ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE INGENIERÍA

DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE REDUCCIÓN DEL RUIDO INDUSTRIAL EN LA COMUNICACIÓN ENTRE OPERADORES

PROYECTO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA Y CONTROL

EDWIN ANÍBAL MUÑOZ RAZA

XIMENA ALEXANDRA TAPIA LEÓN

DIRECTOR: DR. LUIS CORRALES

QUITO, MARZO 2007

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Edwin Aníbal Muñoz Raza y Ximena Alexandra Tapia León, bajo mi supervisión.

Dr. Luis Corrales

DIRECTOR DEL PROYECTO

AGRADECIMIENTOS

Ante todo quiero agradecer a mis padres por brindarme todo su cariño, apoyo, comprensión y sobre todo por su paciencia; la espera ha sido larga para ellos, y para mí también, pero al fin tantos años de esfuerzo y sacrificio rinden su fruto.

Agradezco también de manera muy especial a mis amigos que me acompañaron, en las buenas como en las malas, en especial a Verónica Cadena y Mauricio Enríquez quienes además de brindarme su amistad me brindaron su confianza y siempre tuvieron fe en mis capacidades.

Al Dr. Luis Corrales, director de este Proyecto, quien con su guía, consejos y conocimientos, fue un gran aporte para la culminación de este Proyecto.

Finalmente agradezco a Ximena Tapia y a su familia, particularmente a Paulina y Tatiana Tapia, quienes me brindaron su hospitalidad durante el tiempo en el que, con Ximena, desarrollamos este Proyecto de Titulación.

Aníbal

DEDICATORIA

A mis padres, Aníbal Muñoz Játiva y Lolita Raza de Muñoz, a mis hermanos y a toda mi familia.

Aníbal

AGRADECIMIENTOS

A mis padres y hermanos por impulsarme y ayudarme a superar las dificultades que se han presentado en la travesía hacia la culminación de este proyecto.

A Flavio, quien ha compartido conmigo éxitos y fracasos durante el avance de este importante trabajo.

Al Dr. Luis Corrales por su tiempo y constancia, y por ser quien con sabiduría inspiradora guió todos y cada uno de los pasos dados para sacar este proyecto adelante.

Finalmente agradezco a todos y cada una de las personas que de una u otra manera hicieron posible que este trabajo llegue a su fin, sobretodo a mi compañero de tesis Aníbal.

Ximena

DEDICATORIA

Dedico este trabajo a mis padres por ser el soporte fundamental y apoyo incondicional durante toda mi vida de estudiante y durante el desarrollo de la tesis.

Ximena

CONTENIDO

CAPÍTULO 1

CAPÍTULO 3

CAPÍTULO 4

CAPÍTULO 5

RESUMEN

En este proyecto se diseña e implementa un sistema de comunicación inalámbrico que permite la comunicación entre operadores cuando existe ruido ambiental industrial contaminante afectando sus voces.

Para resolver este problema se diseño un sistema basado en filtros digitales adaptivos para reducir el ruido interferente. El sonido de los interlocutores se captura por medio de micrófonos adaptados a las orejeras industriales que emplean los operadores. Esta información se envía de manera inalámbrica hacia las orejeras del interlocutor, donde un filtro adaptivo implementado en un DSP, reduce el ruido ambiental y permite por medio de audífonos que el operador escuche claramente la voz.

Para la reducción del ruido se prefirió el uso de filtros adaptivos, pues se necesita que ajusten sus parámetros automáticamente sin tener un conocimiento a priori de la señal y del ruido.

El filtro adaptivo usado en este trabajo es lineal y variante en el tiempo y sus parámetros son obtenidos minimizando el error cuadrático medio (nLMS).

Se implementó inicialmente el filtro adaptivo en Matlab y LabVIEW para comprobar el óptimo desempeño del mismo y, posterior a las pruebas realizadas en estos paquetes computacionales, el algoritmo fue implementado en un DSP.

Las pruebas realizadas tanto en Matlab, LabVIEW y en pruebas de campo ratificaron el comportamiento y bondades del filtro diseñado.

PRESENTACIÓN

En este proyecto de titulación se describen los beneficios y ventajas del uso de filtros digitales adaptivos, ya que estos pueden ser utilizados en varias aplicaciones de carácter no estacionario (variantes en el tiempo), que requieren que las características del filtro se ajusten a las necesidades específicas de un problema del mundo real.

La implementación de un filtro digital adaptivo basado en procesadores digitales de señal (DSP) tienen muchas ventajas sobre otros enfoques. No solo por que los requerimientos de potencia, espacio y manufactura, son muy reducidos, sino que además la programabilidad provee flexibilidad para la actualización del sistema y mejoras del software. Adicionalmente, el uso de un DSP posibilita el empleo de los filtros digitales en aplicaciones en tiempo real.

Con la ayuda de un filtro digital adaptivo, corriendo en un DSP, se tiene como objetivo global implementar un sistema que permita la comunicación inalámbrica entre operadores. El filtro digital se encargará de reducir el ruido ambiental que se mezcla con la señal a ser transmitida, haciendo de esta manera que la intercomunicación sea más clara.

Con estos objetivos, en el Primer Capítulo se describe los principales tipos de filtros digitales, así como también los algoritmos adaptivos más utilizados, optando para el desarrollo de este Proyecto de Titulación por la implementación del filtro digital adaptivo nLMS, además se incluye teoría sobre características, ventajas, desventajas y campos de aplicación de los DSP.

El Segundo Capítulo describe el diseño del algoritmo y el hardware a ser implementado, abordando de manera general cada uno de los componentes del equipo, además se explica cada una de las etapas del proyecto en diagramas de

bloque y las características del chip de Motorola DSP56303 elemento con el que se desarrolla el Proyecto de Titulación.

En el Tercer Capítulo se explica el desarrollo del algoritmo nLMS implementado en Matlab y LabVIEW, así como también, una explicación del funcionamiento y uso de los programas creados en estos paquetes computacionales. Para el caso del DSP56303, se explica los programas desarrollados, la manera de compilarlos y como deben ser cargados en el DSP. También, se muestra los diagramas de flujo de los distintos programas implementados en Matlab, LabView y el DSP56303.

En el Cuarto Capítulo se diseña e implementa cada una de las etapas del proyecto, basándose para ello, en los circuitos recomendados en hojas de datos (datasheets) y material bibliográfico correspondiente a cada uno de los elementos que conforman el proyecto; además se incluyen los esquemáticos correspondientes al diseño así como también los respectivos ruteados de los circuitos impresos.

En el Quinto Capítulo se analizan los resultados obtenidos de las pruebas realizadas en los programas desarrollados en Matlab, LabVIEW y para el DSP. Se realiza también un análisis de los costos de implementación así como también los costos por el tiempo de investigación del Proyecto.

El Sexto Capítulo hace referencia a las Conclusiones y Recomendaciones realizadas por los autores.

CAPÍTULO 1

GENERALIDADES

La exposición al ruido excesivo si es suficientemente fuerte y duradero puede producir efectos nocivos sobre la salud de las personas, desde intranquilidad y disminución del potencial productivo, hasta pérdida de la capacidad auditiva y algunos problemas psicológicos agudos.

La intensidad, o potencia del sonido, se mide en decibeles. Los decibeles son medidos logarítmicamente. Esto significa que la intensidad se incrementa en unidades de 10, cada incremento es 10 veces mayor que el anterior. Entonces, 20 decibeles es 10 veces la intensidad de 10 dB, y 30 dB es 100 veces más intenso que 10 dB [1].

Tabla 1.1 Intensidades sonoras y sonidos relacionados

(dB) Nivel aproximado	**FUENTE DE SONIDO**	**TOLERANCIA**
0	Sonido más tenue que percibe el oído humano	Ideal
30	Chistido, Biblioteca silenciosa	Aceptable
50	Conversación normal.	Tolerable
60	Máquina de coser, máquina de escribir	Máximo tolerable
90	Cortadora de pasto, herramientas pesadas, tráfico pesado.	8 horas al día es la máxima exposición tolerable (para el 90% de la gente).
100	Motosierra, Martillo neumático.	2 horas por día es la máxima exposición tolerable sin protección.
115	Concierto de rock pesado, bocina de auto.	15 minutos por día es la máxima exposición tolerable sin protección.

140	Explosión, Motor de jet.	El ruido causa dolor y aún una breve exposición lesiona a oídos no protegidos. Máximo ruido permitido con protectores acústicos.

La mayoría de los expertos concuerdan que la exposición continua a más de 85 dB puede ocasionar la pérdida auditiva, cuando esta comienza se pierden primero las frecuencias altas. La pérdida de las frecuencias altas produce también distorsión del sonido, por lo que se hace difícil entender la palabra aunque se la escuche. Por ello, si se trabaja en un ambiente excesivamente ruidoso se debe usar protectores. Los dispositivos de protección acústica disminuyen la intensidad del sonido que llega al tímpano y pueden prevenir casi totalmente el daño causado al oído.

Colocados adecuadamente los protectores pueden reducir el ruido entre 15 y 30 dB, pero los protectores reducen ligeramente la habilidad para entender una conversación normal. Por ejemplo, en las industrias es necesario que los operadores de las máquinas entiendan claramente alguna instrucción para que la acción que realicen este acorde con esa instrucción y no se la interprete erróneamente [1].

Por ello se ha visto la necesidad de diseñar un dispositivo que reduzca el ruido industrial y además permita la comunicación inteligible entre dos personas.

Puesto que el sistema que aquí se diseña hace uso extensivo de los filtros digitales, en este capítulo se abarca la teoría sobre estos filtros, en particular aquellos temas que serán empleados en este trabajo.

1.1 FILTROS DIGITALES

Un filtro digital es un algoritmo matemático que realiza ciertas operaciones en la señal de entrada, para obtener una señal deseada a la salida.
Hay dos motivos para implementar un filtro digital:

1. Mejorar la calidad de la señal de entrada.
2. Procesar u obtener cierta información de tal señal.

En el primer caso, el filtro digital es un sistema que transforma las entradas de una manera específica eliminando ciertas componentes no deseadas. Cuando esas componentes no deseadas se describen en función de sus contenidos de frecuencia, los filtros, se denominan filtros selectivos en frecuencia.

El segundo caso es una característica única de los filtros digitales. Debido a que el filtro procesa matemáticamente la información de entrada, es posible extraer datos que de otra manera permanecerían ocultos o disimulados, en la misma.

En los últimos años, los filtros digitales han sustituido a los filtros analógicos en muchas aplicaciones debido a su mayor fiabilidad, mayor flexibilidad y superiores prestaciones.

Por estas razones, en ciertas aplicaciones en las que se desea filtrar una señal en tiempo continuo, un filtro analógico se emplea en su versión digital.

Dado un conjunto de especificaciones, el diseño de un filtro digital consiste en obtener una aproximación analítica a las características del filtro deseadas, en forma de una función de transferencia H(z) [2].

Las ventajas de los filtros digitales son [3]:

- Características imposibles con filtros analógicos (fase lineal).
- No cambian cualquiera que sea el entorno.
- Procesamiento de varias señales con un único filtro.
- Posibilidad de almacenar datos.

- Repetitividad.
- Uso en aplicaciones de muy bajas frecuencias.

Entre las desventajas de los filtros digitales se tiene:

- Limitación de velocidad.
- Efectos de la longitud finita de las palabras.
- Tiempos de diseño y desarrollo.

1.1.1 CLASIFICACIÓN

Un filtro digital realiza una transformación de la señal de entrada [4], alterando su contenido espectral, la magnitud y la fase para acomodarla a ciertas especificaciones.

Los dos filtros más utilizados en la cancelación activa de perturbaciones son:

- FIR (Finite Impulse Response)
- IIR (Infinite Impulse Response)

Los filtros digitales también suelen clasificarse de acuerdo con la estructura utilizada para su implementación:

- Realización transversal o no-recursiva
- Realización recursiva

1.1.1.1 Filtros FIR

Los filtros FIR [5], de respuesta impulso finita, reciben ese nombre ya que la salida depende únicamente de la entrada, de forma que el efecto de un impulso en la entrada se extingue en un tiempo finito.

Un filtro FIR de longitud *M* con entrada *x(n)* y salida *y(n)* se describe mediante la ecuación en diferencias:

$$y(n) = b_0 x(n) + b_1 x(n-1) + \ldots + b_{M-1} x(n-M+1)$$

$$y(n) = \sum_{k=0}^{M-1} b_k \; x(n-k) \qquad n \geq 0 \tag{1.1}$$

Donde b_k son los coeficientes del filtro.

Y su función de transferencia discreta es:

$$\frac{Y(z)}{X(z)} = \sum_{k=0}^{M-1} b_k z^{-k} \tag{1.3}$$

Por otra parte se puede expresar la salida del filtro *y(n)* como una convolución de la entrada *x(n)* con la respuesta impulso del filtro *h(n)*:

$$y(n) = \sum_{k=0}^{M-1} h(k) * x(n-k) \tag{1.4}$$

Las ecuaciones (1.1) y (1.4) son idénticas, por lo tanto, los coeficientes b_k=*h(k)*.
Los filtros FIR son siempre estables y son capaces de tener una respuesta de fase que es lineal, lo que equivale a decir que su respuesta tiene un retraso constante.

Se puede demostrar que la respuesta de un filtro FIR es de fase lineal si los coeficientes *h(n)* cumplen:

$$h(n) = \pm h(M-1-n) \qquad n = 0,1,\ldots,M-1$$

Es decir, los coeficientes tienen algún tipo de simetría.

El mayor problema de los filtros FIR es que para cumplir con las mismas especificaciones de diseño que los filtros IIR, requieren un filtro de orden mucho mayor.

Hay dos métodos de diseño de filtros FIR:

- Método de diseño usando ventanas (windowing[1])
- Método Iterativo basado en condiciones óptimas de diseño

1.1.1.1.1 Método de diseño usando ventanas (windowing)

Este método consiste en la convolución $H_D(\omega)*W(\omega)$ indicada en la Figura 1.1, en la cual $H_D(\omega)$ es la respuesta impulso en el dominio de la frecuencia del filtro deseado (en este caso un filtro pasa banda) y $W(\omega)$ es la ventana aplicada.

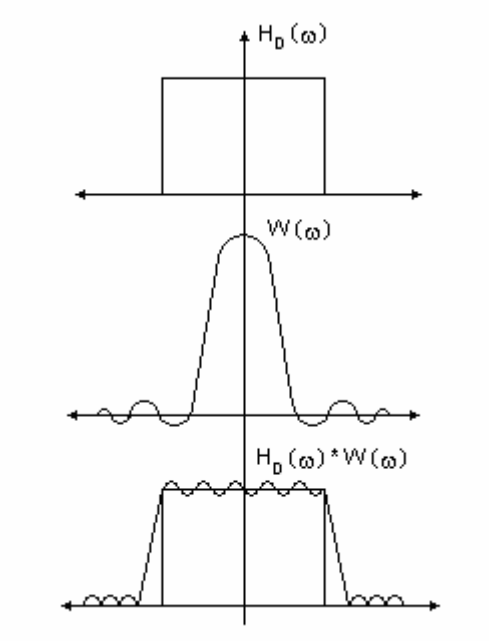


Figura 1.1 Resultado del ventaneo de la respuesta impulso deseada en el dominio de la frecuencia.

Proceso de Diseño de Filtros FIR por este método [5]:

- Normalización de frecuencias por la frecuencia de muestreo
- Truncamiento de la respuesta impulso de un filtro ideal *h(n)* a $h_D(n)$
- Selección de una ventana *W(n)* de M puntos para obtener $h(n)=h_D(n)\cdot W(n)$, donde M es el orden del filtro

[1] **Windowing,** también conocido como método de ventaneo en la traducción de algunos textos de procesamiento digital de señales.

Tipos de Filtros para la función $h_D(n)$:

- Pasa bajo
- Pasa alto
- Pasa banda
- Rechaza banda

Los tipos de ventana más comunes son:

- Rectangular
- Hanning
- Hamming
- Blackman
- Kaiser

El mérito de esta solución reside en su simplicidad, requiriendo comparativamente poco cálculo, sin embargo, el diseño resultante no es usualmente el mejor.

1.1.1.1.2 Método Iterativo basado en condiciones óptimas de diseño

En los últimos años el uso de técnicas asistidas por computador para el diseño de filtros digitales se ha extendido de forma considerable [2], y han aparecido algunos paquetes de software para la realización de los diseños. Se han desarrollado técnicas que, en general, requieren la minimización de una función de coste adecuadamente escogida.

En muchas ocasiones, en vez de minimizar la desviación en todas las frecuencias, se puede escoger realizar la minimización solo en un número finito de frecuencias.

Típicamente, el problema de minimización es bastante complejo y las ecuaciones resultantes no se pueden resolver analíticamente. Generalmente se emplean procedimientos de búsqueda iterativa para determinar un conjunto óptimo de coeficientes del filtro. Se comienza con una selección inicial arbitraria de los

coeficientes del filtro y se van modificando sucesivamente, de forma que la función de coste resultante se reduzca en cada paso.

El procedimiento se detiene cuando las modificaciones posteriores de los coeficientes no producen una reducción de la función de coste. Hay disponibles varios algoritmos estándar y diversos paquetes de software para determinar los coeficientes óptimos del filtro.

Los coeficientes del filtro se escogen para minimizar el error con respecto a la respuesta deseada. Existen también programas de computador para determinar los coeficientes óptimos del filtro en este caso.

1.1.1.2 Filtros IIR

Los filtros IIR [7], de respuesta impulso infinita, reciben ese nombre ya que la salida puede depender tanto de la entrada como de la propia salida del filtro, de forma que el efecto de un impulso en la entrada puede no extinguirse en un tiempo finito.

La respuesta de un filtro IIR de *N* y *M* coeficientes, a_k y b_k, para una entrada *x*(n) es:

$$y(n) = \sum_{k=0}^{M} b_k x(n-k) - \sum_{k=1}^{N} a_k y(n-k) \tag{1.5}$$

y su función de transferencia discreta es:

$$\frac{Y(z)}{X(z)} = \frac{b_0 + b_1 z^{-1} + \ldots + b_M z^{-M}}{1 + a_1 z^{-1} + \ldots + a_N z^{-N}}$$

Entonces:

$$\frac{Y(z)}{X(z)} = \frac{\sum_{k=0}^{M} b_k z^{-k}}{1 + \sum_{k=1}^{N} a_k z^{-k}} \tag{1.6}$$

La forma clásica de diseñar un filtro discreto IIR consiste en transformar un filtro analógico en uno digital que cumpla las especificaciones prescritas.

La principal ventaja de los filtros IIR es que pueden realizar cualquier transformación lineal y discreta con un número finito de parámetros, resultando más eficientes que los filtros FIR. Por otro lado, tienen inconvenientes importantes, como la falta de garantía en su estabilidad o los problemas derivados de la cuantización y redondeo de sus coeficientes, mucho mayores que en el caso de los filtros FIR.

Existen dos métodos muy utilizados para el diseño de estos filtros:

- Método de invarianza al impulso
- Transformación bilineal

1.1.1.2.1 Método de invarianza al impulso

Este método es conocido también como muestreo de la respuesta impulso [2], ya que si la entrada de prueba es un impulso unidad, la condición se convierte en que la respuesta impulso de los dos sistemas (sistema en tiempo discreto y el correspondiente sistema analógico) tengan el mismo valor en los instantes de muestreo; es decir:

$$h(nT) = h(t)$$

Los pasos a seguir para el diseño mediante este método son:

1. Determinar un filtro análogo normalizado, H(s), con las especificaciones requeridas.
2. Descomponer H(s) en fracciones parciales, y determinar la transformada Z de cada término utilizando una tabla de transformadas. Combinar estos términos para obtener H(z).

Aunque la técnica de invarianza al impulso es muy directa, tiene una desventaja, se obtiene un sistema en tiempo discreto a partir de un sistema en tiempo

continuo mediante un proceso de muestreo, el cual introduce solapamiento (aliasing); por lo tanto, el filtro digital resultante no cumplirá exactamente las especificaciones originales de diseño.

1.1.1.2.2 Transformación bilineal

Este método consiste en encontrar un filtro digital H(z), equivalente al filtro análogo H(s), aplicando los siguientes pasos:

1. Encontrar una función de transferencia H(s), para un filtro análogo pasa bajo normalizado (filtro prototipo).
2. Determinar las bandas críticas del filtro deseado aplicando la siguiente relación:

$$\omega_d = tg\left(\frac{\omega_a T}{2}\right)$$

   Siendo ω_a = frecuencia analógica especificada.
3. Denormalizar el filtro análogo prototipo, reemplazando s en H(s) por alguna de las siguientes transformaciones indicadas en la Tabla 1.2:

Tabla 1.2 Transformaciones para denormalizar filtro prototipo

$s = \frac{s}{\omega_d}$	Filtro Prototipo ➔ Pasa bajo
$s = \frac{\omega_d}{s}$	Filtro Prototipo ➔ Pasa alto
$s = \frac{s^2 + {\omega_0}^2}{W.s}$	Filtro Prototipo ➔ Pasa banda
$s = \frac{W.s}{s^2 + {\omega_0}^2}$	Filtro Prototipo ➔ Rechaza banda

Donde: $\omega_0^{\,2} = \omega_{d1} * \omega_{d2}$

$W = \omega_{d2} - \omega_{d1}$

4. Aplicar la Transformada Bilineal para mapear H(s) a H(z):

$$s = \frac{z-1}{z+1}$$

Aunque los filtros IIR presentan características atractivas, tienen algunos inconvenientes como por ejemplo el no poder aprovechar las ventajas de la FFT (Fast Fourier Transform) en la implementación, ya que para esto es necesario un número de puntos finito.

Otra desventaja es que los IIR alcanzan una magnífica respuesta en amplitud a expensas de un comportamiento no lineal en fase.

1.1.1.3 Realización transversal o no-recursiva

Esta basada en la ecuación (1.3) que utiliza la respuesta impulso [4]. En ella se calculan las muestras de la señal de salida en función únicamente de las muestras de la señal de entrada. Examinando las expresiones que se van a utilizar para la implementación de los filtros digitales en el dominio del tiempo, se observa que hay tres operaciones elementales: retardo, multiplicación y suma, como se muestra en la Figura 1.2:

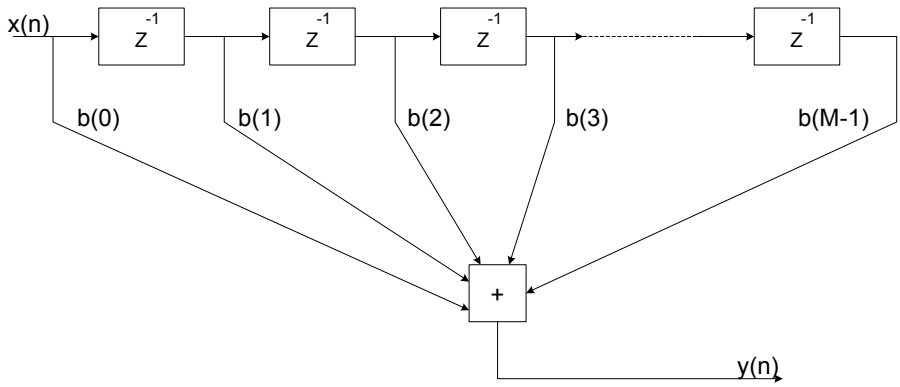


Figura 1.2 Estructura no recursiva.

1.1.1.4 Realización recursiva

Se basa en la ecuación (1.6) donde para calcular las muestras de salida se utilizan también las muestras de salida anteriores. La configuración recursiva se indica en la Figura 1.3 [4]:

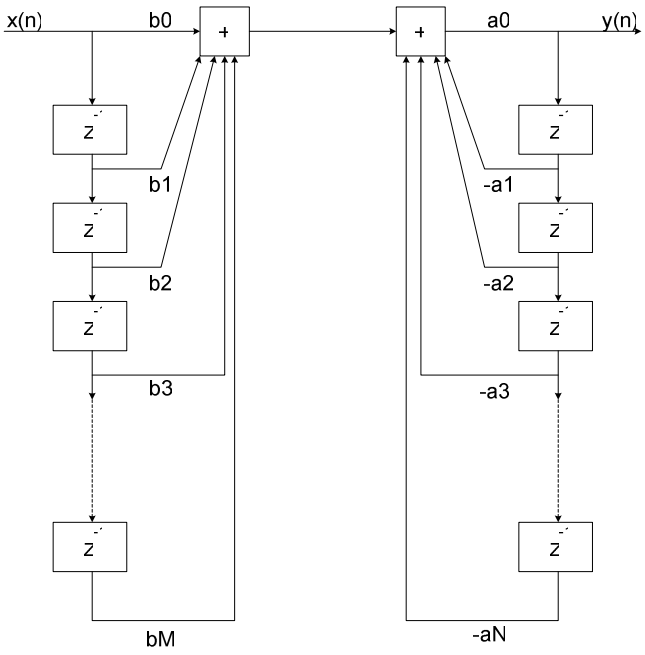


Figura 1.3 Estructura recursiva.

Junto a estas estructuras se utilizan otros métodos, como el empleo directo de la transformada discreta de Fourier de la señal de entrada, modificada adecuadamente de acuerdo con la respuesta en frecuencia deseada, y posterior reconstrucción de la salida por transformada discreta de Fourier inversa.

Un filtro IIR solo puede ser implementado con una estructura recursiva, un filtro FIR puede serlo tanto con estructuras recursivas como no-recursivas.

1.1.2 DIFERENCIAS ENTRE FIR E IIR

Las diferencias entre los filtros FIR e IIR, se resume en la Tabla 1.3 [3]:

Tabla 1.3 Diferencias entre FIR e IIR

	FIR	IIR
Función del sistema H(z)	Solo contiene ceros	Contiene polos y ceros
Respuesta en frecuencia	Para selectividades altas se requieren órdenes altos (todos los polos están en z=0)	Se pueden conseguir selectividades altas con órdenes reducidos al disponer de pares polo-cero
Característica de fase	Es posible conseguir fase lineal	Solo se puede conseguir fase lineal utilizando ecualizadores con lo que el filtro es más complejo
Estabilidad	Son siempre estables	Pueden ser inestables si los polos caen fuera del círculo unidad
Estructura	La estructura más utilizada es la no recursiva	Solo puede usarse la estructura recursiva
Complejidad	La complejidad depende de la longitud de su h(n)	Al utilizarse la transformación bilineal los cálculos se reducen. Son poco complejos

1.2 ALGORITMOS ADAPTIVOS

Los algoritmos de adaptación son algoritmos de optimización cuya finalidad es obtener los parámetros (de un filtro, en este caso) que minimicen algún criterio preestablecido. En esta aplicación el criterio a minimizar gira entorno a la señal capturada en el punto de cancelación; es decir, el error de cancelación.

El tipo de algoritmos de optimización que se utiliza de manera común calcula los parámetros de forma recursiva; es decir, en cada paso de cálculo se actualizarán los parámetros.

Entre los algoritmos de adaptación más utilizados están:

- Algoritmo LMS (Least Mean Square)
- Algoritmo nLMS (Normalized Least Mean Square)

1.2.1 ALGORITMO LMS

También denominado Regla de Widrow–Hoff. Esta regla es muy importante, ya que es la base de la mayoría de los algoritmos de aprendizaje de un amplio conjunto de redes neuronales [6]. Básicamente, la metodología para deducir una regla de aprendizaje se resume en dos puntos:

1. Definición de la función de error, también denominada función de coste mide la bondad del modelo. Mientras más pequeña sea, más eficiente es el modelo. Lógicamente depende de los pesos sinápticos, que son las incógnitas a resolver en el entrenamiento.
2. Optimización de la función de error. Se busca un conjunto de pesos sinápticos que minimice la función de error. Esta búsqueda se realiza mediante un proceso iterativo denominado Descenso por el Gradiente.

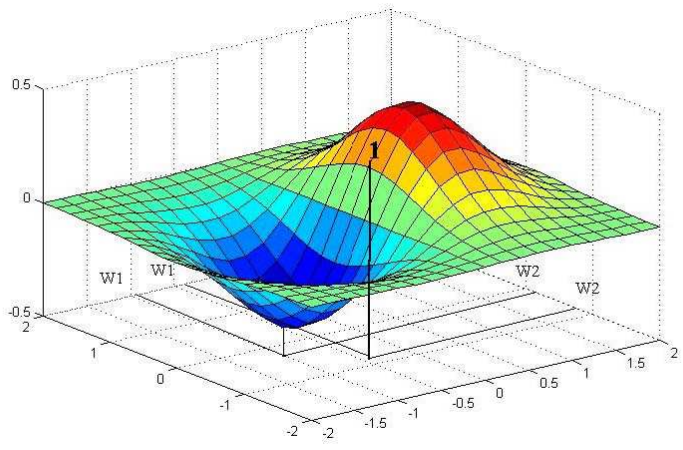


Figura 1.4 Aproximación gráfica del descenso por gradiente.

Se puede realizar una aproximación gráfica del descenso por el gradiente observando la Figura 1.4. El punto uno está representado por un conjunto de pesos (en este caso y para simplificar, dos pesos: W_1 y W_2). Se calcula el sentido de la máxima variación de la función de error y se toma el camino opuesto. Esta máxima variación viene dada por el Gradiente de la función de error en el punto uno. Lógicamente, al tomar la dirección contraria del gradiente se apunta hacía un mínimo, que seguramente será local. El proceso se itera hasta que se alcance ese mínimo.

Dicho con otras palabras: la variación de los pesos depende de su gradiente, pero además, a éste se le multiplicará por un número infinitesimal para que la variación sea pequeña, ya que de otro modo, si ésta fuese grande se tiene el riesgo de que en una variación se pase del mínimo y se vuelva a estar en una colina. Por tanto, el conjunto de pesos en el momento (n + 1) será igual al valor de esos pesos en el momento n menos el gradiente por un número infinitesimal que se llama coeficiente de aprendizaje que definirá el tamaño de cada iteración.

$$W_{n+1} = W_n - \mu \nabla_n \tag{1.7}$$

Donde: ∇_n es el gradiente de la función del error cuadrático medio

μ es el coeficiente de aprendizaje

Hay que llegar a encontrar cierto equilibrio en la definición del coeficiente de aprendizaje ya que si es muy pequeño, el proceso se hace eterno, mientras que si es muy grande no se encontrará jamás el mínimo porque se dedica a oscilar de una pared a otra.

1.2.2 ALGORITMO nLMS

El algoritmo LMS presenta un inconveniente importante, presenta una fuerte dependencia de los datos en la elección de μ, afectando el tiempo de

convergencia. Este inconveniente es subsanado con la normalización de μ por una estimación de la potencia media de la señal, que lleva a la formulación del nLMS.

La ecuación de actualización de datos es:

$$W_{n+1} = W_n - \frac{\mu}{P_n}\nabla_n \tag{1.8}$$

Siendo P_n el estimador de potencia, el cual se calcula de forma recursiva, este calculo de potencia es, de hecho, equivalente a aplicar una ventana exponencial sobre los vectores calculados en instantes de tiempo pasados, ponderando más los valores más recientes y dando menos importancia a los más antiguos como se indica en la siguiente ecuación:

$$\mathrm{P(n+1)} = \gamma\,\mathrm{P(n)} + (1-\gamma) * \mathrm{x'(n)} * \mathrm{x(n)} \tag{1.9}$$

Donde: γ es el factor de olvido $0<\gamma<1$

x'(n) es el vector transpuesto de *x(n)*

En este caso, el error decae con mayor pendiente en comparación con el algoritmo LMS clásico.

1.3 FILTROS ADAPTIVOS

Un filtro digital adaptivo se trata de un filtro digital clásico, con la particularidad de que sus parámetros (pesos), pueden variar a lo largo del tiempo, en contraposición a los coeficientes de los filtros fijos que son invariantes con el tiempo.

El encargado de modificar los pesos es un algoritmo de optimización que, con la finalidad de minimizar algún criterio, calcula constantemente los parámetros más adecuados [7].

La ecuación de entrada-salida de un filtro adaptivo digital es [8]:

$$y(n)=\sum_{k=0}^{M} b_k(n)\,x(n-k)-\sum_{k=1}^{N} a_k(n)\,y(n-k) \qquad n\geq 0 \tag{1.10}$$

Donde y(n) y x(n) son las muestras de salida y entrada, $a_k(n)$ y $b_k(n)$ son los pesos del filtro y N+M+1 es el número total de coeficientes del filtro.

Nótese que en la ecuación anterior los coeficientes del filtro son determinados en cada instante n, además si $a_k(n) = 0$ para $1 \leq k \leq N$ y la longitud del filtro es M resulta un filtro adaptivo FIR no recursivo, esto es:

$$y(n)=\sum_{k=0}^{M-1} b_k(n)\,x(n-k) \tag{1.11}$$

En adelante los pesos del filtro serán conocidos como $\mathbf{W_k}$, de esta manera la ecuación del FIR adaptivo quedará de la siguiente manera:

$$y(n)=\sum_{k=0}^{M-1} W_k(n)\,x(n-k) \tag{1.12}$$

El filtro digital adaptivo podría perfectamente implementarse mediante un filtro IIR, pero los filtros FIR son mucho menos susceptibles que los IIR de ser inestables. Ello no quiere decir que los filtros FIR sean siempre estables, de hecho, su estabilidad depende del algoritmo que se use para ajustar sus coeficientes. Sin embargo, se utilizan generalmente filtros FIR porque su estabilidad/inestabilidad es más controlable que en los IIR.

El criterio de adaptación está relacionado con la obtención de una señal de error indicativa del desajuste entre los parámetros óptimos W_o y el valor actual estimado W del filtro en cuestión. Entre las diversas aplicaciones de este tipo de filtro, se puede citar las de predictor, cancelador y estimador. Cualquiera sea la utilización del módulo adaptivo se debe obtener un mecanismo de actualización

de los pesos W, en el caso particular de este proyecto un combinador lineal adaptivo.

El combinador lineal adaptivo constituye un estimador del tipo no recursivo; su estructura se muestra en la Figura 1.5, donde el estimado y_n (señal de salida del estimador) se define en términos de una combinación lineal de la señal de entrada x_n [9]:

$$y_n = w_1 x_{1n} + w_2 x_{2n} + w_3 x_{3n} + ... + w_M x_{M\,n} \tag{1.13}$$

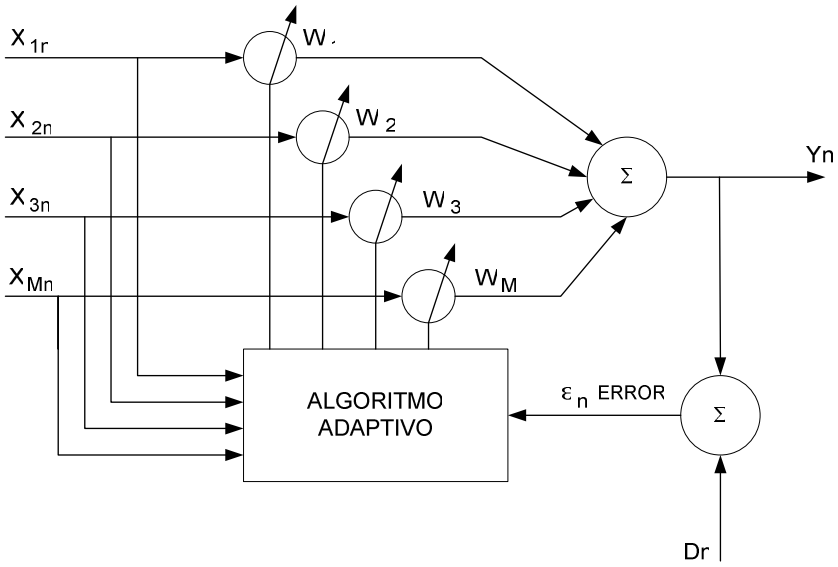


Figura 1.5 Diagrama esquemático del combinador lineal adaptivo.

El vector de la señal de entrada X_j se define como:

$$X_n = \begin{bmatrix} x_{1n} \\ x_{2n} \\ \vdots \\ x_{M\,n} \end{bmatrix} \tag{1.14}$$

La dimensión de este vector esta dada por el orden del filtro y esta formado por las componentes de la señal de entrada en el instante n. los coeficientes de ponderación o pesos $w_1, w_2, w_3, \ldots, w_n$ son ajustables. El vector de coeficientes de ponderación se define de la siguiente manera:

$$W = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_M \end{bmatrix} \tag{1.15}$$

De acuerdo con la ecuación (1.13) la señal de salida es:

$$y_n = X_n{}^T W = W^T X_n \tag{1.16}$$

Donde, W^T y $X_n{}^T$ representan los vectores transpuestos de W y X_n respectivamente.

Se define la señal de error ε_n, como la diferencia entre la señal d_n (la cual se provee externamente y se denomina como la señal deseada actual) y la salida del combinador lineal y_n:

$$\varepsilon_n = d_n - y_n$$

$$\varepsilon_n = d_n - X_n{}^T W = d_n - W^T X_n \tag{1.17}$$

Elevando al cuadrado la ecuación anterior se obtiene una expresión para el error cuadrático medio, dando como resultado una función cuadrática del vector de coeficientes de ponderación en cualquier tiempo en particular, cuya representación tiene la forma de una superficie hiperparaboidal cóncava, tal como se muestra en la Figura 1.4.

Para minimizar el error cuadrático se utiliza el método del gradiente, diferenciando la función de error cuadrático medio con respecto al vector de coeficientes de ponderación.

$$\nabla_n = \frac{\partial E[\varepsilon_n{}^2]}{\partial W} \tag{1.18}$$

Donde $E[\varepsilon_n{}^2]$ es el estimado del error cuadrático y W es el vector de coeficientes de ponderación.

El procedimiento iterativo requiere conocer el gradiente exacto, pero este no se conoce, por lo tanto es estimado a partir de los datos, esto se hace asumiendo que $E[\varepsilon_n^2] = \varepsilon_n^2$ y diferenciando ε_n^2 con respecto a W:

$$\hat{\nabla}_n = \frac{\partial \varepsilon_n^{\,2}}{\partial W} \tag{1.19}$$

Donde $\hat{\nabla}_n$ es el estimado del gradiente. Resolviendo la derivada se tiene:

$$\hat{\nabla}_n = 2\varepsilon_n \frac{\partial \varepsilon_n}{\partial W} \tag{1.20}$$

Diferenciando la ecuación (1.17) se tiene

$$\frac{\partial \varepsilon_n}{\partial W} = \frac{\partial}{\partial W}\left(d_n - X_n{}^T W\right) = -X_n \tag{1.21}$$

De modo que el estimado del gradiente estará dado por:

$$\hat{\nabla}_n = 2\varepsilon_n[-X_n]$$

$$\hat{\nabla}_n = -2\varepsilon_n X_n \tag{1.22}$$

Reemplazando la ecuación (1.22) en la ecuación (1.8) se tiene:

$$W_{n+1} = W_n - \frac{\mu}{P_n}(-2\varepsilon_n X_n) = W_n + \frac{\beta}{P_n}\varepsilon_n X_n \tag{1.23}$$

Donde $\beta = 2\mu$

De acuerdo a esta última expresión la actualización del vector de coeficientes de ponderación W_{n+1}, se realiza añadiendo al vector de coeficientes de ponderación actual W_n, el vector de entrada escalado por el valor de la señal de error.

El algoritmo nLMS esta representado en realidad por la ecuación (1.23), la cual se usa directamente como una fórmula de adaptación del vector de coeficientes de ponderación.

1.4 DSP (DIGITAL SIGNAL PROCESSOR)

Actualmente los DSP se han convertido en elementos muy comunes en el diseño electrónico, sustituyendo en algunas aplicaciones a los microprocesadores y microcontroladores [10]. Los DSP son muy utilizados en circuitos relacionados con las telecomunicaciones, sistemas de audio y en algoritmos avanzados de control de motores. Por ejemplo:

- Tarjetas con múltiples puertos serie en servidores para proveedores de acceso a Internet
- Compresión de voz en telefonía móvil
- Filtros digitales adaptivos
- Generadores de eco
- Reconocimiento de señales DTMF (Dual Tone Multi Frequency)
- Decodificación de canales en telefonía celular (GSM **Global System for Mobile communications**)

Los DSP son sistemas programables que permiten implementar muchos tipos de aplicaciones en función de las posibilidades del sistema y, por supuesto, de las habilidades del programador.

Se puede decir que un DSP es un microprocesador optimizado internamente para realizar cálculos necesarios en el procesamiento de señales. Esta optimización se consigue mediante algunos aspectos principales:

- Implementación de operaciones por hardware
- Instrucciones poco comunes que ejecutan varias operaciones en un solo ciclo
- Modos de direccionamiento especiales
- Memoria de programa con más de 8 bits

La implementación de algunas operaciones mediante hardware consigue mejorar la velocidad media de cálculo, que se da en MIPS (Million Instructions Per Second).

1.4.1 ARQUITECTURA TÍPICA DE UN DSP

Los DSP abandonan la arquitectura clásica de Von Neumann, en la que datos y programas están en la misma zona de memoria, y apuestan por la denominada Arquitectura Harvard. En una arquitectura Harvard existen bloques de memoria físicamente separados para datos y programas. Cada uno de estos bloques de memoria se direcciona mediante buses separados (tanto de direcciones como de datos), e incluso es posible que la memoria de datos tenga distinta anchura de palabra que la memoria de programa (como ocurre en ciertos microcontroladores).

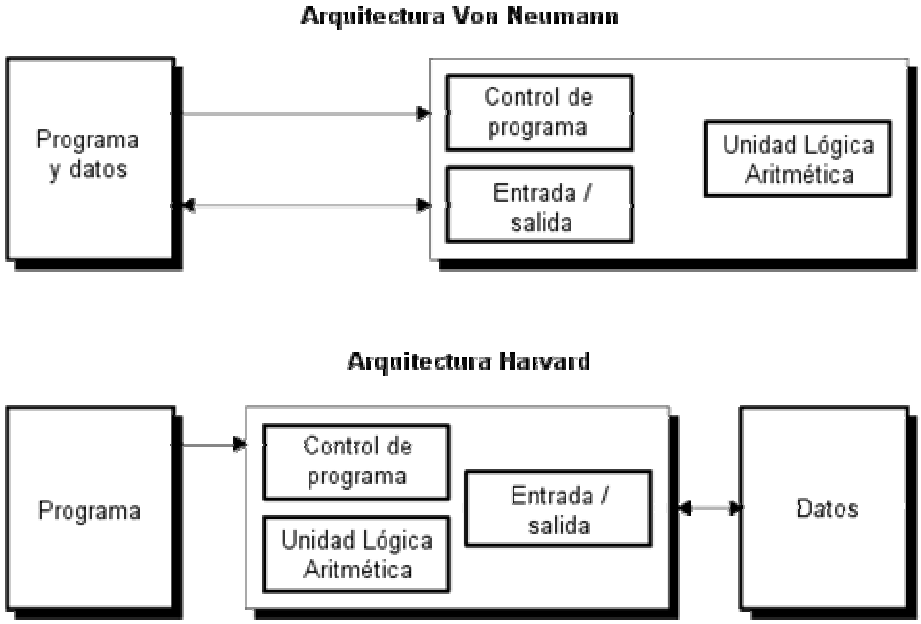


Figura 1.6 Diagrama de bloques de las arquitecturas Von Neuman y Harvard.

Con este diseño se consigue acelerar la ejecución de las instrucciones, ya que el sistema puede ejecutar simultáneamente la lectura de datos de la instrucción “n” y comenzar a decodificar la instrucción “n+1”, disminuyendo el tiempo total de ejecución de cada instrucción.

El ciclo de trabajo de un microprocesador clásico es:

- Leer la posición de memoria apuntada por el contador de programa
- Decodificar la instrucción
- Ejecutar la instrucción

En la ejecución de una instrucción se dan estos pasos:

- Leer los datos de memoria
- Operar con los datos
- Guardar los datos en la RAM

Por lo tanto, durante la lectura de los datos de una instrucción el bus está ocupado y no podría ser usado por otra unidad de decodificación, como aparece en muchos microprocesadores modernos, en los que se realizan simultáneamente la ejecución de la instrucción n y la decodificación de la instrucción n+1.

Normalmente en los DSP se usa una arquitectura Harvard modificada con tres buses: uno de programa y dos de datos, lo cual permite que la CPU (Central Processor Unit) lea una instrucción y dos operandos a la vez. En el procesamiento de señales, las operaciones con dos operandos son muy comunes, motivo por el cual se hace esta modificación.

1.4.2 UNIDADES ARITMÉTICO-LÓGICA DE LOS DSP

En el procesamiento digital de señales, una de las operaciones más comunes es el cálculo de sumas de productos, la multiplicación es una operación que es fácil de programar, aunque por su carácter iterativo es de una duración considerable.

En tareas normales, la multiplicación apenas supone el 1% del total de las operaciones, por lo cual los microprocesadores no suelen incorporar esta operación; pero en el procesamiento digital de señales es una de las operaciones principales, por lo cual se hace imprescindible la presencia de un multiplicador que la realice en el menor tiempo posible. Otra característica interesante de los DSP es la existencia de dos unidades aritmético-lógicas, una general y otra de tipo específico. Estas dos unidades son:

1. Unidad central aritmético – lógica (ALU, Aritmethic Logic Unit)
2. Unidad generadora de direcciones (AGU, Address Generation Unit)

La ALU se encarga de todos los cálculos, excepto los referentes a direcciones efectivas en direccionamiento indexado. En un microprocesador normal, las instrucciones que operan con datos indexados son las más lentas, ya que la ALU primero tiene que calcular la dirección efectiva del dato en cuestión, y luego operar con él. Puesto que en el procesamiento digital de señales es común trabajar con tablas, las operaciones que trabajen con direccionamiento indexado deben acelerarse cuanto se puedan. Para este fin, el DSP incorpora una segunda unidad aritmético-lógica, AGU, que se encarga solamente de hacer las sumas de la dirección base con el registro índice para obtener la dirección efectiva de nuestro dato y conseguir que la ALU principal no tenga que realizarla.

1.4.2.1 Unidad central Aritmético–Lógica (ALU)

Además de las instrucciones aritméticas habituales, la ALU puede realizar operaciones booleanas, facilitando la manipulación de bits que se usa para el trabajo con números enteros con signo.

La ALU de los DSP presenta la posibilidad de multiplicar datos. Realizar una multiplicación en lenguaje ensamblador (assembler code) consume bastante ciclos de reloj, debido a las sumas y desplazamientos que hay que hacer. Para incrementar la velocidad de trabajo y optimizar la implementación de los algoritmos de tratamiento digital de señal, la CPU de un DSP dispone de unidades computacionales específicas que pueden trabajar en paralelo. Entre ellas cabe destacar las unidades MAC (Multiply and Accumulate) ya que considerando la naturaleza de las operaciones que es preciso realizar en aplicaciones de tratamiento digital de señal, la operación básica es el producto acumulativo de dos secuencias.

El elemento básico de esta unidad MAC es el multiplicador hardware (multiplicador cableado) que es capaz de realizar el producto de dos números en un solo ciclo de máquina. La acumulación de los productos se realiza en la ALU, también en un solo ciclo de máquina. Para realizar la operación MAC ambos

elementos operan en paralelo, siguiendo un proceso segmentado en el que mientras el multiplicador realiza el producto de dos operandos la ALU acumula el resultado del producto anterior.

Esto queda puesto de manifiesto en aplicaciones como: filtrado, análisis espectral, correlaciones, entre otras. Los factores del producto pueden ser, dependiendo del caso, muestras de una señal, coeficientes de un filtro o constantes precalculadas (tablas de senos y cosenos).

Además, estas aplicaciones son tareas de tiempo real, por lo que el DSP tiene una arquitectura que permite realizar las operaciones MAC a gran velocidad.

Al repetir las operaciones de acumulación de los productos se puede dar una situación de desborde obteniéndose resultados erróneos. El desborde se debe a la imposibilidad de representar el resultado de salida con el número de bits disponibles. Para evitar esta situación se sobredimensiona el número de bits de la ALU, estos bits extras son denominados bits de guarda. Así por ejemplo, si el DSP opera con datos de 24 bits, entonces el producto de los operandos proporciona un resultado de 48 bits que será acumulado con los productos anteriores. Para evitar situaciones de desborde, se sobredimensiona la ALU añadiendo 8 bits de guarda, con lo que su tamaño final será de 56 bits. Este sobredimensionamiento de la ALU se aplica también al registro o registros donde se almacena el valor acumulado, como se indica en la Figura 1.7:

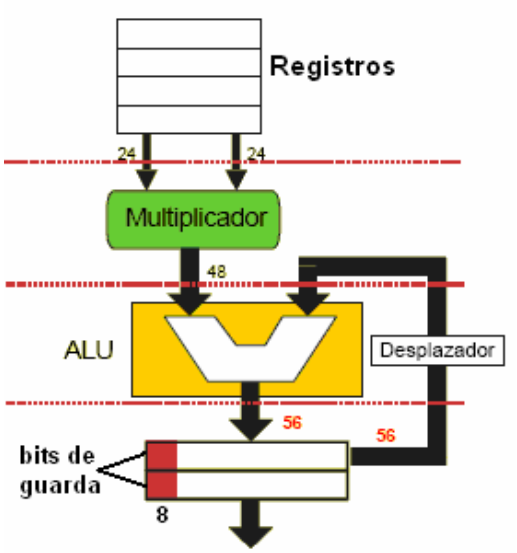


Figura 1.7 Esquema representativo de la función MAC [11].

Los modos de saturación están también presentes en la ALU y se refieren a la posibilidad de configurar el comportamiento del DSP cuando se produce un desbordamiento de bits en una operación aritmética.

Una situación de saturación se manifiesta cuando al sumar dos números positivos se obtiene un resultado negativo y viceversa. Los DSP utilizan técnicas de saturación aritmética incorporando circuitería que al detectar una situación de desborde durante la suma de dos números positivos sustituye la salida errónea por el máximo valor negativo a representar. El resultado sigue siendo incorrecto pero el error cometido en la operación es menor que en el caso de desborde, de esta forma se consigue un comportamiento similar al de un circuito analógico en saturación como se aprecia en la Figura 1.8. La saturación aritmética se puede realizar mediante una instrucción especial o automáticamente. En este último caso, esta facilidad se activa mediante uno de los bits el registro de estado (Status Register).

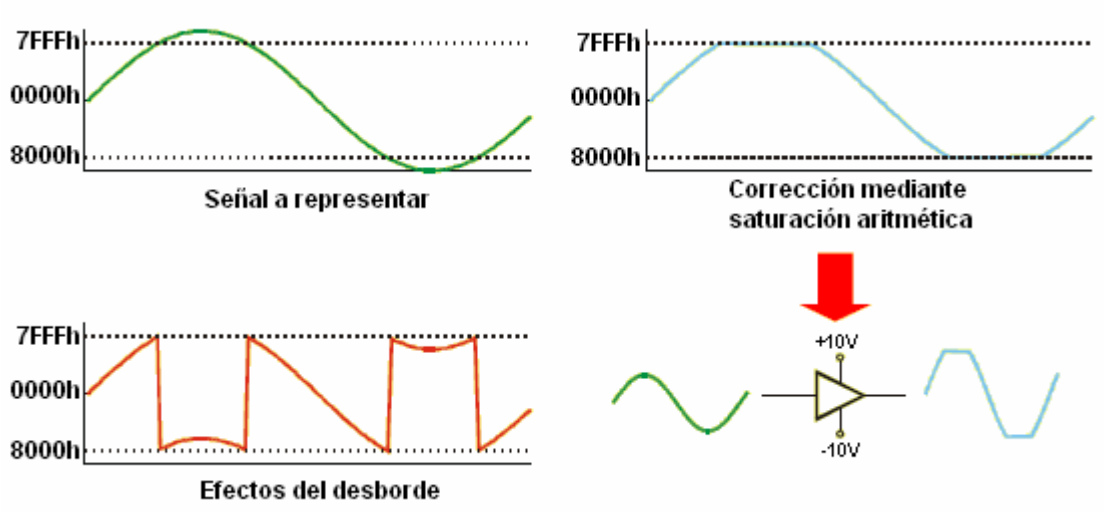


Figura 1.8 Saturación aritmética.

1.4.2.2 Unidad Generadora de Direcciones (AGU)

Muchas de las aplicaciones de tratamiento digital de señal necesitan gestionar un buffer con estructura FIFO donde se van almacenando las muestras que llegan del exterior o de cálculos anteriores. En la gestión del movimiento de los datos dentro y fuera del buffer el programador mantiene un puntero de lectura y otro de escritura, los cuales se suelen almacenar en registros de direcciones. El puntero de lectura apunta a la posición de memoria que almacena el próximo dato que se va a leer del buffer, mientras que el puntero de escritura apunta a la posición de memoria donde se va a almacenar el siguiente dato de entrada. Cada vez que se realice una operación de lectura o escritura, el puntero correspondiente avanza una posición, debiendo comprobar el programador si dicho puntero ha llegado a la última posición de memoria del buffer, inicializándose en tal caso para que apunte al comienzo del buffer. Esta comprobación de si un puntero ha llegado al final del buffer y su inicialización en caso afirmativo consume tiempo [11].

Para solventar este problema las unidades AGU de muchos DSP realizan de forma automática la comprobación del puntero y su ajuste relativo a la dirección de comienzo, si es preciso, después de cada cálculo de dirección. Esta característica se denomina aritmética modular. El término aritmética modular hace referencia a que el resultado de salida se limita a un determinado rango, de

manera que si una operación de suma o resta supera tal rango, se obtendría el valor inicial de dicho rango. Cabe señalar que la AGU de muchos DSP permite implementar módulos de direccionamiento especialmente pensados para la realización de la transformada discreta de Fourier de una forma rápida.

La Figura 1.9 indica el funcionamiento de los punteros de lectura y escritura de la AGU:

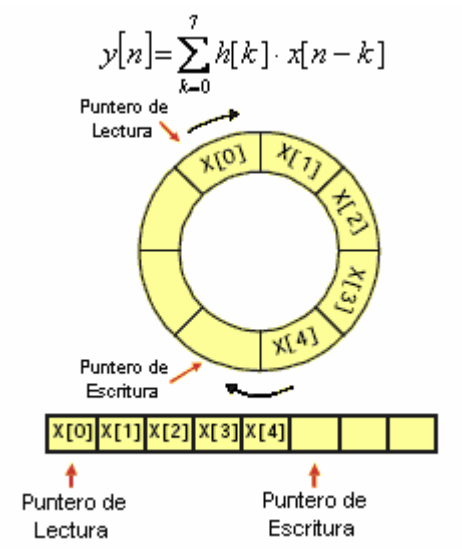


Figura 1.9 Unidad Generadora de Direcciones (AGU).

1.5 COMUNICACIÓN CON MEDIOS EXTERNOS

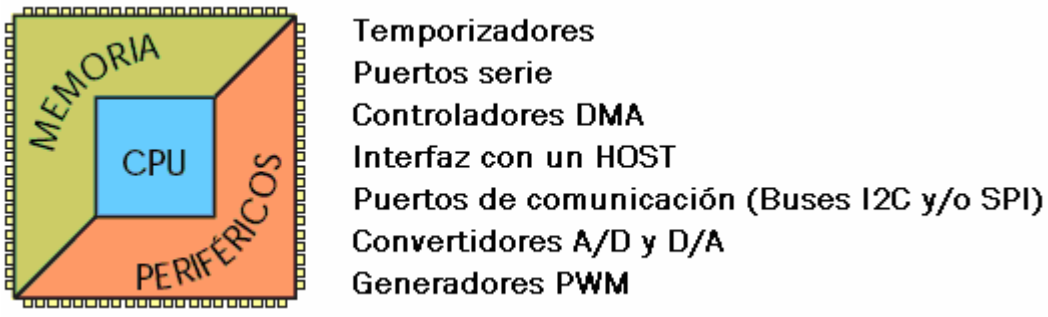


Figura 1.10 Comunicación del DSP con medios externos.

Una característica bastante común de los DSP es su adaptación a diversas aplicaciones por parte de los fabricantes [11], añadiendo diversas características

adicionales y periféricos, de modo que un tipo determinado de DSP sea la respuesta casi perfecta a una necesidad concreta. En los DSP dedicados a tareas de control, se puede encontrar los periféricos típicos de los microcontroladores, y otros no tan comunes. En concreto, se hallan periféricos tales como:

- Temporizadores
- Puertos serie
- Controladores DMA (Direct Memory Access)
- Interfaz con un HOST
- Puertos de comunicación (Buses I2C y/o SPI)
- Convertidores A/D y D/A
- Generadores PWM (Pulse Width Modulate)

1.5.1 TEMPORIZADORES

Normalmente, todos los procesadores incluyen estos dispositivos, que suelen admitir como entrada un reloj interno o externo. Se suelen utilizar para la generación de interrupciones periódicas que sincronizan el proceso de muestreo.

1.5.2 PUERTOS SERIE

Las características y complejidad de estas interfaces varían de un procesador a otro, pero es habitual su presencia en todos ellos. Se usan porque necesitan menos bits que los paralelos:

- Dato
- Reloj
- Sincronización (inicio de palabra)

Suelen utilizarse para comunicarse con circuitos que incluyen conversores A/D y D/A denominados “CODEC”. Esto permite alejar la parte analógica del procesador eliminando el acoplamiento de ruido en la parte analógica. De hecho la mayoría de los fabricantes de DSP diseñan dispositivos de este tipo con un interfaz que permite la conexión directa con el puerto serie.

1.5.3 CONTROLADOR DMA

Este tipo de periférico permite efectuar transferencias en forma rápida sin intervención del procesador. Esto es especialmente útil en aplicaciones en las que el volumen de datos a manejar es considerable como por ejemplo el procesamiento digital de imagen. En tal caso, el DMA se utiliza para llevar bloques de datos desde una memoria externa de gran capacidad a memoria interna, con el propósito de que estos se puedan procesar más rápido.

1.5.4 INTERFAZ CON UN HOST

En algunos casos los DSP actúan como coprocesadores matemáticos, formando parte de un sistema global controlado por un procesador de propósito general. Para comunicarse con este procesador host algunos DSP incorporan un puerto paralelo bidireccional de 8 o 16 bits, que podría incluso estar diseñado específicamente para comunicarse con un bus estándar, como por ejemplo ISA (Industry Standard Architecture) o PCI (Peripheral Component Interconnect). Las comunicaciones a través de este puerto suelen realizarse mediante DMA de manera que es posible transferir datos entre memoria y el puerto sin intervención del procesador. El control del puerto se hace mediante instrucciones específicas. En ocasiones esta interfaz permite incluso el control del funcionamiento del DSP, como forzar la ejecución de rutinas de tratamiento de interrupción, acceder a los registros internos del DSP o a su memoria, o incluso realizar la carga del código a ejecutar por el procesador (bootstrapping[2]).

1.5.5 PUERTOS DE COMUNICACIÓN

Se trata de puertos paralelos diseñados para comunicación entre DSP del mismo tipo que se conectan en red para implementar un sistema multiprocesador. Puesto

[2] La palabra inglesa bootstrapping es generalmente un término más extenso para el arranque, o proceso de inicio de cualquier ordenador.

que el ancho de estos puertos (8 bits) es menor a la palabra de datos del DSP, los puertos disponen de estructuras FIFO (First In First Out) para la fragmentación y reensamblado de los datos que se transmiten a través de ellos. La comunicación de estos puertos suele estar asistida por DMA.

Los DSP diseñados para aplicaciones muy concretas como el control de motores o sistemas de potencia disponen, además, de periféricos específicos para tales aplicaciones como generadores de señal PWM, temporizadores especiales para la implementación de tacómetros digitales, conversores A/D, entre otros.

1.6 PARÁMETROS DE SELECCIÓN DEL DSP ADECUADO

El DSP adecuado para cada tarea depende enormemente de esa tarea [12]. Por ejemplo, un procesador que desarrolla bien ciertas tareas puede ser una pobre elección para otras. Se puede considerar ciertas características que varían de un DSP a otro a la hora de elegir un procesador.

Una forma de clasificar los dispositivos DSP y sus aplicaciones es por su rango dinámico. Se denomina rango dinámico al conjunto de valores, entre el menor y el mayor, que puede ser procesado en el curso de una operación. Esto ha de proporcionar un conjunto de valores para describir por completo una forma de onda señalada, desde el mínimo más profundo hasta la oscilación más alta. El rango ha de ser más amplio que el requerido para los cálculos, ya que se irán generando valores mayores y menores a partir de las multiplicaciones y divisiones.

Un DSP de 32 bits tiene un rango dinámico mayor que uno de 24 bits, y este a su vez, mayor que uno de 16 bits. Los chips de coma flotante tienen rangos dinámicos más amplios que los dispositivos de coma fija. Cada tipo de procesador es ideal para un rango específico de aplicaciones.

Los DSP de 24 bits de coma fija, como por ejemplo la familia de Motorola DSP56300 son buenos para sistemas de voz, ya que estos DSP trabajan con el rango relativamente estrecho de las frecuencias del sonido. El DSP debe ser capaz de manejar los números generados tanto en la transformación análoga–digital como durante los cálculos (multiplicaciones, sumas, divisiones) con dicha señal. Si no es capaz de manejar todo el rango de números ocurrirá overflow[3] o underflow[4], lo cual producirá errores en los cálculos.

1.6.1 FORMATO ARITMÉTICO

Una de las características más fundamentales de los procesadores digitales programables es el tipo de aritmética utilizada por el procesador. La mayor parte de los DSP usan aritmética de coma fija, donde los número se representan como enteros o como fracciones entre -1.0 y +1.0. Otros procesadores usan aritmética de coma flotante, donde los valores se representan por una mantisa y un exponente. La mantisa generalmente es una fracción con rango entre -1.0 y +1.0, mientras el exponente es un entero que representa en binario el número de lugares a partir de la coma que se debe desplazar a izquierda o derecha para obtener el valor representado.

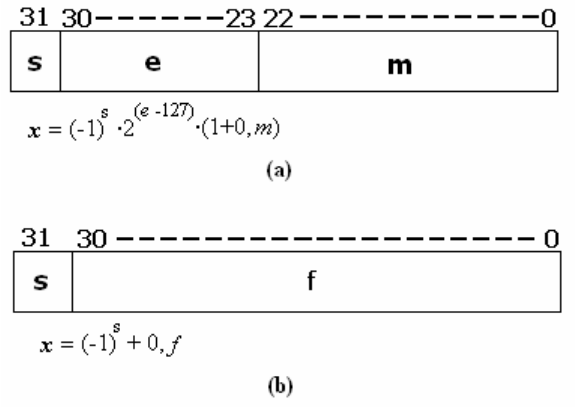


Figura 1.11 (a) Formato de coma flotante; 1 bit de signo, 8 de exponente y 23 de mantisa. (b) Formato en coma fija; 1 bit de signo y 31 bits significativos.

---

[3] **Overflow,** ocurre cuando se sobrepasa la capacidad del acumulador o de algún registro.

[4] **Underflow,** ocurre cuando hay una petición de dato, desde el acumulador o algún registro, y este no esta disponible.

La Figura 1.11 muestra los formatos de coma flotante y coma fija, respectivamente. En el formato de coma flotante la “s” indica que el bit más significativo es el signo, donde un 1 indica que se trata de un número negativo. La “e” indica exponente, formado por 8 bits y la “m”, de 23 bits, la mantisa del número.

Al carecer de exponente el formato en coma fija, éste puede representar números con más bits significativos que el formato en coma flotante del mismo tamaño en bits. En este ejemplo, 31 bits son significativos, “f”, comparados con los 23 del formato de coma flotante.

La aritmética en coma flotante es mucho más flexible que la de coma fija. En coma flotante, los diseñadores de sistemas tienen acceso a un rango dinámico más amplio (la distancia entre mayor y el menor valor que puede representar). Como resultado, los DSP de coma flotante son generalmente más fáciles de programar que sus correspondientes de coma fija, pero normalmente más caros.

El incremento del costo se debe a la compleja circuitería necesaria para realizar los procesos en coma flotante. En coma flotante el programador no necesita conocer en muchos casos ni el rango dinámico ni la precisión, mientras que, en coma fija, los programadores han de ser cuidadosos asegurándose de que sus señales no excedan el rango dinámico.

Los procesadores de coma fija, se usan en muchas aplicaciones debido a su bajo costo. En estas aplicaciones son necesarios programas y algoritmos diseñados para determinar el rango dinámico y la precisión. En las aplicaciones en las que el costo es poco importante o bien es necesario un amplio rango dinámico, se utilizan los procesadores en coma flotante.

1.6.2 ANCHO DE DATOS

El tamaño del dato tiene una mayor repercusión en el costo, ya que influye notablemente en el tamaño del chip y el número de pines que requiere, así como el tamaño de los dispositivos externos conectados al DSP. Por todo esto, los diseñadores tratan de emplear anchos de palabra lo más pequeño posible de acuerdo a sus necesidades.

Como en el caso de la elección entre coma fija y coma flotante, hay que buscar el equilibrio entre el tamaño de palabra y la complejidad de desarrollo. Por ejemplo, un procesador de 16 bits puede llevar a cabo operaciones aritméticas de 32 bits en doble precisión a través de una combinación adecuada de instrucciones. Por supuesto la aritmética de doble precisión es mucho más lenta que la precisión simple. Si el volumen de una aplicación puede manejarse con aritmética de precisión simple, pero la aplicación necesita mas precisión para una sección pequeña del código, se emplea selectivamente la aritmética de doble precisión.

1.6.3 VELOCIDAD

Se puede obtener una medida de que tan adecuado es un procesador para una determinada tarea a través de su velocidad de ejecución. Hay muchas formas de medir la velocidad de ejecución de los procesadores. Una opción es medir el tiempo de ciclo de instrucción: la cantidad de tiempo empleado en ejecutar la instrucción más rápida del procesador. El inverso de este tiempo dividido por un millón es la velocidad de ejecución del procesador en millones de instrucciones por segundo o MIPS.

Un problema al comparar tiempos de ciclos de ejecución es que la cantidad de trabajo terminado por una instrucción varía mucho de un procesador a otro. Por ejemplo, algunas de las características de desplazamiento circular soportada por los DSP requieren que el dato sea desplazado con instrucciones de desplazamiento de un bit repetidas. Así mismo, algunos DSP permiten

movimientos en paralelo de datos independientemente de la instrucción que ejecutemos en la ALU, mientras que otros solo soportan movimientos en paralelo que estén relacionados con los operandos de la instrucción de la ALU.

Para resolver este problema, se toma una operación básica en lugar de una instrucción, y se usa como punto de referencia al comparar. Una operación común es la operación MAC, aunque desafortunadamente proporciona poca información para diferenciar procesadores, ya que en la mayor parte de los DSP modernos una operación MAC se ejecuta en un solo ciclo de instrucción, y algunos DSP son capaces de realizar mucho más en una instrucción MAC que otros. Además los tiempos de MAC no reflejan la implementación de otras operaciones importantes como los bucles.

1.6.4 SEGMENTACIÓN (PIPELINING)

Pipelining es una técnica para incrementar las prestaciones de un procesador, que consiste en dividir una secuencia de operaciones en otras más sencillas y ejecutar en lo posible cada una de ellas en paralelo. En consecuencia se reduce el tiempo total requerido para completar un conjunto de operaciones. Casi todos los DSP del mercado incorporan el uso de la segmentación en mayor o menor medida. Para ilustrar de qué forma la técnica de la segmentación mejora las prestaciones de un procesador, considérese un hipotético procesador que utiliza unidades de ejecución separadas para una única instrucción:

- Obtención de la instrucción de la memoria
- Decodificar la instrucción
- Leer o escribir un operando de la memoria
- Ejecutar la parte de la instrucción relacionada con la ALU o MAC.

Tabla 1.4 Ejecución de instrucciones sin pipeline. I1 y I2 representan dos instrucciones diferentes

	Ciclo de reloj							
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Obtención instrucción	I1				I2			
Decodificación		I1				I2		
Lectura/Escritura operando			I1				I2	
Ejecución				I1				I2

La Tabla 1.4 muestra los ciclos de reloj empleados por varias instrucciones ejecutadas de forma secuencial. Si se supone que cada etapa o unidad de ejecución tarda 20ns en ejecutar su parte de la instrucción, entonces el procesador ejecuta una instrucción cada 80 ns. Sin embargo, también se observa que el hardware asociado a cada etapa de ejecución está inactivo el 75% del tiempo. Esto ocurre porque el procesador no empieza a ejecutar una nueva instrucción hasta que finaliza la ejecución de la instrucción en curso. Un procesador que implementa la técnica de pipelining, obtendría una nueva instrucción inmediatamente después de haber obtenido la anterior. De forma similar, cada instrucción sería descodificada después de haber terminado la decodificación de la instrucción anterior. Con esta filosofía, las instrucciones se ejecutan de forma solapada, tal y como se indica en la Tabla 1.5.

Tabla 1.5 Procesador que utiliza la técnica del pipelining

	Ciclo de reloj							
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Obtención instrucción	I1	I2	I3	I4	I5	I6	I7	I8
Decodificación		I1	I2	I3	I4	I5	I6	I7
Lectura/Escritura operando			I1	I2	I3	I4	I5	I6
Ejecución				I1	I2	I3	I4	I5

Las unidades de ejecución trabajan en paralelo, mientras una obtiene el código de una instrucción, otra está decodificando la anterior y así sucesivamente. En consecuencia, una vez que la pipeline está llena, cada 20ns se ejecuta una

instrucción, lo cual representa un factor de mejora de prestaciones respecto a un procesador que no incorpore dicha técnica.

Aunque la mayoría de los DSP utilizan la técnica de segmentación, su profundidad o número de etapas varía de un procesador a otro. En general, cuanto mayor sea el número de etapas menor tiempo tardará el procesador en ejecutar una instrucción.

En el ejemplo anterior se ha supuesto un procesador con una eficiencia en el uso de la pipeline del 100%. En realidad, esto no siempre ocurre así. La eficiencia se ve disminuida por varias causas, entre las cuales se encuentra el hecho de que un procesador necesite dos ciclos para escribir en memoria, se obtenga el código de una instrucción de salto de programa o bien la petición de una interrupción. La Tabla 1.6 muestra qué es lo que pasa cuando una instrucción de salto llega a la pipeline.

Tabla 1.6 Efecto en la pipeline ante la llegada de una instrucción de salto

	Ciclo de reloj							
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Obtención instrucción	Salto	I2	-	-	N1	N2	N3	N4
Decodificación		Salto	-	-	-	N1	N2	N3
Lectura/Escritura operando			Salto	-	-	-	N1	N2
Ejecución				Salto	NOP	NOP	NOP	N1

En el momento en que el procesador detecta la llegada de una instrucción de salto en la decodificación del segundo ciclo de reloj, la pipeline se vacía y detiene la obtención de nuevas instrucciones. Esto provoca que la instrucción de salto se ejecute en cuatro ciclos. Posteriormente, en el quinto ciclo de reloj, el procesador comienza la obtención de las instrucciones (N1…N4) que se encuentran a partir de la dirección de salto. A causa de este tipo de situaciones, casi todos los DSP incorporan algún tipo de mejora en el uso de la segmentación con el propósito de reducir su posible ineficiencia temporal.

1.6.5 POTENCIA

Los DSP cada vez se usan más en aplicaciones portátiles como teléfonos móviles donde el consumo se convierte en una característica importante. Por esto, la mayoría de los fabricantes de DSP han reducido los voltajes de alimentación de los procesadores e incluyen algunas características que permiten al programador reducir el consumo, algunas de estas características son las siguientes:

- Reducción del voltaje
- Modos "sleep" o "Idle"
- Divisores de reloj programables

1.6.5.1 Reducción del voltaje

Los fabricantes han introducido versiones de baja potencia (3.3 ó 3.0 V) de sus DSP. Estos procesadores consumen aproximadamente un 40% que sus equivalentes de 5 V a la misma frecuencia de reloj.

1.6.5.2 Modos "sleep" o "Idle"

Estos modos permiten desconectar el reloj del procesador, excepto de ciertas secciones del procesador, reduciendo el consumo. En algunos casos se sale de este estado mediante una interrupción no enmascarable, en otros sólo se sale a través de alguna línea externa de interrupción.

1.6.5.3 Divisores de reloj programables

Algunos DSP actuales permiten variar la frecuencia del reloj por software para usar la mínima frecuencia de reloj para cada tarea.

1.6.6 COSTO

Generalmente el precio del DSP es el principal parámetro en todos aquellos productos que se van a fabricar en grandes volúmenes. En esos casos, el diseñador intenta utilizar el DSP con costo inferior y que satisfaga las necesidades de la aplicación, aún cuando ese dispositivo pueda ser considerado poco flexible y más difícil de programar que otros DSP más caros. De entre las familias de DSP, el más barato será aquel que tenga menos características funcionales, menos memoria interna y probablemente menos prestaciones que otro más caro. Sin embargo, una diferencia clave en el precio está en el encapsulado.

Los encapsulados PQFP (Plastic Quad Flat Pack) y TQFP (Thin Quad Flat Pack) son usualmente bastante más baratos que los PGA (Pin Grid Array).

1.6.7 SOPORTE TÉCNICO Y FACILIDAD DE DESARROLLO

En el momento de elegir un DSP u otro será necesario conocer completamente los requisitos de procesado del sistema. Muchos DSP pueden ser eliminados previamente con sólo tener en cuenta consideraciones de falta de potencia de cálculo, resolución insuficiente, costo, entre otros.

Esto probablemente deje todavía a un número de posibles candidatos para los cuales será preciso realizar otro tipo de análisis. Hasta el momento se han visto aquellas características más técnicas de los DSP y que están estrechamente relacionadas con los algoritmos de la aplicación a implementar. Sin embargo, no se han considerado para nada aspectos relacionados con el desarrollo de la aplicación. El DSP que finalmente se elija deberá disponer de un amplio conjunto de herramientas de desarrollo.

Algunos requerimientos básicos son:

- Documentación de diseño detallada

- Herramientas de desarrollo de código en ensamblador y/o en lenguaje de alto nivel
- Herramientas para el test de la funcionalidad del diseño
- Notas de aplicación u otro tipo de ayuda al diseño

El objetivo será seleccionar el DSP que permita terminar el proyecto en el tiempo previsto y que la solución alcanzada sea la que presente la mejor relación costo-eficiencia. En aplicaciones de gran volumen de producción, esto probablemente signifique que el DSP escogido será el más barato que pueda realizar la aplicación. Para aplicaciones con un volumen bajo-medio existirá el compromiso entre el costo de las herramientas de desarrollo y el costo y eficiencia del DSP. En cambio, para aplicaciones con un volumen bajo de producción tendrá más sentido utilizar un DSP que facilite el diseño o que tenga las herramientas de desarrollo más baratas.

Cabe la posibilidad que la elección del DSP sea un proceso iterativo. En otras palabras, puede no haberse escogido el dispositivo correcto. Podría ser que aparecieran problemas imprevistos en la fase de desarrollo y prueba del código o incluso que se encontrara que un DSP más barato y menos potente pudiera ser el elegido. Comúnmente, las especificaciones del diseño alterarán y forzarán a replantear la solución escogida. Los dos primeros casos pueden evitarse haciendo más minuciosa la búsqueda del DSP que se adapte a la aplicación en particular. Algunas veces merece la pena la compra de herramientas de desarrollo tales como los simuladores para algunos DSP y ejercitar el código antes de comprometerse a un solo DSP.

1.7 CAMPO DE USO DE LOS DSP

Después de analizar las arquitecturas de los dispositivos, se ha obtenido una visión general del abanico de aplicaciones que soportan los DSP [13]. Por ejemplo:

- En el campo militar se utilizan los DSP para procesamiento de radar, sonar o guía de misiles
- En el campo del tratamiento de voz y audio para la codificación, síntesis y reconocimiento de voz
- En el sector de las telecomunicaciones para cancelación de eco y en la telefonía móvil

Algunos de los avances de los DSP en instrumentación médica son las imágenes ultrasónicas, radiografías digitales y varias formas de tomografía:

- CAT (Computer Aided Tomography), tomografía asistida por ordenador
- PET (Positron Emision Tomography), tomografía por emisión de positrones
- MRI (Magnetic Resonance Image), imágenes por resonancia magnética

1.8 VENTAJAS Y DESVENTAJAS DE LOS DSP

Entre las principales ventajas de los DSP se destacan las siguientes [14]:

- Los DSP tienen la capacidad de procesar en tiempo real muchas de las señales de interés para aplicaciones en comunicaciones, control, procesamiento de imagen, multimedia, entre otros
- Los sistemas digitales son más confiables que los correspondientes sistemas análogos
- Los sistemas digitales ofrecen una mayor flexibilidad en la reconfiguración de aplicaciones que los correspondientes sistemas análogos
- Mayor precisión y mayor exactitud pueden ser obtenidas con sistemas digitales, comparando con los correspondientes sistemas análogos
- La tolerancia de los componentes en un sistema análogo hacen que esto sea una dificultad para el diseñador al controlar la exactitud de la señal de salida análoga. Por otro lado, la exactitud de la señal de salida para un sistema digital es predecible y controlable por el tipo de aritmética usada y el número de bits usado en los cálculos

A pesar de ellas existen algunos inconvenientes que deberán ser tomados en cuenta al momento de escoger una plataforma para el procesamiento de señales analógicas por medio digitales:

- La conversión de una señal analógica en digital, obtenida muestreando la señal y cuantificando las muestras, produce una distorsión que impide la reconstrucción de la señal analógica original a partir de muestras cuantificadas
- Existen efectos debidos a la precisión finita que deben ser considerados en el procesado digital de las muestras cuantificadas
- Para muchas señales de gran ancho de banda, se requiere procesado en tiempo real. Para tales señales, el procesado analógico, o incluso óptico, son las únicas soluciones válidas. Sin embargo, cuando los circuitos digitales existen y son de suficiente velocidad se hacen preferibles.

En este Capítulo se ha descrito los principales tipos de filtros digitales, así como también los algoritmos adaptivos más utilizados, optando para el desarrollo de este Proyecto de Titulación por la implementación del filtro digital adaptivo nLMS.

CAPÍTULO 2

DISEÑO DEL ALGORITMO Y HARDWARE A IMPLEMENTAR

La estructura típica de un sistema para tratamiento digital de señal responde al diagrama de bloques indicado en la Figura 2.1:

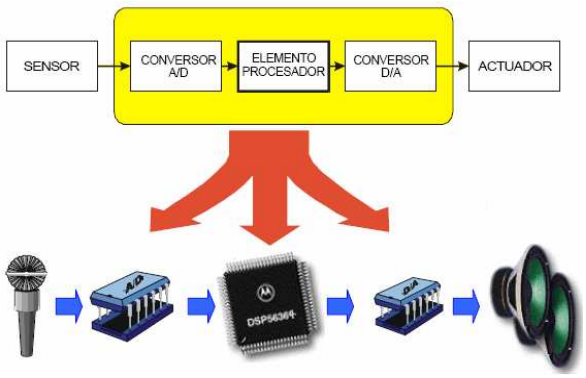


Figura 2.1 Sistema digital para procesamiento de señales

Un conversor A/D convierte la señal analógica en una secuencia numérica. Esta señal analógica suele obtenerse de un sensor que transforma una magnitud física en una señal eléctrica. Estas muestras llegan a un elemento procesador en el que se implementa el algoritmo de tratamiento digital. La salida de este procesador se introduce a un conversor D/A para obtener de nuevo una señal analógica. Esta señal analógica de salida se transforma de nuevo en una magnitud física por medio de un actuador [11].

En un inicio estaba prevista la utilización de los conversores A/D y D/A por separado pero se encontró un circuito integrado denominado Codec, el cual es óptimo para aplicaciones de audio, además tiene una interfaz sencilla diseñada para trabajar con los DSP.

Como elemento procesador se opto por la utilización de un DSP, pues este es apropiado para trabajar en aplicaciones en tiempo real. Basándose en los criterios

de elección de un DSP indicados en el Capítulo anterior se opto por un DSP de 24-bits y formato aritmético de coma fija.

El DSP seleccionado cumple con todos los requisitos para la ejecución del algoritmo a implementar en este proyecto, pero este DSP solo dispone de memoria RAM interna volátil, lo que hace necesario la utilización de una memoria externa no volátil. Esta memoria tiene la función de almacenar el programa a ser ejecutado por el DSP permitiendo trabajar de esta manera al DSP en modo independiente (stand-alone mode).

Para permitir la comunicación entre los módulos que conforman el proyecto se utiliza dos transceivers full-duplex, los cuales transmiten y reciben la señal procesada.

En este capítulo se incluyen todos los diagramas de bloques del hardware a implementar en este proyecto de tesis. Además al analizar cada uno de estos elementos y justificar su diseño se explica la función que desempeñan.

2.1 DESCRIPCIÓN DEL ALGORITMO

En el Capítulo anterior se describió los principales tipos de filtros digitales, así como también los algoritmos adaptivos más utilizados, para esta aplicación se ha optado por la implementación del filtro digital adaptivo nLMS.

El filtrado adaptivo es muy utilizado en casos en los que la señal de voz está inmersa en un ambiente muy ruidoso. En este caso, el esquema general es el que se indica en la Figura 2.2 [8]:

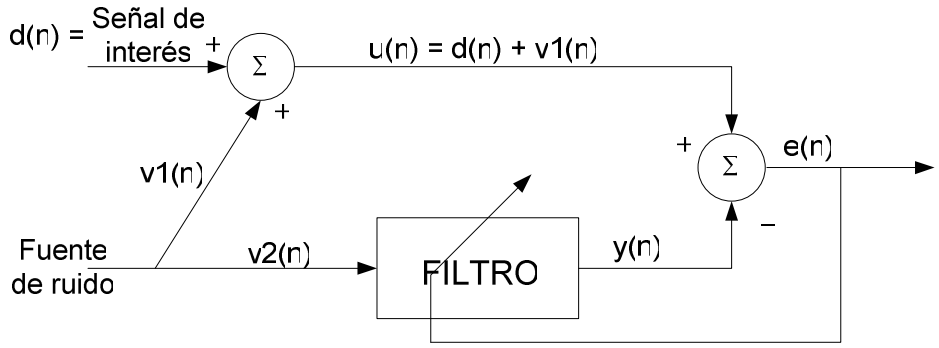


Figura 2.2 Esquema del filtro digital adaptivo.

Como se muestra en la Figura 2.2, la entrada primaria u(n) contiene la señal más el ruido, donde v2(n) es la referencia del ruido, un filtro adaptivo es usado para estimar el ruido de u(n) y el ruido estimado y(n) se resta de la entrada primaria. La salida del cancelador de ruido es entonces e(n).

De lo anteriormente dicho en las ecuaciones (1.12) y (1.23) se tiene:

$$y(n)=\sum_{k=0}^{M-1} W_k(n)\,v2(n-k) \tag{2.1}$$

$$W(n+1)= W(n)+\frac{\beta}{P(n)}\,e(n)v2(n) \tag{2.2}$$

Donde: e(n) = u(n) – y(n)

Reemplazando $(1-\gamma)=\frac{1}{M}$ en la ecuación (1.9), correspondiente al estimador de potencia, se tiene:

$$P(n+1)=\left(1-\frac{1}{M}\right)P(n)+\frac{1}{M}\;v2(n)^2 \tag{2.3}$$

Donde: $v2(n)^2$=v2'(n) * v2(n)

Entonces, el algoritmo nLMS a ser implementado está dado por las ecuaciones (2.1), (2.2) y (2.3).

Idealmente, el sistema elimina el ruido por completo, pero, en la práctica, solo es reducido considerablemente. Esto es muy utilizado en los aviones militares, para mejorar la inteligibilidad en las comunicaciones por radio, ya que, en estado normal, el micrófono del piloto capta su voz, pero también capta ruido. Para cancelar ese ruido (o al menos reducirlo), se usa como referencia de ruido un segundo micrófono para que tan sólo capte el ruido. Como se indica en la Figura 2.3:

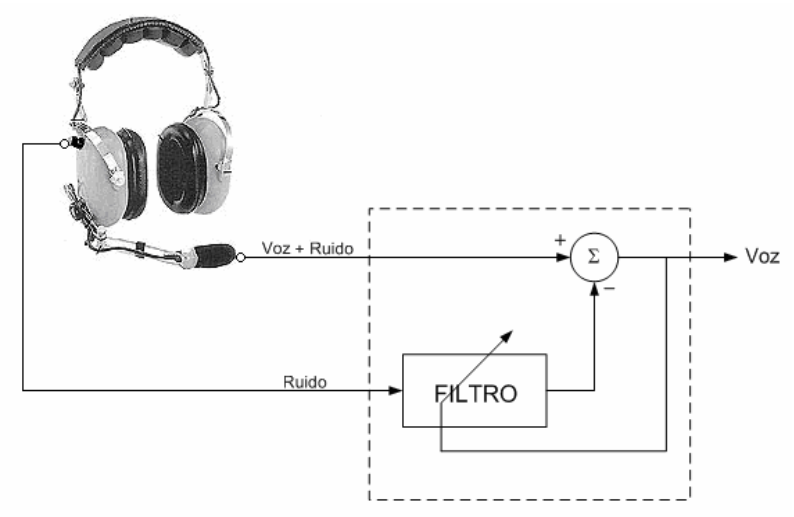


Figura 2.3 Esquema de las entradas del filtro nLMS

El DSP56303 es el encargado de ejecutar el filtro adaptivo nLMS, el cual al estar interconectado con los otros elementos que conforman el proyecto, procesará las señales de entrada.

2.2 HARDWARE IMPLEMENTADO

Como se indico anteriormente, las señales de entrada son obtenidas mediante dos micrófonos, uno capta la voz contaminada con ruido, el otro el ruido que se tomará como referencia, estas señales se amplifican con la ayuda del LM386 [19], el cual es un amplificador de bajo consumo, ideal para aplicaciones de audio. Las salidas de esta etapa de amplificación son ingresadas al Codec CS4218 [15], aprovechando la característica estéreo del Codec el ruido ingresa por el canal

izquierdo, mientras que la señal contaminada con ruido (voz + ruido) ingresa por el canal derecho. Como se observa en la Figura 2.4:

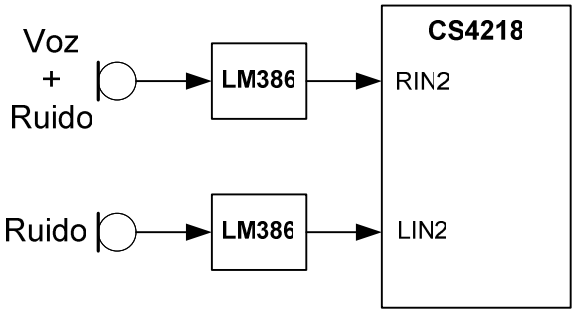


Figura 2.4 Esquema de las señales de entrada al Codec.

El Codec tiene en su interior conversores A/D y D/A, entonces, las señales de entrada son digitalizadas, para que el DSP se encargue del procesamiento de las mismas. Dichas señales procesadas ingresan nuevamente al Codec, pero esta vez se utiliza el conversor D/A. La señal análoga resultante es transmitida por el transceiver EWM-900-FDTC [16]. Para que la transmisión y recepción tengan resultado los transceivers vienen en parejas; uno se denomina BS (base station) y el otro HS (handset), así, de esta manera, la frecuencia programada para el canal de transmisión del BS debe ser la misma que el canal de recepción de HS y viceversa.

Además de los elementos ya descritos se utiliza una memoria Flash EEPROM AT29LV010A [17], la cual guarda el programa a ser ejecutado por el DSP56303. Este programa es cargado desde un computador a través de un Command Converter[5].

Lo anteriormente descrito se aprecia mejor en la Figura 2.5:

[5] **Command Converter,** es el encargado de cambiar los niveles de voltaje TTL del computador en niveles de voltaje CMOS aceptados por el DSP

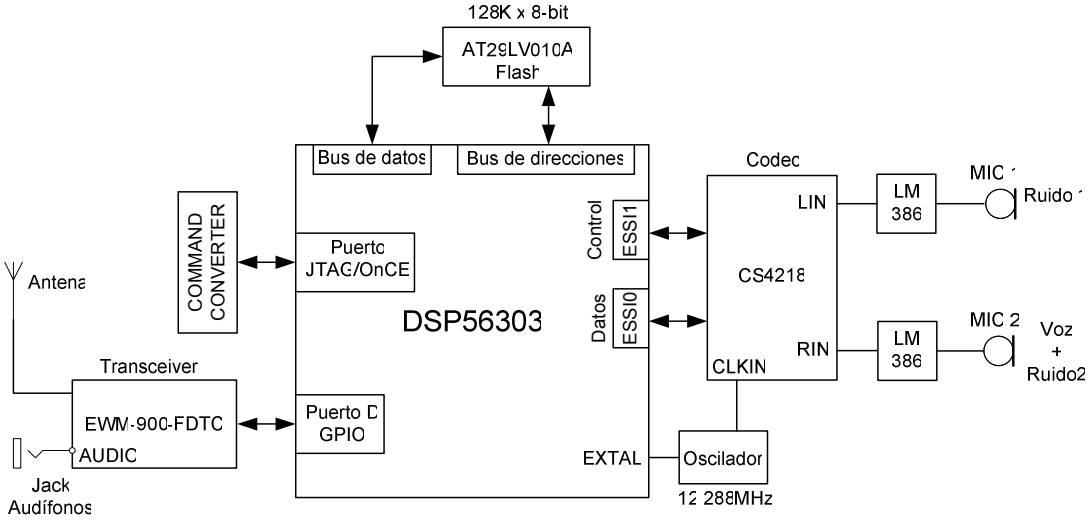


Figura 2.5 Esquema general de conexiones.

El hardware esta conformado por varias etapas:

- Etapa de amplificación
- Etapa de procesamiento de señal
- Etapa de transmisión

2.2.1 ETAPA DE AMPLIFICACIÓN

El núcleo de esta etapa es el circuito integrado LM386, que se encarga de amplificar la señal proveniente de los micrófonos monoaurales, el LM386[6] es un amplificador diseñado para el uso en aplicaciones de bajo consumo, la ganancia interna es de 20 reduciendo la necesidad de elementos externos, pero la adición de un capacitor y una resistencia externos entre los pines 1 y 8 pueden incrementar la ganancia a cualquier valor comprendido entre 20 y 200.

La entrada debe estar referida a tierra, mientras que la salida automáticamente es polarizada a la mitad del voltaje de alimentación. El consumo en estado pasivo es de 24mW cuando opera con una fuente de alimentación de 6V, lo que hace al LM386 ideal para el uso con baterías.

---

[6] Tomado de "*LM386 Low Voltaje Audio Power Amplifier* ", Aug. 2000, pp.1

Las principales características de operación del LM386 son:

- Puede operar con baterías
- Uso mínimo de elementos externos
- Voltajes aplicables en los rangos: 4V-12V o 5V-18V
- Bajo consumo de corriente en estado pasivo: 4mA
- Ganancia de voltaje entre 20 y 200
- Baja distorsión: 0.2% (Av = 20, Vs = 6V, R_L = 8Ω, P_O = 125mW, f = 1KHz)

Entre las aplicaciones más comunes para este circuito integrado están:

- Amplificadores de radio AM-FM
- Amplificadores de toca cintas portátiles
- Intercomunicadores
- Sistemas de sonido de TV
- Manejadores ultrasónicos (Ultrasonic Drivers)

2.2.2 ETAPA DE PROCESAMIENTO DE SEÑAL

Los DSP trabajan con señales del mundo real, incluyendo aquellas como sonido y ondas de radio que se originan en forma análoga. Como se sabe, una señal análoga es continua en el tiempo; cambia suavemente desde un estado a otro. Los computadores digitales, por otro lado, manejan la información discontinuamente, como una serie de números binarios, por lo que se hace necesario como primera etapa en la mayoría de los sistemas basados en DSP transformar las señales análogas en digitales. Esta transformación la hacen los Conversores Análogo – Digital (A/D). Una vez terminada la etapa de conversión análogo – digital, los datos son entregados al DSP el cual está ahora en condiciones de procesarla. Eventualmente el DSP deberá devolver los datos ya procesados, para lo cual es necesaria una etapa final que transforme el formato digital a análogo. Esto es, la información puede ser devuelta a través de una conversión Digital – Análoga (D/A).

Esta etapa esta conformada por los siguientes elementos:

- Codec, encargado de las conversiones A/D y D/A
- DSP56303, encargado de la ejecución del algoritmo nLMS

2.2.2.1 Codec

El Codec de la empresa Crystal Semiconductor, CS4218, es un dispositivo monolítico CMOS para computadores multimedia y aplicaciones de audio portátiles. Este realiza las conversiones A/D y D/A, además tiene ganancia programable de entrada y atenuación programable de salida. Consta de cuatro entradas de audio y dos salidas del mismo tipo. Los conversores A/D y D/A usan modulación Delta-Sigma, incluyen filtros de decimación digital y salidas suavizadas de los filtros con lo cual se elimina la necesidad de filtros anti-aliasing. El Codec se comunica con el DSP por medio del puerto ESSI (Enhanced Synchronous Serial Interface). El software del DSP debe inicializar la velocidad del reloj, y la velocidad de muestreo requerida para la aplicación.

El Codec tiene varios modos de operación. Estos modos son configurados poniendo ciertos pines del Codec en alto o bajo, especialmente SMODE1, SMODE2 y SMODE3. En este proyecto el modo de trabajo elegido es el Serial Mode 4 (SM4). El SM4 permite que la información de control para el Codec este separada de la información de datos.

En el SM4 existe varios sub-modos, estos modos secundarios especifican dos cosas:

1. El Codec puede funcionar ya sea de maestro o esclavo
2. El número de bits por trama de datos

Los modos secundarios son físicamente configurados en sub-modo 0, esto indica que el Codec está configurado como maestro y el tamaño de la trama escogida es de 32 bits.

Cuando el Codec está configurado como maestro, envía un pulso de sincronización, que indica el inicio o final de la trama de datos. Además el sub-modo 0 especifica que cada trama consiste de dos palabras de 16 bits, una para el canal izquierdo y otra para el canal derecho. Estas palabras son enviadas hacia y desde el Codec con el bit más significativo primero (MSB First).

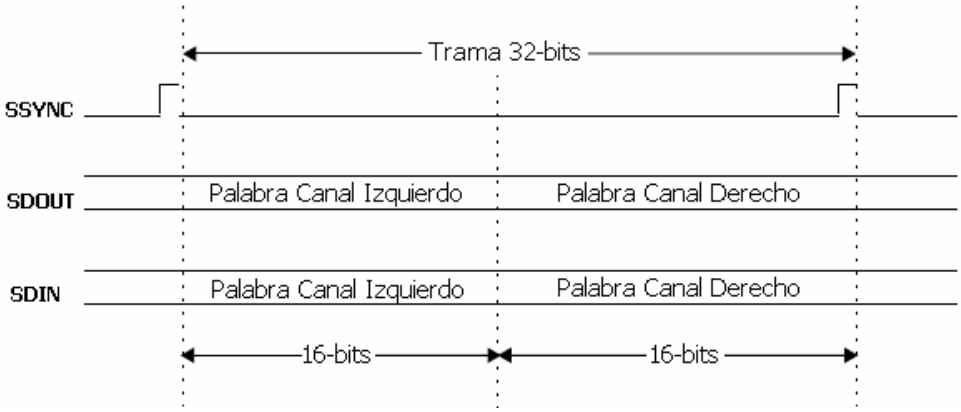


Figura 2.6 Interfaz serial para la transmisión de datos del Codec [7].

En la Figura 2.6 se puede apreciar la interface serial para la transmisión de datos de audio, un pulso en la línea SSYNC indica el inicio de la trama y el final de la misma. La línea SDOUT es la salida de datos de audio desde el Codec hacia el DSP, mientras que la línea SDIN es la entrada de datos de audio provenientes del DSP.

La información de control consiste en una lista de atributos que necesitan estar especificados en orden para establecer ciertas propiedades en la configuración de la atenuación y de la ganancia. Aunque 32 bits conforman la palabra de control, solo 23 bits son útiles, los nueve restantes son cero, tal como se indica en la Figura 2.7:

[7] Tomado de: "*DSP56303EVM User's Manual* ", Motorola, Rev. 3.4, Appendix D, pp. 3, Dec. 1999. www.chameleon.synth.net/files/developer/pdf/motorola/dsp56303evmum.rev3.4.pdf

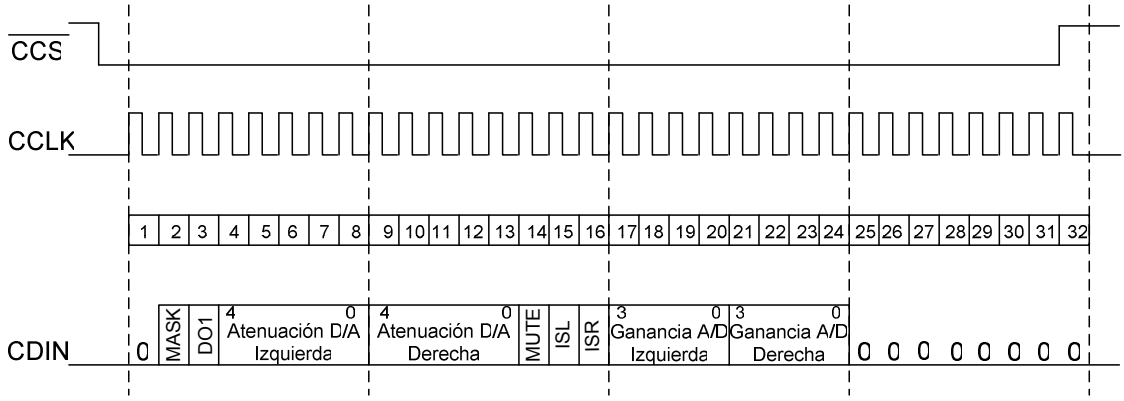


Figura 2.7 Configuración de los registros de control del Codec [8].

Cuando $\overline{CCS}$ esta en bajo los bits que conforman el registro son enviados hacia CDIN y guardados en cada flanco de subida de CCLK.

2.2.2.1.1 Serial mode 4 (SM4)[9]

El SM4 es habilitado poniendo en nivel alto el pin SMODE3, y los sub-modos son seleccionados por los pines SMODE2 y SMODE1 como se muestra en la Tabla 2.1:

Tabla 2.1 SM4, Sub-modos

SMODE1	SMODE2	SM4, Sub-modos
0	0	Maestro, 32BPF
0	1	Esclavo, 128/64/32BPF
1	0	Maestro, 64BPF, TS1
1	1	Maestro, 64BPF, TS2

En el proyecto el Codec está configurado en el Sub-modo 0, es decir, el Codec actúa como maestro con 32BPF (Bits Per Frame). En este sub-modo, la relación de fase entre SCLK/SSYNC y CLKIN es controlada para minimizar el ruido digital dentro de la parte analógica.

---

[8] Tomado de: "*CS4218: 16-Bit Stereo Audio Codec*", CRYSTAL, pp. 24, Sep. 1996

[9] Tomado de: "*CS4218: 16-Bit Stereo Audio Codec*", CRYSTAL, pp. 22-24, Sep. 1996

En SM4, la frecuencia de CLKIN debe ser 256 veces mayor que la frecuencia de muestreo necesaria, además en SM4 la parte de datos de audio (audio data) esta separada de los datos de control (control data), los datos de control están en un puerto serial independiente.

El Codec, utiliza los pines MF6:F1, MF7:F2 y MF8:F3 para seleccionar la frecuencia de muestreo como se indica en la Tabla 2.2:

Tabla 2.2 Selección de frecuencia de muestreo del Codec

MF6:F1	MF7:F2	MF8:F3	N	Fs(kHz) con CLKIN	
				12.288 MHz	11.2896 MHz
0	0	0	256	48.00	44.10
0	0	1	384	32.00	29.40
0	1	0	512	24.00	22.05
0	1	1	640	19.20	17.64
1	0	0	768	16.00	14.70
1	0	1	1024	12.00	11.025
1	1	0	1280	9.60	8.82
1	1	1	1536	8.00	7.35

Todas las frecuencias de la Tabla 2.2 son frecuencias de muestreo estándar para audio y dependen de CLKIN, en este caso para 12.288MHz y 11.2896MHz. Para otro valor de CLKIN la frecuencia de muestreo es determinada por CLKIN/N.

La información de control es ingresada a través de un puerto serial que es asincrónico con respecto al puerto serial de audio, y solo necesita ser actualizado para cambiar los registros de control del Codec. Los principales registros de control del Codec son:

- Atenuación del D/A en el canal izquierdo
- Atenuación del D/A en el canal derecho
- Ganancia del A/D en el canal izquierdo
- Ganancia del A/D en el canal derecho

Las formas de onda para la configuración de los registros de control se muestran en la Figura 2.6.

2.2.2.1.2 Puerto ESSI [10]

El DSP56303 tiene dos puertos ESSI (ESSI0 y ESSI1) los cuales forman una interfaz serial con periféricos externos, cada puerto consiste de 6 pines los cuales permiten realizar múltiples funciones, dependiendo de cómo estos están configurados. Cada puerto puede funcionar ya sea como ESSI o como un puerto de entrada/salida de propósito general (GPIO, General Purpose Input/Output Port). Para configurar cualquiera de estos modos se modifica los registros de control de puerto. Los dos registros PCRC y PCRD determinan, respectivamente, como los puertos ESSI0 y ESSI1 van a ser usados.

Cada bit de estos registros representa un pin del puerto, si en un bit del registro se escribe un 1, esto significa que el pin del puerto está configurado como ESSI caso contrario estará configurado como GPIO.

Ambos puertos ESSI0 y ESSI1 son usados para el control y transferencia de datos entre el DSP y el Codec, el ESSI0 controla la transferencia de datos mientras que el ESSI1 la información de control del Codec. Esto se indica en la Figura 2.8:

---

[10] Tomado de: "*DSP56303EVM User's Manual* ", Motorola, Rev. 3.4, Appendix D, pp. 6-8, Dec. 1999. www.chameleon.synth.net/files/developer/pdf/motorola/dsp56303evmum.rev3.4.pdf

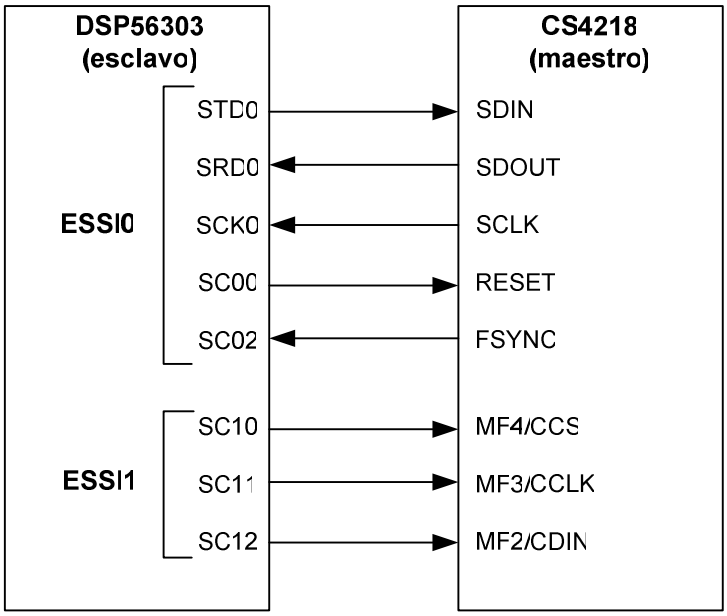


Figura 2.8 Interface digital del Codec.

El ESSI0 ejecuta tres funciones. Primero, transfiere los datos hacia y desde el Codec; segundo, recibe los pulsos de sincronización y finalmente ejecuta la función de reset sobre el Codec. El ESSI1 controla y transfiere la información de control del Codec. Los pines utilizados para conectar el DSP con el Codec se indican en la Tabla 2.3:

Tabla 2.3 Descripción de los pines que conectan al DSP con el Codec

Pines ESSI0/ESSI1	Pines CS4218	Descripción
STD0 (ESSI0)	SDIN	Transfiere datos desde el ESSI0 hacia el Codec
SRD0 (ESSI0)	SDOUT	Transfiere datos desde el Codec hacia el ESSI0
SCK0 (ESSI0)	SCLK	Recibe reloj de sincronismo desde el Codec
SC00 (ESSI0)	RESET*	Resetea al Codec desde el ESSI0
SC02 (ESSI0)	SSYNC	Recibe pulso de sincronismo desde el Codec
SC10 (ESSI1)	CCS\	Habilita/Deshabilita puerto de control del Codec
SC11 (ESSI1)	CCLK	Recibe reloj desde el ESSI1
SC12 (ESSI1)	CDIN	Transfiere datos de control desde el ESSI1
Nota: * Esto quiere decir que las señales con \ son habilitadas con cero lógico.		

2.2.2.2 DSP56303

El DSP56303 es miembro de la familia DSP56300 [18], los cuales son de tecnología CMOS programable. El DSP56303 usa el núcleo del DSP56300, este núcleo ofrece un nuevo nivel de rendimiento en velocidad y consumo, provisto de un amplio set de instrucciones y una baja disipación de potencia.

El núcleo del DSP está compuesto por:

- Unidad Aritmético-Lógica ALU
- Unidad Generadora de Direcciones AGU
- Unidad de control de programa (PCU, Program Control Unit)
- Un oscilador PLL (Phase Lock Loop) y generador de reloj
- Memoria interna (On-chip Memory)
- Unidad de Intefaces por buses (BIU, Bus Interface Unit)
- Un módulo OnCE (On-Chip Emulation)

El DSP56303 además incluye mejoras significativas a la arquitectura del núcleo de la familia DSP56300 como son:

- Desplazador combinacional (Barrel Shifter)
- Direccionamiento a 24 bits
- Memoria cache para instrucciones

El DSP56303 puede ser usado en aplicaciones tales como: procesamiento de voz, datos y fax, videoconferencias, audio, control y procesamiento digital de señales.

Entre las principales características generales se tiene:

- 66/80 MIPS con un reloj de 66/80 MHz a 3.3V
- Un set de instrucciones para procesamiento de datos en paralelo
- Código ensamblador simple, muy parecido al de microprocesadores de la misma marca
- Un puerto TAP (Test Access Port) compatible con JTAG (Joint Action Test Group)

- Diseño CMOS de muy bajo consumo
- Modos de bajo consumo (Wait and Stop)
- Circuitería de manejo de consumo optimizado (instruction-dependent, peripheral-dependent, y mode-dependent)
- Tres bloques de Memoria (Memoria de programa P, memoria de datos X, y memoria de datos Y)

2.2.2.2.1 Descripción de la ALU

La ALU ejecuta todas las operaciones aritméticas y lógicas, los componentes de la ALU son:

- Una MAC paralela de 24 x 24 bits totalmente segmentada (fully pipelined)
- Una unidad de campo de bit, comprendida de un desplazador combinacional (barrel shifter) paralelo de 56 bits.
- Instrucciones condicionales para la ALU
- Aritmética de 24 bits o 16 bits seleccionadas por software
- Cuatro registros de propósito general de 24 bits (X1, X0, Y1 y Y0)
- Seis registros de datos (A2, A1, A0, B2, B1 y B0) que están concatenados dentro de dos acumuladores de propósito general de 56 bits (acumuladores A y B)

2.2.2.2.2 Descripción de la AGU

La Unidad de Generación de Direcciones, que puede operar en paralelo con otros recursos, realiza los cálculos para el direccionamiento de memoria. Esta unidad implementa cuatro métodos para generar direcciones:

- Linear
- Módulo
- Multiple Wrap-around
- Reverse-carry

Para manipular datos en memoria se actualizan los registros de dirección (punteros) en lugar de mover los grandes bloques de datos. El contenido del registro modificador de dirección define el tipo de aritmética o método a ser utilizado para los cálculos de direccionamiento.

La Unidad de Generación de Direcciones posee ocho registros de dirección divididos en dos grupos de cuatro para usarse en paralelo y poder acceder a dos memorias independientes (memoria X y memoria Y) de forma simultanea. Estos grupo de cuatro a su vez están compuestos de tres registros como se indica en la Tabla 2.4:

Tabla 2.4 Registros de la Unidad de Generación de Direcciones

MEMORIA	Registro de dirección (Address Register)	Registro de Offset (Offset Register)	Registro de modificación (Modifier Register)
X	R0	N0	M0
	R1	N1	M1
	R2	N2	M2
	R3	N3	M3
Y	R4	N4	M4
	R5	N5	M5
	R6	N6	M6
	R7	N7	M7

2.2.2.2.3 Descripción de la PCU

La unidad de control de programa se encarga de decodificar las instrucciones, control de lazos DO por hardware y procesa interrupciones. La PCU implementa una pipeline de 7 etapas y controla los diferentes estados de procesamiento del núcleo del DSP. La PCU consiste de tres bloques de hardware:

- Program Decode Controller (PDC)
- Program Address Generator (PAG)
- Program Interrupt Controller (PIC)

El bloque PDC decodifica las instrucciones de 24 bit cargadas dentro del latch de instrucciones y genera todas las señales necesarias para el control de la pipeline. El bloque PAG contiene todo el hardware necesario para la generación de direcciones de programa, y control de lazos. El bloque PIC intercede entre todos los requerimientos de interrupción (interrupciones internas o externas) y genera la dirección apropiada del vector de la interrupción a ser atendida.

Las características de la PCU incluyen:

- Modos de direccionamiento optimizados para aplicaciones con DSP
- Lazos DO anidados
- Rápido auto retorno de interrupciones

Implementa todas estas funciones usando los siguientes registros:

- PC Program Counter
- SR Status Register
- LA Loop Address register
- LC Loop Counter register
- VBA Vector Base Address register
- SZ Size register
- SP Stack Pointer
- OMR Operating Mode Register
- SC Stack Counter register

2.2.2.2.4 PLL y Reloj Oscilador

El generador de reloj en el DSP esta compuesto de dos bloques principales: el PLL, el cual ejecuta la división del reloj de entrada, multiplica la frecuencia y elimina pendientes en el reloj de salida; y el generador de reloj CLKGEN, el cual ejecuta la división y generación de pulsos de reloj.

También permite cambiar el factor de división DF, además el PLL permite al procesador operar con un reloj interno de alta frecuencia utilizando un reloj de baja frecuencia a la entrada, característica que ofrece dos beneficios inmediatos:

- Un reloj de entrada, de baja frecuencia, reduce totalmente las interferencias electromagnéticas generadas
- La habilidad de oscilar en diferentes frecuencias reduce costos, pues reduce la necesidad de osciladores adicionales para el sistema

2.2.2.2.5 Memoria interna (On-Chip memory)

El espacio de memoria esta particionado en: memoria de programa P, memoria de datos X y memoria de datos Y. La memoria de programa incluye una RAM interna y una ROM que pueden ser expandidas, los tamaños de las memorias pueden ser programables como se indica en la Tabla 2.5:

Tabla 2.5 Espacios de memoria configurables en el DSP

Instruction cache	**Switch mode**	**Tamaño Memoria P**	**Tamaño Instruction Cache**	**Tamaño Memoria X**	**Tamaño Memoria Y**
Deshabilitado	Deshabilitado	4096 x 24-bit	0	2048 x 24-bit	2048 x 24-bit
Habilitado	Deshabilitado	3072 x 24-bit	1024 x 24-bit	2048 x 24-bit	2048 x 24-bit
Deshabilitado	Habilitado	2048 x 24-bit	0	3072 x 24-bit	3072 x 24-bit
Habilitado	Habilitado	1024 x 24-bit	1024 x 24-bit	3072 x 24-bit	3072 x 24-bit

2.2.2.2.6 Expansión de memoria

La memoria puede ser expandida de manera externa por:

- Expansión de memoria de datos en dos espacios de memoria de 16M x 24-bit en modo de direccionamiento de 24 bits (64K en modo de direccionamiento de 16 bits)

- Expansión de memoria de programa en un espacio de memoria de 16M x 24-bit en modo de direccionamiento de 24 bits (64K in modo de direccionamiento de 16 bits)

Otras características incluyen:

- Puerto de expansión de memoria externa
- Interface simultanea para una memoria SRAM (Static Random Access Memory) y una memoria DRAM (Dymanic Random Access Memory)

2.2.2.2.7 Buses internos

Para permitir el intercambio de datos entre los bloques de memoria, los siguientes buses están implementados:

- Bus de expansión para periféricos de entrada o salida (PIO_EB, Peripheral I/O Expansion Bus)
- Bus de expansión para memoria ROM de programa (PM_EB, Program Memory Expansion Bus)
- Bus de expansión para memoria de datos X (XM_EB, X Memory Expansion Bus)
- Bus de expansión para memoria de datos Y (YM_EB, Y Memory Expansion Bus)
- Bus de datos Global (GDB, Global Data Bus), para comunicar la PCU y otras estructuras del núcleo
- Bus de datos para memoria P (PDB, Program Data Bus)
- Bus de datos para memoria X (XDB, X Memory Data Bus)
- Bus de datos para memoria Y (YDB, Y Memory Data Bus)
- Bus de direcciones de programa (PAB, Program Address Bus)
- Bus de direcciones de memoria X (XAB, X Memory Address Bus)
- Bus de direcciones de memoria Y (YAB, Y Memory Address Bus)

Con excepción del bus de datos de programa (PDB), todos los buses internos son de 16 bits. El PDB es un bus de 24 bits.

2.2.2.3 Memoria flash

El DSP56303 tiene una memoria RAM interna estática, es decir una memoria de programa volátil, esto significa un problema para una aplicación portátil. Por ello es necesario una memoria externa no volátil (memoria flash), en la cual este almacenado el programa que se quiere ejecutar.

Para almacenar el programa en la memoria flash en un principio se pensó en utilizar un programador de memorias, lo cual implicaba un costo adicional en el proyecto. Después de una exhaustiva investigación se encontró que la memoria podía ser programada directamente desde el DSP, constituyendo esta una opción más óptima.

Un programa en código ensamblador permite programar la memoria flash, este programa copia los datos de la memoria de programa del DSP en la memoria flash.

En el proyecto se utiliza un chip Atmel AT29LV010A-15TC, que es una memoria flash de 128k x 8-bit con tecnología CMOS.

Las principales características de esta memoria son:

- Voltaje de alimentación entre 3V y 3.6V
- Utiliza solo 3V para lectura y escritura
- Rápido acceso de lectura: 120ns
- Baja disipación de potencia
    - 15mA en modo activo
    - 40uA en modo standby
- Programación por sectores
    - Simple ciclo de reprogramación
    - 1024 sectores (128 bytes/sector)
- Rápido ciclo de programación de sectores: 15ms
- Duración típica > 10000 ciclos
- Entradas y salidas compatibles con TTL y CMOS

El DSP tiene un programa de fábrica denominado Bootstrap, que se encuentra en la memoria ROM interna, este programa permite al DSP inicializar después de que un reset es ejecutado o cuando la alimentación es reestablecida, en cualquiera de estas dos condiciones el contador de programa apunta a un vector de reset, para ello el DSP ejecuta un barrido de los pines MODA, MODB, MODC y MODD; dependiendo de la combinación de estos, el DSP inicializa desde cualquiera de los periféricos que se indican en la Tabla 2.6:

Tabla 2.6 Modos de inicialización del DSP [11]

Modo	MODD	MODC	MODB	MODA	Vector de Reset	Descripción
0	0	0	0	0	$C00000	Modo expandido
1	X	0	0	1	$FF0000	Inicializa desde memoria externa
2	X	0	1	0	$FF0000	Inicializa a través del puerto SCI
3	X	0	1	1	-	Reservado
4	X	1	0	0	$FF0000	HI08 inicializa en ISA/DSP5630x
5	X	1	0	1	$FF0000	HI08 inicializa en HC11 no-multiplexado
6	X	1	1	0	$FF0000	HI08 inicializa en un 8051
7	X	1	1	1	$FF0000	HI08 inicializa en un 68302
8	1	0	0	0	$008000	Modo expandido

Para poder inicializar al DSP desde la memoria Flash el pin correspondiente a MODA debe estar en un valor lógico alto, mientras que los otros pines deben estar en un valor lógico bajo.

Los pines de dirección de la memoria flash (A0…A16) están conectados a los pines de dirección en el puerto A del DSP.

[11] Tomado de: "*DSP56303 User's Manual*", Section 4, pp. 4, 1996. http://www.motorola-dsp.com

Los pines de entrada/salida de la flash (I/O0…I/O7) son conectados a los pines D0-D7 del DSP. La habilitación de escritura de la memoria ($\overline{WE}$) y la habilitación de las salidas ($\overline{OE}$) son conectadas, respectivamente, a las líneas de escritura y lectura del DSP ($\overline{WR}$ y $\overline{RD}$), la habilitación de la memoria ($\overline{CE}$, chip enable) es generada por el pin AA1 (Address Attribute 1), tal como se muestra en la Figura 2.9:

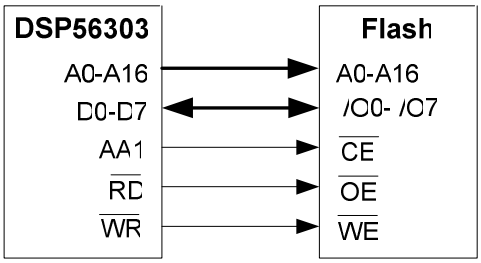


Figura 2.9 Conexión de la memoria flash[12].

2.2.2.4 Command converter

El command converter es utilizado para cargar el programa en la memoria interna del DSP. Los bloques de memoria son programados usando la interface JTAG/OnCE.

El software utilizado en el computador corre primordialmente bajo ambientes de 32-bits de Windows, sin embargo en versiones de 16 y 32-bits de DOS también es posible su uso. El command converter usa el puerto paralelo como interfaz con el computador y el puerto JTAG/OnCE para comunicarse con el DSP, las conexiones se muestran en la Figura 2.10:

[12] Tomado de: "*DSP56303EVM User's Manual* ", Motorola, Rev. 3.4, Chapter 3, pp. 6, Dec. 1999. www.chameleon.synth.net/files/developer/pdf/motorola/dsp56303evmum.rev3.4.pdf

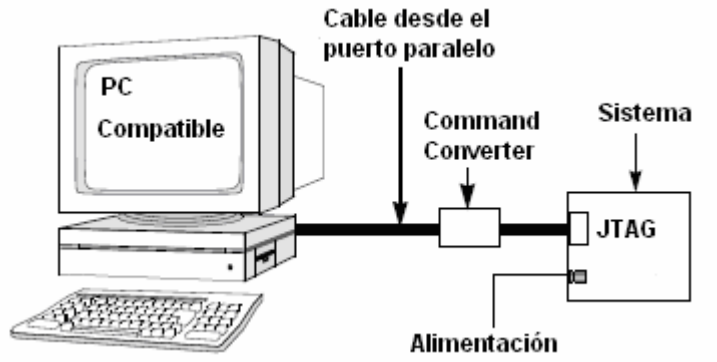


Figura 2.10 Diagrama de conexiones del DSP con el computador a través del command converter.

El command converter actúa como un simple conversor de voltaje, este acepta los voltajes TTL del puerto paralelo y los convierte en 3.3V CMOS para la interface con el puerto JTAG/OnCE del DSP. Solo 4 señales JTAG (TCK, TDI, TDO y TMS) y tierra son requeridas para un correcto funcionamiento tal como se muestra en la Figura 2.11:

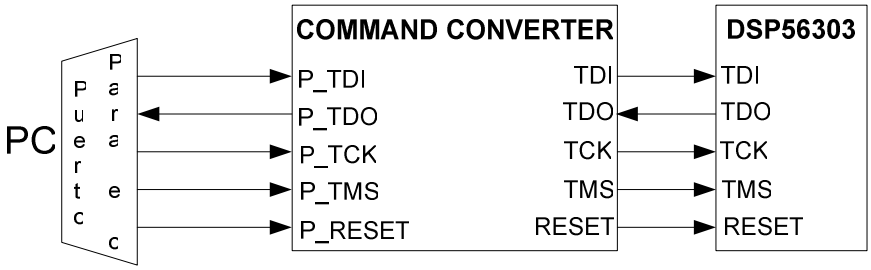


Figura 2.11 Conexión del command converter con el DSP.

2.2.2.4.1 Puerto JTAG/OnCE [13]

Los programas en el DSP se cargan con la ayuda de un software computacional, además se tiene la posibilidad de probar los programas paso a paso conectados directamente con el DSP lo que facilita la tarea de depuración de estos programas. Esto es posible ya que el DSP tiene el puerto JTAG que es un Test

[13] Tomado de: "*DSP56303 User's Manual*", Section 11, pp. 5, 1996. http://www.motorola-dsp.com

Access Port (TAP), el cual permite trabajar con el DSP en modo de depuración (Debug mode).

Un TAP estándar consta de cinco pines de señal pero el puerto JTAG solo requiere de cuatro pines (TDI, TDO, TCK y TMS) para su correcto funcionamiento, el DSP provee el pin opcional $\overline{\text{TRST}}$. En el DSP el pin DEBUG EVENT ($\overline{\text{DE}}$) es provisto para ser usado por el módulo OnCE. Las funciones de los pines se describen a continuación:

Test Clock (TCK), este pin de entrada se utiliza para sincronizar la comunicación.
Test Mode Select (TMS), este pin de entrada es usado para la secuencia de prueba de los controladores de estado de máquina. TMS es tomado en un flanco de subida en TCK.
Test Data Input (TDI), las instrucciones para el test serial y los datos son recibidos en este pin. TDI es tomado en un flanco de subida en TCK.
Test Data Output (TDO), es la salida serial para los datos e instrucciones del test. TDO cambia con un flanco de bajada en TCK.
Test Reset ($\overline{\text{TRST}}$), es usado para inicializar asincrónicamente los controladores del test. Este pin es de entrada.

2.2.3 ETAPA DE TRANSMISIÓN

El EWM-900-FDTC es un transceiver full-duplex que permite la transmisión y recepción tanto de voz como de datos, consta de 56 canales en el rango de 902-928MHz, los cuales son fácilmente seleccionables.

El transceiver es configurado mediante una interface serial de tres líneas (LE, DAT y CLK), las señales para la configuración del transceiver son generadas por el DSP mediante el puerto D funcionando como GPIO, además la señal a transmitir es la señal de salida del Codec como se indica en la Figura 2.12:

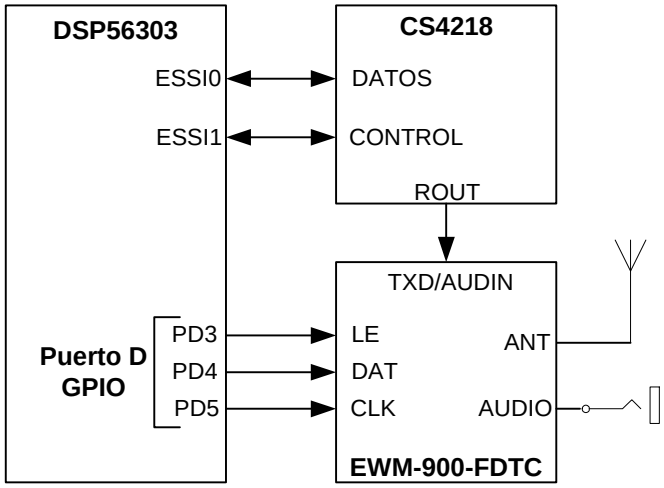


Figura 2.12 Conexión del Transceiver.

Los parámetros que pueden ser configurados mediante esta interface incluyen canal de transmisión, canal de recepción, habilitación de transmisión y habilitación de recepción.

El transceiver tiene cuatro registros de 24 bits cada uno, los mismos que son programados mediante la línea DAT. Cuando LE esta en alto los bits que conforman el registro son enviados hacia DAT y guardados en cada flanco de subida de CLK, como se indica en la Figura 2.13:

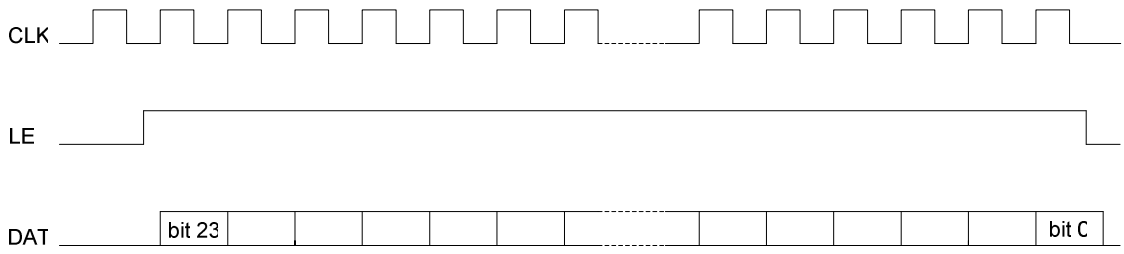


Figura 2.13 Formas de onda para la programación serial del transceiver[14].

Nótese que para configurar los registros se debe enviar a DAT primero el bit más significativo de la palabra que conforma el registro.

[14] Tomado de: "*EWM-900-FDTC*", rev 1.2, pp.6, February 2004.

Los registros del transceiver son [16]:

- Control de frecuencia de referencia
- Control del VCO de recepción
- Control del VCO de transmisión, y
- Modo

Los bits 0 y 1 de cada palabra de 24 bits que ingresan por DAT definen el registro a ser programado tal como se indica en la Tabla 2.7:

Tabla 2.7 Bits de selección de registros del transceiver

Bit 1	Bit 0	Registro
0	0	Control de frecuencia de referencia
0	1	Control del VCO de recepción
1	0	Control del VCO de transmisión
1	1	Modo

Dentro de los registros de transmisión y de recepción hay bits especiales denominados trim bits, estos bits se determinan a partir del valor de trim, que es un valor de fábrica asignado a cada módulo. El valor de trim consta de dos dígitos, el primer dígito corresponde a los bits de trim del registro de transmisión y el segundo dígito corresponde a los bits de trim del registro de recepción.

2.2.3.1.1 Control de frecuencia de referencia

El valor programado en este registro determina la frecuencia de referencia para el transceiver, para un correcto funcionamiento del transceiver esta frecuencia debe ser establecida en 50kHz. Como se aprecia en al Tabla 2.8:

Tabla 2.8 Registro de frecuencia de referencia

Bit								Byte
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**	
1	1	0	0	0	0	0	0	**0**
0	1	0	0	0	0	1	1	**1**
0	0	0	0	0	0	0	0	**2**

2.2.3.1.2 Control del VCO de recepción

Este registro determina el canal de operación del receptor. La frecuencia de recepción es determinada por los contadores M, A y F[15], además los dos bits de trim son utilizados para ajustar el rango de sintonización del receptor. Esto se indica en la Tabla 2.9:

Tabla 2.9 Registro del VCO de recepción

Bit								Byte
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**	
Contador A [2:0]			Registro F [2:0]			Selección Reg. [1:0]		**0**
A2	A1	A0	F2	F1	F0	0	1	
Contador M [5:0]						Contador A [4:3]		**1**
M5	M4	M3	M2	M1	M0	A4	A3	
Pad		Trim [1:0]		Contador M [9:6]				**2**
0	0	T1	T0	M9	M8	M7	M6	

2.2.3.1.3 Control del VCO de transmisión

Este registro determina el canal de operación del transmisor. La frecuencia de transmisión es determinada por los contadores M, A y F, además los tres bits de

[15] Los valores para los contadores M, A y F están disponibles en el manual de configuración del transceiver "*EWM-900-FDTC*", rev 1.2, pp.9, Table 9, February 2004.

trim son utilizados para ajustar el rango de sintonización del transmisor. Como se indica en la Tabla 2.10:

Tabla 2.10 Registro del VCO de transmisión

Bit								Byte
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**	
Contador A [2:0]			Registro F [2:0]			Selección Reg. [1:0]		**0**
A2	A1	A0	F2	F1	F0	1	0	
Contador M [5:0]						Contador A [4:3]		**1**
M5	M4	M3	M2	M1	M0	A4	A3	
Pad	Trim [2:0]			Contador M [9:6]				**2**
0	T2	T1	T0	M9	M8	M7	M6	

2.2.3.1.4 Modo

Determina el modo de operación del transceiver, es decir, el transceiver puede funcionar como transmisor, receptor o ambos a la vez.

Tabla 2.11 Registro de modo de operación

Bit								Byte
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**	
0	0	1	TXE	RXE	1	1	1	**0**
0	0	0	0	0	0	0	0	**1**
0	0	0	0	0	0	0	DAT*	**2**

El bit RXE determina el estado del receptor si es 1 la recepción es habilitada, si es 0 la recepción es deshabilitada. Por otra parte el bit TXE determina el estado del transmisor 1 la transmisión es habilitada, si es 0 la transmisión es deshabilitada.

El hardware implementado, así como los esquemáticos correspondientes al mismo se tratan a fondo en el Capítulo 4.

CAPÍTULO 3

DESARROLLO DEL SOFTWARE DE SOPORTE

En este Capítulo se explica el desarrollo del algoritmo nLMS implementado en Matlab y LabVIEW, así como también, una explicación del funcionamiento y uso de los programas creados en estos paquetes computacionales.

Para el caso del DSP56303, se explica los programas desarrollados, la manera de compilarlos y como deben ser cargados en el DSP; esto incluye, una breve explicación del manejo del software de apoyo proporcionado por Motorola para ejecutar las tareas antes mencionadas.

También se muestra los diagramas de flujo de los distintos programas implementados en Matlab, LabView y el DSP56303, además, los diagramas son explicados de manera detallada en lenguaje estructurado.

3.1 ALGORITMO nLMS EN MATLAB

El algoritmo nLMS se basa en métodos iterativos. Matlab permite realizar este tipo de programas en un lenguaje de alto nivel basado en lenguaje C. Para poder probar el algoritmo nLMS se debe contar con las señales de entrada del filtro que son: la señal de interés y el ruido contaminante. Estas dos señales están contenidas en dos archivos de audio **.wav**, los cuales fueron creados con la ayuda del grabador de sonidos del sistema operativo Windows.

Los dos archivos de audio **.wav** son convertidos en vectores de datos, esta conversión se logra utilizando el comando ‘wavread’ de Matlab. Los dos vectores obtenidos no tienen la misma dimensión, es decir, no tienen el mismo número de datos, por esta razón se debe redimensionar los vectores debido a que el

algoritmo nLMS trabaja con sumas y productos de los vectores de entrada y estos deben tener la misma dimensión para que no se generen errores.

Después de redimensionar los vectores se ejecuta el algoritmo nLMS, el tiempo que demora en procesar las señales de entrada el algoritmo, depende del número de muestras elegidas por el usuario. Cuando finaliza el algoritmo se puede apreciar las graficas correspondientes a: señal de interés, salida (señal filtrada) y ruido contaminante. Además, se puede reproducir la señal de salida mediante Matlab o cualquier otro reproductor de audio, pues al finalizar el algoritmo se crea un archivo **.wav** con los datos de la señal de salida.

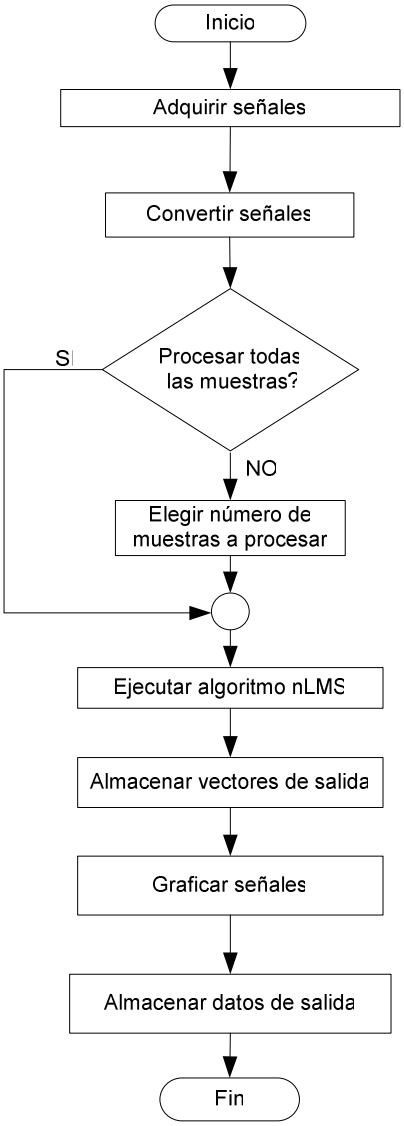


Figura 3.1 Secuencia general del algoritmo de control.

El programa se describe a continuación en lenguaje estructurado:

Adquirir señales

Abrir archivo de sonido de la señal sin contaminar

Abrir archivo de sonido del ruido contaminante

Fin Tarea

Convertir señales

Convertir archivos de sonido en vectores de datos

Redimensionar vectores

Fin Tarea

Elegir número de muestras a procesar

Guardar el valor elegido en la variable n

Redimensionar los vectores de datos al tamaño de n

Generar ruido gaussiano

Sumar señal de interés más ruido generado (señal + ruido)

Sumar ruido contaminante más ruido generado (ruido)

Fin Tarea

Ejecutar algoritmo nLMS

Establecer orden del filtro M=48

Generar vector de ceros como pesos iniciales

Guardar en vector de pesos W

Inicializar estimador de potencia P con valor de 1

Calcular beta en función de M

Para valores comprendidos entre M+1 y n

Calcular la salida del FIR adaptivo

Actualizar el valor del estimador P

Calcular el error

Actualizar el vector de pesos W

Fin

Fin Tarea

Graficar señales

Graficar señal de interés

Graficar salida

Graficar ruido contaminante

Fin Tarea

Almacenar datos de salida

Convertir datos de salida en archivo .wav

Fin Tarea

3.1.1 DESARROLLO DEL PROGRAMA EN MATLAB

La pantalla principal de programa se muestra en la Figura 3.2:

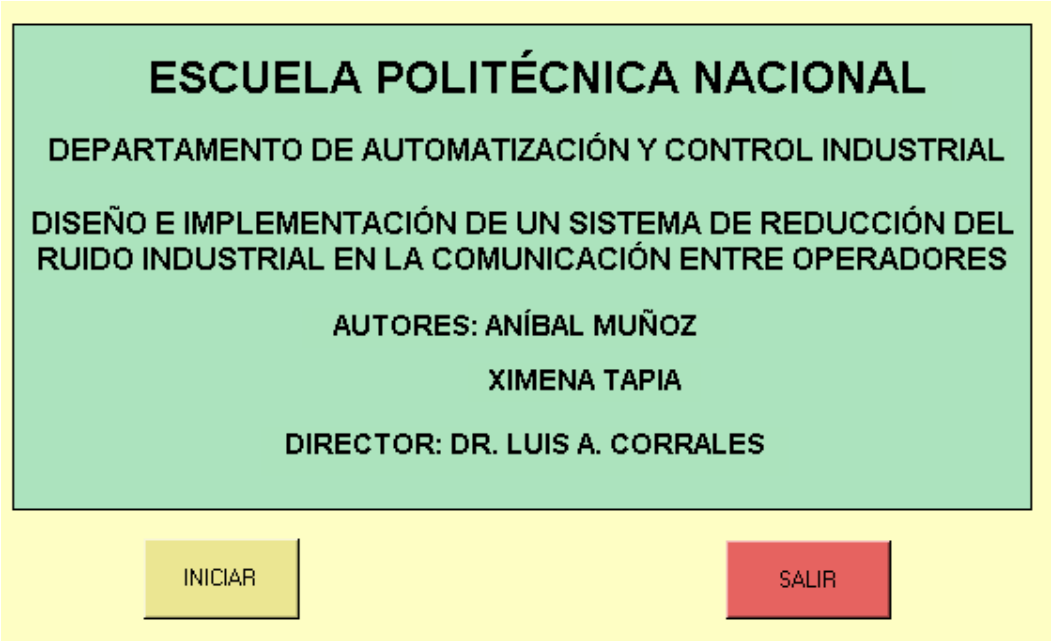


Figura 3.2 Pantalla principal en Matlab.

Al escoger la opción INICIAR, aparece la pantalla indicada en la Figura 3.3:

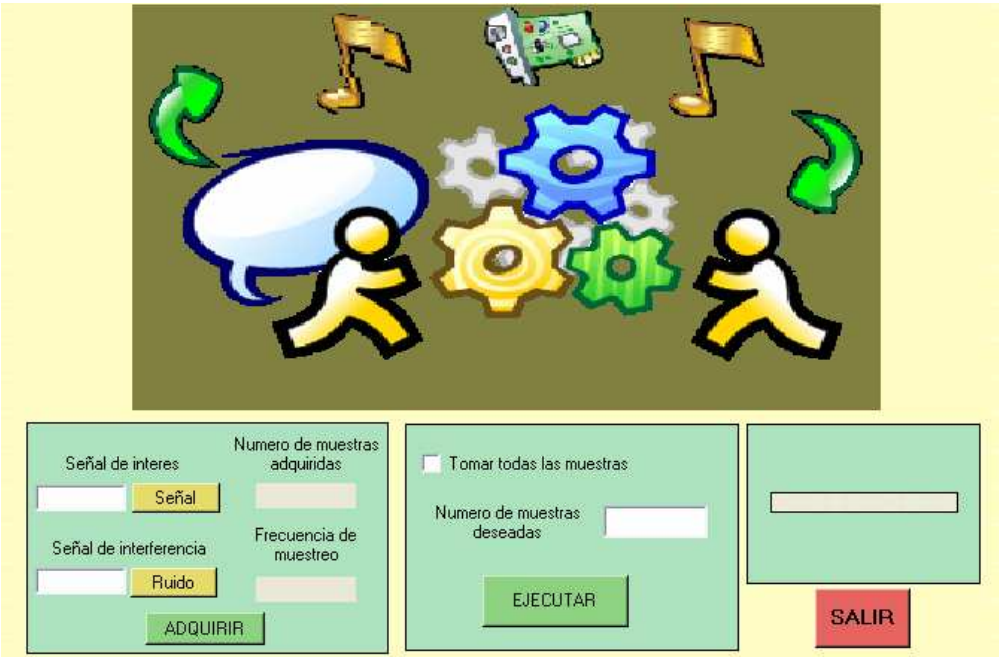


Figura 3.3 Pantalla para la ejecución del algoritmo.

Para ejecutar el algoritmo se debe seguir los siguientes pasos:

1. Elegir la señal de interés de entre los archivos **.wav** disponibles en la carpeta work de Matlab. Esto se lo hace presionando el botón “Señal”
2. Elegir la señal de interferencia de entre los archivos **.wav** disponibles en la carpeta work de Matlab. Esto se lo hace presionando el botón “Ruido”
3. Presionar el botón “ADQUIRIR”, el cual muestra en la pantalla los valores correspondientes a: Número de muestras adquiridas y, la frecuencia de muestreo
4. Seleccionar el número de muestras con las que se desea trabajar. Las opciones son: Tomar todas las muestras o un número determinado por el usuario
5. Presionar el botón “EJECUTAR”. Una vez realizada esta acción el algoritmo nLMS inicia, y se muestra en una barra el porcentaje de las muestras procesadas
6. Si se presiona el botón “SALIR”, el programa regresa a la pantalla indicada en la Figura 3.2

Una vez que el algoritmo ha finalizado se despliega la pantalla indicada en la Figura 3.4:

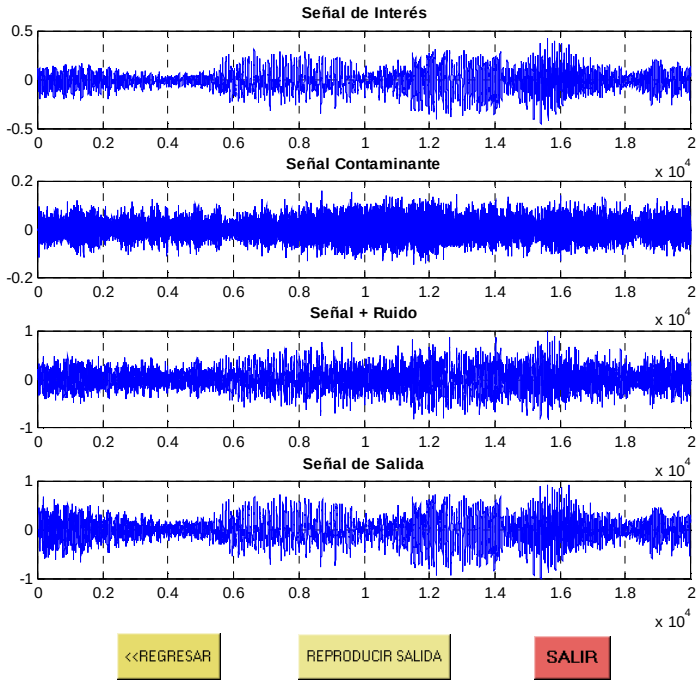


Figura 3.4 Gráficas de los resultados del algoritmo nLMS.

En la Figura 3.4, se muestra: Señal de interés, Señal filtrada y la Señal de ruido.

Además se tienen tres opciones:

- REGRESAR, este botón permite regresar a la pantalla anterior (Figura 3.3) para volver a ejecutar el algoritmo
- REPRODUCIR SALIDA, al presionar este botón se puede escuchar el resultado del filtro
- SALIR, esta opción permite regresar a la pantalla principal

3.2 ALGORITMO nLMS EN LabVIEW

LabVIEW es un lenguaje de programación de alto nivel de tipo gráfico que cuenta con todas las estructuras necesarias. Puede ser usado para elaborar cualquier algoritmo de tipo iterativo, como el nLMS.

Para la implementación del algoritmo en LabVIEW, se sigue el mismo diagrama de flujo que se aprecia en la Figura 3.1

3.2.1 DESARROLLO DEL PROGRAMA EN LabVIEW

La pantalla principal se indica en la Figura 3.5:

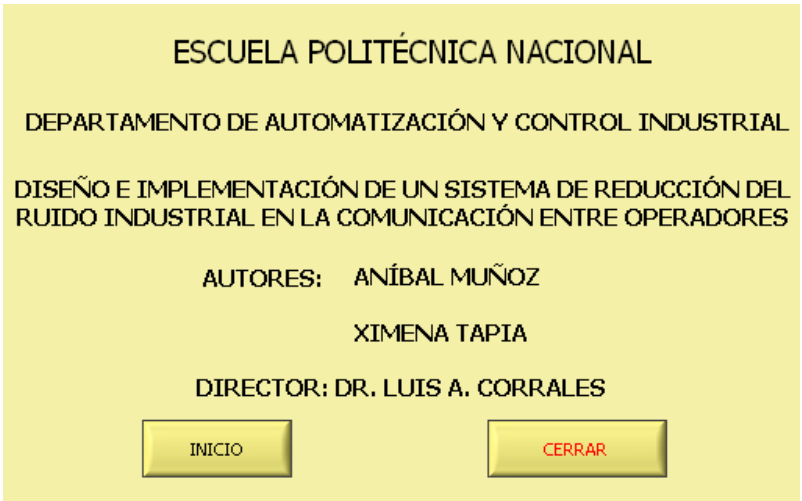


Figura 3.5 Pantalla principal en LabVIEW.

Para ejecutar el algoritmo se debe presionar el botón INICIO y seguir los siguientes pasos:

1. Seleccionar la señal sin contaminar
2. Seleccionar el ruido contaminante
3. Presionar el botón ADQUIRIR SEÑALES. Cuando se realiza esta acción se muestra en la pantalla las gráficas de las señales adquiridas
4. Seleccionar si se desea trabajar con todas las muestras disponibles o con un número de muestras determinado
5. Presionar el botón INICIAR ALGORITMO. Una vez que el algoritmo inicia se muestra una barra que indica el avance del proceso
6. El botón REPRODUCIR SALIDA permite escuchar el resultado del filtro
7. Si se desea utilizar nuevamente el algoritmo, se debe presionar el botón REINICIAR ALGORITMO
8. El botón STOP detiene el proceso y retorna a la pantalla principal indicada en la Figura 3.5

Una vez que el algoritmo ha finalizado se observa la pantalla indicada en la Figura 3.6:

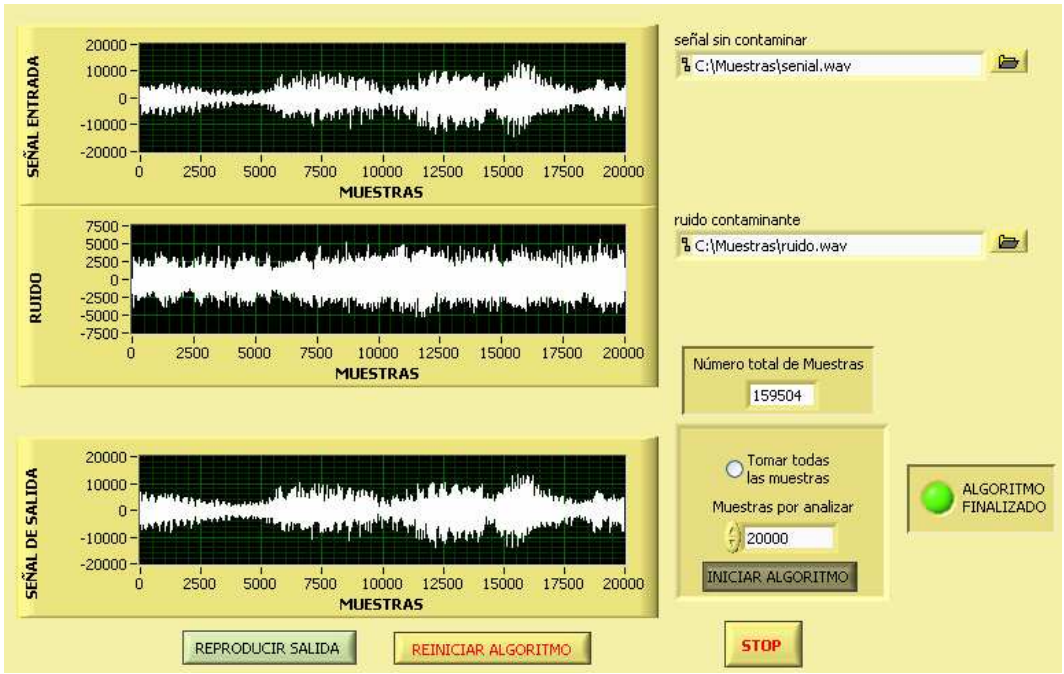


Figura 3.6 Pantalla de gráficas y ejecución del algoritmo nLMS.

3.3 ALGORITMOS IMPLEMENTADOS EN EL DSP56303

Dentro del DSP, además del algoritmo nLMS, se debe cargar la aplicación que permite programar la memoria flash. Ambos algoritmos están desarrollados en código ensamblador, estos deben ser compilados para poder ser cargados en el DSP56303. Cada una de estas aplicaciones se describe a continuación.

El software usado para compilar los programas desarrollados en código ensamblador es el **asm56300**, el cual es una aplicación basada en MS-DOS. Por ello, para compilar un programa, se debe salir de Windows o abrir una ventana de DOS.

El formato general en la línea de comando de DOS para ejecutar el compilador es:
asm56300 [*opciones*] <*archivos*>[16]

Donde **<*archivos*>** corresponde a aquellos con extensión **.asm** y **[*opciones*]** son las diferentes opciones que ofrece el compilador.

Por ejemplo, para compilar un programa se puede poner el siguiente código **asm56300 -a -b -g -l *nombre*.asm.** Cabe señalar que el asm56300 debe estar en la misma carpeta donde se encuentran los archivos **.asm**. Las opciones elegidas (-a -b -g -l) dan como resultado un archivo **.cld**, para el caso del ejemplo sería ***nombre*.cld**.

Con el archivo **.cld** se puede cargar el programa al DSP, esto se hace con ayuda del command converter. Para ello Motorola proporciona un software que esta incluido en el "Motorola DSP Development Tools", que permite depurar los programas realizados en código ensamblador mediante el uso del computador.

---

[16] Tomado de: *"DSP56303EVM User's Manual"*, Motorola, Rev. 3.4, Chapter 2, pp. 5-7, Dec. 1999. www.chameleon.synth.net/files/developer/pdf/motorola/dsp56303evmum.rev3.4.pdf

El GDS56300 es el software que permite cargar y ejecutar el programa en el DSP a través del computador.

3.3.1 ALGORITMO nLMS

El diagrama de flujo del algoritmo nLMS implementado en el DSP se muestra en la Figura 3.7:

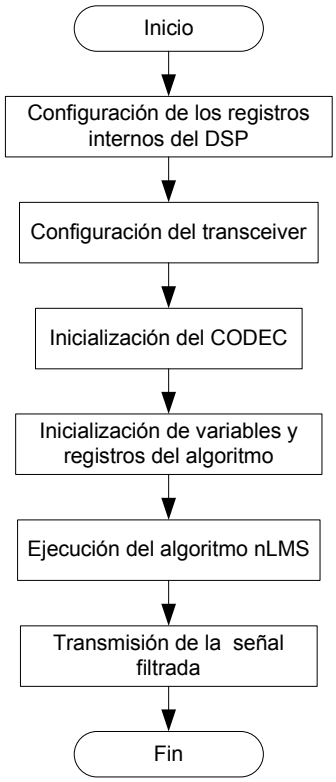


Figura 3.7 Diagrama de flujo del algoritmo nLMS implementado en el DSP56303.

El programa se describe brevemente a continuación en lenguaje estructurado:

Configurar los registros internos del DSP

Limpiar punteros
Limpiar contadores
Deshabilitar interrupciones
Establecer reloj interno en 86.016MHz
Inicializar puntero de recepción de datos rx (señales de entrada)
Inicializar puntero de transmisión de datos tx (señal de salida)
Definir tamaño buffer para muestras
Definir tamaño buffer para coeficientes

Fin Tarea

Configurar el transceiver

Enviar palabra para registro de frecuencia de referencia
Enviar palabra para registro del VCO de recepción
Enviar palabra para registro del VCO de transmisión
Enviar palabra para registro de modo de operación

Fin Tarea

Inicializar el CODEC

Definir buffers y punteros de transmisión y recepción de datos
Resetear puertos ESSI
Resetear CODEC
Modificar los registros de control del CODEC
Habilitar CODEC
Habilitar interrupciones DSP

Fin Tarea

Inicializar variables y registros del algoritmo

Definir orden del filtro M=48
Inicializar buffer circular para muestras
Inicializar buffer circular para coeficientes
Obtener datos canal izquierdo CODEC (ruido)
Obtener datos canal derecho CODEC (señal de interés)

Fin Tarea

Ejecutar algoritmo nLMS

Calcular beta en función de M
Para valores comprendidos entre M+1 y n
Calcular la salida del FIR adaptivo
Actualizar el valor del estimador P
Calcular el error
Actualizar el vector de pesos W
Fin

Fin Tarea

Transmitir la señal filtrada

Enviar datos de salida del filtro al CODEC

Fin Tarea

3.3.2 PROGRAMACIÓN DE LA MEMORIA FLASH

El algoritmo para la programación de la memoria flash se compone de varios bloques[17], pero se deben destacar dos muy importantes que son:

---

[17] Para mayor información sobre la programación de la memoria flash referirse a: "*Interfacing Serial EEPROM To DSP563xx*", I. Naslavsky, L. Smolyansky, Motorola, 1998. **APR38.pdf**

- Bloque de escritura y,
- Bloque de lectura

3.3.2.1 Bloque de escritura

El diagrama de flujo de la Figura 3.8 presenta la rutina de escritura de la memoria flash:

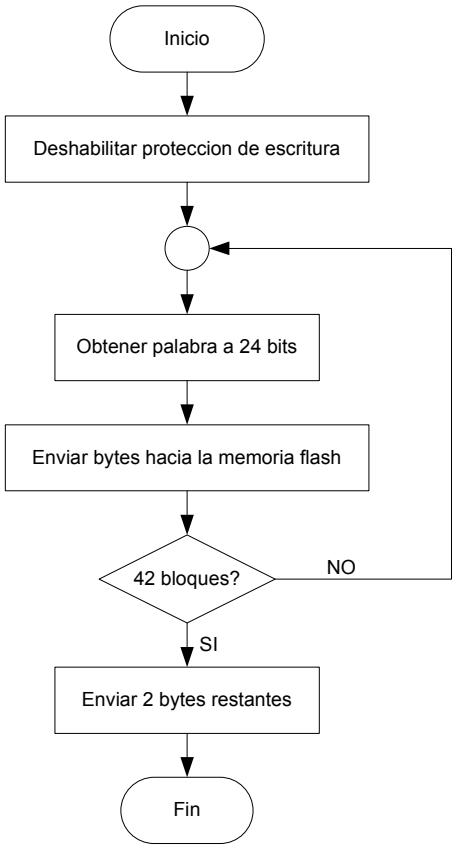


Figura 3.8 Bloque de escritura.

El programa se describe a continuación en lenguaje estructurado:

Deshabilitar protección de escritura

Cargar 8Dh en la dirección de memoria 5555h

Cargar 72h en la dirección de memoria 2AAAh

Cargar 05h en la dirección de memoria 5555h

Fin Tarea

Obtener palabra a 24 bits

Copiar palabra de programa en acumulador

Fin Tarea

Enviar bytes hacia la memoria flash

Enviar byte bajo hacia la memoria flash

Enviar byte medio hacia la memoria flash

Enviar byte alto hacia la memoria flash

Fin Tarea

Enviar 2 bytes restantes

Enviar byte 127 hacia la memoria flash

Enviar byte 128 hacia la memoria flash

Generar retardo de 20 ms

Fin Tarea

3.3.2.2 Bloque de lectura

Este bloque de programa se encarga de copiar el programa dentro de la memoria flash en la memoria de programa P del DSP56303. El diagrama de flujo de la Figura 3.9 presenta la rutina de lectura de la memoria flash:

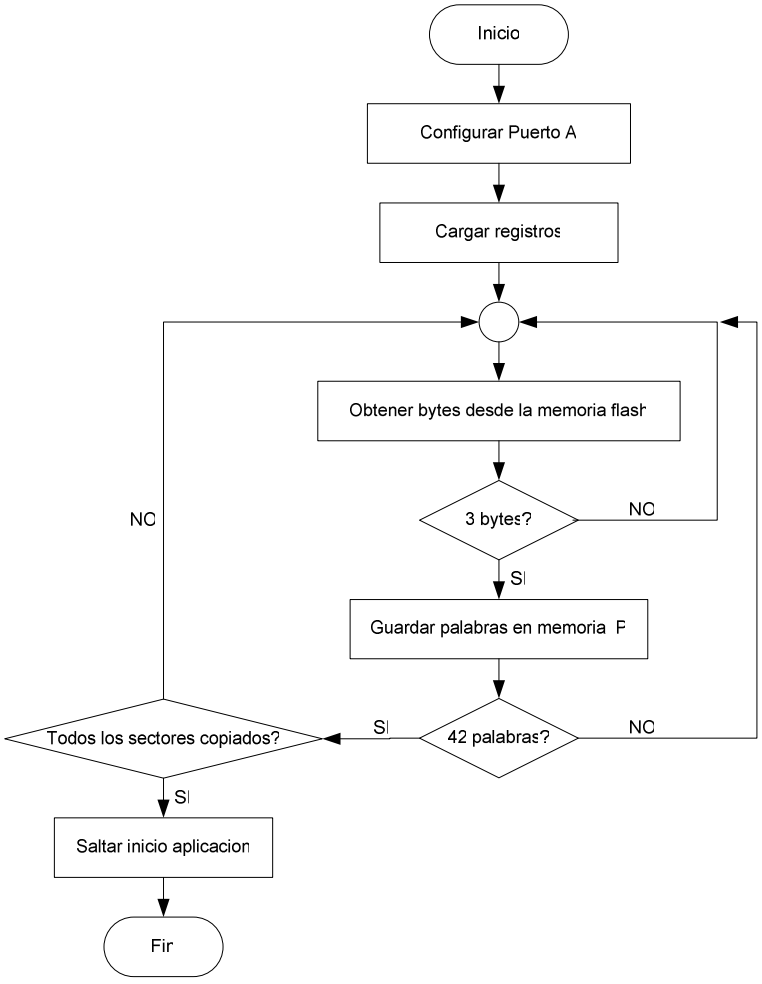


Figura 3.9 Bloque de lectura.

El programa se describe a continuación en lenguaje estructurado:

Configurar Puerto A

Configurar registro BCR

Configurar registro AAR1

Fin Tarea

Cargar registros

Cargar número de sectores en R3

Cargar dirección de inicio de aplicación en R0

Cargar longitud de aplicación en palabras en R2

Fin Tarea

Obtener bytes desde la memoria flash

Obtener 3 bytes desde la memoria flash

Ordenar bytes en palabra de 24 bits

Fin Tarea

Guardar palabras en memoria P

Guardar palabra 24 bits en memoria P del DSP

Repetir 42 veces

Fin Tarea

Saltar inicio aplicación

Cargar dirección de inicio de aplicación en R1

Saltar a dirección indicada en la memoria

Fin Tarea

3.4 CARGAR PROGRAMAS EN EL DSP56303

Para cargar el programa en el DSP se debe seguir los siguientes pasos:

1. Se debe compilar el archivo **.asm**, obteniendo los archivos **.cln**, **.map** y **.cld**, esto se logra con ayuda del **asm56300** y el uso de las diferentes opciones que este ofrece, así:
   - Para obtener el archivo **.cln**, en la línea de comando de DOS se debe ingresar:

     asm56300 -b *nombre*.asm

     Lo que da como resultado el archivo ***nombre*.cln**
   - Con ayuda del archivo **.cln** se obtiene el archivo **.map**, el cual da información de las posiciones de memoria del programa a ser cargado, entonces, se debe ingresar en la línea de comando de DOS:

dsplnk -m *nombre*.cln[18]

- Por último se debe obtener el archivo **.cld** de la siguiente manera:

  asm56300 -a -b -g -l *nombre*.asm

En la pantalla de DOS debe observase lo siguiente:

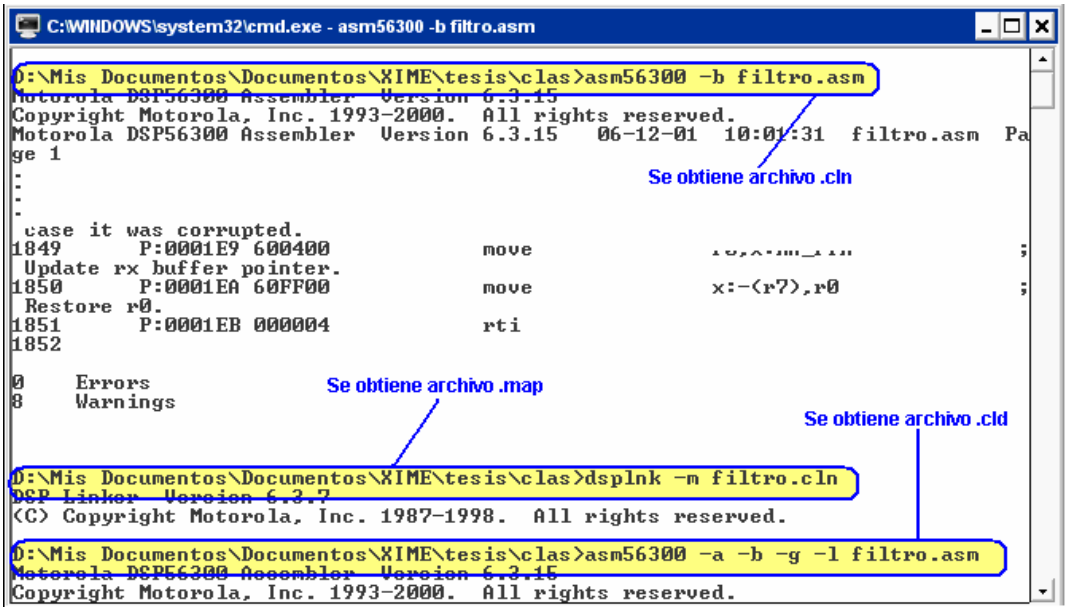


Figura 3.10 Ventana DOS con los pasos para compilar archivo **.asm**.

2. Una vez obtenidos los archivos mencionados en el paso anterior se debe conectar el DSP al command converter y este a su vez al computador. Para ello se necesita la herramienta computacional **GDS56300**[19], este software trabaja bajo ambiente Windows. Otra herramienta necesaria para la comunicación entre el computador y el DSP es el **Command Converter Server**, el cual es un driver, que permite el uso del puerto paralelo (generalmente LPT1) como interfaz con el DSP. Entonces, para cargar el programa en el DSP se realiza lo siguiente:
   - Se debe elegir el dispositivo a programar utilizando el menú Modify, así: Modify>Device>Configure…

[18] El **dsplnk** es otra herramienta proporcionada por Motorola y al igual que el **asm56300** trabaja en DOS.

[19] Para mayor información sobre el manejo del software referirse al manual de usuario **DSPADSUM.pdf**

En la Figura 3.11 se indica la pantalla desplegada para la elección del dispositivo a programar.

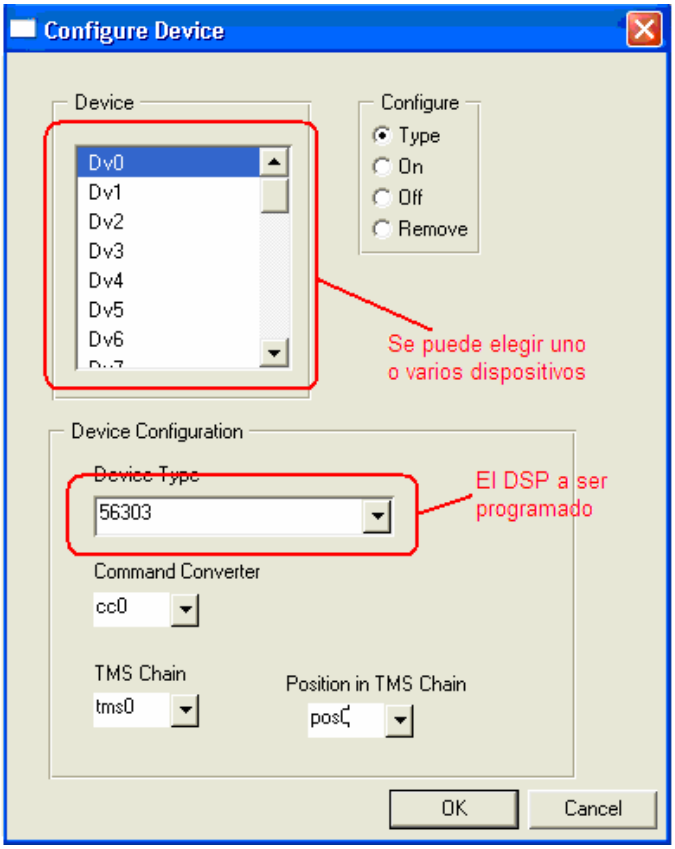


Figura 3.11 Configuración del dispositivo a programar.

- Antes de cargar cualquier aplicación se debe enviar un reset al DSP haciendo clic en el botón RESET de la barra de herramientas:

Figura 3.12 Barra de herramientas.

- Se abre el archivo correspondiente al programa a ser cargado, es decir, aquel con extensión **.cld**. Esta acción es ejecutada cuando en la línea de comando se escribe:

  LOAD <directorio de trabajo>*nombre*.cld

Otro método para realizar la misma acción es utilizando el menú File, así:

File>Load>**Memory COFF…**

- Como se explicó en el capitulo anterior, el programa del DSP debe ser cargado en la memoria flash; por lo que se debe abrir también el archivo ***flash*.cld** siguiendo el mismo procedimiento explicado en el paso anterior.

  En la memoria interna del DSP se ha cargado entonces dos programas. El primero corresponde a la aplicación que se desea ejecutar y el segundo corresponde al programa para manejar la memoria flash.

- Finalmente resta cambiar el valor de dos registros internos del DSP, estos registros son **r0** y **r1**. Los valores que se deben cargar en estos registros son: la dirección de inicio y el tamaño de la aplicación que se desea ejecutar. Estos valores se obtienen del archivo **.map**, como se indica en la Figura 3.13:

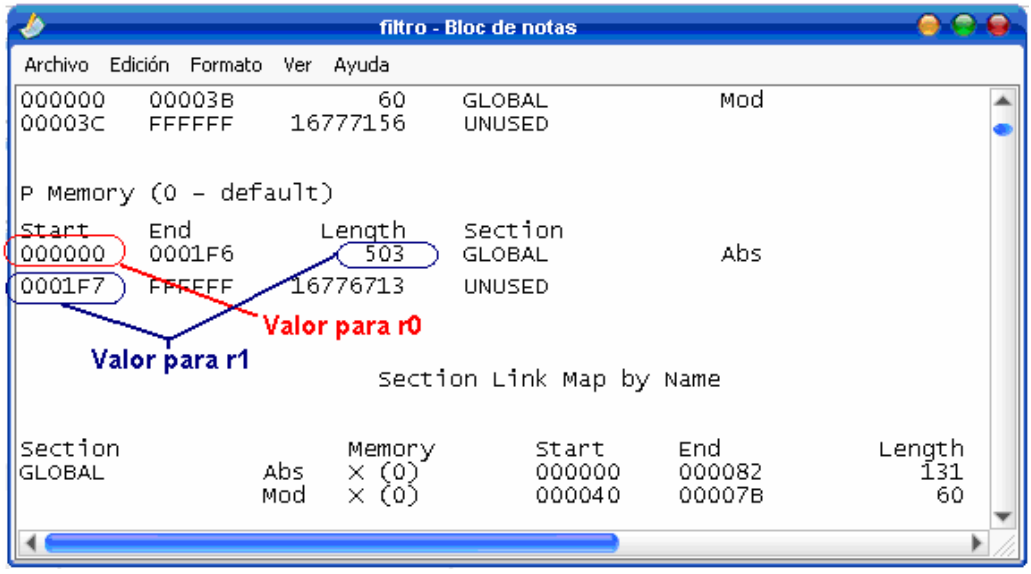


Figura 3.13 Archivo ***filtro*.map** de donde se obtiene los valores de r0 y r1.

- Para cambiar los valores de los registro, en la línea de comando del **GDS56300** se ingresa:

  CHANGE r0 000

  CHANGE r1 1F7

Estos cambios son requeridos por el programa de la memoria flash, pues definen el espacio en memoria a ser ocupado por la aplicación. En la Figura 3.13 se aprecia que el valor para **r1** es 1F7, este valor es particular para esta aplicación ya que la dirección de inicio del programa es la 000, no todas las aplicaciones inician en esta dirección. En esos casos, se debe cargar el valor dado en Length (longitud) pero este valor es decimal así que debe ser convertido a su equivalente hexadecimal.

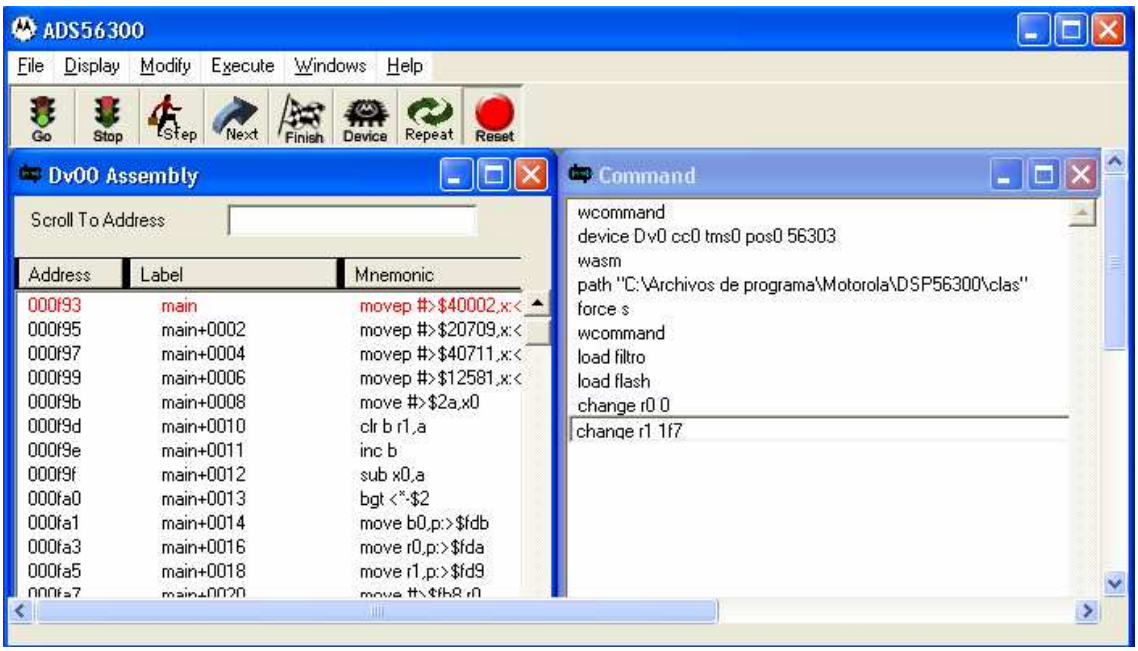


Figura 3.14 Programación del DSP mediante software de MOTOROLA.

- Para copiar el programa en la memoria flash en la línea de comando se debe ingresar:

  GO

  Cuando el programa termina de pasar los datos a la memoria flash en pantalla se muestra el mensaje de la Figura 3.15:

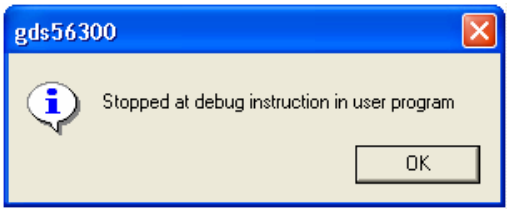


Figura 3.15 Confirmación de programa cargado.

Después de seguir todos estos pasos el programa queda grabado en la memoria flash, permitiendo al dispositivo funcionar en stand-alone mode.

Una vez desarrollados los programas en Matlab, LabVIEW y para el DSP se procede a comprobar el funcionamiento de estos, los resultados obtenidos en estas pruebas se analizan en el Capítulo 5.

CAPÍTULO 4

CONSTRUCCIÓN E IMPLEMENTACIÓN DEL HARDWARE DEL SISTEMA

Como se indicó en el Capítulo 2 el hardware está conformado por varias etapas:

- Etapa de amplificación
- Etapa de procesamiento de señal
- Etapa de transmisión

En el Capítulo 2 se explicó cada una de estas etapas en diagramas de bloque, así pues, en este Capítulo se indica el diseño de las mismas, basándose para ello en los circuitos recomendados en hojas de datos (datasheets) y material bibliográfico correspondiente a cada uno de los elementos que conforman el proyecto.

Este Capítulo también incluye los esquemáticos y ruteados de los circuitos impresos utilizados para la construcción del equipo.

4.1 ETAPA DE AMPLIFICACIÓN

En esta etapa las señales de entrada (voz + ruido y ruido) son obtenidas mediante dos micrófonos, estas señales deben ser amplificadas para posteriormente ser procesadas.

La amplificación de las señales de los micrófonos se logra con la utilización del LM386. Este circuito integrado es también utilizado para amplificar la señal de salida del Codec (resultado del filtro).

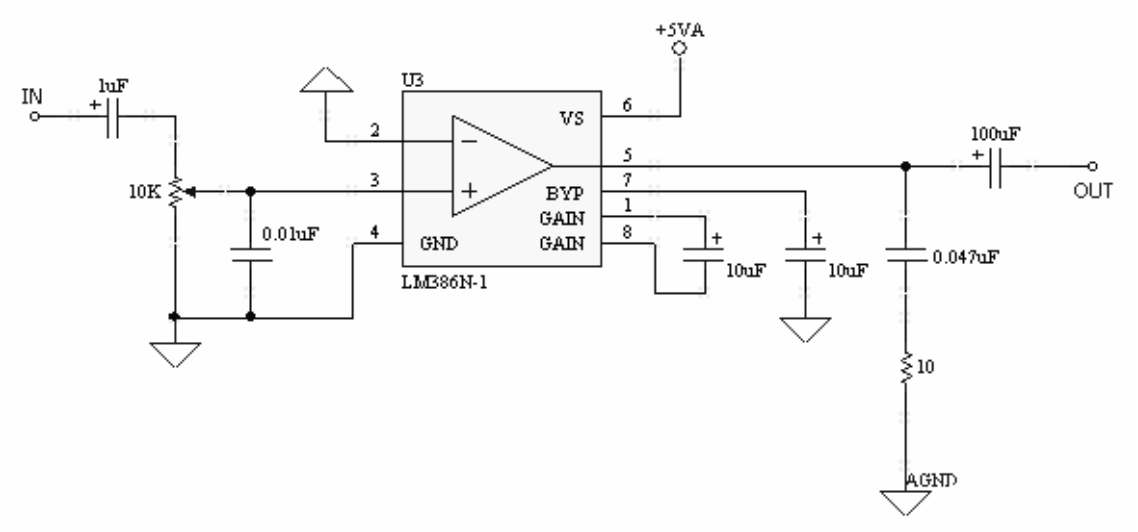


Figura 4.1 Circuito amplificador de audio con el LM386.

En la Figura 4.1 se puede apreciar el circuito más común para el uso del LM386, el volumen de la señal de salida es regulado por el potenciómetro que se encuentra en el circuito. Para los micrófonos este potenciómetro tiene un valor de 10KΩ, mientras que para los audífonos el valor del potenciómetro es de 100Ω.

Los micrófonos para que funcionen necesitan estar polarizados, esto se hace con una resistencia conectada entre el Terminal positivo del micrófono y la fuente de alimentación. Entre tanto, el Terminal negativo del micrófono debe estar conectado a tierra, tal como se indica en la Figura 4.2:

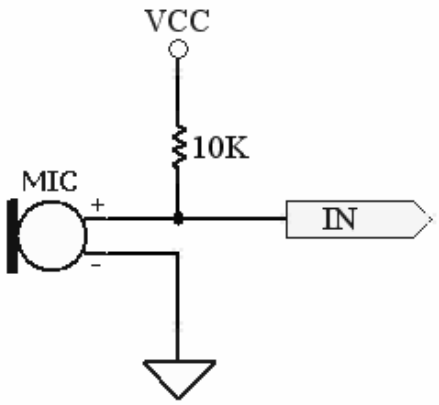


Figura 4.2 Polarización de los micrófonos.

Como se puede observar en el esquemático de la Figura 4.3 se utilizan dos jacks estereo:

- Jack micrófonos
- Jack audífonos

El jack de los micrófonos es el correspondiente a las señales de entrada de los amplificadores LM386, este, al ser un jack estereo tiene tres líneas; dos para los canales (derecho e izquierdo) y la tercera correspondiente a tierra. Por el canal derecho ingresa la **voz + ruido**, mientras que por el otro canal el **ruido**.

El jack de los audífonos permite escuchar el resultado del filtro, pues la salida del Codec es amplificada por el LM386. Lo explicado anteriormente se resume en la Figura 4.3:

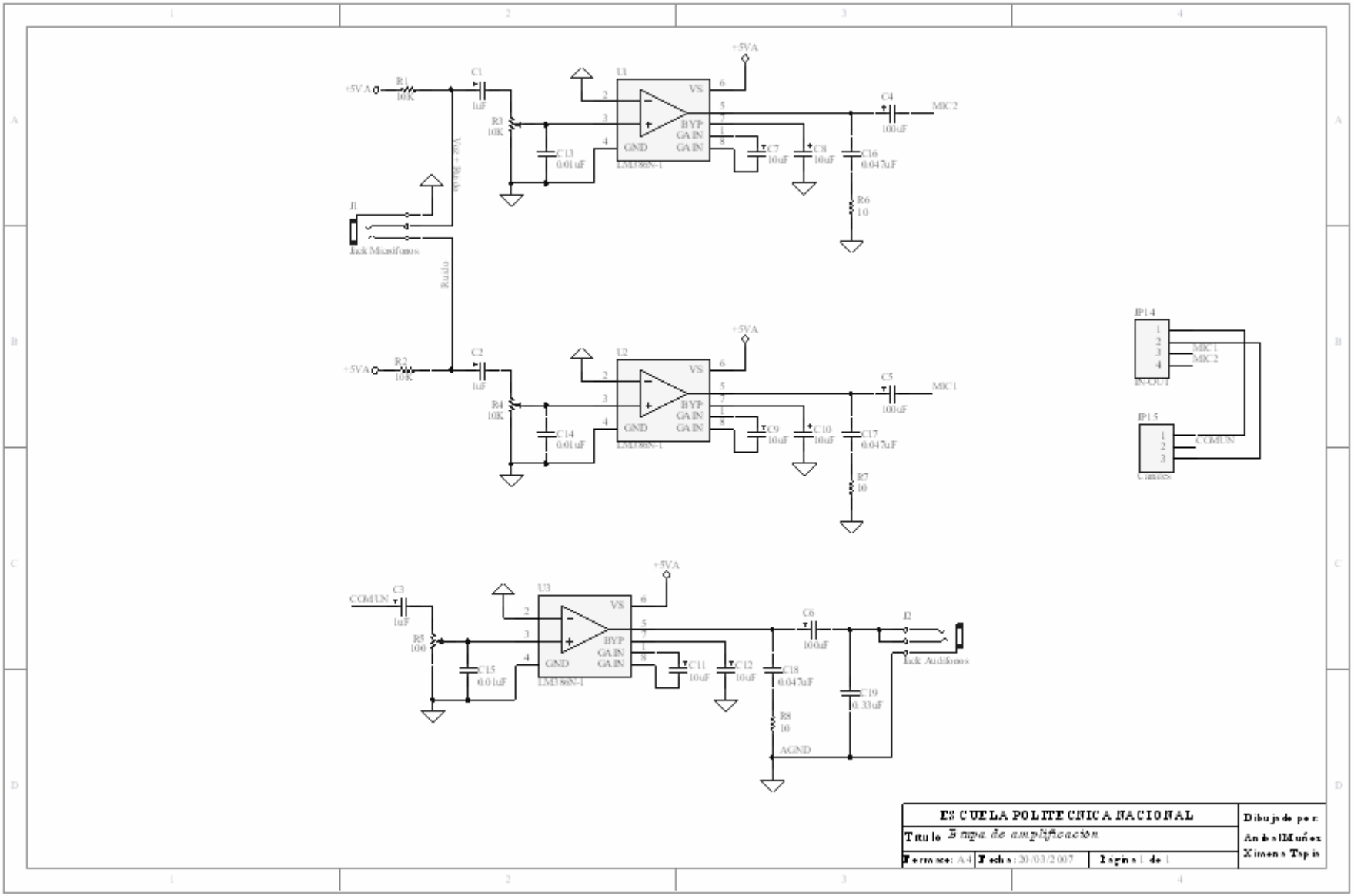


Figura 4.3 Amplificación de las señales de los micrófonos y de la salida del Codec.

4.2 ETAPA DE PROCESAMIENTO DE SEÑAL

Para poder procesar las señales en el DSP56303 es necesario que dichas señales sean digitales.

En esta etapa las señales amplificadas **voz + ruido** y **ruido** ingresan al Codec CS4218 al canal derecho y canal izquierdo respectivamente. El Codec convierte las señales de análogas a digitales y las envía al DSP para ser procesadas. La señal procesada ingresa nuevamente al Codec en donde es convertida de digital a análoga quedando lista para ser amplificada y posteriormente transmitida.

Es muy importante colocar filtros en la entrada y salida del Codec, los filtros recomendados se encuentran en el datasheet del mismo [15]. El filtro para las líneas de entrada del Codec se indica en la Figura 4.4:

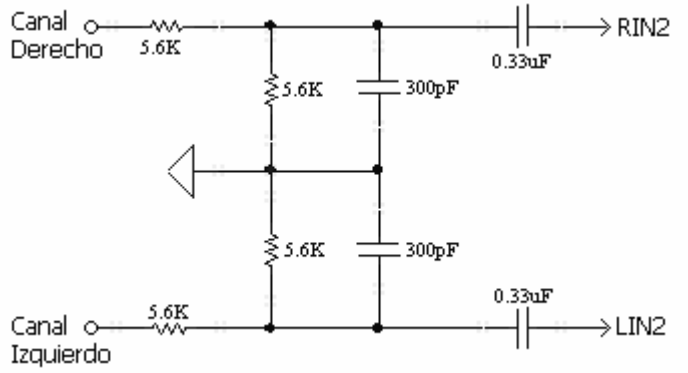


Figura 4.4 Filtros RC para líneas de entrada.

El filtro para las líneas de salida del Codec se indica en la Figura 4.5:

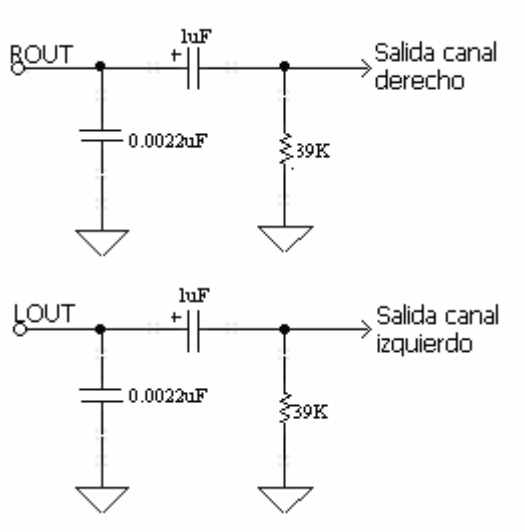


Figura 4.5 Filtros RC para líneas de salida.

En la Figura 4.6 se aprecia las conexiones básicas para el correcto funcionamiento del Codec. El conector JP9 es utilizado para seleccionar la frecuencia de muestreo, estos valores de frecuencia están entre los 8 kHz y los 48 kHz. La Tabla 4.1 muestra la posición de los jumpers en el conector JP9, para seleccionar las frecuencias de muestreo disponibles:

Tabla 4.1 Selección de las frecuencias de muestreo

JP9			**Frecuencia de muestreo (kHz)**
Pines 1-2 (MF6)	**Pines 3-4 (MF7)**	**Pines 5-6 (MF8)**	
Con Jumper	Con Jumper	Con Jumper	48.00
Con Jumper	Con Jumper	Sin Jumper	32.00
Con Jumper	Sin Jumper	Con Jumper	24.00
Con Jumper	Sin Jumper	Sin Jumper	19.20
Sin Jumper	Con Jumper	Con Jumper	16.00
Sin Jumper	Con Jumper	Sin Jumper	12.00
Sin Jumper	Sin Jumper	Con Jumper	9.60
Sin Jumper	Sin Jumper	Sin Jumper	8.00

Los puertos ESSI0 y ESSI1 se utilizan como interfaz entre el Codec y el DSP, estos puertos están representados en el esquemático de la Figura 4.6 por los conectores JP8 (ESSI0) y JP10 (ESSI1). La distribución de pines de estos conectores se indica en las Tablas 4.2 y 4.3:

Tabla 4.2 Conexión del puerto ESSI0 con Codec

JP8	**Pin ESSI0**	**Pin Codec**
1-2	SCK0	SCLK
3-4	SC00	RESET*
5-6	STD0	SDIN
7-8	SRD0	SDOUT
9-10	SC01	-
11-12	SC02	FSYNC
Nota: * Esto quiere decir que la señal con \ es habilitada con cero lógico.		

Tabla 4.3 Conexión del puerto ESSI1 con Codec

JP10	**Pin ESSI1**	**Pin Codec**
1-2	SCK1	-
3-4	SC10	MF4:CCS*
5-6	STD1	-
7-8	SRD1	-
9-10	SC12	MF3:CDIN
11-12	SC11	MF2:CCLK
Nota: * Esto quiere decir que la señal con \ es habilitada con cero lógico.		

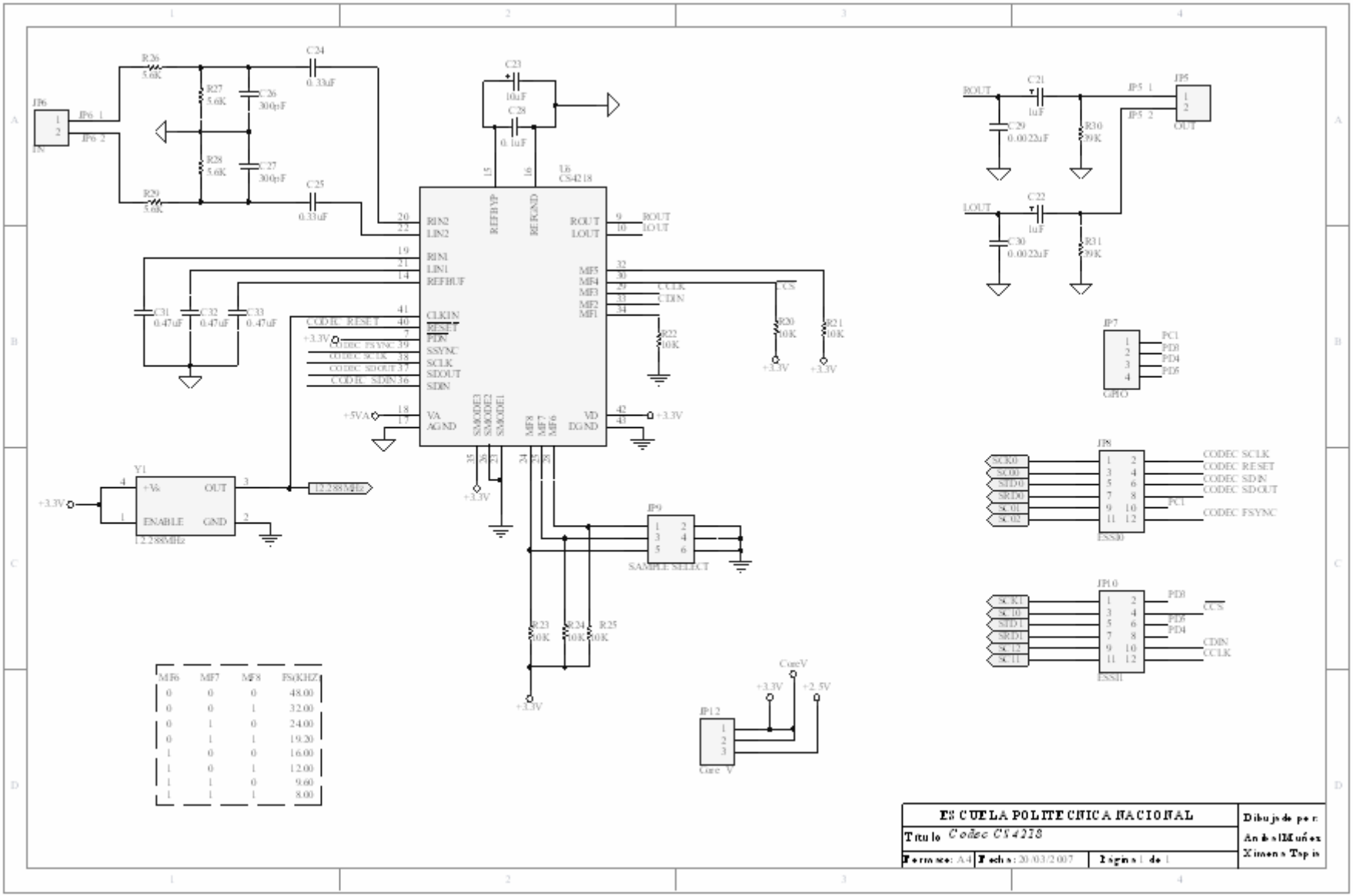


Figura 4.6 Codec CS4218.

En la Figura 4.7 se puede apreciar los pines del DSP que serán ocupados para interconectar a este con los otros dispositivos que conforman el proyecto. El DSP carga su programa desde la memoria Flash, para ello debe estar configurado el modo de inicialización del DSP. Esto se lo hace mediante el conector JP2, tal como se indica en la Tabla 4.4:

Tabla 4.4 Selección del modo de inicio del DSP

Modo*	**JP2**			
	Pines 1-2	**Pines 3-4**	**Pines 5-6**	**Pines 7-8**
8	Sin Jumper	Con Jumper	Con Jumper	Con Jumper
1	Sin Jumper	Con Jumper	Con Jumper	Sin Jumper
2	Sin Jumper	Con Jumper	Sin Jumper	Con Jumper
4	Sin Jumper	Sin Jumper	Con Jumper	Con Jumper
5	Sin Jumper	Sin Jumper	Con Jumper	Sin Jumper
6	Sin Jumper	Sin Jumper	Sin Jumper	Con Jumper
7	Sin Jumper	Sin Jumper	Sin Jumper	Sin Jumper
Nota: * Para mayor información sobre la descripción de los modos de inicialización del DSP referirse a la Tabla 2.6.				

Nótese que los pines impares del conector JP2 están conectados a 3.3V mediante resistencias de 10kΩ, mientras que los pines pares están conectados a la tierra digital (GND). Entonces, tomando como referencia la Tabla 2.6 se tiene que el conector JP2 debe estar configurado en Modo 1.

Se indica también el circuito de $\overline{\text{RESET}}$, el cual consiste en una resistencia de 5.1kΩ conectada a 3.3V. Esta resistencia está conectada en serie con un pulsador, el cual, al ser presionado, envía un cero lógico al pin de $\overline{\text{RESET}}$ del DSP. El pulsador a su vez, se encuentra conectado en paralelo con un capacitor de 4.7μF, el cual cumple la función de circuito anti-rebotes.

Para programar la memoria Flash y el DSP, este último, debe poder conectarse con un computador (PC), esto se lo hace mediante el puerto JTAG a través del command converter, cuyo esquemático se aprecia en la Figura 4.8.

En la Figura 4.7 se puede apreciar el acceso al puerto JTAG representado por el conector JP3, cuya distribución de pines esta dada en la Tabla 4.5:

Tabla 4.5 Conector para el puerto JTAG/OnCE

JP3	Pin DSP	JP3	Pin DSP
1	TDI	2	GND
3	TDO	4	GND
5	TCK	6	GND
7	-	8	-
9	RESET*	10	TMS
11	+3.3V	12	-
13	DE*	14	TRST*
Nota: * Esto quiere decir que la señal con \ es habilitada con cero lógico.			

En la Figura 4.7 se indica la conexión entre el DSP y la memoria Flash, la cual permite que el sistema sea portátil. La memoria Flash tiene 17 líneas de direccionamiento (A0…A16), las cuales están conectadas directamente a los pines de dirección del puerto A en el DSP (A0…A17. Las líneas de datos del DSP (D0…D7), están conectadas con las líneas de entrada y salida de la memoria Flash (I/O0...I/O7) como se indica en la Tabla 4.6:

Tabla 4.6 Conexión de las líneas de datos entre el DSP y la memoria Flash

Pines de Datos DSP (D0…D7)	Pines I/O de la Flash (I/O0…I/O7)
D0	I/O7
D1	I/O6
D2	I/O5
D3	I/O3
D4	I/O4
D5	I/O2
D6	I/O0
D7	I/O1

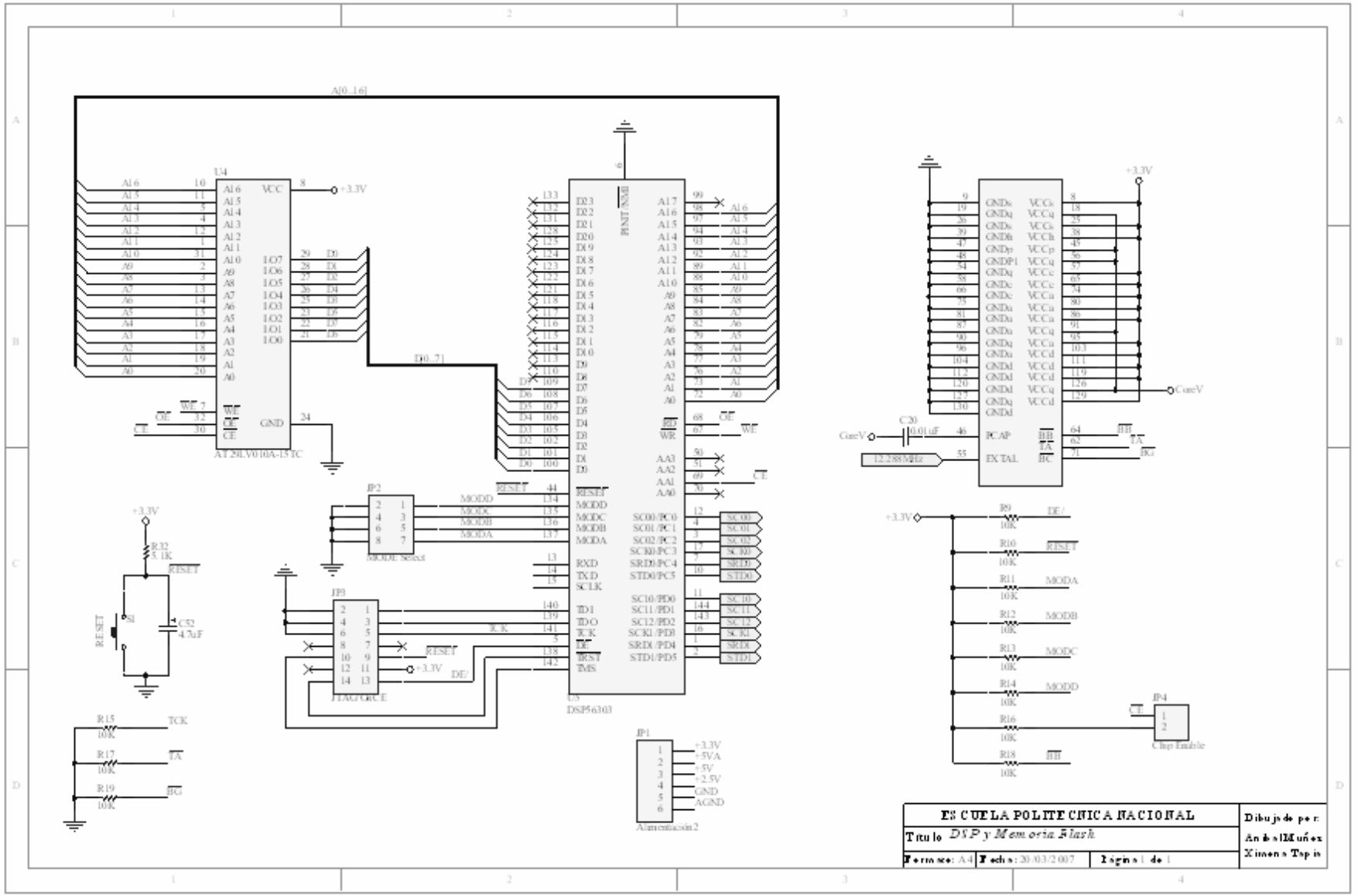


Figura 4.7 Conexión del DSP con la memoria Flash.

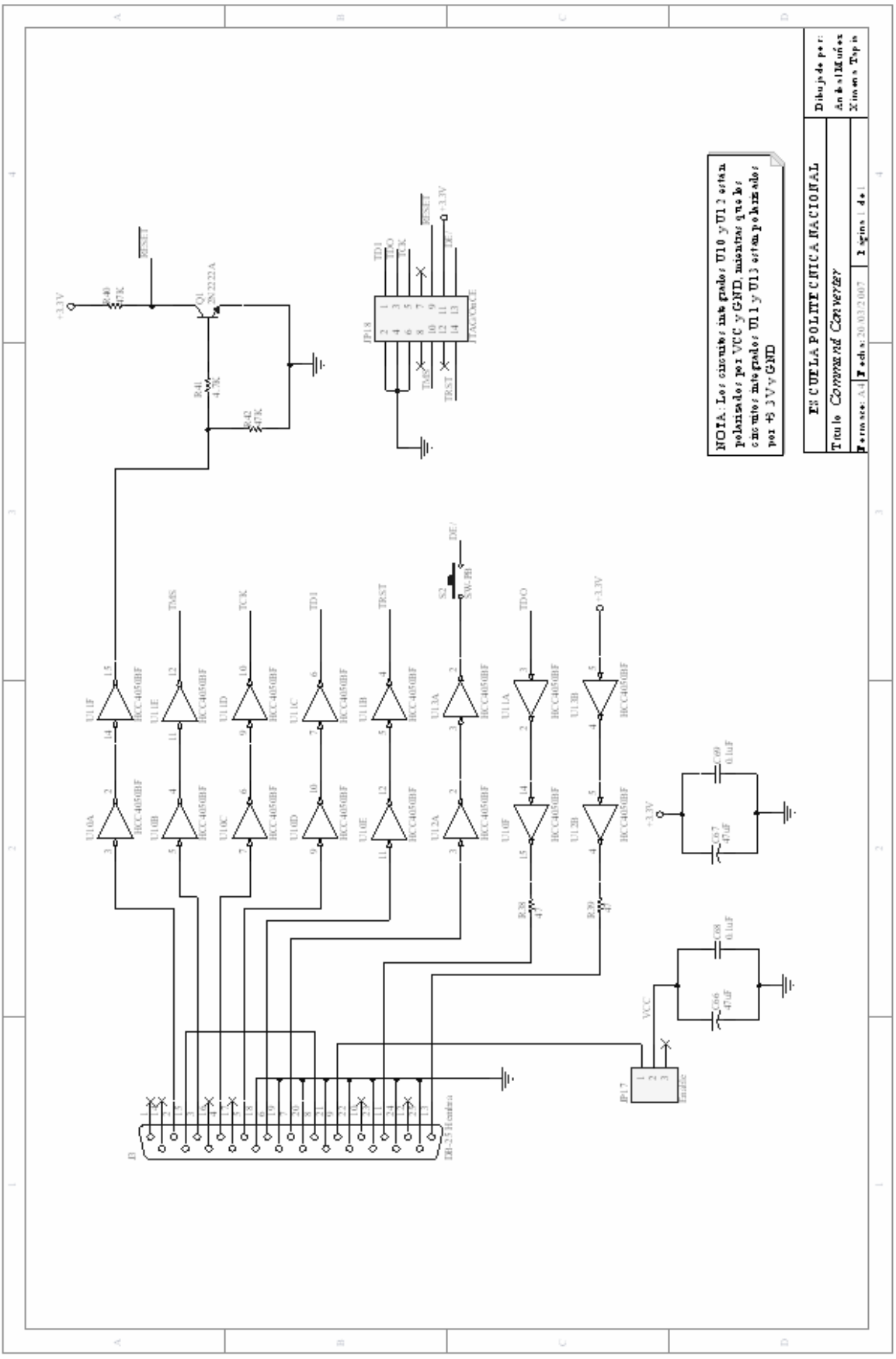


Figura 4.8 Command Converter.

4.3 ETAPA DE TRANSMISIÓN

En el Capítulo 2 estaba previsto la utilización de los transceivers EWM-900-FDTC, pero, no se logró que estos funcionen adecuadamente. Basándose en la información provista en las hojas de datos se consiguió programar los dispositivos, sin obtener la transmisión de las señales filtradas.

Con el propósito de encontrar la solución más viable a este problema, se optó por consultar a la casa manufacturera de dichos dispositivos (Radiotronix), los mismos que no dieron respuesta a las incógnitas planteadas; en segunda instancia se recurrió a los grupos de ayuda en internet (foros), de los cuales se obtuvo valiosa información permitiendo un mejor entendimiento acerca del funcionamiento de los transceivers, dicha información también sugería el uso de otro dispositivo de similares características, puesto que el correcto funcionamiento del EWM-900-FDTC no había sido logrado por ninguno de los participantes en estos grupos de ayuda.

Uno de los dispositivos recomendados fue el RF2400DV de la empresa Laipac, se buscó información referente a este nuevo dispositivo, encontrando que este solo lograba funcionar en un solo sentido (de remoto hacia base), mas no en los dos sentidos (full-duplex) como era requerido para la realización de este proyecto.

En vista de las dificultades ya explicadas y en adición al tiempo necesario para importar nuevos dispositivos, se decidió utilizar transmisores comerciales inalámbricos de audio/video de 2.4GHz.

Cabe destacar que el Capítulo 2 incluye la teoría acerca del funcionamiento y programación del transceiver EWM-900-FDTC, lo cual, deja la libertad para que la parte de transmisión de las señales pueda ser desarrollada de una mejor manera por un estudiante del área de Telecomunicaciones.

4.4 FUENTE DE ALIMENTACIÓN

Para la etapa de amplificación es necesaria una fuente de 5V que alimenta a los amplificadores LM386.

Para la etapa de procesamiento de señal son necesarios varios voltajes; para el funcionamiento del Codec son necesarios dos voltajes: un voltaje análogo (+5VA) que alimenta la parte análoga del dispositivo, mientras que la parte digital es alimentada con 3.3V. En cambio, para el ideal funcionamiento del DSP y de la memoria Flash se necesita 3.3V.

Además se requiere de dos tierras, una digital (GND) y una análoga (AGND). Los elementos conectados a la tierra digital son: el DSP, la memoria Flash y el Codec. Mientras que los elementos conectados a la tierra análoga son: los amplificadores LM386 y el Codec.

La parte análoga esta separada de la parte digital mediante el uso de ferritas (ferrite beads), además se cuenta con filtros capacitivos los mismos que son conectados a las líneas de alimentación de cada uno de los dispositivos que conforman el proyecto.

El esquemático correspondiente a la fuente de alimentación se puede apreciar en la Figura 4.9, mientras que los filtros capacitivos se muestran en el esquemático de la Figura 4.10:

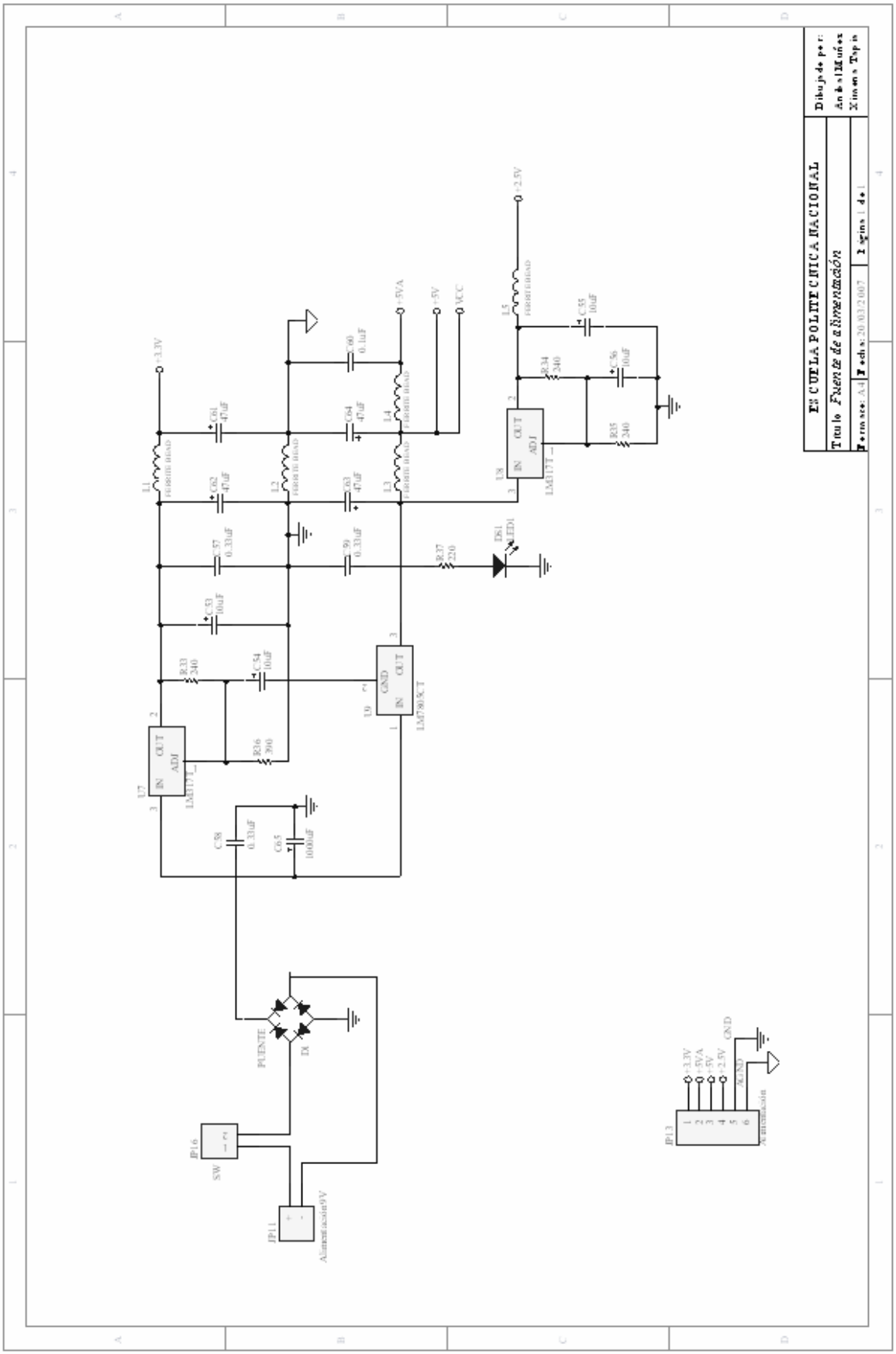


Figura 4.9 Fuente de alimentación.

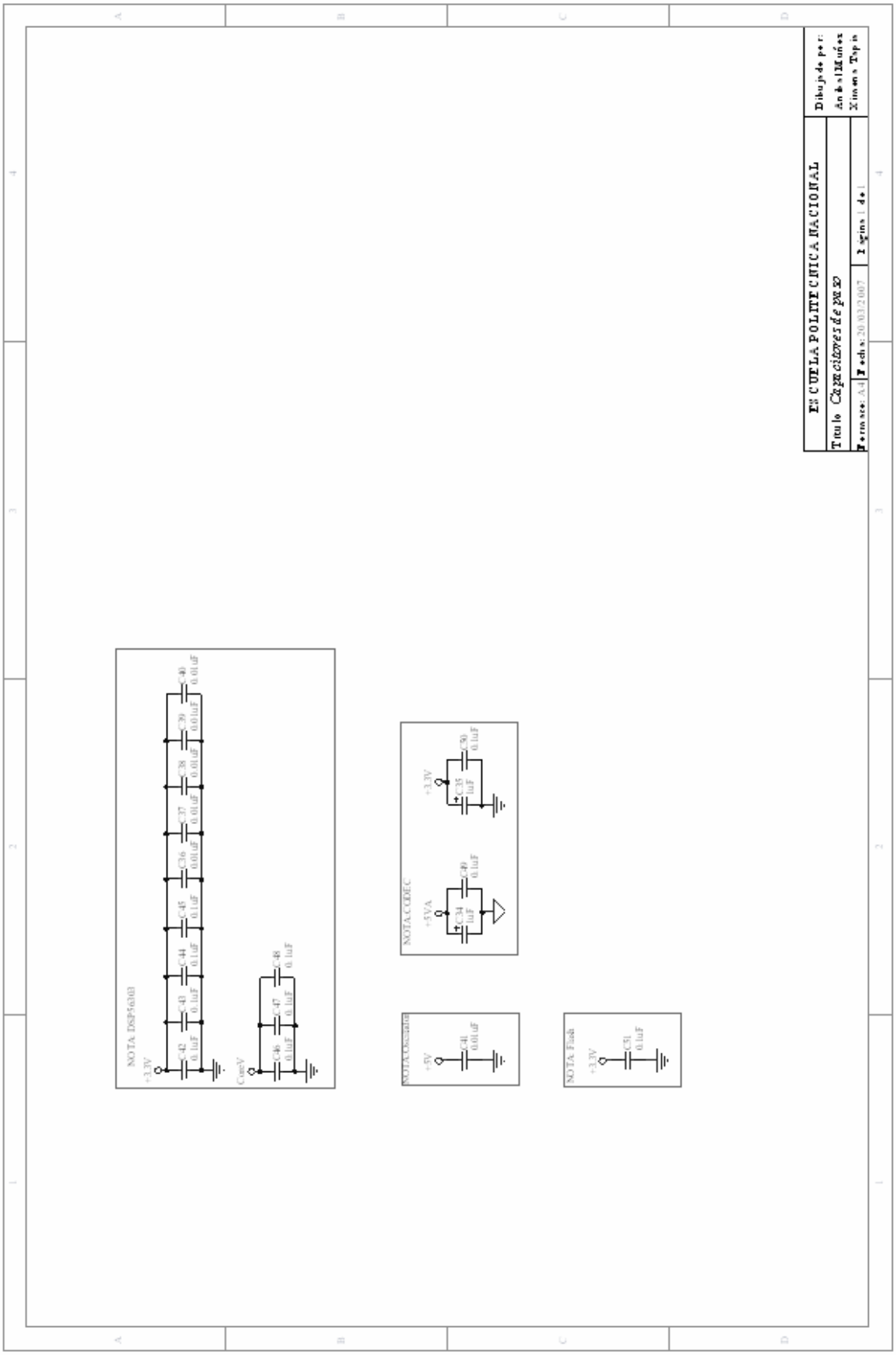


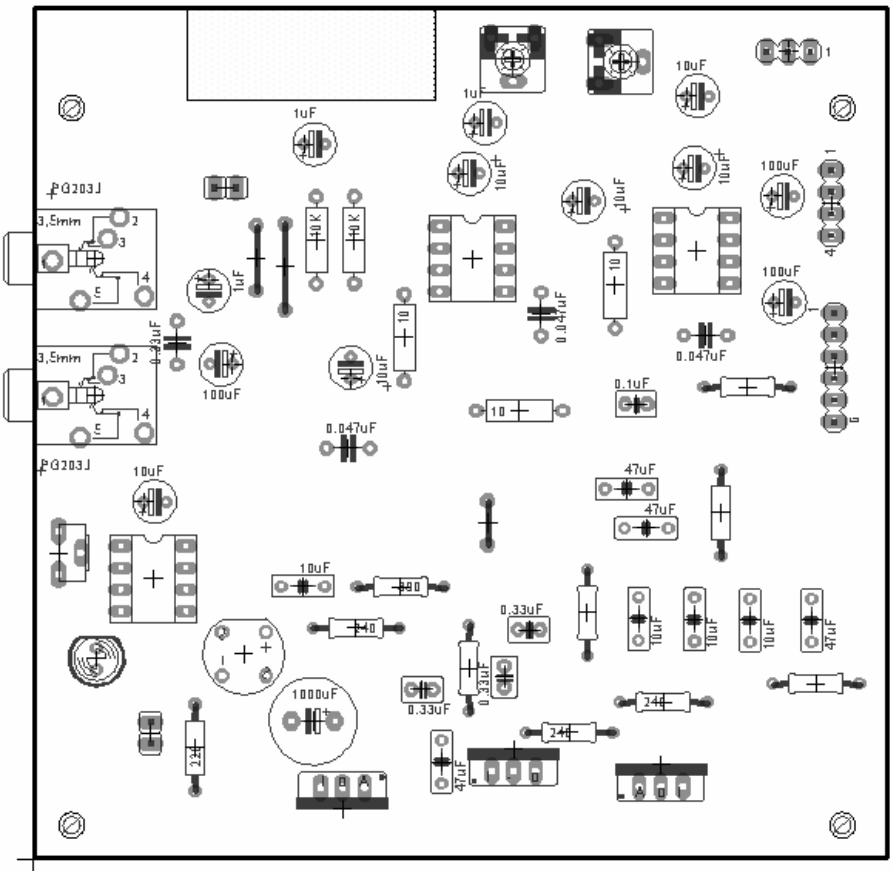
Figura 4.10 Capacitores de paso.

4.5 CIRCUITOS IMPRESOS

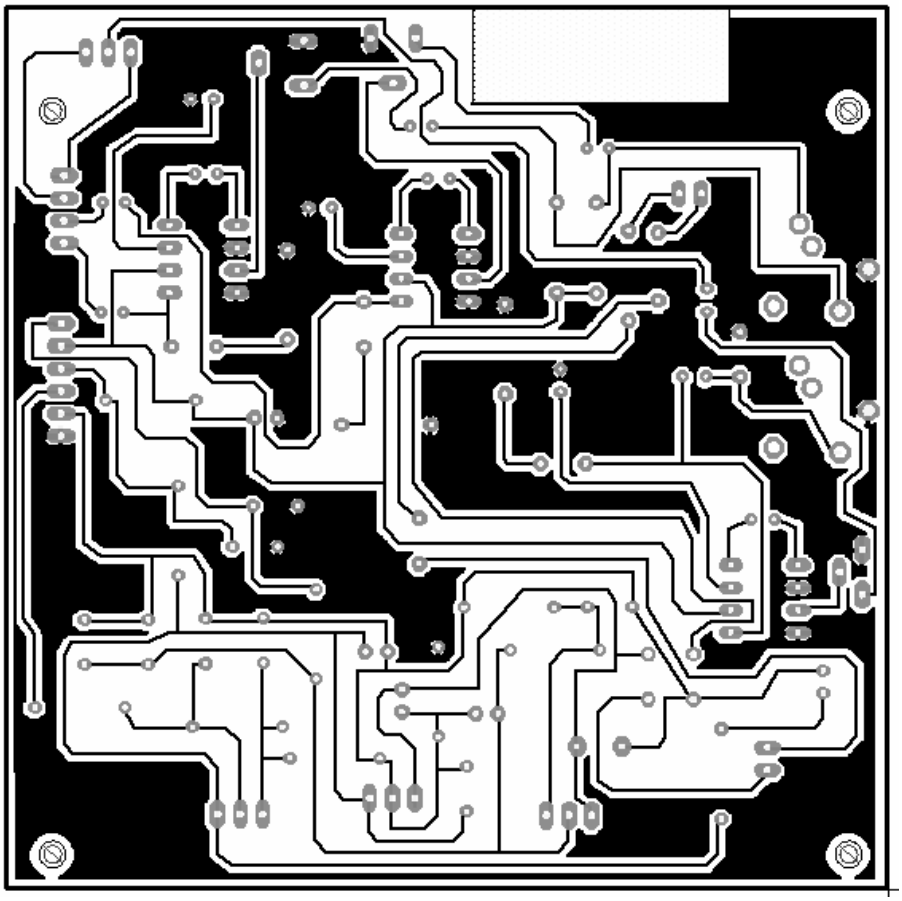
Basándose en los esquemáticos presentados anteriormente se construyeron varios circuitos impresos que abarcan cada una de las etapas que conforman el proyecto.

4.5.1 CIRCUITO IMPRESO PARA LA ETAPA DE AMPLIFICACIÓN Y LA FUENTE DE ALIMENTACIÓN

En la Figura 4.11 se muestran tanto la vista superior como la vista inferior de la etapa de amplificación y la fuente de alimentación:



(a)

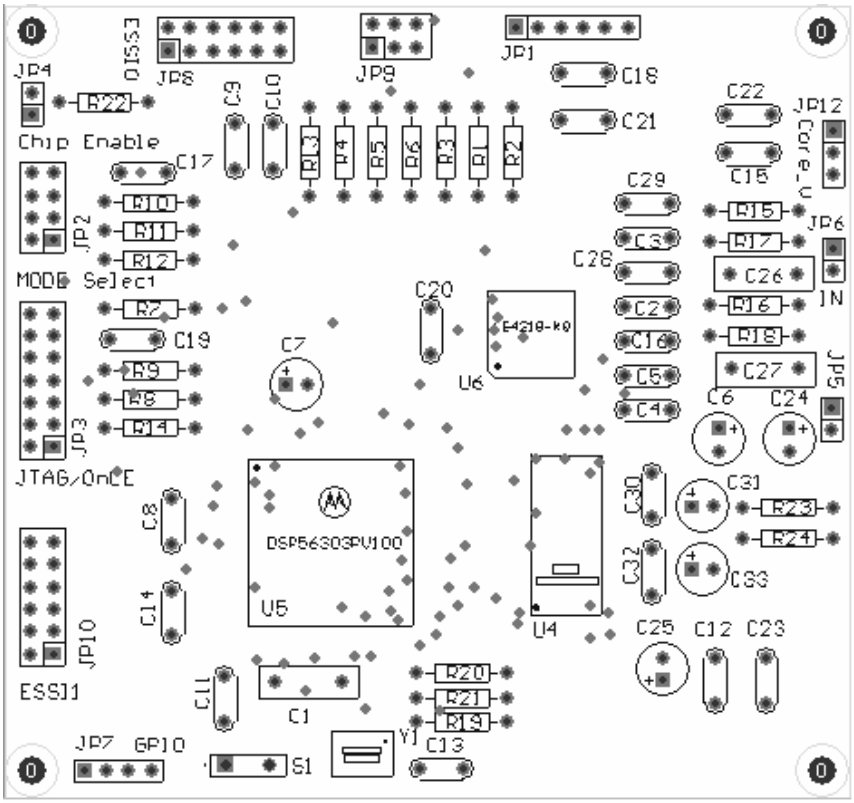


(b)

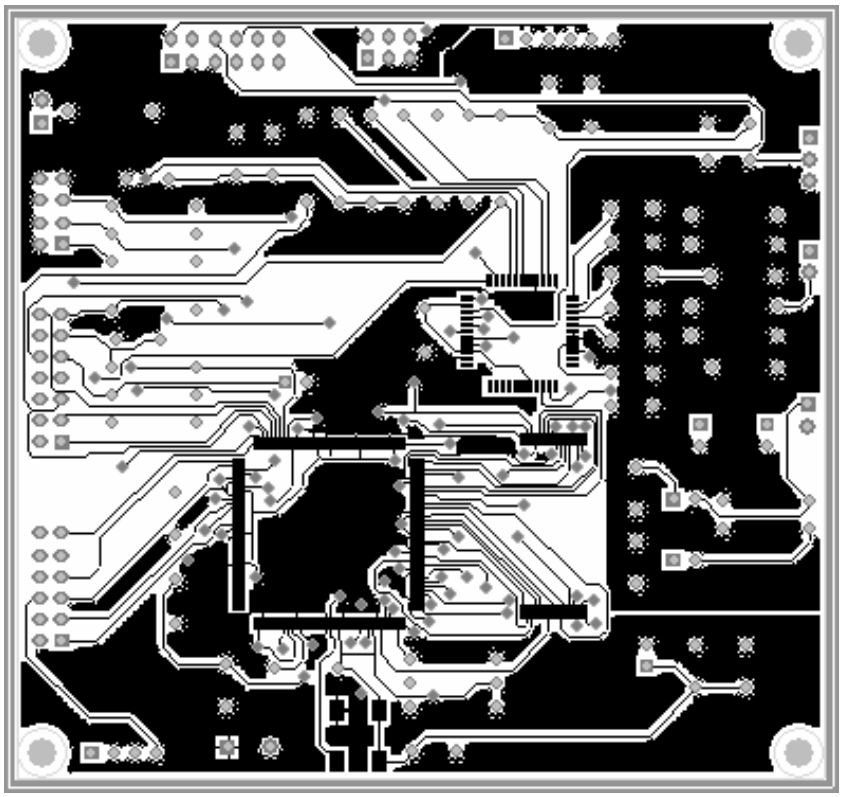
Figura 4.8 **(a)** Vista superior, **(b)** Ruteado capa inferior.

4.5.2 CIRCUITO IMPRESO PARA LA ETAPA DE PROCESAMIENTO DE SEÑAL

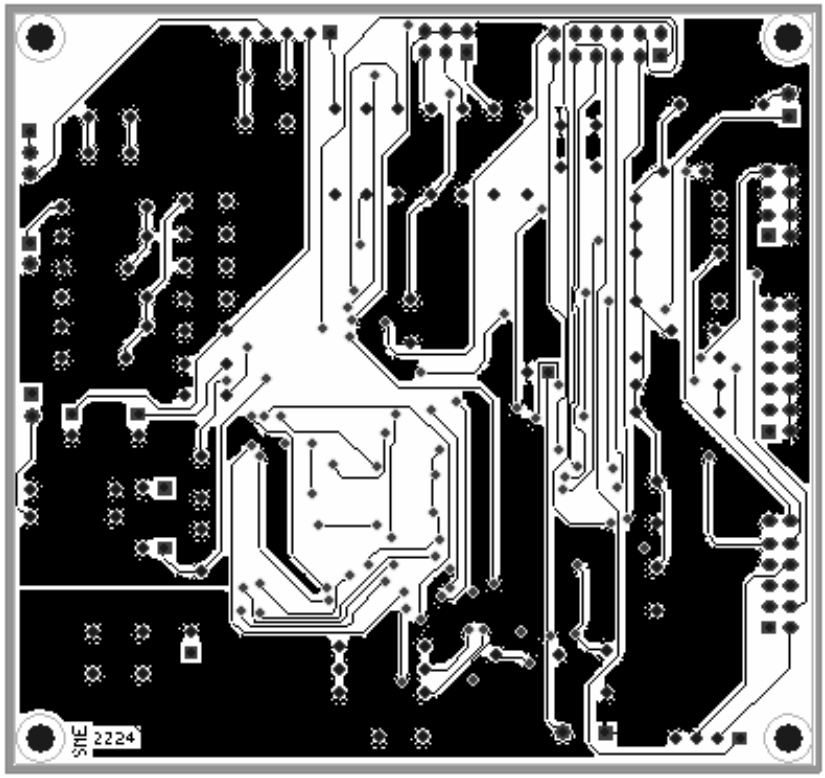
En la Figura 4.12 se indica el circuito impreso de la etapa de procesamiento de señal, la cual incluye al DSP, la memoria Flash y el Codec:


JP4
JP8
JP9
JP1
C18
C21
C22
JP12
C15
Chip Enable
C17
R22
R10
R11
R12
JP2
MODE Select
R7
C19
R9
R8
R14
JP3
JTAG/OnCE
C20
C7
U6
C29
C28
C26
R15
R17
JP6
R16
R18
C27
JP5
C6
C24
C31
R23
R24
C32
C33
C8
C14
DSP56303PV100
U5
U4
C30
C25
C12
C23
JP10
ESSJ1
C11
C1
R20
R21
R19
Y1
C13
JP7
GPIO
S1

(a)



(b)

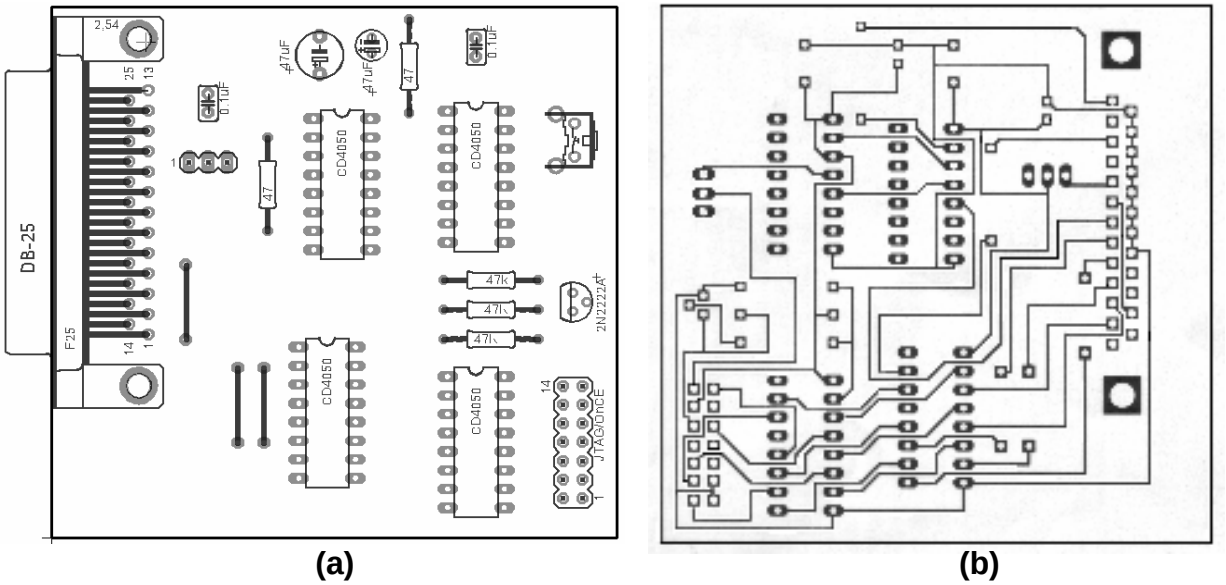


(c)

Figura 4.9 (a) Vista Superior, (b) Ruteado capa superior,
(c) Ruteado capa inferior.

4.5.3 CIRCUITO IMPRESO PARA EL COMMAND CONVERTER

En la Figura 4.13 se muestra el circuito impreso correspondiente al command converter, el cual se usa como interface entre el computador y el DSP:



(a) **(b)**

Figura 4.10 (a) Vista Superior, (b) Ruteado capa inferior.

Una vez que el equipo ha sido armado con todas las partes que lo constituyen se procede a realizar las pruebas del mismo. El análisis de los costos de desarrollo, implementación y fabricación del equipo se muestran en el Capítulo 5.

CAPÍTULO 5

PRUEBAS, RESULTADOS Y COSTOS

En este Capítulo se muestran las pruebas realizadas al sistema, empleando tanto los paquetes computacionales Matlab y LabVIEW, así como los resultados correspondientes a pruebas realizadas en campo.

Los archivos utilizados en las pruebas son los que se obtienen con el grabador de sonidos de Windows. La señal de interés corresponde al archivo **senial.wav**, el ruido contaminante corresponde al archivo **ruido.wav** y la señal resultante (señal filtrada) se graba como **salida.wav**.

5.1 PRUEBAS REALIZADAS EN MATLAB

Para ejecutar el programa en Matlab, en la ventana de comando se debe ingresar **filtroa**, el cual es un archivo **.m** y en el cual están incluidas todas las subrutinas para la realización del filtro adaptivo.

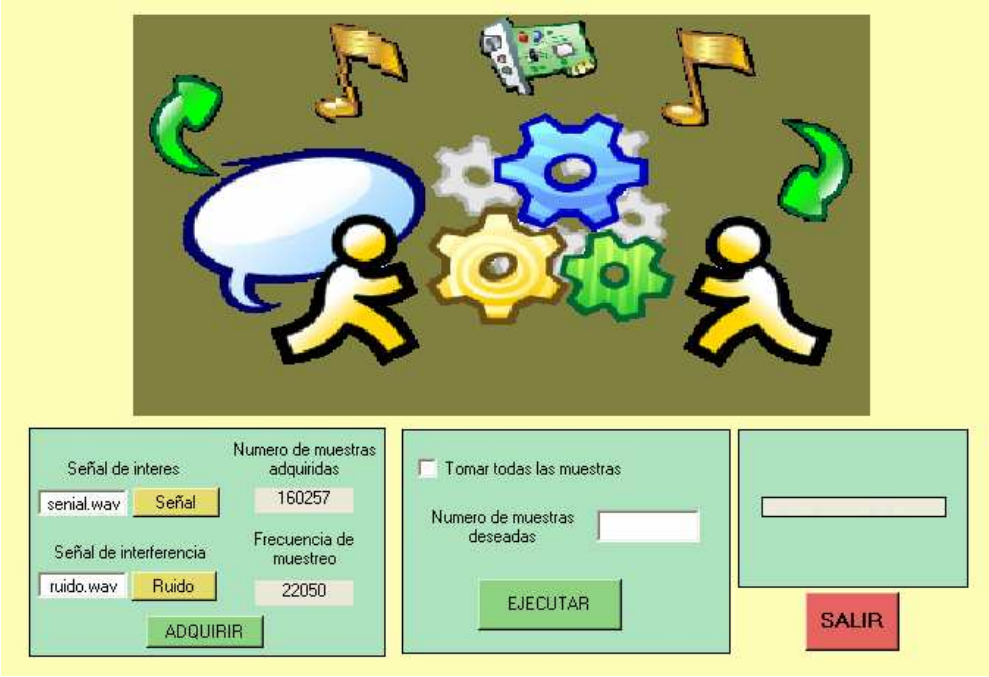


Figura 5.1 Pantalla en Matlab.

Como se indica en la Figura 5.1, las señales de entrada son dos archivos **.wav** (senial.wav y ruido.wav), los cuales al ser transformadas en vectores por Matlab tienen una dimensión de 160257 muestras cada una.

Los resultados obtenidos corresponden a pruebas realizadas para el siguiente número de muestras:

- 10000 muestras
- 20000 muestras
- 80000 muestras y,
- El total de muestras adquiridas

Para 10000 muestras el resultado se indica en la Figura 5.2:

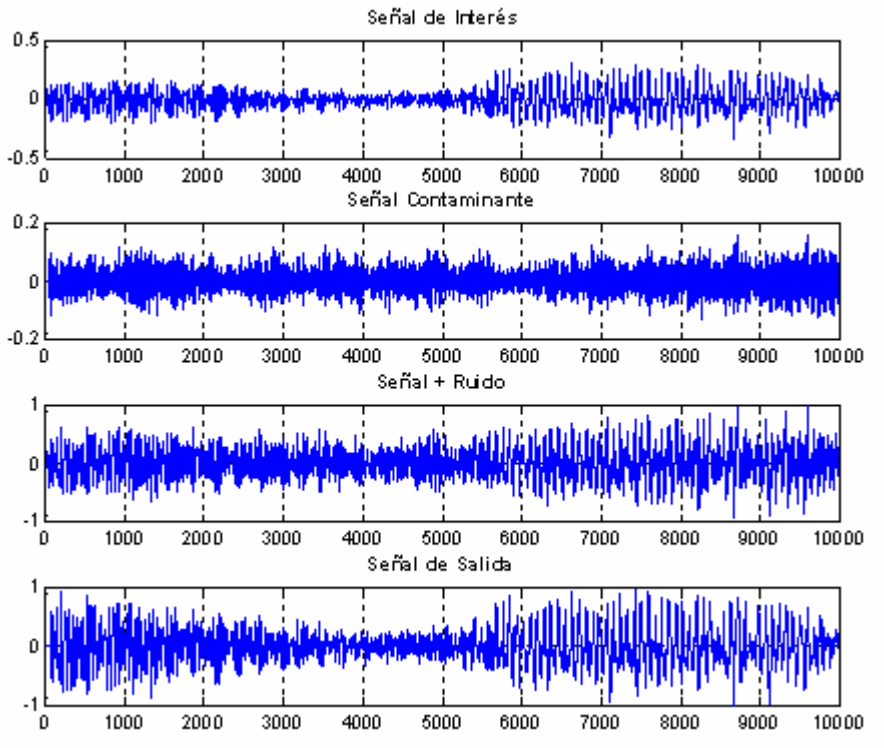


Figura 5.2 Señales obtenidas con 10000 muestras.

Para 20000 muestras el resultado se indica en la Figura 5.3:

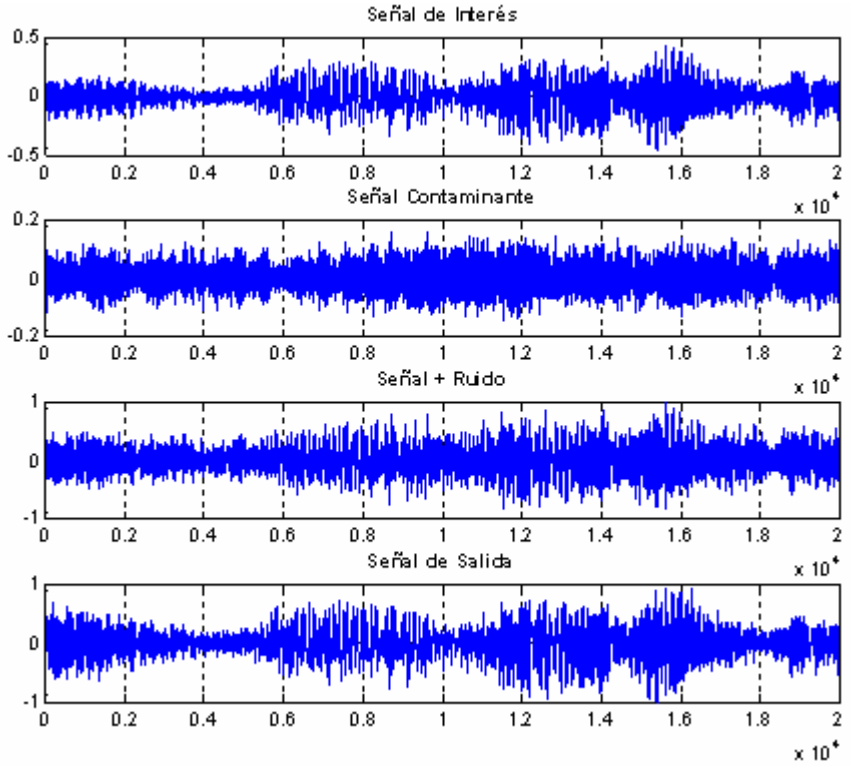


Figura 5.3 Señales obtenidas con 20000 muestras.

Para 80000 muestras el resultado se indica en la Figura 5.4:

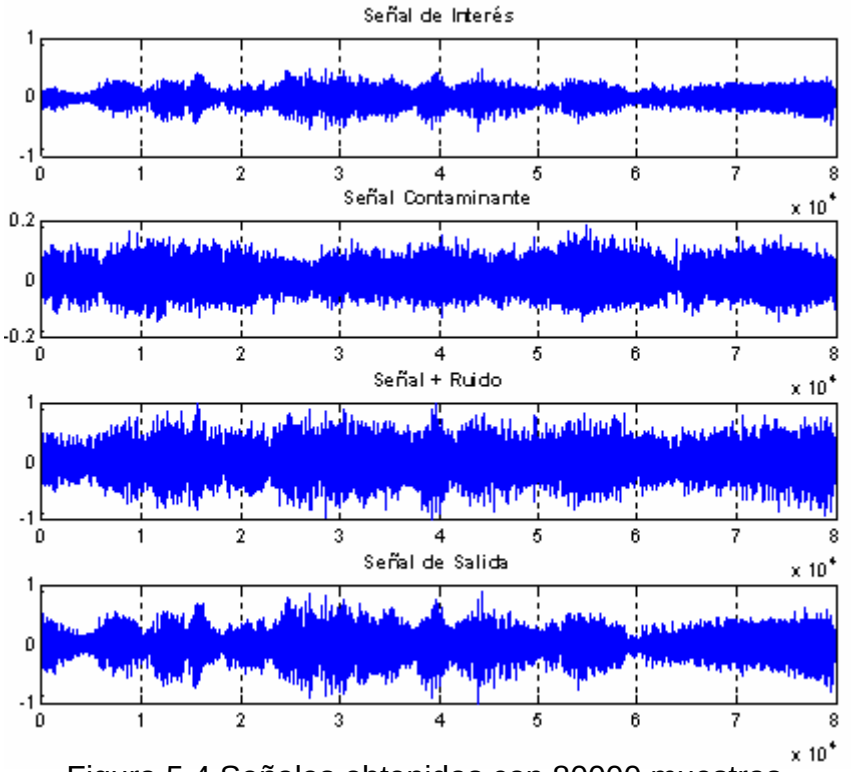


Figura 5.4 Señales obtenidas con 80000 muestras.

Para todas las muestras (160257) el resultado se indica en la Figura 5.5:

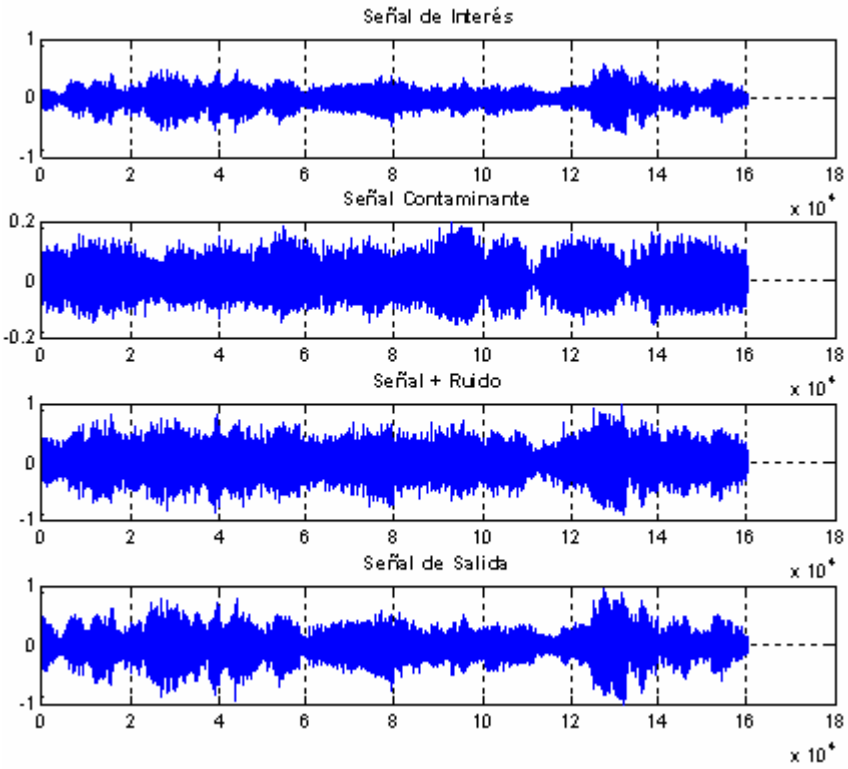


Figura 5.5 Señales obtenidas con todas las muestras.

En las pruebas realizadas en Matlab, se pudo notar que el tiempo en el que la salida del filtro se estabiliza es aproximadamente de dos segundos, lo cual hace ideal la aplicación del filtro seleccionado para dar solución al problema planteado en este proyecto.

5.2 PRUEBAS REALIZADAS EN LABVIEW

Para ejecutar el programa en LabVIEW se debe abrir el archivo **filtroa.vi**, el cual contiene los subVIs necesarios para ejecutar el filtro adaptivo.

Como se indicó en el Capítulo 3, se debe seleccionar las señales de entrada (senial.wav y ruido.wav). Una vez adquiridas dichas señales se procede a escoger el número de muestras a ser analizadas, para este caso en particular se cuenta con un total de 159504 muestras, tal como se indica en la Figura 5.6:

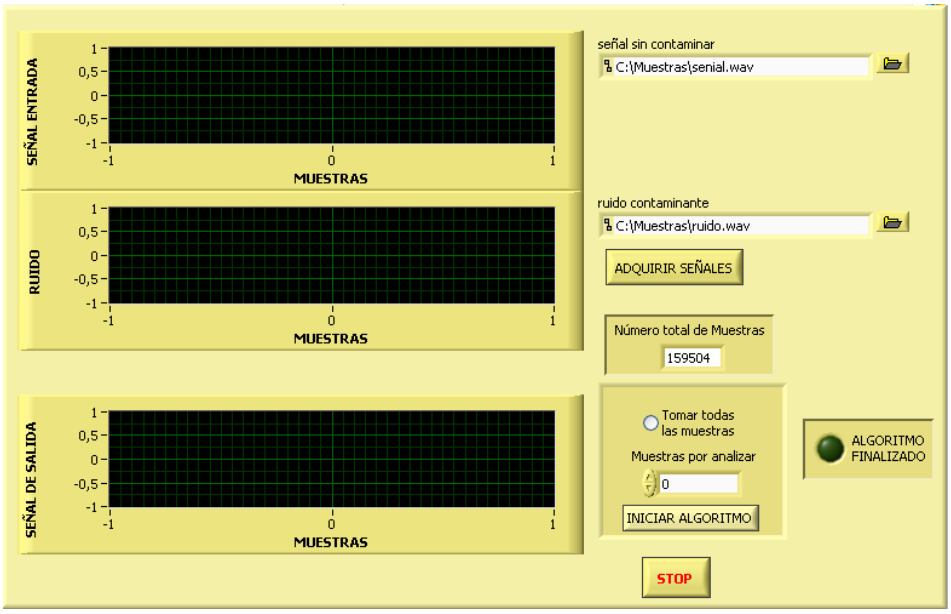


Figura 5.6 Pantalla en LabVIEW para seleccionar las señales de entrada.

Los resultados obtenidos en LabVIEW corresponden a pruebas realizadas para el siguiente número de muestras:

- 10000 muestras
- 20000 muestras y,
- 87000 muestras

Para 10000 muestras el resultado se indica en la Figura 5.7:

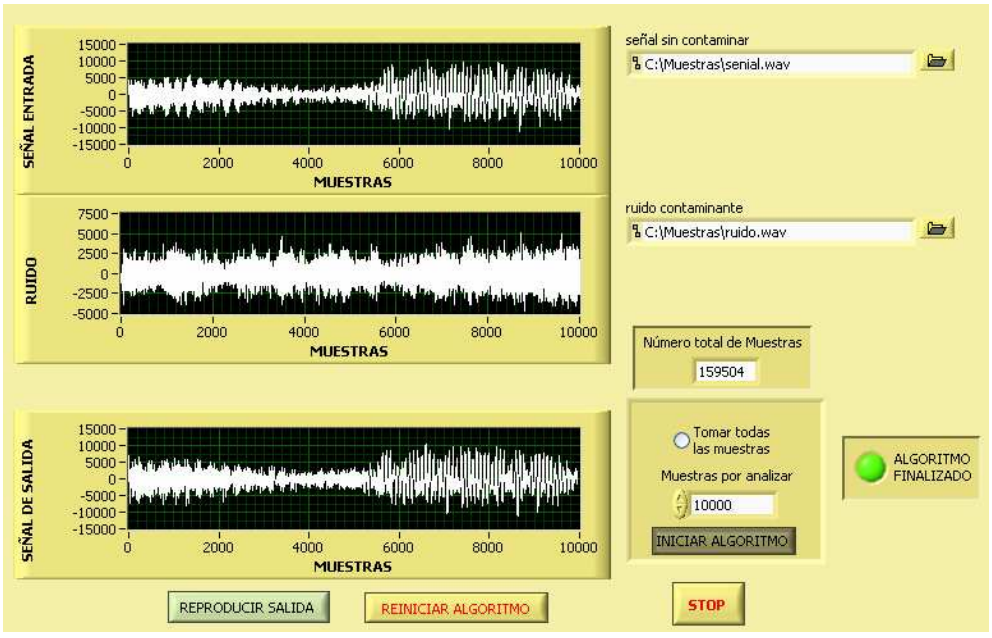


Figura 5.7 Señales obtenidas con 10000 muestras.

Para 20000 muestras el resultado se indica en la Figura 5.8:

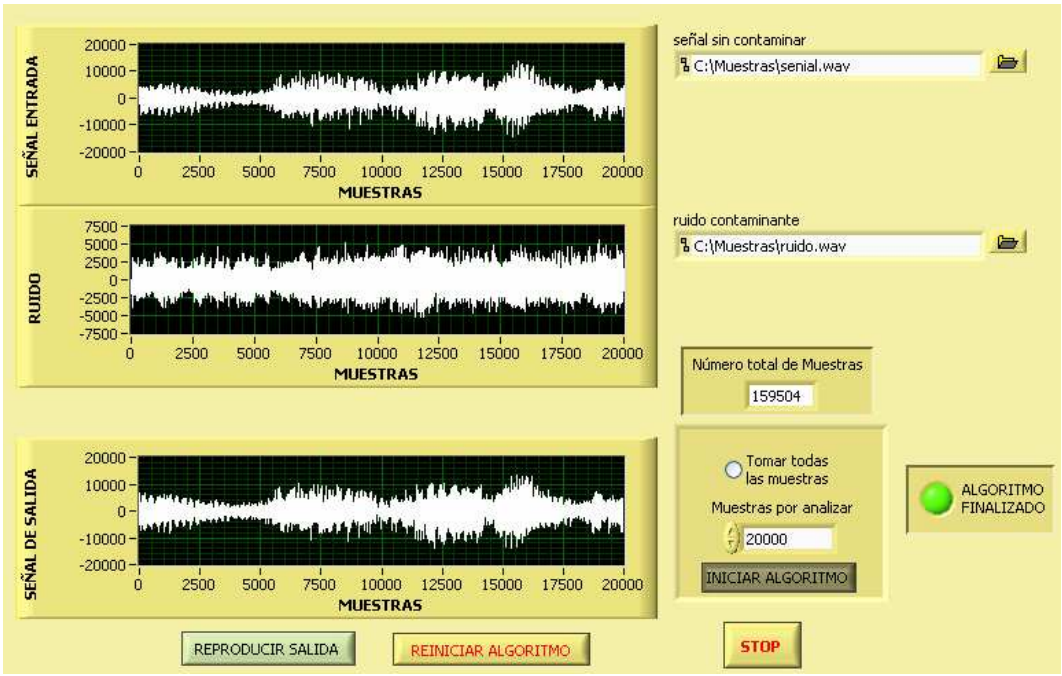


Figura 5.8 Señales obtenidas con 20000 muestras.

Para 87000 muestras el resultado se indica en la Figura 5.9:

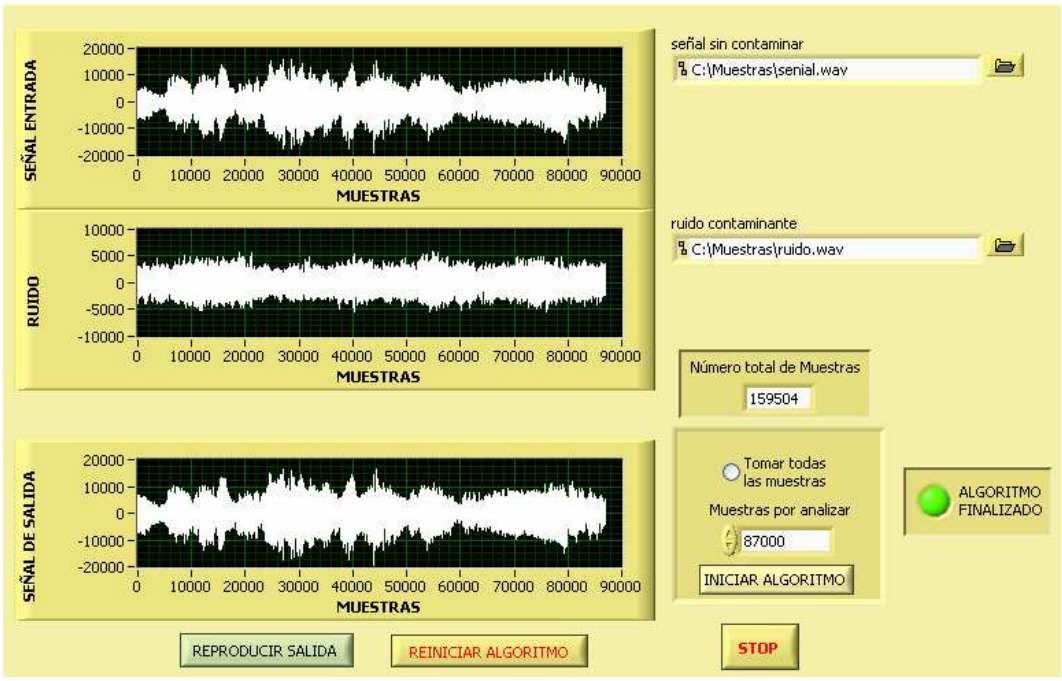


Figura 5.9 Señales obtenidas con 87000 muestras.

En las Figuras 5.7, 5.8, y 5.9 se observa que las pruebas realizadas en LabVIEW proporcionan resultados parecidos a los obtenidos en Matlab. Sin embargo, hay una gran diferencia en la ejecución de los algoritmos en los dos paquetes computacionales. En LabVIEW el procesar 87000 muestras lleva un tiempo aproximado de doce horas, mientras que en Matlab analizar el mismo número de muestras significa un tiempo de procesamiento de una hora, lo cual indica claramente que Matlab es una mejor herramienta en la ejecución de algoritmos iterativos.

Por ello, cabe resaltar la importancia de la selección de un DSP para el desarrollo de este proyecto, ya que el DSP puede ejecutar estas tareas de filtrado en tiempo real.

5.3 PRUEBAS DE CAMPO

Las pruebas de campo hacen referencia a la aplicación del filtro digital adaptivo en el DSP. Estos resultados son de carácter subjetivo, pues para apreciar el funcionamiento del filtro en el DSP la señal resultante debe ser escuchada y juzgada por el usuario del equipo.

5.4 COSTOS DEL EQUIPO

En el Capítulo 4 se pudo observar que el proyecto consta de dos circuitos impresos, uno correspondiente a la etapa de amplificación y fuente de alimentación, y otro correspondiente a la etapa de procesamiento de señal.

Los precios de los elementos que conforman el circuito impreso para la etapa de amplificación y la fuente de alimentación se detallan en la Tabla 5.1:

Tabla 5.1 Precios de los componentes de la etapa de amplificación y fuente

Descripción	Designación	Valor	Cantidad	Valor unitario	Valor total
Capacitor Polarizado	C1 C2 C3	1uF	3	0,06	0,18
Capacitor Polarizado	C4 C5 C6	100uF	3	0,06	0,18
Capacitor Polarizado	C7 C8 C9 C10 C11 C12 C53 C54 C55 C56	10uF	10	0,06	0,60
Capacitor Polarizado	C65	1000uF	1	0,13	0,13
Capacitor Polarizado	C61 C62 C63 C64	47uF	4	0,06	0,24
Capacitor Cerámico	C13 C14 C15	0.01uF	3	0,08	0,24
Capacitor Cerámico	C16 C17 C18	0.047uF	3	0,08	0,24
Capacitor Cerámico	C19 C57 C58 C59	0.33uF	4	0,30	1,20
Capacitor Cerámico	C60	0.1uF	1	0,06	0,06
Resistencia	R1 R2	10KΩ	2	0,02	0,04
Resistencia	R6 R7 R8	10Ω	3	0,02	0,06
Resistencia	R33 R34 R35	240Ω	3	0,02	0,06
Resistencia	R36	390Ω	1	0,02	0,02
Resistencia	R37	220Ω	1	0,02	0,02
Potenciómetro	R3 R4	10KΩ	2	0,49	0,98
Potenciómetro	R5	100Ω	1	0,49	0,49
Molex de 6 pines	JP13		1	0,80	0,80
Molex de 4 pines	JP14		1	0,78	0,78
Header 3 pines 1 fila	JP15		1	0,05	0,05
Header 2 pines 1 fila	JP11 JP16		2	0,05	0,10
Puente de diodos	D1		1	0,35	0,35
Diodo led	DS1		1	0,08	0,08
Ferrita	L1 L2 L3 L4 L5		5		0,00
Jack estéreo 3.5mm	J1 J2		2	0,12	0,24
Amplificador LM386	U1 U2 U3		3	0,56	1,68
Regulador de voltaje LM317	U7 U8		2	0,80	1,60
Regulador de voltaje LM7805	U9		1	0,70	0,70
				TOTAL $:	11,12

Los precios de los elementos que conforman el circuito impreso para la etapa de procesamiento de señal se detallan en la Tabla 5.2:

Tabla 5.2 Precios de los componentes de la etapa de procesamiento de señal

Descripción	Designación	Valor	Cantidad	Valor unitario	Valor total
Capacitor Polarizado	C21 C22 C34 C35	1uF	4	0,06	0,24
Capacitor Polarizado	C23	10uF	1	0,06	0,06
Capacitor Polarizado	C52	4.7uF	1	0,06	0,06
Capacitor Cerámico	C29 C30	0.0022uF	2	0,08	0,16
Capacitor Cerámico	C31 C32 C33	0.47uF	3	0,08	0,24
Capacitor Cerámico	C26 C27	300pF	2	0,08	0,16
Capacitor Cerámico	C20 C36 C37 C38 C39 C40 C41	0.01uF	7	0,08	0,56

Capacitor Cerámico	C24 C25	0.33uF	2	0,30	0,60
Capacitor Cerámico	C28 C42 C43 C44 C45 C46 C47 C48 C49 C50 C51	0.1uF	11	0,06	0,66
Resistencia	R9 R10 R11 R12 R13 R14 R15 R16 R17 R18 R19 R20 R21 R22 R23 R24 R25	10KΩ	17	0,02	0,34
Resistencia	R26 R27 R28 R29	5.6KΩ	4	0,02	0,08
Resistencia	R30 R31	39KΩ	2	0,02	0,04
Resistencia	R32	5.1KΩ	1	0,02	0,02
Molex de 6 pines	JP1		1	0,80	0,80
Molex de 2 pines	JP5 JP6		2	0,34	0,68
Molex de 4 pines	JP7		1	0,78	0,78
Header 3 pines 1 fila	JP12		1	0,05	0,05
Header 2 pines 1 fila	JP4		1	0,05	0,05
Header 8 pines 2 filas	JP2		1	0,05	0,05
Header 14 pines 2 filas	JP3		1	0,05	0,05
Header 6 pines 2 filas	JP9		1	0,05	0,05
Header 12 pines 2 filas	JP8 JP10		2	0,05	0,10
Pulsador	S1		1	0,15	0,15
Oscilador de 12.288MHz	Y1		1	2,38	2,38
Audio Codec CS4218-KQ	U6		1	21,00	21,00
DSP56303PV100	U5		1	27,00	27,00
Flash AT29LV010A-15TC	U4		1	2,62	2,62
				TOTAL $:	58,98

Adicional a los costos ya mencionados se tienen los costos de fabricación, en los cuales se contempla la elaboración de los circuitos impresos, así como también la fabricación de las orejeras, entre otros. Estos costos se detallan en la Tabla 5.3:

Tabla 5.3 Costos de fabricación

Descripción	**Valor Total**
Elaboración de circuitos impresos	50,00
Elaboración orejeras	7,50
Construcción caja	9,00
Batería recargable	9,00
Equipos para transmisión/recepción	65,00
Costos de importación de elementos	15,00
TOTAL $:	155,50

En las Tablas 5.1, 5.2 y 5.3 se hace referencia a los costos de construcción por equipo. Para la realización de este proyecto es necesaria la elaboración de dos equipos, entonces, los costos totales del proyecto se indican en la Tabla 5.4:

Tabla 5.4 Costo total de los equipos

Descripción	Cantidad	Valor Unitario	Valor Total
Etapa de amplificación y fuente	2	11,12	22,24
Etapa de procesamiento de señal	2	58,98	117,96
Costos de fabricación	2	155,50	311,00
		TOTAL $:	451,20

Los costos de investigación se basan en la relación entre, el costo por investigador y el tiempo empleado para el desarrollo del proyecto, estos costos se detallan en la Tabla 5.5:

Tabla 5.5 Costos de investigación

Investigadores	Tiempo utilizado	Mensualidad	Valor Total
Aníbal Muñoz Raza	10 meses	200	2000
Ximena Tapia León	10 meses	200	2000
		TOTAL $:	4000

5.5 FOTOS DEL PROTOTIPO

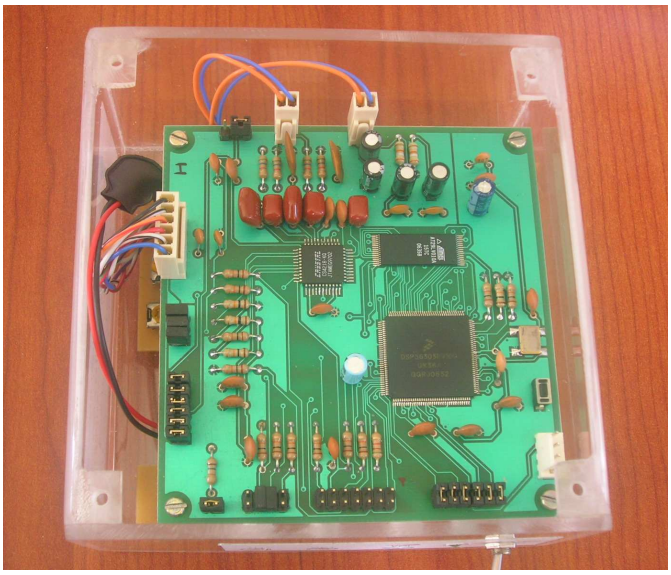
Cada uno de los prototipos construidos para este Proyecto de Titulación están conformados de las siguientes partes:

1. Módulo
2. Orejeras industriales
3. Transmisor y Receptor de A/V

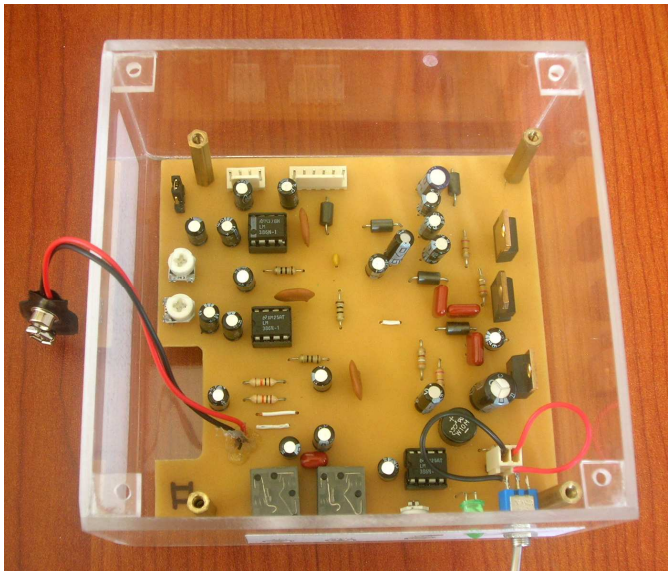
En el interior del módulo se encuentran los circuitos impresos correspondientes a:

- La etapa de procesamiento de señal y,
- La etapa de amplificación y fuente de alimentación

Los circuitos impresos se pueden apreciar en la Figura 5.10:



(a)



(b)

Figura 5.10 (a) Etapa de procesamiento de señal; (b) Etapa de amplificación y fuente de alimentación.

Las orejeras son de tipo industrial, estas están equipadas con dos micrófonos y un par de audífonos. Los micrófonos captan las señales de entrada al filtro, estos

micrófonos están posicionados en las orejeras tal como se indica en la Figura 5.11:

Figura 5.11 Posición de los micrófonos en las orejeras

La señal resultante del filtro es transmitida de manera inalámbrica, de manera que para poder ser escuchada las orejeras cuentan con dos audífonos ubicados en la parte interna de la orejera. Los transmisores y receptores utilizados son de tipo comercial los cuales se muestran en la Figura 5.12:

Figura 5.12 Transmisor y receptor de A/V

Para armar el equipo completo la señal de los micrófonos debe ser conectada al jack de entrada en el módulo, mientras que la salida debe ser conectada a los terminales de audio en el transmisor de A/V. Para poder escuchar la señal los audífonos de las orejeras deben conectarse a los terminales de audio del receptor de A/V. El equipo completo se puede apreciar en la Figura 5.13:

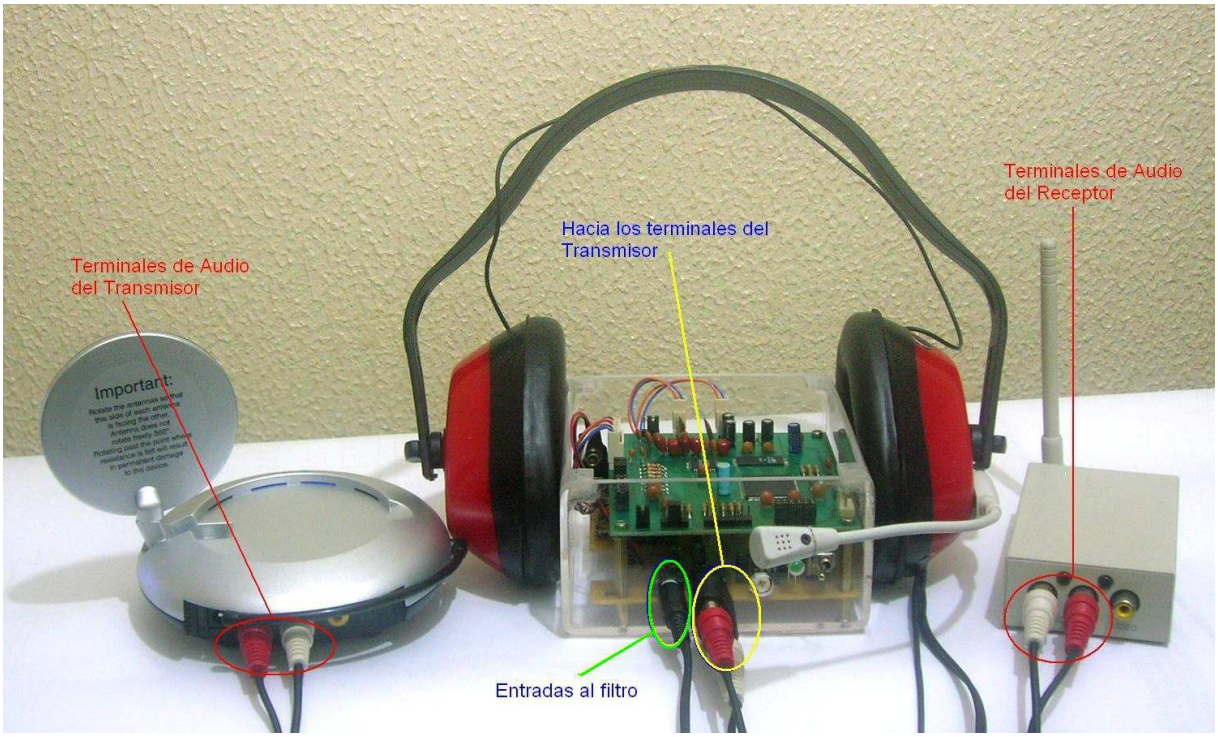


Figura 5.13 Equipo completo

Cada operador debe utilizar un equipo semejante al mostrado en la Figura 5.13.

CAPÍTULO 6

CONCLUSIONES Y RECOMENDACIONES

6.1 CONCLUSIONES

Luego de analizar los resultados de las pruebas a las que se sometió el sistema desarrollado, se puede extraer las conclusiones siguientes:

- La implementación de un filtro digital ofrece mayor flexibilidad, en la reconfiguración de sus parámetros, que sus correspondientes sistemas análogos
- El DSP mostró la capacidad para procesar datos en tiempo real, además, sus funciones de manejo de arreglos, el acceso paralelo a memoria y la ejecución de instrucciones en paralelo hacen del DSP una herramienta muy eficiente
- Las herramientas de simulación en software como Matlab permiten conocer y anticipar el comportamiento de las señales, además sirve de base para la ejecución de los algoritmos en el DSP ,sin embargo, se puede decir que una vez que se familiariza con el lenguaje y herramientas que posee el DSP, la elaboración de los algoritmos se puede optimizar
- Una de las ventajas que implica trabajar en Matlab es el desarrollo del código fuente para la programación del algoritmo, pues esto implica el uso de algunas instrucciones básicas para el manejo de arreglos y unos pocos lazos. Facilidad que no está presente en el uso de LabVIEW, pues el uso de arrays (arreglos) dentro de lazos iterativos suele tornarse confuso y puede llevar a errores en la apreciación del funcionamiento del algoritmo
- Los costos de implementación del equipo son altos debido a que en el país no se cuenta con los elementos necesarios, debiendo estos, ser

importados, además no se cuenta con la tecnología necesaria para trabajar con elementos de montaje superficial como son resistencias, capacitores, transistores, entre otros

- La experiencia adquirida en este proyecto permite concluir que existe en nuestro medio una gran dependencia tecnológica de los países industrializados, pues el tamaño del equipo es grande, esto debido a que los elementos ideales para la implementación del equipo son de montaje superficial, pero en el país no se cuenta la tecnología adecuada para el montaje de estos elementos ni con la distribución de los mismos

6.2 RECOMENDACIONES

En las hojas de datos de los circuitos integrados se hacen sugerencias que deben ser tomadas en cuenta, tales como:

- Tener planos de tierra sólidos tanto para la tierra digital (GND) como para la tierra análoga (AGND), esto garantiza una baja impedancia entre los pines que son conectados a tierra
- Los voltajes de polarización (+5VA y +3.3V) deben conectarse al Codec a través de un núcleo de ferrita, de igual manera la tierra análoga (AGND) y la tierra digital (GND) deben separarse mediante un núcleo de ferrita

Para el manejo del equipo se recomienda:

- Tener cuidado al conectar el puerto JTAG/OnCE del DSP con el Command Converter, respetando la distribución de pines, puesto que un error en la conexión puede quemar el DSP
- Tener cuidado en la manipulación de los circuitos de tecnología CMOS, como son el DSP, Codec y la memoria Flash, pues son susceptibles a la electricidad estática

Se recomienda el uso del transceiver EWM-900-FDTC, ya que, el transmisor y receptor utilizados en este Proyecto son de tipo comercial, perdiendo así la característica de portátil debido al tamaño del equipo.

REFERENCIAS BIBLIOGRÁFICAS

[1] Disponible en Internet: http://www.icop.com.ar/iorlsf/proteccion.html

[2] S. S. Soliman, M. D. Srinath, *Señales y Sistemas continuos y discretos*, 2ed., 1999.

[3] Disponible en Internet: archivo77.iespana.es/apuntes/segundo/ads/adsfiltrosddigitales.pdf

[4] A. J. Rubio, *Procesado Digital de Señales*, Universidad de Granada, Abril 2002.

[5] "*Cápitulo 9: Diseño de filtros digitales (parte 2)"*, 1999. Disponible en internet: www1.ceit.es/asignaturas/tratamiento%20digital/tema9.pdf

[6] Prof Dr. Pablo García Estévez, "*Aplicaciones de las redes Neuronales en las Finanzas"*, 2002. Disponible en Internet: www.ucm.es/BUCM/cee/doc/02-05/0205.pdf

[7] B. Morcego, M. Cugueró, "*Comparación de Implementaciones en C y Matlab de Filtros Adaptativos para DSP*", 1999. Disponible en Internet: www.cea-ifac.es/actividades/jornadas/XXII/documentos/A_06_IC.pdf

[8] Disponible en Internet: http://bips.bi.ehu.es/prj/ruido/

[9] M. Jiménez, *Cancelación adaptiva de ruido*, Tesis de Grado, E.P.N., Quito, 1988.

[10] J. Rubia, J. de la Casa, "*Introducción a los D.S.P.*", 2001. Disponible en Internet: http://www.redeya.com/electronica/tutoriales/dsp1/dsp1.htm

[11] R. Mateos Gil, "*Introducción a los procesadores digitales de señal*", Junio 2000. Disponible en Internet: www.depeca.uah.es/docencia/ING-TELECO/sed/documentos_clase/tema5.pdf

[12] J. Salazar, "*Procesadores digitales de señal (DSP): Arquitecturas y criterios de selección*". Dpto. Ingeniería Electrónica. Centro de sistemas y sensores electrónicos, Universidad Politécnica de Cataluña.

[13] "*Parámetros en la elección de un DSP* ". Disponible en Internet: http://www.geocities.com/Colosseum/Track/8462/parametr.html

[14] R. Huerta, "*Lectura 1: Introducción a los DSP's*", Universidad Técnica Federico Santa María, Dpto. de Electrónica, pp. 10-11, Febrero 2004.

[15] CRYSTAL, "*CS4218-KQ: 16-Bit Stereo Audio Codec*", September 1996. Disponible en Internet: http://pdf1.alldatasheet.net/datasheet-pdf/view/58088/CIRRUS/CS4218-KQ.html

[16] Radiotronix, "*EWM-900-FDTC*", rev 1.2, February 2004. Disponible en Internet: www.radiotronix.com/datasheets/ewm-900-fdtc-datasheet.pdf

[17] ATMEL, "*1-megabit (128k x 8) 3-volt only Flash Memory AT29LV010A*", 2005. Disponible en Internet: www.atmel.fi/dyn/resources/prod_documents/doc0520.pdf

[18] Motorola, "DSP56303 User's Manual", 1996. Disponible en Internet: http://www.motorola-dsp.com

[19] National Semiconductor, "*LM386 Low Voltage Audio Amplifier*". Disponible en Internet: http://www.national.com/ads-cgi/viewer.pl/ds/LM/LM386.pdf

ANEXOS

Anexo A

Motorola DSP56303

MOTOROLA
Semiconductor Product Sector

Order Number: DSP56303P/D
Rev. 1, 10/1999

DSP56303

PRODUCT BRIEF: 24-BIT DIGITAL SIGNAL PROCESSOR

The DSP56303 is a member of the DSP56300 core family of programmable CMOS Digital Signal Processors (DSPs). This family uses a high-performance, single clock cycle per instruction engine providing a twofold performance increase over Motorola's popular DSP56000 core family while retaining code compatibility. Significant architectural features of the DSP56300 core family include a barrel shifter, 24 bit addressing, instruction cache, and DMA. The DSP56303 performs at 66/80/100 MIPS using an internal 66/80/100 MHz clock at 3.0–3.6 volts. The DSP56300 core family offers a rich instruction set and low power dissipation, as well as increasing levels of speed and power, enabling wireless, telecommunications, and multimedia products.

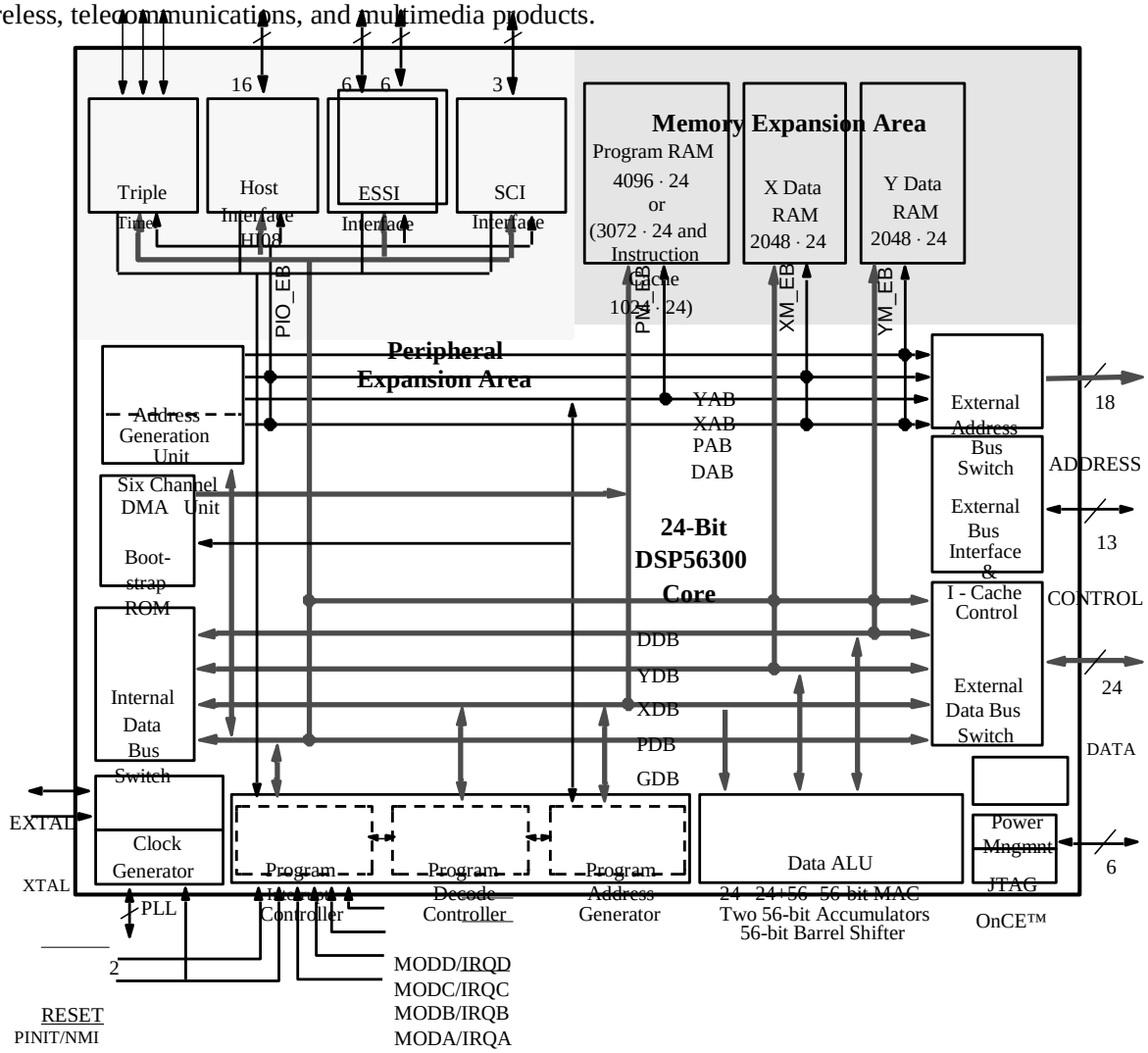


Figure 1. DSP56303 Block Diagram

DSP56303 FEATURES

- High-performance DSP56300 core
  - 66/80/100 Million Instructions Per Second (MIPS) with a 66/80/100 MHz clock
  - Object code compatible with the DSP56000 core
  - Highly parallel instruction set
  - Fully pipelined 24 x 24-bit parallel multiplier-accumulator
  - 56-bit parallel barrel shifter
  - 24-bit or 16-bit arithmetic support under software control
  - Position independent code support
  - Addressing modes optimized for DSP applications
  - On-chip instruction cache controller
  - On-chip memory-expandable hardware stack
  - Nested hardware DO loops
  - Fast auto-return interrupts
  - On-chip concurrent six-channel DMA controller
  - On-chip Phase Lock Loop (PLL) and clock generator
  - On-Chip Emulation (OnCE™) module
  - Joint Test Action Group (JTAG) Test Access Port (TAP)
  - Address tracing mode that reflects internal accesses at the external port
- On-chip memories
  - Program RAM, Instruction Cache, X data RAM, and Y data RAM size are programmable:

Instruction Cache	Switch Mode	Program RAM Size	Instruction Cache Size	X Data RAM Size	Y Data Ram Size
disabled	disabled	4096 · 24-bit	0	2048 · 24-bit	2048 · 24-bit
enabled	disabled	3072 · 24-bit	1024 · 24-bit	2048 · 24-bit	2048 · 24-bit
disabled	enabled	2048 · 24-bit	0	3072 · 24-bit	3072 · 24-bit
enabled	enabled	1024 · 24-bit	1024 · 24-bit	3072 · 24-bit	3072 · 24-bit

  - 192 · 24-bit bootstrap ROM

- Off-chip memory expansion
  - Data memory expansion to two 256 K x 24-bit word memory spaces
  - Program memory expansion to one 256 K x 24-bit word memory space
  - External memory expansion port
  - Chip Select Logic that require no additional circuitry to interface to SRAMs and SSRAMs
  - On-chip DRAM controller requires no additional circuitry to interface to DRAMs
- On-chip peripherals
  - 8-bit parallel Host Interface (HI08), ISA-compatible bus interface, providing a cost-effective solution for applications not requiring the PCI bus
  - Two Enhanced Synchronous Serial Interfaces (ESSI)
  - Serial Communications Interface (SCI) with baud rate generator
  - Triple timer module
  - Up to 34 programmable General-Purpose I/O pins (GPIO), depending on which peripherals are enabled
- Reduced power dissipation
  - Very low power CMOS design
  - Wait and Stop low power standby modes
  - Fully-static logic, operation frequency down to 0 Hz (DC)
  - Optimized power management circuitry

TARGET APPLICATIONS

The DSP56303 targets telecommunication applications, such as multi-line voice/data/fax processing, videoconferencing, audio applications, control, and general digital signal processing.

PRODUCT DOCUMENTATION

The three manuals listed in **Table 1** are required for a complete description of the DSP56303 and are necessary to design with the part properly. Documentation is available from a local Motorola distributor, a Motorola semiconductor sales office, a Motorola Literature Distribution Center, or the World Wide Web.

Table 1. DSP56303 Documentation

Document Name	Description of Contents	Order Number
DSP56300 Family Manual	Detailed description of the DSP56300 family architecture and the 24-bit core processor and instruction set	DSP56300FM/AD
DSP56303 User's Manual	Detailed description of DSP56303 memory, peripherals, and interfaces	DSP56303UM/AD
DSP56303 Technical Data	DSP56303 pin and package descriptions, and electrical and timing specifications	DSP56303/D

Mfax and OnCE are trademarks of Motorola, Inc.

How to reach us:

USA/Europe/Locations Not Listed:
Motorola Literature Distribution
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447
1-303-675-2140

Asia/Pacific:
Motorola Semiconductors H.K. Ltd.
8B Tai Ping Industrial Park
51 Ting Kok Road
Tai Po, N.T., Hong Kong
852-26629298

Japan:
Nippon Motorola Ltd
SPD, Strategic Planning Office141
4-32-1, Nishi-Gotanda
Shinagawa-ku, Tokyo, Japan
81-3-5487-8488

Motorola Fax Back System (Mfax™):
RMFAX0@email.sps.mot.com
TOUCHTONE 1-602-244-6609
US & Canada ONLY:
1-800-774-1848
http://sps.motorola.com/mfax/

Technical Resource Center:
1 (800) 521-6274

DSP Helpline
dsphelp@dsp.sps.mot.com

Customer Focus Center:
1-800-521-6274

Internet:
http://www.mot.com/SPS/DSP/

MOTOROLA

Anexo B

Codec CS4218

CRYSTAL
A Cirrus Logic Company

CS4218

16-Bit Stereo Audio Codec

Complete CMOS Stereo Audio Input and Output System featuring:

- Delta-Sigma A/D and D/A Converters using 64x Oversampling.
- Input Anti-Aliasing and Output Smoothing Filters.
- Programmable Input Gain (0 dB to 22.5 dB).
- Programmable Output Attenuation (0 dB to 46.5 dB).
- Sample frequencies from 4 kHz to 50 kHz.
- Low Distortion, THD < 0.02% for DAC. THD < 0.02% for ADC.
- Low Power Dissipation: 80 mA typical.
- Power-Down Mode : 1 mA typical.
- Pin Compatible with CS4216 when used in Serial Modes 3 and 4 (See Appendix A).
- I^2S(TM) Compatible Serial Mode (SM5).
- Operates from 5V or 3.3V Digital Power Supply. Requires 5V Analog Power Supply.

General Description

The CS4218 Stereo Audio Codec is a monolithic CMOS device for computer multimedia, automotive, and portable audio applications. It performs A/D and D/A conversion, filtering, and level setting, creating 4 audio inputs and 2 audio outputs for a digital computer system. The digital interfaces of left and right channels are multiplexed into a single serial data bus with word rates up to 50 kHz per channel.

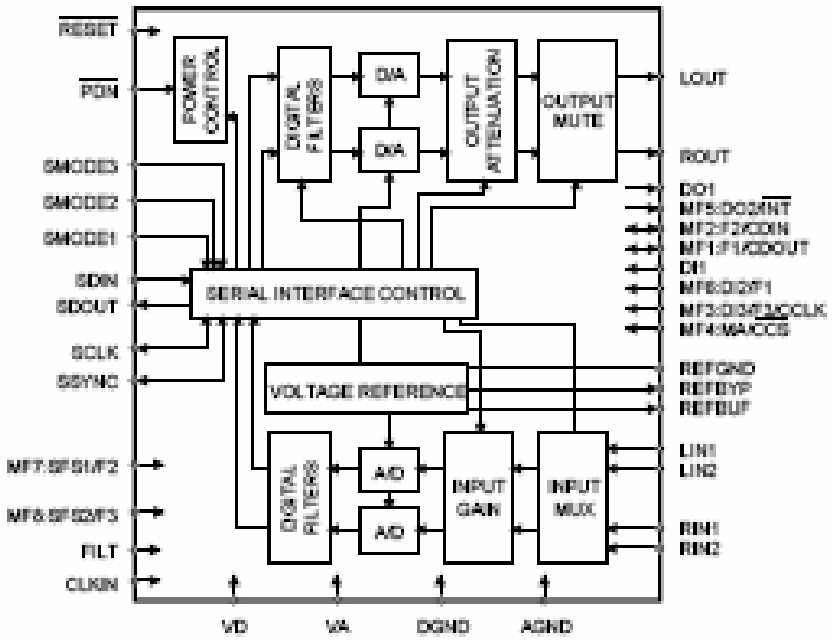
ADCs and the DACs use delta-sigma modulation with 64X oversampling. The ADCs and DACs include digital decimation filters and output smoothing filters on-chip which eliminate the need for external anti-aliasing filters.

The CS4218 is pin and function compatible with the CS4216 when used in Serial modes 3 and 4. See the Appendix A at the end of this data sheet for details.

I^2S is a trademark of Philips.

Ordering Information:

CS4218-KL	0° to 70°C	44-pin PLCC
CS4218-KQ	0° to 70°C	44-pin TQFP



Crystal Semiconductor Corporation
P.O. Box 17847, Austin, TX 78760
(512) 445 7222 FAX: (512) 445 7581
http://www.crystal.com

RECOMMENDED OPERATING CONDITIONS (AGND, DGND = 0V, all voltages with respect to 0V.)

Parameter		Symbol	Min	Typ	Max	Units
Power Supplies:	Digital	VD	4.75	5.0	5.25	V
	Digital (Low Voltage)	VD	3.0	3.3	3.6	V
	Analog	VA	4.75	5.0	5.25	V
Operating Ambient Temperature		T_A	0	25	70	°C

ANALOG CHARACTERISTICS (T_A = 25°C; VA, VD = +5V; Input Levels: Logic 0 = 0V, Logic 1 = VD; 1kHz Input Sine Wave; CLKIN = 12.288 MHz; SM3 Slave sub-mode, 256 BPF; 0dB gain/attenuation;Conversion Rate = 48 kHz; SCLK = 12.288 MHz; Measurement Bandwidth is 10 Hz to 20 kHz; Unless otherwise specified.)

Parameter *		Symbol	Min	Typ	Max	Units
Analog Input Characteristics - Minimum gain setting (0 dB); unless otherwise specified.						
ADC Resolution			16	-	-	Bits
ADC Differential Nonlinearity	(Note 1)		-	-	±0.9	LSB
Instantaneous Dynamic Range	(Note 3)	IDR	80	84	-	dB
Total Harmonic Distortion		THD	-	-	0.02	%
Interchannel Isolation			-	80	-	dB
Interchannel Gain Mismatch			-	-	±0.5	dB
Frequency Response	(Note 1)		-0.5	-	+0.2	dB
Programmable Input Gain			-	22.5	-	dB
Gain Step Size			-	1.5	-	dB
Absolute Gain Step Error			-	-	0.75	dB
Gain Drift	(Note 1)		-	100	-	ppm/°C
Offset Error	0dB Gain		-	-	±50	LSB
	22.5dB Gain		-	–	±500	LSB
Full Scale Input Voltage			2.5	2.8	3.1	Vpp
Input Resistance	(Notes 1,2)		20	-	-	kΩ
Input Capacitance	(Note 1)		-	-	15	pF

Notes: 1. This specification is guaranteed by characterization, not production testing.
2. Input resistance is for the input selected. Non-selected inputs have a very high (>1MΩ) input resistance.
3. Operation in Slave sub-modes may yield results lower than the 80 dB minimum.

* Parameter definitions are given at the end of this data sheet.

Specifications are subject to change without notice.

ANALOG CHARACTERISTICS (Continued)

Parameter *		Symbol	Min	Typ	Max	Units
Analog Output Characteristics - Minimum Attenuation; Unless Otherwise Specified.						
DAC Resolution			16	-	-	Bits
DAC Differential Nonlinearity	(Note 1)		-	-	±0.9	LSB
Total Dynamic Range		TDR	-	93	-	dB
Instantaneous Dynamic Range		IDR	80	83	-	dB
Total Harmonic Distortion	(Note 4)	THD	-	-	0.02	%
Interchannel Isolation	(Note 4)		-	80	-	dB
Interchannel Gain Mismatch			-	-	±0.5	dB
Frequency Response	(Note 1)		-0.5	-	+0.2	dB
Programmable Attenuation	(Note 5)		-	-46.5	-	dB
Attenuation Step Size	(Note 5)		-	1.5	-	dB
Absolute Attenuation Step Error	(Note 5)		-	-	0.75	dB
Gain Drift	(Note 1)		-	100	-	ppm/°C
REFBUF Output Voltage Maximum output current= 400 µA	(Note 6)		1.9	2.1	2.3	V
Offset Voltage	(Note 7)		-	10	-	mV
Full Scale Output Voltage	(Note 4)		2.4	2.7	3.1	Vpp
External Load Impedance			10k	-	-	Ω
Internal Resistor Value for LOUT and ROUT			400	600	800	Ω
Deviation from Linear Phase	(Note 1)		-	-	1	Degree
Out of Band Energy	(22 kHz to 100 kHz)		-	-60	-	dB
Power Supply						
Power Supply Current (Note 8)	Operating (VD = 5.0V) Operating (VD = 3.3V) Power Down		- - -	80 65 -	100 85 1	mA mA mA
Power Supply Rejection	(1 kHz)		-	40	-	dB

Notes: 4. 10 kΩ, 100 pF load.
5. Tested in SM3, Slave sub-mode, 256 BPF.
6. REFBUF load current must be DC. To drive dynamic loads, REFBUF must be buffered. AC variations in REFBUF current may degrade ADC and DAC performance.
7. No DC load.
8. Typical current: VA = 30mA, VD = 50mA with VD = 5.0V. VA = 30mA, VD = 35mA with VD = 3.3V. Power supply current does not include output loading.

* Parameter definitions are given at the end of this data sheet.

SWITCHING CHARACTERISTICS (T_A = 25°C; VA, VD = +5V, outputs loaded with 30 pF; Input Levels: Logic 0 = 0V, Logic 1 = VD)

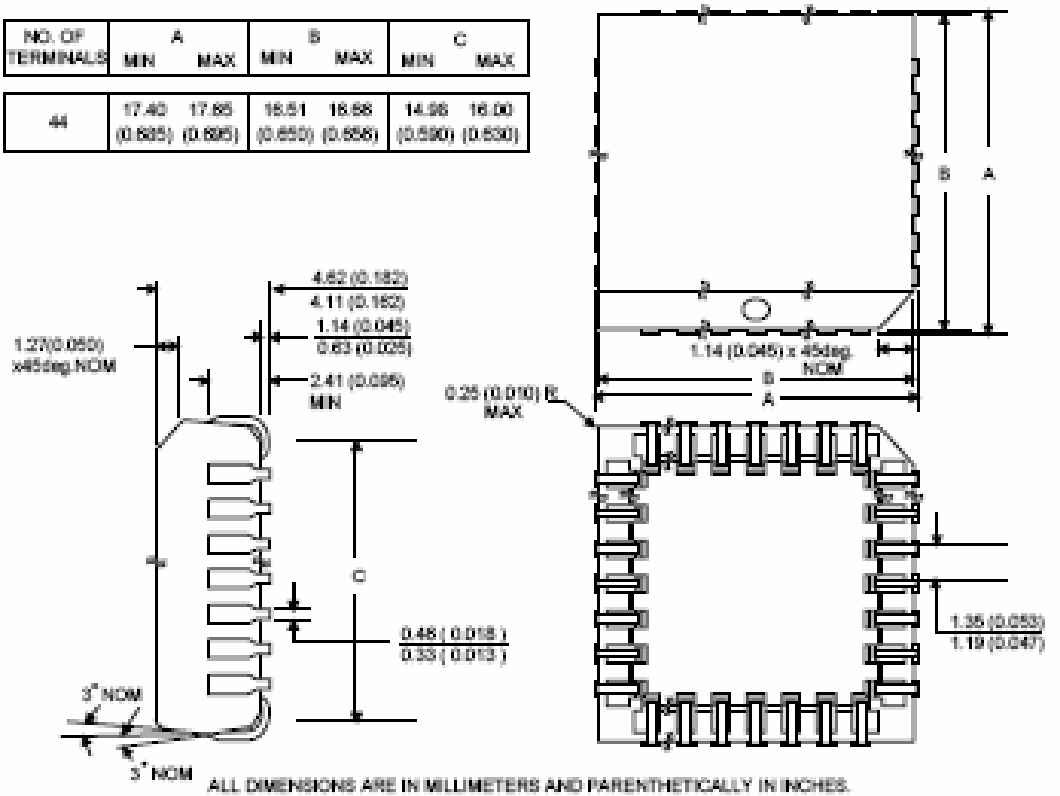
Parameter		Symbol	Min	Typ	Max	Units
Input clock (CLKIN) frequency	SM3 Multiplier Mode	CLKIN	64	768	800	KHz
	SM3 Master and Slave Modes, SM4, SM5	CLKIN	1.024	12.288	12.8	MHz
CLKIN low time		t_{ckl}	15	-	-	ns
CLKIN high time		t_{ckh}	15	-	-	ns
Sample Rate	(Note 1)	Fs	4	-	50	kHz
DI pins setup time to SCLK edge	(Note 1)	t_{s2}	10	-	-	ns
DI pins hold time from SCLK edge	(Note 1)	t_{h2}	8	-	-	ns
DO pins delay from SCLK edge		t_{pd2}	-	-	30	ns
SCLK and SSYNC output delay from CLKIN rising	All master Modes (Note 1)	t_{pd3}	-	-	50	ns
SCLK period	All master Modes (Notes 1,7)	t_{sckw}	-	1/(Fs*bpf)	-	s
	Slave Mode		75	-	-	ns
SCLK high time	Slave Mode	t_{sckh}	30	-	-	ns
SCLK low time	Slave Mode	t_{sckl}	30	-	-	ns
SDIN, SSYNC setup time to SCLK edge	Slave Mode	t_{s1}	15	-	-	ns
SDIN, SSYNC hold time from SCLK edge	Slave Mode	t_{h1}	10	-	-	ns
SDOUT delay from SCLK edge		t_{pd1}	-	-	28	ns
Output to HI-Z state	bit 64 (Note 1)	t_{hz}	-	-	12	ns
Output to non-HI-Z	bit 1 (Note 1)	t_{nz}	15	-	-	ns
RESET pulse width low			500	-	-	ns
CCS low to CCLK rising	SM4 (Note 1)	t_{cslcc}	25	-	-	ns
CDIN setup to CCLK falling	SM4 (Note 1)	t_{dlscc}	15	-	-	ns
CCLK low to CDIN invalid (hold time)	SM4 (Note 1)	t_{ccdlh}	10	-	-	ns
CCLK high time	SM4 (Note 1)	t_{cclhh}	25	-	-	ns
CCLK low time	SM4 (Note 1)	t_{cllhl}	25	-	-	ns
CCLK Period	SM4 (Note 1)	t_{cclkw}	75	-	-	ns
CCLK rising to CDOUT data valid	SM4 (Note 1)	t_{ccdov}	-	-	30	ns
CCLK rising to CDOUT HI-Z	SM4 (Note 1)	t_{ccdot}	-	-	30	ns
CCLK falling to CCS high	SM4 (Note 1)	t_{cccsh}	0	-	-	ns
RESET low time prior to PDN rising		trph	100	-	-	ns
RESET low hold time after PDN rising		trhold	50	-	-	ns

Notes: 7. When the CS4218 is in master modes (SSYNC and SCLK outputs), the SCLK duty cycle is 50%. The equation is based on the selected sample frequency (Fs) and the number of bits per frame (bpf).

PACKAGE DIMENSIONS

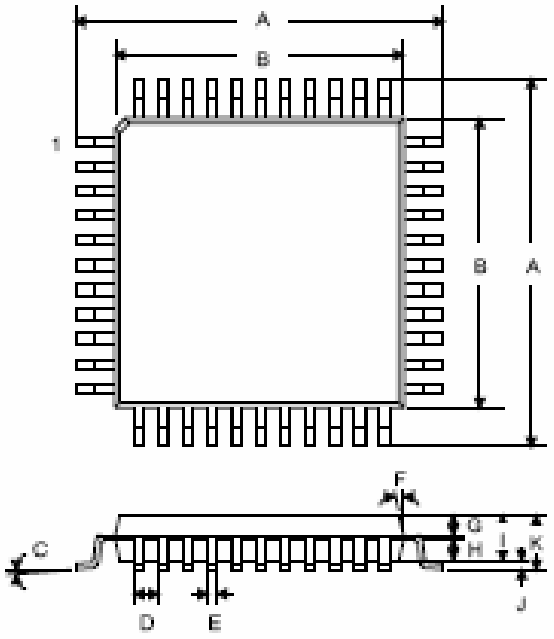
44 PIN PLCC

NO. OF TERMINALS	A MIN	A MAX	B MIN	B MAX	C MIN	C MAX
44	17.40 (0.685)	17.65 (0.695)	16.51 (0.650)	16.66 (0.656)	14.98 (0.590)	16.00 (0.630)

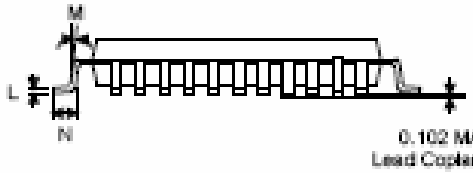


ALL DIMENSIONS ARE IN MILLIMETERS AND PARENTHETICALLY IN INCHES.

44 PIN QUAD FLATPACK



	44 Pin TQFP 1.4 mm Package Thickness			
	MILLIMETERS		INCHES	
DIM	MIN	MAX	MIN	MAX
A	11.75	12.25	0.463	0.482
B	9.90	10.10	0.390	0.398
C	0°	7°	0°	7°
D	0.80 BSC		0.031 BSC	
E	0.35 BSC		0.014 BSC	
F		12o		12°
G	0.54	0.74	0.021	0.029
H	0.54	0.74	0.021	0.029
I	1.35	1.50	0.053	0.059
J	0.05		0.002	
K		1.60		0.063
L		0.17		0.007
M	2°	10°	2°	10°
N	0.35	0.65	0.014	0.026



0.102 MAX
Lead Coplanarity

Anexo C

Memoria Flash AT29LV010

Features

- **Single Supply Voltage, Range 3V to 3.6V**
- **3-volt Only Read and Write Operation**
- **Software Protected Programming**
- **Fast Read Access Time - 120 ns**
- **Low Power Dissipation**
  - **– 15 mA Active Current**
  - **– 40 µA CMOS Standby Current**
- **Sector Program Operation**
  - **– Single Cycle Reprogram (Erase and Program)**
  - **– 1024 Sectors (128 Bytes/Sector)**
  - **– Internal Address and Data Latches for 128 Bytes**
- **Two 8K Bytes Boot Blocks with Lockout**
- **Fast Sector Program Cycle Time – 20 ms**
- **Internal Program Control and Timer**
- **DATA Polling for End of Program Detection**
- **Typical Endurance > 10,000 Cycles**
- **CMOS and TTL Compatible Inputs and Outputs**
- **Commercial and Industrial Temperature Ranges**

1-megabit (128K x 8) 3-volt Only Flash Memory

AT29LV010A

1. Description

The AT29LV010A is a 3-volt only in-system Flash programmable and erasable read only memory (Flash). Its 1 megabit of memory is organized as 131,072 bytes by 8 bits. Manufactured with Atmel's advanced nonvolatile CMOS technology, the device offers access times to 120 ns with power dissipation of just 54 mW over the commercial temperature range. When the device is deselected, the CMOS standby current is less than 40 µA. The device endurance is such that any sector can typically be written to in excess of 10,000 times.

To allow for simple in-system reprogrammability, the AT29LV010A does not require high input voltages for programming. Three-volt-only commands determine the operation of the device. Reading data out of the device is similar to reading from an EPROM. Reprogramming the AT29LV010A is performed on a sector basis; 128 bytes of data are loaded into the device and then simultaneously programmed.

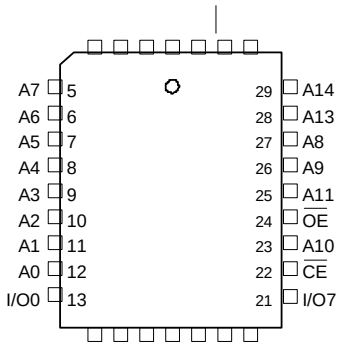
During a reprogram cycle, the address locations and 128 bytes of data are captured at microprocessor speed and internally latched, freeing the address and data bus for other operations. Following the initiation of a program cycle, the device will automatically erase the sector and then program the latched data using an internal control timer. The end of a program cycle can be detected by DATA polling of I/O7. Once the end of a program cycle has been detected, a new access for a read or program can begin.

0520E–FLASH–2/05

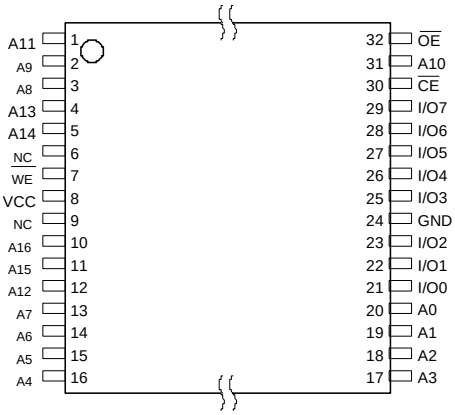
2. Pin Configurations

Pin Name	Function
A0 - A16	Addresses
$\overline{CE}$	Chip Enable
$\overline{OE}$	Output Enable
$\overline{WE}$	Write Enable
I/O0 - I/O7	Data Inputs/Outputs
NC	No Connect

2.1 32-lead PLCC Top View



2.2 32-lead TSOP Top View – Type 1



Anexo D

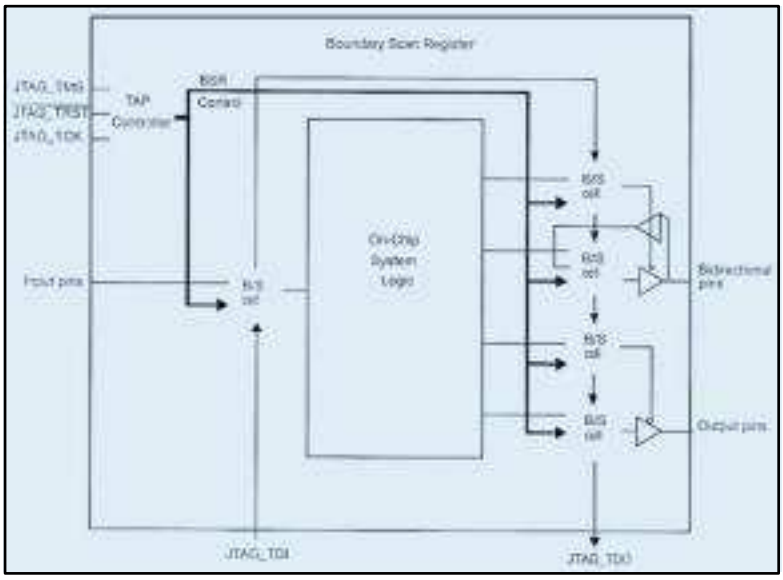
Aplicación del puerto JTAG

Embedded Corner

El uso del bus JTAG para la programación de memorias Flash

Hector Palacios Pérez (Ingeniero de Aplicaciones)

Sistemas Embebidos S.A.
sistemas@embebidos.com



Probablemente hayan oído hablar del bus JTAG que contienen algunos dispositivos electrónicos, pero ¿qué es exactamente y para qué sirve este bus? En el siguiente artículo les explicamos en detalle las líneas que forman el bus JTAG y cómo se puede utilizar para reprogramar memorias Flash EEPROM integradas en un circuito impreso.

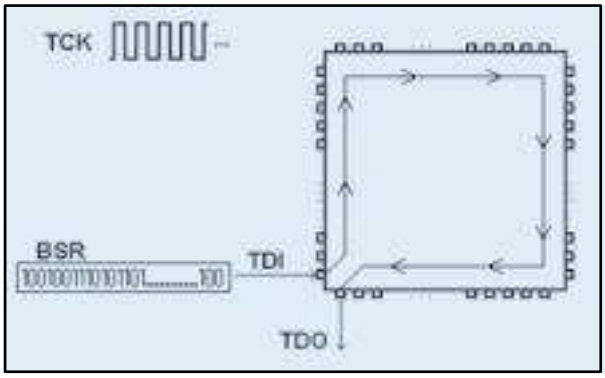
Las siglas JTAG (Joint Test Action Group) recogen el estándar IEEE 1149.1-1990 (Standard Test Access Port and Boundary-Scan Architecture) que se define como: *Electrónica que puede incorporar un circuito integrado para asistir en el test, mantenimiento y soporte de placas de circuito impreso. Esta circuitería incluye un interfaz estándar a través del cual se transfieren datos y comandos con los que el componente puede responder a un conjunto mínimo de instrucciones.*

Esta lógica de test se usa principalmente para comprobar que:

- los componentes funcionan correctamente
- las conexiones entre componentes son correctas
- diferentes componentes interactúan correctamente en un circuito impreso

En sí, es un interfaz serie compuesto de dos líneas de datos (una de entrada, TDI, y otra de salida, TDO) una línea de reloj (TCK) una de reset (TRST) y otra de modo (TMS).

La piedra angular del JTAG es el *Boundary Scan Register* o BSR (ver figura 1), un registro de desplazamiento que contiene tantas celdas como pines tiene el chip, lo que permite, gracias a una máquina de estados llamada *TAP Controller* (Test Access Port), acceder y cambiar el estado de cada uno de los pines del componente.



Por decirlo de forma sencilla, con las cinco líneas del bus JTAG de un microcontrolador podemos poner la combinación que queramos de unos y ceros en las líneas del micro, forzándolo a comportarse como nos convenga.

Si, por ejemplo, queremos escribir el dato 0x12345678 en la dirección 0x00000000 de la memoria RAM, a nivel de líneas del procesador hay que hacer:

- escribir 0x00000000 en el bus de direcciones
- activar el Chip Select conectado a la memoria RAM
- escribir 0x12345678 en el bus de datos
- activar la línea Write Enable

Usando el interfaz JTAG, le pasaríamos al BSR (figura 2) una combinación de cientos de unos y ceros correspondiente a todas y cada una de las líneas del micro, de modo que las líneas de direcciones tuvieran un cero (010100001…). En el segundo ciclo, pasaríamos la misma combinación pero asegurándonos que la línea del CS que va conectada a la memoria RAM, está activada. En un tercer ciclo, se le pasaría otra combinación asegurando que ponemos el dato 0x12345678 en binario en los 32 pines del bus de datos. Y por último asegurando que la línea WE está activa.

Como es obvio, para cambiar de estado un solo pin del micro, es necesario desplazar todo un registro de cientos de bits (tantos como líneas tenga el micro) a golpes de reloj TCK. Debido a ello, el interfaz JTAG es muy lento (una instrucción tan sencilla como escribir un dato en memoria puede llevar miles de ciclos de reloj), pero nos proporciona total control sobre todas las líneas de un dispositivo (generalmente un microprocesador), lo que lo convierte en un interfaz muy potente una vez que el chip está soldado en un circuito impreso.

Programación de memorias Flash EEPROM

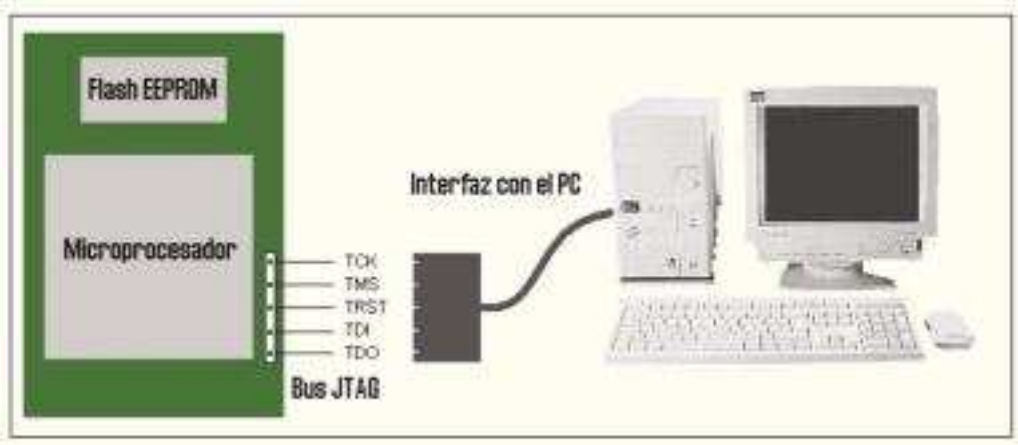
Los sistemas embebidos suelen tener un programa impreso en memoria de sólo lectura ROM, que no puede ser modificado por el usuario, lo que se denomina *firmware*. Esto implica que cuando se actualiza o mejora dicho programa, es necesario cambiar también el hardware del equipo. Para evitarlo, se puede optar por instalar memorias reprogramables eléctricamente, o Flash EEPROM. Ahora bien, para programar estas memorias es necesario extraerlas del circuito y colocarlas en un dispositivo programador. O sea, que necesariamente tenemos que instalar este tipo de memorias en un socket, para poder quitarlas, y disponer de un programador de EEPROMs. Este tipo de programadores lo que hacen es atacar directamente a los pines de la memoria: al bus de datos, al de direcciones, al de control… poniendo en ellos la secuencia correcta que indica el fabricante para programar la Flash.

Pues bien, eso mismo puede hacerse utilizando el bus JTAG de un microprocesador. ¿Qué ganamos con ello? Por una parte, podemos integrar la memoria Flash en el circuito impreso sin necesidad de un socket. Por otro lado, no necesitamos un dispositivo programador de EEPROMs ya que utilizaremos el propio micro del sistema para actualizar el *firmware*.

El interfaz con el PC

(ver figura 3)

Por supuesto, para acceder al bus JTAG, debemos disponer de acceso a sus cinco líneas en nuestro sistema embebido. Y luego crear un interfaz con el PC con el que vamos a controlar los datos que se van a pasar al BSR. Este interfaz puede ser tan sencillo como un simple cable terminado en un conector de puerto paralelo, o tan complicado como queramos diseñarlo (con una máquina de estados paralelo/serie, etc). Creando luego la aplicación adecuada, lo que tendremos finalmente será la posibilidad de generar desde nuestro PC una serie de comandos JTAG con los que variaremos a nuestro gusto todos los pines del microprocesador del sistema embebido y en ellos generaremos la secuencia que indica el fabricante para atacar los pines de la memoria Flash EEPROM y borrarla y reprogramarla como queramos. Todo ello sin necesidad de tocar el circuito impreso original. Gracias a esta técnica, el fabricante del equipo (OEM) puede distribuir actualizaciones del *firmware* y garantizar un mayor tiempo de vida del dispositivo electrónico.



Generalización del interfaz

La observación evidente que cabe hacer acerca de este sistema es que un solo programa no vale para todos los microprocesadores, ya que no todos tienen el mismo número de pines ni estos están dispuestos en el mismo orden. Y tampoco sirve para programar todos los modelos de Flash, ya que las hay de diferentes tamaños y los fabricantes también disponen diferentes algoritmos para programarlas.

Por todo ello, la creación de interfaces de programación vía bus JTAG es en sí una pequeña industria cuyos avances van orientados a la mejora de la rapidez y a la creación de bibliotecas de CPUs y de memorias Flash, de modo que, disponiendo de un solo interfaz con un único software, el usuario pueda programar cientos de tipos de memoria Flash en multitud de dispositivos, cualquiera que sea el microprocesador que lleven. □