Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería Mecánica Eléctrica

TRATAMIENTO DIGITAL DE LA SEÑAL DE AUDIO UTILIZANDO EL DSP SHARCK 21061

Eduardo Roberto Alvarado Erazo
Asesorado por el Ing. Guillermo Antonio Puente Romero

Guatemala, enero de 2016

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

TRATAMIENTO DIGITAL DE LA SEÑAL DE AUDIO UTILIZANDO EL DSP SHARCK 21061

TRABAJO DE GRADUACIÓN

PRESENTADO A LA JUNTA DIRECTIVA DE LA
FACULTAD DE INGENIERÍA
POR

EDUARDO ROBERTO ALVARADO ERAZO
ASESORADO POR EL ING. GUILLERMO ANTONIO PUENTE ROMERO

AL CONFERÍRSELE EL TÍTULO DE

INGENIERO EN ELECTRÓNICA

GUATEMALA, ENERO DE 2016

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

NÓMINA DE JUNTA DIRECTIVA

DECANO	Ing. Pedro Antonio Aguilar Polanco
VOCAL I	Ing. Angel Roberto Sic García
VOCAL II	Ing. Pablo Christian de León Rodríguez
VOCAL III	Inga. Elvia Miriam Ruballos Samayoa
VOCAL IV	Br. Raúl Eduardo Ticún Córdova
VOCAL V	Br. Henry Fernando Duarte García
SECRETARIA	Inga. Lesbia Magalí Herrera López

TRIBUNAL QUE PRACTICÓ EL EXAMEN GENERAL PRIVADO

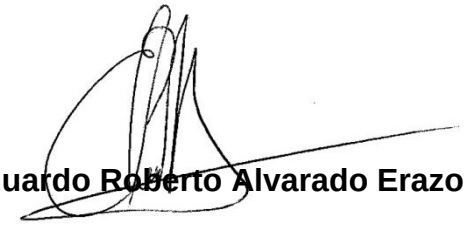
DECANO	Ing. Murphy Olympo Paiz Recinos
EXAMINADOR	Ing. Julio Rolando Barrios Archila
EXAMINADOR	Ing. José Aníbal Silva de los Ángeles
EXAMINADOR	Ing. Julio César Solares Peñate
SECRETARIA	Inga. Marcia Ivónne Véliz Vargas

HONORABLE TRIBUNAL EXAMINADOR

En cumplimiento con los preceptos que establece la ley de la Universidad de San Carlos de Guatemala, presento a su consideración mi trabajo de graduación titulado:

TRATAMIENTO DIGITAL DE LA SEÑAL DE AUDIO UTILIZANDO EL DSP SHARCK 21061

Tema que me fuera asignado por la Dirección de la Escuela de Ingeniería Mecánica Eléctrica, con fecha 7 de febrero 2007.



Eduardo Roberto Alvarado Erazo

Guatemala, 15 de octubre de 2015.

Ing. Carlos Eduardo Guzmán Salazar
Coordinador de Área de Electrónica
Escuela de Ingeniería Mecánica Eléctrica
Facultad de Ingeniería, USAC.

Ingeniero Guzmán:

Por este medio me permito dar aprobación al Trabajo de Graduación titulado: **"TRATAMIENTO DIGITAL DE LA SEÑAL DE AUDIO UTILIZANDO EL DSP SHARCK 21061"**, desarrollado por el estudiante Eduardo Roberto Alvarado Erazo carné No. 2001-80039, ya que considero que cumple con los requisitos establecidos, por lo que el autor y mi persona somos responsables del contenido y conclusiones del mismo.

Sin otro particular, aprovecho la oportunidad para saludarlo.

Atentamente,

Ing. Guillermo Antonio Puente Romero
ASESOR
Colegiado 5898

Guillermo A. Puente R.
INGENIERO ELECTRONICO
COL. # 5898

UNIVERSIDAD DE SAN CARLOS
DE GUATEMALA

FACULTAD DE INGENIERIA

Ref. EIME 68 2015
Guatemala, 23 de OCTUBRE 2015.

Señor Director
Ing. Francisco Javier González López
Escuela de Ingenierìa Mecànica Elèctrica
Facultad de Ingenierìa, USAC.

Señor Director:

Me permito dar aprobación al trabajo de Graduación titulado: TRATAMIENTO DIGITAL DE LA SEÑAL DE AUDIO UTILIZANDO EL DSP SHARCK 21061 del estudiante Eduardo Roberto Alvarado Erazo, que cumple con los requisitos establecidos para tal fin.

Sin otro particular, aprovecho la oportunidad para saludarle.

Atentamente,
ID Y ENSEÑAD A TODOS

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DIRECCION ESCUELA DE INGENIERIA MECANICA ELECTRICA
FACULTAD DE INGENIERIA

Ing. Carlos Eduardo Guzmán Salazar
Coordinador Área Electrónica

sro

UNIVERSIDAD DE SAN CARLOS
DE GUATEMALA

FACULTAD DE INGENIERIA

REF. EIME 68. 2015.

El Director de la Escuela de Ingeniería Mecánica Eléctrica, después de conocer el dictamen del Asesor, con el Visto Bueno del Coordinador de Área, al trabajo de Graduación del estudiante; EDUARDO ROBERTO ALVARADO ERAZO, titulado: TRATAMIENTO DIGITAL DE LA SEÑAL DE AUDIO UTILIZANDO EL DSP SHARCK 21061, procede a la autorización del mismo.

Ing. Francisco Javier González López

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DIRECCION ESCUELA DE INGENIERIA MECANICA ELECTRICA
FACULTAD DE INGENIERIA

GUATEMALA, 11 DE NOVIEMBRE 2,015.

Universidad de San Carlos
de Guatemala

Facultad de Ingeniería
Decanato

DTG. 002.2016

El Decano de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer la aprobación por parte del Director de la Escuela de Ingeniería Mecánica Eléctrica, al Trabajo de Graduación titulado: **TRATAMIENTO DIGITAL DE LA SEÑAL DE AUDIO UTILIZANDO EL DSP SHARCK 21061,** presentado por el estudiante universitario: **Eduardo Roberto Alvarado Erazo,** y después de haber culminado las revisiones previas bajo la responsabilidad de las instancias correspondientes, autoriza la impresión del mismo.

IMPRÍMASE:

Ing. Pedro Antonio Aguilar Polanco
Decano

Guatemala, enero de 2016

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DECANO
FACULTAD DE INGENIERIA

/gdech

ACTO QUE DEDICO A:

Dios	Que siempre ha estado presente en mi vida, guiándome y fortaleciéndome en todo momento, permitiendo que desarrolle las metas que me propongo.
Mis padres	Hilda Erazo y Eduardo Alvarado, que a pesar de estar lejos, siempre me han acompañado, motivado y han sido ejemplo digno de triunfos admirables, gracias a ellos tengo la mejor de las herencias, mi carrera universitaria.
Mi esposa e hijos	María José de Alvarado, José Eduardo y José Roberto Alvarado.
Mis hermanos	Por estar a mi lado, darme su apoyo y confiar en mí.

AGRADECIMIENTOS A:

Universidad de San Carlos de Guatemala	Alma máter, por brindarme la oportunidad de realizar mis estudios en tan gloriosa institución.
Facultad de Ingeniería	Por brindarme todos los conocimientos necesarios para desarrollarme profesionalmente, me enorgullece ser parte de tan prestigiosa Facultad.
Mis amigos	Un agradecido abrazo por su constante recordatorio para terminar lo que empecé.
Ing. Guillermo Puente	Por apoyarme con sus conocimientos y experiencia de cátedra para la realización de este trabajo de graduación.
Ing. Carlos Guzmán	Por el recordatorio constante durante la etapa laboral para culminar este trabajo de graduación.

ÍNDICE GENERAL

ÍNDICE DE ILUSTRACIONES

FIGURAS

TABLAS

LISTA DE SÍMBOLOS

Símbolo	Significado
Θ	Fase, en radianes
Ω	Frecuencia, en radianes por segundo (rad/seg)
F(x)	Función de X
∞	Infinito
Π	Pi
LTI	Sistemas lineales e invariantes con el tiempo

GLOSARIO

ADC	*Analog Digital Converter*
ASP	*Analog Signal Process*
Convolución	Se denomina convolución a una función matemática que, de forma lineal y continua, transforma una señal de entrada en una nueva señal de salida. La función de convolucion se expresa por el símbolo(*).
DAC	*Digital Analog Converter*.
DSP	*Digital signal processor.*
IEEE	Instituto de Ingenieros Electricistas y Eléctrico.
Interconexión	En señales y sistemas, se denomina a la conexión física o lógica entre la salida de un sistema y la entrada de otro sistema independiente. Existen tres interconexiones básicas: en serie, paralelo y mixto (combinación serie-paralelo).
Interpolación	Procesos de reconstrucción de una señal que ha sido muestreada obteniendo de puntos nuevos partiendo de un conjunto discreto de puntos.

Invariancia en el tiempo	Propiedad de los sistemas, la cual se caracteriza por un cambio en el tiempo de la señal de entrada que causa un cambio en la señal de salida. Por ejemplo, un retraso en la señal de entrada, causa un retraso en la señal de salida.
Multiplexacion	Procedimiento por el cual diferentes señales pueden compartir un mismo canal de comunicaciones, haciendo el traslado de información más eficiente que utilizar un canal diferente por cada señal.
Procesador	Conjunto de circuitos electrónicos mediante los cuales se realizan operaciones matemáticas a la información de entrada o señal con una gran velocidad, siguiendo instrucciones dictadas por un lenguaje de bajo nivel obteniendo asi el resultado deseado.
Proceso de filtrado	Proceso de obtención de una o varias características específicas o información de la señal de entrada, mediante un conjunto de elementos de un sistema analógico o digital.
Señal analógica	Se refiere a las magnitudes o valores que varían con el tiempo en forma continua (distancia, temperatura, velocidad, voltaje, frecuencia, amplitud, entre otros) y pueden representarse en forma de ondas.

Señal digital

Cuando amplitud y tiempo se representan con variables discretas en vez de continuas. En el caso de la electrónica, se utilizan ceros y unos (sistema binario). Por ejemplo, las computadoras utilizan dos niveles de voltaje eléctrico para representar distintos valores, el cero o el uno, permitiendo así aplicar lógica y aritmética binaria.

Señales sinusoidales

Señales analógicas, ya que existen valores infinitos entre dos puntos cualesquiera del dominio. Una señal sinusoidal se caracteriza principalmente por cuatro cosas: amplitud, periodo, frecuencia y fase. La amplitud se refiere al máximo voltaje que puede haber en la señal, el periodo se define como el tiempo que tarda en completar un ciclo (medido en segundos), la frecuencia hace referencia al número de veces que se repite un ciclo en un segundo (medido en hertz), y la fase se define como el ángulo de fase inicial (medido en radianes).

RESUMEN

En el presente trabajo de graduación se describirán los conceptos teóricos fundamentales del tratamiento digital de la señal de audio, los cuales son aplicados en el uso del procesador digital de señal 21061 de Analog Devices que se encuentra instalado en un circuito de evaluación llamado EZ-kit ADSP21061. Para llegar a usar de manera óptima el EZ-kit ADSP21061, es necesario conocer a fondo la matemática discreta, así como, conceptos básicos de señales y sistemas.

En el capítulo 1 se presentan los conceptos teóricos del tratamiento digital de la señal, definiendo la señal, tipos de señal y tipos de procesamiento de la señal, así como las ventajas y desventajas de los procesamientos *digital signal procesing* (dsp) contra los *analog signal procesing* (ASP), comparándolos y analizándolos para una mayor comprensión de los mismos. También se definen las señales y sistemas discretos, señales sinusoidales con su representación gráfica para cada una, comandos en Matlab de las secuencias básicas (impulso unitario, escalón unitario discreto y secuencia exponencial).

Luego, se continúa explicando sobre las representaciones matemáticas de los sistemas discretos, como: la función, el vector, las secuencias de los impulsos desplazados y escalados. Siguiendo con el orden lógico de este capítulo, se clasifican los sistemas discretos en: dinámicos/ estáticos, con o sin memoria, sistemas invariantes en el tiempo, sistemas lineales y no lineales, causales y no causales, estables e inestables; explicándose cada uno con sus características y propiedades.

Sobresalen los sistemas lineales e invariantes en el tiempo (LTI), los cuales se caracterizan por su respuesta al impulso y su propiedad de interconexión en serie y en paralelo, representados ambos mediante ecuaciones por diferencia.

Se amplía el tema de la señal mediante un análisis de Fourier para señales de tiempo discreto definiendo la serie discreta de Fourier, la transformada discreta de Fourier, la transformada rápida de Fourier (FFT) y la transformada Z, cuya variable compleja Z es una transformación no lineal de la variable ω de Fourier. Se cierra este capítulo con la explicación sobre el diseño de filtros FIR (*finite impulse response*) e IIR (*infinite impulse response*) para una rápida aplicación sobre la señal de audio.

El capítulo 2 se concentra en el tema de sonido y la conversión de una señal analógica hacia una señal digital mediante los dispositivos *analog to digital converter* (ADC) y *digital to analog converter* (DAC), se explica su funcionamiento, así como sus características y componentes. También se definen conceptos como el punto flotante, el punto fijo y el formato representativo de cada uno de ellos.

Pasando al capítulo 3, se explica la arquitectura de los procesadores DSP SHARCK 21061 y EZ KIT LITE, se detalla cada una de sus partes y su funcionamiento. Finalmente, y con lo aprendido en los capítulos anteriores, en el capítulo 4 se exponen las propuestas de prácticas de laboratorio, elaboradas para la fácil comprensión de los estudiantes y que puedan ser utilizadas como plantillas para el desarrollo de aplicaciones relacionadas con la señal de audio y así comprender de manera rápida y básica el funcionamiento, configuración y aplicación de los DSP de la marca Analog Devices, así como otras marcas con arquitecturas similares.

OBJETIVOS

General

Proponer prácticas de laboratorio para el tratamiento digital de audio utilizando el DSP 21061, para que sirvan de apoyo didáctico al personal docente, auxiliar y estudiantil del curso Comunicaciones 4 y así implementar de una manera práctica y sencilla, las diversas aplicaciones de los procesadores digitales de señal.

Específicos

1. Presentar el tratamiento digital de la señal digital y sus conceptos teóricos.

2. Presentar los conceptos de conversión del sonido de formato analógico a digital y su conversión de señal digital a señal analógica.

3. Presentar las características principales del DSP SHARCK 21061.

4. Presentar las propuestas de laboratorio, con el propósito de facilitar e incentivar a los estudiantes a realizar proyectos relacionados con el tratamiento digital de la señal de audio.

INTRODUCCIÓN

La motivación de este trabajo de graduación surge después de asistir al curso de Comunicaciones 4 de la carrera de Ingeniería Electrónica, el cual se enfoca en tratamiento digital de la señal, enseñando los conceptos de la matemática discreta, pasando por la convolucion, transformadas Z, analisis de Fourier y el diseño de filtros digitales. Al finalizar el curso y conocer la gran variedad de aplicaciones de los procesadores digitales de señal (DSP), nace la necesidad de acompañar la cátedra con ejercicios prácticos realizados en computadora, como el software matemático Matlab y la ejecución de programas en procesadores digitales de señal.

La oportunidad surge en la bodega del Laboratorio de Electrónica de la Escuela Mecánica Eléctrica, en donde existen circuitos de evaluación de DSP de la marca Analog Devices, que es una de las marcas más reconocidas en el campo de los DSP. Lastimosamente se ha tenido el inconveniente de no contar con prácticas diseñadas para los mismos, limitando así su utilización y causando que el estudiante tenga que invertir en la compra de DSP para la realización de sus proyectos de laboratorio. Por medio del coordinador de laboratorios del área de electrónica, se obtuvo el préstamo de un circuito de evaluación del DSP 21061 llamado EZ-KIT ADSP 21061, con el propósito de diseñar prácticas que pudieran complementar de manera práctica la cátedra de Comunicaciones 4.

Durante el diseño de las prácticas se superaron varios retos con el EZ-KIT ADSP 21061, que iniciaron con complicaciones para la instalación según la versión del sistema operativo Windows, la selección y configuración del puerto

correcto de comunicaciones y la carga de programas dedicados para comprobar el funcionamiento del DSP 21061.

Una vez superados estos retos iniciales, se entró en materia con la familiarización con el software de gestión del DSP que permite administrar desde una computadora las locaciones de memoria de programa, configuración y manejo de puertos de entrada y salida. Se define una plantilla de programa que indica las variables que administran la señal de entrada y salida de audio, la frecuencia de muestreo del convertidor Analógico Digital, que es fundamental para la calidad de audio que se requiere en la salida y depende de la frecuencia más alta presente en la señal de entrada de audio y la función que caracteriza al filtro digital a implementar. Finalmente se describen herramientas de apoyo de software que nos ayudan a la visualización de la señal de entrada y salida, así como el diseño de filtros digitales para después exportarlos hacia el EZ-KIT ADSP21061.

En este trabajo de graduación, además, se encuentran las recomendaciones y pormenores que ayudarán al estudiante, auxiliar y catedrático a superar la curva de aprendizaje en el uso del EZ-KIT ADSP 21061 y convertirse en una herramienta de apoyo didáctico para que el estudiante lo utilice en su proyecto final del curso. Uno de los grandes logros es el aprovechamiento de la computadora personal para utilizar el software matemático llamado Matlab como un osciloscopio virtual, o como un generador de señales de audio y como un analizador de espectro, lo que también representa un ahorro para el estudiante, ya que no necesita adquirir un osciloscopio y un generador de señal para trabajar los proyectos en su casa o en la Universidad.

Se espera que este trabajo de graduación también fomente la investigación en el campo del tratamiento digital de la señal de audio por parte del estudiantado, gracias a los retos superados durante la elaboración de estas prácticas de laboratorio para el DSP 21061.

1. TRATAMIENTO DIGITAL DE LA SEÑAL

Para desarrollar el tema del tratamiento o procesamiento digital de la señal es necesario explicar qué es una señal y cada una de sus características, para la obtención de los resultados deseados mediante el procesamiento digital. A continuación se define la señal y sus características más significativas.

1.1. Señal

Las señales son portadoras físicas de información, pueden ser discretas en el tiempo o continuas, sobre una medición en particular, que puede o no ser almacenada para ser procesada e interpretada. Pueden depender de una o más variables independientes.

1.1.1. Tipos de señal

Los tipos de señal están determinados por sus características de origen, formato y dimensión. El origen se refiere a si es analógico o digital, el formato si es unidimensional o multidimensional. A continuación se define cada una de ellas:

- Analógicas

  Una señal analógica o natural puede ser continua o discreta. Es continua cuando su valor en el tiempo es continuo. Es discreta cuando la señal existe únicamente en valores enteros en el tiempo.

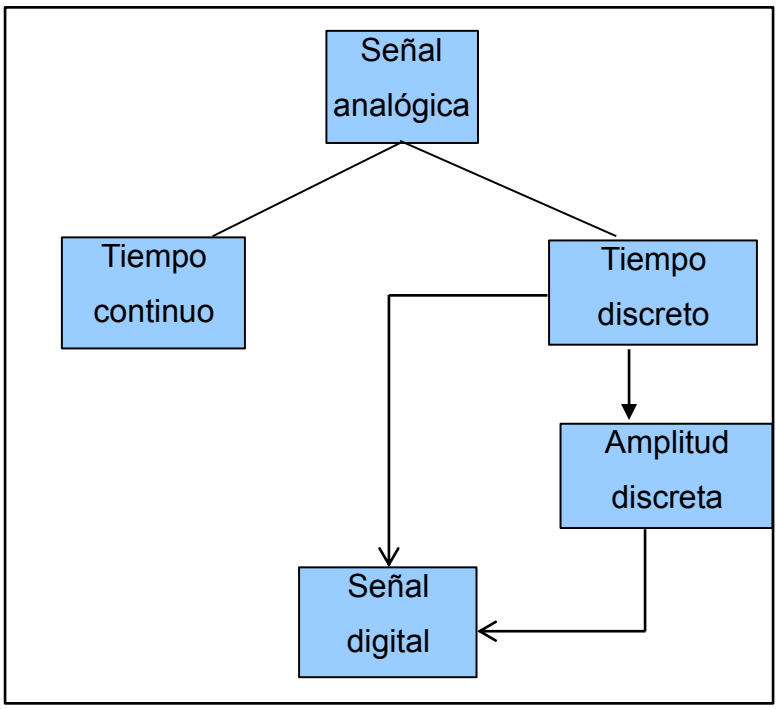
- Digitales

Una señal es digital cuando es discreta en su amplitud y discreta en el tiempo (ver figura 1).

- Unidimensional o multidimensional

Cuando la señal depende de una sola variable es de una dimensión, por ejemplo el audio (amplitud *versus* tiempo). Una señal es multidimensional cuando depende de muchas variables, ejemplo el vídeo depende del color, brillo, pixel *versus* tiempo.

Figura 1. **Tipos de señal**



Fuente: elaboración propia.

1.2. Procesamiento de la señal

Es un proceso que filtra la señal con el fin de obtener la información deseada, por ejemplo: obtener una mejor calidad de la señal eliminando el ruido o frecuencias no deseadas. Hay tres maneras de hacer el procesamiento de la señal: analógica, digital y mixto. A continuación se definen cada una de ellas:

1.2.1. Procesamiento analógico

Conocido también como el *analog signal process* (ASP), es el procesamiento de las señales en el dominio analógico y por medio de circuitos eléctricos con elementos como resistencias, capacitores e inductancias. Asimismo se pueden representar matemáticamente por un conjunto de valores continuos. Hasta los años 70 era la única forma de procesar las señales.

1.2.2. Procesamiento digital

Con la aparición de los circuitos integrados, el procesamiento digital se convirtió en la mejor práctica, ya que es más exacta que el procesamiento analógico (ver tabla I). Los circuitos integrados son menos susceptibles a los cambios de temperatura e interferencias externas, estos se han convertido en un estándar y los costos han bajado debido a una mayor producción y la facilidad de reemplazo.

1.2.3. Procesamiento mixto

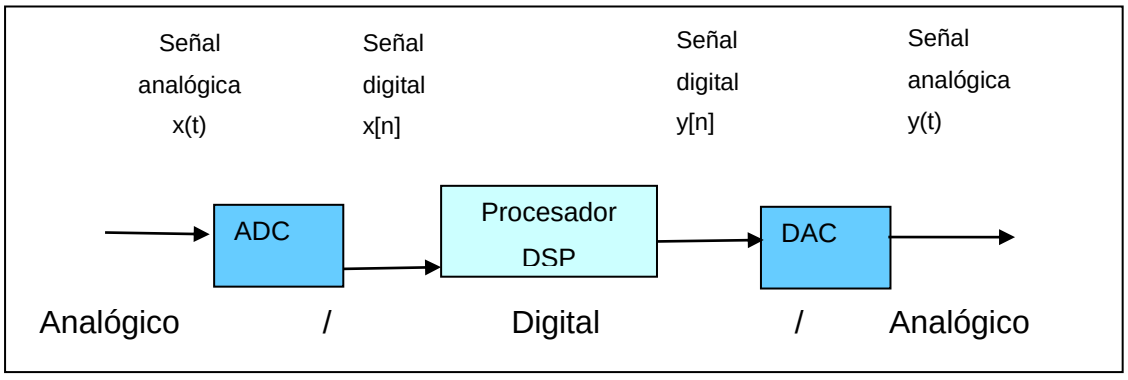
Es una combinación del procesamiento digital y analógico. Este procesamiento toma una entrada digital que es convertida a analógica o viceversa. El ejemplo más claro es el de los transductores digitales, en donde

estos toman una señal analógica, como la temperatura, y la convierten a un valor equivalente en binario. La ventaja del procesamiento mixto es que brinda un enlace entre el mundo analógico y el mundo digital, la desventaja es la necesidad de constante calibración, debido a la vida útil de los transductores y de su vulnerabilidad a interferencias externas por su parte analógica.

1.3. Procesador digital de señal (*digital signal processor*)

Un procesador digital de señal (DSP) es un dispositivo electrónico que realiza cambios en una señal de entrada digital a solicitud de un código de programación, obteniendo una salida deseada. El DSP necesita que la señal de entrada sea convertida a un formato digital por un convertidor de analógico a digital (ADC) para ser procesada. El procesador ejecuta tres funciones básicas: suma, resta y multiplicación. La salida del DSP es digital y necesita ser convertida a una señal análoga por un convertidor de digital a analógico (DAC). Después de eso, esta señal contiene muchas componentes de alta frecuencia, debido al proceso de muestreo, (ver inciso 1.6), para quitarlas se hace pasar por un filtro paso bajo.

Figura 2. **Procesamiento digital de la señal**



Fuente: elaboración propia.

1.4. Comparación del DSP *versus* ASP

Gracias a la arquitectura de los DSP, los cuales están diseñados para el proceso de señales analógicas en tiempo real, se obtienen grandes ventajas del DSP sobre el ASP. La siguiente tabla comparativa enumera las ventajas más significativas.

Tabla I. **Comparación del DPS *versus* ASP**

Procesamiento digital (DSP)	Procesamiento analógico (ASP)
• Menos sensitivo a cambios del medio ambiente y a la tolerancia de los componentes analógicos.	• Elementos como inductancias y capacitores son susceptibles a cambios en sus valores debido al calor y a la vida útil de los componentes.
• Capacidad de producción en grandes cantidades, sin necesidad de ajustar cada circuito.	• Cada circuito analógico debe ser ajustado para tener valores estándar, por lo que es muy difícil de producir en masa.
• En el procesamiento digital basta con modificar una variable del algoritmo para obtener un resultado deseado.	• No es posible obtener valores de inductancias o capacitancias exactas o de valores grandes.
• Se pueden trabajar dos o más señales al mismo tiempo en lo que se llama multiprocesamiento, esto se realiza mediante la multiplexación de señales.	• No se pueden procesar señales analógicas simultáneamente, ellas se deben procesar individualmente.

Fuente: elaboración propia.

1.5. Desventajas del DSP *versus* ASP

También existen desventajas del DSP, entre las cuales se pueden mencionar:

- Se incrementa la complejidad de los componentes en los circuitos digitales, como el muestreador y convertidores ADC y DAC (ver figura 4).
- La frecuencia de muestreo es limitada.
- La resolución de los ADC y DAC es limitada.
- El consumo de potencia de los procesadores es alta por la cantidad de transistores que contienen en su interior.

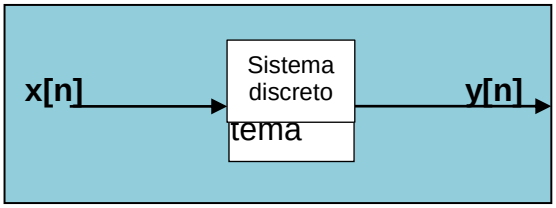
1.6. Sistemas discretos

Se definen como aquello que transforman una señal discreta original en otra final, mediante una conversión analógica digital. La diferencia entre un sistema discreto y uno digital es que en el sistema digital tanto la amplitud como el tiempo son discretos, mientras que en un sistema discreto, la amplitud es continua y el tiempo permanece discreto.

1.6.1. Conversión analógica digital

Las señales discretas y sistemas discretos provienen en la práctica de sistemas continuos, señales que después de aplicarles el muestreo serán discretas en el tiempo.

Figura 3. **Sistema discreto**



Fuente: elaboración propia.

Por ejemplo, sea la señal de audio un tono con frecuencia angular Ωo, también llamada señal sinusoidal.

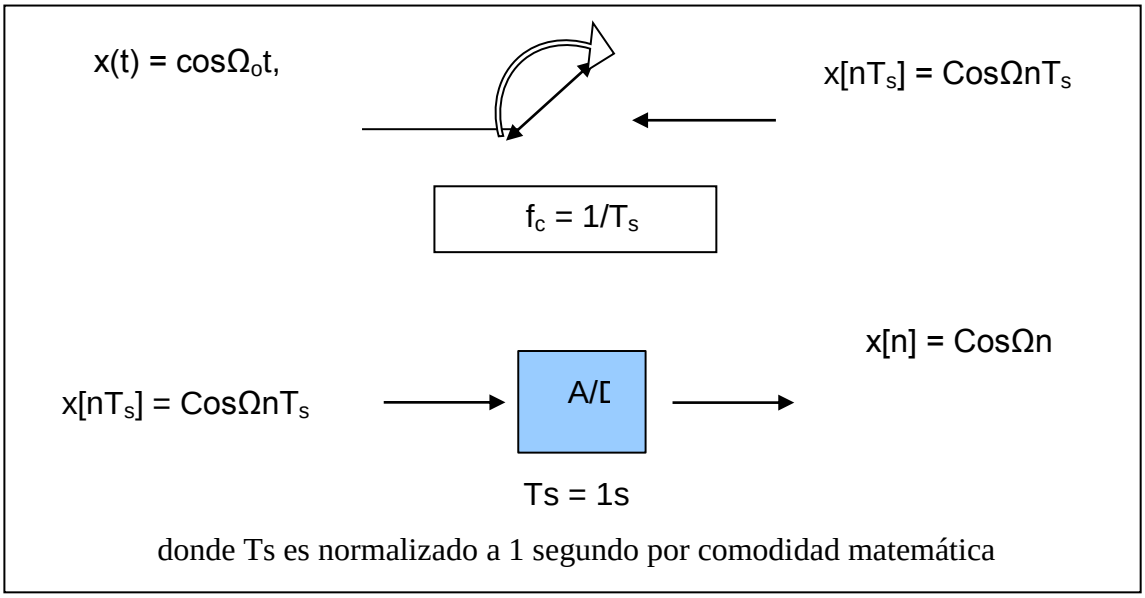
$$x(t) = \cos\Omega_o t, -\infty < t < \infty$$

Donde

T_s: periodo de muestreo

f_c: la frecuencia de muestreo.

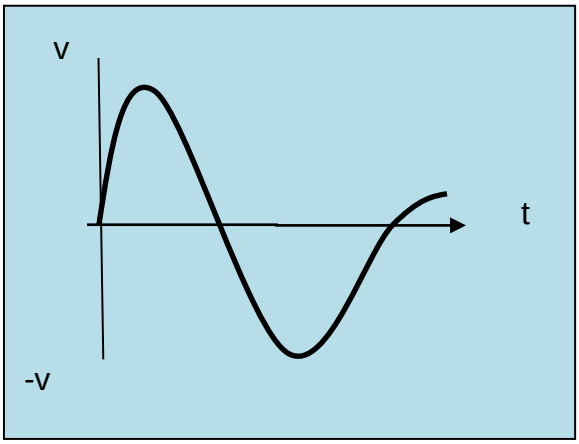
Figura 4. **Muestreo**



Fuente: elaboración propia.

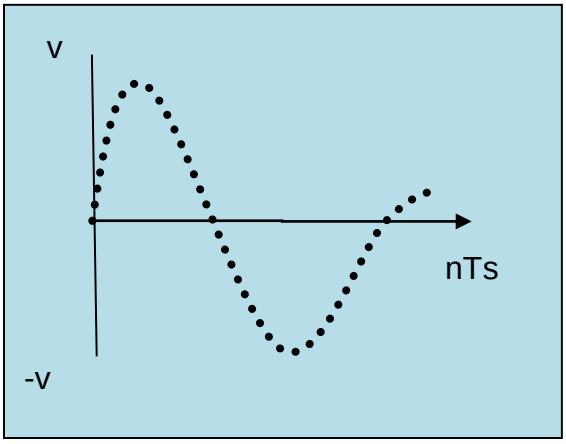
A continuación, la transformación de la señal:

Figura 5. **x(t) señal de audio continua**



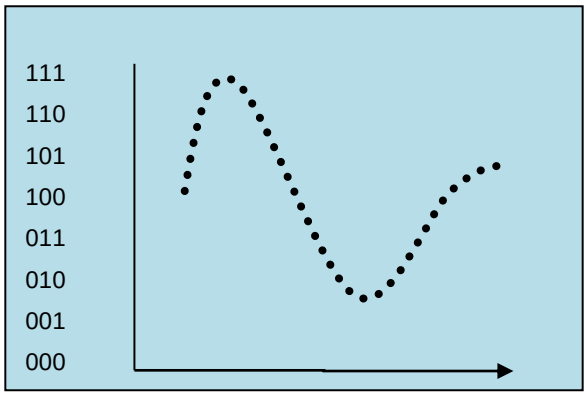
Fuente: elaboración propia.

Figura 6. **x(nTs) señal de audio discreta en el tiempo**



Fuente: elaboración propia.

Figura 7. **x[n] señal digital de audio**



Fuente: elaboración propia.

1.7. Señales sinusoidales

Para visualizar las diferencias de los sistemas analógicos y digitales se analizará una señal sinusoidal en ambos sistemas.

1.7.1. Señales sinusoidales en tiempo continuo

Una simple oscilación armónica se describe matemáticamente mediante la siguiente señal en tiempo continuo.

$$X_a(t) = A\cos(\Omega_o t + \Theta) \quad , -\infty < t < \infty.$$

Esta señal está caracterizada por tres parámetros:

- A*:* amplitud del sinusoide
- Ω_o: frecuencia, en radianes por segundo (rads/s)
- Θ *:* fase, en radianes

A menudo se utiliza (fa) en ciclos por segundo en lugar de

$$\Omega_a = 2\Pi(f_a)$$

$$X_a(t) = A\cos(2\Pi(f_a)t + \Theta)\ ,\ -\infty < t < \infty$$

La señal $X_a(t)$ está caracterizada por las siguientes propiedades:

- Para todo valor fijo de la frecuencia f_a, $X_a(t)$ es periódica

$$X_a(t + 1/fa) = X(t + Tp) = X_a(t)$$

Donde

T_p: periodo fundamental de $X_a(t)$

- Las señales en tiempo continuo con frecuencias diferentes son diferentes.

- El aumento de la frecuencia (f_a) resulta en un aumento en la tasa de oscilación de la señal, en el sentido que se incluyen más periodos en un intervalo de tiempo.

- Se observa, para f = 0 $T_p = \infty$, es consistente con la relación fundamental $f = 1/T_p$

1.7.2. Señales sinusoidales en tiempo discreto

Para representar un sinusoide de forma discreta se sustituye “t” por *nTs,* donde *Ts* representa el periodo de la frecuencia de muestreo. Esto es

equivalente a decir que la señal, al ser muestreada, solo existe en puntos *n* a una distancia equidistante *Ts.*

$$x(t) = A\cos(2\Pi ft + \Theta),\ -\infty < t < \infty \text{ , señal continua}$$

sustituyendo t por nTs

$$x[nTs] = A\cos(2\Pi fnTs + \Theta),\ -\infty < nTs < \infty$$

Como *Ts* es constante en todo el sistema, se puede sustituir por 1, quedando de la siguiente forma:

$$x[n] = A\cos(2\Pi fn + \Theta),\ -\infty < n < \infty$$

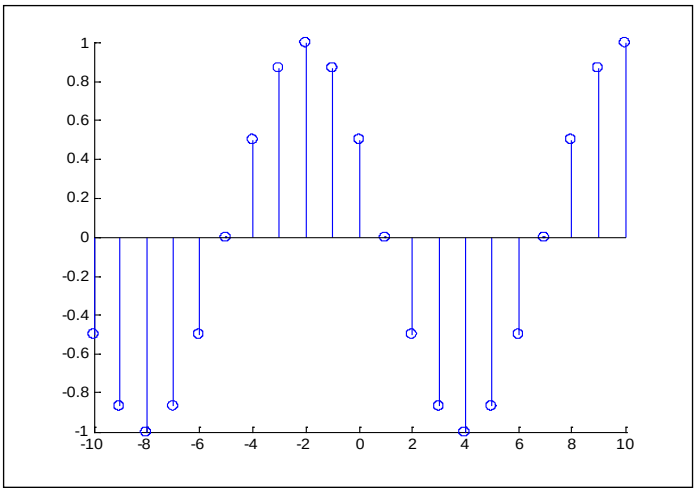
La frecuencia (f) tiene dimensiones de ciclos por muestra. Como se muestra a continuación, en un intervalo de 20 muestras, la señal discreta solo existe en puntos nTs.

Figura 8. **Señal sinusoidal discreta gráficada en Matlab**

si $x[n] = \cos(\Pi/6\, n + \Pi/3)$

comandos de Matlab

```
>> n=-10:10;
>> x=cos(n*pi/6+pi/3);
>> stem(n,x)
```



Fuente: elaboración propia, empleando Matlab.

En contraste con las señales en tiempo continuo, las sinusoides en tiempo discreto están caracterizadas por las propiedades siguientes:

- Por definición, una señal en tiempo discreto x[n] es periódica con periodo N (N>0) si y solo si x[n + N] = x[n] para todo n. El valor más pequeño de N para el que se cumple lo anterior se denomina periodo fundamental.

$$\cos[2\Pi f(N + n) + \Theta] = \cos[2\Pi fn + \Theta],$$

esto es cierto si $2\Pi fN = 2k\Pi$, entonces $f = k/N$.

- Una señal sinusoidal en tiempo discreto es periódica solo si su frecuencia (f) se puede expresar como el cociente de dos enteros. Para determinar el periodo fundamental (N) de una sinusoide periódica se debe expresar su frecuencia como f = k/N y cancelar los factores comunes de k y N hasta que sean números primos relativos, en donde k es un factor común de N. Entonces, el periodo fundamental de la sinusoide es N. Obsérvese que una pequeña variación en la frecuencia puede originar un gran cambio en el periodo. Por ejemplo si f = 31/60 implica N = 60, mientras que si f = 30/60 implica que N =2.

- Los sinusoides en tiempo discreto cuyas frecuencias están separadas por un múltiplo entero de 2Π, son idénticos.

$$\cos[(\Omega + 2\Pi)n + \Theta] = \cos[\Omega n + 2\Pi n + \Theta] = \cos[\Omega n + \Theta]$$

Como resultado todas las secuencias sinusoidales

$X_k[n] = A\cos[\Omega kn + \Theta]$ $\quad k = 1, 2, 3, ...$

Donde

$$\Omega k = \Omega + 2k\Pi \qquad -\Pi <= \Omega <= \Pi , \qquad \text{son idénticas}$$

- Las secuencias de dos sinusoides cualesquiera, de frecuencias en el rango $-\Pi \leq \Omega \leq \Pi$ o $-{}^1/_2 \leq f \leq {}^1/_2$ son distintas. En consecuencia, las señales sinusoidales en tiempo discreto de frecuencia $|\Omega| \leq \Pi$ o $f \leq {}^1/_2$ son únicas, porque son de periodo distinto.

- Cualquier secuencia que resulte de una sinusoide con una frecuencia $|\Omega| \leq \Pi$ o $|f| \leq {}^1/_2$ es idéntica a una secuencia obtenida a partir de una señal sinusoidal de frecuencia $\Omega < \Pi$.

- La mayor tasa de oscilación (*w)* ocurre cuando la frecuencia de muestreo f_s es mínima, por el teorema de Nyquist, la menor frecuencia de muestreo f_s para que la señal pueda ser reconstruida debe de ser mayor o igual que 2f. Sustituyendo en la última ecuación.

$$\Omega max = 2\Pi f/f_s = 2\Pi (f_s/2)/f_s = \Pi$$

$$\Omega = \Pi \text{ (o } \omega = -\Pi\text{) o equivalente a } f = {}^1/_2 \quad (\text{o } f = -{}^1/_2)$$

Entonces el rango de variación de la frecuencia angular discreta es:

$$-\Pi < \Omega < \Pi$$

Para ilustrar esto:

$x[n] = \cos(\omega n), \quad 0 < \Pi$

$\omega = -\Pi/8, \ -\Pi/4, \ -\Pi/2, \ -\Pi, \ 0 \ , \Pi/8, \ \Pi/4, \ \Pi/2, \ \Pi$

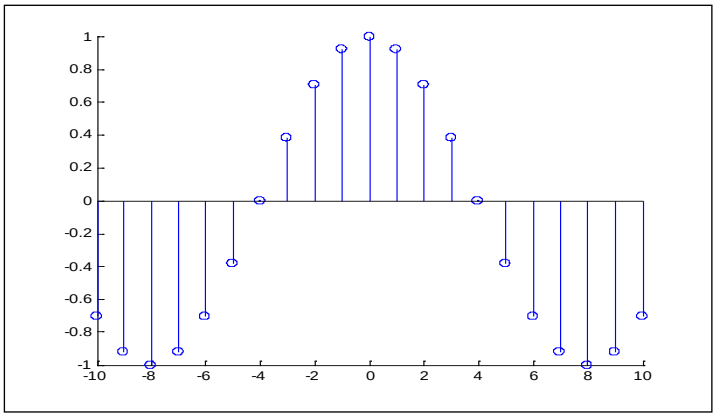
Correspondientes a $f = \omega/2\Pi = 0, 1/16, 1/8, 1/4, ½$; que dan luego a secuencias con periodos.

N = k/f (k = 1) N = ∞, 16, 8, 4, 2

Figura 9. **Comandos Matlab ω = Π/8**

Comandos Matlab

```
>> n= -10:10;
>> x=cos(n*pi/8);
>> stem(n,x);
```

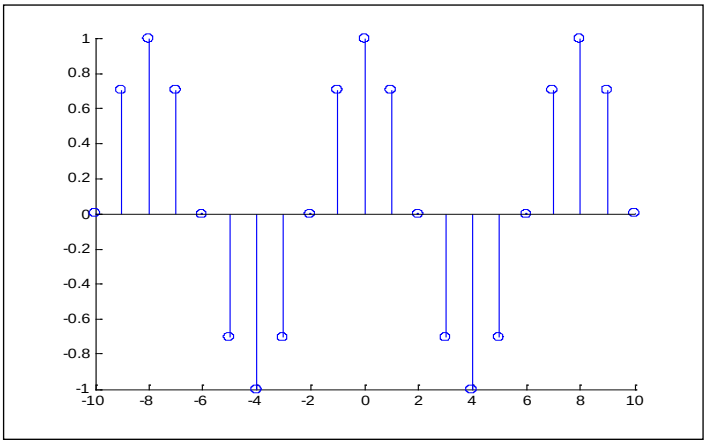


Fuente: elaboración propia, empleando Matlab.

Figura 10. **Comandos Matlab ω = Π/4**

Comandos Matlab

```
>> n= -10:10;
>> x=cos(n*pi/4);
>> stem(n,x);
```

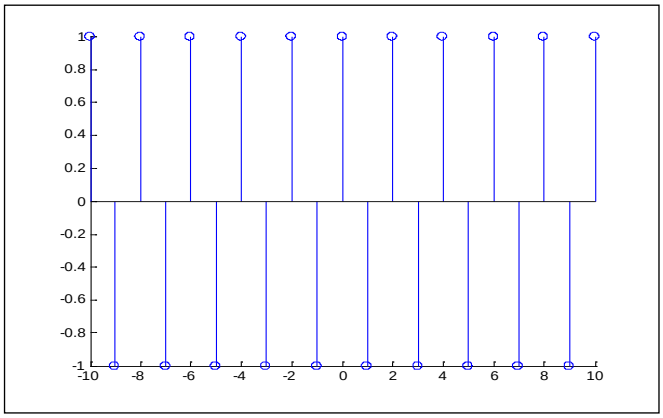


Fuente: elaboración propia, empleando Matlab.

Figura 11. **Comandos Matlab ω = Π/2**

Comandos Matlab

```
>> n= -10:10;
>> x=cos(n*pi/2);
>> stem(n,x);
```

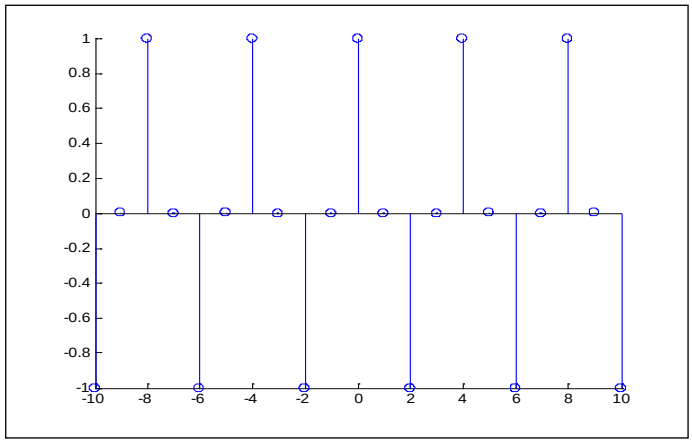


Fuente: elaboración propia, empleando Matlab.

Figura 12. **Comandos Matlab ω = Π**

Comandos Matlab

```
>> n= -10:10;
>> x=cos(n*pi);
>> stem(n,x)
```



Fuente: elaboración propia, empleando Matlab.

1.8. Secuencias básicas

Existen tres tipos de secuencias básicas el impulso unitario, escalón unitario y secuencia exponencial, a continuación se define cada una de ellas.

1.8.1. Impulso unitario

La secuencia más simple por definición es la muestra unitaria o impulso.

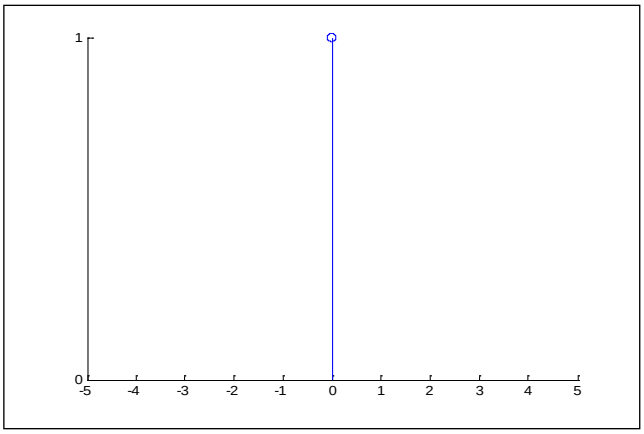
Figura 13. **Impulso unitario**

δ[n] = 0, n ≠ 0

1, n = 0

Comandos Matlab

```
>> n = -5:5;
>> x = [0,0,0,0,0,1,0,0,0,0,0];
>> stem(n,x);
```



Fuente: elaboración propia, empleando Matlab.

1.8.2. Escalón unitario

El escalón unitario discreto (U[n]) se define por una serie impulsos continuos.

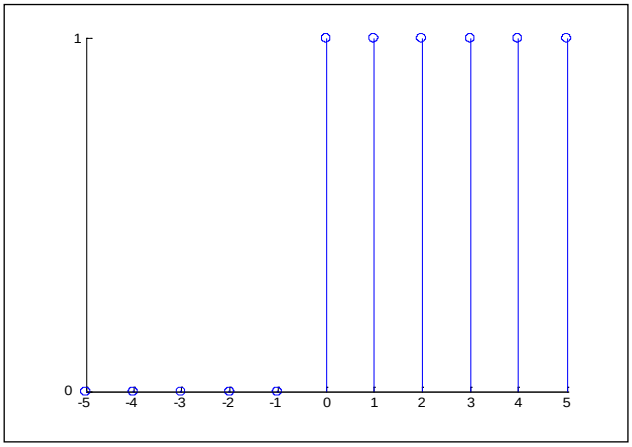
Figura 14. **Escalón unitario**

δ[n] = 0, n < 0

1, n ≥ 0

Comandos Matlab

```
>> n = -5:5;
>> x = [0,0,0,0,0,1,1,1,1,1,1];
>> stem(n,x);
```



Fuente: elaboración propia, empleando Matlab.

Para obtener una representación alternativa de la secuencia escalón en función del impulso unitario, se utliliza

$$\delta[n] \text{ es: } U[n] = \delta[n] + \delta[n-1] + \delta[n-2] + \delta[n-3] + \ldots\ldots$$

También se puede expresar cualquier secuencia en términos de U[n], por ejemplo:

$$\delta[n] = U[n] - U[n-1].$$

1.8.3. Secuencia exponencial

Una secuencia exponencial se puede expresar generalmente como $A\alpha^n U[n]$. La amplitud es limitada cuando $|\alpha| < 1$ si α es mayor que la amplitud tiende a crecer ilimitadamente en función de n.

1.9. Representaciones matemáticas de los sistemas discretos

Las secuencias se pueden representar como funciones, vectores o como secuencias de impulsos desplazados y escalados como a continuación se presenta:

- Función

$$x[n] = \begin{matrix} 1, & 4 > n \geq 1 \\ 0, & n < 1 \end{matrix}$$

- Vector

$$x[n] = (0,1,1,1)$$

- Secuencias de impulsos desplazados y escalados

$$x[n] = \delta[n-1] + \delta[n-2] + \delta[n-2] + \delta[n-3]$$

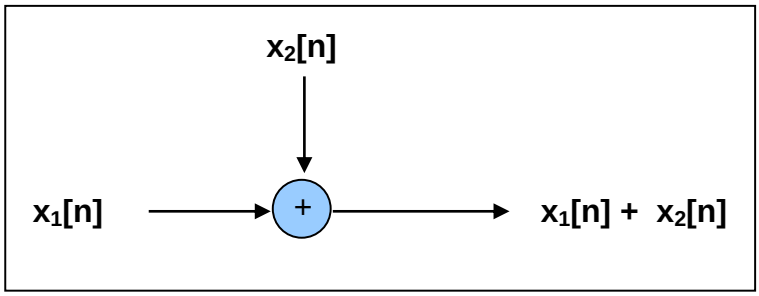
1.10. Representación gráfica de sistemas discretos

La representación gráfica es una manera de visualizar cada uno de los operandos matemáticos involucrados en la ecuación del sistema, ayudando al diseñador a optimizar, ordenar y desarrollar las etapas del sistema.

1.10.1. Suma

La suma de dos o más señales discretas es posible cuando estas se encuentran en un formato similar.

Figura 15. **Suma de señales discretas**

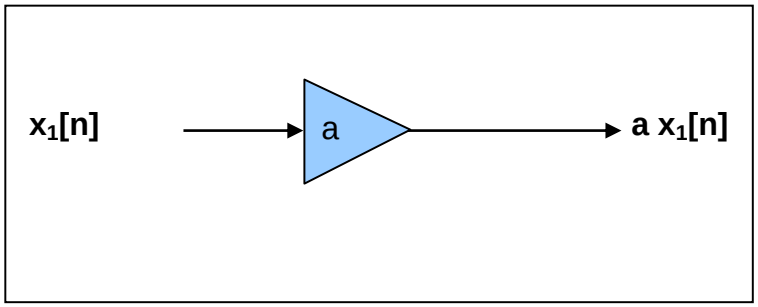


Fuente: elaboración propia.

1.10.2. Escalado

Cuando la señal de entrada es multiplicada por un coeficiente.

Figura 16. **Escalado de una señal discreta**

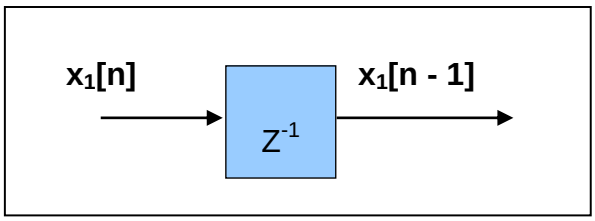


Fuente: elaboración propia.

1.10.3. Retraso en el tiempo

Se produce por un dispositivo de memoria representado por Z^{-1}.

Figura 17. **Retraso en el tiempo de una señal discreta**

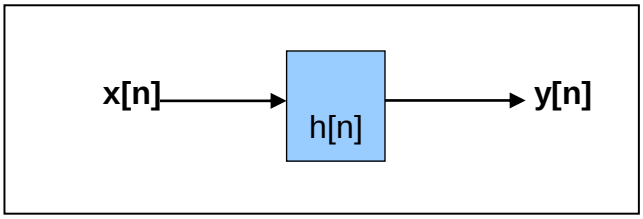


Fuente: elaboración propia.

1.10.4. Sistema

Un sistema puede representarse por su respuesta al impulso.

Figura 18. **Respuesta al impulso del sistema**



Fuente: elaboración propia.

1.11. Clasificación de los sistemas discretos

Los sistemas discretos se clasifican en sistemas con o sin memoria (dinámicos/estáticos), sistemas invariantes en el tiempo, sistemas lineales y no lineales, sistemas causales/no causales y sistemas estables/inestables, a continuación se explica cada uno.

1.11.1. Sistemas con o sin memoria (dinámicos/estáticos)

Un sistema en tiempo discreto se denomina sin memoria si su salida en cualquier instante n depende a lo sumo de la muestra de entrada en ese mismo instante, pero no de las muestras pasadas o futuras de la entrada.

y[n] = A* x[n] // no así y[n] = x[n] + 3x[n-1]
y[n] = n* x[n] + b*x^3*[n] //

1.11.2. Sistemas invariantes en el tiempo

Un sistema se dice invariable en el tiempo si sus características de entrada y salida no cambian con el tiempo.

Si y[n] = T{ x[n]}

Si la entrada se retarda k unidades de tiempo x[n-k] y después se aplica al sistema, si las características del sistema no cambian con el tiempo, la salida del sistema en reposo será y[n-k]. Es decir que la salida es la misma que la correspondiente a x[n], pero esta y[n] estará retardada las mismas k unidades.

x[n] *:* y[n]
x[n-k] *:* y[n-k]

- Ejemplo:

y[n] = x[n]- x[n1]
// *no así* y[n] = n* x[n]
x[n-1] - x[(n-k)-1] = x[n-k] –x[n-1-k]

1.11.3. Sistemas lineales y no lineales

Sistema lineal es aquel que satisface el principio de superposición. El principio de superposición exige que la respuesta del sistema a una suma ponderada de señales sea igual a la correspondiente suma ponderada de las salidas a cada una de las señales de entrada.

$$F\{a_1 x_1[n] + a_2 x_2[n]\} = a_1 F\{x_1[n]\} + a_2 F\{x_2[n]\} = y_1[n] + y_2[n]$$

- Ejemplo

$y[n] = n^* x[n]$

$y_1[n] = n^* x_1[n]$

$y_2[n] = n^* x_2[n]$

$y_3[n] = F\{a_1^* x_1[n] + a_2^* x_2[n]\} = n\{a_1^* x_1[n] + a_2^* x_2[n]\} = a_1^* n^* x_1[n] + a_2^* n^* x_2[n]$

Otra forma:

$a_1^* y_1[n] + a_2^* y_2[n] = a_1^* n^* x_1[n] + a_2^* n^* x_2[n]$ *mismo resultado //*no así $y[n] = x^2[n]$.

1.11.4. Sistemas causales /no causales

Un sistema es Causal si la salida del sistema en cualquier instante n (y[n]) depende solo de entradas pasadas y presentes.

$$(x[n], x[n-1], x[n-2], \ldots\ldots)$$

Pero no de las futuras (es decir, x[n+1],x[n+2],…..)

En un sistema no Causal, la salida depende no solo de las entradas pasadas y presentes sino también de las futuras. Un sistema no causal no puede ser realizado físicamente.

- Ejemplo

Causal $y[n] = x[n] – x[n-1]$

No causal $y[n] = x[n] + 3x[n+4]$

1.11.5. Sistemas estables/inestables

Un sistema estable es aquel que si tiene una entrada acotada, tiene una salida acotada. Por el contrario, si tiene una entrada acotada y la salida es no acotada, es decir infinita, entonces el sistema se clasifica como inestable.

$y[n] = y^2[n-1] + C$ y $[-1] = 0$ es causal

$y[0] = y^2[-1] + C = C$

$y[1] = y^2[1-1] + C = C^2 + C$

$y[2] = y^2[2-1] + C = (C^2)^2 = C^4$

$y[3] = y^2[3-1] + C = ((C^2)^2)^2 = C^8$

1.12. Sistemas lineales e invariantes con el tiempo (LTI)

Una clase de sistemas particularmente importante es la de los sistemas lineales e invariantes con el tiempo. Estos conducen a representaciones especialmente convenientes de los sistemas que la cumplen y tienen importantes aplicaciones en tratamiento de señales. Si la propiedad de linealidad se combina con la representación general de una secuencia, como una combinación lineal de impulsos retrasados, se puede concluir que el sistema lineal queda completamente caracterizado por su respuesta al impulso.

Sea $h_k[n]$ la respuesta del sistema a la entrada δ[n-k], un impulso en n = k

$y[n] = T\{\sum x[k]\, \delta[n-k]\}, -\infty \leq k \leq \infty$

Aplicando el principio de superposición

$y[n] = \sum x[k]\, T\{[n-k]\} = \sum x[k]\, h_k[n]\ , -\infty \leq k \leq \infty$

$h_k[n]$ es la respuesta al impulso, depende de n y k

Si se impone la restricción adicional de invariancia en el tiempo:

$\longrightarrow \quad \delta[n] \qquad h[n]$

$y[n] = \sum x[k]h[n-k]\ , -\infty \leq k \leq \infty$

$\longrightarrow \qquad \delta[n-k] \qquad h[n-k]$

Como consecuencia los sistemas LTI quedan caracterizados por su respuesta al impulso h[n].

y[n] = a la convolución de x[n] con h[n]

y[n] = x[n]*h[n]

- Explicación conceptual:

Entrada → LTI → salida

Si entra un impulso δ[n] → h[n] es la respuesta al impulso

Invariancia en el tiempo δ[n-k] → h[n-k] la salida también se traslada

Sea x[k] es una constante x[k]δ[n-k] → x[k]h[n-k] se multiplica por x[k]

Secuencia de impulsos $\Sigma\, x[k]\delta[n-k] \rightarrow \Sigma\, x[k]h[n-k]\ , \quad -\infty \leq k \leq \infty$

x[n] → y[n]

Entonces $y[n] = \Sigma\, x[k]h[n-k]$ o también $= \Sigma\, x[n-k]h[k]$

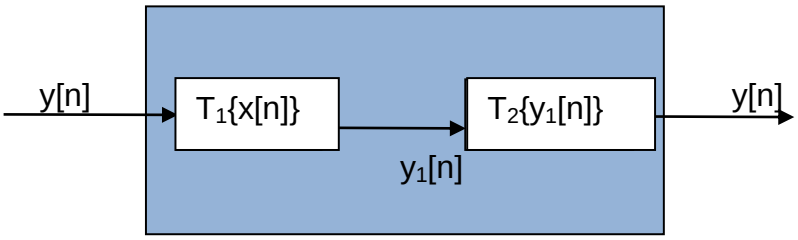
y[n] = x[n]*h[n]

1.12.1. Interconexión de sistemas discretos (LTI)

Los sistemas discretos se pueden interconectar entre sí, a continuación se describe cada uno de estos tipos de interconexión.

- Interconexión en serie: cuando la señal discreta atraviesa dos sistemas distintos en secuencia.

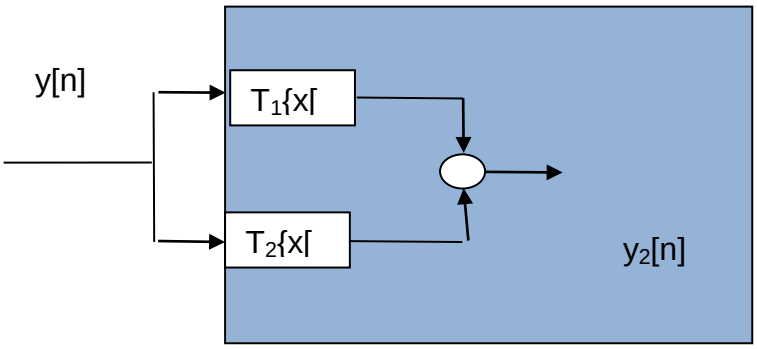
Figura 19. **Sistemas en serie o cáscada**



Fuente: elaboración propia.

- Interconexión en paralelo: cuando la señal discreta atraviesa dos sistemas distintos simultáneamente sumando la salida de cada sistema.

Figura 20. **Sistemas en paralelo**



Fuente: elaboración propia.

1.12.2. Ecuaciones en diferencias con coeficientes constantes

En un sistema LTI se puede representar con sus características:

$$y[n] = \sum x[k]h[n-k] \quad , \quad -\infty \leq k \leq \infty$$

En el cual y[n] depende de las entradas pasadas, presentes y posiblemente futuras de x[n]. Es a esto a lo que se le llama ecuaciones en diferencia. Por otro lado, si el sistema depende de las salidas pasadas se tiene:

$$y[n] = \sum (b_m/a_0)\, x[n-m] - \sum (a_k/a_0)\, y[n-k], \quad 0 \leq m \leq M, \quad 1 \leq k \leq N$$

$$\sum b_m\, x[n-m] = \sum a_k\, y[n-k] \, , \quad 0 \leq m \leq M \quad , \quad 1 \leq k \leq N$$

Si el sistema es no recursivo, entonces:

$$y[n] = \sum (b_m/a_0)\, x[n-m] \quad , \quad 0 \leq m \leq M$$

Si el sistema es recursivo de orden 1, entonces:

$$y[n] = \sum (b_m/a_0)\, x[n-m] - (1/a_0)\, y[n-1] \, , \quad 0 \leq m \leq M$$

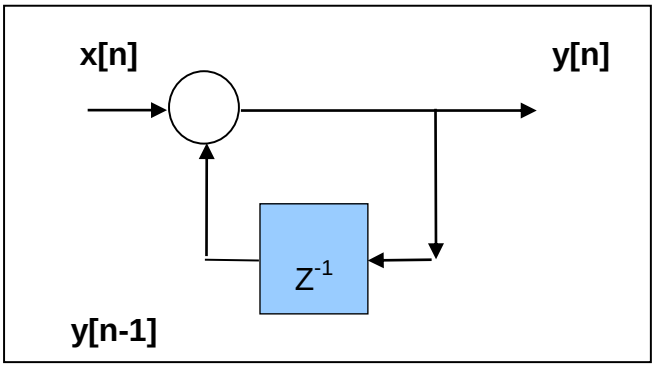
$$y[n] + (1/a_0)\, y[n-1] = \sum (b_m/a_0)\, x[n-m] \, , \quad 0 \leq m \leq M$$

- Ejemplo 1

$$y[n] = x[n] + y[n-1]$$

$$y[n] - y[n-1] = x[n]$$

Figura 21. **Representación gráfica del sistema en diferencia**



Fuente: elaboración propia,

- Ejemplo 2

$y[n] = x[n] - 2x[n-2] + x[n-3]$

Caracterizada por $\sum b_m\, x[n-m] = \sum a_k\, y[n-k]$ $0 \le m \le M$, $1 \le k \le N$ de la forma: $a_0y[n-0] = b_0x[n-0] + b_2x[n-2] + b_3x[n-3]$,

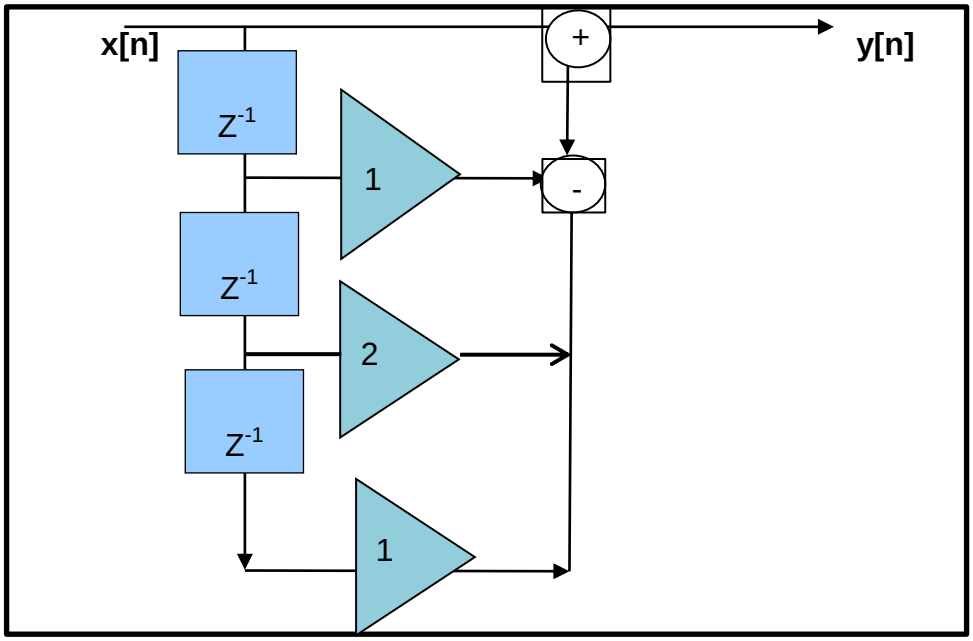
Notar que $b_1 = 0$, $a_0=1$, $b_0=1$, $b_2=2$, $b_3=1$, entonces $k = 0$ y $m = 0, 2, 3$

Por definición $h[n] = b_m / a_0$ cuando $m = 0, 2, 3$, para otro caso = 0.

Entonces:

$$h[n] = (1,0,-2,1) = \delta[n]- 2\delta[n-2] + \delta[n-3]$$

Figura 22. **Representación gráfica del sistema y[n] = x[n] - 2x[n-2] x[n-3]**



Fuente: elaboración propia.

En resumen, hay dos tipos de ecuaciones en diferencias:

- Recursivas: valores pasados de la salida son sumados a la entrada y[n] – ky[n-1]= x[n], k es una constante.

- No recursivas: la salida no depende de valores pasados de la salida y[n] = x[n] + x[n-1] + x[n-2].

En general, los sistemas LTI son expresados como la siguiente ecuación:

$$y[n] + b_1y[n-1] + \ldots\ldots + b_Ny[n-N] = a_0x[n] + a_1x[n-1] + \ldots\ldots + a_mx[n-M]$$

En donde N y M pueden ser diferentes y el orden de la ecuación es igual al número representado por N y M.

1.13. Análisis de Fourier para señales y sistemas en tiempo discreto

El análisis de tiempo continuo de Fourier tiene un gran valor para el análisis de las propiedades de los sistemas y señales. Las fórmulas para el procesamiento de datos digitales con el objetivo de producir aproximaciones numéricas, interpolaciones, integraciones y diferenciaciones nacieron en 1600 por Isaac Newton.

Actualmente se usa para un sinnúmero de cálculos, como predicción de indicadores económicos, análisis de datos demográficos y el uso de datos tomados de la naturaleza con base en la observación de algún fenómeno físico en particular.

1.13.1. Serie discreta de Fourier

Al igual que en el caso continuo, la serie de Fourier tiene como objetivo construir una señal periódica a partir de la sumatoria de sus componentes en frecuencia, pero en el caso de la serie discreta será en el dominio discreto es decir, solo existiendo en valores nTs en donde Ts es el periodo de la frecuencia de muestreo.

Si se tiene una secuencia x[n] con periodo N, se puede representar como una serie que cambiará con cada valor de n hasta N-1.

$$x[0] = \Sigma\, a_k \quad , \quad k = N$$

$$x[1] = \Sigma\, a_k e^{j2\Pi k/N} \quad , \quad k = N$$

$$x[N-1] = \Sigma\, a_k e^{j2\Pi k(N-1)/N} \quad , \quad k = N$$

Se observa que la serie estará determinada por:

$$x[n] = \Sigma\, a_k e^{jk(2\Pi/N)n}$$

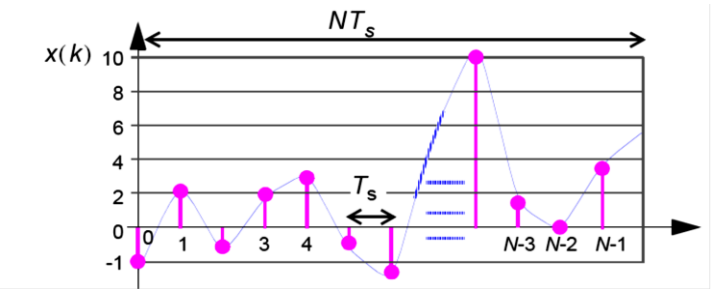
Los coeficientes a_k por la fórmula:

$$a_k = (1/N)\, \Sigma\, x[n]\, e^{-jk(2\Pi/N)n}$$

- Transformada discreta de Fourier (DFT)

La motivación principal de la transformada de Fourier es analizar el espectro de frecuencias que componen una señal, que puede venir de un sensor o transductor, por ejemplo la señal de audio proveniente de un micrófono. Es así que partiendo de la transformada en el tiempo continuo, para el caso de la DFT, se puede decir que la señal es muestreada a intervalos de Ts segundos, convirtiéndola en una señal discreta.

Figura 23. **Transformada discreta de Fourier**



Fuente: *Física de la comunicación*.
http://www.gts.tsc.uvigo.es/ssd/practicas/practica4.pdf. Consulta: junio de 2010.

Entonces, la transformada de Fourier discreta, denominada *discrete Fourier transform* (DFT) será:

$$X(f) = \int_{-\infty}^{\infty} x(kT_s)e^{-j2\pi fkT_s}d(kT_s)$$

Donde k es el índice del tiempo.

Se podrá escribir x (kTs) como una señal discreta (la señal existe solo para valores en intervalos de Ts), x [k] lo que es igual a decir x [n]. Además si t = kTs y fs = 1/Ts, entonces la ecuación para la transformada discreta de Fourier se puede escribir como sigue:

$$X(f) = \sum_{k=-\infty}^{\infty} x(kT_s)e^{-j2\pi fkT_s} = \sum_{k=-\infty}^{\infty} x(k)e^{\frac{-j2\pi fk}{f_s}}$$

Reescribiendo la ecuación para un sistema causal, desde la muestra k = 0, y suponiendo que la señal es periódica con un total de k muestras, o *kTs* segundos, entonces solo se necesita evaluar esta ecuación en frecuencias específicas, las cuales son la frecuencia cero, o frecuencia DC, y a los armónicos (n) de la frecuencia fundamental fo = 1/(KTs) = fs/K y K frecuencias discretas de 0, fo, 2fo, hasta (K-1) fo.

$$X\left(\frac{nf_s}{K}\right) = \sum_{k=0}^{K-1} x(k)e^{\frac{-j2\pi nf_sk}{Kf_s}} \quad \text{for } n = 0 \text{ to } K-1$$

Finalmente, la ecuación de la DFT:

$$X(n) = \sum_{k=0}^{K-1} x(k) e^{\frac{-j2\pi nk}{K}} \quad \text{for } n = 0 \text{ to } K/2$$

1.13.2. Transformada rápida de Fourier (*fast* Fourier *transform* FFT)

Aunque el procesamiento de señales mediante computadores digitales ofrecía tremendas ventajas, aún así el procesado no se podía realizar en tiempo real. Las aportaciones de Cooley y Tukey (1965) de un algoritmo eficiente para el cálculo de las transformadas de Fourier aceleraron el uso del computador digital, este algoritmo se llama FFT. Muchas aplicaciones desarrolladas requerían del análisis espectral de la señal, gracias a la FFT se redujo en varios órdenes de magnitud el tiempo de cómputo. Además, se dieron cuenta de que el nuevo algoritmo se podría implementar en un hardware digital específico, por lo que muchos algoritmos de tratamiento digital de señales que previamente eran impracticables comenzaron a verse como posibles.

Cuanto más y más uso se hace de la computadora mayor es el traslape de aplicaciones con técnicas de tiempo discreto y continuo, la flexibilidad y crecimiento de las capacidades de las computadoras hacen que el uso de sistemas discretos sea más habitual. Muchos de estos sistemas discretos requieren numerosos cálculos de transformadas de Fourier. La investigación surgió con el objetivo de que el cálculo de transformadas de Fourier fueran prácticas y rápidas. Gracias al desarrollo de codificadores de voz, o *vocoders*, analizadores de espectro digitales y otros sistemas digitales surgió el algoritmo llamado *fast* Fourier *transform* o FFT. Este algoritmo probado y diseñado para la implementación digital, reduce el tiempo de cálculo de las transformadas de

Fourier. Básicamente toma un *array* de muestras en el dominio del tiempo y los transforma a un *array* de muestras de frecuencia en el dominio de la frecuencia.

1.14. Transformada Z

Es un modelo matemático que se emplea, entre otras aplicaciones, en el estudio del procesamiento de señales digitales, como el análisis y proyecto de circuitos digitales, los sistemas de radar o telecomunicaciones y especialmente, los sistemas de control de procesos por computadoras. En los sistemas se establece una clasificación:

- Para el caso de tiempo continuo se emplean las transformadas de Laplace o la de Fourier.
- Para el caso de tiempo discreto se emplean las transformadas zeta o la transformada de fourier discreta (basada en la serie de Fourier exponencial).

La transformada Z es importante en el análisis y representación de los sistemas LTI. Partiendo del teorema de convolución, donde X (z);Y (z) son las transformadas de la entrada, salida y respuesta al impulso del sistema, respectivamente, H (z) es una función de transferencia o la función del sistema. Para los sistemas caracterizados por ecuaciones lineales de diferencias con coeficientes constantes, las propiedades de la transformada Z proveen un procedimiento muy conveniente para obtener la función de transferencia, la respuesta en la frecuencia o la respuesta en el dominio del tiempo del sistema.

1.14.1. Transformada Z por definición

La variable compleja Z es una transformación no lineal de la variable ω de Fourier, con el propósito de que la función de transferencia del sistema discreto obtenida de esta transformación sea racional. Dada una secuencia real {xk} se define su transformada en Z como una función compleja:[1]

$$X(z) = Z\{x_k\} = \sum_{k=-\infty}^{\infty} x_k z^{-k} \qquad z \in C$$

Además, es fácil ver que la definición de transformada de Fourier a Z, se consigue haciendo:

$$z = e^{j\omega}$$

Así, por ejemplo, la transformada en Z de una secuencia impulso será:

$$\{\delta_k\} = \{1,0,0,\ldots\} \Rightarrow \Delta(z) = \sum_{k=-\infty}^{\infty} \delta_k z^{-k} = z^0 = 1$$

La transformada de la secuencia en escalón resultará:

$$\{X_k\} = \{1,1,1,\ldots\} \Rightarrow X(z) = \sum_{k=-\infty}^{\infty} x_k z^{-k} = \sum_{k=0}^{\infty} z^{-k} = \frac{1}{1-z^{-1}} = \frac{z}{z-1}$$

Otro ejemplo típico, generalización del anterior, es la secuencia exponencial:

$$\{x_k\} = \{1, a, a^2, a^3, \ldots\} \Rightarrow X(z) = \sum_{k=0}^{\infty} a^k z^{-k} = \sum_{k=0}^{\infty} \left(az^{-1}\right)^k$$

[1] Universidad Politécnica de Madrid. *Introducción al Procesamiento digital de señales.* http://www.elai.upm.es/webantigua/spain/Publicaciones/pub01/intro_procsdig.pdf. p. 19.

Por ser una serie geométrica será convergente, es decir para |z|>|a|, y su suma valdrá:

$$X(z)=\frac{1}{1-az^{-1}}=\frac{z}{z-a}$$

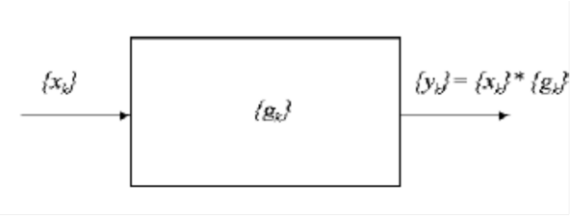
Por último, la transformada del desplazamiento será:

$$Z(x_{k-n})=\sum_{k=-\infty}^{\infty}x_{k-n}z^{-k}=\sum_{k=-\infty}^{\infty}x_{k-n}z^{-(k-n)}z^{-n}=z^{-n}X(z)$$

1.14.2. Función de transferencia en Z

Teniendo en cuenta la relación de la convolución entre la entrada y el sistema y la relación entre la convolución de dos secuencias y sus transformadas, el operador G z), transformada Z de la secuencia de ponderación del sistema, se denomina función de transferencia en Z del sistema considerado a continuación demuestra que la salida es Y (z) = G (z) X (z).

Figura 24. **Transferencia en Z**



Fuente: *Introducción al procesamiento digital de señales*. www.elai.upm.es/webantigua/spain/Publicaciones/pub01/intro_procsdig.pdf. Consulta: junio de 2010.

Esta función de transferencia del sistema puede ser calculada a partir de la ecuación en diferencias que lo determina. Partiendo de la ecuación que define el comportamiento de un sistema dinámico lineal:

$$y_k + a_1 y_{k-1} + a_2 y_{k-2} + \ldots + a_n y_{k-n} = b_0 x_k + b_1 x_{k-1} + \ldots + b_m x_{k-m}$$

Se calcula la transformada Z de las dos partes de la igualdad y teniendo en cuenta las propiedades de linealidad y desplazamiento, se obtiene:

$$Y(z) + a_1 z^{-1} Y(z) + a_2 z^{-2} Y(z) + \ldots + a_n z^{-n} Y(z) = b_0 X(z) + b_1 z^{-1} X(z) + \ldots + b_m z^{-m} X(z)$$

Entonces, la relación entrada/salida del sistema será:

$$Y(z) = \frac{b_0 + b_1 z^{-1} + \ldots + b_m z^{-m}}{1 + a_1 z^{-1} + a_2 z^{-2} + \ldots + a_n z^{-n}} X(z)$$

Por tanto, la función de transferencia del sistema será:

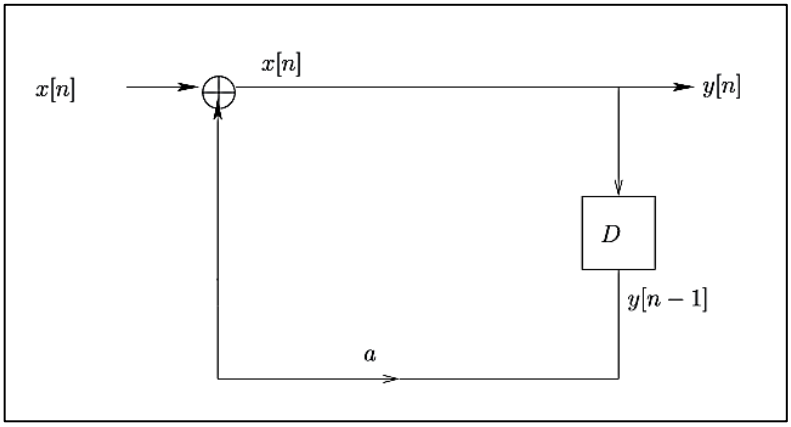
$$G(z) = \frac{b_0 + b_1 z^{-1} + \ldots + b_m z^{-m}}{1 + a_1 z^{-1} + a_2 z^{-2} + \ldots + a_n z^{-n}}$$

- Ejemplo:

Resolver el siguiente circuito con:

y[n] = x[n] + ay[n-1] sea x[n] = u[n]

Figura 25. **Circuito retraso en salida**



Fuente: RONDÓN CARREÑO, Nubia Alexandra. *La transformada Z y algunas aplicaciones.* http://repositorio.uis.edu.co/jspui/bitstream/123456789/7226/2/116771.pdf. Consulta: agosto 2010.

Se despeja x [n] y [n] – ay [n-1] = x[n].

Se aplica la transformada Z en ambos lados de la ecuación y se obtiene:

$$Y(z) - az^{-1}Y(z) = X(z)$$
$$Y(z)[1 - az^{-1}] = \frac{1}{1 - \frac{1}{z}}$$
$$Y(z) = \frac{1}{1 - \frac{1}{z}} \frac{1}{1 - \frac{a}{z}}$$
$$Y(z) = \frac{z^2}{(z-1)(z-a)}$$

Aplicando el teorema de convolución para $a \neq 1$

$$Y(z) = \frac{1}{1-\frac{1}{z}} \frac{1}{1-\frac{a}{z}}$$

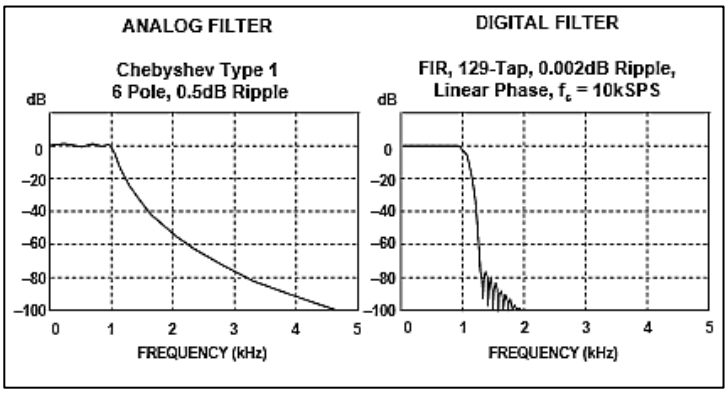
$$y[n] = \sum_{k=-\infty}^{+\infty} a^k u[k] u[n-k]$$

$$y[n] = \sum_{k=0}^{n} a^k u[k] u[n-k] = 1 + a + a^2 + a^3 + a^4 + \cdots + a^{n-1} + a^n.$$

1.15. Diseño filtros digitales FIR e IIR

Una de las aplicaciones más utilizadas del DSP es la emulación de filtros análogos. Un sistema DSP filtrará una señal de entrada digital (selectivamente discrimina señales en diferentes bandas de frecuencia) de acuerdo a algún criterio prediseñado y, en algunas situaciones, los filtros digitales son usados para modificar fase de la señal de entrada. Las características de los filtros digitales se visualizan en una gráfica de la respuesta en frecuencia como se muestra a continuación.

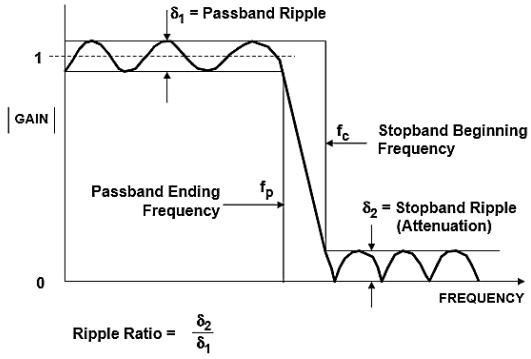
Figura 26. **Filtros análogos *versus* filtros digitales, comparación de su respuesta en frecuencia**



Fuente: *Filtros digital*es. http://www.analog.com/media/en/training-seminars/design-handbooks/MixedSignal_Sect6.pdf. Consulta: agosto 2010.

Para la respuesta en frecuencia, la característica en atenuación o ganancia puede ser especificada en una escala linear de ganancia o logarítmica. los filtros digitales según su función generalmente se clasifican como pasa bajos, pasa altos, paso banda y rechaza banda, por lo general se solicitan las características mostradas en la figura 27.

Figura 27. **Parámetros de diseño de filtros digitales**



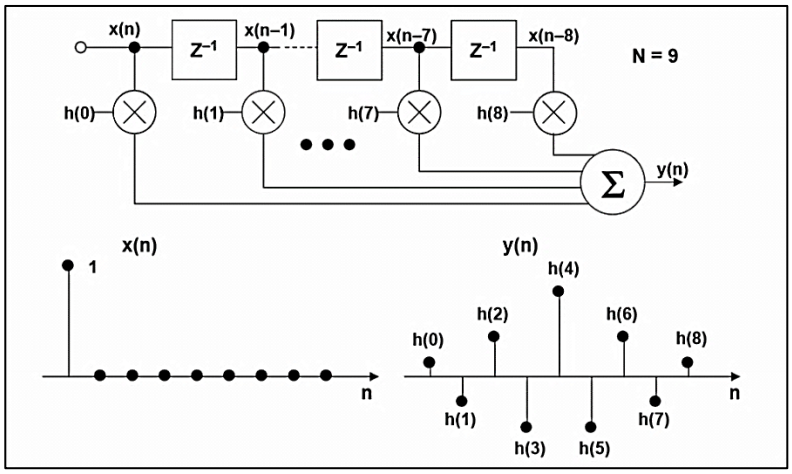
Fuente: *Filtros digital*es. http://www.analog.com/media/en/training-seminars/design-handbooks/MixedSignal_Sect6.pdf. Consulta: agosto 2010.

Existen dos tipos de filtros digitales, FIR (*finite impulse response*) y los IIR (*infinite impulse response*), estas clasificaciones se refieren a la respuesta al impulso. Variando el valor de los coeficientes y el número de *taps* , virtualmente cualquier respuesta en frecuencia puede ser realizada con el filtro FIR, que es capaz de obtener buen rendimiento, lo que no se lograría con un filtro análogo como por ejemplo una perfecta respuesta linear de la fase. Por otro lado, los filtros IIR tienden a imitar el rendimiento de los filtros análogos y hacen uso del *feedback* (muestras de la salida son reinsertadas en la entrada). Los filtros IIR pueden ser implementados con pocos coeficientes a comparación de los FIR.

1.15.1. Respuesta de impulso finito (FIR)

El concepto fundamental del filtro tipo FIR es que la respuesta en frecuencia es determinada por la respuesta al impulso, además esta cuantificación de la respuesta al impulso y los coeficientes del filtro son idénticos. La figura 28 sumariza las características de los filtros FIR, así como la forma más popular de las técnicas de diseño.

Figura 28. **Filtro FIR**



Fuente: *Filtros digital*es. http://www.analog.com/media/en/training-seminars/design-handbooks/MixedSignal_Sect6.pdf. Consulta: agosto 2010.

El flujo de la señal y la ecuación de salida para un filtro FIR, el último N de las muestras de entrada son multiplicadas por los coeficientes de filtro para producir la salida y(n)

$$Y(n) = h(n) * x(n) = \sum_{k=0}^{N-1} h(k)\, x(n-k)$$

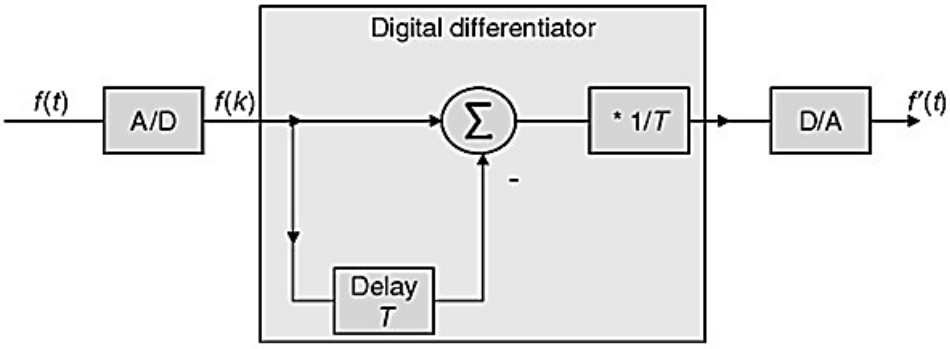
- Diferenciador: un ejemplo sencillo de un filtro FIR es un diferenciador que tiene dos coeficientes, este se implementa usando un elemento de retraso y un elemento de suma para calcular:

$$y(k) = x(k) – x(k-1)$$

Cuando se observa el dominio en frecuencia del diferenciador

$$Y(z) = X(z) - z^{-1}X(z)$$
$$\Rightarrow \quad \frac{Y(z)}{X(z)} = 1 - z^{-1}$$

Figura 29. **Diferenciador**



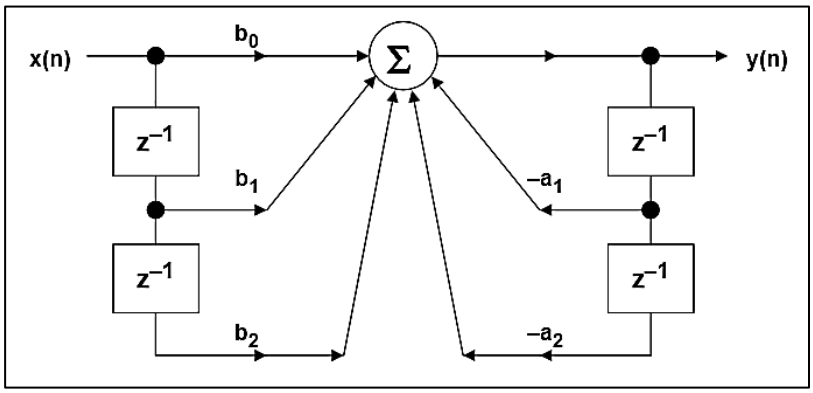
Fuente: *Las herramientas básicas de procesamiento de señales digitales*. http://what-when-how.com/voip-protocols/the-basic-tools-of-digital-signal-processing-voip-protocols/. Consulta: 11 de octubre de 2014.

1.15.2. *Infinite impulse response* (IIR)

Un filtro IIR es un filtro digital que utiliza líneas de retorno (*feedback*) de su salida a la entrada. A diferencia de los filtros FIR, los filtros IIR pueden mostrar inestabilidad por lo que deben ser diseñados cuidadosamente. El nombre de infinito se refiere al hecho de que la salida, debido al impulso, mostrará un

resultado distinto de cero para un tiempo arbitrario. Los filtros IIR tienen su contraparte análoga, por ejemplo los tipo Butterworth, Chebyshev, Elliptic y Bessel, pueden analizarce y sintetizarce usando técnicas de diseño tradicionales. Si el filtro digital es IIR, entonces dos ramas pueden ser definidas: una para los ceros y otra para los polos, por este motivo también son llamados *biquad,* porque son descritos con una ecuación cuadrática en el dominio Z. El filtro IIR *biquad* básico de la figura 30 muestra que los ceros son los coeficientes b0, b1 y b2 ; los polos son formados por los coeficientes en *feedback* a1 y a2.

Figura 30. **Filtro IIR de segundo orden**



Fuente: *Filtros digitales.* http://www.analog.com/media/en/training-seminars/design-handbooks/MixedSignal_Sect6.pdf. Consulta: 11 de octubre de 2014.

$$y(n) = b_0x(n) + b_1x(n-1) + b_2x(n-2) - a_1y(n-1) - a_2y(n-2)$$

Reescribiendo la ecuación en diferencias como sumatoria

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

$$M = 1,\ N = 2$$

La función de transferencia en dominio de z es:

$$H(z) = \frac{\sum_{k=0}^{M} b_k z^{-k}}{1 + \sum_{k=1}^{N} a_k z^{-k}} \quad \begin{matrix}\text{(Zeros)} \\ \text{(Poles)}\end{matrix}$$

- Integrador: un ejemplo sencillo de un filtro IIR es un integrador que es descrito por la ecuación:

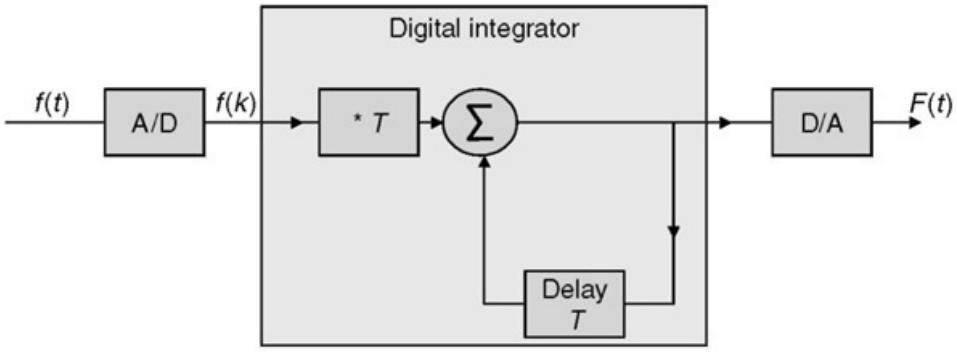
$$y(k) = x(k) + y(k-1)$$

En el dominio Z la función de transferencia del integrador es:

$$Y(z) = X(z) + z^{-1} Y(z)$$
$$\Rightarrow \quad \frac{Y(z)}{X(z)} = \frac{z}{z-1}$$

La respuesta en frecuencia del integrador se concluye que tiene características de un simple filtro paso bajo.

Figura 31. **Integrador**



Fuente: *Las herramientas básicas de procesamiento de señales digitales*. http://what-when-how.com/voip-protocols/the-basic-tools-of-digital-signal-processing-voip-protocols/. Consulta: 11 de octubre de 2014.

2. CONVERSIÓN DE SEÑAL ANALÓGICA A DIGITAL Y DIGITAL A ANALÓGICA

Muchos de los equipos y dispositivos modernos requieren procesar las señales analógicas y convertirlas en señales digitales para funcionar óptimamente. Con el avance de la ciencia y la tecnología, tanto la transmisión como la grabación de los sonidos e imágenes analógicas ha sufrido varios cambios, estos permiten mayores ventajas y más versatilidad en la comunicación. Una gran variedad de los equipos que se conoce como digitales, reciben o captan primero señales analógicas que luego se convierten en señales digitales, utilizando como elemento intermedio un dispositivo denominado conversor analógico digital. En este capítulo se trata sobre el sonido y su conversión analógica a digital y viceversa.

2.1. Sonido

Es la sensación producida en el órgano del oído por el movimiento vibratorio de los cuerpos, transmitido por un medio elástico como el aire. Para describir el sonido se ejemplifican las siguientes características.

2.1.1. Frecuencias audibles

La frecuencia está determinada por la velocidad del cambio de presión en el aire. El rango de frecuencias audibles, según pruebas de laboratorio, se encuentra entre los 20 Hz y 20 Khz sin importar el medio en que se propagan. La frecuencia se percibe como un tono, que es agudo para frecuencias altas y graves para frecuencias bajas.

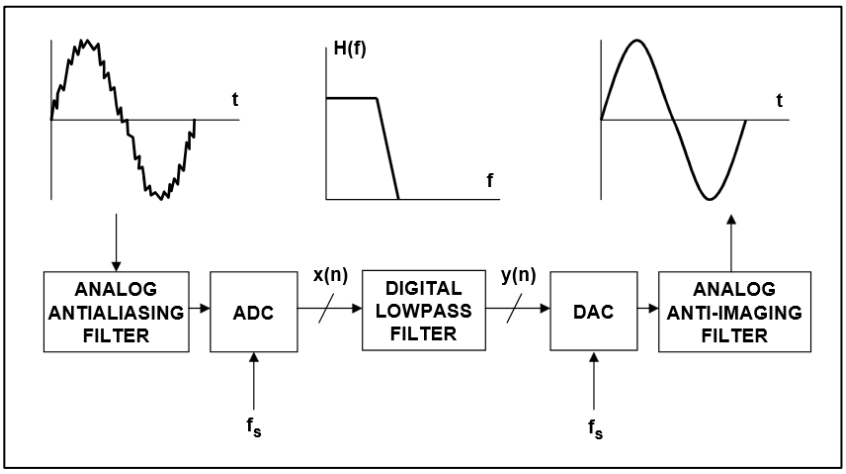
2.1.2. Intensidad

La intensidad está determinada por la amplitud del tono, volumen o sonoridad que es una medición bastante subjetiva, ya que depende de la sensibilidad del oído humano.

2.2. Conversión A/D y D/A

Los dispositivos ADC y DAC son la interfaz con el mundo analógico, como su nombre lo dice ADC (*analog to digital converter*) y DAC (*digital to analog converter*). El ADC tiene la función de convertir la señal de entrada a un lenguaje digital entendible por el DSP, seguidamente, el DSP hace la función de un filtro paso bajo y finalmente el DAC lo convierte nuevamente de un formato digital a un formato analógico, como se muestra en la figura 32.

Figura 32. **Procesamiento digital de la señal de audio**

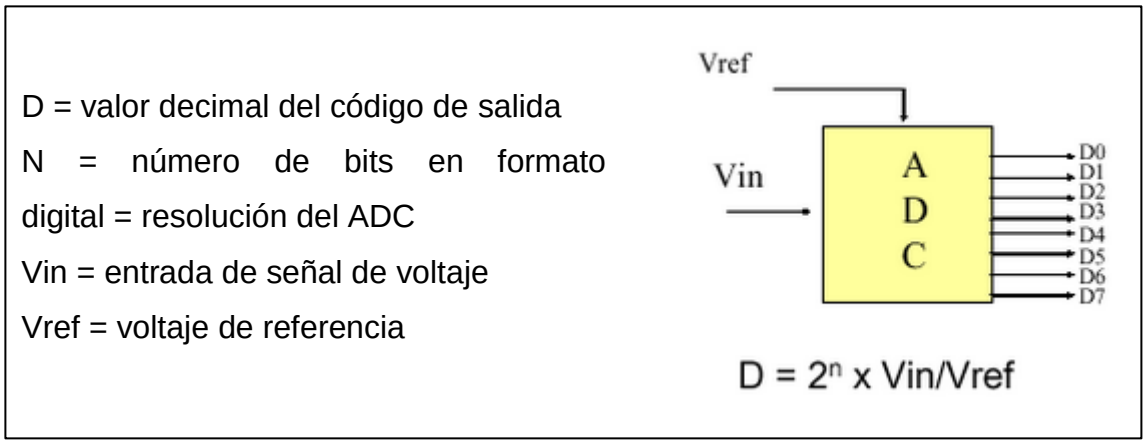


Fuente: *Filtros digitales*. http://www.analog.com/media/en/training-seminars/design-handbooks/MixedSignal_Sect6.pdf. Consulta: 11 de octubre de 2014.

2.2.1. ADC (*analog to digital converter*)

Es un dispositivo que provee una salida que digitalmente representa el voltaje de entrada. La mayoría de los ADC convierten un voltaje de entrada a una palabra digital que cambia de acuerdo a la resolución del dispositivo.

Figura 33. **Diagrama de un ADC básico**

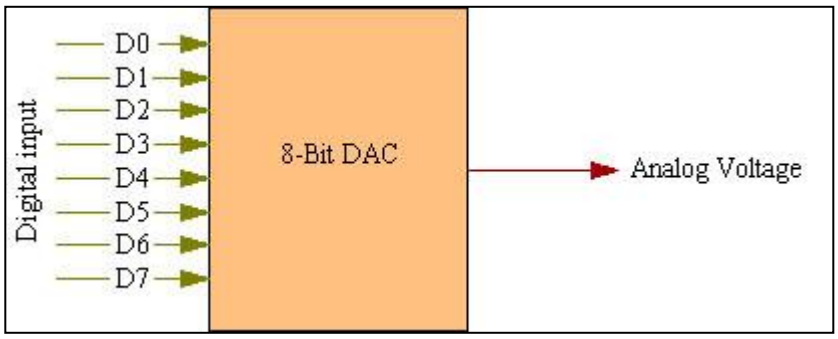


Fuente: *Convertidor analógico a digital (ADC).* http://www.electronics.dit.ie/staff/tscarff/DT089_Physical_Computing_1/adc/adc.htm. Consulta: 11 de octubre de 2014.

2.2.2. DAC (*digital to analog converter*)

Consta de un muestreador o *sampler* que convierte una señal continua a una señal discreta, un cuantizador que le asigna un valor digital a cada muestra de voltaje.

Figura 34. **Diagrama básico de un DAC**



Fuente: *8-bit digital a analógico (DAC).* http://www.ikalogic.com/8-bit-digital-to-analog-converter-dac/. Consulta: 11 de octubre de 2014.

2.3. Frecuencia de muestreo

El tiempo de muestreo es crucial para la calidad de la señal de salida, para determinarlo se debe conocer las propiedades en frecuencia de la señal de entrada. Debe haber un balance a la hora de determinar la frecuencia de muestreo del sistema, del teorema de Nyquist se sabe que la frecuencia mínima de muestreo debe ser mayor o igual a dos veces la frecuencia máxima de la señal a muestrear, Fs >= 2fm, por lo tanto el rango de frecuencia de muestreo va desde 2fm hasta fs*max* del codificador o *codec* (llámese *codec* al conjunto de circuitos que contienen al ADC y DAC).

- Rango de la frecuencia de muestreo 2fm < fs < fc*max.*

- Experimental y teóricamente se ha determinado que cuando fs es igual a 2fm, la calidad de la señal reconstruida es muy pobre y que cuanto la señal fs es igual a la fs máxima permitida por el *codec.* Exige mayor capacidad de procesamiento del DSP, reduce la vida útil del *codec* y del

DSP, genera más ruido se producen más errores de cuantización. Por lo tanto, no se debe sobredimensionar la fs del sistema, sino buscar un valor intermedio que no reduzca la vida útil del DSP.

2.4. Formatos de operación punto flotante y punto fijo

La respuesta de un sistema depende en gran medida de la elección de algoritmos y de las arquitecturas de sistema que se dividen en dos grandes grupos:

- Arquitectura de punto fijo
- Arquitectura de punto flotante

2.4.1. Arquitectura de punto fijo

Fue introducida a comienzos de la década de los 80 y está basada en una representación que contiene una cantidad fija de dígitos después del punto decimal. La mayoría de los DSP de bajo costo utilizan esta arquitectura, aunque en determinados casos esta alternativa ofrece también mejor desempeño o mayor exactitud.

Los bits a la izquierda del punto decimal se denominan bits de magnitud y representan valores enteros, en cambio, los bits a la derecha representan valores fraccionales o potencias inversas de 2. Es decir que el primer bit fraccional es ½, el segundo es ¼ y el tercero es 1/8, entre otros.[2]

La representación en punto fijo está dada de la siguiente manera:

[2] VisualDSP++ 4.0 C/C++ Compiler and Library. *Manual for SHARC Processors*. p. 29.

- Para números positivos

$$2^{m-1} - \frac{1}{2^f}$$

- Para números negativos

$$-2^{m-1}$$

Donde m son los bits de magnitud y f son los bits fraccionales.

Por ejemplo, si se dispone de 16 bits en total y se utilizan 11 bits para representar los enteros (m) y los restantes 5 bits para los fraccionales (f), se encuentra que el máximo número positivo representable es 1023,96875. En cambio, así se asigna m = 12 y f = 4, el mayor número positivo que podremos representar es 2047,9375, si m=13 y f=3, este número resulta ser 4095,875. En conclusión, la arquitectura punto fijo permite representar magnitudes mayores solo con reducir la precisión del punto decimal.

2.4.2. Arquitectura de punto flotante

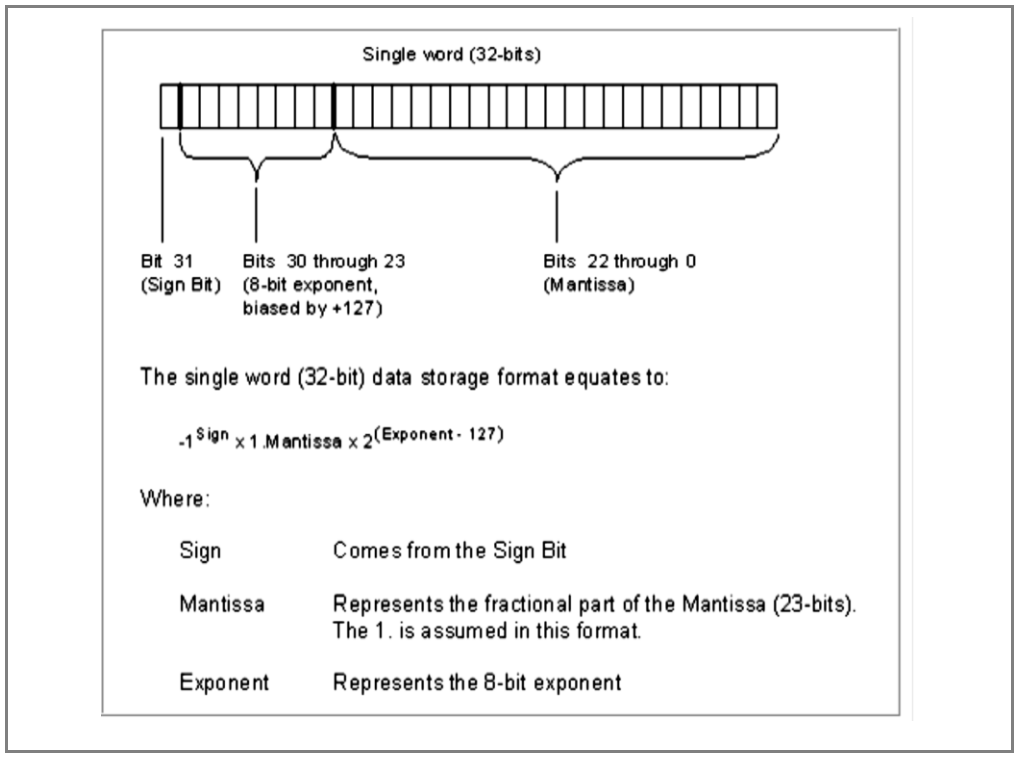
Es más moderna y resulta suficientemente exacta y rápida para la mayoría de aplicaciones. Se logra una muy buena aproximación al número que se desea representar pero muchas veces requiere un redondeo a su limitada precisión. La representación requiere de un número entero llamado mantisa, multiplicado por una base que es siempre 2, elevado a un exponente.[3] Cualquier número de punto flotante puede ser representado como:

[3] VisualDSP++ 4.0 C/C++ Compiler and Library. *Manual for SHARC Processors*. p. 35.

$$a = m \times b^e$$

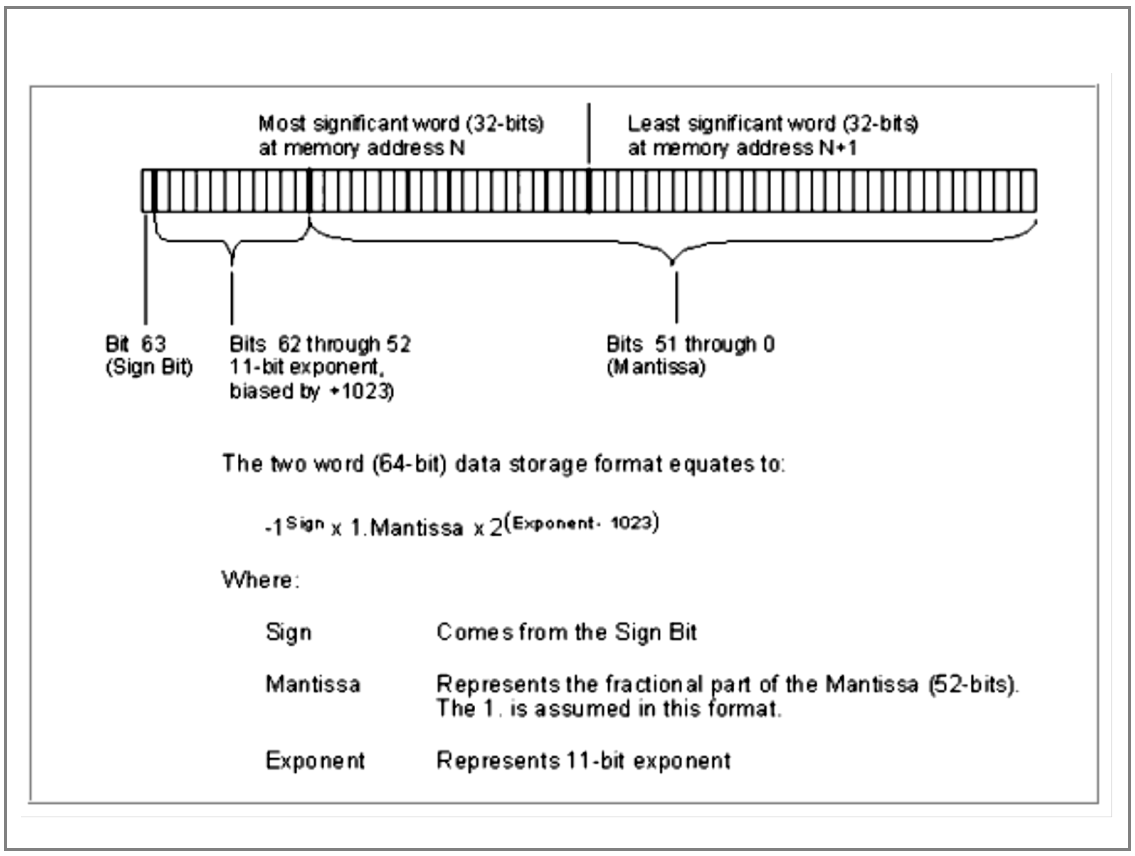
Esta arquitectura tiene una gran ventaja y es que permite la representación de un rango de magnitudes mucho más amplio que en la arquitectura de punto fijo. De acuerdo a la cantidad de bits utilizados para almacenar un número de punto flotante, se dice que es de precisión normal cuando tiene 32 bits y de precisión doble cuando es de 64 bits. En el caso de la precisión normal se le asignan a la mantisa los 23 bits menos significativos (0 al 22), el exponente ocupa los siguientes 8 bits (23 al 30), el bit 31 está dedicado al signo (0 = positivo, 1 = negativo).

Figura 35. **Operación punto flotante (32 bit IEEE precisión única)**



Fuente: *Manual for SHARCK Processors.* http://www.electronics.dit.ie/staff/tscarff/DT089_Physical_Computing_1/adc/adc.htm. Consulta: 11 de octubre de 2014.

Figura 36. **Operación punto flotante (64 bit IEEE doble precisión)**



Fuente: *Manual for SHARCK Processors.* http://www.electronics.dit.ie/staff/tscarff/DT089_Physical_Computing_1/adc/adc.htm. Consulta: 11 de octubre de 2014.

2.5. Formatos punto flotante en lenguaje de programación C++

El lenguaje de programación de alto nivel posee términos que equivalen a formatos de números punto flotante. Es importante que la selección de los acrónimos corresponda al formato utilizado por el DSP, ya que la precisión y exactitud de las entradas y salidas al DSP pueden no ser compatibles.

Tabla II. **Formatos y tamaños utilizados en C++ para ADSP -21xxx Processors**

Type	Bit Size	Result of sizeof operator
int	32 bits signed	1
unsigned int	32 bits unsigned	1
long	32 bits signed	1
unsigned long	32 bits unsigned	1
char	32 bits signed	1
unsigned char	32 bits unsigned	1
short	32 bits signed	1
unsigned short	32 bits unsigned	1
pointer	32 bits	1
float	32 bits float	1
fract	32 bits fixed-point	1
double	either 32 or 64 bits float (default 32)	either 1 or 2 (default 1)
long double	64 bits float	2

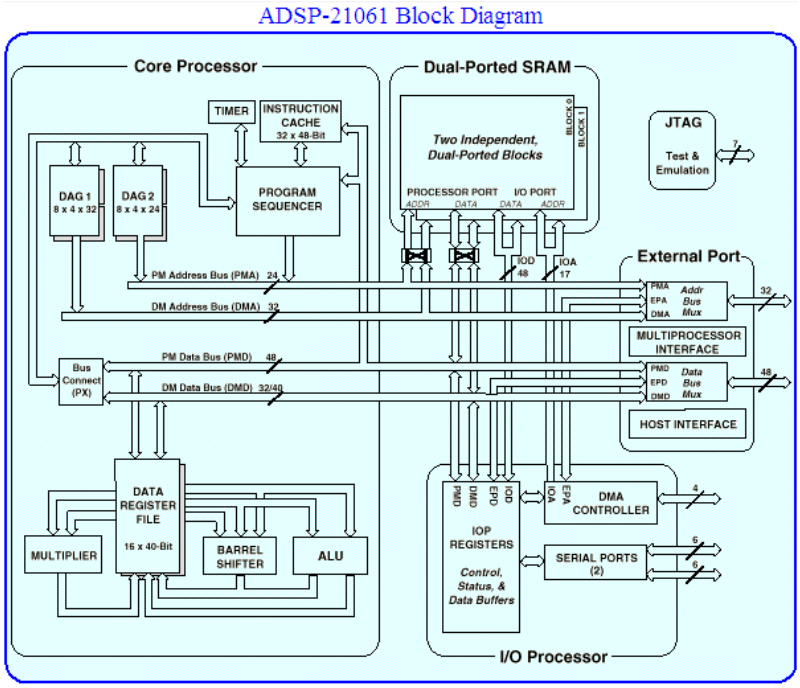
Fuente: *Manual for SHARCK Processors.* http://www.electronics.dit.ie/staff/tscarff/DT089_Physical_Computing_1/adc/adc.htm. Consulta: 11 de octubre de 2014.

3. DSP SHARCK 21061 Y EZ KIT LITE DE ANALOG DEVICE

3.1. Procesador SHARCK 21061

SHARCK (Super Harvard Architecture Computer) es un procesador digital de la señal de 32 bits de alto desempeño. La arquitectura permite cálculos con la norma 32 bit IEEE (*floating point computation*). Las unidades de cálculo contenidas en el procesador son multiplicador, ALU y *shifter*. A continuación, en la figura 37, se describe cada una de las partes que forman este procesador.

Figura 37. **Diagrama de bloques ADSP**

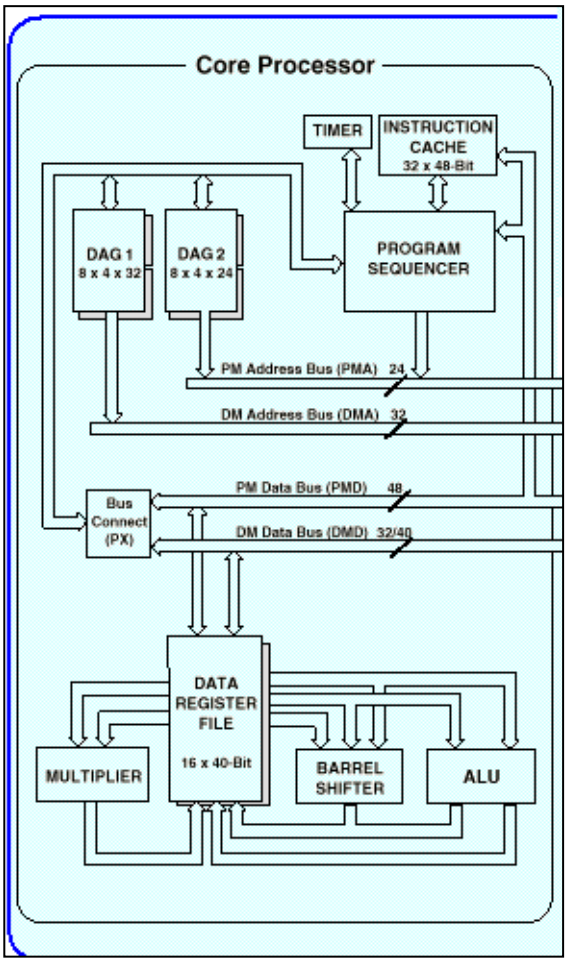


Fuente: *Manual ADSP 21061*. http://www.analog.com/media/en/technical-documentation/data-sheets/ADSP-21061_21061L.pdf. Consulta: 11 de noviembre de 2014.

3.1.1. *Core procesor*

Es el corazón del procesador, lee y decodifica las instrucciones del código de programa para ejecutar las operaciones matemáticas con los datos ubicados en los registros de entrada. A continuación, una breve explicación de cada uno de ellos.

Figura 38. **Diagrama del procesador del *core***



Fuente: *Manual ADSP 21061*. http://www.analog.com/media/en/technical-documentation/data-sheets/ADSP-21061_21061L.pdf. Consulta: 11 de noviembre de 2014.

- *Multiplier*: como su nombre lo indica realiza las multiplicaciones entre dos números contenidos en los registros según la instrucción.

- *Shifter*: deposita o extrae un número de otro, convierte un número en determinada posición a su complemento.

- ALU: realiza las operaciones de suma y resta.

- *Data register file*: provee los datos de entrada para las unidades de computación como el *multiplier*, El *shifter* y la ALU, guardando el resultado de las mismas.

- *Bus exchange*: realiza transferencias de datos entre los registros *data memory bus* y el Program memory bus.

- DAG 1 (*data address generator* 1): realiza el direccionamiento de los datos en el *data memory bus*.

- DAG 2 (*data address generator* 2): realiza el direccionamiento de los datos en el *program memory bus*.

- *Program sequencer*: lleva un control de cada una de las instrucciones dadas por el programa, contadores, salto de programas, direcciones de memoria, control de interrupciones, entre otros.

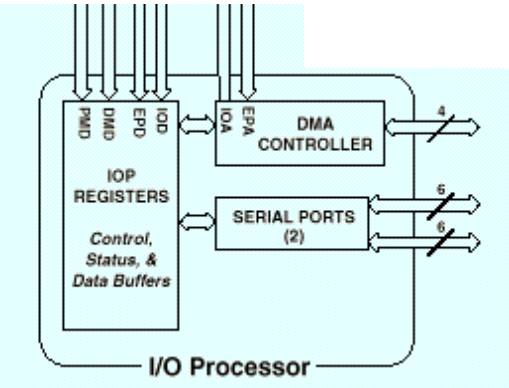
- *Instruction cache*: es un espacio de memoria destinado a guardar instrucciones que provocarían un conflicto con la memoria del programa, el caché permite la ejecución a toda velocidad de operaciones cíclicas como filtros digitales, multiplicación – adición, procesamiento FFT.

- *Timer* del ADSP 21061: es un temporizador programable que permite la generación periódica de interrupciones.

3.1.2. Procesador I/O

Incluye dos puertos seriales, cada uno con dos puertos transmisores, dos puertos receptores y un controlador DMA (*direct memory access*).

Figura 39. **Diagrama del I/O del procesador**

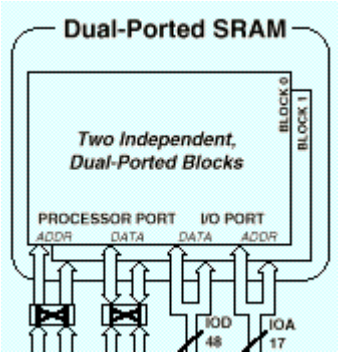


Fuente: *Datasheet ADSP 21061.* http://www.analog.com/en/products/processors-dsp/Sharck/adsp-21061.html#product-overview. Consulta: 11 de octubre de 2014.

3.1.3. SRAM con puerto dual

Es la memoria que guarda el código de programa o las instrucciones de programa, así como los registros dedicados a las constantes, datos, resultados, entre otros.

Figura 40. **Diagrama del SRAM con puerto dual**

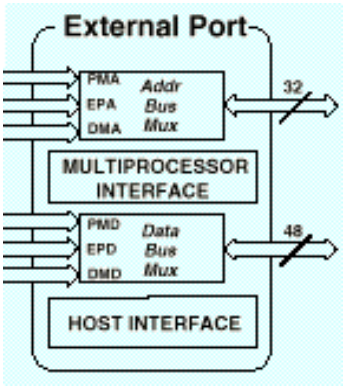


Fuente: *Datasheet ADSP 21061.* http://www.analog.com/en/products/processors-dsp/Sharck/adsp-21061.html#product-overview. Consulta: 11 de octubre de 2014.

3.1.4. *External port*

Los puertos externos del procesador proveen la interfaz entre la memoria y periféricos.

Figura 41. **Diagrama del *external port***

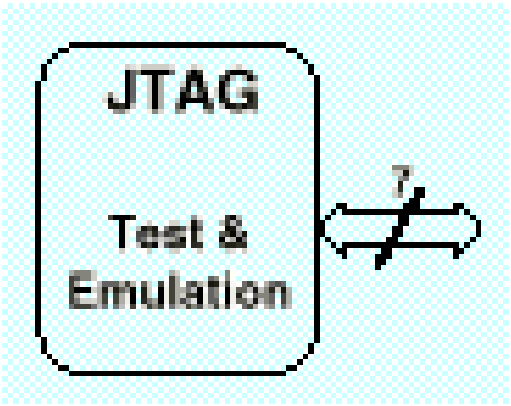


Fuente: *Datasheet ADSP 21061.* http://www.analog.com/en/products/processors-dsp/Sharck/adsp-21061.html#product-overview. Consulta: 11 de octubre de 2014.

3.1.5. Puerto JTAG

Es un puerto que permite la emulación del EZ kit por medio del procesador de la computadora, además de esto el ADSP 21061 permite el estándar de la IEEE “JTAG” (*join test action group*), que es un estándar para la prueba del sistema. Este estándar define un método para el escaneo serial del estatus de cada componente.

Figura 42. **Diagrama del JTAG componente en el sistema**

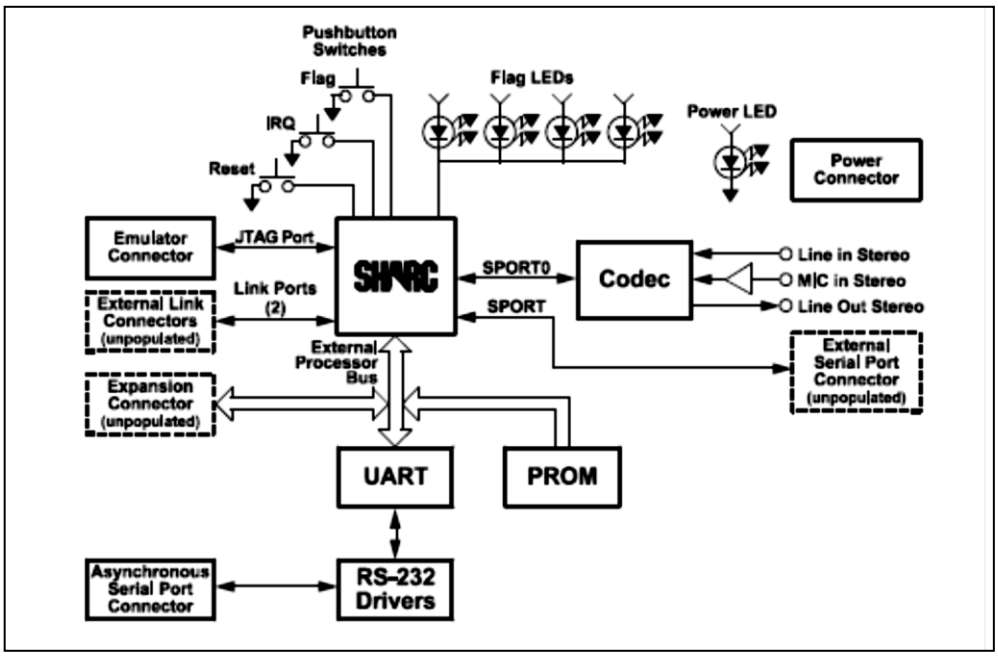


Fuente: *Datasheet ADSP 21061.* http://www.analog.com/en/products/processors-dsp/Sharck/adsp-21061.html#product-overview. Consulta: 11 de octubre de 2014.

3.2. ADSP-21061 EZ-kit lite

Es un kit de evaluación que, en conjunto con el software Visual DSP ++, crea un ambiente de desarrollo basado en el SHARCK 21061, que es un procesador digital de la señal de punto flotante. Se puede acceder al ADSP 21061 SHARCK usando una computadora a través del puerto serial o un convertidor USB a serial, teniendo acceso además a la memoria interna del procesador.

Figura 43. **Diagrama de bloques ADSP-21061 EZ-kit lite**



Fuente: *Datasheet ADSP 21061.* http://www.analog.com/en/products/processors-dsp/Sharck/adsp-21061.html#product-overview. Consulta: 11 de octubre de 2014.

El kit de evaluación incluye las siguientes características:

- *Analog devices* ADSP 21061 DSP de 40Mhz
- *Analog devices* AD1847 16 bit stereo *codec* Sound Port
- Interfaz RS232
- *Socket* EPROM
- Botones de usuario
- Led programables
- Fuente de voltaje regulada
- Conectores de expansión

El EZ-kit puede correr programas de manera autónoma al remover el chip que contiene la memoria del programa, también llamado EPROM, y remplazarlo por otra EPROM con otro código de programa o puede conectarse al puerto RS-232 de la computadora.

3.3. Visual DSP++

Es un ambiente de desarrollo integral que contiene herramientas para generar código de programa y revisión del mismo. A continuación se enumeran los componentes del VISUAL DSP ++

3.3.1. *Integrated development environment* (IDE)

Es una interfaz gráfica para la administración de los proyectos, permitiendo configurar las opciones de los proyectos y el acceso a las herramientas de generación de código

3.3.2. *Debugger*

Permite ver dentro del procesador y realizar operaciones como lectura y escritura de memoria, lectura y escritura de registros, cargar programas, correr programas paso a paso, detenerlos y muchas cosas más.

3.3.3. SHARCK *family code generation tools*

El software Visual DPS++ proporciona un método eficiente de gestión de la memoria interna del DSP. Al igual que en las generaciones anteriores de la familia de procesadores SHARCK, el software que lo administra tiene varios componentes que se describen a continuación:

- Compilador C/C++: toma un texto o código fuente escrito en C/C++, que es un lenguaje de alto nivel, y lo traslada a un lenguaje entendible por el DSP SHARCK-21061.
- Ensamblador (*assembler*): toma un código escrito en lenguaje *assembler* y lo traduce a un archivo entendible por el DSP.
- Librerías y *linker*: son un conjunto de funciones que incluyen instrucciones necesarias para la ejecución de programas. El *linker,* o enlazador, junta los recursos y bibliotecas para correr el programa. En el kit de evaluación el *linker* restringe el espacio de memoria de programa en el chip a 25 %(4k *words*) de su capacidad. Esta restricción puede ser removida al adquirir una licencia completa del Ez-kit.
- Proyectos de ejemplo: el visual DSP y el ADSP 21061 Ez-kit lite poseen proyectos de ejemplo, código C/C++ y código *assembler* para demostrar varias características de las herramientas y del ADSP 21061.

3.4. Conexiones de hardware

Las conexiones del Ez-kit permiten la interacción con la computadora que tiene el software de administración VisualDsp++ , las señales de entrada y salida de audio y conector de energía. a continuación una breve explicación de cada una de ellas.

3.4.1. Conector puerto serial (RS-232)

Es un conector hembra de 9 pines usado para comunicarse con la computadora usando niveles RS232 y protocolos seriales asíncronos.

3.4.2. Salida de audio estéreo

El conector de salida estéreo de audio conecta el canal derecho e izquierdo a los pines de salida de línea del *codec* AD1847. Usa cables de audio estándar de 1/8 de pulgada (3,5 mm) para conectar estas señales a un set de parlantes amplificados.

3.4.3. Entrada de audio estéreo

El conector de entrada estéreo de audio conecta el canal derecho e izquierdo a los pines de entrada del *codec* AD1847. Usa cables de audio estándar de 1/8 de pulgada (3,5 mm) para conectar señales con nivel de línea y se pueden conectar entradas con nivel de micrófono cambiando el *jumper* de selección de entrada.

3.4.4. Conector de fuente de voltaje de corriente directa

El conector de fuente de voltaje de corriente directa es usado para proveer un voltaje de 9v DC al kit de evaluación Ez-kit.

3.5. Dispositivos de entrada y salida

El Ez-kit lite posee dispositivos de entrada y salida que lo ayudan a interactuar con el mundo exterior. A continuación se detallan cada uno de ellos.

3.5.1. Banderas (*flags*)

El ADSP21061 tiene cuatro banderas de entrada y salida que se pueden programar como entradas o salidas. Los bits en el registro MODE2 (ver anexo 1) controlan la dirección de los *flags*.

Al momento del *reset* todas las *flags* están configuradas como entradas. Bits en el registro ASTAT (ver anexo 1) contiene el valor para cada una de las *Flags.*

3.5.2. FLAG 0

Controla el encendido del *codec* AD1847. FLAG 0 = 0 mantiene el AD1847 en *reset*, FLAG0 = 1 libera el *reset* y reinicia el AD1847. El programador deberá configurar el FLAG0 como salida y configurar en 1 durante la inicialización del programa. El FLAG0 está conectada al led D2 en el ADSP 21061 Ez-kit lite. El led D2 se ilumina cuando FLAG0 = 0.

3.5.3. FLAG1

El FLAG1 está conectado al pulsador S1 y al led D3. El programador deberá configurar como entrada durante la inicialización del programa. El led D3 se ilumina cuando el valor de entrada es 0, cuando el botón es liberado el valor es 1.

3.5.4. FLAG2

El FLAG2 está conectado al led D4. El programador deberá configurarlo como salida durante la inicialización del programa. El led D4 se iluminará cuando FLAG2 es cero.

3.5.5. FLAG3

El FLAG3 está conectado al led D6. El programador deberá configurarlo como salida durante la inicialización del programa.

3.5.6. Interrupciones externas

El ADSP SHARCK-21061 tiene tres pines de interrupción externa. Ellos están priorizados, individualmente enmascarados (registro IMASK) y pueden ser configurados como sensitivos en subida o sensitivos en nivel (bits en el registro MODE2). Al momento del *reset*, todas las interrupciones externas son sensitivas en nivel y enmascaradas.

- IRQ0: no está conectada en el ADSP21061 Ez kit lite, pero sí está conectada hacia uno de los conectores de expansión.
- IRQ1: está conectada al pulsador en el ADSP21061 Ez-kit lite, presionando el pulsador se genera la interrupción.
- IRQ2: está conectada al UART 16550, la cual puede ser programada para generar una interrupción en caso sea necesaria

3.5.7. Puertos seriales

El ADSP21061 posee dos puertos síncronos de alta velocidad (SPORT). Ellos pueden operar en conexión punto a punto en modo *full* dúplex con transmisión y recepción de datos y relojes, o ellos pueden estar conectados con múltiples SPORT para operar en modo TDM (*time división multiplex*).

- Rutinas POST: son rutinas de prueba e inicialización que el Ez-kit lite realiza en el encendido o el *reset*.
- Revisión/inicialización del UART: se realiza en tres etapas, ellas son:

    - Escritura de registro: esta prueba confirma que el ADSP 21061 SHARCK es capaz de escribir el registro en el UART. Tres patrones son escritos y leídos en el registro del UART estos deben estar bien escritos para pasar la prueba.
    - *Loop back* interno: en esta prueba, 256 bytes son enviados y leídas desde el UART. Esta prueba revisa la funcionalidad de las conexiones del UART hacia el ADSP 2106 SHARCK
    - *Loop back* transmitido: es realizada por la computadora después de que el POST ha terminado. En esta prueba la computadora manda el protocolo de prueba del UART. Este protocolo especifica el número de bytes que fueron transmitidos al Ez-kit lite y ordena al Ez-kit hacer eco y enviar el mismo chorro de bytes hacia la computadora. Esta prueba determina que el Ez-kit está configurado a la velocidad de transmisión correcta, así como la conexión entre el Ez-kit y la computadora

3.5.8. Operación del programa monitor

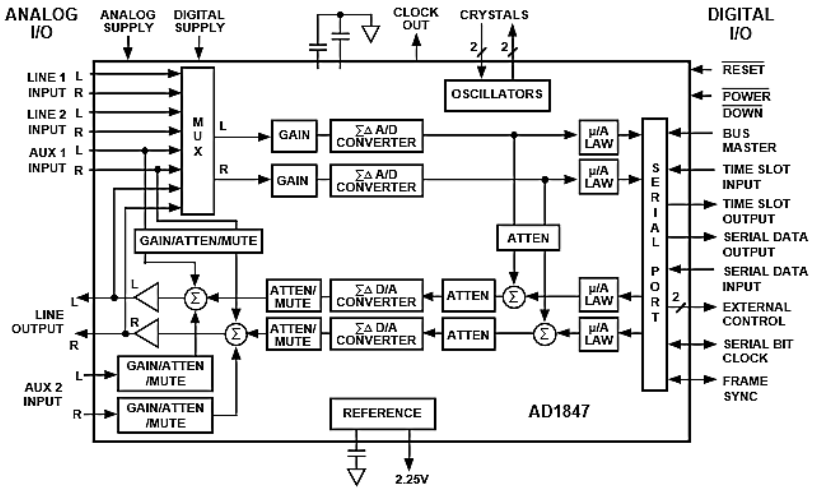
El Visual DSP++ actúa como monitor del Ez-kit lite permitiendo la revisión y la carga de los programas del usuario.

3.6. *Codec* AD1847 *sound port*

El AD1847 *sound port* es el *codec* utilizado en el Ez-kit ADSP21061, posee las siguientes cualidades:

- Un solo circuito integrado contiene *codec* de audio digital estéreo.
- Soporta el sistema de audio de Microsoft Windows.
- Múltiples canales de entrada estéreo.
- Soporta los siguientes formatos: - 16bit PCM, 96dB de rango dinámico - 8bit *compamding* Ley m y A, 48dB rango dinámico
- Ganancia y atenuación programable
- Filtros incluidos en el chip: interpolación y decimación.
- Salida paso bajo análoga.
- Frecuencias de muestreo entre 5,5 Khz hasta 48 Khz.
- Voltaje de operación de +5V
- Interfaz digital serial compatible con ADSP-21xx Fixed-Point DSP
- El *Codec* incluye un par esterero de ADC y un par estéreo de DAC.
- Las entradas al ADC pueden seleccionar entre cuatro pares estéreo de señales análogas: línea 1, línea 2, línea auxiliar #1 y una salida DAC poscombinada.

Figura 44. **Diagrama *codec* AD1847 *sound port***



Fuente: *Datasheet ADSP 21061*. http://www.analog.com/en/products/processors-dsp/Sharck/adsp-21061.html#product-overview. Consulta: 11 de octubre de 2014.

Tomar las siguientes condiciones de operación para futuros proyectos: (ver anexo 1 para detalles de los registros contenidos en el AD1847).

- El AD1847 es un dispositivo sensible a descargas electrostáticas
- Impedancia de entrada mínima 10K ohm
- Capacitancia de entrada 15 pF
- Voltaje de entrada 1 Vrms
- Voltaje de salida 0,707 Vrms
- Impedancia de salida 600 ohm
- Capacitancia de salida 15 pF

4. PROPUESTA DE PRÁTICAS DE LABORATORIO EZ KIT DSP SHARCK 21061

4.1. Familiarización con el entorno de programación Visual DSP++

- Equipo a utilizar

    - Ez kit SHARCK 21061
    - Visual DSP++ 3,0 (software)
    - 2 cables estéreo estándar
    - Bocinas estándar para computadora

- Herramientas de programación

    - C/C++ Compiler: o compilador, este programa lee la sintaxis del programa escrita en lenguaje C que es un lenguaje de alto nivel y lo traduce a lenguaje de máquina o secuencias de bits que el DSP ejecutará como instrucciones.
    - *Assembler*: es un lenguaje de bajo nivel que utiliza nemónicos necesarios para programar la arquitectura del DSP, este código es traducido a lenguaje de máquina. Este tipo de programación optimiza los recursos del DSP haciéndolo eficiente a la hora de ejecutar el programa.
    - *Linker*: o un editor de enlaces es un programa que toma uno o más objetos generados por el compilador y los combina en un solo programa ejecutable.
    - *Loader*: carga el programa en la PM del DSP.

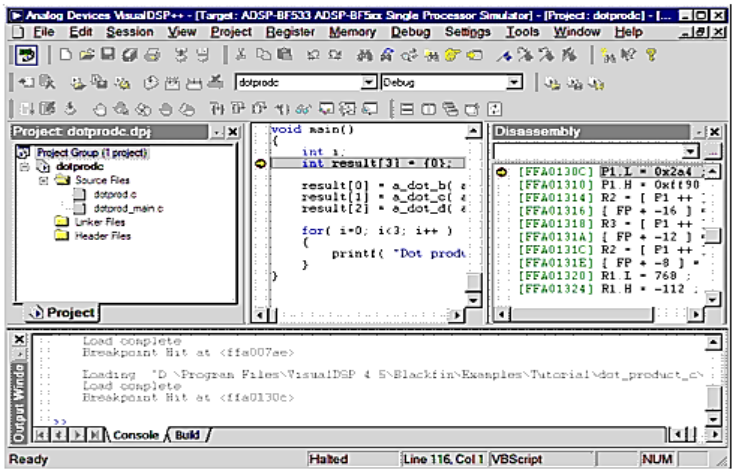
- *Simulator*: el simulador permite la ejecución de programas en un procesador virtual, la conexión de él Ez kit no es necesaria en esta modalidad.

4.2. Opciones de construcción de proyectos

El entorno de Visual DPS++ permite construir archivos de programación o proyectos, así como la actualización del programa cargado en el DPS después de las modificaciones, realiza reportes de error en el código de programa mostrando el origen de los mismos, reduciendo tiempo en la revisión de errores.

Visual DSP++ es un entorno flexible que permite editar, construir y cargar el programa al Ez-kit 21061. A continuación, la figura 45, muestra el entorno del IDDE.

Figura 45. **IDDE**



Fuente: *Linker and utilites manual*. http://www.analog.com/media/en/dsp-documentation/software-manuals/50_lnkr_mn_rev_3.5.pdf. Consulta: agosto de 2014.

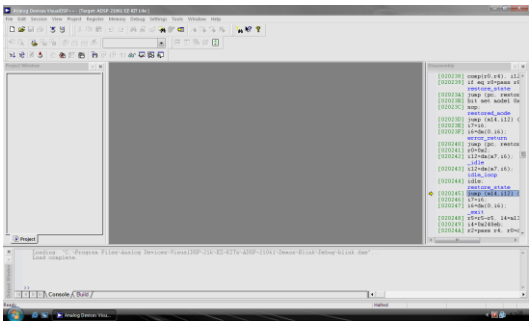
4.3. Conectando el equipo

- Se recomienda usar brazalete antiestático al manipular el equipo.
- Conectar el cable serial del Ez kit 21061 al puerto com1 de su computadora.
- Conectar la fuente de poder al Ez kit 21061, tener el cuidado de conectar primero el terminal al equipo antes de conectar los 110 volts de CA comercial.
- Conectar los parlantes externos a la salida de audio del Ez kit, deberá de escucharse una melodía indicando que el equipo está en perfecto estado.

4.3.1. Ejercicio número 1: construyendo y ejecutando un programa en lenguaje C

- Presionar el botón de inicio, diríjase a programas, luego a la carpeta Analogs Devices y, finalmente al programa visual DSP++. El programa en su primera ejecución se verá como en la figura 46.

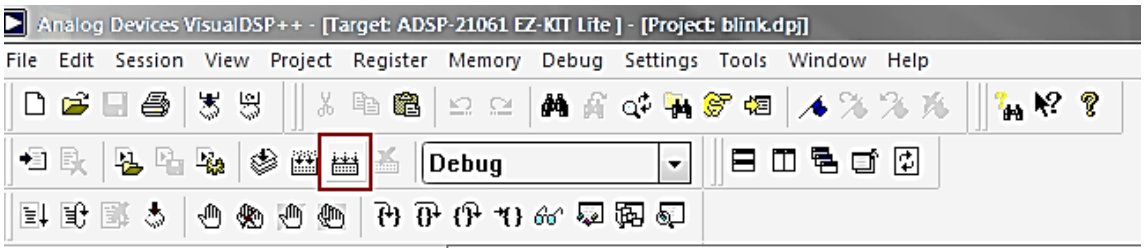
Figura 46. **IDDE Visual DSP++**



Fuente: *Linker and utilites manual*. http://www.analog.com/media/en/dsp-documentation/software-manuals/50_lnkr_mn_rev_3.5.pdf. Consulta: agosto de 2014.

- Abrir un proyecto de demostración llamado Blink, seguidamente en el menú Project, diríjase a Open…, en la dirección *C:\Program Files\AnalogDevices\Visual DSP\21k\EZ-KITs\ADSP-21061\Demos\Blink*. Presione Open.
- Para compilar el programa se presiona el botón Rebuild all.

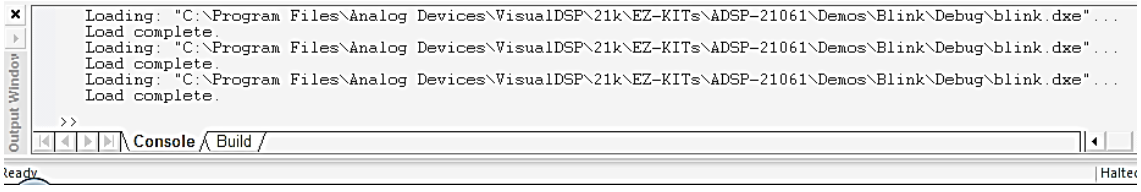
Figura 47. **Compilando el programa**



Fuente: *Linker and utilites manual*. http://www.analog.com/media/en/dsp-documentation/software-manuals/50_lnkr_mn_rev_3.5.pdf. Consulta: agosto de 2014.

- Luego de compilar aparecerá el estatus de la carga del programa en la ventana *Ouput Window*.

Figura 48. ***Ouput window***

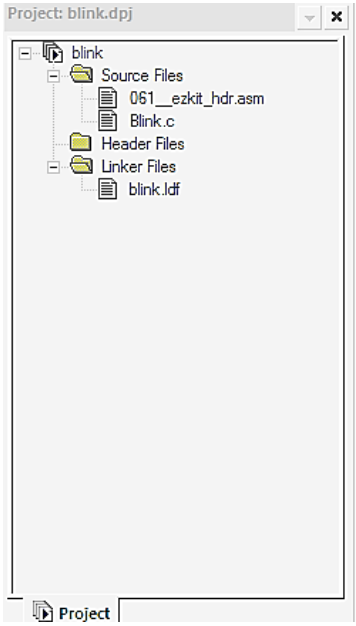


Fuente: *Linker and utilites manual*. http://www.analog.com/media/en/dsp-documentation/software-manuals/50_lnkr_mn_rev_3.5.pdf. Consulta: agosto de 2014.

Esta ventana muestra el estatus de la compilación, carga del programa hacia la PM del DSP y los posibles errores, indicando la línea y columna en donde se encuentra el error. Si el programa fue cargado exitosamente se verá el mensaje l*oad complete* al final de la mensajería.

También aparecerá en pantalla el archivo en lenguaje C llamado Blink.c . que puede ser desplegado desde la ventana “Project: blink.dpj”.

Figura 49. **Ventana Project**



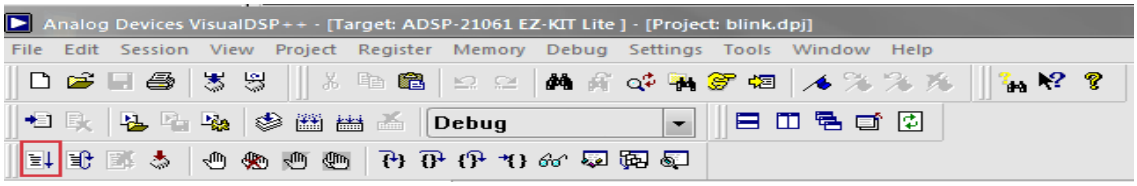
Fuente: *Linker and utilites manual*. http://www.analog.com/media/en/dsp-documentation/software-manuals/50_lnkr_mn_rev_3.5.pdf. Consulta: agosto de 2014.

Esta ventana contiene tres tipos de archivos:

- *Sources files*: contiene los archivos *.C o *asm con los programas a ejecutar

- *Header files*: contiene los coeficientes de los filtros digitales o constantes a usar en el programa y básicamente se usa para cargar constantes de una manera más ordenada.
- *Linker files*: este archivo contiene la configuración de registros, direcciones de memoria, interrupciones, entre otros. En pocas palabras, contiene la configuración correspondiente a la arquitectura del DSP.
- Para ejecutar el programa presionar el botón Run en la barra de herramientas.

Figura 50. **Botón Run**



Fuente: *Linker and utilites manual*. http://www.analog.com/media/en/dsp-documentation/software-manuals/50_lnkr_mn_rev_3.5.pdf. Consulta: agosto de 2014.

El programa Blink.c hace que los leds correspondientes a los FLAG2 y FLAG3 se activen alternadamente.

- Recomendaciones: antes de apagar asegurarse de:

  - Detener el programa que se está ejecutando, presionar botón Halt de la barra de herramientas.
  - Cerrar el proyecto en el menú Project después Close.
  - Cerrar el programa Visual DSP++.
  - Presionar el botón *reset* en el Ez kit 21061.
  - Desconectar la fuente de poder de Ez kit 21061.

4.4. Estructura del archivo de programación en lenguaje C Visual DSP++

- Equipo a utilizar

    - Ez kit SHARCK 21061
    - Visual DSP++ 3,0 (software).
    - 2 cables estéreo estándar
    - Bocinas estándar para computadora.

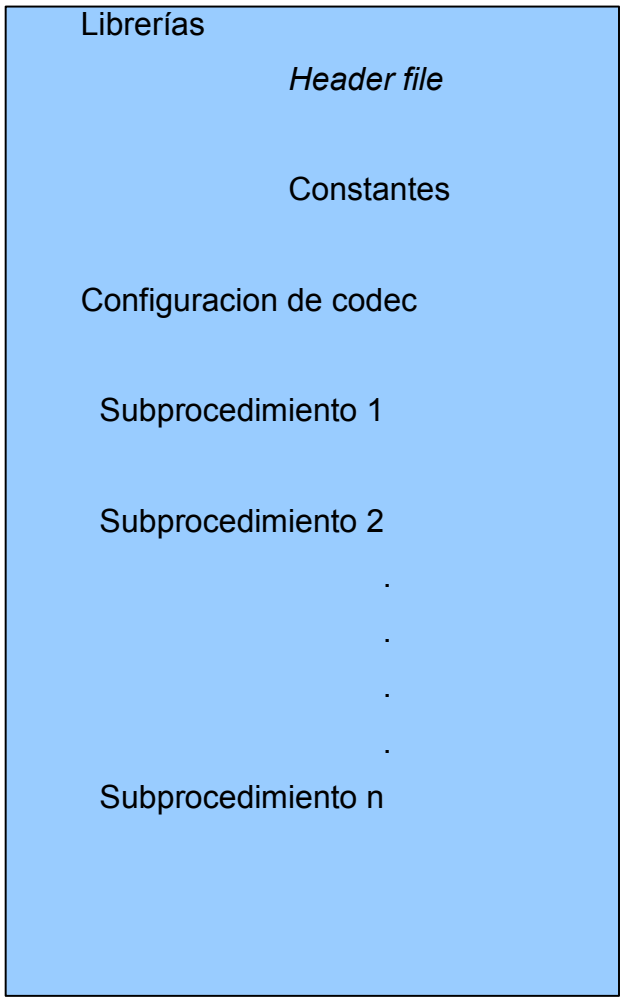
4.4.1. Estructura del archivo de programación en lenguaje C para el IDDE Visual DSP++

Para esta práctica se utilizará como ejemplo el archivo Blink.C que está ubicado en la dirección C:\Program Files\Analog Devices\VisualDSP\21k\EZ-KITs\ADSP-21061\Demos\Blink.C

La estructura de un programa en lenguaje C para el IDDE Visual DSP++ generalmente se enumera de la siguiente forma.

- Librerías (*include files*), Archivo de coeficiente (*header file*) y declaración de constantes.
- Configuración de *codec* (se verá más detalladamente en prácticas posteriores).
- Subprocedimientos (*void*).
- Procedimiento principal (*void main*).

Figura 51. **Estructura de un programa en Visual DSP++**



Fuente: *Linker and utilites manual*. http://www.analog.com/media/en/dsp-documentation/software-manuals/50_lnkr_mn_rev_3.5.pdf. Consulta: agosto de 2014.

4.4.2. Librerías

Están compuestas por funciones, muchas de las funciones que tienen otros lenguajes de programación no están en C, un ejemplo de ello es que no hay funciones para manejo de cadenas de caracteres, funciones matemáticas, gráficos, funciones especiales de tratamiento digital de la señal. Estas últimas

se encuentran en la librería filters.h que contiene funciones tales como: fir , iir, convolve, entre otras. De las que se explicará más adelante. El programador puede crear sus propias librerías con funciones especiales.

Tabla III. **Librerías estándar**

Header	Purpose	Standard
assert.h	Diagnostics	ANSI
ctype.h	Character Handling	ANSI
cycle_count.h	Basic Cycle Counting	Analog extension
cycles.h	Cycle Counting with Statistics	Analog extension
device.h	Macros and data structures for alternative device drivers	Analog extension
device_int.h	Enumerations and prototypes for alternative device drivers	Analog extension
errno.h	Error Handling	ANSI
float.h	Floating Point	ANSI
iso646.h	Boolean Operators	ANSI
limits.h	Limits	ANSI
locale.h	Localization	ANSI
math.h	Mathematics	ANSI
setjmp.h	Non-Local Jumps	ANSI
signal.h	Signal Handling	ANSI
stdarg.h	Variable Arguments	ANSI

Continuación de la tabla III.

Description	Prototype
Generate Bartlett window	`void gen_bartlett (float w[], int a, int n)`
Generate Blackman window	`void gen_blackman (float w[], int a, int n)`
Generate Gaussian window	`void gen_gaussian (float w[], float alpha, int a, int n)`
Generate Hamming window	`void gen_hamming (float w[], int a, int n)`
Generate Hanning window	`void gen_hanning (float w[], int a, int n)`
Generate Harris window	`void gen_harris (float w[], int a, int n)`
Generate Kaiser window	`void gen_kaiser (float w[], float beta, int a, int n)`
Generate rectangular window	`void gen_rectangular (float w[], int a, int n)`
Generate triangle window	`void gen_triangle (float w[], int a, int n)`
Generate von Hann window	`void gen_vonhann (float w[], int a, int n)`

Fuente: *Linker and utilites manual*. http://www.analog.com/media/en/dsp-documentation/software-manuals/50_lnkr_mn_rev_3.5.pdf. Consulta: agosto de 2014.

Tabla IV. **Funciones de la librería filters.h**

biquad	fir	iir
`float biquad (float sample, const float pm coeffs[], float dm state[], int sections);`	`float fir (float sample, const float pm coeffs[], float dm state[], int taps);`	`float iir (float sample, const float pm a_coeffs[], const float pm b_coeffs[], float dm state[], int taps);`

Fuente: *Linker and utilites manual*. http://www.analog.com/media/en/dsp-documentation/software-manuals/50_lnkr_mn_rev_3.5.pdf. Consulta: agosto de 2014.

4.4.2.1. Librería

La sintaxis para declarar el uso de una librería se hace anteponiendo la palabra *include* por ejemplo:

- #include <def21060.h>: librería específica para el uso del procesador Sharck 21061.
- #include <21060.h>: librería específica para el uso del procesador Sharck 21061.
- #include <signal.h>: maneja las funciones de interrupción durante la ejecución del programa.
- #include <sport.h>:provee definiciones, configuraciones, activa y desactiva funciones para los puertos seriales.
- #include <macros.h>: usada en el manejo de registros circulares en los ADSP-21xxx.

4.4.2.2. Definiciones de constantes y variables

A continuación se da la descripción de constantes y variables.

4.4.2.2.1. Constantes

Es un valor que no variará a lo largo de la ejecución del programa. Para declarar una constante se antepone la palabra *#define.*

Declaraciones de constantes y macros:

- #define CLOCK_RATE 20000000 /* Periodo del ciclo */.

- #define SetIOP(addr,val) (* (volatile int *) addr)=(val) //toma un valor y lo carga como constante.
- #define GetIOP(addr) (* (volatile int *) addr).

4.4.2.2.2. Variables

Son espacios en memoria que pueden cambiar su valor a lo largo de la ejecución del programa, estas pueden almacenar diferentes tipos de datos como números o caracteres. A continuación se encuentran listados los tipos de dato y su tamaño

Figura 52. **Tipos de datos**

Type	Bit Size	Result of sizeof operator
int	32 bits signed	1
unsigned int	32 bits unsigned	1
long	32 bits signed	1

Type	Bit Size	Result of sizeof operator
unsigned long	32 bits unsigned	1
long long	64 bits signed	2
unsigned long long	64 bits unsigned	2
bool	32 bits signed	1
char	32 bits signed	1
unsigned char	32 bits unsigned	1
short	32 bits signed	1
unsigned short	32 bits unsigned	1
pointer	32 bits	1
float	32 bits float	1
short fract	32 bits fixed-point	1
fract	32 bits fixed-point	1
long fract	32 bits fixed-point	1
unsigned short fract	32 bits unsigned fixed-point	1
unsigned fract	32 bits unsigned fixed-point	1
unsigned long fract	32 bits unsigned fixed-point	1
double	either 32 or 64 bits float (default 32)	either 1 or 2 (default 1)
long double	64 bits float	2

Fuente: *Linker and utilites manual*. http://www.analog.com/media/en/dsp-documentation/software-manuals/50_lnkr_mn_rev_3.5.pdf. Consulta: Agosto de 2014.

4.4.2.3. Subprocedimientos

Son tareas o conjuntos de códigos que realizan una función o devuelven un valor para ser utilizado por el procedimiento principal *(void main).*

- Interrupción periódica del *timer void* timer_prior(int sig_num)
- Advertencia silenciosa del compilador
- Encendiendo y apagando los leds
- set_flag(SET_FLAG2, TGL_FLAG)
- set_flag(SET_FLAG3, TGL_FLAG)
- SetIOP(MSGR0, GetIOP(MSGR0)+1)
- Return

4.4.2.4. Procedimiento principal

El procedimiento principal necesita de los subprocedimientos para funcionar correctamente.

- Función: *main*
- Argumento: ninguno
- Devuelve valor: ninguno
- Descripcion: configura , procesa los datos y realiza un ciclo sin fin
- *Void main* (*void*)
- *Int data*; *timer_off*
- Inicializando los datos data = CLOCK_RATE
    - set_flag(SET_FLAG2, CLR_FLAG)
    - set_flag(SET_FLAG3, SET_FLAG)
    - SetIOP(MSGR0, 0)

- *Asm* ("BIT SET MODE2 0x00000002;"); haciendo las interrupciones 0,1,2 sensitivas en 1 lógico, note que *asm* significa que está en código *assembler*

- Configura e inicia el *timer*

    - timer_set(data, data)
    - timer_on

- Configuración del controlador de interrupciones

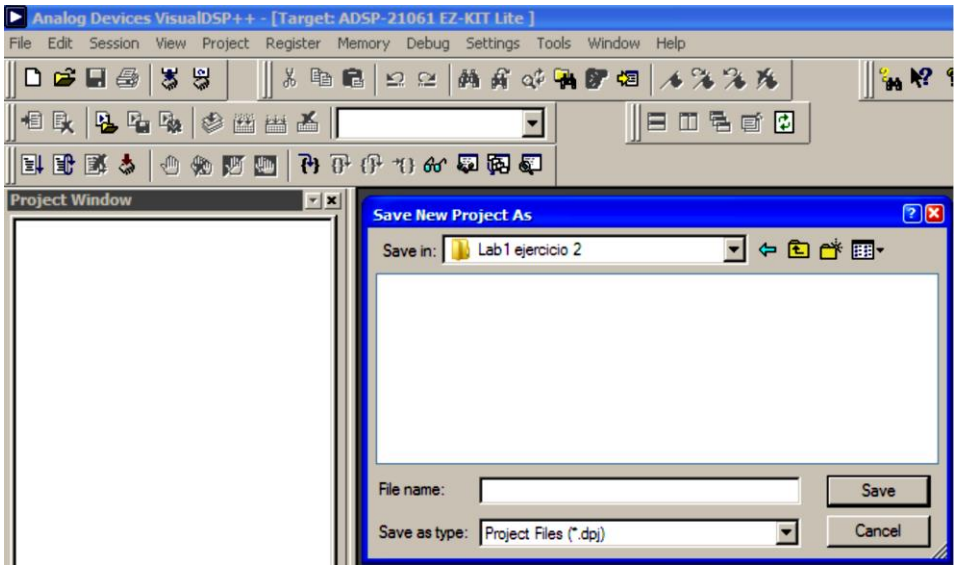
    - interrupt(SIG_TMZ0, timer_prior); llama a ejecutar el subprocedimiento “timer_prior” que actua como controlador de la interrupción que provoca el SIG_TMZ0 (timer0) ver Help signal.h while(1) ciclo para siempre {idle(); }

- Fin de programa Blink.C

4.4.3. Cómo crear un proyecto, compilación y edición

- Crear proyecto: menú Project → New → crear una carpeta y guardar el proyecto como Ejercicio 2.

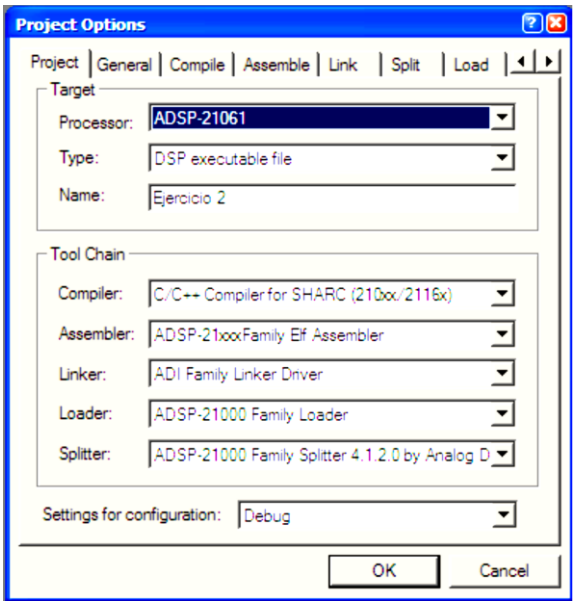
Figura 53. **Nuevo proyecto**



Fuente: *Linker and utilites manual*. http://www.analog.com/media/en/dsp-documentation/software-manuals/50_lnkr_mn_rev_3.5.pdf. Consulta: agosto de 2014.

- Copiar el archivo C:\Program Files\Analog Devices\VisualDSP\21k\EZ-KITs\ADSP-21061\Demos\Blink\Blink.c a la carpeta.
- Copiar el archivo C:\Program Files\Analog Devices\VisualDSP\21k\EZ-KITs\ADSP-21061\Demos\Blink\blink.ldf a la carpera Lab1 ejercicio2
- Asegurarse de que el *target* a utilizar sea 21061.

Figura 54. **Opciones de proyecto**



Fuente: *Linker and utilites manual*. http://www.analog.com/media/en/dsp-documentation/software-manuals/50_lnkr_mn_rev_3.5.pdf. Consulta: agosto de 2014.

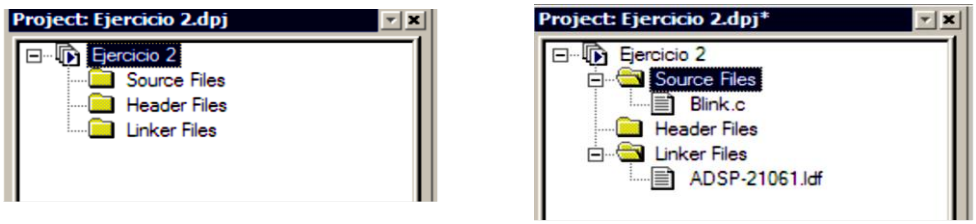
4.4.3.1. Adicionando *files* al proyecto

Diríjase a la ventana Project, en donde existen tres carpetas: Source Files, Header Files y Linker Files.

- Cargar el archivo *.C, hacer clic derecho en Source files → Add file(s) to folder.
- Cargar el archivo *.ldf hacer clic derecho en Linker files → Add file(s) to folder.

Quedará de la siguiente forma:

Figura 55. **Ventana Proyect Files**

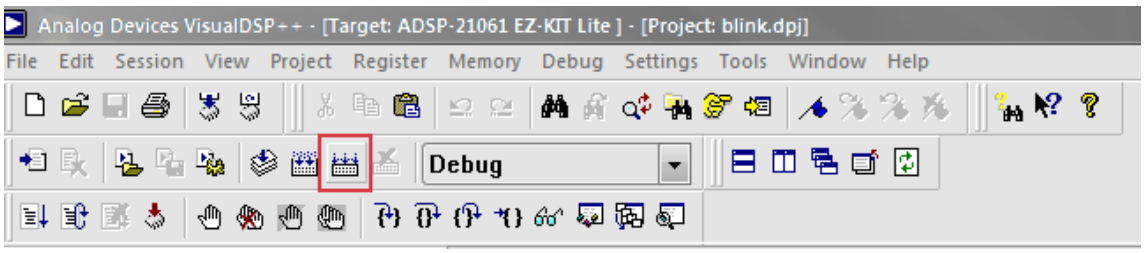


Fuente: *Linker and utilites manual*. http://www.analog.com/media/en/dsp-documentation/software-manuals/50_lnkr_mn_rev_3.5.pdf. Consulta: agosto de 2014.

4.4.3.2. Compilación

Para compilar el programa se presiona el botón Rebuild all.

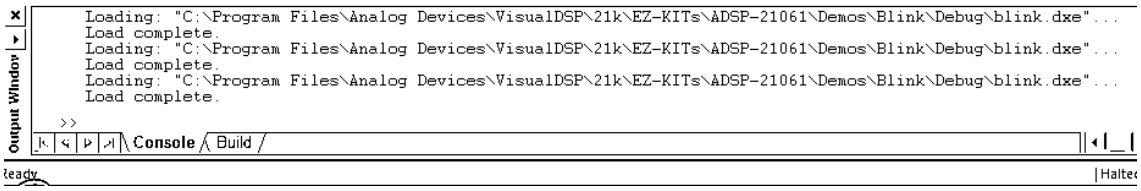
Figura 56. **Compilando el programa**



Fuente: *Linker and utilites manual*. http://www.analog.com/media/en/dsp-documentation/software-manuals/50_lnkr_mn_rev_3.5.pdf. Consulta: agosto de 2014.

Luego de compilar aparecerá el estatus de la carga del programa en la ventana *Ouput window*.

Figura 57. ***Ouput window***



Fuente: *Linker and utilites manual*. http://www.analog.com/media/en/dsp-documentation/software-manuals/50_lnkr_mn_rev_3.5.pdf. Consulta: agosto de 2014.

Esta ventana muestra el estatus de la compilación, carga del programa hacia la PM del DSP y los posibles errores, indicando la línea y columna en donde se encuentra el error. Si el programa fue cargado exitosamente se verá el mensaje *Load complete* al final de la mensajería.

4.4.3.3. Edición

Pasos para la edición del código una vez compilado y cargado en la memoria del kit:

- Realizar el cambio deseado en los archivos *.c , *.ldf o *h (*header files* que se usará más adelante en los laboratorios de filtros digitales).
- Guardar el cambio con el botón Save file.
- Compilar de nuevo el programa con el botón Rebuild all.

4.5. Visualización y generación de señales Matlab–Visual DSP++

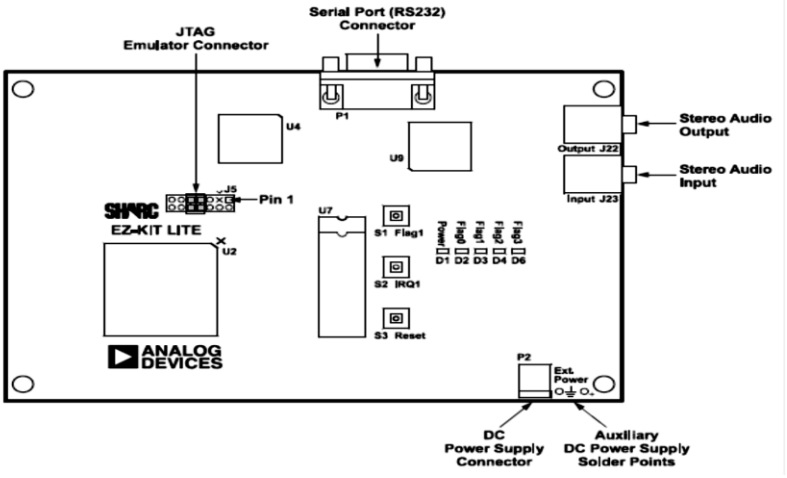
- Equipo a utilizar

    - Ez kit SHARCK 21061.
    - Visual DSP++ 3,0 (sofware).
    - Matlab 7 (software).
    - Micrófono estándar para computadora.
    - Bocinas estándar para computadora.
    - Cable estéreo 3,5 mm

Las características de las señales están limitadas por la capacidad del *codec* AD1847, el cual tiene límites en cuanto al voltaje y la frecuencia máxima que puede manejar.

El Ez kit 21061 posee una entrada de audio estéreo que conecta el canal izquierdo (l) y derecho (r) con la entrada del codificador -> decodificador AD1847. Para conectar, se utilizan los cables de audio estándar con el conector estéreo de 1/8 pulgada (3,5 mm), se puede conectar también una señal de micrófono o de línea, cambiando los puentes del selector de la fuente de entrada. La salida estéreo conecta el canal izquierdo (l) y derecho (r) con la salida del *codec* AD1847 y utiliza el cable de audio estándar con el conector estéreo de 1/8 pulgada (3,5 mm).

Figura 58. **Diagrama esquemático de puertos**



Fuente: *Linker and utilites manual*. http://www.analog.com/media/en/dsp-documentation/software-manuals/50_lnkr_mn_rev_3.5.pdf. Consulta: agosto de 2014.

4.5.1. Voltajes

- Voltaje de entrada 1 Vrms
- Voltaje de salida 0,707 Vrms, 1 vp-p

4.5.2. Frecuencias

Las frecuencias de la señal de entrada están limitadas por la frecuencia de muestreo del *codec* que está en el rango de 5,5 Khz hasta 48 Khz. Si se toma en consideración el teorema de Nyquist, la frecuencia máxima de entrada al Ez kit es de 24 Khz a una frecuencia de muestreo de 48 Khz.

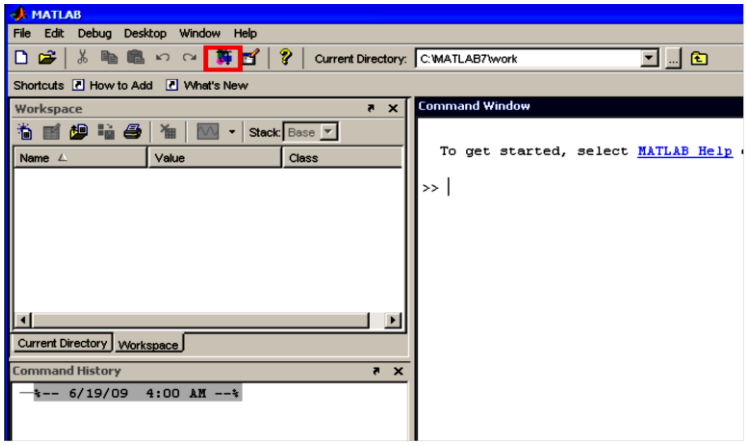
4.5.3. Impedancia y capacitancia

- Impedancia de entrada mínima 10K ohm
- Capacitancia de entrada 15 pF
- Impedancia de salida 600 ohm
- Capacitancia de salida 15pF

4.6. Utilizando Matlab como visualizador de espectro de frecuencias

Una herramienta que ayudará a visualizar el espectro de frecuencias de la señal de audio a trabajar, es un componente de Matlab llamado Simulink.

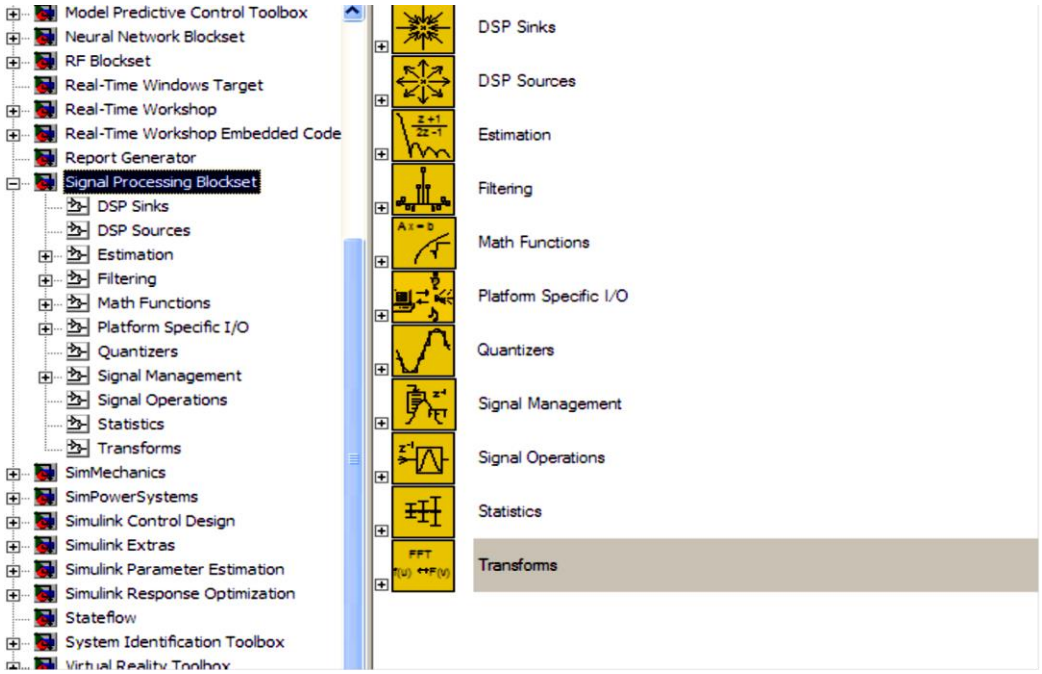
Figura 59. **Accesando al *tool* Simulink de Matlab**



Fuente: *Linker and utilites manual*. http://www.analog.com/media/en/dsp-documentation/software-manuals/50_lnkr_mn_rev_3.5.pdf. Consulta: agosto de 2014.

Este acceso directo desplegara la ventana Simulink Library Browse. Se buscará en la sección Signal Processing Blockset.

Figura 60. ***Signal processing blockset***



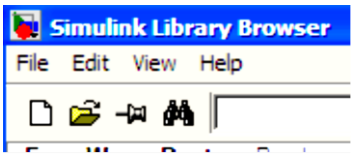
Fuente: *Linker and utilites manual*. http://www.analog.com/media/en/dsp-documentation/software-manuals/50_lnkr_mn_rev_3.5.pdf. Consulta: agosto de 2014.

La idea es usar la entrada de micrófono de la computadora como punto de prueba para conocer las cualidades de la señal de entrada al sistema o las cualidades de la señal de salida de los filtros digitales a diseñar.

4.6.1. Crear un nuevo modelo

En el menú principal hacer clic en Create a new model.

Figura 61. **Menú principal de Simulink**



Fuente: *Linker and utilites manual*. http://www.analog.com/media/en/dsp-documentation/software-manuals/50_lnkr_mn_rev_3.5.pdf. Consulta: agosto de 2014.

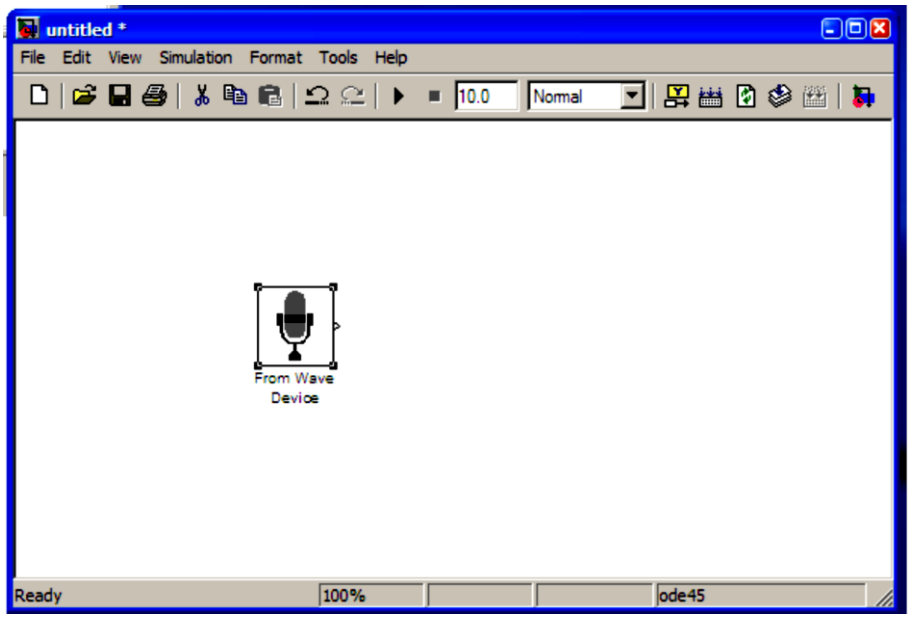
4.6.2. Agregar dispositivos al modelo

En la librería Plataform specific I/vO se encuentran los elementos de Windows (win32) que por lo general son cuatro:

- *From wave device*: dispositivo de entrada de audio, como la entrada de micrófono y la entrada de línea de la tarjeta de audio de la computadora.
- *From wave file*: lee el audio grabado en un archivo WAV PCM de cualquier directorio y también es posible especificar cuantas veces se quiere reproducir el archivo.
- *To wave device*: dispositivo de salida de audio o salida de audífono o bocina de la tarjeta de audio de la computadora.
- *To wave file:* guarda un archivo de audio de formato wav PCM.

Arrastrar el dispositivo *from wave device* hacia la ventana en blanco del nuevo modelo.

Figura 62. **Modelo Simulink**



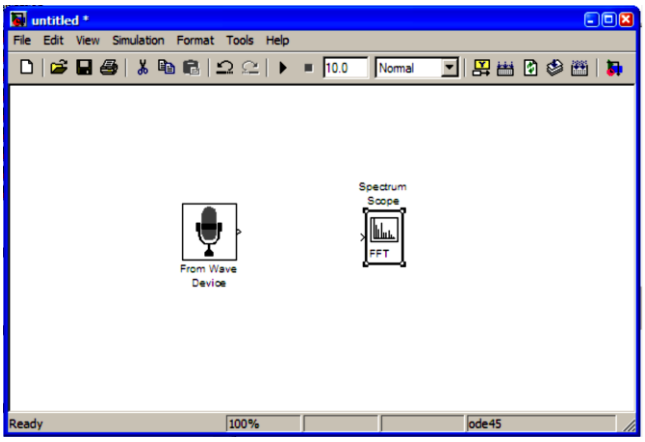
Fuente: elaboración propia, empleando Matlab.

En la librería DSP sinks se encuentran diversos visualizadores de señal:

- *Spectrum scope*: gráfica la señal de entrada en unidades de frecuencia *versus* amplitud (dB). Las iniciales FFT significa *fast furier transform*, algoritmo que aprovecha y que transforma la señal de su dominio en tiempo hacia su dominio en frecuencia.
- *Time scope*: gráfica la señal de entrada en el dominio del tiempo.
- *Vector scope*: gráfica la señal de entrada en dominio del tiempo, Frecuencia o cualquier otro dato que el usuario requiera

Arrastre el dispositivo *Spectrum scope* hacia la ventana del nuevo modelo.

Figura 63. **Espectro de frecuencia**

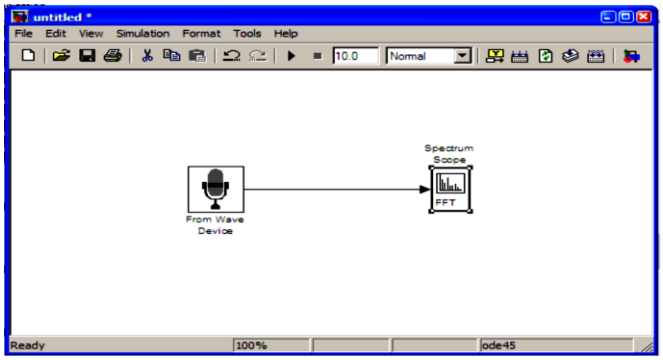


Fuente: elaboración propia, empleando Matlab.

4.6.3. Interconexión de dispositivos

Colocando el cursor sobre el símbolo > dispositivo, luego manteniendo el boton derecho del ratón y arrastrando hacia el visor de espectro de frecuencia se conectarán ambos dispositivos.

Figura 64. **Interconexión de dispositivos**



Fuente: elaboración propia, empleando Matlab.

4.6.4. Configuración de dispositivos

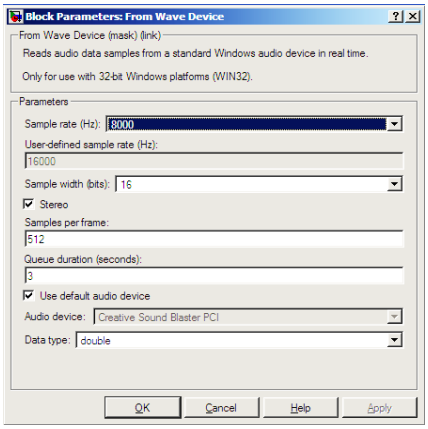
Como parte del proceso de configuración, se asigna un conjunto único de recursos del sistema al dispositivo que se va a instalar.

4.6.5. Configuración *From wave device*

Al hacer doble clic en el dispositivo *From wave device* se desplegará la ventana de configuración que toma en cuenta los siguientes parámetros:

- *Sample rate*: frecuencia de muestreo que se aplicara a la señal.
- *Sample bits*: resolución en bits de cada muestra.
- *Sample per frame*: cantidad de muestras por bloque.
- *Stereo*: se desplegara una o dos señales.
- *Queue duration*: duración máxima en segundos de la consulta de información.
- *Data type*: formato del dato que saldrá del dispositivo.

Figura 65. **Configuración *From wave device***



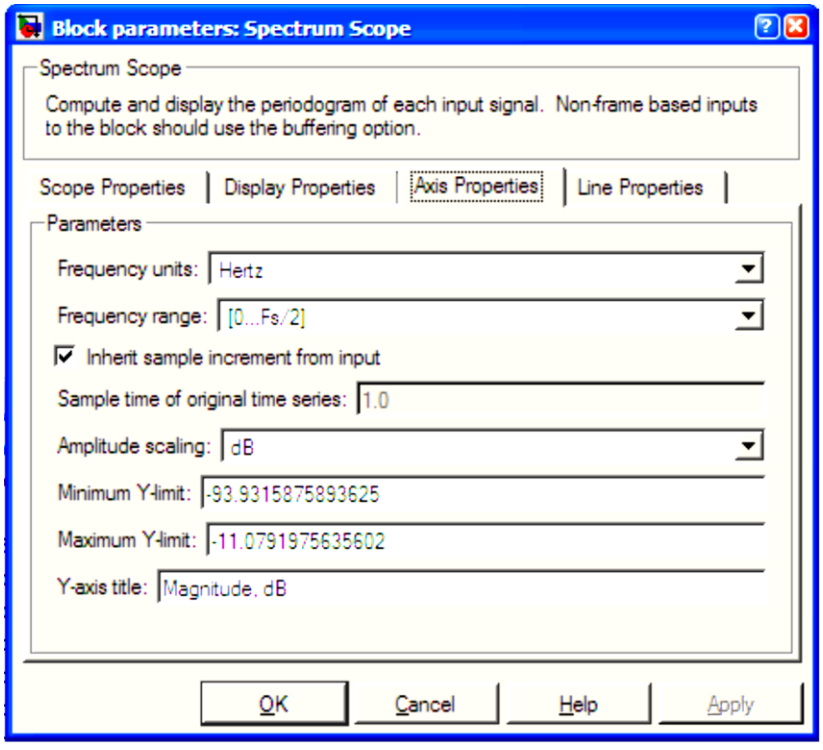
Fuente: elaboración propia, empleando Matlab.

4.6.5.1. Configuración *Spectrum scope*

Haciendo doble clic en el dispositivo *Spectrum scope* se desplegará la ventana de configuración que toma en cuenta los siguientes parámetros:

- *Scope properties*: tamaño del *buffer* de entrada y longitud de la FFT.
- *Display properties*: propiedades de la gráfica como grilla y etiquetas.
- *Axis properties*: unidades de frecuencia como radianes por segundo o hertz, rango de frecuencia a visualizar, unidades de amplitud como magnitud o decibelio y límites de visualización.

Figura 66. **Configuración *Spectrum scope***

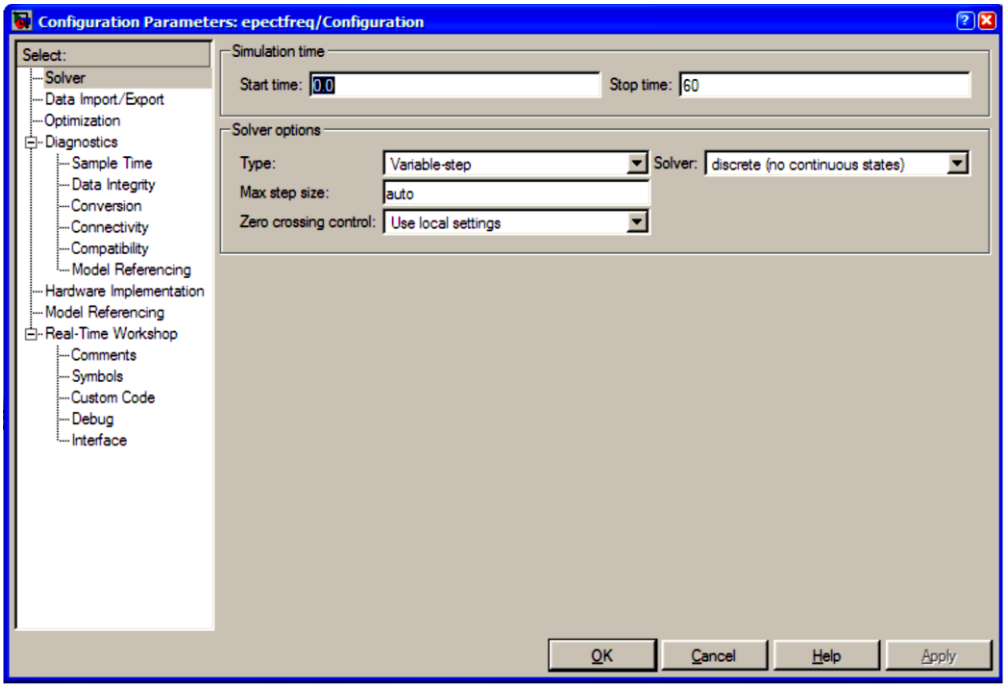


Fuente: elaboración propia, empleando Matlab.

4.6.6. Configuración de modelo

En el menú Simulation → Configuration Parameters... → desplegará una serie de condiciones en las que el modelo operará.

Figura 67. **Configuración del espectro de frecuencias**



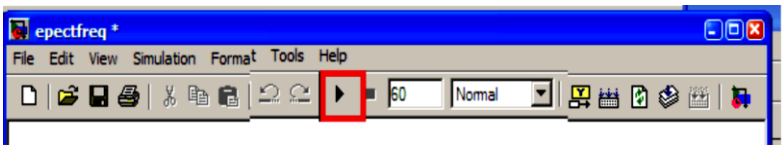
Fuente: elaboración propia, empleando Matlab.

Debe notarse que el sistema está configurado para operar en tiempo discreto, esto se hace porque la FFT es un cálculo estrictamente discreto.

4.6.7. Simulando el modelo

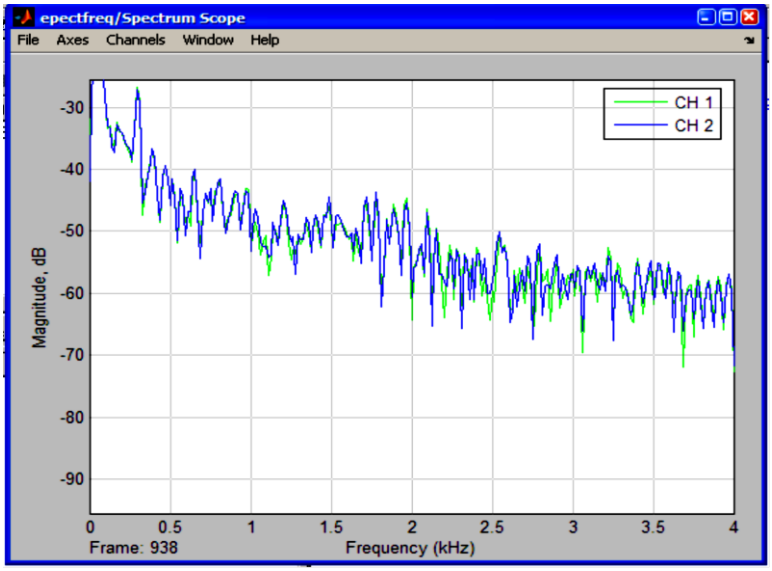
Para comenzar la simulación presionar en el botón siguiente.

Figura 68. **Para comenzar la simulación**



Fuente: elaboración propia, empleando Matlab.

Figura 69. **Espectro de frecuencias**



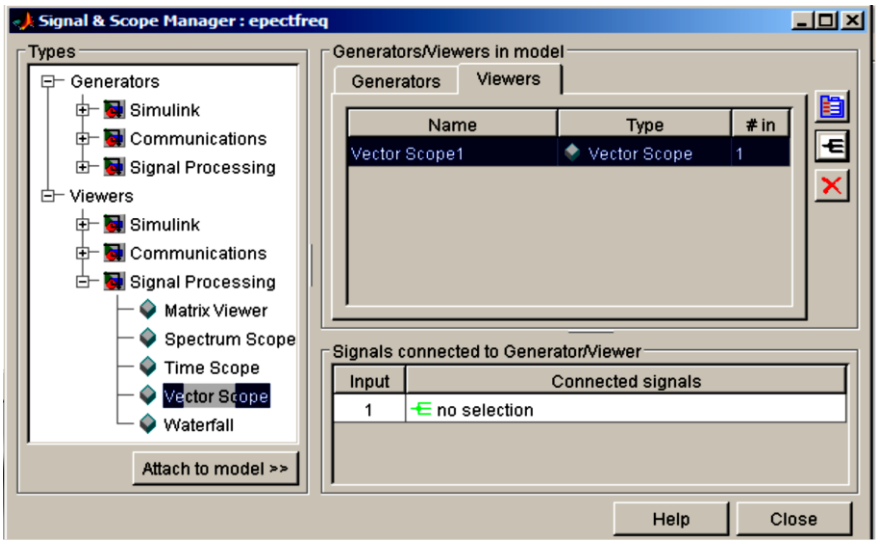
Fuente: elaboración propia, empleando Matlab.

La gráfica es el espectro de frecuencias de la señal que está captando la tarjeta de sonido de la computadora. Como se puede observar, se muestran 2 canales que representan los canales izquierdo y derecho de la entrada de micrófono. El eje x representa la frecuencia en kilohertz y el eje y representa la magnitud de la potencia en decibeles, los niveles actuales de la gráfica son demasiado bajos, en el orden de -30 db a -80 db.

Proceder a conectar el micrófono a la computadora y comienzar a hacer sonidos altos o graves y observar cómo se comporta la gráfica. Ahora se puede observar el espectro de frecuencia de la entrada y salida de los filtros digitales a implementar en futuros proyectos.

- Utilizando Matlab como osciloscopio: en el menú Tools – Signal scope & manager, en la sección de Viewers – Signal procesing – Vector scope.

Figura 70. **Signal & scope manager**

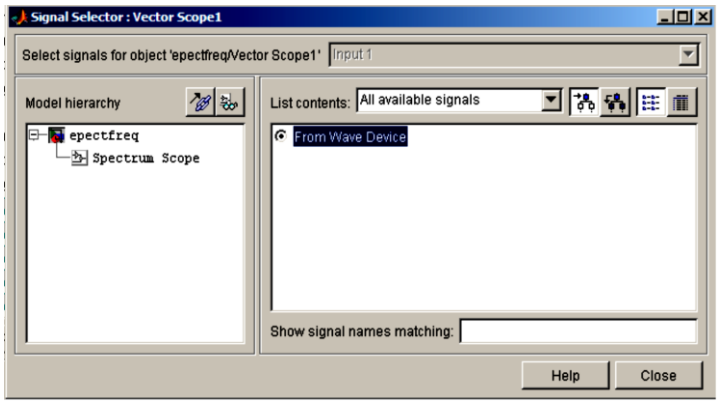


Fuente: elaboración propia, empleando Matlab.

El visualizador de vectores puede representar gráficas en dominio del tiempo, frecuencia o cualquier otro dato.

Para seleccionar la señal a graficar presione el boton  Signal selector, este le desplegará una lista de las señales en el modelo; en este caso solo existirá la señal *From wave device*.

Figura 71. **Selector de señales**

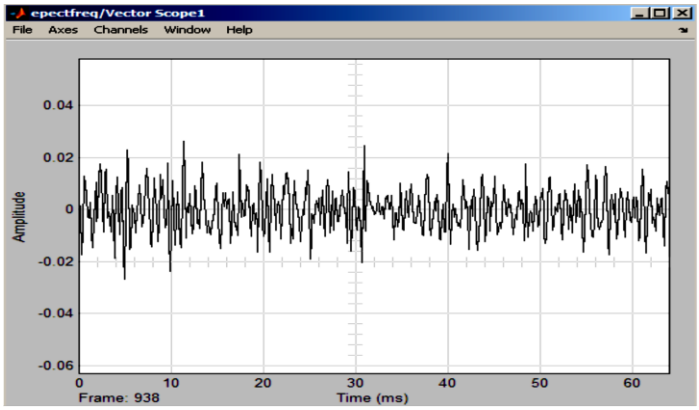


Fuente: elaboración propia, empleando Matlab.

Presionar Close para salvar la configuración del selector de señales.

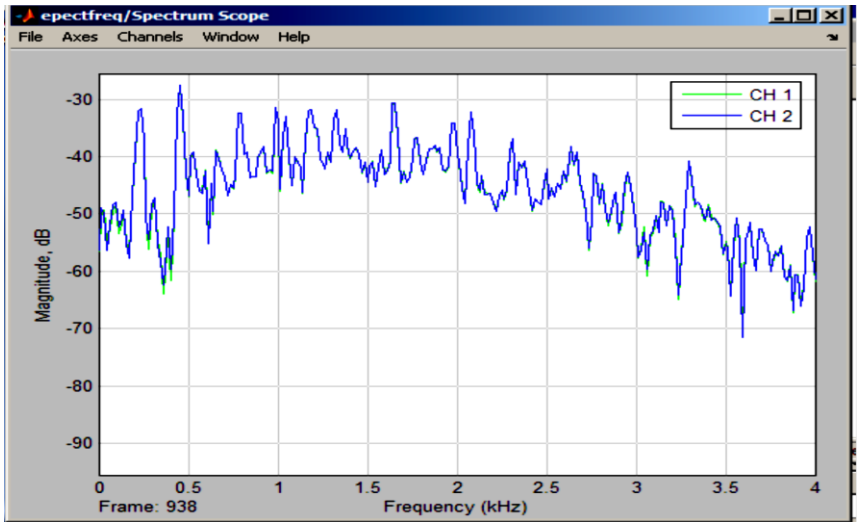
Comenzar la simulación, la gráfica del espectro de frecuencia y la amplitud de la señal de entrada en función del tiempo (osciloscopio) se estarán desplegando simultáneamente.

Figura 72. **Osciloscopio**



Fuente: elaboración propia, empleando Matlab.

Figura 73. **Espectro de frecuencia**



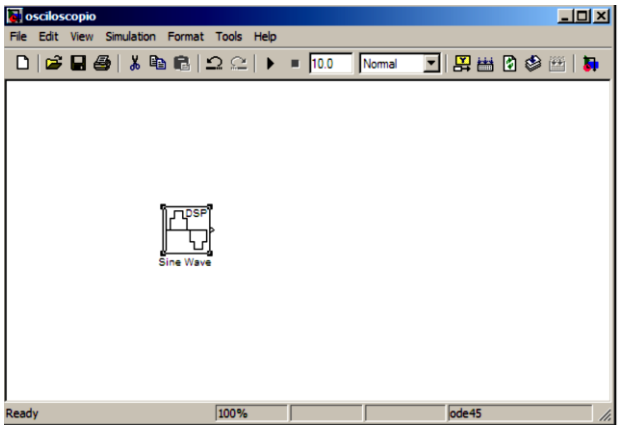
Fuente: elaboración propia, empleando Matlab.

- Generación de señales en Matlab: la generación de señales que servirá en proyectos futuros para introducir señales de prueba al Ez kit 21061 y confirmar el buen funcionamiento de los filtros elaborados.

4.6.8. Generacion de tonos

En el Signal processing blockset de Simulink se encuentra el menú Source y hay diferentes tipos de señales para generar. Seleccionar Sine wave (señal senoidal) y arrastrarlo hasta un nuevo modelo de Simulink.

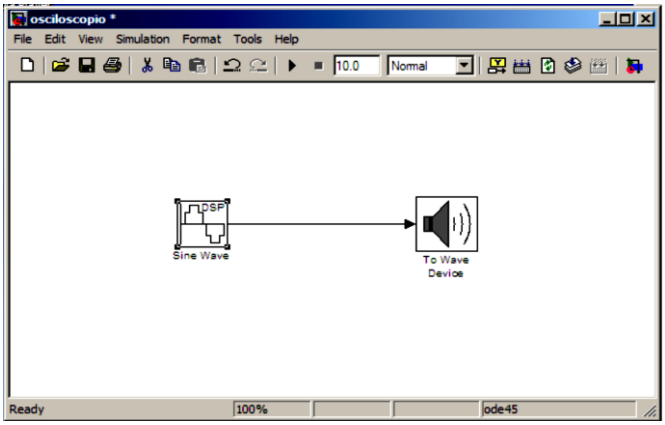
Figura 74. **Generador de onda senoidal**



Fuente: elaboración propia, empleando Matlab.

Luego, en el menú Plataforn specific I/O seleccionar *To wave device* que es la salida de la tarjeta de audio de la computadora y arrástrelo hacia el modelo.

Figura 75. **Generador de señales conectado a la salida de audio**

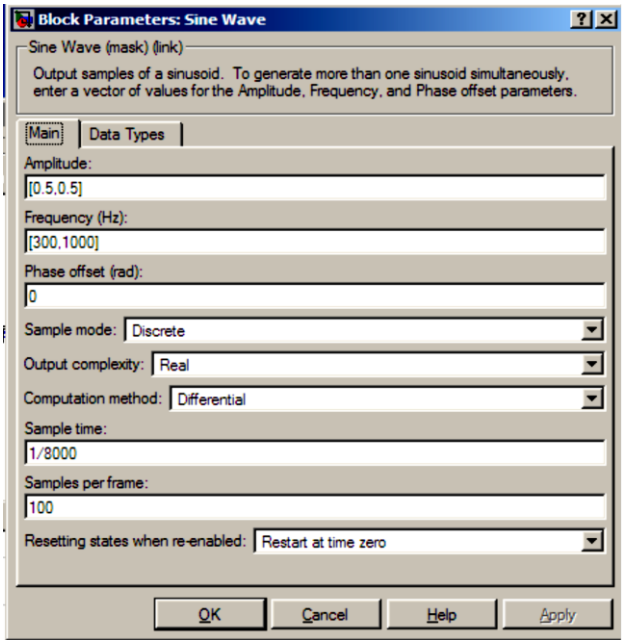


Fuente: elaboración propia, empleando Matlab.

4.6.8.1. Configuración del generador de onda senoidal

Al hacer doble clic en el generador de onda senoidal se desplegará un menú de configuración.

Figura 76. **Figura senoidal**



Fuente: elaboración propia, empleando Matlab.

La amplitud puede ser un vector de dos dimensiones, esto quiere decir que se generarán dos señales de 0,5 volts cada una, no debe excederse de 1 volt para no dañar la tarjeta de sonido.

La frecuencia en hertz puede ser un vector de uno o dos valores. El método de muestreo es discreto debido a que la tarjeta de sonido trabaja con señales discretas que son transformadas en análogas por el DAC y el filtro paso

bajo internos. El tiempo de muestreo es la frecuencia a la que trabajará el *codec* de la tarjeta de audio.

La cantidad de muestras por *frame* es la cantidad a mostrar en el visualizador.

4.6.8.2. Generando la señal senoidal

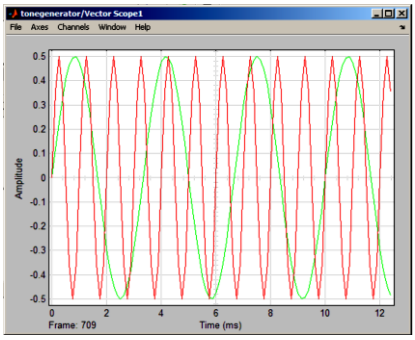
Antes de comenzar la simulación se debe configurar los parámetros de la simulación en tiempo discreto.

En la simulación se podrá escuchar dos tonos de 300 hertz y 1 000 hertz, con una amplitud de 0,5 v cada uno y modificar los valores según la necesidad de los futuros proyectos.

4.6.8.3. Visualizando la salida del generador

En el menú Tools → Signal & scope manager → Viewers, adjuntar al modelo el Vector scope, en el selector de señales escoger la señal senoidal como entrada.

Figura 77. **Vector scope**

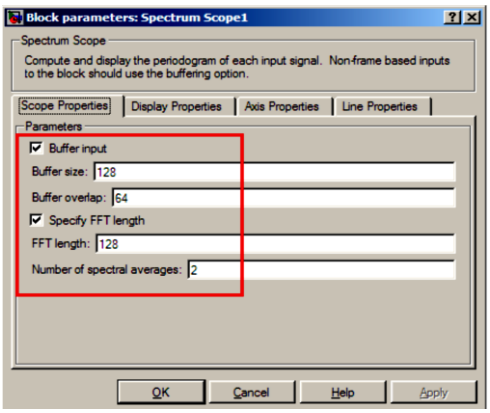


Fuente: elaboración propia, empleando Matlab.

Observar las dos frecuencias generadas: en color verde la señal de 300 hertz y en color rojo la señal de 1 000 hertz de la misma manera puede adjuntar un visualizador del espectro.

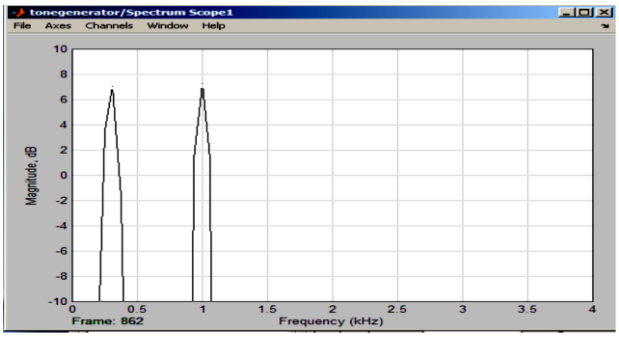
En la configuración del Spectrum scope deben habilitarse las opciones que se muestran en la figura 78.

Figura 78. **Configuración del visualizador de espectro de frecuencias**



Fuente: elaboración propia, empleando Matlab.

Figura 79. **Spectrum scope**



Fuente: elaboración propia, empleando Matlab.

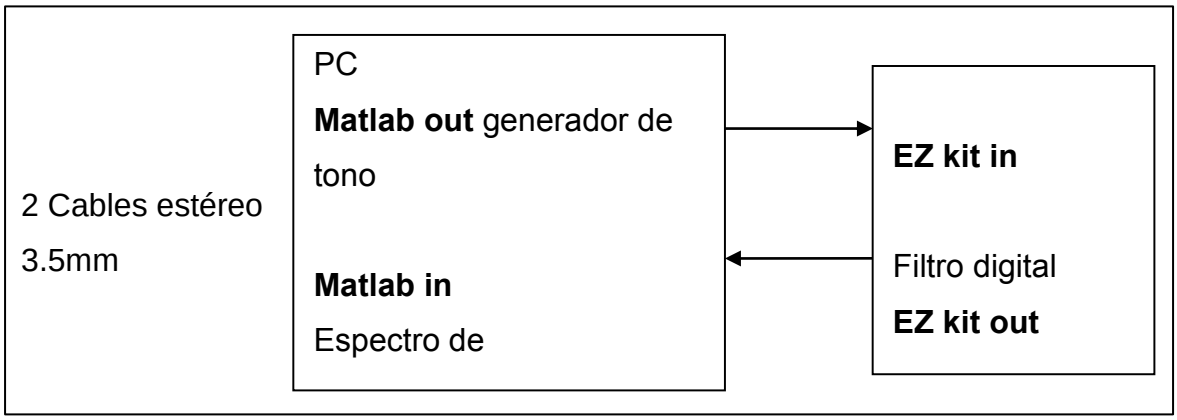
- Topologías de conexión Matlab – EZ kit

  Las topologías de conexión entre Matlab y EZ kit se basan en la generación de señales, ya sea por Matlab, micrófono, instrumentos Musicales y equipos portátiles, que serán procesados por medio de los filtros digitales realizados en la plataforma Visual DSP ++.

Matlab se usa como visualizador o generador de señales, para comprobar el funcionamiento de los filtros digitales cargados en el EZ kit.

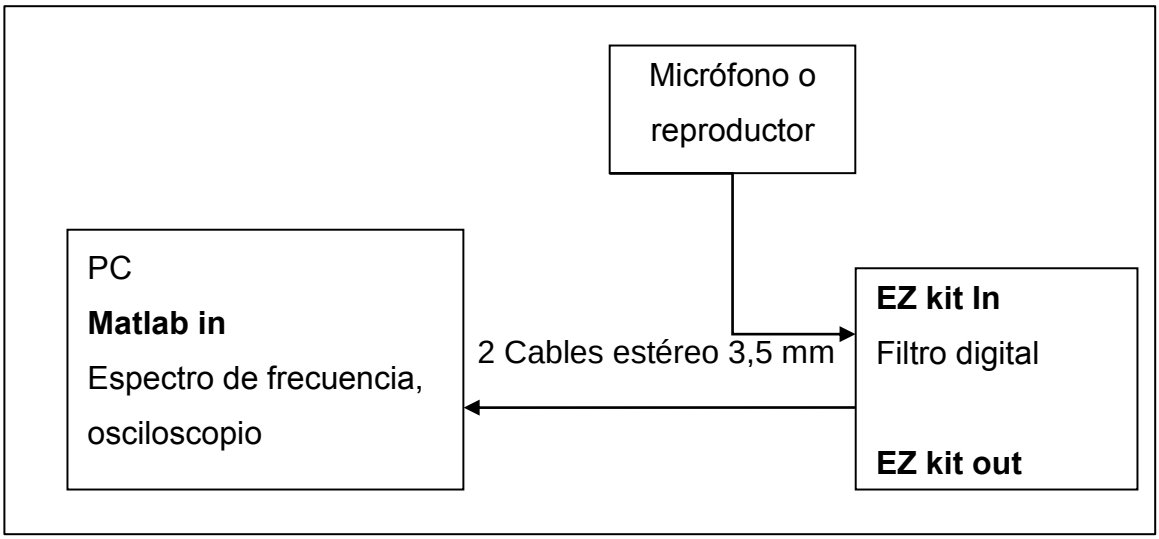
El EZ kit es capaz de generar señales, prueba de ello es la canción que se reproduce cuando se conecta. La licencia de evaluación limita la cantidad de código de programa en el procesador y por este motivo no se generarán señales desde el EZ kit.

Figura 80. **Topología cerrada**



Fuente: elaboración propia, empleando Matlab.

Figura 81. **Topología abierta**



Fuente: elaboración propia, empleando Matlab.

4.7. Efecto eco

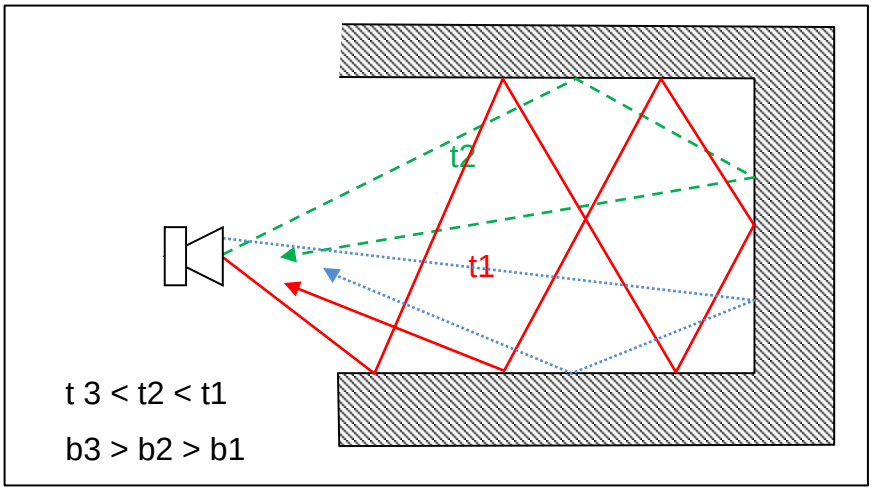
- Equipo a utilizar

    - Ez kit SHARCK 21061
    - Visual DSP++ 3,0 (software)
    - Matlab 7 (software)
    - Micrófono estándar para computadora
    - Bocinas estándar para computadora
    - Cable estéreo 3,5 mm

4.7.1. Análisis del efecto eco

Es uno de los efectos más famosos que se produce en la naturaleza, auditorios y estadios, se produce cuando las ondas sonoras chocan con obstáculos y retornan a su origen en diferentes tiempos, produciendo el efecto.

Figura 82. **Efecto eco**



Fuente: elaboración propia, empleando Matlab.

Esta físicamente comprobado que la intensidad del sonido disminuye cuando la distancia recorrida es mayor, así que asociaremos cada tiempo t1,t2,t3 con una atenuación respectiva a1, a2, a3 siendo la atenuación proporcional al tiempo de retraso.

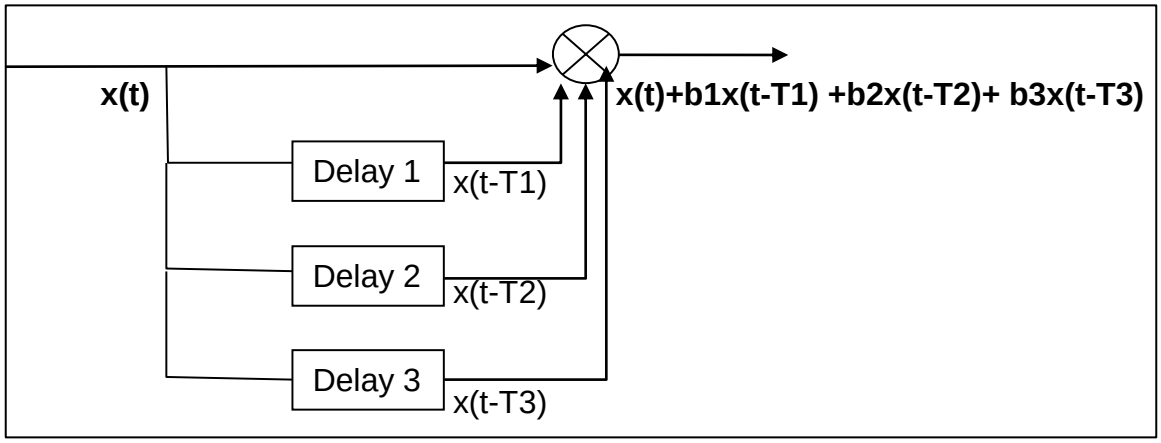
4.7.2. Diseñando el eco

Gracias al procesamiento digital es posible reproducir el efecto de forma artificial y según la necesidad es posible eliminarlo con un algoritmo llamado Eco canceling .

4.8. Concepto teórico

A partir del concepto de cómo se produce el eco, se genera el siguiente diagrama.

Figura 83. **Eco sistema analógico**



Fuente: elaboración propia, empleando Matlab.

La señal de audio es almacenada en cada *delay* para después ser reproducida a una diferencia de tiempo de la señal original, todas las señales retrasadas son sumadas creando el efecto.

Partiendo de la ecuación resultante:

$$x(t) + b_1x(t-T_1) + b_2x(t-T_2) + b_3x(t-T_3)$$

En donde las variables T_1, T_2, T_3 son el retraso en segundos, siendo y(t) la salida al sistema entonces:

$$y(t) = x(t) + b_1x(t-T_1) + b_2x(t-T_2) + b_3x(t-T_3)$$

Convirtiendo a tiempo discreto t por nTs, donde Ts es el periodo de la frecuencia de muestreo.

$$y(nTs) = x(nTs) + b_1x(nTs- T_1) + b_2x(nTs- T_2) + b_3x(nTs-T_3)$$

Ts es constante en toda la ecuación, por comodidad matemática se hace Ts = 1

$$y[n] = x[n] + b_1x[n - d_1] + b_2x[n- d_2] + b_3x[n-d_3]$$

Donde d_1,d_2 y d_3 es un entero que representa el retraso en muestras respecto a x[n], si se compara esta ecuación con la ecuación en diferencias:

$$y[n] = \sum (b_m/a_0)\, x[n-m] - \sum(a_k/a_0)\, y[n-k],$$
$$0 \le m \le M,\ 1 \le k \le N$$

Para sistemas no recursivos la ecuación es:

$$y[n] = \sum (b_m/a_0)\, x[n-m] \text{ que es similar a}$$
$$y[n] = x[n] + b_1x[n - d_1] + b_2x[n- d_2] + b_3x[n-d_3]$$

Debe tenerse en consideración que las atenuaciones b_1, b_2, b_3, y los retrasos d_1, d_2 y d_3 no son valores consecutivos, con el objetivo de reproducir el efecto eco.

El eco, en su forma más simple es aquel que solo tiene una etapa de retraso y esta distanciada de muestras de x[n], escribiendo de nuevo la ecuación en su forma general:

$$y[n] = b_0\, x[n] + b_1\, x[n-1] + b_2\, x[n-2] + \ldots\ldots + b_k\, x[n-d]$$

Haciendo:

$$b_0 = 1$$

Todos los coeficientes entre:

$$b_1 \text{ y } b_{d-1} = 0$$

Para brindar un mejor realismo, la atenuación debido a la distancia recorrida

b_d debe de ser un valor entre $0 < b_d < 1$

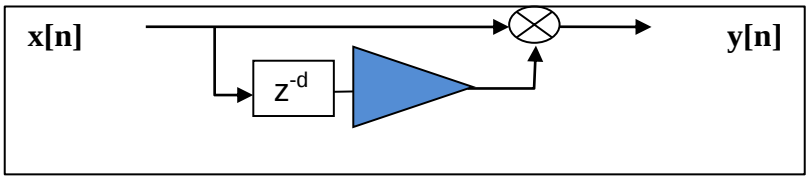
Finalmente, la ecuación resultante para el efecto eco es:

$$y[n] = x[n] + b_d x[n-d]$$

4.9. Representación gráfica del sistema

Esta es la representación gráfica del efecto eco, como se puede observar, Z^{-d} es el retraso d muestras de la señal x[n].

Figura 84. **Eco sistema digital**



Fuente: elaboración propia, empleando Matlab.

- Calculando d: es la cantidad de muestras que la señal original será retrasada; para hacer este cálculo se tiene que tener en cuenta los siguientes parámetros.

4.9.1. Frecuencia de muestreo

La frecuencia de muestreo a elegir es muy importante para el realismo del efecto, 8 kilohertz es una frecuencia de muestreo para tener un buen efecto eco de la voz humana que está en el orden de los 3 00 hertz a 3 000 kilohertz. Para música, se recomienda usar la frecuencia de muestreo más alta que es de 44,1 kilohertz.

- Tiempo continuo en segundos a retrasar la señal original

  Si el *delay* a realizar es de 1 segundo, está en función de la cantidad de muestras que se obtienen en un segundo por ejemplo:

  - Si la frecuencia de muestreo es de 8 kilohertz quiere decir que hay 8000 muestras en un segundo, si se quiere un *delay* de un segundo d debería valer 7 999, recordando que se cuenta desde el valor 0.

Se puede decir que el *delay* en segundos Ds:

$$D_S = k/fs$$

Donde fs es la frecuencia de muestreo.

- ¿Cuántos *delay* se desean agregar a la señal original? Queda a discreción del usuario la cantidad de elementos de retraso en el sistema; para mantener el realismo se recomienda no realizar más de 4 etapas de retraso.

- Eco un filtro digital

    - Topología de un filtro FIR: como se puede observar, el efecto eco puede ser realizado con un filtro FIR de un solo coeficiente b_d

$$y[n] = b_d x[n-d]$$

    - Implementación en Visual DSP ++
    - Función FIR Visual DSP++

- La sintaxis de la función FIR es la siguiente:

    #include <filters.h> float fir (float sample, const float pm coeffs[], float dm state[], int *taps*);

    Se declara la librería <*filters.h*> para utilizar la función FIR.

- Básicamente, hay tres vectores:

    - *Float sample* : guarda las muestras de entrada al filtro, el tamaño del vector es igual al número de *taps* .
    - *Float dm state[]:* lleva el puntero a la línea de retrazo como su primer elemento, el tamaño de este vector es *taps* + 1.

- *Const float pm coeffs[]*: guarda cada coeficiente en el vector *coeffs*, los coeficientes deben ser guardados en orden inverso, es decir coeff[0] = guarda el ultimo coeficiente (*taps* -1) y el coeff[*taps* -1] guarda el primer coeficiente.

- *Int samples:* el número de *taps* o tamaño del filtro.

4.9.1.1. Frecuencia de muestreo

Para determinar y modificar la frecuencia de muestreo en que operará el filtro digital se necesita conocer algunos registros de configuración del *codec* 1847 en el EZ kit.

Los valores cargados en el vector SZ_regs_1847 16

```
#define SZ_regs_1847 16    // configuración del codec1847
int regs_1847[SZ_regs_1847] = {
  0xc000,          /* index 0 - Control de entrada Izquierdo */
  0xc100,          /* index 1 - Control de entrada Derecha */
  0xc280,          /* index 2 - left aux 1 input control */
  0xc380,          /* index 3 - right aux 1 input control */
  0xc480,          /* index 4 - left aux 2 input control */
  0xc580,          /* index 5 - right aux 2 input control */
  0xc600,          /* index 6 - left dac control */
  0xc700,          /* index 7 - right dac control */
  0xc85c,           /* index 8 - Formato del dato, el valor  c  determina la
frecuencia de muestreo acorde a la tabla 2.1*/
  0xc909,          /* index 9 - configuracion de interfaz*/
  0xca00,          /* index 10 - pin control */
  0xcb00,          /* index 11 - no register */
```

```
  0xcc40,              /* index 12 - miscellaneous information */
  0xcd00,              /* index 13 - digital mix control */
  0xce00,              /* index 14 - no register */
  0x8f00};             /* index 15 - no register */
unsigned rx_buf[3];                    /* receive buffer */
unsigned tx_buf[3] = {0xcc40, 0, 0};       /* transmit buffer */
```

Tabla V. **Index 8, frecuencia de muestreo**

valor menos significativo index 8	**Frecuencia de muestreo**
0	8 Khz
1	5,5125 Khz
2	16 Khz
3	11,025 Khz
4	27,42857 Khz
5	18,9 Khz
6	32 Khz
7	12,05 Khz
8	37,8 Khz
9	N/A
A	N/A
B	44,1 Khz
C	48 Khz
D	9,6 Khz
E	6,615 Khz

Fuente: elaboración propia.

En el ejemplo de configuración, el index 8 tiene un valor de *0xc85***C**.

El valor menos significativo **C** que corresponde a una frecuencia de muestreo de 48 Khz, si el valor fuera 0xc85**0** la frecuencia de muestreo del sistema seria 8 Khz.

4.10. Programa eco Visual DSP++

El código de programa para este filtro consta de tres partes cruciales.

4.10.1. Carga de coeficientes desde un archivo *.h

Los coeficientes del filtro son cargados en el *array* coeffs[NUM_*TAPS*] de la siguiente manera:

float pm coeffs[NUM_*TAPS*] ={#include eco.h //archivo que contiene los coeficientes}.

Según la ecuación en diferencias para un filtro fir, $y[n] = b_d x[n-d]$, para reproducir el efecto eco si $b_0 = 1$ $d = 0$ entonces $y[n] = x[n]$, indicando que lo que entra es igual a la salida, si $b_d = 0.7$ $d = 600$ entonces.

$$y[n] = x[n] + 0.7x[n-600],$$ siendo equivalente a

Salida = entrada + entrada retrasada 600 muestras. Si el muestreo del sistema es de 8 000 muestras por segundo, quiere decir que el retraso en segundos es igual a 600/8000 = 0,075 s.

El archivo eco. h está estructurado de la siguiente forma:

Tabla VI. **Carga de coeficientes desde un archivo *.h**

0.0,	0.0,	0.0,	**0.7,**	0.0,	0.0,	0.0,	0.0,
0.0,	0.0,	0.0,	0.0,	0.0,	0.0,	0.0,	0.0,
0.0,	0.0,	0.0,	0.0,	0.0,	0.0,	0.0,	0.0,
0.0,	0.0,	0.0,	0.0,	0.0,	0.0,	0.0,	0.0,
0.0,	0.0,	0.0,	0.0,	0.0,	0.0,	0.0,	0.0,
0.0,	0.0,	0.0,	0.0,	0.0,	0.0,	0.0,	0.0,
0.0,	0.0,	0.0,	0.0,	0.0,	0.0,	0.0,	0.0,
0.0,	0.0,	0.0,	0.0,	0.0,	0.0,	0.0,	0.0,
....	....	....	....	....	....	....	....
....	....	....	....	....	....	....	....
....	....	....	....	....	....	....	....
0.0,	0.0,	0.0,	0.0,	0.0,	0.0,	0.0,	0.0,
0.0,	0.0,	0.0,	0.0,	0.0,	0.0,	0.0,	0.0,
0.0,	0.0,	0.0,	0.0,	0.0,	0.0,	0.0,	0.0,
0.0,	0.0,	0.0,	0.0,	0.0,	0.0,	0.0,	0.0,
0.0,	0.0,	0.0,	0.0,	0.0,	0.0,	0.0,	0.0,
0.0,	0.0,	0.0,	0.0,	0.0,	0.0,	0.0,	0.0,
0.0,	0.0,	0.0,	0.0,	0.0,	0.0,	0.0,	0.0,
0.0,	0.0,	0.0,	0.0,	0.0,	0.0,	0.0,	0.0,
0.0,	0.0,	0.0,	0.0,	0.0,	0.0,	0.0,	0.0,
0.0,	0.0,	0.0,	0.0,	0.0,	0.0,	0.0,	**1.0,**

Fuente: elaboración propia.

Donde cada coeficiente está separado por una, y el primer coeficiente b_0 es el último coeficiente de la esquina inferior izquierda, recordando que los coeficientes se cargan de forma inversa en la función FIR. Finalmente, la ecuación en diferencias será y[n] = x[n] + 0,7*x[n-597].

4.10.2. Función filtro FIR del ejemplo ECO.C

En el código de programa aparece la función FIR de la siguiente manera:

tx_buf[1] = fir(rx_buf[1], &coeffs[0], &state[0], (int)NUM_*TAPS*);//canal izquierdo.

tx_buf[2] = tx_buf[1]; //salida del canal Izquierdo igual al canal derecho.

En donde tx_buf[1] es el buffer de salida del canal izquierdo del *CODEC* y rx_buf[1] es el buffer de entrada del canal izquierdo del *CODEC*, coeffs[] guarda los coeficientes b_n de la ecuación en diferencias, state[] este array guarda los resultados de la consolación de la entrada con el filtro y finalmente NUM_*TAPS* es la cantidad de coeficientes del filtro.

4.11. Ejecutando el programa

Es la acción de iniciar la carga de un programa o de cualquier archivo ejecutable.

4.11.1. Cargando el programa ECO.C

En la carpeta C:\Program Files\Analog Devices\VisualDSP\21k\EZ-KITs\ADSP-21061\Demos\eco, están los archivos necesarios para realizar este proyecto.

- ECO.C
- ECO.H
- *Linker* files

Como se vio en el laboratorio 1 cargar los archivos y guardar el proyecto como lab3eco .

Seguidamente hacer pruebas cambiando la frecuencia de muestreo, el número de muestras de retraso en el archivo ECO.H y agregar más etapas de retraso al sistema.

4.12. Filtro FIR e IIR Visual DSP++

- Equipo a utilizar

  - EZ kit SHARCK 21061
  - Visual DSP++ 3.0 (sofware)
  - Matlab 7 (software)
  - Micrófono estándar para computadora
  - Bocinas estándar para computadora
  - Cable estéreo 3,5 mm

Laboratorio 1: filtro FIR e IIR Visual DSP ++

4.13. Concepto teórico de los filtros FIR e IIR

Hay dos clases de filtros digitales ellos son:

- FIR *finite impulse response* (respuesta finita al impulso)
- IIR *infite impulse response* (respuesta infinita al impulso)

Ambos parten de la ecuación en diferencias de los sistemas LTI, la diferencia entre ambos sistemas es la propiedad de recursividad.

$$y[n] = \sum (b_m/a_0)\, x[n-m] - \sum (a_k/a_0)\, y[n-k], \quad 0 \le m \le M, \quad 1 \le k \le N$$

$$\sum b_m\, x[n-m] = \sum a_k\, y[n-k] \quad , \quad 0 \le m \le M \quad , \quad 1 \le k \le N$$

En los filtros FIR el sistema es no recursivo:

$y[n] = \sum (b_m/a_0)\, x[n-m] \quad , \quad 0 \le m \le M$

Siendo esta ultima la ecuación básica de un filtro FIR

En los filtros IIR el sistema recursivo de orden 1:

$y[n] = \sum (b_m/a_0)\, x[n-m] - (a_m/a_0)\, y[n-1] \quad 0 \le m \le M$

$y[n] + (1/a_0)\, y[n-1] = \sum (b_m/a_0)\, x[n-m]$

En los filtros IIR la orden del filtro lo indica el retraso en la secuencia y[n-m]

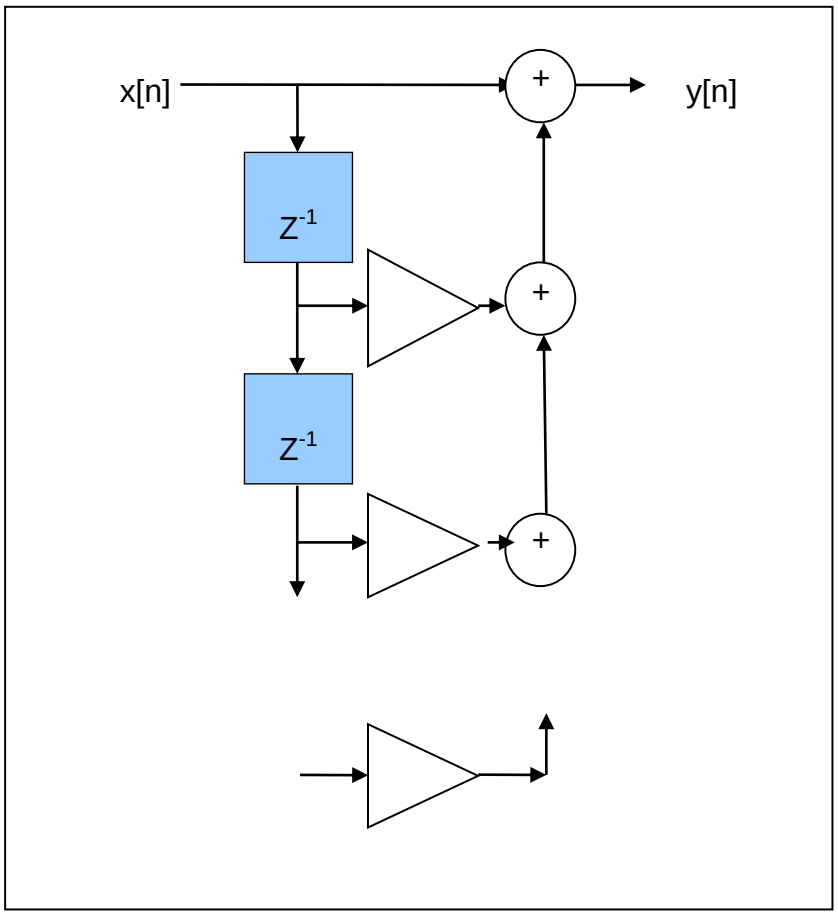
4.14. Diagrama de bloque básico de los filtros FIR e IIR

De las ecuaciones en diferencias se pueden determinar los diagramas de bloque de cada tipo de filtro.

4.14.1. Diagrama de filtro FIR

De la ecuación $y[n] = \sum (b_m/a_0)\, x[n-m]$, por comodidad $a_0 = 1$, se obtiene el siguiente diagrama sistema no recursivo

Figura 85. **Diagrama de filtro FIR**



Fuente: OPPENHEIM, Alan. *Signal and systems*. p. 153.

4.15. Diagrama de filtro IIR

Se parte de la ecuación para sistemas recursivos:

$$y[n] = \sum (b_m/a_0)\, x[n-m] - (a_m/a_0)\, y[n-1]$$

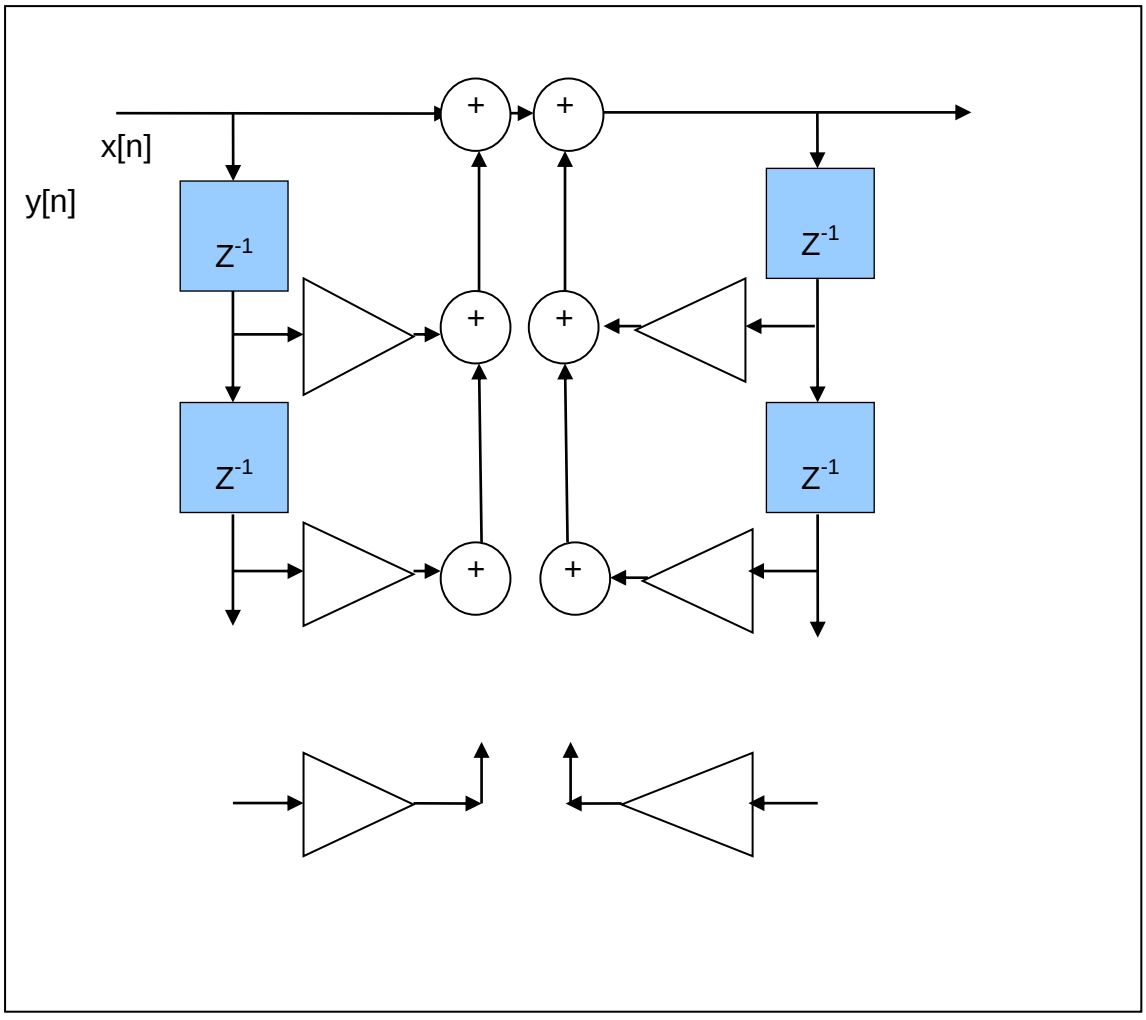
Este ejemplo de orden uno por el elemento y[n-1], se puede reescribir para cualquier orden:

$$y[n] = \sum (b_m/a_0)\, x[n-m] - (a_m/a_0)\, y[n-m]$$

por comodidad $a_0 = 1$.

Se obtiene el siguiente diagrama de un sistema recursivo:

Figura 86. **Diagrama de filtro IIR**



Fuente: OPPENHEIM, Alan. *Signal and systems*. p. 153.

4.16. Función FIR Visual DSP++

La sintaxis de la función FIR es la siguiente:

- #include <filters.h> loat fir (float sample, const float pm coeffs[], float dm state[], int *taps*);

Se declara la librería <*filters.h*> para utilizar la función *fir.*

4.16.1. Vectores de la función FIR

- *Float sample*: guarda las muestras de entrada al filtro, el tamaño del vector es igual al número de *taps* .
- *Float dm state*[]: lleva el puntero a la línea de retrazo como su primer elemento, el tamaño de este vector es *taps* + 1.
- *Const float pm coeffs*[]: guarda cada coeficiente en el vector *coeffs*, los coeficientes deben ser guardados en orden inverso, es decir coeff[0] = guarda el ultimo coeficiente (*taps* -1) y el *coeff*[*taps* -1] guarda el primer coeficiente.
- *Int samples*: el número de *taps* o tamaño del filtro.

4.17. Función IIR visual DSP++

La sintaxis de la función IIR es la siguiente:

- #include <filters.h> float iir (float sample, onst float pm a_coeffs[], const float pm b_coeffs[], float dm state[], int *taps*)

Se declara la librería <filters.h> para utilizar la función iir.

4.17.1. Vectores de la función IIR

- *Float sample* : guarda las muestras de entrada al filtro.
- *Float state*[]: lleva el puntero a la línea de retrazo como su primer elemento, el tamaño de este vector es *taps* + 1.
- *Float pm a_coeffs*[*TAPS*]; contiene los coeficientes a_m de la ecuación en diferencias.
- *Float pm b_coeffs*[*TAPS* +1]; contiene los coeficientes b_m de la ecuación en diferencias, el tamaño de este vector es *taps* + 1.
- *Int taps* : es el número de *taps* del filtro.

4.18. Transición Matlab – Visual DSP++

La mayoría de algoritmos usan la ecuación en diferencias:

$$y[n] = b_0x[n] + b_1x[n-1] + b_2x[n-2]\ldots\ldots -a_0y[n-1] - a_1y[n-2]$$

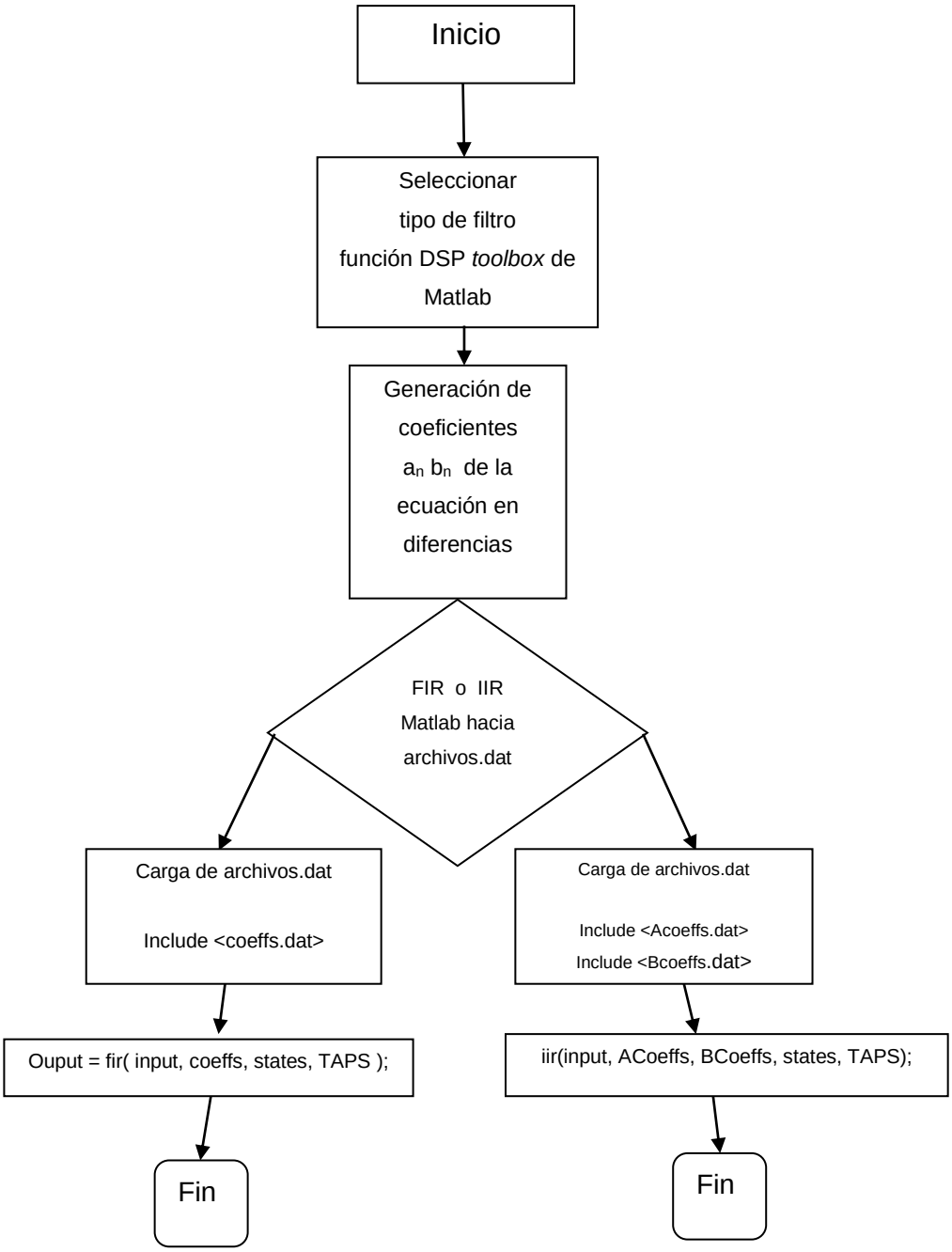
Pero las librerías C, usa el siguiente algoritmo:

$$y[n] = b_0x[n] + b_1x[n-1] + b_2x[n-2]\ldots\ldots +a_0y[n-1] + a_1y[n-2]$$

Nótese que los coeficientes a están sumados no restados.

El proceso quedará descrito por el siguiente diagrama de bloques.

Figura 87. **Diagrama Matlab -Visual DSP**



Fuente: elaboración propia.

Para demostrarlo, se usará Matlab con el *toolbox* de procesamiento digital y se diseñará un filtro *butterworth* IIR pasa banda de segundo orden (4 *taps*).

4.18.1. Copiando y pegando desde Matlab

Desde que se copia y pega desde la ventana de comandos de Matlab, es necesario desplegar los datos con la máxima precisión numérica posible.

Entonces se declara >> format long;

4.18.2. Creando un filtro *buttherworth* en Matlab

Ahora se utiliza la función Butter para generar un filtro *butterworth* de segundo orden con una región de paso entre f_1 y f_2

[b,a] = butter(2,[.075 .75]);

Para entender esto, se indica el significado de cada variable: el número 2 indica el orden del filtro, 0,075 indica la frecuencia normalizada en radianes/muestra de la primera frecuencia de corte y 0,75 es la segunda frecuencia de corte normalizada en radianes/muestra.

4.18.2.1. Frecuencia normalizada

Es aquella que depende de la frecuencia de muestreo del sistema. La variable continua en el tiempo se sustituye por *nTs*.

Supongamos que la señal de entrada es $\cos w_0 t$ y después es muestreada a un periodo de T_s, es decir que $\cos w_0$ solo existe en intervalos de nT_s haciendo la sustitución se obtiene:

$$\cos w_0 nT_s$$

$$\text{Como } w_0 = 2\pi f_0 \text{ y } T_s = 1/f_s$$

$$\cos 2\pi (f_0/f_s)\, n$$

La velocidad angular normalizada es:

$$Wn = 2\pi (f_0/f_s)$$

Por ejemplo, si f_1 es la frecuencia de corte y es igual a 300 hertz y la frecuencia de muestreo del sistema es de 8 000 hertz, quiere decir que la frecuencia normalizada es

$W_1 = 2\,\pi(300/8000) = 0{,}075\,\pi$

Para que f_2 sea la segunda frecuencia de corte de 3000hertz

$W_1 = 2\,\pi(3\,000/8\,000) = 0{,}75\,\pi$

4.18.2.2. Coeficientes a_n y b_n Matlab –> Visual DSP++

De la función Butter resultarán los vectores a_n y b_n

```
[b,a] = butter(2,[.075 .75])
```

Los coeficientes al ser cargados en las funciones IIR y FIR de Visual DSP ++, deben estar en sentido inverso, para eso se declara en Matlab:

```
b = flipud(b');
a = -flipud(a');
```

Nótese que los coeficientes del vector a_n están negados, según la ecuación en diferencias.

Los vectores contienen los siguientes coeficientes:

```
a
a =
   -0,2477
   -0,0869
    0,4681
    0,7489
   -1,0000
b
b =
    0,4749
         0
   -0,9498
         0
    0,4749
```

Los coeficientes generados en Matlab se copian al *block* de notas, colocando una coma al final de cada coefieciente, en el vector a_n el elemento -1,00000 corresponde a $a_0 = 1$ y se elimina por comodidad. Después de eso, el archivo se guarda como *acoeffs.dat* y *bcoeffs.dat* respectivamente.

4.18.3. Implementación Visual DSP++

Lo primero que se debe hacer es declarar la librería al inicio del programa. Se define la cantidad de *taps* del filtro.

#define *TAPS* 4

Se declaran las variables de las funciones a utilizar, recordar de inicializarlas al inicio de cada programa para evitar tener basura almacenada.

float dm states[*TAPS* +1]
float pm ACoeffs[*TAPS*] ={#include "acoeffs.dat"}
float pm BCoeffs[*TAPS* +1] ={#include "bcoeffs.dat }
//inicializando variablesint i
for (i=0; i<*TAPS* +1; i++) states[i] = 0,0

Finalmente, se declara la rutina IIR que deberá ejecutarse con cada muestra tomada por el ADC:

output = iir(input, ACoeffs, BCoeffs, states, *TAPS*);

CONCLUSIONES

1. En el tratamiento digital de la señal se detallan tres procesos: la señal se transforma de analógica a digital, la señal digital es procesada por un procesador DSP y la salida se transforma de digital a analógico.

2. La calidad del procesamiento del sonido está determinada por la frecuencia de muestreo en el proceso de conversión de la señal de analógico a digital y de la cantidad de bits que representan la amplitud de la señal de audio.

3. Los puertos de entrada y salida del EZ kit ADSP 21061 son para señales de audio y en conjunto con el software Visual DSP++ permiten desarrollar filtros digitales con el DSP 21061 de Analog Devices.

4. Las prácticas de laboratorio presentadas exponen el uso de las herramientas de software como Matlab y Visual DSP++ y el equipo EZ kit 21061, mostrando paso a paso la implementación de los filtros digitales básicos.

RECOMENDACIONES

1. Es importante que el estudiante considere leer la teoría expuesta en el tratamiento digital de la señal, esta pueda ser comprobada prácticamente buscando el equivalente de las funciones matemáticas en el lenguaje de programación del DSP.

2. En el proceso de conversión de analógico a digital es importante que la frecuencia de muestreo a seleccionar sea mayor a 2 veces la frecuencia máxima de la señal de audio de entrada y menor que la frecuencia de muestreo máxima del EZ kit que es de 48 Khz.

3. Para evitar dañar el EZ kit 21061 al trabajar con fuentes de audio no confiables, tomar en cuenta utilizar un osciloscopio para revisar la señal de entrada, verificando que la misma no posea nivel de decibeles ni exceda 1 voltio pico a pico.

4. Es importante que el estudiante tenga uno o varios programas base que funcionen perfectamente en el EZ kit 21061 y que partir de ellos pueda modificarlos según la aplicación que desee desarrollar. Esta práctica ahorrará tiempo de programación y evitará errores de sintaxis.

BIBLIOGRAFÍA

1. *Convertidor analógico a digital* (ADC). [en línea]. <http://www.electronics.dit.ie/staff/tscarff/DT089_Physical_Computing_1/adc/adc.htm.>. [Consulta: 11 de octubre de 2014].

2. *Datasheet ADSP 21061*. [en línea]. <http://www.analog.com/en /products /processorsdsp/Sharck/adsp21061.html#productoverview>. [Consulta: 4 de noviembre de 2014].

3. *Filtros digitale*s. [en línea]. <http://www.analog.com/media/en/training-seminars/design-handbooks/MixedSignal_Sect6.pdf>. [Consulta: 11 de octubre de 2014].

4. *Física de la comunicación.* [en línea] <http://www.gts.tsc.uvigo.es/ssd/practicas/practica4.pdf>. [Consulta: 11 de octubre de 2014].

5. *Las herramientas básicas de procesamiento de señales digitales*. [en línea]. <http://what-when-how.com/voip-protocols/the-basic-tools-of-digital-signal-processing-voip-protocols/>. [Consulta: 11 de octubre de 2014].

6. *Manual ADSP 21061*. [en línea]. <http://www.analog.com/media/en /technical-documentation/data-sheets/ADSP-21061_21061L.pdf>.

7. OPPENHEIM, Alan. WILLSKY; Alan; YOUNG Ian. *Signals and systems*. México: Prentice Hall, 1998. 956 p.

8. PAPOULIS, Athanasios; BERTRAN, Miquel. *Sistemas y circuitos digitales y analógicos.* [en línea]. <http://repositorio.uis.edu.co/jspui /bitstream/123456789 /7226/2/ 116771.pdf>. [Consulta: 7 de noviembre de 2014].

9. RONDÓN CARREÑO, Nubia Alexandra. *La transformada z y algunas aplicaciones.* [en línea]. <http://repositorio.uis.edu.co/jspui /bitstream/123456789/7226/2/116771.pdf>. [Consulta: 7 de noviembre de 2014].

.

ANEXOS

Anexo 1. Registros de control del AD1847

El AD1847 contiene seis registros de control de 16 bits y 13 registros de control de 8 bits.

La información de control es enviada al AD 1847 en los 16 bit de la palabra de control, la información del estatus es enviada a través de los 16 bits de la palabra estatus. Los datos de la captura y reproducción tienen 2 registros de 16 bits para el canal izquierdo y derecho.

Registros de 16 bits

Slot	Register Name (16-Bit)
0	Control Word Input
1	Left Playback Data Input
2	Right Playback Data Input
3	Status Word/Index Readback Output
4	Left Capture Data Output
5	Right Capture Data Output

Data 15	Data 14	Data 13	Data 12	Data 11	Data 10	Data 9	Data 8
CLOR	MCE	RREQ	res	IA3	IA2	IA1	IA0
Data 7	**Data 6**	**Data 5**	**Data 4**	**Data 3**	**Data 2**	**Data 1**	**Data 0**
DATA7	DATA6	DATA5	DATA4	DATA3	DATA2	DATA1	DATA0

- Data 0 a 7: dato a ingresar al ad1847 por el CPU o DSP que administra el *CODEC*.
- IA0 a IA3: dirección (address) a ingresar por el CPU o DSP al *CODEC*.
- RREQ: requisición de Lectura, lee el contenido de los registros.

- MCE Mode Change Enable, debe ser alto (MCE =HI) para cambiar los datos en los registros de control. Los registros de formato, los registros de información no pueden ser cambiados a menos que el bit MCE este en Alto.

Datos de captura y reproducción

Data 15	Data 14	Data 13	Data 12	Data 11	Data 10	Data 9	Data 8
DATA15	DATA14	DATA13	DATA12	DATA11	DATA10	DATA9	DATA8
Data 7	**Data 6**	**Data 5**	**Data 4**	**Data 3**	**Data 2**	**Data 1**	**Data 0**
DATA7	DATA6	DATA5	DATA4	DATA3	DATA2	DATA1	DATA0

- Data 15 a 0 : contiene el dato de 16 bits

Registros de 8 bits

Index	Register Name
0	Left Input Control
1	Right Input Control
2	Left Aux #1 Input Control
3	Right Aux #l Input Control
4	Left Aux #2 Input Control
5	Right Aux #2 Input Control
6	Left DAC Control
7	Right DAC Control
8	Data Format
9	Interface Configuration
10	Pin Control
11	Invalid Address
12	Miscellaneous Information
13	Digital Mix Control
14	Invalid Address
15	Invalid Address

- Descripción de los Registros

Left Input Control Register (Index Address 0)

IA3:0	Data 7	Data 6	Data 5	Data 4	Data 3	Data 2	Data 1	Data 0
0000	LSS1	LSS0	res	res	LIG3	LIG2	LIG1	LIG0

- LIG 3 a 0 : Left input gain. Entrada canal izquierdo, el bit menos significativo representa una ganancia de +1,5 dB. La máxima ganancia de es de +22,5dB.
- res: Reservado para futuras expansiones. Escribir ceros para todos los bits res.
- LSS 1 a 0: Left input source select. Selector de fuente de entrada del canal izquierdo.

0	Left Line 1 Source Selected
1	Left Auxiliary 1 Source Selected
2	Left Line 2 Source Selected
3	Left Line 1 Post-Mixed Output Loopback Source Selected

Right Input Control Register (Index Address 1)

IA3:0	Data 7	Data 6	Data 5	Data 4	Data 3	Data 2	Data 1	Data 0
0001	RSS1	RSS0	res	res	RIG3	RIG2	RIG1	RIG0

- RIG 3 a 0 : Right input gain. Entrada canal derecha, el bit menos significativo representa una ganancia de +1,5 dB. La máxima ganancia de es de +22,5 dB.
- res: Reservado para futuras expansiones. Escribir ceros para todos los bits res.
- RSS 1 a 0: Right input source select. Selector de fuente de entrada del canal izquierdo.
-

Left Auxiliary #1 Input Control Register (Index Address 2)

IA3:0	Data 7	Data 6	Data 5	Data 4	Data 3	Data 2	Data 1	Data 0
0010	LMX1	res	res	LX1G4	LX1G3	LX1G2	LX1G1	LX1G0

- LX1G 4 a 0: Left auxiliary Input Gain Select.
- res: reservado para futura expansión.
- LMX1 Left auxiliary #1 mute: Cuando es seteado alto la entrada es silenciada.

Right Auxiliary #1 Input Control Register (Index Address 3)

IA3:0	Data 7	Data 6	Data 5	Data 4	Data 3	Data 2	Data 1	Data 0
0011	RMX1	res	res	RX1G4	RX1G3	RX1G2	RX1G1	RX1G0

- RX1G 4 a 0: Rigth auxiliary Input gain select.
- res : reservado para futura expansión.
- RMX1 Left auxiliary #1 rigth: cuando es seteado alto la entrada es silenciada.

Left Auxiliary #2 Input Control Register (Index Address 4)

IA3:0	Data 7	Data 6	Data 5	Data 4	Data 3	Data 2	Data 1	Data 0
0100	LMX2	res	res	LX2G4	LX2G3	LX2G2	LX2G1	LX2G0

- LX2G 4 a 0: left auxiliary input gain select
- res : reservado para futura expansión.
- LMX2 Left auxiliary #2 mute: cuando es seteado alto la entrada es silenciada.

Right Auxiliary #2 Input Control Register (Index Address 5)

IA3:0	Data 7	Data 6	Data 5	Data 4	Data 3	Data 2	Data 1	Data 0
0101	RMX2	res	res	RX2G4	RX2G3	RX2G2	RX2G1	RX2G0

- RX2G 4 a 0: rigth auxiliary input gain select
- res : reservado para futura expansión.
- RMX2 Left auxiliary #2 rigth: cuando es seteado alto la entrada es silenciada.

Left DAC Control Register (Index Address 6)

IA3:0	Data 7	Data 6	Data 5	Data 4	Data 3	Data 2	Data 1	Data 0
0110	LDM	res	LDA5	LDA4	LDA3	LDA2	LDA1	LDA0

- LDA5:0 selección de atenuación del DAC derecho. El ultimo bit menos significativo representa -1,5 dB LDA:0 =0preoduce una atenuación de 0 dB. La atenuación máxima es de -94,5 dB
- Reservado para una futura expansión.
- LDM silenciar DAC derecho (lefth DAC mute). Cuando este bit es puesto en estado alto silenciara la salida del canal derecho.

Right DAC Control Register (Index Address 7)

IA3:0	Data 7	Data 6	Data 5	Data 4	Data 3	Data 2	Data 1	Data 0
0111	RDM	res	RDA5	RDA4	RDA3	RDA2	RDA1	RDA0

- RDA5:0 Selección de atenuación del DAC izquierdo. El ultimo bit menos significativo representa -1,5 dB LDA:0 = 0 produce una atenuación de 0 dB. La atenuación Máximo es de -94,5 dB
- res Reservado para una futura expansión.

Data Format Register (Index Address 8)

IA3:0	Data 7	Data 6	Data 5	Data 4	Data 3	Data 2	Data 1	Data 0
1000	res	FMT	C/L	S/M	CFS2	CFS1	CFS0	CSL

- RDM silenciar DAC izquierdo (Rigth DAC mute). Cuando este bit es puesto en estado alto silenciara la salida del canal derecho.
- EL contenido de este registro no puede ser modificado a menos que el AD1847 este en estado MCE (mode change enable).
- CSL clock source select. Este bit selecciona la fuente de reloj a ser usada para el la freucuencia de muestreo de audio. 0 = XTAL1 (24,576 Mhz), 1 = XTAL2 (16,9344 Mhz).

- CFS2:0 clock frecuency divide select. Estos bits seleccionan la tasa de muestreo de audio. La tasa de muestreo de audio depende de cual fuente de clock es seleccionada.

CFS2:0	Divide Factor	XTAL1 24.576 MHz	XTAL2 16.9344 MHz
0	3072	8.0 kHz	5.5125 kHz
1	1536	16.0 kHz	11.025 kHz
2	896	27.42857 kHz	18.9 kHz
3	768	32.0 kHz	22.05 kHz
4	448	Not Supported	37.8 kHz
5	384	Not Supported	44.1 kHz
6	512	48.0 kHz	33.075 kHz
7	2560	9.6 kHz	6.615 kHz

- S/M stereo/mono Select. Este bit determina el formato de los datos de audio. Al seleccionar estéreo los canales serán muestreados alternadamente entre los canales derecho e izquierdo. Al seleccionar monofónico reproduce la misma muestra de audio en los dos canales. Mono captura solo los datos del canal Derecho.

0 = monofónico, 1 = Estereofónico

- C/L Companded/Linear Select. Este bit selecciona entre dos tipos de representación de audio: linear y companded. EL tipo de PCM Lineal o Companded es definido por los FMT bits

0 = Linear PCM

1 = Companded

FMT Format select. Este Bit define el formato para todas las entradas o salidas de audio basadas en el estado del bit C/L

	Linear PCM (C/L = 0)	Companded (C/L = 1)
0	8-bit unsigned linear PCM	8-bit µ-law companded
1	16-bit signed linear PCM	8-bit A-law companded

Fuente: *Convertidor analógico a digital* (ADC). http://www.electronics.dit.ie /staff/tscarff/DT089_Physical_Computing_1/adc/adc.htm. Consulta: 11 de octubre de 2014].

Anexo 2. Código visual DSP

Código Visual DSP++ eco.c

```
/* Eco                                                    */
/* Tesis: Procesamiento Digital de audio                  */
/* Eduardo Alvarado                                       */
#include <def21060.h>
#include <21060.h>
#include <signal.h>
#include <sport.h>
#include <macros.h>
#include <math.h>
#include <filters.h>

/* DMA Chain pointer bit definitions */
#define CP_PCI 0x20000          /* Program-Controlled Interrupts bit */
#define CP_MAF 0x1ffff          /* Valid memory address field bits   */

#define SetIOP(addr, val)  (* (volatile int *) addr) = (val)

#define GetIOP(addr)       (* (volatile int *) addr)
```

#define NUM_*TAPS* 600 // Cantidad de Coeficientes de La ecuación en diferencias

float pm coeffs[NUM_*TAPS*] ={#include echo.h //archivo que contiene los coeficientes};

float dm state1[NUM_*TAPS* +1]; //array del canal izquierdo

float dm state2[NUM_*TAPS* +1]; //array del canal derecho

#define SZ_regs_1847 16 // configuracion del *codec*1847

int regs_1847[SZ_regs_1847] = {/* Note that the MCE bit is maintained throughout initial

programming to hold off premature autocalibration. */

0xc000, /* index 0 - Control de entrada Izquierdo */

0xc100, /* index 1 - Control de entrrada Derecha */

0xc280, /* index 2 - left aux 1 input control */

0xc380, /* index 3 - right aux 1 input control */

0xc480, /* index 4 - left aux 2 input control */

0xc580, /* index 5 - right aux 2 input control */

0xc600, /* index 6 - left dac control */

0xc700, /* index 7 - right dac control */

0xc851, /* index 8 - Formato del dato, aca se puede modificar la frecuencia de muestreo*/

0xc909, /* index 9 - configuracion de interfaz*/

0xca00, /* index 10 - pin control */

0xcb00, /* index 11 - no register */

0xcc40, /* index 12 - miscellaneous information */

0xcd00, /* index 13 - digital mix control */

0xce00, /* index 14 - no register */

0x8f00}; /* index 15 - no register */

unsigned rx_buf[3]; /* receive buffer */

```
unsigned tx_buf[3] = {0xcc40, 0, 0};         /* transmit buffer */
/* DMA chaining Transfer Control Blocks */
typedef struct {
  unsigned   lpath3;     /* for mesh mulitprocessing  */
  unsigned   lpath2;     /* for mesh multiprocessing  */
  unsigned   lpath1;     /* for mesh multiprocessing  */
  unsigned   db;         /* General purpose register  */
  unsigned   gp;         /* General purpose register  */
  unsigned** cp;         /* Chain Pointer to next TCB */
  unsigned   c;          /* Count register            */
  unsigned   im;         /* Index modifier register   */
  unsigned * ii;         /* Index register            */} _tcb;

_tcb rx_tcb = {0, 0, 0, 0, 0, 0, 3, 1, 0};      /* receive tcb */
_tcb tx_tcb = {0, 0, 0, 0, 0, 0, 3, 1, 0};      /* transmit tcb */

int cmd_blk[8];                          /* command block */

static int xmit_count;
static int * xmit_ptr;

/* Encendido y apagado de led's solo para comprobar que el programa
esta * ejecutandose */

void timer_lo_prior( int sig_num )
{sig_num=sig_num;// Toggle flag 2 LED.set_flag(SET_FLAG2,
TGL_FLAG);}
/* Escritura tipo DMA a la salida del Codec txbuf                */
void spt0_asserted( int sig_num )
```

{// Check if there are more commands left to transmit.if(xmit_count)

{// If so, put the command into the transmit buffer and update count.tx_buf[0] = *xmit_ptr++; xmit_count--;}

/* funcion fir */

void spr0_asserted(int sig_num)

{ tx_buf[1] = fir(rx_buf[1], &coeffs[0], &state1[0], (int)NUM_*TAPS*);//canal Izquierdotx_buf[2] = tx_buf[1]; //salida del canal Izquierdo igual al canal derecho}

Configuracion de *Codec* */

void setup_sports (void)

{/* configurando el puerto serial del Sharck SPORT0*//*configuracion de las comunicaciones multicanal */

sport0_iop.mtcs = 0x00070007; /* transmitir en palabras de 0,1,2,16,17,18 */

sport0_iop.mrcs = 0x00070007; /* recibir en palabras de 0,1,2,16,17,18 */

sport0_iop.mtccs = 0x00000000; /* no hacer companding en la transmision */

sport0_iop.mrccs = 0x00000000; /* no hacer compandin en la recepcion */

/* TRANSMIT CONTROL REGISTER */

/* STCTL0 <= 0x001c00f2 */

/* An alternate (and more efficient) way of doing this would be to */

/* write the 32-bit register all at once with a statement like this: */

/* SetIOP(STCTL0, 0x001c00f2); */

/* But the following is more descriptive... */

```
        sport0_iop.txc.mdf   = 1;    /* multichannel frame delay (MFD)       */
        sport0_iop.txc.schen = 1;    /* Tx DMA chaining enable               */
        sport0_iop.txc.sden  = 1;    /* Tx DMA enable                        */
        sport0_iop.txc.lafs  = 0;    /* Late TFS (alternate)                 */
        sport0_iop.txc.ltfs  = 0;    /* Active low TFS                       */
        sport0_iop.txc.ditfs = 0;    /* Data independent TFS                 */
        sport0_iop.txc.itfs  = 0;    /* Internally generated TFS             */
        sport0_iop.txc.tfsr  = 0;    /* TFS Required                         */
        sport0_iop.txc.ckre  = 0;    /* Data and FS on clock rising edge     */
        sport0_iop.txc.gclk  = 0;    /* Enable clock only during transmission*/
        sport0_iop.txc.iclk  = 0;    /* Internally generated Tx clock        */
        sport0_iop.txc.pack  = 0;    /* Unpack 32b words into two 16b tx's   */
        sport0_iop.txc.slen  = 15;   /* Data word length minus one           */
        sport0_iop.txc.sendn = 0;    /* Data word endian 1 = LSB first       */
        sport0_iop.txc.dtype                                                 =
SPORT_DTYPE_RIGHT_JUSTIFY_SIGN_EXTEND;
                                /* Data type specifier                  */
        sport0_iop.txc.spen  = 0;    /* Enable (clear for MC operation)      */

        /* RECEIVE CONTROL REGISTER */
        /* SRCTL0 <= 0x1f8c20f2     */
        sport0_iop.rxc.nch   = 31;   /* multichannel number of channels - 1  */
        sport0_iop.rxc.mce   = 1;    /* Activacion multicanal                */
        sport0_iop.rxc.spl   = 0;    /* Loop back configure (test)           */
        sport0_iop.rxc.d2dma = 0;    /* Enable 2-dimensional DMA array       */
        sport0_iop.rxc.schen = 1;    /* Rx DMA chaining enable               */
        sport0_iop.rxc.sden  = 1;    /* Rx DMA enable                        */
        sport0_iop.rxc.lafs  = 0;    /* Late RFS (alternate)                 */
        sport0_iop.rxc.ltfs  = 0;    /* Active low RFS                       */
```

```
        sport0_iop.rxc.irfs  = 0;    /* Internally generated RFS            */
        sport0_iop.rxc.rfsr  = 1;    /* RFS Required                        */
        sport0_iop.rxc.ckre  = 0;    /* Data and FS on clock rising edge    */
        sport0_iop.rxc.gclk  = 0;    /* Enable clock only during transmission*/
        sport0_iop.rxc.iclk  = 0;    /* Internally generated Rx clock       */
        sport0_iop.rxc.pack  = 0;    /* Pack two 16b rx's into 32b word     */

        sport0_iop.rxc.slen  = 15;   /* Data word length minus one          */
        sport0_iop.rxc.sendn = 0;    /* Data word endian 1 = LSB first      */
        sport0_iop.rxc.dtype                                                =
SPORT_DTYPE_RIGHT_JUSTIFY_SIGN_EXTEND;
                                /* Data type specifier                 */
        sport0_iop.rxc.spen = 0;     /* Enable (clear for MC operation)     */

        /* Enable sport0 xmit & rcv irqs (DMA enabled) */
        interrupt(SIG_SPR0I, spr0_asserted);
        interrupt(SIG_SPT0I, spt0_asserted);

        /* Set up Transmit Transfer Control Block for chained DMA */
        tx_tcb.ii = tx_buf;          /* DMA source buffer address           */
        tx_tcb.cp = &tx_tcb.ii;      /* define ptr to next TCB (point to self)   */
        SetIOP(CP2, (((int)&tx_tcb.ii) & CP_MAF) | CP_PCI);
                                /* define ptr to current TCB (kick off DMA) */
                                /* (SPORT0 transmit uses DMA ch 2)          */

        /* Set up Receive Transfer Control Block for chained DMA */
        rx_tcb.ii = rx_buf;          /* DMA destination buffer address          */
        rx_tcb.cp = &rx_tcb.ii;      /* define ptr to next TCB (point to self)   */
        SetIOP(CP0, (((int)&rx_tcb.ii) & CP_MAF) | CP_PCI);
```

```
        /* define ptr to current TCB (kick off DMA) */
        /* (SPORT0 receive uses DMA ch 0)          */}
        /*  Procedimiento que envia los                                    */
        /* comandos de configuracion al CODEC                              */
        void send_1847_config_cmds( void )
        {//Configurando el puntero y contador para transmitir comandos. xmit_ptr
= regs_1847; xmit_count = SZ_regs_1847;

          // Esperar que todos los comandos sean transmitidos
            while( xmit_count )idle();

           //Esperando que empiece la autocalibracion del AD1847
            while( !(rx_buf[0] & 0x0002) ) idle();

           //Esperar la finalizacion de la autocalibracion del AD1847
          while( rx_buf[0] & 0x0002 ) idle();return;}
        /*    Inializacion del ADSP 21061                                   */
        void init_21k( void )
        {//deshabilitando el temporizador y configurandolo para 4 Hertz.
            timer_off();
            timer_set( 10000000, 10000000 );

            //inicializando los punteros y contadores de transmision de comandos..
            xmit_count = 0;
            xmit_ptr   = regs_1847;
            // habilitando la solicitud de interrupcion.
            asm( "#include <def21060.h>" );
            asm( "bit set mode1 NESTM;"  );
            // Habilitando el temporizador(baja prioridad)de interrupcion.
```

```
    interrupt( SIG_TMZ, timer_lo_prior );

    // apagando los Flags
    set_flag( SET_FLAG2, SET_FLAG );return;}

/*  Procedimiento principal                                          */
void main ( void )
{ int i;int x; // inicializando el array state para el filtro fir for( i=0 ;
 i<NUM_TAPS +1 ; i++ ){state[i] = 0.0; }
    // inicializando algunos registros del Sharck 21061.
    init_21k();

    // enviando reset al codec.
    set_flag( SET_FLAG0, CLR_FLAG );    /* enviando reset al codec */
    for( x=0 ; x<0xffff ; x++ )         /* manteniendo el reset */
        ;
    set_flag( SET_FLAG0, SET_FLAG );    /*liberando el reset del codec */
    // llama al procedimiento del  puerto serial del Sharck.
    setup_sports();

    // enviando los comandos de configuracion al codec.
    send_1847_config_cmds();
    // encendiendo todos los Flags.
    set_flag(SET_FLAG2, CLR_FLAG);
  // encendiendo el Temporizador.
    timer_on();
  // ciclo sin fin
    for(;;)/*fin del programa echo.c                          */
```

Fuente: *Código visual DSP.*http://www.electronics.dit.ie /staff/tscarff/DT089_Physical_Computing_1/adc/adc.htm. Consulta: 11 de octubre de 2014].

Anexo 3. Instalación de Visual DPS++ 3.0 en Windows 7

- Copiar contenido de CD instalación visual DPS++ a una carpeta local
- Clic derecho en ADI/Setup.exe –>properties->Compatibility ->Activar Windows xp SP3 y “run as adminsitrator”
- Apply -> OK
- Doble clic Setup.exe
- Install wisua lDSP++

Fuente: *Código visual DSP 3.0 En Windows 7.* http://www.electronics.dit.ie /staff/tscarff/DT089_Physical_Computing_1/adc/adc.htm. Consulta: 11 de octubre de 2014].

Anexo 4. Instalación de licencia en Windows 7

- Clic derecho en Licenses/Setup.exe –>properties->Compatibility ->Activar “Windows xp SP3” y “run as adminsitrator”.
- Install Single User License.
- Introduzca serial# ubicado en el sobre que contiene el CD y listo

Fuente: *Instalación de licencia en Windows 7*. http://www.electronics.dit.ie /staff/tscarff/DT089_Physical_Computing_1/adc/adc.htm. Consulta: 11 de octubre de 2014].

Anexo 5. Instalación de EZ kit software en Windows 7

- Copiar contenido de CD instalación EZ KIT a una carpeta local
- Clic derecho en EZ-kit/Setup.exe –>properties->Compatibility ->"Activar Windows xp SP3" y "run as adminsitrator"
- Apply -> OK
- Doble clic Setup.exe
- Install EZ-kit21061

Fuente: *Instalación de EZ kit software en Windows 7.* http://www.electronics. dit.ie /staff/tscarff/DT089_Physical_Computing_1 /adc/ adc.ht. Consulta: 11 de octubre de 2014].