITUD := AMPLITUD + 0.01;

               '<' : AMPLITUD := AMPLITUD - 0.01;

               '=' : AMPLITUD := AMPLITUD + 0.1;

               '>' : AMPLITUD := AMPLITUD - 0.1;
```

```
                              BEGIN
                                   CAR:=PALABRA[ULTIMA LETRA
                                        CORRECTA+AUX+1];
                                   TABLA FONEMAS[TOTAL
                                        DE FONEMAS].
                                        REGLAS[1]:=CAR;
                              END;
                              FOR I:=1 TO LONG MAXIMA DO
                                   IF TABLA FONEMAS[TOTAL DE
                                        FONEMAS].FONEMA[I]<>''
                                        THEN
                                             ULTIMA LETRA CORRECTA
                                                  :=ULTIMA LETRA
                                                   CORRECTA
                                                  + 1 ;
                                        ULTIMA LETRA ANALIZADA
                                             := ULTIMA LETRA
                                             CORRECTA + 1 ;
                                   END ;
     UNTIL FONEMA PERTENECE A PALABRA ;
     FONEMA:=TABLA FONEMAS[TOTAL DE FONEMAS].FONEMA ;
     REPEAT
          CAPTURA FONEMA ;
          GRAFICA MUESTRAS;
          INICIA PANTALLA ;
          REPEAT
               WRITE(' ESTAS DE ACUERDO CON EL PROCESO (S/N) -> [S] ') ;
               READLN( PREGUNTA ) ;
               IF PREGUNTA = CHAR(26) THEN
               PREGUNTA:='S' ;
          UNTIL (PREGUNTA='S') OR (PREGUNTA='N') ;
     UNTIL PREGUNTA='S' ;
     TABLA FONEMAS [ TOTAL DE FONEMAS ].REPETICIONES :=
          REPETICIONES  ;
     ALMACENA ;
     TABLA FONEMAS [ TOTAL DE FONEMAS ].BLOQUE INICIAL :=
          BLOQUE INICIAL ;
     DESCOMPACTA DIFERENCIAS ;
     TABLA FONEMAS [ TOTAL DE FONEMAS ].LONG FONEMA :=
          MUESTRA FINAL - MUESTRA INICIO ;
     CONCATENA ( TABLA FONEMAS [TOTAL DE FONEMAS].LONG FONEMA
          ) ;
     INICIA PANTALLA ;
END;
```

```
BEGIN
     ULTIMA LETRA ANALIZADA := 1 ;
     ULTIMA LETRA CORRECTA := 0 ;
     NUMERO DE FONEMAS ENCONTRADOS := 1 ;
     REPEAT
          FONEMA TEMPORAL := -1 ;
          FONEMA LOCALIZADO := FALSE ;
          NUEVA REGLA := FALSE ;
          F := 1 ;
          FONEMA:='       ';
          LONGITUD DE FONEMA := LONG MAXIMA;
          REPEAT
               FONEMA[F] := PALABRA[ULTIMA LETRA ANALIZADA] ;
               BUSCA FONEMA ;
               F := F + 1 ;
               ULTIMA LETRA ANALIZADA := ULTIMA LETRA ANALIZADA + 1 ;
          UNTIL (FONEMA LOCALIZADO AND REGLA LOCALIZADA) OR
               (F>=(LONG MAXIMA+1)) OR
               (PALABRA[ULTIMA LETRA ANALIZADA]=' ');
           IF (FONEMA TEMPORAL <> -1) THEN
          BEGIN
               I:=FONEMA TEMPORAL ;
               BORRA MENSAJES;
               WRITE(' NO SE ENCONTRO LA REGLA DEL FONEMA '',
                    TABLA FONEMASET-1].FONEMA,'' POR TANTO SE',
                    ' ASUME LA REGLA [');
               IF (ULT LET ANAL TEMP+1) > 14 THEN
                    WRITELN(' ]')
               ELSE
                    WRITELN(PALABRA[ULT LET ANAL TEMP + 1], ']');
               REPEAT
                    WRITE('ESTAS DE ACUERDO S/N ->');
                    READLN(ALFA);
               UNTIL  (ALFA='S') OR (ALFA='N');
               IF ALFA = 'S' THEN
               BEGIN
                    NUEVA REGLA := TRUE ;
                    FONEMA LOCALIZADO := TRUE ;
                    I:=0;
                    REPEAT
                         I:=I+1;
                    UNTIL (TABLA FONEMASET-1].REGLASCI]=CHAR(0))
                         OR (I=20);
                    IF (I<=20) AND (TABLA FONEMASET-1].REGLASCI]
                         = CHAR(0))
                         THEN
                         BEGIN
```

```
      IF (ULT LET ANAL TEMP+1)>14
            THEN
                  CAR:=CHAR(32)
      ELSE
      BEGIN
            CAR:= PALABRA
                  [ULT LET ANAL TEMP+1];
                  TABLA FONEMASET-1]
                  .REGLASCI] := CAR ;
      END;
      ASSIGN(TABLAS.NOMBRE TABLAS);
      RESET(TABLAS);
      SEEK(TABLAS.(T-2));
      WRITE(TABLAS.TABLA FONEMASET-1]);
      BORRA MENSAJES;
      WRITELN('*** SE ACTUALIZO
            LA REGLA EN DISCO
            <RETURN>'.CHAR(7));
      WRITELN('******* POS APUNT
            = '.FILEPOS(TABLAS));
      WRITELN('******* LONGITUD
            = '.FILESIZE(TABLAS));
      FLUSH(TABLAS);
      CLOSE(TABLAS);
      REPEAT UNTIL KEYPRESSED;
END
ELSE
BEGIN
      BORRA MENSAJES;
      WRITELN('*** SE EXCEDIO LA
            CAPACIDAD'.
            ' DE ALMACENAMIENTO DE REGLAS
            '.CHAR(7)) ;
END ;
ULTIMA LETRA CORRECTA
      :=ULT LET CORR TEMP ;
ULTIMA LETRA ANALIZADA
      :=ULT LET ANAL TEMP ;
BORRA MENSAJES ;
WRITE('EL FONEMA LOCALIZADO ES : '.
TABLA FONEMASET-1].FONEMA.'
      CON LA REGLA [') ;
IF (ULT LET ANAL TEMP+1) > 14 THEN
      WRITELN(' ]')
ELSE
      WRITELN(PALABRA
      [ULT LET ANAL TEMP + 1]. ']') ;
WRITE(' <RETURN> ');
```

```
                    REPEAT
                    UNTIL KEYPRESSED ;
          END
          ELSE
               FONEMA LOCALIZADO:=FALSE ;
     END;
     IF (NOT ( FONEMA LOCALIZADO ) ) THEN
     BEGIN
          PREGUNTA SEPARACION FONETICA;
          SALVA TABLAS;
     END
     ELSE
     IF NOT(REGLA LOCALIZADA) THEN
     BEGIN
          FONEMA LOCALIZADO:=FALSE ;
          AUX := 0;
          FOR I := 1 TO LONG MAXIMA DO
               IF TABLA FONEMAS
                    [FONEMA TEMPORAL-1].FONEMA[I]
                    <>CHAR(32) THEN
                         AUX := AUX+1;
                         ULTIMA LETRA CORRECTA
                              := ULT LET CORR TEMP +
                              AUX;
                         ULTIMA LETRA ANALIZADA
                              := ULTIMA LETRA CORRECTA
                              + 1 ;
     END
     ELSE
     BEGIN
          ULTIMA LETRA CORRECTA := ULTIMA LETRA ANALIZADA - 1 ;
     END;
     IF (FONEMA LOCALIZADO
          AND REGLA LOCALIZADA) OR NUEVA REGLA
          THEN
          BEGIN
               LEE ( TABLA FONEMAS [ T - 1 ].BLOQUES,
               TABLA FONEMAS [ T - 1 ].LONG FONEMA,
               TABLA FONEMAS [ T - 1 ].BLOQUE INICIAL,
               TABLA FONEMAS [ T - 1 ].REPETICIONES ) ;
               DESCOMPACTA DIFERENCIAS ;
               CONCATENA ( TABLA FONEMAS [ T - 1
               ].LONG FONEMA ) ;
          END ;
          VECTOR FONEMAS[NUMERO DE FONEMAS ENCONTRADOS]
               := FONEMA ;
          NUMERO DE FONEMAS ENCONTRADOS
               :=NUMERO DE FONEMAS ENCONTRADOS + 1;
UNTIL (ULTIMA LETRA ANALIZADA>=15)
     OR (PALABRA[ULTIMA LETRA ANALIZADA]=' ');
```

```
 PROCEDURE INICIALIZA TABLA DE FONEMAS ;

VAR
     T : INTEGER ;

BEGIN
     FOR T := 1 TO 720 DO
     BEGIN
          FOR I:= 1 TO LONG MAXIMA DO
          TABLA FONEMAS[T].FONEMA[I] := ' ' ;
          FOR I:=1 TO 20 DO
               TABLA FONEMAS[T].REGLAS[I] := CHAR(0) ;
     END;
END ;



PROCEDURE DESPLIEGA SEPARACION FONETICA ENCONTRADA;

BEGIN
     BORRA MENSAJES;
     WRITELN('SEPARACION FONETICA ENCONTRADA');
     FOR I:=1 TO NUMERO DE FONEMAS ENCONTRADOS-1 DO
     BEGIN
          WRITE(VECTOR FONEMAS[I]);
          IF I<>NUMERO DE FONEMAS ENCONTRADOS-1 THEN
               WRITE('-');
     END;
     WRITE('<RETURN>');
     REPEAT
     UNTIL KEYPRESSED;
END;


PROCEDURE REPRODUCE PALABRA ;

VAR

     T1 : INTEGER ;
     A  : CHAR ;

BEGIN
     REPEAT
     FOR I:= 1 TO APUNT DO
     BEGIN
          PORT[$79C]:=VECTOR REPRODUCCION [I];
          FOR T1:=1 TO 2 DO ;
          T1:=T1;
```

```
        T1:=T1:
        T1:=T1:
        T1:=T1:
    END:
        WRITE ('DESEAS REPETIR LA PALABRA (S/N) ->'):
        READLN (A):
    UNTIL A='N':
    APUNT := 0 :
```

ESTOS PROCEDURES CAPTURAN EL FONEMA, RELACIONAN LA PARTE A ANALIZAR CON EL FONEMA GRABADO, SE GENERAN TABLAS DE SILABAS Y DE FONEMAS GRABADOS. QUE SE PUEDEN BORRAR O ACTUALIZAR

```
PROCEDURE CREA TEXTO ;

BEGIN
     WRITELN ('              *** CREACION DE TEXTO ***');
     WRITELN ;
     WRITELN (' TECLEE UN RENGLON COMPLETO (80 CARACTERES) Y
          <RETURN> ');
     WRITELN ;
     WRITELN (' PARA TERMINAR LA CAPTURA TECLEE COMO PRIMEROS
          SIMBOLOS  *F* ');
     GOTOXY(1,7) ;
     REPEAT
          READLN (RENGLON) ;
          IF (RENGLON [1] <> '*') AND (RENGLON [2] <> 'F')
               AND (RENGLON [3] <> '*') THEN
          WRITELN(TEXTO,RENGLON);
     UNTIL (RENGLON [1] = '*') AND (RENGLON [2] = 'F') AND
          (RENGLON [3] = '*') ;
```

ESTE PROCEDIMIENTO CREA UN ARCHIVO DE TEXTO QUE PUEDE SER EDITADO MEDIANTE UN PROCESADOR DE TEXTO. ADEMAS QUE ESTE ES ANALIZADO POR RENGLONES, PALABRAS Y SILABAS

```
PROGRAMA PRINCIPAL DEL SISTEMA FONETICO

BEGIN
     LONG BLOQUE    := 40 ;
     APUNT          :=  0 ;
     PALABRA        := '                    ' ;
     FOR I := 1 TO 80 DO
          RENGLON [ I ] := ' ' ;
     REPETICIONES        :=     1 ;
     NO MUESTRAS         := 7000 ;
     REPEAT
          INICIA PANTALLA;
          REPEAT
               A TEX := TRUE ;
               BORRA MENSAJES ;
               WRITE(' DAME EL NOMBRE DEL ARCHIVO DE TEXTO ->
                    ');
               READLN(NOMBRE);
               ASSIGN (TEXTO,NOMBRE);
               {$I-} RESET (TEXTO) {$I+} ;
               OK := (IORESULT = 0 ) ;
               IF NOT OK THEN
               BEGIN
                    BORRA MENSAJES ;
                    WRITELN (' *** EL ARCHIVO '.NOMBRE.' NO
                         EXISTE *** <RETURN> '.CHAR(7));
                    REPEAT UNTIL KEYPRESSED ;
                    REPEAT
                         BORRA MENSAJES ;
                         WRITE (' DESEAS CREAR EL ARCHIVO
                              '.NOMBRE.' (S/N) -> ');
                         READLN (ALFA) ;
                    UNTIL (ALFA = 'S') OR (ALFA = 'N') ;
                    IF ALFA = 'S' THEN
                    BEGIN
                         REWRITE(TEXTO);
                         CLRSCR ;
                         CREA TEXTO ;
                         A TEX := FALSE ;
                         OK := TRUE ;
                         INICIA PANTALLA ;
                    END;
               END ;
          UNTIL OK ;
          REPEAT
               A FON := TRUE ;
               BORRA MENSAJES ;
               WRITE(' DAME EL NOMBRE DEL ARCHIVO DE FONEMAS
                    -> ') ;
```

```
          READLN(NOMBRE_ARCHIVO_FONEMAS) ;
          ASSIGN (FONEMAS,NOMBRE_ARCHIVO_FONEMAS);
          {$I-} RESET (FONEMAS) {$I+} ;
          OK := (IORESULT = 0 ) ;
          IF NOT OK THEN
          BEGIN
               BORRA_MENSAJES ;
          WRITELN (' *** EL ARCHIVO
               ',NOMBRE_ARCHIVO_FONEMAS,
               ' NO EXISTE *** <RETURN> '.CHAR(7));
          REPEAT UNTIL KEYPRESSED ;
          REPEAT
               BORRA_MENSAJES ;
               WRITE (' DESEAS CREAR EL ARCHIVO
                    ',NOMBRE_ARCHIVO_FONEMAS
                    ,' (S/N) -> ');
               READLN (ALFA) ;
          UNTIL (ALFA = 'S') OR (ALFA = 'N') ;
          IF ALFA = 'S' THEN
               BEGIN
                    REWRITE(FONEMAS);
                    A_FON := FALSE ;
                    OK := TRUE ;
               END;
          END ;
UNTIL OK ;
REPEAT
     A_TAB := TRUE ;
     BORRA_MENSAJES;
     WRITE('DAME EL NOMBRE DEL ARCHIVO DE TABLAS -> ') ;
     READLN(NOMBRE_TABLAS) ;
     ASSIGN (TABLAS,NOMBRE_TABLAS) ;
     {$I-} RESET (TABLAS) {$I+} ;
     OK := (IORESULT = 0 ) ;
     IF NOT OK THEN
     BEGIN
          BORRA_MENSAJES ;
          WRITELN (' *** EL ARCHIVO '.NOMBRE_TABLAS,
               ' NO EXISTE *** <RETURN>'.CHAR(7));
          REPEAT UNTIL KEYPRESSED ;
          REPEAT
               BORRA_MENSAJES ;
               WRITE (' DESEAS CREAR EL ARCHIVO
                    '.NOMBRE_TABLAS,
                    ' (S/N) -> ');
               READLN (ALFA) ;
          UNTIL (ALFA = 'S') OR (ALFA = 'N') ;
```

```
          IF ALFA = 'S' THEN
               BEGIN
                    REWRITE(TABLAS) ;
                    A TAB := FALSE :
                    OK := TRUE :
               END;
     END :
UNTIL OK :
CLOSE (TEXTO) :
CLOSE (FONEMAS) :
CLOSE (TABLAS) :
INICIALIZA TABLA DE FONEMAS :
ABRE ARCHIVO(NOMBRE):
IF A FON AND A TAB THEN
BEGIN
     REPEAT
          BORRA MENSAJES:
          WRITE('DESEAS CARGAR TABLAS ANTERIORES (S/N)
          -> '):
          READLN(TABLAS ANTERIORES):
     UNTIL (TABLAS ANTERIORES='S') OR
          (TABLAS ANTERIORES='N') :
     IF TABLAS ANTERIORES = 'S' THEN
     BEGIN
          LEE TABLAS :
          BORRA MENSAJES :
          WRITE('DESEAS EL LISTADO DE LAS TABLAS LEIDAS
               (S/N) -> ') :
          READLN(ALFA) :
          IF ALFA='S' THEN
          BEGIN
               FOR J := 1 TO TOTAL DE FONEMAS DO
               BEGIN
                    WRITE(TABLA FONEMAS[J].FONEMA,'+'):
                    FOR I := 1 TO 20 DO
                         WRITE(TABLA FONEMAS
                              [J].REGLAS[I],':',
                         INTEGER(TABLA FONEMAS[J]
                              .REGLAS[I]),':'):
                         WRITELN:
                         WRITELN     ('     BLOQUE INICIAL     =       ',
                         TABLA FONEMAS[J].BLOQUE INICIAL
                              ,
                              '          LONG FONEMA          =       ',
                         TABLA FONEMAS[J].LONG FONEMA ,
                              '                    REPETICIONES          =
                              ',TABLA FONEMAS[J].
                              REPETICIONES ,
                         ' BLOQUES = ',TABLA FONEMAS[J].BLOQUES) :
```

```
                IF ((J MOD 5) = 0) THEN
                  BEGIN
                    WRITELN;
                    WRITELN('<RETURN>');
                    REPEAT
                    UNTIL KEYPRESSED;
                  END ;
                END;
                WRITE(' <RETURN> ');
                REPEAT
                UNTIL KEYPRESSED;
                INICIA PANTALLA;
           END;
     END
ELSE
TOTAL DE FONEMAS := 0 ;
IF TABLAS ANTERIORES='N' THEN
BEGIN
     REPEAT
          BORRA MENSAJES;
          WRITE('DESEAS BORRAR EL ARCHIVO
               ANTERIOR (S/N) -> ');
          READLN(BORRAR ARCHIVO);
     UNTIL (BORRAR ARCHIVO='S') OR (BORRAR ARCHIVO='N')
          OR (BORRAR ARCHIVO=' ');
     IF BORRAR ARCHIVO='S' THEN
     BEGIN
     REPEAT ;
          BORRA MENSAJES ;
          WRITELN (' *** DESEAS CONTINUAR CON EL
               PROCESO' DE BORRADO ') ;
          WRITE ('      DE LOS ARCHIVOS '.NOMBRE TABLAS.'
               Y '. NOMBRE ARCHIVO FONEMAS.' (S/N) ->
               ');
          READLN (BORRAR ARCHIVO);
     UNTIL (BORRAR ARCHIVO='S') OR (BORRAR ARCHIVO='N')
          ;
     IF BORRAR ARCHIVO='S' THEN
     BEGIN
          BORRA MENSAJES ;
          WRITELN ('*** BORRADO DEL ARCHIVO DE TABLAS EN
          PROCESO '.CHAR(7) );
          ASSIGN(TABLAS,NOMBRE TABLAS);
          ERASE(TABLAS);
          CLOSE(TABLAS);
          ASSIGN(TABLAS,NOMBRE TABLAS);
          REWRITE(TABLAS);
```

```
                FLUSH(TABLAS);
                CLOSE(TABLAS);
                WRITELN ('*** BORRADO DEL ARCHIVO DE FONEMAS
                    ' EN PROCESO '.CHAR(7) );
                ASSIGN(FONEMAS,NOMBRE_ARCHIVO_FONEMAS);
                ERASE(FONEMAS);
                CLOSE(FONEMAS);
                ASSIGN(FONEMAS,NOMBRE_ARCHIVO_FONEMAS);
                REWRITE(FONEMAS);
                FLUSH(FONEMAS);
                CLOSE(FONEMAS);
            END;
    END;
END;
END;
IF (A_FON AND (NOT A_TAB)) OR (A_TAB AND (NOT A_FON)) THEN
BEGIN
     BORRA_MENSAJES ;
     WRITELN (' SELECCION INVALIDA. EL ARCHIVO DE FONEMAS Y
          EL DE TABLAS
          ');
     WRITELN (' DEBEN SER CREADOS Y/O DESTRUIDOS
          SIMULTANEAMENTE <RETURN>'.CHAR(7));
     REPEAT UNTIL KEYPRESSED ;
END ;
UNTIL (A_FON AND A_TAB) OR ( (NOT (A_FON)) AND (NOT (A_TAB)))
    ;
FIN_PROGRAMA:=' ';
REPEAT
     FOR I := 1 TO 80 DO
          RENGLON[I] := ' ' ;
     LEE_RENGLON ;
     FIN_RENGLON := FALSE ;
     NO_LETRA := 1 ;
     REPEAT
          PALABRA := '                    ' ;
          FOR I:=1 TO 10 DO
               VECTOR_FONEMAS[I]:='       '; (* LONG MAXIMA*)
          LEE_PALABRA ;
          BORRA_RASTRED;
          IF FIN_PROGRAMA<>'F' THEN
          BEGIN
               INTERPRETA_FONEMA ;
               DESPLIEGA_SEPARACION_FONETICA_ENCONTRADA ;
               REPRODUCE_PALABRA ;
          END;
     UNTIL (FIN_RENGLON) OR (FIN_PROGRAMA='F');
UNTIL EOF(TEXTO) OR (FIN_PROGRAMA='F');
CLRSCR;
CLOSE (TEXTO) ;
```

```
                '?' : AMPLITUD := AMPLITUD + 0.5;
                '@' : AMPLITUD := AMPLITUD - 0.5;
                'A' : AMPLITUD := AMPLITUD + 1 ;
                'B' : AMPLITUD := AMPLITUD - 1 ;
            END;
        UNTIL B='D';
    END;


    PROCEDURE MUESTRAS EN PANTALLA;

    BEGIN
        REPEAT
            TEXTMODE(BW80);
            GOTOXY(10,10);
            WRITE('***** DAME EL INCREMENTO ENTRE LAS MUESTRAS
                EN PANTALLA [',
            INC MUESTRAS,']  (100.0.01) -> ');
            READLN(INC MUESTRAS);
        UNTIL (INC MUESTRAS<=100) AND (INC MUESTRAS>=0.01) ;
        B:='D';
        GRAPHCOLORMODE;
```

ESTAS RUTINAS NOS PERMITEN VARIAS OPCIONES DE PANTALLAS CON LAS QUE SE PUEDAN MANIPULAR LOS DIFERENTES PROCESOS DE CAPTURA, GRAFICACION DE LA SEÑAL, ALMACENAMIENTO Y REPRODUCCION

```
PROCEDURE CORTE DE MUESTRAS;

BEGIN
     MUESTRA:= 1 ;
     REPEAT
          INC:=1;
          FOR I:= MUESTRA INICIAL TO (MUESTRA INICIAL +
               MUESTRA - 1 ) DO
          INC:=INC+INC MUESTRAS;
          FOR I:= 1 TO 10 DO
          PLOT(ROUND(INC),
               (YM-ROUND(VECTOR[MUESTRA INICIAL
               +MUESTRA]*AMPLITUD)+
          DESP Y+1-5),3);
          M AUX:=MUESTRA;
          IF MUESTRA INICIO>MUESTRA FINAL THEN
          BEGIN
               I:=MUESTRA INICIO;
               MUESTRA INICIO:=MUESTRA FINAL;
               MUESTRA FINAL:=I;
          END;
          GOTOXY(1,21);
          WRITELN;
          WRITELN;
          WRITELN;
          WRITELN('                                          ');
          REPEAT
               GOTOXY(1,21);
               WRITELN('[F1] INICIO CORTE     [F2] FIN CORTE     ');
               WRITELN('[F3] <  [F4] >   [F5] <<     [F6] >>    ');
               WRITELN('[F7] REPRODUCIR  [F8] REGEN  [F10] FIN  ');
               WRITELN('INI=',MUESTRA INICIO:4,' FIN=',
                    MUESTRA FINAL:4,' ACT=',MUESTRA +
                    MUESTRA INICIAL:4,' M I P=',MUESTRA INICIAL:4);
               READ(KBD,B,B);
          UNTIL B IN COMANDOS ;
          CASE B OF
               ':' : MUESTRA INICIO:=MUESTRA + MUESTRA INICIAL  ;

               '<' : MUESTRA FINAL:=MUESTRA + MUESTRA INICIAL ;

               '=' : BEGIN

                         IF MUESTRA-1>=0  THEN
                              MUESTRA:=MUESTRA
                                   -ROUND(1/INC MUESTRAS)
                         ELSE
                         BEGIN
                              GOTOXY(1,21);
                              WRITELN(CHAR(7));
                         END;
                    END;
```

```
      '>' : BEGIN
              IF MUESTRA-1<0 OR
                   (MUESTRA+1+MUESTRA INICIAL
                   <=TOTAL DE MUESTRAS) THEN
                        MUESTRA:=MUESTRA+
                             ROUND(1/INC MUESTRAS)
              ELSE
              BEGIN
                   GOTOXY(1,21);
                   WRITELN(CHAR(7));
              END;
            END;


       '?' : BEGIN
              IF MUESTRA-10>=0  THEN
                   MUESTRA:=MUESTRA-
                        ROUND(10/INC MUESTRAS)
              ELSE
              BEGIN
                   GOTOXY(1,21);
                   WRITELN(CHAR(7));
              END;
            END;

      '@' : BEGIN
              IF (MUESTRA+10<=MUESTRAS PANTALLA) AND
                   (MUESTRA+10+MUESTRA INICIAL
                   <=TOTAL DE MUESTRAS) THEN
                        MUESTRA:=MUESTRA+
                         ROUND(10/INC MUESTRAS)
              ELSE
              BEGIN
                   GOTOXY(1,21);
                   WRITELN(CHAR(7));
              END;
            END;

      'A' : REPRODUCE;

      'B' : REGENERA ;

END;

FOR I:= 1 TO 10 DO

     PLOT(ROUND(INC),
          (YM-ROUND(VECTORDEMUESTRA INICIAL+M AUX]
          *AMPLITUD)+ DESP Y[ 5),0);
```

```
    UNTIL B='D';
    INICIO FONEMA := MUESTRA INICIO ;
    FIN FONEMA := MUESTRA FINAL ;
END;


PROCEDURE LIMITE PANTALLA;

BEGIN
    REPEAT
        TEXTMODE(BW80);
        GOTOXY(10,10);
        WRITE('***** DAME LOS LIMITES MAXIMOS
            DE LA PANTALLA C',XM,',',YM,'] =>');
        READLN(XM,YM);
    UNTIL (XM>0) AND (YM>0) AND (XM<320) AND (YM<200);
    GRAPHCOLORMODE;
    B:='D';
```

ESTAS RUTINAS PERMITEN HACER CORTES DE ANALISIS DE LA SEÑAL PARA ALMACENAR ESTA EN UN ESPACIO ACEPTABLE DE MEMORIA Y QUE AL REPRODUCIRLA SIGA SIENDO LA MISMA SEÑAL

```
PROCEDURE INICIA;

BEGIN
     XM                  :=319;
     YM                  :=149;
     AMPLITUD            :=0.3;
     DESP Y              :=-30;
     MUESTRA INICIAL     :=  1;
     INC MUESTRAS :=0.05; (* 0.0455714 PARA 7000 MUESTRAS *)
     GRAPHCOLORMODE;
     GRAPHWINDOW(1,1,XM,YM);
     PROCESO COMPLETO :=FALSE;
     DESP X MIN          :=100;
     DESP X MAX          :=1000;
     DESP Y MIN          :=1;
     DESP Y MAX          :=10;
END;


PROCEDURE INICIALIZA PARAMETROS;

BEGIN
     REPEAT
          TEXTMODE(BW80);
          GOTOXY(7,10);
          WRITE('***** DESEAS CAMBIAR LOS
               PARAMETROS ACTUALES POR LOS ORIGINALES
               (S/N) =>');
          READLN(R);
     UNTIL (R='S') OR (R='N');
     IF R='S' THEN
          INICIA
     ELSE
     BEGIN
          GRAPHCOLORMODE ;
          GRAPHWINDOW(1,1,XM,YM) ;
     END ;
     R:='N' ;
END;


PROCEDURE DESPLAZAMIENTOS;

BEGIN
     TEXTMODE(BW80) ; GOTOXY (7,10) ;
     WRITELN('***** CAMBIO DE LOS
          PARAMETROS DE DESPLAZAMIENTO *****');
     WRITELN ;WRITELN ;WRITELN ;WRITELN ;
```

```
          REPEAT
               WRITE(' DESPLAZAMILNTO MINIMO
                    EN X -> [',DESP X MIN,']: ');
               READLN(DESP X MIN); WRITELN (DESP X MIN) ;
          UNTIL DESP X MIN>0 ;
          REPEAT
               WRITELN DESPLAZAMIENTO MAXIMO
                    EN X -> [',DESP X MAX,']: ');
               READLN(DESP X MAX); WRITELN (DESP X MAX) ;
          UNTIL DESP X MAX>0 ;
          REPEAT
               WRITE(' DESPLAZAMIENTO MINIMO
                    EN Y -> [',DESP Y MIN,']: ');
               READLN(DESP Y MIN); WRITELN (DESP Y MIN) ;
          UNTIL DESP Y MIN>0 ;
          REPEAT
     END;
               WRITE(' DESPLAZAMIENTO MAXIMO
                    EN Y -> [',DESP Y MAX,']: ');
END;
END;
               READLN(DESP Y MAX); WRITELN (DESP Y MAX) ;
IF (A FON AND (NOT A TAB)) OR (A TAB AND (NOT A FON)) THEN
          UNTIL DESP Y MAX>0 ;
BEGIN
     GRAPHCOLORMODE;
     BORRA_DESPACHO;
     WRITELN (' SELECCION INVALIDA. EL ARCHIVO DE FONEMAS Y
     END;
          EL DE TABLAS
          ');
     WRITELN (' DEBEN SER CREADOS Y/O DESTRUIDOS
          SIMULTANEAMENTE <RETURN>',CHAR(7));
     REPEAT UNTIL KEYPRESSED ;
END ;
     BEGIN
UNTIL (A FON AND A TAB) OR ( (NOT (A FON)) AND (NOT (A TAB)))
     REPEAT
FIN PROGRAMA:=INICIA:
     A:=' ';
REPEAT
     FOR I := TOTAL DE MUESTRAS :=  NO MUESTRAS :
          COLOR GRAFICA:=7; (* LIGHTGRAY *)
          COLOR ESCALAS:=3;(* CYAN *)
     LEE RENGLON;
     FIN RENGLON:=
     NO LEYO A:=
     REPEAT
          MUESTRAS PANTALLA := ROUND(XM/INC MUESTRAS) ;
          (* GRAFICACION DE LA SEÑAL *)
          GRAPHCOLORMODE;
          LEE PALABRA;
          BORRA_DESPACHO;
          IF FIN PROGRAMA='F' THEN
          BEGIN
               WRITELN;
               WRITELN('META FONEMA ;
               WRITELN;
               WRITELN('DESPLIEGA SEPARACION FONETICA ENCONTRADA ;
               REPEAT
               REPRODUCE PALABRA ;
          END;
     UNTIL (FIN RENGLON) OR (FIN PROGRAMA='F');
UNTIL EOF(TEXTO) OR (FIN PROGRAMA='F');
```

```
      WRITELN('[F1] <-'   [F2]  ->   [F5] ^    [F6]
         v '):
      WRITELN('[F3] <<-   [F4] ->>   [F7] ^^   [F8]
         vv'):
      WRITELN('[F9] ++   [F10] FIN
            '):
      READ(KBD,A,A):
UNTIL (A IN COMANDOS) :
A 1:=MUESTRA INICIAL:
D Y:=DESP Y:
AMP:=AMPLITUD:

CASE A OF

      ':' :       IF (MUESTRA INICIAL + DESP X MIN)
                     <=
                     (TOTAL DE MUESTRAS-
                     MUESTRAS PANTALLA) THEN
                          MUESTRA INICIAL
                               := MUESTRA INICIAL +
                               DESP X MIN
                  ELSE
                       WRITELN(CHAR(7)):

      '<' :       IF (MUESTRA INICIAL - DESP X MIN)
                     >= 1 THEN
                          MUESTRA INICIAL :=
                               MUESTRA INICIAL -
                               DESP X MIN
                  ELSE
                       WRITELN(CHAR(7)):

      '?' :       DESP Y:=DESP Y-DESP Y MIN :

      '@' :       DESP Y:=DESP Y+DESP Y MIN :

      '=' :       IF (MUESTRA INICIAL + DESP X MAX)
                     <=
                     (TOTAL DE MUESTRAS-
                     MUESTRAS PANTALLA) THEN
                          MUESTRA INICIAL :=
                               MUESTRA INICIAL +
                               DESP X MAX
                  ELSE
                       WRITELN(CHAR(7)):
```

```
'>' :       IF (MUESTRA INICIAL - DESP X MAX)
                >= 1 THEN
                     MUESTRA INICIAL
                          := MUESTRA INICIAL -
                          DESP X MAX
            ELSE
                 WRITELN(CHAR(7));

'A' :        DESP Y:=DESP Y-DESP Y MAX;

'B' :        DESP Y:=DESP Y+DESP Y MAX;

'C' : BEGIN
           REPEAT
                GOTOXY(1,20);
                WRITELN;
                WRITELN;
                WRITELN;
                WRITELN;
                REPEAT
                     GOTOXY(1,20);
                     WRITELN('CONTROL DE
                          PARAMETROS');
                     WRITELN('CF1] LISTA CF2]
                          AMPLI CF5] LIM' CF6]
                          INI');
                     WRITELN('CF3] MUEST CF4]
                          CORTE CF7] DESP CF8]
                          CAP');
                     WRITELN('CF9] CF10] FIN
                          ');
                     READ(KBD,B,B);
                UNTIL B IN COMANDOS :
                CASE B OF

                     ':' : LISTA PARAMETROS;

                     '<' : AJUSTE AMPLITUD;

                     '=' :MUESTRA EN PANTALLA;

                     '>' : CORTE DE MUESTRAS;

                     '?' : LIMITE PANTALLA;

                     '@':INICIALIZA PARAMETROS;

                     'A' : DESPLAZAMIENTOS :
```

```
                                    'B' : BEGIN
                                          CAPTURA FONEMA ;
                                          B:='D';END;

        END;

 UNTIL B='D';
 GOTOXY(1,20);
 WRITELN;
END;
END;
UNTIL (A='D') OR (PROCESO COMPLETO);
IF NOT ( PROCESO COMPLETO ) THEN
BEGIN
      CLRSCR ;
      WRITELN('*** EL PROCESAMIENTO DE LAS
           MUESTRAS          NO          ESTA          COMPLETO
           <RETURN>') ;
      WRITELN('( ENTRE A LA RUTINA DE <CORTE> Y DESPUES A
           LA DE <REPRODUCCION> )',
           CHAR(7));
      REPEAT UNTIL KEYPRESSED ;
 END ;
 UNTIL PROCESO COMPLETO = TRUE ;
 TEXTMODE(BW80);
```

ESTAS RUTINAS INICIALIZAN LOS PARAMETROS POR LOS CUALES SE VA A ANALIZAR LA SEÑAL

```
PROCEDURE CAPTURA FONEMA;

VAR
     I : INTEGER ;
     J : INTEGER ;
     T : INTEGER ;
     A : CHAR ;

BEGIN
     CLRSCR;
     WRITELN(' CAPTURA FONEMA');
     REPEAT
          REPEAT
               WRITELN(' INICIA LECTURA CON CICLO
                    INTERNO T=1.2 FRECUENCIA DE',
                    ' MUESTREO = 7692.3 HZ ');
               WRITELN(' CON UN CICLO INTERNO DE T=1.5 LA
                    FRECUENCIA DE ',
                    ' MUESTREO = 6.25 KHZ ');
               WRITE(' CUANTAS MUESTRAS DESEAS CAPTURAR
                    (1-7000) -> [',NO MUESTRAS,'] ');
               READLN(NO MUESTRAS);
          UNTIL (NO MUESTRAS>=1) AND (NO MUESTRAS<=7000);
          CLRSCR;
          FOR I:=1 TO 7000 DO
               VECTOR[I] := 0;
          WRITELN(' PRESIONE <RETURN> CUANDO ESTE LISTO PARA
               DAR EL FONEMA');
          REPEAT UNTIL KEYPRESSED;
          WRITELN(' *** SE INICIA LECTURA ',CHAR(7));
          FOR I:=1 TO NO MUESTRAS DO
          BEGIN
               PORTC$7003:=$0 ;
               FOR T:=1 TO 1 DO ;
               T:=T;
               VECTOR[I]:=PORTC$7000 ;
               END;  CLRSCR;
          REPEAT
               WRITE(' DESEAS EL LISTADO DE LAS
                    MUESTRAS (S/N) -> [N] ');
               READLN(A);
               IF A=CHAR(26) THEN A:='N' ;
          UNTIL (A='S') OR (A='N') ;
          IF A='S' THEN
          BEGIN
               REPEAT
                    WRITE(' CUANTAS MUESTRAS DESEAS LISTAR
                         (0-',
                    NO MUESTRAS,') -> [',J,'] ');
                    READLN(J);
               UNTIL (J>=0) AND (J<=NO MUESTRAS);
```

```
            FOR I:=1 TO J DO
                WRITE(I,'=> ',VECTOR[I],' : ');
            WRITELN ;
            WRITELN ('<RETURN>');
            REPEAT UNTIL KEYPRESSED;
        END ;   CLRSCR ;
        WRITELN (' *****  REPRODUCCION DE LAS MUESTRAS
            ***** ');
        J := 1;
        REPEAT
        WRITELN (' ***** CICLO DE REPRODUCCION NUMERO
            ',J,' ');  WRITELN;
        FOR I:= 1 TO NO MUESTRAS DO
        BEGIN
             PORT[$79C]:=VECTOR [I];
             FOR T:=1 TO 2 DO ;
             T:=1;
             T:=T;
             T:=T;
             T:=1;
        END ;
        J := J + 1 ;
        WRITE (' DESEAS REPRODUCIR UNA VEZ MAS (S/N) -> ');
        READLN (A);
    UNTIL A='N' ;
    REPEAT CLRSCR ;
        WRITE   (' DESEAS  REPETIR  LA  CAPTURA  (S/N)  -> [S] ')
        ; READLN (A) ;
        IF A=CHAR(26) THEN A:='S' ;
    UNTIL (A='S') OR (A='N') ; CLRSCR ;
UNTIL A='N' ;
END;

PROCEDURE PREGUNTA SEPARACION FONETICA ;

VAR

    AUX : INTEGER ;
    CAR : STRING[1] ;
    PREGUNTA : CHAR ;

BEGIN
    FONEMA PERTENECE A PALABRA := FALSE ;
    TOTAL DE FONEMAS := TOTAL DE FONEMAS + 1 ;
    REPEAT
        BORRA TABLAS;
        WRITELN (CHAR(7),'***  NO ENCONTRE EL FONEMA
            ',FONEMA,'');
```

```
WRITELN ('                DE LA PALABRA ''.PALABRA,''
     ') :
WRITE ('                EN LA TABLA. DAME EL FONEMA
     CORRECTO ->'):
TABLA FONEMASETOTAL DE FONEMAS].FONEMA:='
     ': (* LON MAXIMA *)
READLN(TABLA FONEMASETOTAL DE FONEMAS].FONEMA) :
FONEMA:='      ':
FONEMA:=TABLA FONEMASETOTAL DE FONEMAS].FONEMA :
FOR I:=1 TO LONG MAXIMA DO
     IF TABLA FONEMASETOTAL DE FONEMAS]
          .FONEMACI]=CHAR(0) THEN
          TABLA FONEMASETOTAL DE FONEMAS]
               .FONEMACI]:=CHAR(32):
          COMPRUEBA QUE FONEMA PERTENEZCA A PALABRA
               :
     IF (NOT (FONEMA PERTENECE A PALABRA)) THEN
     BEGIN
          BORRA MENSAJES:
          WRITELN ('*** EL FONEMA NO PERTENECE A LA
               PALABRA'.CHAR(7)):
     END
     ELSE
     BEGIN
          FOR I:=1 TO LONG MAXIMA DO
               FONEMA:=' ':
          FONEMA:= TABLA FONEMAS
               ETOTAL DE FONEMAS].FONEMA:
          AUX:=0:
          FOR I := 1 TO LONG MAXIMA DO
               IF FONEMACI]=CHAR(0) THEN
                    FONEMACI]:=CHAR(32)
               ELSE
               IF (FONEMACI]<>CHAR(32)) AND
                    (FONEMACI]<>CHAR(0)) THEN
                    AUX:=AUX+1:
               TABLA FONEMASETOTAL DE FONEMAS]
                    .FONLMA:=FONEMA:
               FONEMA PERTENECE A PALABRA:=TRUE:
               FOR I:= 1 TO AUX DO
               BEGIN
               IF PALABRAEULTIMA LETRA CORRECTA+I]
                    <>
                    FONEMACI] THEN
                         FONEMA PERTENECE A
                              PALABRA:=FALSE:
               END:
               IF (ULTIMA LETRA CORRECTA+AUX)>=14
               THEN
                    TABLA FONEMASETOTAL DE FONEMAS]
                         .FONEMACI]:=CHAR(32)
```

```
                    BEGIN
                         CAR:=PALABRA[ULTIMA LETRA
                              CORRECTA+AUX+1];
                         TABLA FONEMAS[TOTAL
                              DE FONEMAS].
                              REGLAS[I]:=CAR;
                    END;
                    FOR I:=1 TO LONG MAXIMA DO
                         IF TABLA FONEMAS[TOTAL DE
                              FONEMAS].FONEMA[I]<>'
                              THEN
                                   ULTIMA LETRA CORRECTA
                                        :=ULTIMA LETRA
                                         CORRECTA
                                        + 1 ;
                              ULTIMA LETRA ANALIZADA
                                   := ULTIMA LETRA
                                   CORRECTA + 1 ;
                         END ;
     UNTIL FONEMA PERTENECE A PALABRA ;
     FONEMA:=TABLA FONEMAS[TOTAL DE FONEMAS].FONEMA ;
     REPEAT
          CAPTURA FONEMA ;
          GRAFICA MUESTRAS;
          INICIA PANTALLA ;
          REPEAT
               WRITE(' ESTAS DE ACUERDO CON EL PROCESO (S/N) -> [S] ') ;
               READLN( PREGUNTA ) ;
               IF PREGUNTA = CHAR(26) THEN
               PREGUNTA:='S' ;
          UNTIL (PREGUNTA='S') OR (PREGUNTA='N') ;
     UNTIL PREGUNTA='S' ;
     TABLA FONEMAS [ TOTAL DE FONEMAS ].REPETICIONES :=
          REPETICIONES  ;
     ALMACENA ;
     TABLA FONEMAS [ TOTAL DE FONEMAS ].BLOQUE INICIAL :=
          BLOQUE INICIAL ;
     DESCOMPACTA DIFERENCIAS ;
     TABLA FONEMAS [ TOTAL DE FONEMAS ].LONG FONEMA :=
          MUESTRA FINAL - MUESTRA INICIO ;
     CONCATENA ( TABLA FONEMAS [TOTAL DE FONEMAS].LONG FONEMA
          ) ;
     INICIA PANTALLA ;
END;
```

```
BEGIN
     ULTIMA LETRA ANALIZADA := 1 ;
     ULTIMA LETRA CORRECTA := 0 ;
     NUMERO DE FONEMAS ENCONTRADOS := 1 ;
     REPEAT
          FONEMA TEMPORAL := -1 ;
          FONEMA LOCALIZADO := FALSE ;
          NUEVA REGLA := FALSE ;
          F := 1 ;
          FONEMA:='      ';
          LONGITUD DE.FONEMA := LONG MAXIMA;
          REPEAT
               FONEMA[F] := PALABRA[ULTIMA LETRA ANALIZADA] ;
               BUSCA FONEMA ;
               F := F + 1 ;
               ULTIMA LETRA ANALIZADA := ULTIMA LETRA ANALIZADA + 1 ;
          UNTIL (FONEMA LOCALIZADO AND REGLA LOCALIZADA) OR
               (F>=(LONG MAXIMA+1)) OR
               (PALABRA[ULTIMA LETRA ANALIZADA]=' ');
          IF (FONEMA TEMPORAL <> -1) THEN
          BEGIN
               T:=FONEMA TEMPORAL ;
               BORRA MENSAJES;
               WRITE(' NO SE ENCONTRO LA REGLA DEL FONEMA "',
                    TABLA FONEMAS[T-1].FONEMA,'" POR TANTO SE',
                    ' ASUME LA REGLA L');
               IF (ULT LET ANAL TEMP+1) > 14 THEN
                    WRITELN(' ?')
               ELSE
                    WRITELN(PALABRA[ULT LET ANAL TEMP + 1], '?');
               REPEAT
                    WRITE('ESTAS DE ACUERDO S/N ->');
                    READLN(ALFA);
               UNTIL  (ALFA='S') OR (ALFA='N');
               IF ALFA = 'S' THEN
               BEGIN
                    NUEVA REGLA := TRUE ;
                    FONEMA LOCALIZADO := TRUE ;
                    I:=0;
                    REPEAT
                         I:=I+1;
                    UNTIL (TABLA FONEMAS[T-1].REGLAS[I]=CHAR(0))
                         OR (I=20);
                    IF (I<=20) AND (TABLA FONEMAS[T-1].REGLAS[I]
                         = CHAR(0))
                         THEN
                         BEGIN
```

```
        IF (ULT_LET_ANAL_TEMP+1)>14
             THEN
                  CAR:=CHAR(32)
        ELSE
        BEGIN
             CAR:= PALABRA
                  [ULT_LET_ANAL_TEMP+1];
                  TABLA_FONEMAS[T-1]
                  .REGLAS[I] := CAR ;
        END;
        ASSIGN(TABLAS,NOMBRE_TABLAS);
        RESET(TABLAS);
        SEEK(TABLAS,(T-2));
        WRITE(TABLAS,TABLA_FONEMAS[T-1]);
        BORRA_MENSAJES;
        WRITELN('*** SE ACTUALIZO
             LA REGLA EN DISCO
             <RETURN>',CHAR(7));
        WRITELN('******* POS APUNT
             = ',FILEPOS(TABLAS));
        WRITELN('******* LONGITUD
             = ',FILESIZE(TABLAS));
        FLUSH(TABLAS);
        CLOSE(TABLAS);
        REPEAT UNTIL KEYPRESSED;
END
ELSE
BEGIN
        BORRA_MENSAJES;
        WRITELN('*** SE EXCEDIO LA
             CAPACIDAD',
             ' DE ALMACENAMIENTO DE REGLAS
             ',CHAR(7)) ;
END ;
ULTIMA_LETRA_CORRECTA
     :=ULT_LET_CORR_TEMP ;
ULTIMA_LETRA_ANALIZADA
     :=ULT_LET_ANAL_TEMP ;
BORRA_MENSAJES ;
WRITE('EL FONEMA LOCALIZADO ES : ',
TABLA_FONEMAS[T-1].FONEMA,'
     CON LA REGLA [') ;
IF (ULT_LET_ANAL_TEMP+1) > 14 THEN
     WRITELN(' ]')
ELSE
     WRITELN(PALABRA
     [ULT_LET_ANAL_TEMP + 1], ']') ;
WRITE(' <RETURN> ');
```

```
                    REPEAT
                    UNTIL KEYPRESSED ;
          END
          ELSE
               FONEMA LOCALIZADO:=FALSE ;
     END;
     IF (NOT ( FONEMA LOCALIZADO ) ) THEN
     BEGIN
          PREGUNTA SEPARACION FONETICA;
          SALVA TABLAS;
     END
     ELSE
     IF NOT(REGLA LOCALIZADA) THEN
     BEGIN
          FONEMA LOCALIZADO:=FALSE ;
          AUX := 0;
          FOR I := 1 TO LONG MAXIMA DO
               IF TABLA FONEMAS
                    [FONEMA TEMPORAL-1].FONEMA[I]
                    <>CHAR(32) THEN
                         AUX := AUX+1;
                         ULTIMA LETRA CORRECTA
                              := ULT LET CORR TEMP +
                              AUX;
                         ULTIMA LETRA ANALIZADA
                              := ULTIMA LETRA CORRECTA
                              + 1 ;
     END
     ELSE
     BEGIN
          ULTIMA LETRA CORRECTA := ULTIMA LETRA ANALIZADA - 1 ;
     END;
     IF (FONEMA LOCALIZADO
          AND REGLA LOCALIZADA) OR NUEVA REGLA
          THEN
          BEGIN
               LEE ( TABLA FONEMAS [ T - 1 ].BLOQUES,
               TABLA FONEMAS [ T - 1 ].LONG FONEMA,
               TABLA FONEMAS [ T - 1 ].BLOQUE INICIAL,
               TABLA FONEMAS [ T - 1 ].REPETICIONES ) ;
               DESCOMPACTA DIFERENCIAS ;
               CONCATENA ( TABLA FONEMAS [ T - 1
               ].LONG FONEMA ) ;
          END ;
          VECTOR FONEMAS[NUMERO DE FONEMAS ENCONTRADOS]
               := FONEMA ;
          NUMERO DE FONEMAS ENCONTRADOS
               :=NUMERO DE FONEMAS ENCONTRADOS + 1;
UNTIL (ULTIMA LETRA ANALIZADA>=15)
     OR (PALABRA[ULTIMA LETRA ANALIZADA]=' ');
```

```
PROCEDURE INICIALIZA TABLA DE FONEMAS ;

VAR
     T : INTEGER ;

BEGIN
     FOR T := 1 TO 720 DO
     BEGIN
          FOR I:= 1 TO LONG MAXIMA DO
          TABLA FONEMAS[T].FONEMA[I] := ' ' ;
          FOR I:=1 TO 20 DO
               TABLA FONEMAS[T].REGLAS[I] := CHAR(0) ;
     END;
END ;



PROCEDURE DESPLIEGA SEPARACION FONETICA ENCONTRADA;

BEGIN
     BORRA MENSAJES;
     WRITELN('SEPARACION FONETICA ENCONTRADA');
     FOR I:=1 TO NUMERO DE FONEMAS ENCONTRADOS-1 DO
     BEGIN
          WRITE(VECTOR FONEMAS[I]);
          IF I<>NUMERO DE FONEMAS ENCONTRADOS-1 THEN
               WRITE('-');
     END;
     WRITE('<RETURN>');
     REPEAT
     UNTIL KEYPRESSED;
END;



PROCEDURE REPRODUCE PALABRA ;

VAR

     T1 : INTEGER ;
     A  : CHAR ;

BEGIN
     REPEAT
     FOR I:= 1 TO APUNT DO
     BEGIN
          PORT[$79C]:=VECTOR REPRODUCCION [I];
          FOR T1:=1 TO 2 DO ;
          T1:=T1;
```

```
        T1:=T1;
        T1:=T1;
        T1:=T1;
    END;
        WRITE ('DESEAS REPETIR LA PALABRA (S/N) ->');
        READLN (A);
    UNTIL A='N';
    APUNT := 0 ;
```

ESTOS PROCEDURES CAPTURAN EL FONEMA, RELACIONAN LA PARTE A ANALIZAR CON EL FONEMA GRABADO. SE GENERAN TABLAS DE SILABAS Y DE FONEMAS GRABADOS, QUE SE PUEDEN BORRAR O ACTUALIZAR

```
PROCEDURE CREA TEXTO ;

BEGIN
     WRITELN ('              *** CREACION DE TEXTO ***');
     WRITELN ;
     WRITELN (' TECLEE UN RENGLON COMPLETO (80 CARACTERES) Y
          <RETURN> ');
     WRITELN ;
     WRITELN (' PARA TERMINAR LA CAPTURA TECLEE COMO PRIMEROS
          SIMBOLOS  *F* ');
     GOTOXY(1,7) ;
     REPEAT
          READLN (RENGLON) ;
          IF (RENGLON [1] <> '*') AND (RENGLON [2] <> 'F')
               AND (RENGLON [3] <> '*') THEN
          WRITELN(TEXTO,RENGLON);
     UNTIL (RENGLON [1] = '*') AND (RENGLON [2] = 'F') AND
          (RENGLON [3] = '*') ;
```

ESTE PROCEDIMIENTO CREA UN ARCHIVO DE TEXTO QUE PUEDE SER EDITADO MEDIANTE UN PROCESADOR DE TEXTO. ADEMAS QUE ESTE ES ANALIZADO POR RENGLONES, PALABRAS Y SILABAS

```
 PROGRAMA PRINCIPAL DEL SISTEMA FONETICS

BEGIN
     LONG BLOQUE    := 40 ;
     APUNT          :=  0 ;
     PALABRA        := '                    ' ;
     FOR I := 1 TO 80 DO
          RENGLON [ I ] := ' ' ;
     REPETICIONES        :=     1 ;
     NO MUESTRAS         := 7000 ;
     REPEAT
          INICIA PANTALLA:
          REPEAT
               A TEX := TRUE ;
               BORRA MENSAJES ;
               WRITE(' DAME EL NOMBRE DEL ARCHIVO DE TEXTO ->
                    ');
               READLN(NOMBRE);
               ASSIGN (TEXTO,NOMBRE);
               {$I-} RESET (TEXTO) {$I+} ;
               OK := (IORESULT = 0 ) ;
               IF NOT OK THEN
               BEGIN
                    BORRA MENSAJES ;
                    WRITELN (' *** EL ARCHIVO '.NOMBRE.' NO
                         EXISTE *** <RETURN> '.CHAR(7));
                    REPEAT UNTIL KEYPRESSED ;
                    REPEAT
                         BORRA MENSAJES ;
                         WRITE (' DESEAS CREAR EL ARCHIVO
                              '.NOMBRE.' (S/N) -> ');
                         READLN (ALFA) ;
                    UNTIL (ALFA = 'S') OR (ALFA = 'N') ;
                    IF ALFA = 'S' THEN
                    BEGIN
                         REWRITE(TEXTO);
                         CLRSCR ;
                         CREA TEXTO ;
                         A TEX := FALSE ;
                         OK := TRUE ;
                         INICIA PANTALLA ;
                    END;
               END ;
          UNTIL OK ;
          REPEAT
               A FON := TRUE ;
               BORRA MENSAJES ;
               WRITE(' DAME EL NOMBRE DEL ARCHIVO DE FONEMAS
                    -> ') ;
```

```
          READLN(NOMBRE_ARCHIVO_FONEMAS) ;
          ASSIGN (FONEMAS,NOMBRE_ARCHIVO_FONEMAS);
          {$I-} RESET (FONEMAS) {$I+} ;
          OK := (IORESULT = 0 ) ;
          IF NOT OK THEN
          BEGIN
               BORRA_MENSAJES ;
          WRITELN (' *** EL ARCHIVO
               '.NOMBRE_ARCHIVO_FONEMAS.
               ' NO EXISTE *** <RETURN> '.CHAR(7));
          REPEAT UNTIL KEYPRESSED ;
          REPEAT
               BORRA_MENSAJES ;
               WRITE (' DESEAS CREAR EL ARCHIVO
                    '.NOMBRE_ARCHIVO_FONEMAS
                    ' (S/N) -> ');
               READLN (ALFA) ;
          UNTIL (ALFA = 'S') OR (ALFA = 'N') ;
          IF ALFA = 'S' THEN
               BEGIN
                    REWRITE(FONEMAS);
                    A_FON := FALSE ;
                    OK := TRUE ;
               END;
          END ;
UNTIL OK ;
REPEAT
     A_TAB := TRUE ;
     BORRA_MENSAJES;
     WRITE('DAME EL NOMBRE DEL ARCHIVO DE TABLAS -> ') ;
     READLN(NOMBRE_TABLAS) ;
     ASSIGN (TABLAS,NOMBRE_TABLAS) ;
     {$I-} RESET (TABLAS) {$I+} ;
     OK := (IORESULT = 0 ) ;
     IF NOT OK THEN
     BEGIN
          BORRA_MENSAJES ;
          WRITELN (' *** EL ARCHIVO '.NOMBRE_TABLAS.
               ' NO EXISTE *** <RETURN>'.CHAR(7));
          REPEAT UNTIL KEYPRESSED ;
          REPEAT
               BORRA_MENSAJES ;
               WRITE (' DESEAS CREAR EL ARCHIVO
                    '.NOMBRE_TABLAS.
                    ' (S/N) -> ');
               READLN (ALFA) ;
          UNTIL (ALFA = 'S') OR (ALFA = 'N') ;
```

```
          IF ALFA = 'S' THEN
               BEGIN
                    REWRITE(TABLAS) ;
                    A TAB := FALSE ;
                    OK := TRUE ;
               END;
     END ;
UNTIL OK ;
CLOSE (TEXTO) ;
CLOSE (FONEMAS) ;
CLOSE (TABLAS) ;
INICIALIZA TABLA DE FONEMAS ;
ABRE ARCHIVO(NOMBRE);
IF A FON AND A TAB THEN
BEGIN
     REPEAT
          BORRA MENSAJES;
          WRITE('DESEAS CARGAR TABLAS ANTERIORES (S/N)
          -> ');
          READLN(TABLAS ANTERIORES);
     UNTIL (TABLAS ANTERIORES='S') OR
          (TABLAS ANTERIORES='N') ;
     IF TABLAS ANTERIORES = 'S' THEN
     BEGIN
          LEE TABLAS ;
          BORRA MENSAJES ;
          WRITE('DESEAS EL LISTADO DE LAS TABLAS LEIDAS
               (S/N) -> ') ;
          READLN(ALFA) ;
          IF ALFA='S' THEN
          BEGIN
               FOR J := 1 TO TOTAL DE FONEMAS DO
               BEGIN
                    WRITE(TABLA FONEMAS[J].FONEMA,'+');
                    FOR I := 1 TO 20 DO
                         WRITE(TABLA FONEMAS
                              [J].REGLAS[I],':',
                         INTEGER(TABLA FONEMAS[J]
                              .REGLAS[I]),':');
                         WRITELN;
                         WRITELN    ('     BLOQUE INICIAL     =       ',
                         TABLA FONEMAS[J].BLOQUE INICIAL
                              ;
                              '          LONG FONEMA          =          ',
                         TABLA FONEMAS[J].LONG FONEMA ,
                              '                REPETICIONES                =
                              '.TABLA FONEMAS[J].
                              REPETICIONES ,
```

```
                IF ((J MOD 5) = 0) THEN
                  BEGIN
                    WRITELN;
                    WRITELN('<RETURN>');
                    REPEAT
                    UNTIL KEYPRESSED;
                  END ;
                END;
                WRITE(' <RETURN> ');
                REPEAT
                UNTIL KEYPRESSED;
                INICIA PANTALLA;
           END;
     END
ELSE
TOTAL DE FONEMAS := 0 ;
IF TABLAS ANTERIORES='N' THEN
BEGIN
     REPEAT
          BORRA MENSAJES;
          WRITE('DESEAS BORRAR EL ARCHIVO
               ANTERIOR (S/N) -> ');
          READLN(BORRAR ARCHIVO);
     UNTIL (BORRAR ARCHIVO='S') OR (BORRAR ARCHIVO='N')
          OR (BORRAR ARCHIVO=' ');
     IF BORRAR ARCHIVO='S' THEN
     BEGIN
     REPEAT :
          BORRA MENSAJES :
          WRITELN (' *** DESEAS CONTINUAR CON EL
               PROCESO DE BORRADO ') :
          WRITE ('     DE LOS ARCHIVOS '.NOMBRE TABLAS.'
               Y '. NOMBRE ARCHIVO FONEMAS.' (S/N) ->
               ');
          READLN (BORRAR ARCHIVO);
     UNTIL (BORRAR ARCHIVO='S') OR (BORRAR ARCHIVO='N')
          ;
     IF BORRAR ARCHIVO='S' THEN
     BEGIN
          BORRA MENSAJES :
          WRITELN ('*** BORRADO DEL ARCHIVO DE TABLAS EN
          PROCESO '.CHAR(7) );
          ASSIGN(TABLAS.NOMBRE TABLAS);
          ERASE(TABLAS);
          CLOSE(TABLAS);
          ASSIGN(TABLAS.NOMBRE TABLAS);
          REWRITE(TABLAS);
```

```
                FLUSH(TABLAS);
                CLOSE(TABLAS);
                WRITELN ('*** BORRADO DEL ARCHIVO DE FONEMAS
                     EN PROCESO '.CHAR(7) );
                ASSIGN(FONEMAS,NOMBRE_ARCHIVO_FONEMAS);
                ERASE(FONEMAS);
                CLOSE(FONEMAS);
                ASSIGN(FONEMAS,NOMBRE_ARCHIVO_FONEMAS);
                REWRITE(FONEMAS);
                FLUSH(FONEMAS);
                CLOSE(FONEMAS);
           END;
     END;
END;
END;
IF (A_FON AND (NOT A_TAB)) OR (A_TAB AND (NOT A_FON)) THEN
BEGIN
     BORRA_MENSAJES ;
     WRITELN (' SELECCION INVALIDA. EL ARCHIVO DE FONEMAS Y
          EL DE TABLAS
          ');
     WRITELN (' DEBEN SER CREADOS Y/O DESTRUIDOS
          SIMULTANEAMENTE <RETURN>'.CHAR(7));
     REPEAT UNTIL KEYPRESSED ;
END ;
UNTIL (A_FON AND A_TAB) OR ( (NOT (A_FON)) AND (NOT (A_TAB)))
     ;
FIN_PROGRAMA:=' ';
REPEAT
     FOR I := 1 TO 80 DO
          RENGLONCII] := ' ' ;
     LEE_RENGLON ;
     FIN_RENGLON := FALSE ;
     NO_LETRA := 1 ;
     REPEAT
          PALABRA := '                    ' ;
          FOR I:=1 TO 10 DO
               VECTOR_FONEMASCI]:='       ': (* LONG MAXIMA*)
          LEE_PALABRA ;
          BORRA_RASTREO;
          IF FIN_PROGRAMA<>'F' THEN
          BEGIN
               INTERPRETA_FONEMA ;
               DESPLIEGA_SEPARACION_FONETICA_ENCONTRADA ;
               REPRODUCE_PALABRA ;
          END;
     UNTIL (FIN_RENGLON) OR (FIN_PROGRAMA='F');
UNTIL EOF(TEXTO) OR (FIN_PROGRAMA='F');
CLOSE;
```

REFERENCIAS

REFERENCIAS

1.- **INFORMATION THEORY**
Ingels Franklin M.

2.- **COMPUTERIZED MANAGEMENT INFORMATION**
Kelly Joseph F.
1976.

3.- **TEORIA DE LA INFORMACION**
Abrahamson Norman.

4.- **ELECTRONICS MUSIC SYNTHESIS COMPUTER COMPOSITION**
How Huberts

5.- **INFORMATION STORAGE AND RETRIEVAL SYSTEM**
James Martin.

6.- **INFORMATION SERVICES**
Hug William.

7.- **TRANSACTION ON ACOUSTICS SPEECH AND PROCESSING SIGNAL I.E.E.E.**
Lawrence Rabiner.
1976.

8.- **COMMUNICATIONS MAGAZINE I.E.E.E.**
"Vector Quantization: A Pattern-Matching Technique for Speech Coding".
Vladimir Cuperman.
December 1983.

9.- **COMMUNICATIONS MAGAZINE I.E.E.E.**
"Current Perspectives in Digital Speech".
R. E. Crochiere and J. Flanagan.
1980.

10.- **TRANSACTION ON COMMUNICATIONS I.E.E.E.**
"Speech Coding".
J. L. Flanagan.
April 1979.

11.- **COMUNICACION HOMBRE-MAQUINA POR VOZ**
Parte I, II, III y IV.
Golderos R. Martínez, J. Nombela.
Julio de 1980.

12.- **SIMULACION EN COMPUTADORAS DIGITALES.**
Korn Granino, Arthur.

13.- **METODOS DE INFORMACION POR COMPUTADORA.**
Taylor, Thomas.

14.- **SPECTRUM I.E.E.E.**
"Synthesis Voices for Computers".
J. L. Flanagan, C. H. Coker.
October, 1970.

15.- **DIGITAL PROCESSING OF SPEECH SIGNALS**
Rabiner / T. W. Schafer.
Prentice Hall.

16.- **DIGITAL CODING OF WAVE FORM, PRINCIPLES AND APPLICATIONS TO SPEECH AND VIDEO**
N. S. Jayant / Peter Noll

17.- **ANALOG TO DIGITAL / DIGITAL TO ANALOG CONVERSION TECHNIQUES.**
Hoeschele.

18.- **DISEÑO CON CIRCUITOS INTEGRADOS TTL**
Robert L. Morris, John R. Miller.
Editorial SECSA.

19.- **TRANSACTIONS ON INFORMATION THEORY I.E.E.E. VOL. 24 No. 2**
"Quantizing Transform of Random Time - Series".
Neal C. Gallagher.

20.- **NATIONAL SEMICONDUCTOR APPLICATION**
Walt Sirovy.
1984.

21.- **LINEAR DATABOOK NATIONAL SEMICONDUCTOR 1980**

22.- **MICROELECTRONICS**
Jacob Millman.
McGraw Hill.