53

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES
CUAUTITLÁN

"DISEÑO DE PRÁCTICAS CON EL DSP TMS320C50 DE T.I"

T E S I S

QUE PARA OBTENER EL TÍTULO DE
INGENIERO MECÁNICO ELECTRICISTA

P R E S E N T A N :
JACOBO LÓPEZ SUÁREZ
RIGOBERTO VIZCAYA CÁRDENAS

ASESOR: ING. JORGE BUENDÍA GÓMEZ

CUAUTITLÁN IZCALLI, EDO. DE MÉXICO 2002

PAGINACION DISCONTINUA

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES CUAUTITLAN
UNIDAD DE LA ADMINISTRACION ESCOLAR
DEPARTAMENTO DE EXAMENES PROFESIONALES

ASUNTO: VOTOS APROBATORIOS

DEPARTAMENTO DE EXÁMENES PROFESIONALES

DR. JUAN ANTONIO MONTARAZ CRESPO
DIRECTOR DE LA FES CUAUTITLAN
P R E S E N T E

ATN: Q. Ma. del Carmen García Mijares
Jefe del Departamento de Exámenes
Profesionales de la FES Cuautitlán

Con base en el art. 28 del Reglamento General de Exámenes, nos permitimos comunicar a usted que revisamos la TESIS:

Diseño de prácticas con el DSP TMS320C50 de T.I.

que presenta el pasante: Jacobo López Suárez
con número de cuenta: 9206643-2 para obtener el título de :
Ingeniero Mecánico Electricista

Considerando que dicho trabajo reúne los requisitos necesarios para ser discutido en el EXAMEN PROFESIONAL correspondiente, otorgamos nuestro VOTO APROBATORIO.

A T E N T A M E N T E
"POR MI RAZA HABLARA EL ESPIRITU"
Cuautitlán Izcalli, Méx. a 8 de Mayo de 2002

PRESIDENTE	Ing. Jorge Buendía Gómez	
VOCAL	Ing. José Manuel Medina Monroy	
SECRETARIO	Ing. Juan Antonio Preciado Valtierra	
PRIMER SUPLENTE	Ing. Jorge Ramírez Rodríguez	
SEGUNDO SUPLENTE	Ing. Rodolfo López González	

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES CUAUTITLAN
UNIDAD DE LA ADMINISTRACION ESCOLAR
DEPARTAMENTO DE EXAMENES PROFESIONALES

ASUNTO: VOTOS APROBATORIOS

U. N. A. M.
FACULTAD DE ESTUDIOS
SUPERIORES-CUAUTITLAN

DEPARTAMENTO DE
EXAMENES PROFESIONALES

DR. JUAN ANTONIO MONTARAZ CRESPO
DIRECTOR DE LA FES CUAUTITLAN
P R E S E N T E

ATN: Q. Ma. del Carmen García Mijares
Jefe del Departamento de Exámenes
Profesionales de la FES Cuautitlán

Con base en el art. 28 del Reglamento General de Exámenes, nos permitimos comunicar a usted que revisamos la TESIS:

Diseño de prácticas con el DSP TMS320C50 de T.I.

que presenta el pasante: Rigoberto Vizcaya Cárdenas
con número de cuenta: 9202070-6 para obtener el título de :
Ingeniero Mecánico Electricista

Considerando que dicho trabajo reúne los requisitos necesarios para ser discutido en el EXAMEN PROFESIONAL correspondiente, otorgamos nuestro VOTO APROBATORIO.

A T E N T A M E N T E
"POR MI RAZA HABLARA EL ESPIRITU"
Cuautitlán Izcalli, Méx. a 8 de Mayo de 2002

PRESIDENTE	Ing. Jorge Buendía Gómez	
VOCAL	Ing. José Manuel Medina Monroy	
SECRETARIO	Ing. Juan Antonio Preciado Valtierra	
PRIMER SUPLENTE	Ing. Jorge Ramírez Rodríguez	
SEGUNDO SUPLENTE	Ing. Rodolfo López González	

Agradecimientos:

Quiero agradecer a mis padres José Luis López Hernández y Juventina Suárez por haberme apoyado durante mis estudios profesionales, a nuestro asesor Ing. Jorge Buendía Gómez por aceptar dirigir este trabajo de tesis, también a los pocos buenos profesores de la FES Cuautitlán de los que aprendí mucho y a Dios por dejarme vivir esta maravillosa pesadilla llamada vida.

A: *José Luis López Hernández*

Juventina Suárez López

Aurora Noriega

Mis hermanos y amigos

De todos he aprendido.

Jacobo López Suárez.

Agradezco a Dios por haberme permitido formar parte de este mundo y darme una familia maravillosa a la que quiero mucho.

A mis padres de quines me siento muy orgulloso y me han apoyado siempre, sin importar los sacrificios por los que haya que pasar.

A la Universidad Nacional Autónoma de México por haberme abierto sus puertas y darme la formación recibida.

A los profesores de la Facultad de Estudios Superiores Cuautitlán, de quienes herede una de las cosas más codiciadas, el conocimiento.

Al Ing. Jorge Buendía Gómez por haber aceptado dirigir este trabajo de tesis y por sus acertados puntos de vista.

A toda mi familia y amigos con quienes siempre he contado.

Rigoberto Vizcaya Cárdenas

Índice

Página

Introducción

En la vida real, casi todo proceso se encuentra gobernado por señales y sistemas, como es sabido, estas señales se encuentran normalmente en forma analógica. En las últimas décadas, la tendencia ha sido reemplazar el procesamiento analógico por las técnicas de procesamiento digital debido a que en la actualidad existen dispositivos programables de alta velocidad, capaces de realizar los algoritmos necesarios para procesar las señales en forma digital. Un proceso digital posee ventajas sobre los procesos analógicos. En general, si se desea cambiar las características de algún sistema procesado en forma digital sólo es necesario modificar el diseño vía software, en cambio, en procesos analógicos, esto implica cambiar muchos de los componentes del sistema, por lo tanto, un sistema digital es más flexible que un analógico. Un sistema digital es menos sensible a variables externas (temperatura, humedad, ruido, etc.). Otra ventaja es que la vida útil de los circuitos digitales es más larga que los analógicos.

Se podría pensar que una de las desventajas de usar procesamiento digital es la de no poder manejar señales con gran ancho de banda ya que estas señales requieren velocidades de muestreo muy altas para poder digitalizar las señales, sin embargo, con el gran auge que están teniendo los circuitos integrados, este problema tiende a desaparecer. Entonces, cada que sea posible elegir entre un sistema analógico y uno digital, es preferible el digital.

Uno de nuestros objetivos en esta tesis es la introducción al conocimiento de los procesadores digitales de señales (DSP's) ya que en la actualidad, aunque muchos dispositivos que usamos en la vida cotidiana tienen fundamentos en los DSP's, en la carrera de Ingeniería Mecánica Eléctrica de FES-Cuautitlán no se ha abordado este tema. Se pretende que ese conocimiento se adquiera tanto teórico como práctico, y para eso diseñaremos e implementaremos prácticas de procesamiento digital de señales usando un DSP, tanto en tiempo real como en tiempo congelado. El DSP que usaremos es el TMS320C50 de Texas Instruments. Seleccionamos este dispositivo debido a su costo y disponibilidad en el mercado. Otra razón, es que este dispositivo cumple con requisitos necesarios para nuestro propósito de realización de prácticas, estos requerimientos son; velocidad de conversión analógico-digital y digital-analógico adecuada, instrucciones y arquitectura dedicadas al procesamiento de señales en forma eficiente (consumo bajo de potencia y velocidad en los cálculos), lo cuál es suficiente para nuestro propósito.

¿Por qué usar un DSP? Como los fundamentos del procesamiento digital de señales se basan en operaciones aritméticas con números digitales, esto es 0's y 1's, este procesamiento se puede hacer "*teóricamente*" con cualquier dispositivo que trabaje con números binarios, como microprocesadores, microcontroladores, FPGA's, ASIC's, etc., pero estos dispositivos no son tan eficientes para el procesamiento de señales en tiempo real como lo son los DSP. Por ejemplo, como se verá en más adelante, un DSP está diseñado para hacer operaciones de multiplicación y acumulación de una forma muy eficiente, (en un solo ciclo de instrucción); característica que no posee un microcontrolador o microprocesador. Estos últimos necesitarían de varios ciclos de instrucción para poder realizar las operaciones adecuadas (multiplicaciones y acumulaciones eficientes), aunque debido a la velocidad que poseen algunos de ellos, esas instrucciones se podrían ejecutar en tiempo real, se desaprovecharían muchas de las características de estos dispositivos ya que estos son diseñados para otros fines. Con un FPGA se podrían implementar las funciones de un DSP, pero los FPGA usualmente disipan más potencia y son más caros que un DSP, aunque para ciertas aplicaciones, estos dos dispositivos se usan en conjunto para generar una mayor flexibilidad, estas aplicaciones no entrarán en nuestro estudio; un ASIC no resulta tan flexible como un DSP ya que son de aplicaciones específicas, no permiten muchos cambios en su programación y normalmente se usan como interfaces u otras aplicaciones. Entonces, el dispositivo más apropiado para el procesamiento de señales en forma digital, es el DSP.

Para el desarrollo de las prácticas en tiempo real, se pretenden implementar, entre otras, filtros digitales, ya que el filtrado de señales es uno de los temas esenciales en el procesamiento de señales. Para ello, la tesis se divide en cinco capítulos, que son:

CAPITULO I. En este capítulo se presenta una introducción a los filtros digitales. La finalidad de este capítulo es entender la teoría básica de filtros digitales, para finalmente llegar a la obtención de *ecuaciones en diferencias*, las cuáles describen a los filtros digitales. Para obtener estas ecuaciones en diferencias se hace uso de algunas herramientas y conceptos como la transformada z y transformada z inversa, convolución, entre otras. Una vez obtenida una ecuación en diferencias para un diseño determinado, lo que se hace es programar esa ecuación en el DSP y probar su funcionamiento de acuerdo al diseño realizado. Aquí se mencionan los dos tipos de filtros digitales que existen, que son los de respuesta finita al impulso (FIR) y los de respuesta infinita el impulso (IIR). El filtrado de

señales es uno de los temas fundamentales en el procesamiento de señales, esa es la razón de incluir este tema como primer capítulo.

CAPITULO II. Aquí se hace una introducción a los procesadores digitales de señales, para poder entender la teoría básica sobre estos dispositivos. Se presentan las principales características de estos dispositivos, como funcionan, arquitecturas, funciones que realizan, entre otras. Además se presentan las principales ventajas sobre otros dispositivos programables vía software. También se presentan las principales aplicaciones de los DSP's.

CAPITULO III. Aquí presentamos la arquitectura del DSP TMS320C50 de Texas Instruments, que es el dispositivo que finalmente usamos para implementar las prácticas. Es importante saber como está estructurado internamente el DSP, estos es; conocer sus registros, memoria, puertos, etc. También se muestran las diferentes formas en que se puede direccionar la memoria del DSP, que son, el direccionamiento corto y largo, direccionamiento directo e indirecto y direccionamiento circular. Se da una descripción de las interrupciones que posee el dispositivo. También se describen sus unidades de procesamiento numérico.

CAPITULO IV. Aquí se presenta la información básica del kit de desarrollo para el TMS320C50, que es la tarjeta sobre la cuál se encuentra montado el DSP en cuestión. Se presenta la descripción de los elementos que constituyen la tarjeta, que son;

- TLC32040 que es el convertidor analógico-digital y digital-analógico
- Un TMS320C50 que es el DSP
- Memoria PROM de 32 k para el programa kernel.
- 1 reloj a 50 Mhz.

También se presenta la información de cómo crear un programa, depurarlo, ejecutarlo, etc.

CAPITULO V. Finalmente, reuniendo la información de los capítulos anteriores, se procede a implementar las prácticas. Para ello se inicia con un programa de demostración del uso de las principales instrucciones para el dispositivo, depuración de programas, uso de comentarios, etc. para entender el funcionamiento del TMS320C50. Después se presenta la forma de cómo implementar los filtros digitales. Se presentan por separado los filtros de tipo IIR y FIR. Es importante mencionar que los programas que se presentan funcionan para cualquier filtro que se quiera implementar, sólo es

necesario cambiar los coeficientes de su ecuación en diferencias de acuerdo al diseño del filtro deseado y el resto del programa será el mismo, esa es una de las grandes ventajas de los filtros digitales sobre los analógicos, ya que en los digitales sólo es necesario cambiar los coeficientes en el programa y se obtiene el cambio deseado. Por último se presenta un ejemplo sencillo de cómo realizar un modulador en amplitud, en este caso, la señal portadora se genera con el DSP y la señal de información se toma de un generador de señales.

Finalmente se anexan tres apéndices, los cuáles contienen:

Apéndice A. Uso de las instrucciones de MATLAB para el procesamiento de señales en forma digital, que son útiles para calcular y simular filtros digitales.

Apéndice B. Se presenta una lista de las principales instrucciones usadas por el TMS320C50 y que es lo que hace cada una de ellas.

Apéndice C. Datos técnicos del TMS320C50.

La finalidad del presente trabajo es que pueda servir como punto de inicio para la introducción a los DSP en la carrera de Ingeniería Mecánica Eléctrica, ya que siendo este un tema importante no se encuentra en el plan de estudio de la carrera. Podría servir como complemento de la materia de procesamiento digital de señales, ya que en esta se imparte la teoría, pero no la práctica.

Filtros Digitales

1.1. Introducción a los filtros digitales

En procesamiento de señales, la finalidad de un filtro es remover las señales no deseadas, tales como el ruido, o extraer las partes de la señal deseadas, tales como ciertas frecuencias de interés. La figura 1.1 muestra la idea anterior:

Figura 1.1 .Diagrama de bloques de un filtro.

Podemos clasificar los filtros en dos grandes grupos, analógicos y digitales;

Filtros Analógicos. Estos filtros se diseñan con componentes eléctricos analógicos, tales como resistencias, capacitares e inductores.

Filtros Digitales. Estos filtros usan un procesador digital para desarrollar cálculos numéricos con las muestras de una señal y algunas constantes. Para la implementación de un filtro digital se puede usar una PC o bien, como se hará en el presente trabajo, usando un procesador digital de señales (DSP).

Como las señales normalmente se encuentran en formato analógico, lo que debemos de hacer para procesarlas digitalmente, es pasarla por un convertidor analógico-digital para obtener la señal en forma binaria, se procesa la señal, luego se pasa por un convertidor digital-analógico y finalmente se obtiene la señal en forma analógica ya procesada. La figura 1.2 muestra la idea:

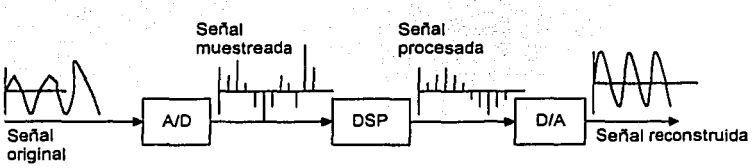


Figura 1.2. Proceso digital

Ventajas del procesamiento digital sobre el analógico

- La programación de un sistema digital permite más flexibilidad en la reconfiguración del sistema, simplemente un cambio del programa permite cambiar el algoritmo o la aplicación, en cambio en los sistemas analógicos, el cambiar las características del sistema, implica cambiar muchos de los componentes del sistema, lo a parte de ser más caro, no es flexible ya que se pierde mucho tiempo en hacer los cambios.
- El procesamiento digital de señales provee un mejor control de los requerimientos de precisión, ya que se trabaja con número binarios. Debido a ala tolerancia de los circuitos analógicos es muy difícil el control de precisión.
- Una señal digital es muy fácil almacenarla sin deterioro o pérdida de fidelidad, las señales almacenadas son fácilmente transportables para ser analizadas y/o procesadas remotamente, se puede grabar la historia.
- Una limitación podría ser la velocidad de operación de un convertidor A/D. Las señales con un gran ancho de banda requieren velocidades de muestreo y conversión muy altas, sin embargo, con el avance de la tecnología actual, esta limitación tiende a desaparecer.

La figura 1.3 muestra la idea básica de un filtro digital;

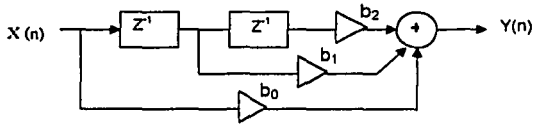


Figura 1.3 Filtro digital

De la figura se observa lo siguiente;

$$y(n) = b_0 x(n) + b_1 x(n-1) + b_2 x(n-2) \ldots\ldots\ldots (1)$$

A la ecuación anterior se le conoce como ecuación en diferencias y nos describe la salida de un filtro digital de segundo orden, donde;

x(n) es la muestra actual de la señal que se quiere procesar.

x(n-1) es la muestra anterior de la señal que se quiere procesar, representada por z^{-1}.

x(n-2) es la muestra anterior a x(n-1) de la señal que se quiere procesar, que en la figura se representa pro z^{-2}.

b_0, b_1, y b_2 son coeficientes por los que se multiplican cada una de las muestras para obtener la salida del filtro.

y(n) es la salida actual del filtro en forma digital.

Se dice que el filtro es de orden 2, porque tiene dos elementos de retardo, x(n-1) y x(n-2). Ese tipo de filtro es llamado *no recursivo* porque la salida depende sólo de la entrada y sus retardos, en la mayoría de las bibliografías a este tipo de filtros se les nombran *de respuesta finita al impulso (FIR)*. Existe otro tipo de filtros en el cuál la salida depende de la entrada y sus retardos, y a demás de la salida y sus retardos, esto es;

$$y(n) = b_0 x(n) + b_1 x(n-1) + b_2 x(n-2) - a_1 y(n-1) - a_2 y(n-2) \ldots\ldots\ldots (2)$$

A estos filtros se les llama *recursivos*, porque la actual salida depende tanto de la entrada actual, las entradas anteriores x(n), x(n-1) y x(n-2); y además de las salidas anteriores y(n-1) y y(n-2). En los libros de texto, a estos filtros se les conoce como *filtros de respuesta infinita al impulso (IIR)*.

En el presente trabajo de tesis se trabajará con señales lineales e invariantes en el tiempo, las cuáles se discretizarán para poder procesarlas digitalmente, a partir del tiempo cero; a estas señales se les conoce como sistemas discretos lineales invariantes en el tiempo y causales (SLITD), en la mayoría de la bibliografía sobre señales y sistemas.

Un (SLITD) es un conjunto de operaciones o transformaciones matemáticas que se aplican a una o varias entradas para producir una o varias salidas discretas. En nuestro caso se trabajará con sistemas "SISO", simple entrada, simple salida. Estos sistemas están descritos por alguna de las siguientes formas, que son equivalentes:

- Ecuaciones en diferencias.
- Respuestas al impulso del sistema.
- Función de transferencia.

1.2. Herramientas de procesamiento digital de señales

Para poder entender la teoria de filtros digitales, se hace uso de las siguientes herramientas;

1.2.1. Transformada Z

La transformada Z sigue siendo una invaluable herramienta para la representación, análisis y diseño de sistemas y señales en tiempo discreto. Las aplicaciones incluyen el uso de la transformada Z para describir señales y sistemas en tiempo discreto, en forma tal que podemos inferir su grado de estabilidad y visualizar sus respuestas en frecuencia, el análisis de la cuantificación de errores en filtros digitales, etc.

La transformada Z de una secuencia x(n), la cuál es valida para toda n, es definida como;

$$X(z) = \sum_{n=-\infty}^{\infty} x(n) Z^{-n} \text{.....(3)}$$

Donde Z es una variable compleja.

Como en el presente trabajo, y en la mayoría de las ocasiones en la vida real se trabaja sólo con señales causales, que están definidas a partir de cero; la transformada Z queda;

$$X(z) = \sum_{n=0}^{\infty} x(n)Z^{-n} \text{.....(4)}$$

A la que también se le conoce como transformada Z unilateral. Se puede inferir que para sistemas causales de duración finita, la transformada Z converge en cualquier punto, excepto cuando Z = 0.

Una de las propiedades muy importantes de la transformada Z, al menos para nuestro caso, es la del retardo o corrimiento. Esta propiedad es ampliamente usada en la conversión de funciones de transferencia de sistemas discretos en el tiempo al dominio de las ecuaciones en diferencias y viceversa. Si la transformada Z de una secuencia x(n) es X(z), entonces, la transformada Z de la secuencia retardada por *m* muestras es $Z^{-m}X(z)$. Esto es;

$$x(n) \longrightarrow X(z)$$
$$x(n-m) \longrightarrow Z^{-m}$$

Las ecuaciones en diferencias especifican las operaciones que debemos desarrollar por sistemas discretos en el tiempo en la entrada de los datos, en el dominio del tiempo, en orden para generar la salida deseada. Las ecuaciones en diferencias para la mayoría de los casos de interés; se escriben como:

$$y(n) = \sum_{i=0}^{N} b_i x(n-i) - \sum_{i=1}^{M} a_i y(n-i) \text{.....(5)}$$

Donde x(n) es la entrada muestreada, y(n) es la salida muestreada, y(n-i) es la salida previa y a_i y b_i son los coeficientes del sistema. La ecuación (5) indica que la salida actual y(n), es obtenida de la presente y anterior entradas muestreadas y las salidas previas y(n-i).

La ecuación en diferencias anterior "como se menciono anteriormente", representa los sistemas de respuesta infinita al impulso IIR para sistemas SLTID (lineales e invariantes en el tiempo y causales).

Cuando en la ecuación en diferencias anterior los coeficientes b_i son igual a cero, es decir, no existe realimentación, la ecuación queda de la siguiente forma;

$$y(n) = \sum_{i=0}^{N} b_i x(n-i) \text{.....(6)}$$

La cuál nos representa a los sistemas de respuesta finita al impulso FIR, para sistemas SLTID, donde la longitud de b_i es finita.

Una de las aplicaciones más importantes de la transformada Z en el procesamiento digital de señales es el diseño y análisis de errores en filtros digitales, especialmente en IIR. Es usada para calcular los coeficientes de los filtros y para analizar los efectos de la cuantificación de errores en el desarrollo del filtro digital.

Las ecuaciones en diferencias para sistemas de tiempo discreto son obtenidas de sus funciones de transferencia y viceversa usando la propiedad de retardo de la transformada Z. Aplicando esta propiedad, la ecuación en diferencias puede ser escrita como:

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{i=0}^{N} b_i Z^{-i}}{1 + \sum_{i=0}^{M} a_i Z^{-i}} \text{.....(7)}$$

La ecuación (7) nos representa la función de transferencia en el dominio de Z del sistema discreto H(z). En este caso los coeficientes a_i y b_i representan la respuesta al impulso del sistema. Aplicando la *transformada Z inversa ITZ* se puede volver ala ecuación en diferencias anterior.

1.2.2. Transformada Z inversa

La transformada Z inversa permite recobrar una secuencia en el tiempo discreto x(n). La ITZ es particularmente usada en DSP por ejemplo, para encontrar la respuesta al impulso de filtros digitales. La ITZ se define como:

$$x(n) = Z^{-1}[X(z)] \ldots\ldots\ldots (8)$$

Donde X(z) es la transformada Z de x(n) y Z^{-1} es el símbolo para transformada Z inversa. Para un sistema causal FIR, la transformada Z que lo representa es;

$$X(z) = \sum_{n=0}^{\infty} x(n)Z^{-n} = x(0) + x(1)Z^{-1} + x(2)Z^{-2} + x(3)Z^{-3} + \ldots\ldots\ldots (9)$$

Donde se ve que los valores de x(n) son los coeficientes de Z^{-n} para (n = 0, 1, 2, 3,.....). Luego aplicando la transformada Z inversa, re regresa a la ecuación en diferencias (6) que nos define el sistema en el dominio del tiempo.

1.2.3. Convolución

La convolución es una de las operaciones más comunes en el procesamiento digital de señales. Es la operación básica del filtrado digital. Dadas dos secuencias causales de longitud finita, x(n) y h(n), de longitudes N1 y N2, su convolución es definida como;

$$y(n) = h(n) \otimes x(n) = \sum_{k=0}^{\infty} h(k)x(n-k) \ldots\ldots\ldots (10)$$

$$n = 0, 1, 2, 3, \ldots\ldots\ldots (M-1)$$

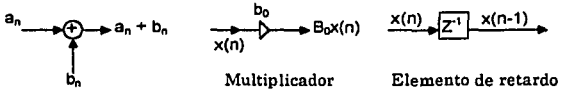
Donde M = N1 + N2-1. Los DSP están diseñados para realizar operaciones de multiplicación-acumulación eficientemente, por lo tanto es un dispositivo apropiado para desarrollar convoluciones de señales en tiempo real. El significado de la convolución es más claro cuando se observa en el dominio de la frecuencia, la convolución en el dominio del tiempo es equivalente a la multiplicación en el dominio de la frecuencia.

Como ya se mencionó, una clase importante de filtros digitales está definida por la ecuación en diferencias;

$$y(n) = \sum_{k=0}^{N-1} h(k)x(n-k) \quad \text{.....(1 1)}$$

Donde h(k), k = 0, 1, 2, 3, ..., N-1, son los coeficientes del filtro, x(n) y y(n) son la entrada y salida del filtro respectivamente. Se observa que el filtrado es de hecho la operación de la convolución de la señal a filtrar x(n) con su respuesta al impulso en el dominio del tiempo h(k).

Para poder representar los filtros digitales se hace uso de las siguientes herramientas;



De tal forma, que un filtro digital de segundo orden recursivo (IIR) lo podemos representar como en la figura 1.4.

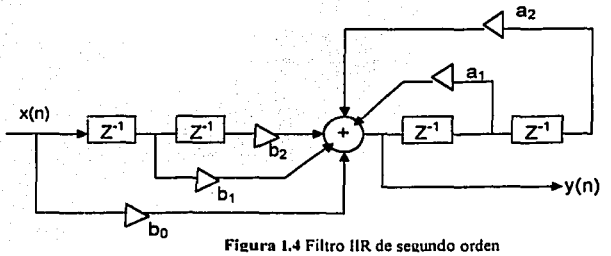


Figura 1.4 Filtro IIR de segundo orden

1.3. Procedimiento general para el diseño de filtros.

1) Dar las especificaciones del filtro, para el caso de un pasa bajas;

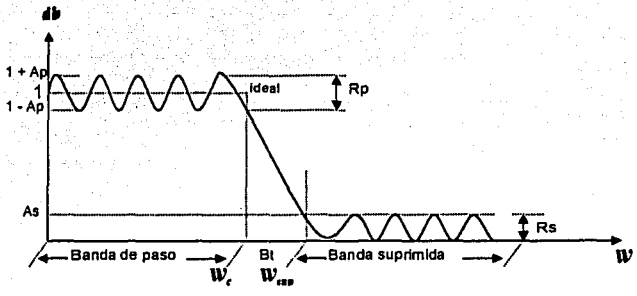


Figura 1.5 Características en magnitud de filtros físicamente realizables

Donde:

Ap = Ganancia en la banda de paso (Ap ≈ 1)

As = Atenuación en banda suprimida

Rp = Rizo en banda de paso

Rs = Rizo en banda suprimida

W_c = Frecuencia de corte en banda de paso

W_{sup} = Frecuencia de corte en banda suprimida

Bt = Banda de transición

2) Elegir el tipo de aproximación H(*w*), Butterworht, Chevyshev, elíptico, etc.;

3) Cálculo de parámetros, en caso de analógicos se calculan los valores de resistencias, capacitares e inductores; en el caso de filtros digitales, se calculan coeficientes.

4) Sustituir los parámetros o coeficientes en H(*w*) y graficar, verificando si cumple con especificaciones, de lo contrario regresar a 3.

5) Implementar, en caso de analógicos armar, en caso de digitales, programar la ecuación en diferencias.

6) Verificar su funcionamiento.

1.4. Filtros digitales de respuesta infinita al impulso (IIR)

Los filtros digitales de respuesta infinita al impulso (IIR) se caracterizan por la siguiente ecuación recursiva:

$$y(n) = \sum_{i=0}^{\infty} h(i)x(n-i) = \sum_{i=0}^{k} b_i x(n-i) - \sum_{i=1}^{k} a_i y(n-i) \text{.....(12)}$$

En forma desarrollada:

$$y(n) = b_0 x(n) + b_1 x(n-1) + \text{.....} + b_k x(n-k) - a_1 y(n-1) - a_2 y(n-2) - \text{....} - a_k y(n-k) \text{....(13)}$$

Donde; h (k) es la respuesta al impulso del filtro, la cuál es teóricamente de duración infinita. a_i y b_i son los coeficientes del filtro y x(n) y y(n) son la entrada y la salida del filtro. La función de transferencia está dada por;

$$H(z) = \frac{Y(z)}{X(z)} = \frac{B_0 + B_1 z^{-1} + \text{.....} + B_i z^{-k}}{1 + A_1 z^{-1} + \text{.....} + A_i z^{-k}} = \frac{\sum_{i=0}^{k} B_i z^{-i}}{1 + \sum_{i=1}^{k} A_i z^{-i}} \text{.....(14)}$$

Donde los z^{-1}, z^{-2}, ... z^{-i}, nos representan los retrasos (n-1), (n-2), ... (n-i), de la ecuación en diferencias anterior.

Nuestro objetivo es encontrar los valores apropiados para los coeficientes B_i y A_i, tales que la función de transferencia H (z), cumpla con los requerimientos en frecuencia que nos describa el filtro que se quiere calcular en forma digital. Esto Es;

$$H(\omega) = H(z) \Big|_{z=e^{j\omega}}$$

Se conocen cuatro métodos para el cálculo o aproximación de los coeficientes de interés, que son:

1) Ubicación de polos y ceros en el plano Z.
2) Método de in varianza al impulso.
3) Aproximaciones numéricas de las ecuaciones en diferencias.
4) El método de la transformada bilineal, que es el más común y que usaremos en este trabajo.

1.4.1 Cálculo de los coeficientes mediante la Transformada bilineal

La función de transferencia en el dominio Z puede ser obtenida aplicando la igualdad:

$$z = e^{sT} \text{ ..... (15)}$$

La relación entre el dominio s y z queda establecida mediante esa ecuación. Despejando a s se obtiene;

$$s = \frac{1}{T}\ln z \text{.....(16)}$$

Si expresamos ln (z) en forma de series, obtenemos;

$$s = \frac{2}{T}\left[\frac{z-1}{z+1} + \frac{1}{3}\left(\frac{z-1}{z+1}\right)^3 + \frac{1}{5}\left(\frac{z-1}{z+1}\right)^5 + \text{.....}\right] \text{.....(17)}$$

$$s = \frac{1}{T}\left[\frac{z-1}{z} + \frac{1}{2}\left(\frac{z-1}{z}\right)^2 + \frac{1}{3}\left(\frac{z-1}{z}\right)^3 + \text{.....}\right] \text{.....(18)}$$

La primera serie converge en la región en que z se aproxima a 1. Entonces se obtienen los dos casos de la transformación bilineal.

$$s = \frac{2}{T}\left(\frac{z-1}{z+1}\right) \text{.....(19)}$$

$$s = \frac{1}{T}\left(\frac{z-1}{z}\right) \text{.....(20)}$$

Esas dos relaciones permiten mapear el plano "s" en el plano "z". La relación (20) introduce mucha deformación. La figura 1.6 ilustra los dos casos. Si se hace c = 2/T en la transformación bilineal se obtiene;

$$s = c\frac{z-1}{z+1} \quad \text{.....(21)}$$

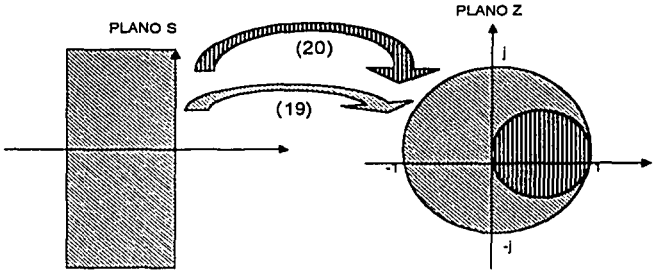


Figura 1.6 Mapeo del plano S al plano Z mediante la transformación bilineal

Entonces; si conocemos una función de transferencia H(s) que emula un sistema analógico, con la transformada bilineal se puede obtener el correspondiente sistema discreto.

La constante c nos corrige la distorsión del eje de la frecuencia y se calcula como sigue;

Como $s = j\omega_A$ y $z = e^{j\omega D}$ donde ω_A es la frecuencia analógica y ω_D es la frecuencia digital, entonces, sustituyendo;

$$j\omega_A = c\frac{e^{j\omega D} - 1}{e^{j\omega D} + 1} \quad \text{.....(22)}$$

Multiplicando numerador y denominador por $e^{-j\omega D/2}$ se tiene:

$$j\omega_A = c\frac{\frac{e^{j\frac{\omega DT}{2}} - e^{-j\frac{\omega DT}{2}}}{2}}{\frac{e^{j\frac{\omega DT}{2}} + e^{-j\frac{\omega DT}{2}}}{2}} \quad \text{.....(23)}$$

$$j\omega_A = c\frac{jsen\frac{\omega DT}{2}}{\cos\frac{\omega DT}{2}} \quad \text{.....(24)}$$

Como en las tablas de filtros se encuentran las funciones de transferencia normalizadas, las frecuencias de corte del filtro analógico es $\omega_A = 1$, si despejamos la ecuación anterior queda;

$$c = \cot\frac{\omega_D T}{2} \quad \text{.....(25)}$$

Donde ω_D es la frecuencia de corte del filtro digital. Sustituyendo a $\omega_D = 2\pi f_D$ y $T = 1/f_s$ se obtiene;

$$c = \cot\frac{\pi f_D}{f_s} \quad \text{.....(26)}$$

Donde f_D es la frecuencia de corte no normalizada digital y f_s es la frecuencia de muestreo. La ecuación (21) y constante (26), funcionan para hacer la transformación bilineal de un filtro pasa bajas. Si se quiere hacer un filtro pasa altas, se usan las siguientes ecuaciones.

$$s = c_1\frac{z+1}{z-1} \quad \text{.....(27)}$$

donde:

$$c_1 = \tan\frac{\pi f_{1D}}{f_s} \quad \text{.....(28)}$$

Una vez que se obtuvieron los coeficientes de la función de transferencia de un filtro deseado, se tiene la función que es de la forma $H(z) = Y(z)/X(z)$, donde $Y(z)$ es la salida del sistema; sólo hay que despejar $Y(z)$ y mediante la transformada Z inversa se llega a la ecuación en diferencias que describe el filtro digital deseado. Una vez que se obtuvo la ecuación en diferencias, se procede a simularla en

algún paquete de procesamiento de señales y comprobar que su respuesta sea la adecuada según las especificaciones dadas. Finalmente se procede a implementar la ecuación en algún dispositivo de procesamiento de señales en forma digital. En nuestro caso lo haremos en el DSP TMS320C50.

Cuando se diseñen filtros de orden mayor a 2, es conveniente pasar su función de transferencia a alguna estructura digital, ya sea en cascada o paralela, las cuáles describimos más adelante. Ello es debido a que la acumulación de errores de redondeo en el desarrollo del filtro causa inestabilidad en estructuras directas.

Podemos resumir el diseño de filtros digitales IIR en los siguientes pasos;

1.4.2. Procedimiento para el diseño de filtros IIR

1) Dar las especificaciones del filtro en forma analógica, frecuencias de corte, atenuación tanto de banda de paso como de banda suprimida, frecuencia de muestreo, etc.

2) Elegir la función de transferencia normalizada H(s), que pueden ser Chevyshev, Elípticas, Bessel, etc. En nuestro caso usaremos la aproximación polinómica de Butterworth. En este caso calcular el orden mínimo del filtro mediante la ecuación (29)[1], que calcula el orden mínimo de un filtro analógico con aproximación de Butterworht.

$$n \geq \frac{Log(\frac{10^{(0.1)As}-1}{10^{(0.1)Ap}-1})}{2Log(\frac{f_{sup}}{f_c})} \quad \text{.....(29)}$$

Para el caso pasa bajas; donde:

As = Atenuación en la banda de paso (en valor absoluto).
Ap = Atenuación en banda suprimida (en valor absoluto).

[1] Ifeachor Jervis .,*Digital signal proscessing a practical aproach* Adisson wesley1989

f_c = frecuencia de corte de la banda suprimida.

f_{sup} = frecuencia corte de la banda de paso.

Para el caso pasa altas, se deben invertir los coeficientes f_{sup} y f_c.

3) De tablas escoger su correspondiente H(s). Los polinomios de Butterworth hasta orden 6 son;

ORDEN	POLINOMIOS FACTORIZADOS
1	$(S + 1)$
2	$(S^2+\sqrt{2}\,S +1$
3	$(S^2 + S +1)(S + 1)$
4	$(S^2 + 0.76535\,S +1)(S^2 + 1.84776\,S +1)$
5	$(S + 1)(S^2 + 0.618\,S +1)(S^2 + 1.618\,S +1)$
6	$(S^2 + 0.5176\,S +1)(S^2 + \sqrt{2}\,S +1)(S^2 + 1.9318\,S + 1)$

4) Aplicar la transformada z bilineal para obtener H(z). Aplicando alguna de las siguientes transformaciones según sea el caso.

$$H(z)_{P.B.} = H(s)\Big|_{s=C1\left(\frac{z-1}{z+1}\right)}$$

$$H(z)_{P.A.} H(s)\Big|_{s=C2\left(\frac{z+1}{z-1}\right)}$$

$$H(z)_{P.Bw.} = H(s)\Big|_{s=C1\left(\frac{z-1}{z+1}\right)+C2\left(\frac{z+1}{z-1}\right)}$$

donde:

$$C1 = ctg\left(\frac{\pi f_{c.a}}{f_s}\right)$$

$$C2 = \tan\left(\frac{\pi f_{c.a}}{f_s}\right)$$

Donde; $f_{c.a}$ es la frecuencia de corte en forma analógica y f_s es la frecuencia de muestreo.

5) Como H (z) = Y(z) / X(z), despejar Y(z) y mediante la transformada Z inversa obtener y(n), la cuál es la salida del sistema. Aquí se han obtenido los coeficientes b_i y a_i de la ecuación en diferencias que describe y(n).

6) Elegir una estructura para el filtro:
Directa I, directa II, cascada o paralelo

7) Implementar el filtro.

1.4.3. Estructuras de filtros IIR

Cuando se diseñan filtros digitales, se hace asumiendo que serán implementados en algún dispositivo de precisión infinita, pero como en la realidad todos los dispositivos son de precisión finita, es necesario aproximar los coeficientes a una forma ideal, esta aproximación introduce errores en la cuantización de los coeficientes al hacer el redondeo. Cuando se diseñan filtros IIR de bandas muy estrechas, el resultado es que los polos caen muy cercanos al círculo unitario, y en consecuencia se requerirán dispositivos con una longitud de palabra muy grande, para poder cuantizar los coeficientes de una forma adecuada. Como se verá en la siguiente sección, las estructuras de cascada y paralelo implementan cada par de conjugados complejos por separado, como resultado, cada par de conjugados complejos son independientes de los demás, ayudando en minimizar los errores en el redondeo al momento de realizar las operaciones de multiplicación y acumulación del filtro. Otros errores que se suman a los anteriores, es la cuantización de las muestras de entrada y salida al dispositivo. La suma de todos los errores puede causar que la posición de polos y ceros dentro del círculo unitario se mueva, provocando una respuesta no deseada, o incluso que el sistema se haga inestable.

Las estructuras de filtros IIR en forma directa son muy sensibles a los errores de cuantización, sobre todo cuando el orden del filtro es mayor que 2. La solución a este problema es implementar el filtro ya sea en una estructura en cascada o paralelo, las cuáles son menos sensibles a la suma de errores.

Otro punto importante en este tema, es que como los coeficientes son números con punto decimal y muy pequeños; para poder hacer operaciones con estos números en el DSP, debemos de ocupar lo más posible de la longitud de palabra del dispositivo, como la longitud de palabra del TMS320C50 es de 16

bits, si todos los coeficientes obtenidos en el diseño se encuentran entre 0 y 1, debemos normalizarlos a formato Q15, es decir, multiplicar cada uno de ellos por 2^{15}, para dejar el bit de signo necesario y de esta forma ocupar todo el ancho de palabra de la memoria. Si alguno o algunos de los coeficientes se encuentran entre 1 y 2 se normaliza a Q14 y así sucesivamente. De no hacerse de esa forma, al hacer operaciones con números muy pequeños, el resultado será cero.

Forma directa I y II

Una vez que se obtiene la ecuación en diferencias de la forma;

$$y(n) = b_0\, x(n) + b_1\, x(n-1) + b_2\, x(n-2) + \ \ldots + b_k\, x(n-k) - a_1\, y(n-1) - a_2\, y(n-2) - \ldots - a_k\, y(n-k)$$

Su estructura se le nombra forma directa I y se representa de la siguiente forma:

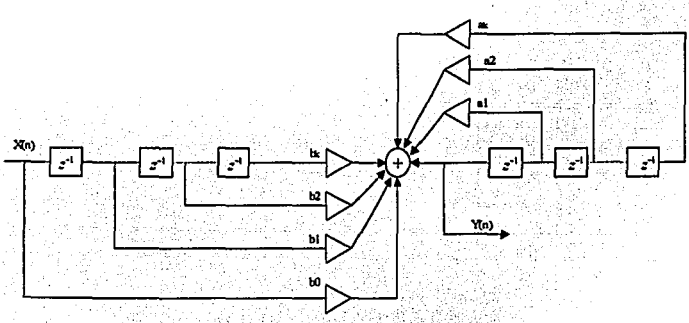


Figura 1.7 Forma directa I del filtro digital IIR de

Donde los z^{-1} son los correspondientes retrasos expresados en la ecuación en diferencias como x(n-1), x(n-2), .. x(n-i) y también y(n-1), y(n-2), .. y(n-i), y que en la realidad significan las muestras de entrada y salida, en este caso las muestras de la señal a filtrar y sus salidas correspondientes.

A la estructura anterior de un filtro se le conoce como forma directa I, la cuál se puede dibujar de la siguiente forma que es equivalente.

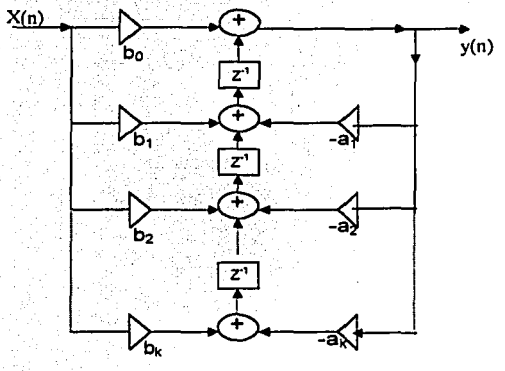


Figura 1.8 Forma directa I del filtro IIR de orden k

A la estructura de la figura 1.8 también como forma canónica debido a que tiene tantos elementos de retardo como el orden del filtro. Cambiando los nodos por sumadores y sumadores por nodos, cambiando la dirección de las señales e invirtiendo la dirección de los amplificadores, se llega a la estructura de forma directa II.

La figura 1.9 muestra la estructura de forma directa II para un filtro IIR de segundo orden.

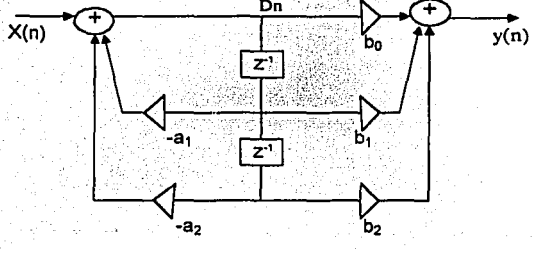


Figura 1.9 Forma directa II de un filtro IIR de segundo orden

Esta estructura es equivalente a la anterior y además es más fácil de programar ya que consume menos memoria y contiene un número menor de operaciones a realizar. Esta es la estructura que usaremos para programar los filtros ya que es la más recomendable. En este caso, como se observa en la figura, lo que debemos programar para obtener y(n) es lo siguiente:

$$d(n) = x(n) - a_1\, d(n-1) - a_2\, d(n-2) \ldots\ldots\ldots\ldots (30)$$
$$y(n) = b_0\, d(n) + b_1\, d(n-1) + b_2\, d(n-2) \ldots\ldots\ldots (31)$$

Esta estructura es muy sensible a los efectos de cuantización de errores en el cálculo de los coeficientes cuando se diseñan filtros de orden mayor a dos, lo cuál causa inestabilidad y origina respuestas no deseadas. Lo que se debe de hacer al diseñar filtros de orden alto, es descomponer su función de transferencia en la estructura de cascada o paralela.

Estructura en cascada

Para obtener la estructura en cascada, lo que se hace es descomponer la función de transferencia original en productos de fracciones parciales, esto es;

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + \ldots}{1 + a_1 z^{-1} + a_2 z^{-2} + \ldots\ldots} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (31)$$

$$H(z) = b_0 \left[\frac{(z-c_1)}{(z-p_1)}\right]\left[\frac{(z-c_2)}{(z-p_2)}\right]\left[\frac{(z-c_3)}{(z-p_3)}\right]\left[\frac{(z-c_4)}{(z-p_4)}\right] \ldots (32)$$

Donde;

$$H(z) = b_0\, H_1(z) * H_2(z) * H_3(z) * \ldots\ldots * H_i(z) \ldots\ldots\ldots\ldots (33)$$

Y también;

$$c_1 = c^*_2; \quad c_3 = c^*_4; \quad p_1 = p^*_2; \quad p_3 = p^*_4; \quad y;$$

$H_1(z)$, $H_2(z)$, ... son funciones de transferencia de 2° orden y se les llama *mallas biquad*, se le nombra estructura en cascada porque la salida de cada etapa es la entrada a la siguiente etapa. Esto se muestra mejor con el siguiente ejemplo;

Suponer que se tiene la siguiente función de transferencia y se debe de obtener su estructura en cascada.

$$H(z) = \frac{3 + 3.6z^{-1} + 0.6z^{-2}}{1 + 0.1z^{-1} - 0.2z^{-2}} \quad \text{.....(34)}$$

Debemos de descomponer la función de transferencia en productos parciales, donde la salida de una etapa es la entrada a la siguiente, esto es;

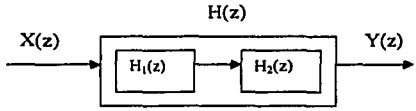


Figura 1.10 Diagrama de bloques para un sistema en serie

La H(z) se puede descomponer como sigue:

$$H(z) = \left(\frac{3 + 3z^{-1}}{1 + 0.5z^{-1}}\right)\left(\frac{1 + 0.2z^{-1}}{1 - 0.4z^{-1}}\right) \quad \text{.....(35)}$$

Despejando cada H(z) y obteniendo la transformada z inversa de cada producto se obtiene la ecuación en diferencias de cada función, esto es:

$$y_1(n) = 3\,x(n) + 3\,x(n-1) - 0.5\,y_1(n-1); \qquad y_2(n) = y_1(n) + 0.2\,y_1(n-1) + 0.4\,y_2(n-1)$$

Y para poder programarla se dibuja como sigue:

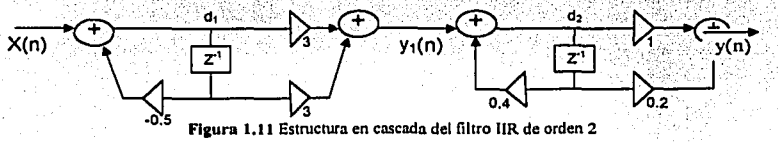


Figura 1.11 Estructura en cascada del filtro IIR de orden 2

Estructura en paralelo

La estructura en paralelo es la otra forma que es menos sensible a la cuantificación de errores en el cálculo de los coeficientes. Esta consiste en dividir la función de transferencia H(z) en sumas de fracciones parciales, tal como se ilustra a continuación:

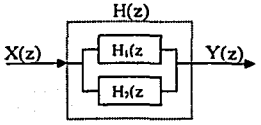


Figura 1.12 Diagrama de bloques para un sistema en cascada

De tal forma que la y(n) queda en función de varias sumas de ecuaciones en diferencias. Por ejemplo, considere el ejemplo anterior. La función de transferencia es;

$$H(z) = \frac{3(z+1)(z+0.2)}{(z+0.5)(z-0.4)} \quad \text{.....(36)}$$

Si se descompone en fracciones parciales:

$$\frac{H(z)}{z} = \frac{3(z+1)(z+0.2)}{z(z+0.5)(z-0.4)} = \frac{A}{z} + \frac{B}{z+0.5} + \frac{C}{z-0.4} \quad \text{.....(37)}$$

Calculando las constantes se obtiene;

$$A = -3, \quad B = -1, \quad C = 7;$$

De donde se obtiene que:

$$H(z) = -3 - \frac{1}{1+0.5z^{-1}} + \frac{7}{1-0.4z^{-1}} \quad \text{.....(38)}$$

Las funciones de transferencia parciales son;

$$H_1(z) = -3$$
$$H_2(z) = \frac{-1}{1+0.5z^{-1}}$$
$$H_3(z) = \frac{7}{1-0.4z^{-1}}$$

Y sus correspondientes ecuaciones en diferencias son:

$y_1(n) = -3\ x(n)$, $\quad y_2(n) = -x(n) - 0.5\ y_2(n-1)$, $\quad y_3(n) = 7\ x(n) + 0.4\ y_3(n-1)$,

Las cuáles se pueden programar de la siguiente forma:

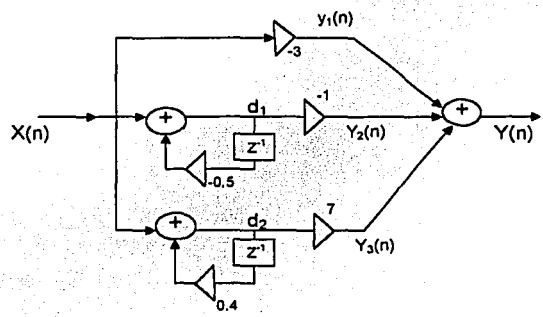


Figura 1.13 Estructura paralela del filtro digital

Aplicando alguna de estas dos últimas estructuras se pueden realizar filtros de orden alto ayudando a minimizar los errores de cuantificación. Nótese que en las dos estructuras, tanto cascada como paralela, va implícita la forma canónica II, ya que resulta más fácil implementarla de esta forma.

1.5. Filtros digitales de respuesta finita al impulso (FIR)

El filtro digital básico FIR con longitud M, entradas *x (n)* y salidas *y(n)* se describe mediante la ecuación en diferencias

$$y(n) = b_0 x(n) + b_1 x(n-1) + \ldots\ldots + b_{M-1} x(n-M-1) \ldots\ldots\ldots (39)$$

$$= \sum_{k0}^{M-1} b_k x(n-k) \ldots\ldots\ldots (40)$$

Donde $\{b_k\}$ es conjunto de coeficientes del filtro, alternativamente podemos expresar la secuencia de salida como la convolución de la respuesta impulso del sistema *h(n)* con la señal de entrada. Así tenemos;

$$y(n) = \sum_{k=0}^{N-1} h(k) x(n-k) \ldots\ldots\ldots (41)$$

El filtro también se puede caracterizar por su función de transferencia, el cual se puede ver como un polinomio de grado N-1 en la variable z^{-k} las raíces de este filtro corresponden a los ceros del polinomio.

$$H(z) = \sum_{k=0}^{N-1} b_k z^{-k} \ldots\ldots\ldots (42)$$

Cuando un filtro se implementa de esta forma se llama de forma directa y el sistema es siempre estable se resume entonces que el filtro FIR posee las siguientes características:

-Número de coeficientes igual a N.

-Número de retardos igual a N-1.

-Los filtros FIR son muy simples de implementar (fáciles de programar).
-Requieren un número elevado de multiplicaciones.

1.5.1.-Estructura de implementación del filtro FIR

La figura 1.14 muestra la forma de implementar un filtro de tipo FIR de orden N.

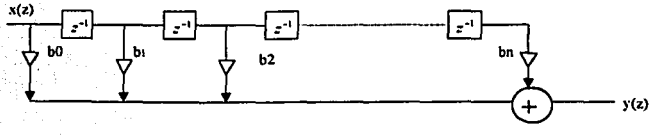


Figura 1.14 Estructura de un filtro FIR en forma directa

1.5.2. Técnicas para calcular los coeficientes del filtro

Se conocen cuatro métodos para calcular la respuesta al impulso de un filtro FIR que son:
1.-Por aproximación a una respuesta ideal.
2.-Uso de ventanas.
3.-Muestreo de la frecuencia.
4.-Solución a una aproximación matemática de la ecuación en diferencias.

1.5.3. Aproximación a una respuesta Ideal

Al tener un espectro normalizado se desea encontrar una función con una respuesta al impulso de tipo h(n) la cual genere el espectro normalizado, se encuentra entonces dicha función para el caso del filtro pasa bajas .

Como la respuesta en frecuencia de un filtro no recursivo es una función periódica de ω_s Puede ser expresada en forma de una serie exponencial de Fourier, podemos entonces escribir:

$$H(e^{j\omega T}) = \begin{cases} 1, & |\omega| \le \omega_c \quad \omega_c = \text{frecuencia de corte} \\ 0, & \text{para todos los demás } \omega \end{cases}$$

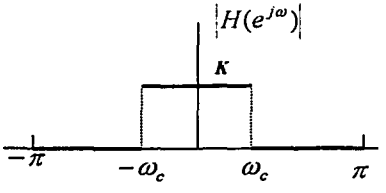


Figura 1.15 Espectro normalizado de un filtro

$$h(nT) = \frac{1}{2\pi}\int_{-\omega c}^{\omega c} H(\omega)e^{-j\omega n}d\omega = \frac{1}{j2\pi n}(e^{-j\omega c} - e^{j\omega c}) = \left[\frac{e^{j\omega c} - e^{-j\omega c}}{2j}\right] \text{.....(43)}$$

$$= \frac{sen\omega_c n}{\pi n} \text{.....(44)}$$

Sustituyendo $\omega_c = \omega_{cd}$

ω_c= Frecuencia de corte

ω_{cd}= Frecuencia de corte digital

$f_{c.a}$= Frecuencia de corte analógico

f_s= Frecuencia de muestreo

$$\omega_{cd} = \left[\frac{2\pi f_{c.a}}{f_s}\right]$$

$$h(nT) = \frac{sen(\omega_{cd} n)}{\pi n} \text{.....(45)}$$

De aquí definimos a la función para obtener los coeficientes del filtro pasa bajas.

$$h(n)=\begin{cases}\omega_{cd} & n=0\\ \dfrac{sen\omega_{cd}n}{\pi n} & n\neq 0\end{cases}$$

Este resultado indica que un número infinito de términos es requerido para implementar el filtro pasa bajas, tal resultado es en la práctica irrealizable, así que debemos truncar la expresión para h(n) con un número finito de valores n.

Debemos notar también que la expresión de h(n) es una respuesta no causal porque existen términos n<0, la causalidad es un problema que se resuelve fácilmente para los valores obtenidos, consiste entonces en correr todos coeficientes de n a la derecha (N-1)/2 veces. La figura 1.16 muestra los coeficientes de un sistema no causal.

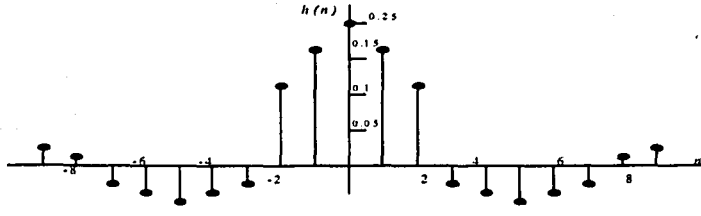


Figura 1.16 Coeficientes para un filtro no causal del tipo no recursivo

La figura 1.17 muestra los coeficientes del sistema causal. Nótese que solo hay que recorrer los coeficientes, para que estos se definan a partir de n = 0 y no antes de cero.

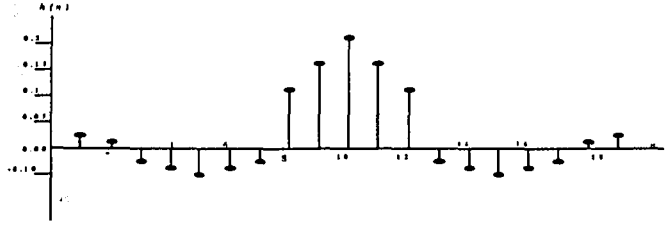


Figura 1.17 Coeficientes causales para un filtro no recursivo FIR

La siguiente tabla ulistra las diferentes ecuaciones que se usan para el cálculo de filtros digitales FIR.

Tipo de Filtro	*Respuesta ideal al impulso $h_D(n)$*	
	$h(n), n \neq 0$	ω_{cd}
Pasa bajas	$h(n) = \frac{sen(\omega_{cd} n)}{\pi n}$	$\omega_{cd} = \frac{2\pi f_{c.a}}{f_s}$
Pasa altas	$h(n) = (-1)^n \frac{sen(\omega_{cd} n)}{\pi n}$	$\omega_{c.d} = \pi - \omega_{cd}$
Pasa banda (PW)	$h(n) = 2\cos(\omega_0)\frac{sen(\omega_1 n)}{\pi n}$	$\omega_0 = \frac{\omega_H + \omega_L}{2}$ $\omega_1 = \frac{\omega_H - \omega_L}{2}$ $\omega_H = 2\pi f_{c.aH}$ $\omega_L = 2\pi f_{c.aL}$
Supresor de banda	$h(n) = 1 - h(0)_{PW}$ $h(n) = -h(n)_{PW}$	

Tabla con las formulas para el calculo de coeficientes de filtros FIR[2]

La respuesta en frecuencia deseada en un filtro nunca podrá ser exactamente realizada debido a dos razones, no se puede implementar un filtro con un número infinito de términos esto porque no existe arquitectura fisica con un número infinito de localidades de memoria para almacenar los datos de las muestra de entrada y coeficientes del filtro.

Segundo si se pudiera utilizar un número infinito de términos el filtro cortaría exactamente en la frecuencia de corte que elegimos (se mejora la pendiente) pero esto aumentaría los rizos en la banda de paso y en la banda de rechazo, esto es causado por el fenómeno de Gibss.

[2] Strum Rober Kirk Donal, First principle of discrete systems and digital signal processing., Adisson Wesley 1989.pag 629

1.5.4. Método de Ventanas

El método de ventanas nos permite alcanzar resultados más cercanos a los ideales, el simple hecho de truncar una serie infinita de términos puede ser visto en términos de una función de ventana.

Sabemos que los coeficientes del filtro se extienden infinitamente entonces a través de una ventana solo vemos una parte de estos coeficientes, un método de ventanas consiste en multiplicar los resultados truncados por otra función.

Ventana Rectangular

Esta ventana corresponde a truncar directamente a la $h(n)_D$, debido a que la ventana multiplica directamente a la $h(n)_{CAUSAL}$ está ventan produce atenuaciones pobres y no provee una muy buena pendiente entre la banda de paso y la banda de supresión.

Si se aumenta el número de coeficientes se obtienen pendientes más grandes pero aumenta el número de rizos en la banda de paso y la banda suprimida debido a los efectos de convolución en el dominio de la frecuencia. La ventana rectangular se define como;

$$W(nT)=\begin{cases} 1, & 0\le n \le M-1 \\ 0 & \text{Para todos los demás} \end{cases}$$

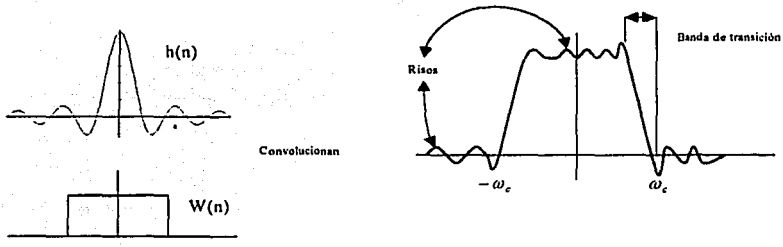


Figura 1.18 Operación de truncamiento de la respuesta al impulso y el efecto producido en el dominio de la frecuencia

Para aliviar la presencia de oscilaciones en la banda de paso como en la banda de rechazo se debe usar otro tipo de funciones que tengan una sima y decaigan hacia cero gradualmente en lugar de abruptamente como ocurre en la ventana rectangular, estas funciones de ventana efectivamente eliminan los efectos de rizado, la función de ventana $W(n)$ debe ser multiplicada por el valor de $h(n)_{CAUSAL}$ para calcular los coeficientes del filtro.

Existe un gran número de ventanas a continuación se enumeran las más importantes;

-Rectangular
-Triangular o de Bartlett
-Hanning
-Hamming
-Blackman

Ventana triangular (Bartlett)

A pesar que esta función de ventana no es ampliamente usada se presenta por que es sencilla

$$W_{TRIAN}(n) = \begin{cases} \dfrac{2n}{N-1} & 0 \le n \le \dfrac{N-1}{2} \\ 2 - \dfrac{2}{N-1} & \dfrac{N-1}{2} \le n \le N-1 \end{cases}$$

La ventana de Hanning es una función de coseno y esta dada por

Ventana de Hanning

La ventana de Hanning es una función de coseno y esta dada por;

$$W_{HANN(N)} \begin{cases} 0.5 - 0.5\cos\left(\dfrac{2\pi n}{N-1}\right) & 0 \le n \le N\text{-}1 \\ 0 \quad \text{para todos los demás} \end{cases}$$

Ventana de Hamming

La ventana de Hamming es similar a la ventana Hanning excepto que pose un una caída más suave y esta dada por;

$$W_{HAMM}(n)=\begin{cases}0.54-0.46\cos\left(\frac{2\pi n}{N-1}\right), & 0\le n\le N\text{-}1\\ 0 \text{ para todos los demás}\end{cases}$$

Ventana de Blackman

Es similar a las anteriores pero contiene un segundo termino y esta dada por;

$$W_{Black}(n)=\begin{cases}0.42-0.5\cos\left(\frac{2\pi n}{N-1}\right)+0.082\cos\left(\frac{4\pi n}{N-1}\right) & 0\le n\le N\text{-}1\\ 0 \text{ para todos los demás}\end{cases}$$

La siguiente tabla resume cada una de las ventanas y sus ecuaciones;

Nombre de la ventana	*Rizo en la banda de paso (dB)*	*Máxima atenuación en la banda de supresión en (dB)*	*Función de la ventana w(n)*
Rectangular	0.7416	21	1
Triangular		27	$\frac{2}{N-1}$ $\quad 0\le n\le\frac{N-1}{2}$ $2-\frac{2}{N-1}$ $\quad \frac{N-1}{2}\le n\le N-1$

Hanning	0.0546	44	$0.5+0.5\cos\left(\frac{2\pi n}{N-1}\right)$
Hamming	0.0194	53	$0.54+0.46\cos\left(\frac{2\pi n}{N-1}\right)$
Blackman	0.0017	74	$0.42+0.5\cos\left(\frac{2\pi n}{N-1}\right)+0.082\cos\left(\frac{4\pi n}{N-1}\right)$

Resumen de las características de las funciones de ventana más comunes[3]

Las siguientes figuras muestran las gráficas de las ventanas más importantes

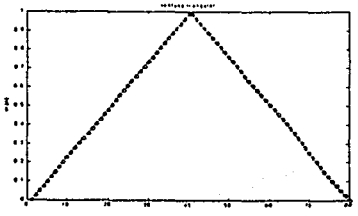


Figura 1.19 Ventana triangular

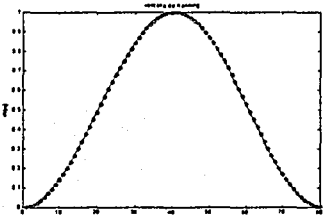


Figura 1.20 Ventana de Hanning

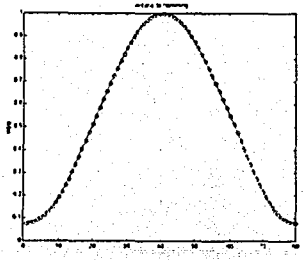


Figura 1.21 Ventana de Hamming

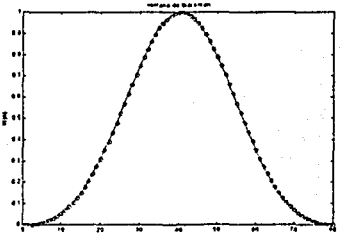


Figura 1.22 Ventana de Blackman

[3] Ifeachor Jervis. Digital signal processing a practical aproach., Adisson wesley 1989., pag. 294

1.5.5. Procedimiento de diseño para un filtro (FIR), usando método de ventanas

1.-.Especificaciones: Del filtro se especifica el tipo de filtro deseado, por ejemplo pasa bajas, las variaciones de amplitud en la banda de paso, la atenuación en la banda de rechazo, así como la frecuencia de muestreo.

2.-Calculo de los coeficientes Calcular los coeficientes del filtro usando las formulas para el filtro deseado, ordenar los coeficientes en forma causal y multiplicarlos por la ventana que satisface las especificaciones, también es necesario calcular el número de coeficientes necesario para implementar el filtro.

Calcular el orden del filtro se puede realizar utilizando la formula de Kaiser, si no proponer un N, se recomienda valores de N mayores o iguales a 60 para obtener respuestas adecuadas. La fórmula de Kaiser es la siguiente;

$$N = \frac{A_{sup} - 7.96}{14.36\Delta f} \text{.....Formula de Kaiser}^{4}$$

Dnde;

A_{sup} Atenuación en la banda de corte esta se encuentra en [dB] en valor absoluto.

$$\Delta f = \frac{\omega_c - \omega_{sup}}{2\pi}$$

ω_c= frecuencia de corte de la banda de paso.

ω_{sup}= frecuencia de corte en la banda suprimida.

Los coeficientes también se pueden calcular usando algún paquete de procesamiento de señales, en el apéndice A se dan algunas instrucciones de Matlab para calcular los valores de los coeficientes usando el método de ventanas.

Se recomienda también simular la respuesta en frecuencia del filtro con los coeficientes calculados par ver si cumple con las especificaciones del diseño, si no cumple con las especificaciones es necesario aumentar el número de coeficientes.

[4] Ifearchor Jervis. Digital signal processing a practical aproach .,Adisson wesley., 1989., pag 295

3.-Análisis del largo de palabra Aquí analizamos los efectos de cuantización del filtro, los efectos de las entradas de las muestras así como el tipo de procesador que estamos usando ya sea de aritmética de punto fijo o flotante.

4.- Implementación. Esto involucra generar el código de software o diseño de hardware para implementar el filtro.

5.-Prueba del filtro. Esto consiste en probar el código fuente o el hardware diseñado para ver si cumple con las especificaciones del diseño, si no regresar al paso 4 y si es necesario al paso 2.

1.5.6 Comparación entre los filtros FIR e IIR

En este capítulo describimos con cierto detalle las técnicas más usadas para diseñar filtros digitales FIR e IIR por lo que a continuación se da una comparación entre los filtros FIR e IIR.

(a) Estabilidad.

Losa filtros FIR son siempre estables pues solo poseen ceros, en cambio un sistema IIR posee tanto polos y ceros, y en algunos casos esto implica que nuestro filtro no sea estable.

(b) Atenuación y número de coeficientes.

Tanto los filtros FIR como los IIR logran buena atenuación (mayor a 40 [dB]), sin embargo un filtro FIR necesita un número elevado de coeficientes N, aproximadamente N> a 60, los filtros IIR pueden lograr la misma atenuación con un número mucho menor de coeficientes.

(c) Efectos de cuantización.

En un sistema IIR (polos, ceros) la salida y(n) depende de x(n), los retrasos de x(n) y los retrasos de y(n) esto provoca que para funciones de transferencia de orden mayor a dos los sistemas poseen

errores a la salida, la razón es que cada vez que se calcula una salida esta posee un pequeño error causada por la cuantización de los coeficientes de entrada x(n), truncar o redondear los coeficientes del sistema provoca otro error, estos errores se acumulan y se incrementan cada vez que entre una nueva muestra, por lo que al final obtenemos una salida no deseada, si se quiere minimizar los errores acumulados es necesario descomponer el sistema en funciones de trasferencia de grado dos conectándolas en serie o en paralelo.

Procesadores digitales de señales

2.1. Conceptos generales

En el modelo básico de procesamiento digital de señales es necesario realizar la operación de convolución en el tiempo más breve posible, la operación de convolución se define como;

$$y[n] = \sum_{k=0}^{N-1} h[k]x[n-k] \text{.....(2.1)}$$

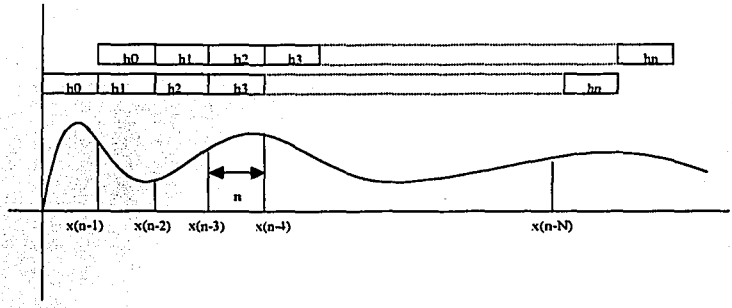


Figura 2.1 Operación de convolución entre las muestras de entrada x(n-k) por una ventana h(n) la ventana se recorre un lugar y se vuelve a calcular con nuevos valores de señal muestreada

La figura 2.1 muestra el esquema general que representa la operación de la convolución. Aquí los h_n son los valores de los coeficientes que se calculan para generar la respuesta al impulso que se quiere tener de el sistema, y los x(n-N) son las muestras de la señal de entrada; nótese que para x(0) se tiene la señal presente de entrada, mientras que los x(n-1), x(n-2,) etc. son las muestras que están retrasadas con respecto a la primer muestra de entrada. Para poder realizar la operación de convolución en el DSP asumimos que debemos tener N + 1 coeficientes en un vector de memoria de almacenamiento h y los valores más recientes de las muestras de la señal de entrada en un vector de memoria x de tamaño N + 1 de la señal. Justo antes de los cálculos correspondientes al tiempo n el vector x retiene;

$$\{x[n], x[n-1], \ldots x[n-N]\},$$

y al tiempo n + m necesitamos realizar las siguientes operaciones:

1. Poner una variable temporal *y*, cuando la operación se cumpla retendrá el cálculo obtenido para salida *y*[n].
2. Implementar un ciclo de conteo desde k = 0 hasta k = N.
3. Repetir la siguiente operación hasta K = N.
   (a) Cargar h[n]desde la localidad k – enésima de h a la CPU.
   (b) Cargar x[n-k] desde localidad k – enésima de x a la CPU.
   (c) Multiplicar h[k] por x[n-k] y opcionalmente redondear el resultado en precisión simple.
   (d) Carga *y*, sumar el producto y almacenarlo devuelta a *y*.
   (e) Incrementar el ciclo del contador k por 1, verificar si k ≥N y salir si la condición es cierta.
4. Sacar *y* a un destino (convertidor digital-analógico).
5. Recorrer el elemento *x* un lugar, borrando x[n - N] y hacer lugar para x[n + 1] en la primera posición del vector de memoria
6. Introducir x[n + 1] del convertidor analógico–digital y almacenarla en la primera posición de **x.**

Desde el punto de vista del hardware es necesario:

1. Tener dos áreas de memoria que puedan ser accesadas simultáneamente, entonces podemos, tener *x* en un área y *h* en otra, cargar *x*[n – k] y *h*[k] simultáneamente.
2. Podemos tener simultáneamente la variable *y* en un registro de la CPU entonces eliminar su carga de la memoria y almacenarla devuelta para cada ciclo de reloj podemos permitirnos que el acumulador sea de una longitud mayor comparado con x[n – k] y h[k] entonces el producto no necesita ser redondeado y se puede agregar directamente a un valor de *y*. La combinación de multiplicador de longitud doble y de acumulador de longitud doble es llamada MAC (Multiplicador – Acumulador)
3. Podemos usar hardware de conteo de ciclos el cual provoca que una operación se repita N + 1 veces sin ninguna instrucción especifica de repetición.
4.

La figura 2.2 muestra la arquitectura del hardware necesaria para implementar la operación de la convolución de una forma eficiente.

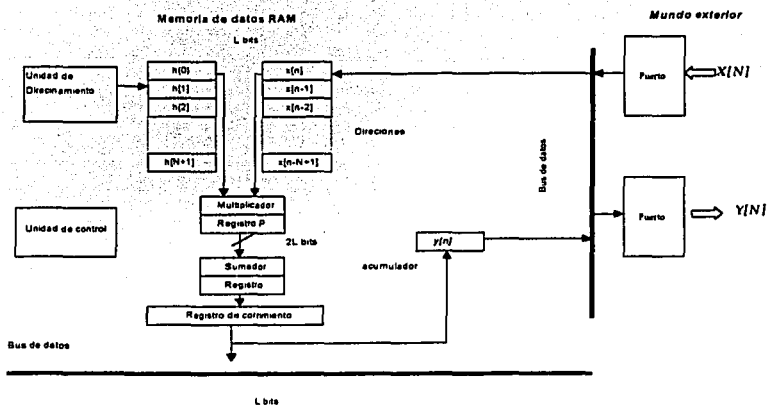


Figura 2.2 Arquitectura básica para implementar el algoritmo de convolución

Las operaciones que realice el procesador deben ser lo más rápidas posibles es decir en tiempo real, tiempo real significa realizar operaciones sobre una muestra de entrada y entregar resultados antes de que entre la siguiente muestra.

La figura 2.3 muestra la idea anterior, en esta figura, las líneas marcadas en negro indican la entrada de las muestras de la señal que se va a procesar, mientras que las líneas punteadas indican las muestras de salida generadas por el DSP. Estas muestras son las que se van directamente al convertidor analógico-digital, para regenerar la señal en forma analógica.

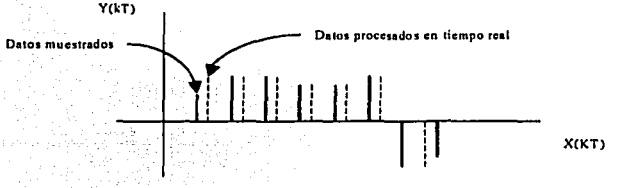


Figura 2.3 Muestras de entrada y datos entregados en tiempo real

2.2. Arquitectura

Si se desea utilizar la operación convolución es necesario contar con una arquitectura que permita

- Tener una estructura de buses múltiples con espacios de memoria separados, para memoria de datos y memoria de programas, generalmente la memoria de datos retine datos de entrada y salidas de muestras, también como coeficientes fijos.
- Puertos de entrada salida los cuales proveen un medio de pasar datos a, y de dispositivos externos tales como convertidores analógico-digital y digital-analógico y accesos directo a memoria DMA.
- Unidades de Aritmética lógica las cuales incluyen ALU y hardware multiplicador acumulador en paralelo dos datos de entra simultáneos

La arquitectura estándar de los microprocesadores (arquitectura Von Newman) no es conveniente para la implementación de operaciones de procesamiento digital de señales, donde es necesario implementar operaciones de multiplicación y acumulación, y acceso a memoria en un solo ciclo. Por lo que es necesario hacer uso de los conceptos de procesamiento en paralelo.

En particular, se usan las siguientes técnicas:

- Arquitectura Harvard.
- Ejecución en cascada (pipeline).
- Hardware rápido de multiplicación y acumulación MAC.
- Instrucciones especiales dedicadas al procesamiento digital de señales.
- Duplicación.
- Memoria caché en el circuito integrado.
- Desplazar datos en la memoria (retrasos).
- Manejo de arreglos de memoria.

2.3. Arquitectura Harvard

La principal característica de la arquitectura Harvard, es la de mantener en espacios separados la memoria de datos y la memoria de programa permitiendo una total superposición de instrucciones de búsqueda y ejecución.

Normalmente la memoria de programa retiene el código de programa mientras que la memoria de datos almacena variables tales como las entradas de datos muestreados, estrictamente la arquitectura Harvad es usada solo en algunos procesadores digitales de señal DSP (por ejemplo la familia ADSP5600 y ADSP9600) de Motorola.

Muchos DSP's utilizan una arquitectura harvard modificada, por ejemplo la familia TMS320CXX de Texas que posee una arquitectura Harvard modificada con memoria de datos y de programas separados pero, aun es posible la multiplicación entre ambos espacios de memoria, a diferencia de una arquitectura Harvard. La figura 2.4 muestra la arquitectura Harvard usada en los DSP's de Texas Instruments, este tipo de arquitectura también es usada en DSP's de otros fabricantes.

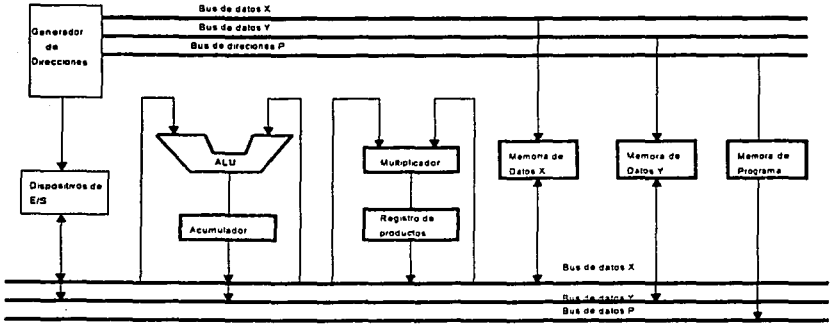


Figura 2.4 Arquitectura Harvard con unidad de multiplicación usada en Procesamiento Digital de Señales

2.4. Ejecución en cascada (Pipeline)

La ejecución en cascada es una técnica que permite que dos o más instrucciones se lleven a cabo en un mismo ciclo de reloj. Cada tarea es dividida en sub tareas por lo que en cada ciclo de reloj pueden activarse diferentes instrucciones al mismo tiempo. La siguiente figura muestra el esquema de cómo se desarrollan las instrucciones en cascada.

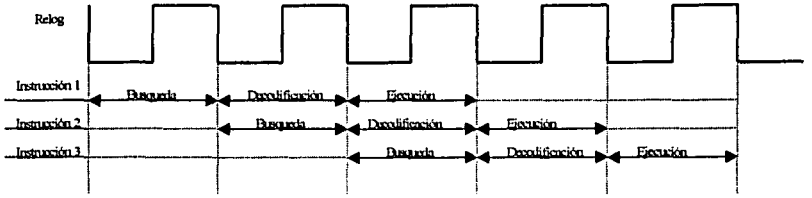


Figura 2.5 Pipeline ilustra el concepto de superponer instrucciones durante un mismo ciclo de reloj

2.5. Hardware Multiplicador Acumulador MAC

Las operaciones numéricas básicas en el procesamiento digital de señales son multiplicación y acumulación de los resultados. Por software es notorio el tiempo necesario para realizar las multiplicaciones y las acumulaciones e incluso se consume más tiempo si se utiliza aritmética de punto flotante. Para realizar operaciones en tiempo real se utiliza un hardware dedicado a multiplicar y acumular (MAC). La MAC en la actualidad es un estándar para todos los procesadores digitales de señales. En punto fijo el Hardware MAC acepta multiplicaciones en 16 bits, en complemento a 2 y entrega resultados en 32 bits en un solo ciclo. El tiempo usado en una MAC se puede reducir significativamente si se usa con instrucciones de repetición.

El Hardware MAC típico para un DSP tiene un par de registros de tamaño L que retienen las entradas de la multiplicación y un registro de salida de tamaño 2L el cual retiene el resultado de la multiplicación, la salida del registro de productos P es conectada a través de un acumulador de doble

precisión donde los productos son acumulados. La siguiente figura muestra una unidad típica de multiplicación-acumulación de una MAC.

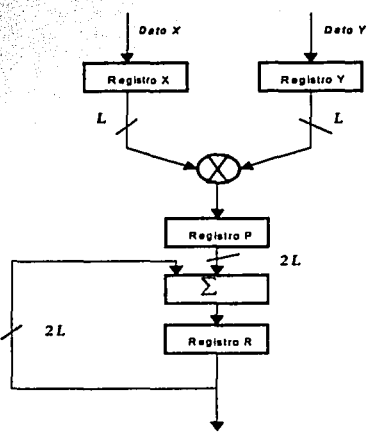


Figura 2.6 Unidad típica de multiplicación acumulación MAC usada en los DSP's

Duplicación

En un DSP, la duplicación involucra usar más de dos unidades básicas, por ejemplo usar más de una ALU, multiplicador o unidades de memoria; algunas veces las unidades son arregladas para trabajar simultáneamente por lo que es común tener una CPU con uno o más elementos duplicadores.

Memoria Caché en el integrado

En algunos casos, un integrado DSP opera tan rápido que memorias lentas y baratas son incapaces de operar simultáneamente. En la práctica común, los procesadores son detenidos por estados de espera lo que ocasiona que los procesadores no operen a toda su velocidad.

Para aliviar este problema muchos procesadores incluyendo a los DSP contienen integrados rápidos RAM o ROM externamente o incluidos directamente en el integrado.

Instrucciones especiales

Los procesadores digitales de señales proveen instrucciones especiales para optimizar el procesamiento de señales. Los beneficios son dobles, ellas conducen a un código más compacto, el cual requiere un menor espacio en memoria, y ello permite incrementar la velocidad de ejecución de algoritmos de procesamiento digital, ejemplos son la instrucción MACD (multiplica acumula y desplaza) que proveen la familia TMS320C10, C20, C25 y C50, DMOV (mueve), que desplaza una dato a la localidad siguiente para implementar un retrazo. EL DSP provee instrucciones especiales que permiten que datos muestreados se copien en direcciones más altas de memoria o que operen sobre ellos, todo en un solo ciclo.

Retrasos

Los procesadores digitales permiten crear arreglos de memoria los cuales se pueden copiar a una dirección más alta a una dirección más baja cada vez que entre una nueva muestra creando así un retraso z^{-1}, los retrasos pueden implementarse en cualquier procesador, sin embargo un DSP permite combinar los retrasos con multiplicaciones y acumulaciones en un solo ciclo.

Manejo de arreglos de memoria

Cuando se posee un arreglo de memoria con muestras de entrada de localidades de memoria consecutivas, se requiere acceder a ellos de una manera eficiente, lo que involucra generar la siguiente dirección del dato en memoria, por esta razón un DSP posee registros de direcciones los cuales pueden ser usados para marcar a una dirección y generar la siguiente dirección, estos apuntadores de dirección a diferencia de una pila, pueden apuntar a los datos en forma descendente, ascendente o saltar entre los arreglos.

2.6. Procesador digital de señales de propósito general

Un Procesador digital de señales es básicamente un microcontrolador de alta velocidad. Estos integrados hacen uso extensivo del concepto de paralelismo, arquitectura Harvard y ejecución en cascada, para realizar operaciones de convolución eficientemente. Sus características son;

- Deben operar en tiempo real (manejar grandes volúmenes de información en tiempos cortos).

ii Memorias rápidas.

iii Ciclos de instrucción muy rápidos.

iv Poseer un hardware multiplicador que efectúe operaciones de multiplicación en un ciclo de operación.

v Tener una poderosa unida de direccionamiento.

vi Poseer puertos para transferencia de información.

vii Manejo eficiente de interrupciones.

viii Conjunto de instrucciones orientadas a procesamiento digital de señales.

ix Trabaje en paralelo.

Las siguientes figuras muestran de una manera informal las diferencias entre los microprocesadores, microcontroladores y DPS's

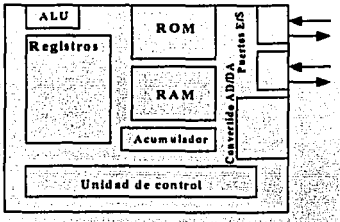


Figura 2.7 Microcontrolador

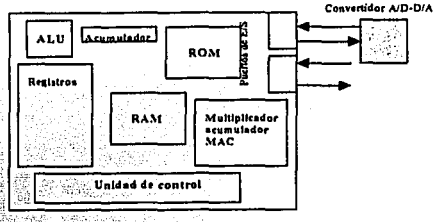


Figura 2.8 Procesador Digital de señales

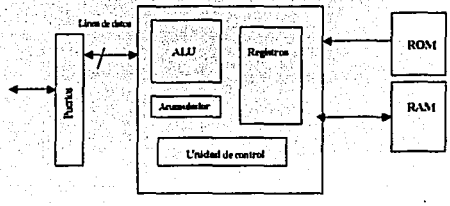


Figura 2.9 Microprocesador

Nótese las características de un DSP que lo hacen ser más apropiado para el procesamiento de señales que los microcontroladores o microprocesadores, la unidad multiplicación y acumulación, convertidor A/D y D/A muy rápido, entre otras.

En cualquier arquitectura programable vía software es posible implementar el algoritmo de convolución el cual es la base de todas las operaciones para procesamiento digital de señales, si embargo, ¿por que usar un DSP y no un microprocesador o un microcontrolador?, la diferencia básica es que un DSP es más rápido que un microcontrolador posee una unida de multiplicación de carga en paralelo que le permite realizar multiplicaciones y acumulaciones de multiplicaciones en un solo ciclo, sin embargo en un microcontrolador típico cada multiplicación implicaría aproximadamente 10 ciclos por multiplicación y dos o ciclos más para el desplazamiento de datos, por lo tanto si se quisieran realizar operaciones en tiempo real para procesamiento de señales en forma eficiente es necesario utilizar un DSP, la diferencia entre el DSP y el microprocesador, es que el microprocesador es un sistema más flexible puede expandirse la memoria y poseen velocidades superiores al DSP pero usar un microprocesador para implementar algoritmos de procesamiento digital es desaprovechar todas las demás funciones del microprocesador además de que el sistema sería más caro, otra razón es que el microprocesador no fue diseñado para procesar señales y en ciertos tipos de algoritmos un procesador no entregaría resultados en tiempo real.

La siguiente lista enuncia las principales aplicaciones de los DSP's.

Automóvil	**Consumidor**	**Control**
Teléfono celular	Radios/TV digitales	Control de manejo
Radios digitales	Juguetes educativos	de disco
Posicionamiento global	Sintetizadores de música	Control de motor
Navegación	Maquinas contestadoras de estado sólido	Control de
Análisis de vibraciones		impresión láser
Comandos de voz		Control de motores
Control de motor de		eléctricos
combustión interna		Control de Robots
Frenos antiderrapantes		Control de
		servomecanismos

Propósito general	**Gráficos / Imágenes**	**Industria**
Filtrado Adaptativo Convolución Correlación Filtrado digital Transformada rápida de Fourier Generación de ondas Transformadas de Hilbert Ventaneo	Rotación en 3-D Mapas digitales animados Reconocimiento de formas Mejoramiento de imagen Compresión / transmisión de imagen Visón de robots Centrales de trabajo	Control numérico Monitoreo de líneas de potencia Robótica Accesos de seguridad
Instrumentación	**Medicina**	**Militar**
Filtrado digital Generador de funciones Diseño de maquinas Análisis espectral Análisis transitorio	Equipo de diagnostico Monitoreo fetal Implantes auditivos Monitoreo de pacientes Equipo de ultrasonido	Procesamiento de imágenes Mísiles guiados Navegación Procesamiento de radar Modems de radio frecuencia Comunicaciones seguras Procesamiento de sonar

Telecomunicaciones		**Voz y audio**
Modems de 1200 a 19200 bps	Codificación/Decodificación DTMF	Reconocimiento de
Ecualizadores adaptativos	Cancelador de eco	voz
Transcoders ADPCM	Fax	Sintetizadores de
Teléfono celular	Líneas repetidoras	audio
Multiplexaje de canales	Contestadores de teléfono	Verificación de voz
Encriptación de datos	Vídeo conferencias	Correo de vos
PBX digitales	Sistemas de comunicación personal	Convertidores de
Intercambio de paquetes X.25		texto a sonido
Asistencia personal digital (PDA)		

Aplicaciones típicas para la familia TMS320 de Texas Instruments

Arquitectura del DSP TMS320C50

3.1. Evolución de la familia TMS320C50

En 1982, Texas Instruments introdujo el TMS32010, el primer DSP de la familia TMS320 de punto fijo. Este producto fue el modelo para futuras generaciones de esta familia. El día de hoy, la familia TMS320 consiste de ocho generaciones: la 'C1x, 'C2x, 'C2xx, 'C5x de punto fijo, y los 'C3x, 'C4x y 'C6x son de punto flotante y la 'C8x que es un multiprocesador.

El código central se ha hecho compatible entre una generación y la siguiente de punto fijo, excepto la generación 'C54x. También se ha desarrollado compatible con las generaciones de punto flotante. Este trabajo se basa en el DSP TMS320C50, el cuál está basado en el c25, con algunas mejoras.

Aplicaciones típicas de la familia TMS320.

Los DSP han sido exitosamente aplicados en voz y audio, en productos de tiempo real, tales como módems, canceladores de eco, codificación de voz y audio, síntesis de voz, reconocimiento de voz, digitalización de audio etc. En el capítulo 2 se presento una lista con las aplicaciones más importantes de las familias de Texas Instruments.

3.2. Arquitectura del DSP TMS320C50

El TMS320C50 es un procesador de punto fijo de la familia TMS320 que está basado en el TMS325C25 con una arquitectura adicional que mejora el desempeño, entre otras características están:

- Ciclo de instrucción más rápido.
- Manipulación de bits vía la unidad paralela lógica.
- Gran cantidad de memoria interna.
- Estados de espera por software.

- Respuesta rápida a interrupciones.
- Bloques de repetición.
- Buffer circular.
- Corrimientos de 0 a 16 bits en el acumulador.
- Registros shadow.

Los dispositivos de la generación 'C5x son capaces de efectuar operaciones en dos veces la velocidad de la generación 'C2x y su código es compatible hacia arriba con las familias 'C1x y 'C2x. La familia 'C5x está diseñada para ejecutar 28 millones de instrucciones por segundo.

3.2.1. Principales características del DSP TMS320C50

- 35 ns de ciclo de instrucción para instrucciones en punto entero (28.6/20 MIPS)
- 9k x 16 bits de memoria RAM interna, con acceso en un ciclo (SARAM)
- 2k x 16 bits en memoria ROM acceso en un ciclo.
- 1056 x 16 bits en memoria RAM interna, puede accesarce dos veces por ciclo de máquina (DARAM)
- 224k x 16 bits máximo espacio direccionable en memoria externa (64k palabras de programa, 64k de palabras de datos 64k puertos I/O y 32 palabras globales)
- Acumulador (ACC) de 32 bits y un acumulador buffer (ACCB) de 32 bits.
- Unidad Lógica Paralela (PLU) de 16 bits.
- Multiplexor paralelo de 16 x 16 con capacidad de productos de 32 bits.
- Instrucciones de multiplicación / acumulación en un ciclo simple.
- Ocho registros auxiliares con una unidad aritmética para direccionamiento indirecto.
- Ocho niveles de pila por hardware.
- 0 a 16 corrimientos a la izquierda o derecha.
- Dos buffer de direccionamiento indirecto para direccionamiento circular.
- Instrucciones de repetición de bloques.
- Instrucciones para el manejo de movimiento de bloques entre dos programas.
- Puerto serial síncrono full-duplex para comunicación directa con otros 'C5x y otros dispositivos seriales.

- Puerto serial de múltiple acceso por división de tiempo (TDM)
- Temporizador
- 64k puertos I/O paralelos, con 16 puertos mapeados en memoria.
- 16 estados de espera programables por software, para programa datos o espacio I/O.
- Operación de DMA.
- Cuatro niveles de Pipeline.
- Direccionamiento de bit reverso para efectuar FFT's radix 2.
- Generador interno de reloj.
- Polarización de 5 volts, con modo baja potencia (power down).
- Empacado en 132 pines.
- 28 registros mapeados en memoria.
- Cuatro interrupciones externas y cinco internas una del timer y cuatro para puerto serie.
- 32 interrupciones por software.

El TMS50 consta de cuatro bloques básicos.

- **LA ESTRUCTURA DEL BUS.**
- **UNIDAD CENTRL DE PROCESO (CPU).**
- **MEMORIA**
- **INTERFAZ A PARIFERICOS**

3.2.2. Estructura del bus

El contar con buses de programa y datos separados, permite acceder simultáneamente al programa de instrucciones y datos, proveyendo un alto grado de paralelismo. Por ejemplo mientras un dato es multiplicado, un producto previo puede ser cargado en el acumulador, sumado o sustraído de este, al mismo tiempo, una nueva dirección puede ser generada. Tal paralelismo soporta un potente grado de operaciones aritméticas, lógicas y manipulación de bits, las cuáles pueden ser desarrolladas en un solo ciclo de máquina. Además, el 'C5x incluye los mecanismos de control para administrar las interrupciones, repetición de operaciones y llamadas de funciones.

La arquitectura del 'C5x está constituida por cuatro buses principales, como se indica en la figura3.1.

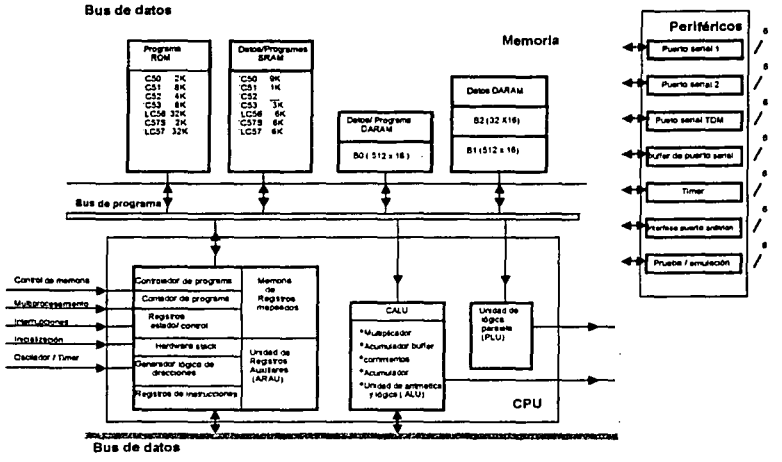


Figura 3.1 Esquema general de la arquitectura del tms320c50

- Bus de programa (PB)
- Bus de direcciones de programa (PAB)
- Bus de lectura de datos (DB)
- Bus de lectura de direcciones de memoria (DAB)

El PAB provee de direcciones al espacio de memoria de programa, tanto para lectura como escritura. El PB acarrea el código de instrucción y operandos inmediatos de la memoria hacia el CPU. El DB interconecta varios elementos del CPU hacia el espacio de memoria de datos.

El programa y el bus de datos pueden trabajar juntos para transferir datos desde la memoria de datos y de la memoria interna o externa hacia el multiplicador para realizar operaciones de multiplicar – acumular en un solo ciclo.

3.2.3 Unidad central de proceso (CPU).

El CPU de la generación 'C5x consta de los siguientes elementos:

- Unidad aritmética lógica central (CALU).
- Unidad lógica paralela (PLU).
- Unidad aritmética de registro auxiliar (ARAU).
- Registros de mapa de memoria.
- Controlador de programa.

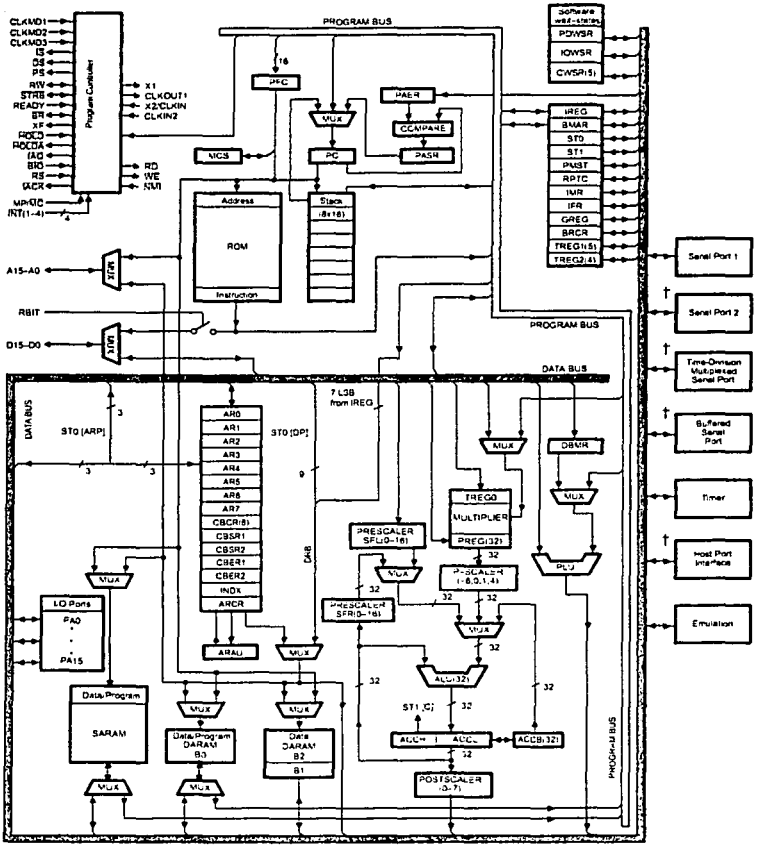
El CPU de la generación 'C5x mantiene el código fuente compatible con la generación 'C1x y 'C2x, y logra un mejor rendimiento y gran versatilidad. Algunas de sus mejoras son, un buffer acumulador de 32 bits, más capacidades y una multitud de nuevas instrucciones.

El hardware interno del 'C5x ejecuta funciones internas que otros procesadores típicos lo implementan en software o con micro código. Por ejemplo, el 'C5x contiene hardware para multiplicación de 16 x 16 bits en un solo ciclo, corrimiento de datos, y manipulación de direcciones. En la siguiente página se muestran los bloques principales y las pistas del 'C5x.

La siguiente lista presenta un resumen del hardware interno del 'C5x. Los registros mapeados se mencionan más adelante.

Símbolo	Nombre
A15-A0	Bus de direcciones
ACC(32)	Acumulador
ACCB(32)	Buffer acumulador
ACCH	Acumulador alto
ACCL	Acumulador bajo
ALU(32)	Unidad aritmética lógica
ARAU	Unidad aritmética de registros auxiliares
ARB(3)	Buffer de registros auxiliares
ARP(3)	Apuntador de registros auxiliares
MRAF(1)	Bandera activa de repetición de bloque
CALU	Unidad aritmética lógica central

Figure 3–1. Block Diagram of 'C5x DSP – Central Processing Unit (CPU)



Notes: All registers and data lines are 16-bits wide unless otherwise specified.
† Not available on all devices.

D15-D0	Bus de datos
DRB	Bus de direccionamiento directo de memoria de datos
MCS	Pila microcall
MULTIPLIER	Multiplicador
MUX	Multiplexor
P-SCALER (-6,0,1,4)	Desplazador de productos
PLU	Unidad lógica paralela
PROGRAM BUS	Bus de programa
RAM(1)	Bit habilitador del programa RAM
STACK	Pila

3.2.3.1 Unidad de Aritmética y lógica central

La CALU consiste de las siguientes partes:

- Multiplicador paralelo de 16 x 16 bits
- Unidad aritmética lógica (ALU) de 32 bit de complemento a dos
- Acumulador (ACC) de 32 bits
- Buffer acumulador de 32 bits (ACCB)
- Corrimiento de 1, 2 o 4 bits a la izquierda o 6 bits a la derecha
- Registro de corrimiento de 1 a 16 bits a la izquierda
- Registro de corrimiento de 1 a 16 bits a la derecha
- Registro de corrimiento de 1 a 7 bits a la izquierda

Multiplicador, registro de producto (PREG) y registro temporal (TREG0)

El hardware multiplicador de 16 x 16 bit puede calcular y signar o no signar un producto de 32-bit en un solo ciclo de máquina. Está compuesto por tres principales elementos; el registro TREG0 (de 16 bits), el registro (PREG o P) de 32 bits y un arreglo multiplicador. Todas las instrucciones de multiplicación excepto la multiplicación no signada producen un producto signado en el multiplicador, esto es, dos números a ser multiplicados son tratados como números en complemento a dos, y el resultado es un número de 32 bits en complemento a dos.

Una entrada hacia el multiplicador es del registro temporal de memoria mapeada (TREG0), y la otra entrada es del bus de datos o del bus de programa. El resultado de 32 bits del multiplicador es alojado en el PREG y está disponible hacia la ALU. LA ALU usa palabras de 16 bits tomadas de la memoria de datos o derivadas de una instrucción inmediata. LA ALU también puede ejecutar operaciones Booleanas. El resultado de la ALU es almacenado en el acumulador (ACC); el acumulador también puede proveer una segunda entrada hacia la ALU.

La instrucción LT (load TREG0) carga TREG0 desde el bus de datos con el primer operando; la instrucción MPY provee un segundo operando para operaciones de multiplicación. Para ejecutar una multiplicación con un corto o largo operando inmediato, se usa la instrucción con un operando inmediato. Un producto puede ser obtenido cada dos ciclos excepto cuando es usado un operando inmediato largo.

Existen cuatro instrucciones de multiplicación - acumulación en el 'C5x, las cuáles pueden ser procesadas simultáneamente. Los datos de estas operaciones pueden ser transferidas al multiplicador cada ciclo vía los buses de programa y datos. Cuando alguna de estas cuatro instrucciones de multiplicación – acumulación son usadas con la instrucción RPT o RPTZ, la instrucción llega a hacer la función multiplicar –acumular en un solo ciclo. En estas instrucciones de repetición, las direcciones de coeficientes son generadas por el PC mientras las direcciones de datos son generadas por el ARAU. La instrucción RPTZ también limpia el ACC y el PREG para inicializar la operación multiplicación – acumulación.

3.2.3.2. Unidad aritmética lógica (ALU) y acumuladores

La ALU es una unidad aritmética de propósito general que opera con palabras de 16 bits provenientes de la RAM de datos o derivada de instrucciones inmediatas o por el empleo del registro producto (PREG) del multiplicador de 32 bits. Los 32 bits de propósito general de la ALU y ACC implementan un amplio rango de funciones aritméticas y lógicas, la mayoría de cada instrucción se ejecuta en un solo ciclo de reloj. Una vez que una operación es realizada en la ALU, el resultado es transferido hacia el ACC, donde operaciones adicionales, tales como corrimiento, pueden ocurrir.

Los siguientes pasos ocurren en la implementación de una instrucción típica en la ALU.

1) El dato es traído de la memoria sobre el bus de datos.
2) El dato pasa a través del prescaler y la ALU, donde la aritmética es ejecutada, y
3) El resultado es movido al ACC.

La ALU opera con palabras de 16 bits tomadas de la memoria de datos o derivada de instrucciones inmediatas. En adición a las instrucciones de aritmética usuales, la ALU puede desarrollar operaciones Booleanas y en consecuencia, facilitar la habilidad requerida para manipular bits de un controlador de alta velocidad: Una entrada a la ALU es siempre suministrada por el ACC. La otra entrada puede ser transferida del PREG, por el multiplicador, el ACCB, o la salida del prescaler (que ha sido leído de la memoria de datos o del ACC). Después de que la ALU ha desarrollado las operaciones aritméticas o lógicas, el resultado es almacenado en el ACC. Los 32 bits del ACC se pueden dividir en dos segmentos de 16 bits (ACCL y ACCH) para almacenamiento.

El 'C5x soporta operaciones de punto flotante para operaciones que requieran un gran rango dinámico. También ejecuta una variedad de instrucciones, que dependen del estado de la ALU o del ACC. El acumulador Buffer es un registro de 32 bits que acompaña al ACC, para efectuar operaciones de 32 bits, intercambio de datos y comparación de datos de una tabla.

Se pueden realizar corrimientos de o a 16 bits a un dato de entrada, para ello se programa directamente en la instrucción, o en el registro TREG1. El acumulador puede ser corrido a la derecha de 0 a 31 bits en dos ciclos de instrucción o de 0 a 16 bits en un ciclo de instrucción mediante la instrucción BSAR. Se pueden realizar corrimientos al resultado de productos sin el uso de operandos, para ello se utilizan los bits "PM", que es el modo de corrimiento:

PM	RESULTADO
00	No hay corrimiento
01	Corrimiento a la izquierda de un bit
10	Corrimiento a la izquierda de cuatro bits
11	Corrimiento a la izquierda de cuatro bits

Estos corrimientos se usan con las instrucciones:
PAC ; el contenido del registro PR corrido como se especifica en PM es cargado al acumulador.

APAC ; el contenido del registro PR corrido como se especifica en PM es sumado al acumulador y SPAC ; el contenido del registro PR corrido como se especifica en PM es restado del acumulador.

3.2.3.3. Unidad paralela lógica (PLU)

La Unidad Lógica Paralela (PLU) pone en set, clear, test o toggle múltiples bits en un registro de estado – control. Efectúa operaciones lógicas independientes de la unidad ALU y ARAU, es decir, que no utiliza el acumulador para efectuar operaciones. Efectúa operaciones lógicas directas sin afectar al acumulador o al registro producto.

En las operaciones de la PLU, un operando es buscado en memoria dato y el otro proviene del operando inmediato en la instrucción, después de efectuar la operación lógica entre los operandos, el resultado es escrito de nuevo en la localización de la memoria dato direccionado, normalmente, estas instrucciones se hacen en un ciclo de instrucción.

3.2.3.4. Unidad aritmética de registro auxiliar (ARAU)

El registro auxiliar contiene ocho registros auxiliares mapeados (AR0 – AR7), los cuáles pueden ser usados para direccionamiento indirecto de la memoria de datos o para alojamiento de datos temporales. El registro auxiliar de direccionamiento indirecto permite la colocación de un operando de instrucción de una dirección de memoria de datos dentro de un AR. Los registros auxiliares son indicados por un registro auxiliar indicador de 3 bits, que es cargado con un valor de 0-7, designando de A0 hasta A7, respectivamente. Los registros auxiliares (AP's) y (ARP) pueden ser cargados desde la memoria de datos, el ACC o el PREG o por un operando inmediato definido en la instrucción. Los contenidos de los AR's pueden ser almacenados en la memoria de datos o usados como entradas hacia la CALU.

El registro auxiliar (AR0 – AR7) es conectado con la unidad aritmética de registro auxiliar (ARAU), la ARAU puede indicar el actual AR mientras la localidad de la memoria de datos está siendo direccionada; este se indica ya sea con ±1 o con el contenido del registro índice (INDIX). Como un resultado, no es necesitada para manipulación de direcciones cuando se accessa listas de información; esta está libre para otras operaciones en paralelo. Para manipulación de direcciones más avanzadas, tales como ordenamiento de direccionamiento multidimensional, la CALU puede directamente leer de o escribir hacia ARs.

El ARAU actualiza los AR's durante la fase de decodificación (segunda fase) de el pipeline, mientras la CALU escribe durante la fase de ejecución (cuarta fase). Por lo tanto, las dos siguientes instrucciones inmediatas de la CALU no usarán el mismo AR para la generación de direcciones.

El ARAU puede servir como una unidad aritmética de propósito general, ya que el registro auxiliar puede comunicarse directamente con la memoria de datos. El ARAU implementa aritmética de 16 bis no signada, mientras que el CALU implementa aritmética de 32 bits con complemento a dos. Las instrucciones BANZ y BANZD permiten que los registros AR's sean usados como contadores de ciclo.

3.2.3.5. Registros mapeados

El TMS320C50 tiene 28 registros mapeados en la parte baja de la memoria dato, se ubican de la siguiente forma;

Registro	Dir. Hex.	Descripción.
---------	0-3	Reservado
IMR	4	Registro de máscara de interrupción
GREG	5	Registro para localización global de memoria
IFR	6	Registro de banderas de interrupción
PMST	7	Registro de modo de estado del DSP
RPTC	8	Registro contador de repetición
BRCR	9	Contador de repetición de bloque
PASR	A	Dirección inicial del bloque de repetición
PAER	B	Dirección final del bloque de repetición
TREG0	C	Registro temporal para multiplicando
TREG1	D	Reg. Temp. Para contador de corrimiento dinámico
TREG2	E	Reg. Temp. Usado como apuntador de bit en prueba dinámica de bit
DBMR	F	Manipulación dinámica de bit
AR0-AR7	10-17	Registros auxiliares
INDX	18	Registro índice
ARCR	19	Registro auxiliar de comparación

CBSR1	1A	Dirección inicial de buffer circular 1
CBER1	1B	Dirección final de buffer circular 1.
CBSR2	1C	Dirección inicial de buffer circular 2
CBER2	1D	Dirección final de buffer circular 2
CBSR	1E	Registro de control de buffer circular
BMAR	1F	Registro de direccionamiento dinámico
	20-35h	Periféricos mapeados en memoria
	36-4Fh	Reservado
	50-5F	Puertos mapeados en memoria

Registro de comparación auxiliar (ARCR)

El registro (ARCR) de 16 bits es usado para comparación de direcciones límites. La instrucción CMPR compara el ARCR con él AR seleccionado y coloca el resultado de la comparación en el bit TC o ST1.

Registro de movimiento de direcciones en bloque (BMAR)

El BMAR de 16 bits guarda el valor de una dirección para ser usada en movimiento de bloques y operaciones de multiplicación – acumulación. Este registro provee direcciones de 16 bits para un segundo direccionamiento indirecto.

Registro de repetición de bloques (RPTC, BRCR, PASR, PAER)

Los registros contadores de repetición (RPTC) guardan el contador de repetición de un operando de una sola instrucción de repetición y es cargada por las instrucciones RPT y RPTZ. Aunque el RPTC es un registro de memoria mapeada, se debe evitar escribir en este registro ya que esto podría causar resultados indeseables.

El registro contador de repetición en bloques (BRCR) guarda el valor calculado de repetición de bloque. Este valor es cargado antes de que se inicie la operación de repetición de bloque. El valor puede ser cambiado mientras una repetición de bloque está en progreso, sin embargo se debe tener cuidado de no crear ciclos de repetición infinitos. El registro (PASR) indica la dirección de 16 bits en la que se encuentra el código de inicio de repetición en bloques. Mientras que el registro (PAER) indica la dirección en la que se encuentra el código de término de repetición en bloques.

Registros de buffer circular (CBSR1, CBER1, CBSR2, CBER2 CBCR)

Los dispositivos 'C5x soportan dos buffer circulares operando en conjunto con registros auxiliares de uso especifico. Dos buffer circulares de inicio de 16 bits (CBSR1 y CBSR2) indican la dirección donde inicia el buffer circular. Dos buffer circulares de término (CBER1 y CBER2) indican la dirección donde terminan los buffer circulares. El registro de buffer circular de control (CBCR) controla la operación de los buffer circulares e identifica los registros auxiliares a ser usados.

Registro dinámico de manipulación de bits (DBMR)

El DBMR de 16 bits es usado en conjunto con la PLU como un registro dinámico mascarable.

Registro global de asignación de memoria (GREG)

El GREG de 16 bits asigna partes del espacio de datos local como memoria global y define que cantidad del espacio de datos local será cubierto por espacio global de datos.

Registro índice (INDX)

El INDX de 16 bits es usado por la ARAU para modificar la dirección en los AR's durante direccionamiento indirecto. La ARAU puede sumar o sustraer el valor alojado en el registro INDX del actual AR como parte de una operación de direccionamiento indirecto.

Registro de instrucciones (IREG)

El IREG de 16 bits guarda el código de máquina de las instrucciones a ser ejecutadas. El IREG es usado durante el programa de control.

Registros de interrupciones (IMR, IFR)

El registro de interrupción mascarable (IMR) de 16 bits individualmente enmascara interrupciones en los tiempos requeridos. El registro de bandera de interrupción (IFR) de 16 bits indica el estado actual de las interrupciones. Los estados de las interrupciones son actualizados además por el IMR y el bit INTM en el ST0.

Registro de procesador de modo de estado (PMST)

El PMST contiene los estados y control de información para el dispositivo 'C5x. En un capítulo posterior se describe más detalladamente.

Registro de productos (PREG)

El PREG de 32 bits guarda el resultado de una operación de multiplicación. La palabra alta y baja pueden ser accesadas individualmente.

Registros con interfase con puerto serial (SPC, DRR, XSR, RSR)

Cinco registros de operación y control con la interfase con el puerto serial. El registro de control del puerto serial (SPC) de 16 bits, contiene los bits de modo de control y estado del puerto serial. El registro receptor de datos (DRR) de 16 bits guarda el dato serial entrante, y el registro de transmisión de datos (DXR) guarda el dato serial saliente. El registro de corrimiento de transmisión (XSR) de 16 bits, controla el corrimiento del dato del DXR hacia el pin e salida. El registro de corrimiento de recepción (RSR) de 16 bits controla el alojamiento del dato de la entrada hacia l DRR.

Registros de software programable y estados de espera (PDWSR, IOWSR, CWSR)

El software de estados de espera es determinado por tres registros. Estos registros sirven para diferentes propósitos en diferentes dispositivos. En todos los dispositivos 'C5x el registro de estado de espera de programa – dato (PDWSR) de 16 bits contiene el conteo de estado de espera para los ocho bloques de 16k de palabras de la memoria de datos y programa. El PDWSR es internamente dividido en 8 campos de dos bits de estados de espera asignados para cada bloque de palabras de 16k. El espacio I/O dentro del registro de estado de espera I/O de (IOWSR) de 16 bits bajo el control del registro de control de estado de espera (CWSR) de 5 bits. El CWSR determina el rango de estados de espera seleccionado.

Registros de estado (ST0, ST1)

Los dos registros de estado de 16 bits contienen los bits de estado y control para el CPU. Estos dos registros los mencionaremos con más detalle, ya que son los que nos indican el estado del DSP.

Registros temporales (TREG0, TREG1, TREG2)

El registro TREG0 de 6 bits contiene uno de los multiplicandos para el multiplicador. TREG0 también puede ser cargado vía la CALU con las siguientes instrucciones: LT, LTA, LTD, LTP, LTS, SQRA, SQRS, MAC, MACD, MADS y MADD. Los 5 bits del TREG1 un contador de corrimiento dinámico para el prescaling de desplazamiento. Los 4 bits de TREG2 guardan un bit dinámico de dirección para la instrucción BIT.

Registros de tiempo (TIM, PRD, TCR)

Tres registros para control y operación del tiempo. El registro contador de tiempo (TIM) proporciona el actual conteo del tiempo. El registro de periodo de tiempo (PRD) define el periodo de tiempo. El registro de control de tiempo (TCR) de 16 bits, controla las operaciones de tiempo.

Registros de puerto serial TDM (TRCV, TDXR, TSPC, TCSR, TRTA, TRAD, TRSR)

El multiplexor de división de tiempo (TDM) de la interfase con el puerto serial es una característica de la interfase con el puerto serial y soporta aplicaciones que requieren comunicación serial en un medio de multiprocesamiento. Seis registros controlan y operan el TDM de la interfase de puerto serial. Los 16 bits del registro de control de puerto serial TDM (TSPC) contienen los bits de modo de control y estado de la interfase de puerto serial TDM. El registro receptor de datos TDM (TRCV) de 16 bits contiene el dato serial TDM entrante, y el registro de transmisión de datos TDM (TDXR) de 16 bits contiene el dato serial saliente. El registro de corrimiento de recepción de datos TDM (TRSR) controla el almacenamiento de datos, del pin de entrada hacia el TRCV. El registro de selección de canal TDM (TCSR) de 16 bits en que slot transmite cada dispositivo 'C5x. El registro de dirección recepción – transmisión TDM de 16 bits (TRTA) especifica la dirección de recepción de los ocho LSB's (RA0, RA7) del dispositivo 'C5x y las ocho direcciones de transmisión MSB's (TA0, TA7). El registro de dirección de recepción TDM (TRAD) de 16 bits contiene información en cuanto al estado de la línea de dirección del TDM.

Registros de estado

En el TMS320C5050 existen cuatro registros de estado, los registros de estado ST0, ST1, contienen el estado de varias condiciones y modos de operación, mientras que los registros PMST y CBCR contiene estados extras y control de la información. Los registros de estado ST0, ST1 y PMST tienen asociado un registro sombra para salvarse automáticamente cuando ocurre una interrupción. En la rutina de atención de interrupción se usa alguna instrucción (RETI o RETE) para devolver a los registros sus valores originales antes de la llamada a subrutina. Estos registros se encuentran configurados como sigue:

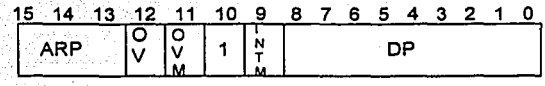


Figura 3.2 ST0

DP Apuntador de página de memoria dato.

INTM Habilita interrupciones mascarables.

OVM Bit de modo de sobre flujo.

OV Bit de bandera de sobre flujo.

ARP Apuntador de registros auxiliares.

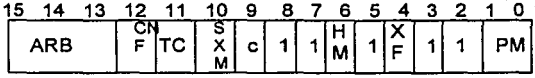


Figura 3.3 ST1

PM Modo de corrimiento de registro producto.

XF Bit de estado de pin externo.

HM Bit de modo sostenido.

C Bit de carry.

SXM Bit de modo de signo extendido.

TC Bandera de control de prueba de bit.

CNF Bit de control de configuración de memoria RAM interna.

ARB Buffer apuntador de registros auxiliares.

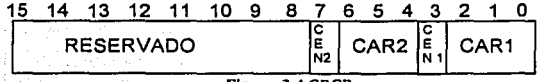


Figura 3.4 CBCR

CAR1 Tres bits que identifican el AR que se usa en buffer circular 1.

CAR2 Tres bits que identifican el AR que se usa en buffer circular 2.

CEN1 Habilita (1) o deshabilita (0) el buffer circula 1.

CEN2 Habilita (1) o deshabilita (0) el buffer circular 2.

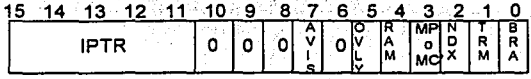


Figura 3.5 PMST

BRA Bandera de repetición de bloque, en uno indica la repetición de un bloque.

TRM Habilita múltiples TREGs, en habilita el uso de los registros TRG1 y TREG2.

NDX Habilita el registro índice.

MP/mc Modo microprocesador (1) o microcomputadora (2).

RAM Habilita el programa en memoria RAM.

OVLY Habilita el acceso de programa en memoria RAM, si es uno, el bloque de memoria es mapeado en el espacio de dato, si es cero, el bloque de memoria no es direccionable en memoria dato, fija a cero el reset.

AVIS En uno permite la visualización de líneas de dirección externas cuando el programa efectúe direccionamientos internos.

IPTR Apuntador de vector de interrupción. Permite el mapeo de vectores de interrupción.

3.3. Memoria

El diseño del 'C50 está basado en una arquitectura Harvard modificada, la cuál presenta múltiples espacios de memoria que puede ser accesada por dos buses paralelos, esto hace posible acceder al programa y datos simultáneamente. Estos dos buses paralelos son el bus de datos (DB)y el bus de programa (PB).

El total de direcciones de memoria del 'C5X es de 224K de 16 bits de palabra, el espacio de memoria esta dividido en 4 segmentos individuales

- 64 K de memoria de programa.
- 64 K de memoria de datos local.
- 64 K de memoria para puertos de entrada / salida.
- 32K de memoria global de datos.

Los 64K de memoria de programa contienen las instrucciones a ser ejecutadas, 64K de palabra de memoria de datos, almacenan las instrucciones, 32K de 16 bits de palabra de la memoria global de datos puede compartir datos con otros procesadores, con el sistema o puede servir para espacio adicional de datos, los 64K de palabra de memoria de puertos de entrada-salida son interfaces a memorias externas periféricos y también pueden servir de memoria de datos.

La figura 3.6 muestra cómo se encuentra organizada la memoria del TMS320C50.

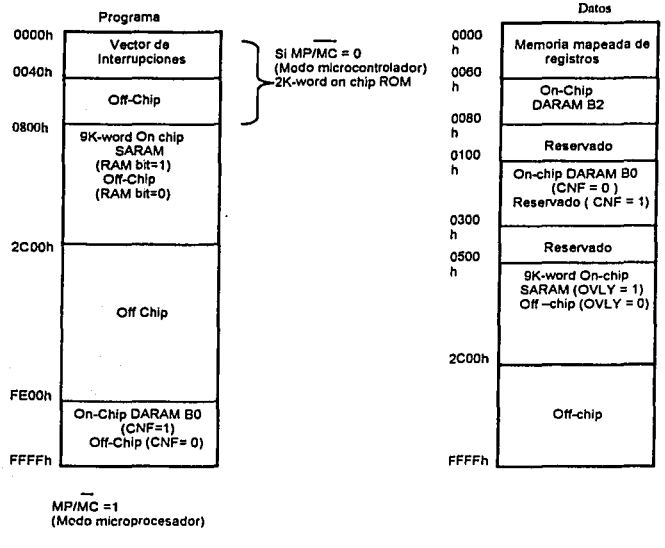


Figura 3.6 Organización de la memoria del TMS320C50

3.3.1. Memoria de programa

Las direcciones del espacio de memoria de programa de 64k-Word incluyen el on-chip ROM, SARAM, y la DARAM. Cuando las celdas de la memoria son mapeadas dentro del espacio de programa, el 'C50 automáticamente las accesa cuando estas direcciones están dentro de sus límites.

Cuando la CALU genera una dirección fuera de estos límites, el 'C50 automáticamente genera un acceso externo (off-chip). Las ventajas de operar desde una memoria interna (on-chip) son;

- Alto rendimiento, ya que no son requeridos estados de espera para memorias externas lentas.
- Costo más bajo que las memorias externas.
- Bojo consumo de potencia.

La ventaja de operar desde una memoria externa (off-chip) es la habilidad para acceder a espacios largos de memoria.

La memoria de programa puede estar tanto como on-chip u off-chip. Esta configuración se determina por el pin MP/MC. Si el pin se pone en nivel alto, el dispositivo es configurado como microprocesador, el on-chip ROM no es direccionado. Si éste pin está en bajo, el dispositivo es configurado como micro computador, y el on-chip ROM es habilitado.

3.3.2. Modos de direccionamiento

El TMS320C50 puede direccionar un total de 64k-palabras de memoria de programa y 64k de memoria de datos. Los modos de direccionamiento del 'C5X son los siguientes

- Modo Inmediato
- Modo directo
- Modo indirecto
- De registros mapeados
- Direccionamiento circular

3.3.2.1. Modo de direccionamiento inmediato corto y largo

Cuando un operando inmediato es empleado, el operando esta contenido en la instrucción, el modo inmediato corto acepta constantes de 8 bits, en consecuencia el código de instrucción es de 16 bits. El modo largo acepta constantes de 16 bits, entonces el código de instrucción será de 32 bits. Se expresa con un "#" antes de la constante. Por ejemplo;

LACL #0FFA5h,4 ; Carga el número 0FFA5 en la parte baja del acumulador, con corrimiento de 4 bits a la izquierda.

3.3.2.2. Modo de direccionamiento directo

A este modo se le llama directo ya que en el código de la instrucción esta contenida una parte de la dirección de datos a operar. En este modo la memoria total de datos esta dividida en páginas. Los 9 bits del apuntador de pagina DP pueden apuntar a 512 paginas de 128 palabras cada una (64k de palabra). El dato de memoria direccionado es especificado por los 7 bits menos significativos de la instrucción para apuntar a la palabra deseada dentro de la página. Antes de usar este modo e necesario cargar el número de pagina con el registro D, esto se hace con la instrucción LDP #P.

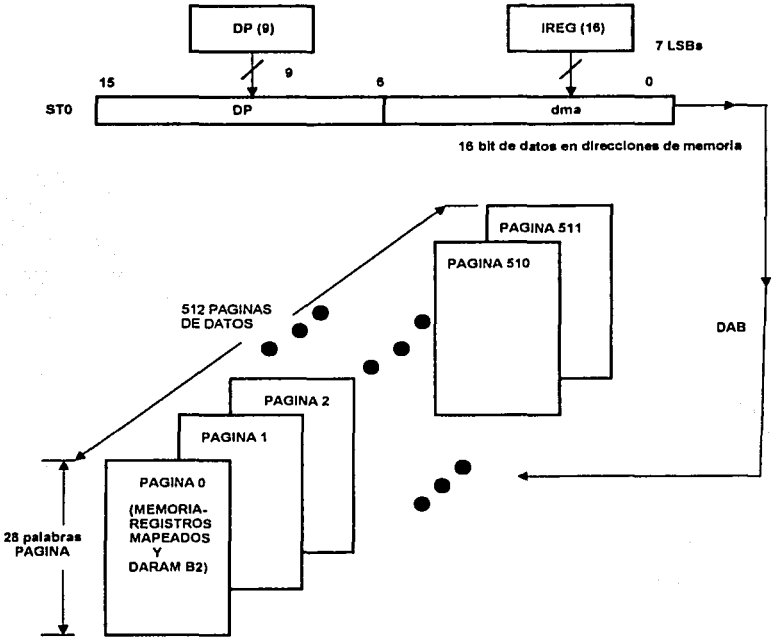


Figura 3.7 Direccionamiento directo usando la memoria de datos divida en 512 partes llamadas paginas

Algunos ejemplos son los siguientes;

```
LACL X      ; Carga la parte baja del ACC con el valor de X.
ADD X1      ; Suma al ACC el valor de X1.
```

3.3.2.3. Modo de direccionamiento Indirecto

En este modo la dirección del dato a operar esta en otro registro este modo es muy eficiente para direccionamiento de tablas y arreglos, en la instrucción no se especifica ninguna dirección de memoria. Los 16 bits de la dirección son seleccionados por el registro auxiliar en uso direccionando el dato de memoria a través del bus de registros auxiliares.

Registros auxiliares.

El TMS320C50 posee 8 Registros auxiliares: AR0-AR7 los cuales pueden ser utilizados para direccionamiento indirecto, o en almacenamiento temporal de datos Estos registros son seleccionados por un apuntador de registros auxiliares de tres bits ARP cargándolo con los valores de 0 a 7 para designar los daos desde AR0 a AR7 respectivamente, los registros auxiliares deben inicializarse y seleccionarse antes de ser utilizados, para cargarlos se utiliza la instrucción LAR y se seleccionan con la instrucción MAR.

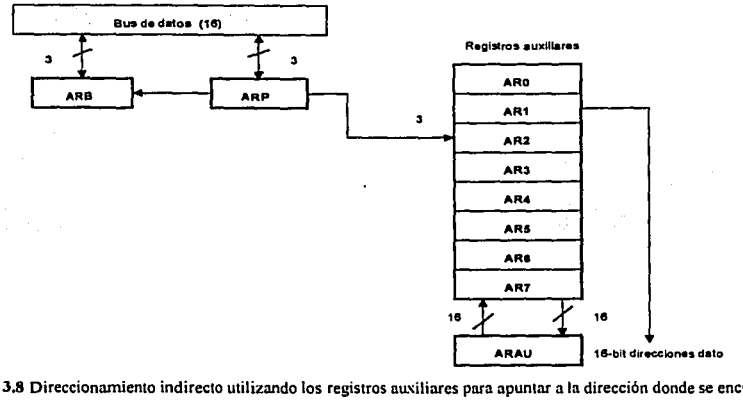


Figura 3.8 Direccionamiento indirecto utilizando los registros auxiliares para apuntar a la dirección donde se encuentra el dato

En las siguientes instrucciones se usa el direccionamiento indirecto:

```
LAR AR1,XN    ; Apunta a AR1 a la dirección de XN.
MAR *,AR1     ; Marca a AR1 como registro auxiliar en uso.
MAC #B0,*+    ; Multiplica B0 por lo que apunta AR1 y además acumula.
```

3.3.2.4. Direccionamiento de registros mapeados

Este direccionamiento permite un acceso rápido a registros mapeados en memoria. Opera similar al direccionamiento directo y se utiliza para acceder a los registros mapeados, en este caso se obliga los 9 bits más significativos de la dirección que sean cero, independientemente del valor que tenga el registro de página, esto permite hacer un acceso directo a estos registro sin necesidad de hacer cambio de página. Las instrucciones utilizadas para este direccionamiento son SAMM y LAMM.

Por ejemplo:

```
LAMM DRR    ; Carga ACC con el contenido de DRR; registro de recepción
SAMM DXR    ; Escribe en DXR (registro de transmisión), el contenido de ACC.
```

3.3.2.5. Direccionamiento circular

El direccionamiento circular es utilizado para direccionar eficientemente una ventana de datos que se están procesando repetidamente, los siguientes registros son utilizados.

CBSR1	1A	Dirección inicial del buffer circular 1
CBER1	1B	Dirección final del buffer circular 1
CBER2	1C	Dirección del buffer circular 2
CBCR	1E	Registro de control d buffer circular

La figura 3.9 muestra como se encuentra el registro CBRCR, que es de 8 bits, los cuáles se definen como;

Bit	Nombre	Función
0-2	CAR1	Identifica que registro auxiliar se esta utilizando para el buffer circular 1
3	CENB1	Habilitar buffer circular 1 con un uno, con cero lo deshabita
4-6	CCAR2	Identifica que registro auxiliar es utilizado por el buffer circular 2
7	CENB!	Habilita el buffer circular 2 con un uno, con cero lo deshabilita

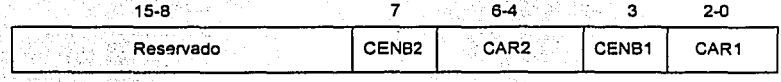


Figura 3.9 Registro mapeado CBCR de 16 bits

Antes de utilizar el modo de direccionamiento circular hay que cargar los registros de inicio y final de buffer y escribir los registros de control CBCR estos registros se pueden cargar con las instrucciones:

```
LACC #0100h
SAMM CBSR1
LACC #010Ah
```

Estos registros también pueden cargarse con la instrucción SPLK que significa carga en paralelo una constante inmediata

```
SPLK #100h,CBSR1
SPLK #10Ah,CBER1
```

3.3.3. Manejo de la memoria

El 'C50 cuenta con varias instrucciones para la administración de la memoria, como son;

- Instrucciones de movimiento de bloques de programa y datos.
  - BLDD Mueve un bloque dentro de la memoria de datos.
  - BLDP Mueve un bloque de la memoria de datos a la memoria de programa.
  - BLPD Mueve un bloque de la memoria de programa a la memoria de datos.
- Instrucciones de transferencia de palabras de programa y datos
  - La instrucción leer tabla (TBLR) lee palabras de la memoria de programa dentro de la memoria de datos.
  - La instrucción escribir tabla (TBLW) escribe palabras de la memoria de datos hacia la memoria de programa.

Estas instrucciones transfieren datos de las siguientes formas:

- De la memoria externa de datos hacia la memoria externa de datos
- De la memoria externa de datos hacia la memoria interna de datos
- De memoria interna a memoria interna
- De memoria interna a memoria externa.

Interrupciones

Los vectores de interrupción son direccionados en espacio de programa. La siguiente tabla presenta las direcciones de los vectores de interrupción del C50.

Nombre	Ubicación		Prioridad	Función
	Dec.	Hex.		
RESET'	0	0	1(Highest)	Señal de reset externa no mascarable
INT1'	2	2	3	Interrupción de usuario externo #1
INT2'	4	4	4	Interrupción de usuario externo #2
INT3'	6	6	5	Interrupción de usuario externo #3
TINT	8	8	6	Interrupción de tiempo interno
RINT	10	A	7	Interrupción de recepción de puerto serial
XINT	12	C	8	Interrupción de transmisión de puerto serial
TRNT	14	E	9	Interrupción de recepción de puerto TDM
TXNT	16	10	10	Interrupción de transmisión de puerto TDM
INT4'	18	12	11	Interrupción de usuario externo #4
----	20-23	14-17	N/A	Reservado
HINT	24	18	----	HINT ('c57 only)
----	26-33	1A-21	N/A	Reservado
TRAP	34	22	N/A	Instrucción de trampa de software
NMI'	36	24	2	Interrupción no mascarable
----	38-39	26-27	N/A	Reservado para prueba y emulación
----	40-63	28-3F	N/A	Interrupciones por software

Nota: el símbolo ' indica que se habilita con cero.

3.4. Interfaz a periféricos y puerto serie.

Los periféricos que maneja el tmsc50 son controlados por algunos registros mapeados, para la inicialización de estos periféricos, en sus registros deben escribirse las palabras necesarias para que operen correctamente. Dentro de estos periféricos se encuentran el puerto serie, el puerto serie multiplexado por división de tiempo (TDM), 64 k de puertos paralelos y el timer. Como en el presente trabajo trabajaremos sólo con el puerto serie, mencionamos este con más detalle.

3.4.1. Puerto serie

El tmsc50 cuenta con dos puertos serie que son síncronos a una velocidad de 7.14 Mbits/s. Uno de ellos es multiplexado por división de tiempo (TDM) y sirve para multiprocesamiento con otros dispositivos, si no se utiliza, se convierte en un segundo puerto serie síncrono. Para su operación cuenta con los siguientes pines externos;

CLKX Señal de reloj de transmisión.
CLKR Señal de reloj de recepción.
DX Señal de transmisión de dato serial, transmite el dato actual.
DR Señal de recepción de dato serial, recibe el dato actual.
FSX Señal de sincronización de frame de transmisión, inicia transferencia de un frame.
FSR Señal de sincronización de frame de recepción.

Este puerto se encuentra organizado como se indica en la figura 3.10.

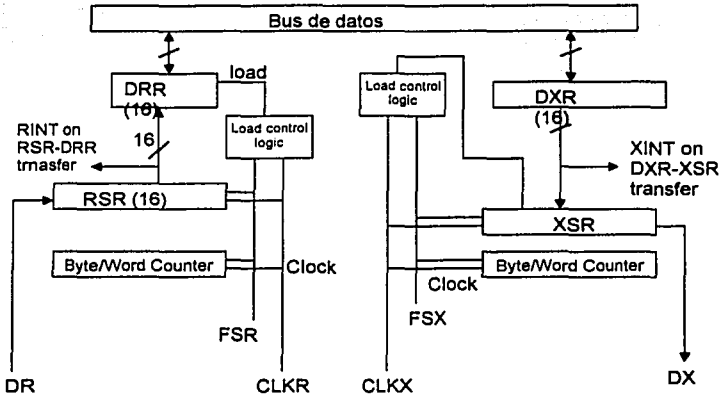


Figura 3.10 Diagrama de bloques del puerto serie.

Para la configuración del puerto serie se usan los siguientes registros;

SPC Registro de control de puerto serie.
DXR Registro de transmisión de datos.
XSR Registro de corrimiento en la transmisión.
RSR Registro de corrimiento en la recepción.

Para la recepción, un dato del convertidor A/D y D/A es recibido en el pin DR, se corre dentro del registro RSR, el cuál es copiado en el registro de recepción de dato (DRR) y el dato puede ser leído por el DSP. Una vez completada la copia de RSR a DRR existe una transición 0-1 en el bit de recepción lista (RRDY bit 10) del registro de control SPC y a la vez se genera una interrupción de recepción del puerto serie (RINT).

El proceso de transmisión es muy similar, es decir que un dato a transmitir es escrito en el registro DXR el cuál copia el dato al registro XSR que efectúa el corrimiento de bits para hacer la transmisión serial sobre el pin DX, tan pronto como se copia DXR a XSR se permite escribir otro dato en el registro DXR, a la vez ocurre una transición 0-1 en el bit de transmisión lista (XRDY bit 11) del registro de control SPC y se genera una interrupción de transmisión de puerto serie (XINT).

Los 16 bits del registro de control SPC son los siguientes;

Bit	Nombre	Descripción
0		Reservado
1	DLB	Digital loopback
2	F0	Longitud de palabra 0 = 16 bits, 1 = 8 bits
3	FSM	Modo de sincronía de frame 0 = modo continuo, 1= un pulso zirconio/palabra
4	MCM	Fuente de reloj 0 = CLKX 1 = CLKOUT1/4
5	TXM	0 = FX es entrada, 1 = FSX es salida
6	XRST	0 = reset al transmisor, 1 = transmisor en operación
7	RRST	0 = reset al receptor, 1 = receptor en operación
8	IN0	Nivel de entrada en CLKR
9	IN1	Nivel de entrada en CLKX
10	RRDY	1 = Dato en recepción listo
11	XRDY	1 = dato en transmisión listo
12	XSREMPTY	0 = transmiter underrun
13	RSRFULL	1 = sobreflujo en recepción
14	SOFT	Determina el estado del puerto serial cuando se inserta un break point en el debuger.
15	FREE	

Donde los 14 y 15 significan:

FREE	SOFT	ACCION
1	x	Corrida libre
0	0	Paro inmediato
0	1	Paro después de completar palabra

Descripción del DSK y del ensamblador

4.1. Descripción del DSK

El DSP starter kit del TMS320C50 es una tarjeta sobre la cuál se encuentra montado el DSP y además otros componentes que sirven para que pueda operar el DSP, esta tarjeta se conecta con una computadora personal a través de un puerto serial, la tarjeta incluye los siguientes componentes:

- Un DSP TMS320C50
- Un Convertidor analógico-digital y digital-analógico programable (TLC32040)
- Una Memoria PROM
- Entradas y salidas analógicas
- Rectificador de voltaje
- Un reloj a 50 Mhz.

La memoria interna del DSK es únicamente la que posee el TMS320C50, el programa kernel esta contenido en la memoria PROM de 32K el cual se utiliza para arrancar al TMS320C50 creando un ambiente y no se puede acceder después.

El circuito TLC32040 es un convertidor analógico-digital y digital-analógico. Este convertidor es la interfase entre el DSP y el mundo analógico. Este convertidor recibe la señal analógica y la muestrea para pasarla al puerto de recepción del DSP. El voltaje que puede recibir es de ±2.5 volts y la convierte en secuencias digitales de 14 bits con frecuencia de muestro variable y filtro integrado, la máxima velocidad de muestreo que se puede programar es de es de 19.2 khz, que es suficiente para nuestros propósitos de prácticas en el laboratorio. Además la tarjeta posee seis cabezas conectadas a los puertos para implementar diseños propios del usuario o expandir la capacidad del sistema.

Para su funcionamiento es necesario conectar la tarjeta a la PC a través de un cable RS232 con entrada DB9 y alimentarla con un trasformador de 9 Vc.a a 250 *m*A. La tarjeta puede alimentarse antes o después de hacer las conexiones. La figura 4.1 muestra un esquema de la tarjeta, y en la figura 4.2 se muestra un esquema general de la conexión con la PC.

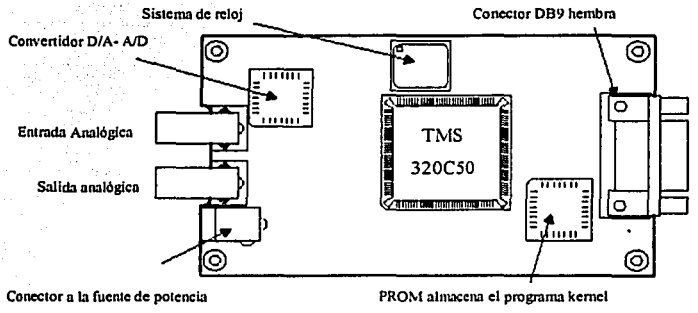


Figura 4.1. Tarjeta de evaluación TDMSC50

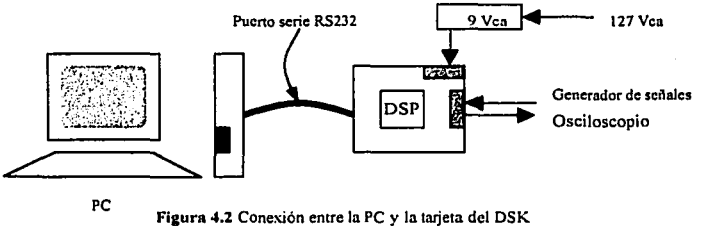


Figura 4.2 Conexión entre la PC y la tarjeta del DSK

4.2. Creación de código fuente

Para la creación del código fuente se cuenta con el software que provee el fabricante. Este software trabaja bajo el ambiente de MS-DOS y permite convertir los neumónicos de un programa fuente, en un código directamente ejecutable. Este programa fuente se debe escribir en algún editor de textos compatible con ASCII, como el edit de MS-DOS y guardarlo con la extensión .asm. Luego, estando en el directorio en el que se encuentra el software se ensambla el código fuente con el comando *DSK5A*. De esta forma se crea un archivo ejecutable que tiene la extensión .dsk, siempre y cuando no haya errores en el código fuente.

El DSK ensamblador, no tiene que pasar por un proceso de ligado para crear un archivo de salida. En lugar de eso el DSK, usa directivas especiales para ensamblar código a una dirección absoluta, durante

la fase de ensamblado, el software genera el programa en código objeto, esto permite efectuar una simulación del mismo en ambiente de ventanas.

4.3. Características del lenguaje ensamblador

El proceso de ensamblado se realiza en dos pasadas en la primera el ensamblador mantiene un contador de localización la cual define la dirección de memoria de programa asignada a la palabra resultante en código objeto, la segunda pasada el ensamblador produce el código objeto que corresponde al código de operación asignándole su respectiva palabra.

Un programa fuente DSK consiste en expresiones que constan de cuatro campos

-Etiquetas

-Comandos

-Operandos

-Comentarios

Comentarios

Las expresiones que contienen (*) o punto y coma (;) corresponden a comentarios, una línea de comentarios puede extenderse hasta 80 caracteres, el ensamblador por encíma de esta cantidad marca error, el (*) se usa cuando el comentario empieza en la columna uno de cualquier renglón.

Etiquetas

Inicia en la primera columna y puede contener hasta 26 caracteres alfanuméricos (A-Z, a-z 0-9) y el primer carácter no debe ser un número. La etiqueta es opcional y al igual que en otros ensambladores se utiliza para indicar hacia donde se transfiere la información.

Campo de comandos

El campo de comandos deberá empezar en la columna uno o seguir a una etiqueta, El campo de comandos puede contener

-Mnemónicos

-Directivas del ensamblador

4.4. Ensamblador

El código fuente puede ser editado en cualquier tipo de editor ASCII, el código debe tener una extensión (.asm) una vez editado el código se ensambla con las siguiente sintaxis;

-dsk5a [nombre].asm ; genera un archivo ejecutable con extensión dsk, Si no existen errores

-dsk5a [nombre].asm –l ; Además de un archivo dsk genera un archivo de salida con extensión .lst

-dsk5a [nombre].asm –k Genera un archivo de salida no importando si existen errores

Directivas del ensamblador

Al igual que otros ensambladores, las directivas no generan código; son ordenes que le indican como y donde generar código al ensamblador, las directivas definen símbolos y constantes.

Directivas que definen secciones

.data ; Ensambla en memoria de datos (la memoria de datos contiene datos iniciales)

.ds ; Ensambla en memoria de datos (inicializa la dirección donde están los datos)

.entry ; Inicializa las direcciones del contador de programa cuando se carga el archivo

.ps ; Ensambla datos en la memoria de programa (inicializa una dirección)

Directivas que hacen referencia a otros archivos

Cuando un programa es muy usado el DSK permite incluirlo en otros programas a través de las directivas;

.copy ; Incluye expresiones de otro archivo en el archivo actual

.include ; Incluye argumentos de otro archivo

Directivas que inicializan constantes

.bfloat val1,val2,.... ; Inicializa una constante de 16 bits de exponente y 32 de mantisa.

.bbyte val1,val2,... ; Inicializa una o más variables de 8 bits de acuerdo al estándar IEEE.

.double val1,val2,.. ; Inicializa una o más constantes de 64 bits.

.qxx val1,val2,... ; Inicializa una o más palabras de 16 bits en complemento a 2 con un

punto hipotético desplazado xx bits de LSB.

.word val1,val2,... ; Inicializa una o más variables de 16 bits.

Miscelánea de directivas

.end ; Finaliza un programa, es opcional.

.set ; Iguala una variable simbólica, no genera código.

.mmregs ; Habilita los registros mapeados de memoria para utilizarlos dentro del programa.

4.5 Uso del depurador

El depurador es una interfase que ayuda al usuario a desarrollar y probar programas. La forma de invocar al depurador es escribir dsk5d . La desventaja de esta interfase es la necesidad de tener conectada la tarjeta a la PC, a diferencia de otras interfaces, este depurador solo es depurador y no es simulador.

El ambiente esta constituido por un menú principal, código ensamblado, registros, y vigilante, la barra de menú se activa presionando la tecla iluminada, para abandonar cada submenú presionar ESC. Los principales comandos del depurador son;

LD	Carga un archivo con extensión DSK, si esta en otro directorio es necesario especificar la ruta
F5 ó X	Para correr un programa
F8	Para correr un programa paso por paso
Q	Para salir
R	Reset al TMS320C50
DA	Despliega la memoria a partir de una localidad especificada por el usuario
DAI	Despliega la memoria en entero
DAH	Despliega la memoria en hexadecimal
BA	Detiene un programa hasta la dirección indicada por el usuario
I	Inicia de nuevo el programa cargado

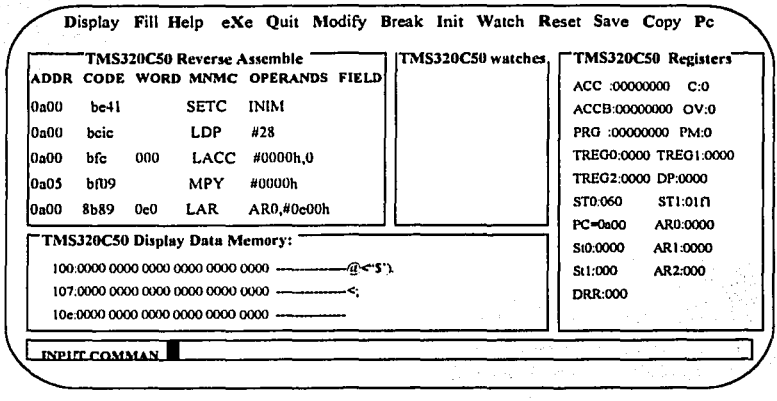


Figura 4.3 Esquema de la pantalla del depurador

A continuación se presenta la estructura o bloques que puede tener un programa en ensamblador para el C50.

```
****************
*Comentarios
****************
  .mmregs     ; El ensamblador reconoce las abreviaturas de los registros mapeados.
  .ds[dir hex] ; Define el inicio del segmento de datos con una dirección hexadecimal.

N  .word  1  ; Asigna un el valor de uno a la variable N. Es una palabra de 16 bits.
x  .word  0 ;
y  .word  2
z  .q8   2.75 ; Convierte el 2.75 a formato q8 y lo escribe en memoria.

* definición de  vectores de interrupcione
.ps804
```

```
*****************************************
*Bloque de código o el programa principal
*****************************************
   .ps 0a00h  ; Inicio del programa
   .entry     ; Inicia el código de programa principal
   -
   -
   -          ; instrucciones del ensamblador C50
   -
   -
************************
* Bloque de subrutinas
************************
   SUB_INIT
         -
         -
         -
         -
      RETE
   SUB1
        -
        -
        -RET

.end       ; Fin del programa.
```

4.6 Inicializar el convertidor TLC32040

El controlador de interfase analógica (AIC) es un dispositivo que hace posible que los algoritmos diseñados puedan realizarse en tiempo real con las siguientes consideraciones;

- Rango de voltaje de entrada analógica de ±2.5 Volts.
- Frecuencia de muestro de hasta 19.2 khz.

- Filtro corrector pasa bajas a la salida del convertidor

El AIC se configura mediante un protocolo serial, posee un pin de reset, necesita también una base de tiempo. Estas cuestiones fueron resultas en la arquitectura de la tarjeta como sigue;

1. -Se empleo un puerto serial para comunicarse con el AIC
2. -EL timer 0 se empleo como base de tiempo

Este dispositivo es capaz de realizar intercambio de información con otros dispositivos mediante un protocolo serial (es decir una taza variable de transmisión), con una longitud de palabra de 16 bits con dos bits de configuración de intercambio de información.

Tanto como para la transmisión como para la recepción los registros TX y RX deben configurarse a través de dos parámetros llamados TA y RA estos se calculan con las siguientes formulas;

$$f_s = \frac{10[Mhz]}{2TATB}$$

$$f_c = \frac{62.5[khz]}{TA}$$

Donde;

f_s = frecuencia de muestreo
f_c =Frecuencia a la cual el AIC no permite el paso de frecuencia

Por lo general se escoge valores de frecuencias mayores a $\frac{f_s}{2}$ para evitar el efecto de submuestreo del convertidor, en la frecuencia de corte del convertidor o f_c, el rango de valores que pueden tomar los registros TA y TB son los siguientes:

$$4 \leq TA \leq 31^{[1]}$$

$$2 \leq TB \leq 63$$

En los programas que escribiremos en el siguiente capítulo sólo calculamos estos valores para configurar el convertidor con los valores de que necesitamos de a cuerdo a nuestro diseño, el resto de la configuración del TLC32040 es una rutina tomada del software proporcionado con la tarjeta de desarrollo.

[1] Texas Instruments, Initieling the TLC 320C40 AIC on TMS320C5X DSK

Prácticas

5.1. Práctica No.1 Modos de direccionamiento.

OBJETIVO:

En esta práctica, lo que se pretende es observar los distintos modos de direccionamiento de la memoria, además aprender el uso del depurador, ya que los programas se cargarán en la memoria y se correrán paso a paso, con el objeto de observar los cambios en los registros para cada instrucción. Y observar las diferencias entre los diferentes modos en que se direcciona la memoria

Pasos para el desarrollo

1. Lo primero que se debe hacer es capturar el código del programa en un editor de textos compatible con ASCCII, como el edit de MS-DOS. Guardarlo con la extensión .ASM

2. Salir del edit, y en el directorio en el que se encuentra el software del TMS320C50 ensamblarlo con la instrucción: DSK5A [archivo].ASM. Recuerde que se pueden usar las opciones /l y /k, si así se desea.

3. Si no se indican errores, entrar al depurador con el comando DSK5D. Recordar que para ello, debe de estar conectada la tarjeta al puerto de la máquina, y alimentada con la fuente de 9 v.a.

4. Una vez dentro del depurador, cargar el archivo con los comandos "LD" cargar de disco. En la línea de comandos indicar el path completo del archivo ejecutable. Por ejemplo: C\:tsmc50\POG1.DSK. Con esta instrucción se llama el archivo ejecutable llamado PROG1, el cuál se encuentra en el subdirectorio \tmsc50. No es necesario indicar la extensión .DSK, ya que por omisión el programa busca el archivo ejecutable con el nombre indicado. En seguida dar doble enter, el programa se carga en memoria, y el código se observa en la pantalla del depurador.

5. Ejecutar el programa paso a paso, observando los cambios en cada uno de los registros involucrados en cada instrucción. El código de los programas es el siguiente:

Esta práctica la dividimos en tres partes, la primera consiste en instrucciones para sumas, en la segunda se dan las instrucciones para multiplicación y la tercera es para ver como se puede hacer movimientos de bloques de memoria. Los códigos son los siguientes;

5.1.2. Programa uno, direccionamiento inmediato, directo e indirecto.

```
        .mmregs             ; incluye registros mapeados
        .ds 0f00h           ; segmento de datos en memoria dato localidad 00F00h

D1 .set 1                   ; variable D1, no genera código
D2 .set 2                   ; variable D2, no genera código
D3 .set 3
D4 .set 4
D5 .set 5

dat1 .word 5                ; variables de datos, si generan código
dat2 .word 10
dat3 .word 15
dat4 .word 2
dat5 .word 3
total .word 0

        .ps         ; Inicio del segmento de programa
     .entry 0a00h   ; Localidad donde inicia la ejecución del programa

* Direccionamiento inmediato, el operando se incluye en la instrucción    *

        LDP  #dat1      ; Apunta a la pagina donde esta dat1
        LACL #D1        ; ACC =D1=1
        ADD #D2         ; ACC =D1+D2=1+2=3
        ADD #D3         ; ACC =D1+D2+D3 =1+2+3= 6
        ADD #D4         ; ACC =D1+D2+D3+D4=1+2+3+4=10
        ADD #D5         ; ACC =D1+D2+D3+D4+D5=1+2+3+4+5=15
                        ; Observar el resultado en el ACC en hexadecimal
        SACL total      ; Acumula el resultado en la variable total
        ZAP             ; Limpia el acumulador
        NOP             ; No hace nada
        NOP

* Suma de 5 constantes usando direccionamiento directo    *

        LDP #dat1       ; carga la página donde se encuentran los datos

        ZAP             ; cero en el acumulador y el registro de producto
```

```
        LACL dat1   ; pone en la parte baja del acumulador dat1
        ADD dat2    ; suma dat2
        ADD dat3    ; suma dat3
        ADD dat4    ; suma dat4
        ADD dat5    ; suma dat5
        SACL total  ; pone el resultado  de la suma en total;

      NOP
      NOP
*****************************************************************
* Suma de las mismas constantes usando direccionamiento indirecto *
*****************************************************************

      LDP #dat1     ; carga la página donde se encuentran los datos
       ZAP          ; Cero en el acumulador y el registro de producto
       SACL total   ; Pone en cero la localidad de total
       MAR *,AR0      ; Elige el registro AR0 como registro auxiliar en uso
       LAR AR0,dat1   ; pone la dirección de dat1 en el registro auxiliar AR0
       ADD *+       ; Suma dat1 al acumulador y post incrementa en uno AR0
       ADD *+       ; Suma dat2 al acumulador y post incrementa en uno AR0
       ADD *+       ; Suma dat3 al acumulador y post incrementa en uno AR0
       ADD *+       ; Suma dat4 al acumulador y post incrementa en uno AR0
       ADD *+       ; Suma dat5 al acumulador y pos incrementa en una AR0
       SACL *       ; Pone el resultado de la suma en total

      NOP
      NOP
*****************************************************************************
* Suma de las mismas constantes con direccionamiento indirecto y usando repeticiones *
*****************************************************************************

       LDP #dat1      ; Carga la página de datos
       ZAP            ; Cero en el acumulador y el registro de producto
       SACL total     ; Pone en cero la localidad total
       MAR *,AR0      ; Elige el registro AR0 como registro auxiliaren uso
       LAR AR0,#dat1  ; Pone la dirección de dat1 en el reg. auxiliar AR0
       RPT #4         ; repetir 5 veces la siguiente instrucción.

       ADD *+         ; Suma los cinco números al acumulador
       SACL *         ; Pone el resultado en total

      NOP
      NOP

*****************************************
* Uso de repetición de bloques RPTB *
*****************************************

       ZAP              ; cero a ACC y a P
       SACL total
       LAR AR0,#dat1    ; Pone la dirección dato1 en AR0
```

```
        MAR *,AR0       ; Selecciona el registro auxiliar  AR0

        LACL #4         ; Pone en la parte baja de ACC un 4
        SAMM BRCR       ; Pone lo que estaba en la parte baja de ACC en
                        ; el registro de repetición de bloques BRCR
        ZAP             ; Cero a ACC y a P. Acumulador y registro producto.

        RPTB SUMA       ; repite el bloque SUMA 4+1 veces
        ADD *+          ; suma de dat1 hasta dat5 al acumulador
          NOP

SUMA    NOP             ; Observa el retorno de la repetición de bloque y
                        ; el decremento del registro BRCR

        MAR *,AR1
        LAR AR1,#total
        SACL *          ; Pone el resultado en total.

       NOP
       NOP
       B FIN
FIN
       .end
```

5.1.3. Programa dos, uso de multiplicaciones

```
*****   ************   *********   *********   *********   ****
* Uso de multiplicaciones con repetición *

      .mmregs
      .ds 0f00h             ; segmento de datos
A1    .word 001h            ; variables
A2    .word 002h
A3    .word 003h
A4    .word 004h
A5    .word 005h
A6    .word 006h
*
X1    .word 001h
X2    .word 002h
X3    .word 001h
X4    .word 002h
X5    .word 001h
X6    .word 002h
*
```

```
Y0    .word 0          ; Variable para el resultado
*
N1    .set 5           ; Variable que no genera código

      .ps 0a00h
      .entry

      LDP #Y0
      SETC SXM         ; Activa el modo de signo extendido.
      LAR AR1,#X1      ; Apunta AR1 a la variable X1
      LAR AR2,#A1      ; Apunta a AR2 a la variable A1
      MAR *,AR1        ; Marca a AR1 como registro auxiliar en uso
      LACL #N1         ; Carga ACCL con N1
      SAMM BRCR        ; Carga el registro contador de repetición con N1 = 5
      ZAP              ; Limpia el ACC y registro producto.

      RPTB FIN_1    ; Repite el siguiente bloque 5+1 veces
      LT *+,AR2     ; Carga el valor apuntado por AR2 en el registro TREG0 e
        ; incrementa ARP. TREG0 se usa para cargar un multiplicando
      MPY *+,AR1    ; Multiplica el valor apuntado por AR2 con el de AR1 e incrementa
        ; AR1. El resultado se guarda en el registro producto (PREG).
      APAC          ; Suma el registro producto al acumulador.
FIN_1 NOP           ; Fin del bloque. Observar la realización del bloque 6 veces.

      SACL Y0          ; Guarda el resultado en Y0

        NOP
        NOP
```

*** Uso de la instrucción LTA = Carga y acumula.**

```
      LDP #Y0
      SETC SXM
      LAR AR1,#X1      ; Apunta AR1 a X1
      LAR AR2,#A1      ; Apunta AR2 a A1
      MAR *,AR1        ; Marca AR1 como registro auxiliar en uso
      LACL #N1         ; Carga un 5 en ACC
      SAMM BRCR        ; Carga un 5 en el registro de repetición
      ZAP
      SACL Y0          ; Pone en cero a Y0

      RPTB FIN_1       ; Repite el siguiente bloque 6 veces
      LTA *+,AR2       ; Carga TREG0 con AR2 y acumula el resultado anterior
      MPY *+,AR1       ; Multiplica AR1 con AR2
FIN_1 NOP              ; Fin de bloque
      APAC             ; Suma final
      SACL Y0
```

```
        NOP
        NOP
* Uso de la instrucción MPYA = multiplica y acumula

      LAR AR1,#X1
      LAR AR2,#A1
      MAR *,AR1
      LACL #N1
      SAMM BRCR
      ZAP
      LACL Y0         ; Carga Y0 con cero.

      RPTB FIN_1
      LT *+,AR2
      MPYA *+,AR1  ; Multiplica AR1 con AR2 y acumula el resultado anterior
FIN_1  NOP
      APAC            ; Suma el resultado fina al ACC.
      SACL Y0

        NOP
        NOP
* Uso de la instrucción MAC = Multiplica y acumula sin usar el registro TREG0

      LAR AR1,#X1
      LAR AR0,#A1
      MAR *,AR0

      LACC #00        ; Limpia el acumulador
      SACL Y0
      RPT #N          ; repite 5+1 veces la siguiente instrucción
      MAC #X1,*+      ; Multiplica y acumula X1 con AR0 e incrementa AR0 en uno, al
        ; mismo tiempo, X1 se incrementa en uno por el registro de                ;
prebúsqueda PFC.

      APAC            ; Suma final
      SACL Y0

        NOP
        NOP
        B FIN
FIN

      .END
```

5.1.4. Movimiento de bloques

```
* Movimiento de bloque de memoria a memoria
      .mmregs
      .ds 0f00h
X1    .word 001h
X2    .word 002h
X3    .word 003h
X4    .word 004h
X5    .word 005h
X6    .word 006h
*
Y     .word 0,0,0,0,0,0,0,0,0,0
*
N     .set 5

      .ps 0a00h
      .entry

      LDP #Y
      LAR AR0,#X1
      MAR *,AR0

      LACC #00

      RPT #N              ; Repite 6 veces la siguiente instrucción
      BLDD *+,#Y          ; Mueve AR0 a Y en incremente Y en uno.

      NOP
      NOP
      .END
```

Cuestionario

1.-Se tienen 20 datos en memoria 1,2,3... 20realize un programa que sume los datos cada 3 posiciones de memoria *Utilice el registro INDX y uno de los registros AR'i.

2.- Consulte el manual e investigue como habilitar el buffer circular, también investigue que condiciones son necesarios para habilitarlo.

3.-Se tiene 20 datos en memoria 1,2,.3,...,20 realice un programa que realice 5 veces la suma 1+2+3 +...20 utilice el buffer circular.

5.2 Práctica No.2. Programas en tiempo real

Filtros de tipo IIR

OBJETIVOS:

1.-A través del ejemplo mostrado, se aprenda como diseñar filtros de respuesta infinita al impulso IIR usando el método de la transformada bilineal.

2.-Usar los modos de direccionamiento directo e indirecto del TMS320C50 para implementar este tipo de filtros.

3.-Aprender la rutina para inicializar al TMS320C50 y al convertidor A/D D/A TLC32040 usada siempre en programas de tiempo real.

Pasos para el desarrollo

1. Para implementar esta práctica, primero se diseña el filtro digital de acuerdo a la teoría del capítulo I.
2. Una vez obtenida la ecuación en diferencias, se proceda a simular su respuesta en Matlab, para probar que la respuesta sea la adecuada de acuerdo al diseño realizado.
3. Una vez vista que el filtro tiene la respuesta esperada, se procede a implementarlo con el DSP.

Para la demostración de cómo implementar este tipo de filtros en el DSP, se desarrolló un filtro pasa bajas de orden 2 con estructura directa II y un filtro supresor de banda de orden 6 con estructura en cascada.

Filtro pasa bajas

El procedimiento que se sigue es el que se mencionó en el capítulo de filtros digitales IIR, y es el siguiente;

Diseñar y desarrollar un filtro digital pasa bajas tipo IIR con frecuencia de corte en la banda de paso de 1 kh, atenuación en esta banda de -3 db máximo, la frecuencia de corte de la banda suprimida es 2100 con -12 db de atenuación. La frecuencia de muestreo de 8 kh.

En la figura 5.1 se muestra el diagrama de la respuesta que debe de tener el filtro que pretendemos implementar;

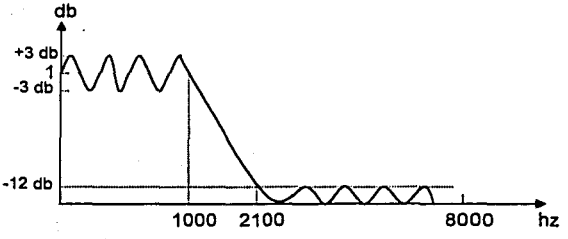


Figura 5.1 Respuesta en frecuencia del filtro de segundo orden

Dadas las especificaciones, se procede con el segundo paso, que es calcular el orden del filtro. En nuestro caso se aplicarán polinomios de Butterworth, por lo tanto, el orden se calcula como sigue;

$$n \geq \frac{\log\left(\frac{10^{0.1(12)}-1}{10^{0.1(3)}-1}\right)}{2\log\left(\frac{2100}{1000}\right)} = \frac{1.1738}{0.6444} = 1.82$$

Por lo tanto, debemos de escoger un orden de 2. De tablas, se obtiene el polinomio de Butterworth normalizado para orden 2, que es $s^2 + \sqrt{2}\, s + 1$. Por lo tanto, la función de transferencia para un filtro analógico es;

$$H(s) = \frac{1}{s^2 + \sqrt{2}s + 1}$$

El siguiente paso es aplicar la transformada bilineal para el caso pasa bajas para obtener la función de transferencia H(z) discreta, esto es;

$$H(z) = H(s)\Big|_{s=c\left(\frac{z-1}{z+1}\right)}$$

Donde:

$$c = \cot\left(\frac{\pi f_{ca}}{f_s}\right) = \cot\left(\frac{\pi(1000)}{8000}\right) = 2.4142$$

Entonces;

$$s = 2.4142\left(\frac{z-1}{z+1}\right)$$

Sustituyendo;

$$H(z) = \frac{1}{\left(2.4142\left(\frac{z-1}{z+1}\right)\right)^2 + \sqrt{2}\left(2.4142\left(\frac{z-1}{z+1}\right)\right)+1}$$

Multiplicando el numerador y denominador por $(z+1)^2$

$$H(z) = \frac{z^2+2z+1}{\left(5.8289(z-1)^2\right)+3.4142(z-1)(z+1)+(z+1)^2}$$

Despejando se obtiene;

$$H(z) = \frac{z^2+2z+1}{10.2426z^2-9.656z+3.4142}$$

Dividiendo el numerador y denominador entre $10.2426z^2$

$$H(z) = \frac{0.0976+0.1953z^{-1}+0.0976z^{-2}}{1-0.9427z^{-1}+0.3333z^{-2}}$$

La cuál es la función de transferencia deseada. Como;

$$H(z) = \frac{Y(z)}{X(z)} = \frac{0.0976 + 0.1953z^{-1} + 0.0976z^{-2}}{1 - 0.9427z^{-1} + 0.3333z^{-2}}$$

Despejando a Y(z) se obtiene;

$$Y(z) = 0.0976X(z) + 0.1953z^{-1}X(z) + 0.0976z^{-2}X(z) + 0.9427z^{-1}Y(z) - 0.3333z^{-2}Y(z)$$

Aplicando la transformada Z inversa se obtiene la ecuación en diferencias que describe el filtro digital deseado;

$$y(n) = 0.0976\ x(n) + 0.1953\ x(n-1) + 0.0976\ x(n-2) + 0.9427\ y(n-1) - 0.3333\ y(n-2)$$

Ya se tiene la ecuación en diferencias deseada, ahora lo que sigue es implementarla en alguna estructura. Como el filtro es de orden 2, podemos implementarla en la forma directa II. Como los coeficientes obtenidos son muy pequeños para poderlos procesar matemáticamente con el TMS320C50, ya que este cuenta con localidades de memoria de 16 bits, los coeficientes obtenidos se deben normalizar a formato Q_{15}, para dejar un bit de signo. Multiplicando cada coeficiente por 2^{15} se tiene;

$$y(n) = 3198\ x(n) + 6396\ x(n-1) + 3198\ x(n-2) + 30890\ y(n-1) - 10923\ y(n-2)$$

Por lo tanto, lo que debemos programar en el DSP es;

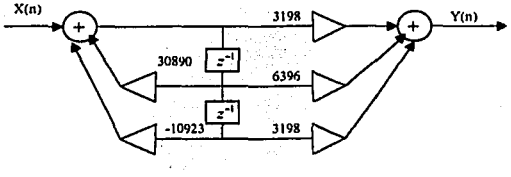


Figura 5.2 Coeficientes del filtro de segundo orden en la forma canónica II

Donde se ve que;

$$d(n) = x(n) + 30890\ d(n-1) - 10923\ d(n-2)$$
$$y(n) = 3198\ d(n) + 6396\ d(n-1) + 3198\ d(n-2)$$

Para poder comprobar que nuestros cálculos están correctos, podemos hacerlo en Matlab con la siguiente instrucción;

[b , a] = butter (2, 1000/4000)

Esta instrucción pide al programa que entregue una función de transferencia de un filtro digital con plantilla de Butterworth, de segundo orden, pasa bajas con frecuencia de corte en 1000 hz, y frecuencia de muestreo de 8 000 hz. El resultado es el siguiente;

```
» [b,a] = butter (2, 1000/4000)

b =
   0.0976   0.1953   0.0976
a =
   1.0000  -0.9428   0.3333
```

Aquí "b" son los coeficientes del numerador y "a" los coeficientes del denominador en la función de transferencia, observe que es la misma que obtuvimos; la siguiente instrucción nos simula el filtro;

Freqz (b, a, 300, 8000)

El resultado es el siguiente;

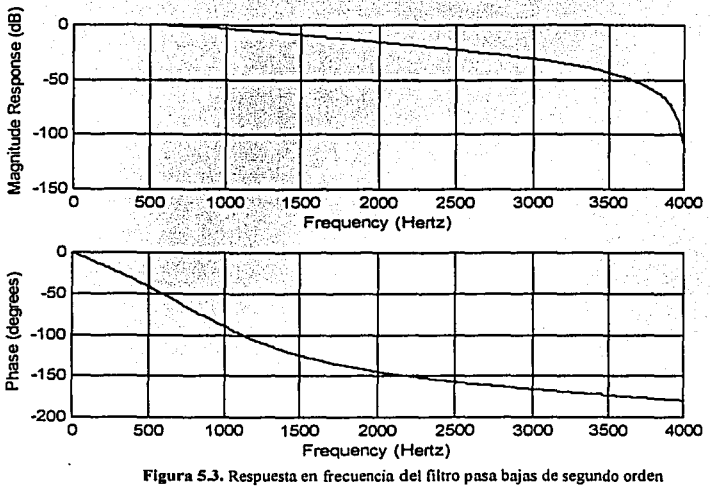


Figura 5.3. Respuesta en frecuencia del filtro pasa bajas de segundo orden

Finalmente, lo que tenemos que hacer es implementarlo en el DSP tms320c50. Para poder implementarlo, se pondrá la memoria de datos de la siguiente forma;

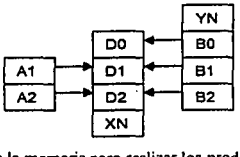


Figura 5.4 Como se acomodan los datos en la memoria para realizar los productos de las ecuaciones en diferencias de la forma canónica II

Como se observa en la figura, tenemos las cinco localidades para los coeficientes; primero se calcula a D0, que es $x(n) + A_1d(n-1) + A_2d(n-2)$. D0, D1 y D2 son d(n), d(n-1) y d(n-2) respectivamente. Después de calcular d(n), se calcula la muestra de salida y(n) que es $y(n) = B_0d(n) + B_1d(n-1) + B_2d(n-2)$. Como D1 y D2 son los respectivos retazos, debemos desplazar estos números para dejar listas las localidades para la siguiente muestra x(n), es decir, realizar los retrasos, de tal forma que para la

siguiente salida, D2 se desplaza y se pierde el dato, D1 se desplaza y queda en la posición de D2, D0 se desplaza y queda en la posición de D1, quedando D0 libre para el siguiente cálculo de d(n). XN sólo se usa para guardar la muestra x(n) entrante. YN se usa para escribir la muestra de salida ya procesada.

Para hacer los retrasos se pueden usar las instrucciones DMOV, LTD o MACD, cada una de ellas mueve un dato a la siguiente localidad. El código para el filtro anterior es el siguiente;

```
*****************************************************************
* FILTRO DE TIPO IIR PASA BAJAS, PLANTILLA DE BUTTERWORT        *
*                                                               *
*                                                               *
*              ARCHIVO: IIRBAJA.ASM                             *
*                                                               *
*****************************************************************
* INICIO DEL PROGRAMA                *
**************************************

      .mmregs       ; incluye registros mapeados
      .ds 0f00h     ; inicio del segmento de datos
TA    .word 25      ; valor de TA para Fs = 8 kh de A/D y D/A
RA    .word 25      ; valor de RA para Fs = 8 kh  "   "   "
;
TB    .word 25      ;  "   TB  "    "     "    "   "
RB    .word 25      ;  "   RB  "    "     "    "   "
;
AIC_CTR .word 08h     ;

YN    .word 0       ; localidad reservada para escribir la salida y(n)
B0    .word 3198    ; coeficiente b0
B1    .word 6396    ;  "       b1
B2    .word 3198    ;  "       b2
A1    .word 30890   ;  "       a1
A2    .word -10923  ;  "       a2
D0    .word 0       ; localidad para d(n)
D1    .word 0       ; localidad para d(n-1)
D2    .word 0       ; localidad para d(n-2)
DN    .word 0       ; localidad para guardar la muestra x(n)

      .ps 0080ah    ; inicio de vectores de interrupción
rint:  B RECEIVE    ; fija el vector de interrupción

      .ps 0a00h     ; inicio del segmento de programa
      .entry        ; inicia dirección de PC
```

```
*****************************************
* ESTA RUTINA INICIALIZA EL DSP         *
*****************************************
*---------------------------------------
      SETC INTM          ; deshabilita interrupciones mascarables
      LDP #0             ; carga la página 0 para transmisión
      OPL #0834h,PMST    ; reubica vectores de interrupción
      LACC #0            ; carga 0 al acumulador
      SAMM CWSR          ; salva ACC en CWSR
      SAMM PDWSR         ; salva ACC en PDWSR
      SETC SXM           ; habilita como bit de signo extendido
      SPLK #022h,IMR     ; habilita interrupción de transmisión

      CALL AICINIT   ; llama a la rutina ACINIT para A/D y D/A
      SPLK #12h,IMR  ; habilita interrupción de recepción
      CLRC OVM       ; evitar sobre flujo
      SPM 0          ; configura el puerto serial
      CLRC INTM      ; deshabilita interrupciones mascarables

WAIT: NOP
      NOP
      B WAIT       ; espera interrupción

*****************************************************************
* RUTINA DE ATENCION DE INTERRUPCION Y REALIZACION DEL FILTRO   *
*****************************************************************

RECEIVE:
      LDP #DN        ; carga la página donde está x(n)
      CLRC INTM      ; deshabilita interrupciones

      LAMM DRR       ; carga en ACCL la muestra del puerto DRR
      AND #0fffch    ; pone en cero los bits de comunicación
      SACL DN        ; copia ACCL en DN
      LACC DN,15     ; carga DN en ACC con corrimiento de 15 bits

      LT D1          ; carga en TREG0 D1
      MPY A1         ; multiplica d1 con a1
      LTA D2         ; D2 a TREG0 y ACC = x(n) + d1*a1
      MPY A2         ; multiplica d2 con a2

      APAC           ; ACC = x(n) + a1*d(n-1) + a2*d(n-2) en (Q30)
      SACH D0,1      ; D0 = ACCH con corrimiento de un bit en (Q15)
      LACC #0        ; ACC = 0
      MPY B2         ; multiplica d2 con b2, en TREG0 estaba d2
      LTD D1         ; carga d1 en TREG0 y desplaza d2
      MPY B1         ; multiplica d1 con b1
```

```
        LTD D0          ; carga d0 en TREG0 y desplaza d1
        MPY B0          ; multiplica d0 con b1
        APAC            ; ACC = b2*d(n-2) + b1*d(n-1) + b0*d(n) en (Q30)
        SACH YN,2       ; YN = ACCH con corrimiento de dos en (Q15)
        LACL YN         ; carga YN en ACCL

        AND #0fffch     ; pone en cero los dos bits de comunicación
        SAMM DXR        ; escribe la muestra y(n) en DXR
        RETE            ; retorna al programa principal

*****************************************************
* INICIALIZA AL TLC320C46, CONVERTIDOR A/D Y D/A
*****************************************************

AICINIT: SPLK #20h,TCR
        SPLK #01h,PRD
        MAR *,AR0
        LACC #0008h
        SACL SPC
        LACC #00c8h
        SACL SPC
        LACC #080h
        SACH DXR
        SACL GREG
        LAR AR0,#0ffffh
        RPT #10000
        LACC *,0,AR0
        SACH GREG
        ;-----------------------------
        LDP #TA
        SETC SXM
        LACC TA,9
        ADD RA,2
        CALL AIC_2ND
        ;-----------------------------
        LDP #TB
        LACC TB,9
        ADD RB,2
        ADD #02h
        CALL AIC_2ND
        ;-----------------------------
        LDP #AIC_CTR
        LACC AIC_CTR,2
        ADD #03h
        CALL AIC_2ND
        RET
AIC_2ND:
```

```
LDP #0
SACH DXR
CLRC INTM
IDLE
ADD #6h,15
SACH DXR
IDLE
SACL DXR
IDLE
LACL #0
SACL DXR
IDLE
SETC INTM
RET
.END
```

Ahora solo queda comprobar el diseño en el DSP, para ello, estando en el directorio en el que se encuentra el software del C50, se puede teclear el comando DSK5L IRRBAJA.DSK, con esa interacción se ejecuta el archivo ejecutable y sólo queda observar el resultado en el osciloscopio. Otra forma de realizar el filtro es la siguiente;

```
*****************************************************************
* RUTINA DE ATENCION DE INTERRUPCION Y REALIZACIÓN DEL FILTRO    *
*****************************************************************
      LAMM DRR,15       ; carga la muestra con corrimiento de 15 bits
      AND #0fffch       ; limpia bits de comunicación
      LAR AR1,#D1       ; apunta AR1 a la localidad D1
      MAR *,AR1         ; marca a AR1 como registro auxiliar en uso
      RPT #1            ; repite la siguiente instrucción 2 veces
      MAC  #A1,*+       ; multiplica y acumula A1 con D1 y mueve AR1 a D2
      APAC              ; suma el producto en ACC
      SACH D0,1         ; pone ACCH en D0 y quita un bit de signo

      LAR AR1,#D2       ; apunta AR1 a D2
      ZAP               ; pone en cero ACC y PREG
      RPT #2            ; repite tres veces la siguiente instrucción
      MACD #B0,*-       ; multiplica  y acumula B0 con D2 y desplaza D2, en paralelo
            ; AR1 apunta a D1, para hacer B1*D1 y después B2*D0
      APAC              ; suma final
      SACH YN,2         ; saca ACCH en YN con corrimiento de 2 bits a la izquierda
      LACL YN
      AND #0FFFch
      SAMM DXR
```

NOTA: la instrucción MACD en el caso anterior funciona adecuadamente para el filtro, porque B0 es igual a B2, de lo contrario se tendría que intercambiar las posiciones de estos dos coeficientes en el

segmento de datos, para que el resultado sea adecuado. En los siguientes programas omitimos el código para la inicialización del DSP y el convertidor A/D y D/A, ya que son iguales, a menos que se diga lo contrario.

Cuestionario

1.-Un integrador analógico esta dado mediante la función $H_a(s)=\frac{1}{s}$, un integrador digital esta dado por:

$$H(z)=\frac{T}{2}\left(\frac{1+z^{-1}}{1-z^{-1}}\right)$$

(a) Escriba las ecuaciones en diferencias del integrador, dibuje el diagrama del sistema

(b) Escriba la función de trasferencia H(z) par un derivador digital ,escriba las ecuaciones en diferencias y dibuje el diagrama del sistema.

(c)

2.-Describa las características de los siguientes filtros

(a) Butterworth

(b) Chebychev I

(c) Chebychev II

(d) Elíptico

3.-Investigue la formula para obtener el grado del los filtros Chebychev I y Chebychev II.

4.-Se requiere un filtro IIR pasa bajas con las siguientes especificaciones

1. Rizo en la banda de paso 0.5 [dB]
2. Frecuencia de corte en la banda de paso 1.2 [Khz]
3. Atenuación en la banda de rechazo 40 [dB]
4. Frecuencia de corte en la banda de rechazo 2 [Khz]
5. Frecuencia de muestreo de 8 [Khz]

Determine el orden mínimo requerido para un filtro digital, usar aproximación Chebychev

5.-Dada la figura siguiente que representa un sistema de orden 2, escriba las ecuaciones en diferencias para $\omega_1(n)$ y $\omega_2(n)$ que describe al diagrama (forma transpuesta II) si la ecuación de implementación esta dada por

$$y(n) = b_0 x(n) + \omega_1(n-1)$$

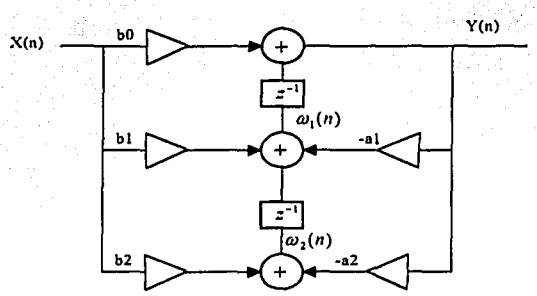


Figura 5.5 Sistema de segundo grado con estructura en forma transpuesta II

6.-Modifique el programa de manera que los productos se realicen en forma de direccionamiento indirecto sin instrucciones de repetición

7.-Modifique el programa y realice el filtro con la estructura Directa I

5.3 Practica No.3 Diseño de Filtros de Orden superior a 2

Filtro supresor de banda de orden 6

Objetivo:

1.-Mostrar como diseñar filtros IIR de orden superior a 2 para alcanzar mejores pendientes.

2.-Usar algunas instrucciones de Matlab par calcular los coeficientes de filtros IIR.

La función de transferencia del siguiente filtro supresor de banda de orden 6 se calcula en Matlab, después se pasa la función de transferencia en productos parciales, para poder realizar el filtro en su estructura en cascada, el procedimiento es el siguiente.

Se desea implementar un filtro digital supresor de banda de orden 6, donde la frecuencia suprimida sea de 1400 hz a 1600 hz y frecuencia de muestreo de 10 khz. Las instrucciones en Matlab y sus resultados son los siguientes;

```
» [b,a] = butter (3, [1400 1600]/5000, 'stop')

b =
   0.8818   -3.1161    6.3160   -7.6734    6.3160   -3.1161    0.8818

a =
   1.0000   -3.3858    6.5733   -7.6570    6.0448   -2.8630    0.7776
```

La cuál es la función de transferencia siguiente;

$$H(z) = \frac{0.8818 - 3.1141z^{-1} + 6.3160z^{-2} - 7.6734z^{-3} + 6.3160z^{-4} - 3.1161z^{-5} + 0.8818z^{-6}}{1 - 3.3858z^{-1} + 6.5733z^{-2} - 7.657z^{-3} + 6.0448z^{-4} - 2.863z^{-5} + 0.7776z^{-6}}$$

En Matlab calculamos las raíces del numerador y denominador obteniendo;

```
roots (b)

ans =

   0.5890 + 0.8082i
   0.5890 - 0.8082i
   0.5889 + 0.8082i
   0.5889 - 0.8082i
   0.5890 + 0.8082i
   0.5890 - 0.8082i

» roots (a)

ans =

   0.5260 + 0.8125i
   0.5260 - 0.8125i
   0.6128 + 0.7523i
   0.6128 - 0.7523i
   0.5541 + 0.7580i
   0.5541 - 0.7580i
```

Ahora multiplicamos los conjugados complejos para obtener raíces reales y obtener la función de transferencia de la siguiente forma;

$$H(z) = \frac{0.8818(1 - 1.178z^{-1} + z^{-2})(1 - 1.178z^{-1} + z^{-2})(1 - 1.178z^{-1} + z^{-2})}{(1 - 1.052z^{-1} + 0.9368z^{-2})(1 - 1.2256z^{-1} + 0.9415z^{-2})(1 - 1.1082z^{-1} + 0.8816z^{-2})}$$

Entonces, podemos dejar la función de transferencia como productos de fracciones parciales;

$$H(z) = H_1(Z) \times H_2(z) \times H_3(z) \times H_4(Z)$$

Donde;

$$H_1(z) = 0.8818$$

$$H_2(z) = \frac{1-1.178z^{-1}+z^{-2}}{1-1.052z^{-1}+0.9368z^{-2}}$$

$$H_3(z) = \frac{1-1.178z^{-1}+z^{-2}}{1-1.2256z^{-1}+0.9415z^{-2}}$$

$$H_4(z) = \frac{1-1.178z^{-1}+z^{-2}}{1-1.1082z^{-1}+0.8816z^{-2}}$$

Y finalmente con la transformada Z inversa se obtiene las ecuaciones en diferencias que se deben poner en cascada para obtener la respuesta del filtro deseado, es decir, la salida de una es la entrada a la siguiente;

$y_1(n) = 0.8818x(n)$

$y_2(n) = y_1(n) - 1.178y_1(n-1) + y_1(n-2) + 1.1052y_2(n-1) - 0.9368y_2(n-2)$

$y_3(n) = y_2(n) - 1.178y_2(n-1) + y_2(n-2) + 1.2256y_3(n-1) - 0.9415y_3(n-2)$

$y_f(n) = y_3(n) - 1.178y_3(n-1) + y_3(n-2) + 1.1082y_f(n-1) - 0.8816y_f(n-2)$

Donde la y_f es la salida del sistema. Para poder programar estas ecuaciones en el DSP, debemos multiplicar los coeficientes por 2^{14}, para normalizar a 14 bits y después programar la siguiente estructura.

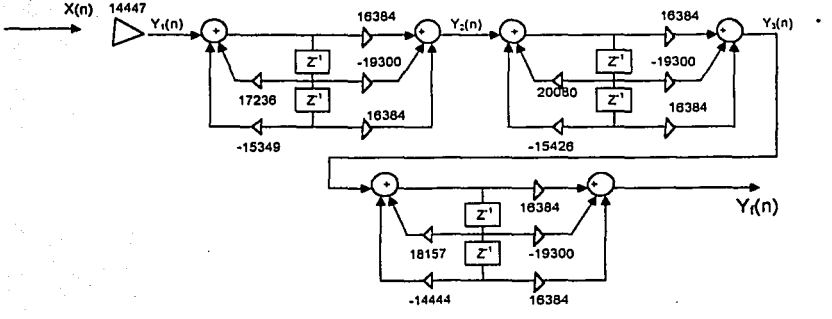


Figura 5.6 Estructura en cascada del filtro digital de 6º grado

Finalmente, la rutina que realiza el filtro anterior es la siguiente; antes presentamos los coeficientes normalizados y los valores que deben tener los registros de configuración del convertidor A/D y D/A para las frecuencias de muestreo y de corte usadas.

```
*********************************
* FILTRO IIR SUPRESOR DE BANDA  *
*   ARCHIVO: IIRSUP.ASM         *
* *******************************

      .mmregs
      .ds 0f00h
TA    .word 18
RA    .word 18
;
TB    .word 24
RB    .word 24
;
AIC_CTR .word 08h
************************
* Variables usadas     *
************************

XN    .word 0  ; localidad para x(n)
Y1    .word 0  ; localidad para y1(n)
Y2    .word 0  ;   "       y2(n)
Y3    .word 0  ;   "       y3(n)
YF    .word 0  ; localidad para la muestra de salida
*********************************
* COEFICIENTES
*********************************

B10   .word 14447

B22   .word 16384
B21   .word -19300
B20   .word 16384
A21   .word 17236
A22   .word -15349

B32   .word 16384
B31   .word -19300
B30   .word 16384
A31   .word 20080
A32   .word -15426

BF2   .word 16384
BF1   .word -19300
```

```
BF0   .word 16384
AF1   .word 18147
AF2   .word -14444

*****************************************************************************
* RETARDOS
**********************
D10   .word 0
D11   .word 0
D12   .word 0
D20   .word 0
D21   .word 0
D22   .word 0
D30   .word 0
D31   .word 0
D32   .word 0
************************
* INICIO DEL PROGRAMA
************************
```

La rutina de atención de interrupción y realización del filtro es la siguiente.

```
*****************************************************************
* RUTINA DE ATENCION DE INTERRUPCION Y REALIZACION DEL FILTRO *
*****************************************************************
RECEIVE:
    LDP #XN
    CLRC INTM

    LAMM DRR
    AND #0fffch
    SACL XN
    LT XN           ; carga el registro TREG con x(n)
    MPY B10         ; multiplica B10 con x(n)
    APAC            ; suma el resultado al ACC
    SACH Y1,1       ; Salva Y1
    ZAP             ; limpia ACC y PREG

    LACC Y1,15      ; guarda Y1 en ACCH
    LAR AR1,#D11    ; apunta AR1 a D11
    MAR *,AR1       ; marca AR1 como registro auxiliar
    RPT #1          ; repite la siguiente instrucción 2 veces
    MAC #A21,*+     ; multiplica y acumula A21 con D11
    APAC            ; suma el último producto
    SACH D10,1      ; salva el resultado en D10
    ZAP             ; limpia ACC y PREG
    LAR AR1,#D12    ; apunta AR1 a D12
    RPT #2          ; repite 2 veces la siguiente instrucción
```

```
MACD #B22,*-    ; multiplica, acumula y desplaza B22 con D12
APAC            ; suma final
SACH Y2,1       ; salva el resultado en Y2
ZAP

LACC Y2,15
LAR AR2,#D21
MAR *,AR2
RPT #1
MAC #A31,*+
APAC
SACH D20,1
ZAP
LAR AR2,#D22
RPT #2
MACD #B32,*-
APAC
SACH Y3,1

ZAP
LACC Y3,15
LAR AR3,#D31
MAR *,AR3
RPT #1
MAC #AF1,*+
APAC
SACH D30,1
LAR AR3,#D32
RPT #2
MACD #BF2,*-
APAC
SACH YF,3

LACL YF         ; salva la muestra final
AND #0FFFch

SAMM DXR        ; escribe la muestra final
RETE            ; retorno al programa
```

Nótese que se usa la misma estructura que en el filtro pasa bajas, sólo que con más etapas. El mismo programa se puede hacer con repeticiones como en el ejemplo anterior.

En la figura 5.7 se presenta la simulación de los coeficientes obtenidos en Matlab. Debe notarse que en caso de querer cambiar los filtros por diseños diferentes, sólo será necesario calcular los coeficientes, los programas anteriores funcionan para cualquier tipo de filtro, esa es una característica de los sistemas procesados en forma digital que pretendíamos demostrar al inicio de nuestro trabajo.

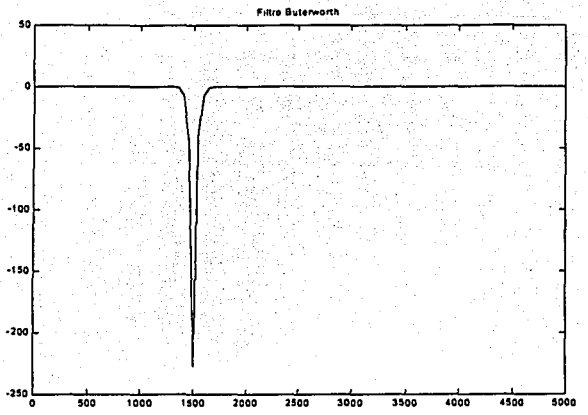


Figura 5.7. Respuesta en frecuencia del filtro de 6° grado

Cuestionario

1.-Explique la razón por la que se implementan filtros en cascada y en paralelo

2.-Investigue la representación de números en aritmética de punto fijo y en aritmética de punto flotante de acuerdo a los estándares IEEE, investigue también cuales DSP de Texas Instruments utilizan aritmética de punto fijo y cuales aritmética de punto flotante (*Recomendación viste el sitio de Internet www.ieee.org, y www.ti.com)

3.-Investigue la razón de utilizar filtros antialiasing en la entrada de un convertidor A/D y en la salida de un convertidor D/A

5.4 Práctica No. 4 Filtros FIR

OBJETIVOS:

1.-Aprender a diseñar filtros FIR de respuesta finita al impulso usando método de ventanas a través del ejemplo mostrado.

2.-Usar el modo de direccionamiento indirecto con estructura de repetición para implementar el filtro FIR

Para el caso de los filtros FIR, se hacen los cálculos en una hoja de cálculo para mayor facilidad, también se pueden hacer directamente en Matlab con las instrucciones mencionadas en el apéndice B.

Filtro pasa bajas

Diseñar un filtro digital FIR pasa bajas con frecuencia de corte de 1 khz y frecuencia de muestreo de 8 khz. Usar ventana de Blackman. El orden del filtro de 81.

Como es un filtro pasa bajas, usamos la siguiente ecuación para calcular los coeficientes del filtro;

$$h(n) = \frac{Sen\omega_{CD}n}{\pi n}$$

$$\text{Donde ;} \quad \omega_{CD} = \frac{2\pi f_{c.o}}{f_s}$$

Y la ventana de Blackman se describe como;

$$W_{BLACK} = 0.42 - 0.5Cos\left(\frac{2\pi n}{N-1}\right) + 0.082Cos\left(\frac{4\pi n}{N-1}\right)$$

Para N = 0, 1, 2, ..., 80.

En la siguiente tabla se muestran los resultados obtenidos para las ecuaciones del filtro y de la ventana. Esos resultados se obtuvieron en una hja de cálculo de Excel. En la figura 5.8 se muestra la respuesta en frecuencia del filtro simulado en Matlab.

n	h(n)n.c	WBLACK		h(n)c	h(n)real	coeficientes
0	0.1000	0.0020	h0=h80	0.0000	0.0000	0.00
1	0.1156	0.0025	h1=h79	-0.0030	0.0000	-0.25
2	-0.0578	0.0041	.....	0.0030	0.0000	0.41
3	-0.1633	0.0069	.....	0.0132	0.0001	2.98
4	-0.1225	0.0108	.....	0.0136	0.0001	4.82
5	0.0000	0.0160	.....	0.0000	0.0000	0.00
6	0.0816	0.0227	.....	-0.0144	-0.0003	-10.71
7	0.0700	0.0309	.....	-0.0148	-0.0005	-15.03
8	0.0145	0.0408		-0.0036	-0.0001	-4.84
9	-0.0128	0.0526		0.0037	0.0002	6.43
10	0.0000	0.0664		0.0000	0.0000	0.00
11	0.0105	0.0824		-0.0040	-0.0003	-10.77
12	-0.0096	0.1008		0.0041	0.0004	13.63
13	-0.0377	0.1215		0.0181	0.0022	72.24
14	-0.0350	0.1448		0.0188	0.0027	89.40
15	0.0000	0.1707		0.0000	0.0000	0.01
16	0.0306	0.1992		-0.0204	-0.0041	-133.18
17	0.0288	0.2302		-0.0213	-0.0049	-160.66
18	0.0064	0.2638		-0.0053	-0.0014	-45.44
19	-0.0061	0.2998		0.0055	0.0017	54.09
20	0.0000	0.3380		0.0000	0.0000	0.01
21	0.0055	0.3782		-0.0061	-0.0023	-75.43
22	-0.0053	0.4202		0.0064	0.0027	88.45
23	-0.0213	0.4637		0.0288	0.0134	437.76
24	-0.0204	0.5082		0.0306	0.0156	509.79
25	0.0000	0.5534		0.0000	0.0000	0.03
26	0.0188	0.5988		-0.0350	-0.0210	-686.49
27	0.0181	0.6440		-0.0377	-0.0243	-795.17
28	0.0041	0.6886		-0.0096	-0.0066	-217.43
29	-0.0040	0.7319		0.0105	0.0077	252.11
30	0.0000	0.7736		0.0000	0.0000	0.01
31	0.0037	0.8130		-0.0128	-0.0104	-342.29
32	-0.0036	0.8498	.....	0.0145	0.0123	402.49
33	-0.0148	0.8835	.....	0.0700	0.0618	2025.92
34	-0.0144	0.9137	.....	0.0816	0.0746	2444.28
35	0.0000	0.9399	.....	0.0000	0.0000	0.04
36	0.0136	0.9619	....	-0.1225	-0.1178	-3859.64
37	0.0132	0.9792	.....	-0.1633	-0.1599	-5239.22
38	0.0030	0.9918	.....	-0.0578	-0.0573	-1879.08
39	-0.0030	0.9994	h39=h41	0.1156	0.1156	3786.97
40	0.0000	1.0020	h40	0.1000	0.1002	3283.35

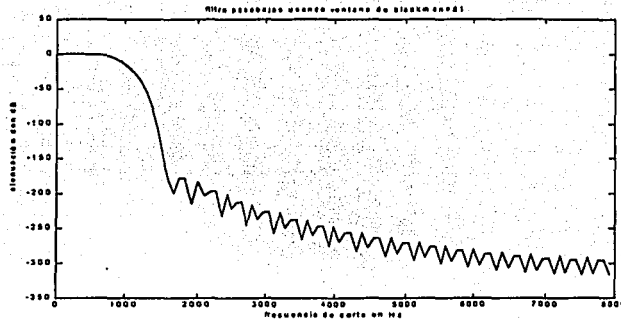


Figura 5.8 Respuesta en frecuencia del filtro FIR con ventana de Blackman con 81 coeficientes

Los coeficientes están normalizados a formato Q15.

Ahora se tiene su ecuación en diferencias,

$$y(n) = h_0x(n) + h_1x(n-1) + \ldots\ldots\ldots + h_{80}x(n-80)$$

Que nos define el filtro deseado. Para poder programar esa ecuación en una forma eficiente, se usa la instrucción MACD y se colocan los datos en memoria de la siguiente forma;

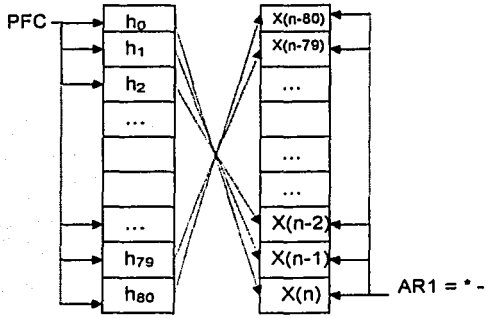


Figura5.9 Acomodo de la memoria dato

Las flechas indican como se deben de hacer los productos. El PFC es un registro de prebúsqueda, el cuál, en un ciclo de repetición siempre apuntará a la siguiente localidad en cada una de las repeticiones, mientras que el AR1 es un registro auxiliar, el cuál usaremos para apuntar a cada uno de los retardos de la señal x(n), al indicar un " *- ", le indicamos que después de cada repetición apunte a la siguiente localidad más baja, por lo tanto se irá decrementando. Entonces la ecuación en diferencias la podemos realizar con la instrucción;

"MACD #h_0,*-"

En este caso, al seleccionar a AR1 como registro auxiliar en uso y apuntarlo como se indica en la figura 5.9, con la MACD sucede lo siguiente;

1. Primero multiplicamos h_0 con x(n), al mismo tiempo el PFC se mueve una localidad hacia adelante y el AR1 apuntará a una localidad inferior, quedando listo para el siguiente cálculo.
2. Luego multiplica h_1 con x(n-1), se mueven el PFC y AR1 de la misma forma que en 1) y al mismo tiempo acumula lo que hay en el registro producto PREG, que fue el resultado del cálculo anterior.
3. Multiplica h_2 con x(n-2) y hace lo mismo que en 2).

Entonces, metiendo la rutina anterior en un ciclo de repetición de 1 hasta 80, se realiza la operación en una forma muy eficiente y rápida. La rutina que realiza el filtro es la siguiente;

```
*************************************************************************
* RUTINA DE ATENCION DE INTERRUPCION Y REALIZACION DEL FILTRO*
*************************************************************************
RECEIVE:
    LDP #XN             ; carga la página donde está x(n)
    CLRC INTM           ; deshabilita interrupciones

    LAMM DRR            ; carga la muestra en ACC
    AND #0fffch         ; quita los bits de comunicación
    SACL XN             ; guarda la muestra en XN

    LAR AR0,#XNLAST     ; apunta AR0 a XNLAST
    ZAP                 ; ceros en ACC y PREG
    MAR *,AR0           ; marca a AR0 como registro auxiliar
    RPT #80             ; repite la siguiente instrucción 81 veces
    MACD #h0,*-         ; multiplica h0 con x(n) y suma el producto anterior
    APAC                ; suma el último producto
    SACH OUTPUT,2       ; pone ACCH en OUTPUT con corrimiento de 2
```

```
    LACL OUTPUT              ; escribe OUTPUT en ACCL

    AND #0fffch              ; pone en ceros los dos bits de comunicación
    SAMM DXR                 ; escribe la muestra en el registro de transmisión
    RETE                     ; reotorno al programa
```

Los valores de los registros TB, TA, RA, y RB, los coeficientes del filtro y las localidades para retardo son:

```
    .mmregs
    .ds 0f00h
TA    .word 25
RA    .word 25
;
TB    .word 25
RB    .word 25
;
AIC_CTR .word 08h

OUTPUT .word 0
TEMP   .word 0
TEMP1  .word 0
*
* COEFICIENTES fc = 1Khz  fs = 8 khz, normalizados en Q15
*
h0    .word 0
h1    .word 0
h2    .word 0
h3    .word -1
h4    .word -0
h5    .word 3
.....
....
h72   .word 0
h73   .word 6
h74   .word 6
h75   .word 3
h76   .word 0
h77   .word -1
h78   .word 0
h79   .word 0
h80   .word 0

* LOCALIDADES PARA ENTRADA XN Y SUS RETARDOS
XN    .word 0,0,0,0,0,0,0,0,0,0
XN1   .word 0,0,0,0,0,0,0,0,0,0
```

```
XN2    .word 0,0,0,0,0,0,0,0,0,0
XN3    .word 0,0,0,0,0,0,0,0,0,0
XN4    .word 0,0,0,0,0,0,0,0,0,0
XN5    .word 0,0,0,0,0,0,0,0,0,0
XN6    .word 0,0,0,0,0,0,0,0,0,0
XN7    .word 0,0,0,0,0,0,0,0,0,0
XNLAST .word 0;
```

Los programas anteriores funcionan para cualquier filtro que se quiera implementar, basta con cambiar los coeficientes de acuerdo al diseño del filtro.

Cuestionario

1.-Dada la siguiente estructura de un filtro FIR de orden 7 con coeficientes h(0), h(1), h(2), h(3), h(4), h(5), h(6) Conteste falso o verdadero a las siguientes afirmaciones (Justifique sus respuesta)

(a) La ecuación en diferencias esta dada por

$Y(n)=h_0[x(n)+x(n-6)]+h_1[x(x-1)+x(n-5)]+h_2[x(n-2)+x(n-4)]+h_3x(n-3)$

(b) Ahorra 3 multiplicaciones

(c) Gasta 3 localidades de memoria

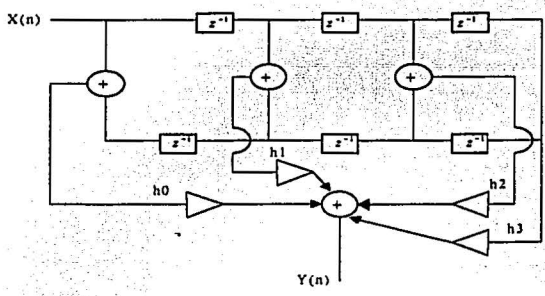


Figura 5.10 Estructura de un filtro FIR

2.-Obtenga el número de coeficientes de un filtro pasa altas con frecuencia de corte de 1.8 [Khz] con banda de transición de 200 [Hz] y una atenuación de 60 [dB] utilice la formula de Kaiser recuerde que la frecuencia de corte y atenuación deberán ser divididas por la frecuencia de muestreo.

3..-Obtenga los coeficientes de un filtro supresor de banda de 2 [Khz] a 2.2 [Khz] con una frecuencia de muestreo de 19 [kHz] y una atenuación de 40[dB] utilice el método de ventanas , Incluya en la respuesta el tipo de ventana y la razón por la que la escogió

4.-Con los valores anteriores implemente el filtro

5.- Investigue la forma de calcular los coeficientes de un derivador usando estructura FIR, escriba la formula.

5.5 Práctica No.5 Implementación de un modulador en amplitud

OBJETIVO:

Usar las herramientas vistas para implementar un modulador AM utilizando multiplicaciones.

Desarrollo

Es muy sencillo implementar un modulador en amplitud con un DSP, sólo se necesita generar un oscilador para que sea la señal portadora, se mete al DSP la señal de información en forma digital; y sólo se deben de multiplicar las muestras de la portadora con las de la señal de información una a una, el resultado es la señal modulada en amplitud.

Un ejemplo sencillo es el siguiente:

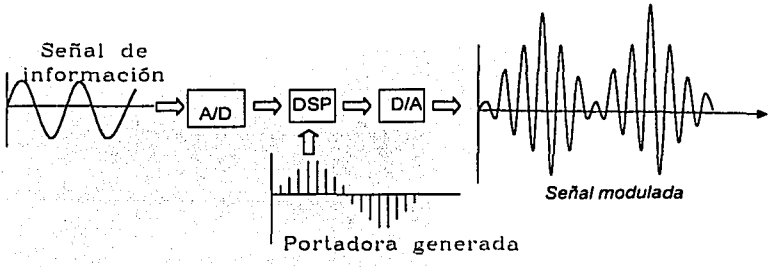


Figura 5.11 Ejemplo de modulación en amplitud

En el ejemplo que damos, la frecuencia de la señal que se genera es de 1.7 khz, por lo tanto; la señal de la información debe de ser de unos 200 hz, para poder observar la modulación adecuadamente.

El código para el ejemplo es el siguiente;

```
*****************************************
* PROGRAMA PARA LA MODULACION AM *
*****************************************

      .mmregs
      .ds 0f00h
TA    .word 24
RA    .word 24
;
TB    .word 18
RB    .word 18
;
AIC_CTR .word 09h
***************
* VARIABLES
***************

XN        .word 0          ; localidad para guardar la muestra entrante
Y         .word 17980      ; valor normalizado para la amplitud de la señal generada
Y1        .word 00000h     ; localidad temporal
COEFF     .word 19261      ; frecuencia normalizada de la señal generada
*
      .ps 0080ah
rint:  B RECEIVE
      .ps 0a00h
      .entry
*
* INICIALIZACION DEL DSP
*---------------------------------------
      SETC INTM
      LDP #0
      OPL #0834h,PMST
      LACC #0
      SAMM CWSR
      SAMM PDWSR
      SETC SXM
      SPLK #022h,IMR

      CALL AICINIT
      SPLK #12h,IMR
      CLRC OVM
```

```
    SPM 0
    CLRC INTM

WAIT:  NOP
    NOP
    B WAIT

* REALIZACION DE LA SEÑAL PORTADORA Y LA MODULACIÓN
*
RECEIVE:
    LDP #Y              ; apunta a la página donde está Y


    LAMM DRR            ; guarda la muestra de la señal entrante (información)
    AND #0FFFch         ; limpia los bits de comunicación
    SACL XN             ; guarda la muestra entrante en XN
    ZPR                 ; limpia el registro producto

* GENERACION DE LA SEÑAL PORTADORA

    LACC Y1,15          ; carga Y1 en ACCH
    NEG                 ; niega el acumulador
    MACD COEFF,Y        ; multiplica COEFF con Y
    APAC
    APAC                ; 2*coeff*y-y1
    SACH Y,1            ; guarda la muestra generada en Y
    LACC Y,15           ;
    AND #0fffch,15      ; pone en cero los dos primeros bits de Y
    RPT #14
    SFR
    NOP
    LT Y                ; carga en el registro TREG la muestra generada
    MPY XN              ; multiplica la muestra generada por la muestra entrante
    APAC                ; suma el resultado
    BSAR 14             ; corre 14 bits la muestra de salida a la derecha
    AND #0FFFch         ; pone en ceros los bits de comunicación para decir que es
            ; dato y no comando

    SAMM DXR            ; escribe la muestra de salida en el registro de transmisión
    RETE
```

La siguiente rutina es la misma que la que se usó en los filtros digitales y es para la configuración del convertidor A/D y D/A de la tarjeta.

```
*****************************************************
* INICIALIZA AL TLC320C46, CONVERTIDOR A/D Y D/A
*****************************************************

AICINIT: SPLK #20h,TCR
      SPLK #01h,PRD
      MAR *,AR0
      LACC #0008h
      SACL SPC
      LACC #00c8h
      SACL SPC
      LACC #080h
      SACH DXR
      SACL GREG
      LAR AR0,#0ffffh
      RPT #10000
      LACC *,0,AR0
      SACH GREG
      ;-----------------------------
      LDP #TA
      SETC SXM
      LACC TA,9
      ADD RA,2
      CALL AIC_2ND
      ;-----------------------------
      LDP #TB
      LACC TB,9
      ADD RB,2
      ADD #02h
      CALL AIC_2ND
      ;-----------------------------
      LDP #AIC_CTR
      LACC AIC_CTR,2
      ADD #03h
      CALL AIC_2ND
      RET

AIC_2ND:
      LDP #0
      SACH DXR
      CLRC INTM
      IDLE
      ADD #6h,15
      SACH DXR
      IDLE
      SACL DXR
      IDLE
```

```
LACL #0
SACL DXR
IDLE
SETC INTM
RET
.END
```

Cuestionario

1.- Defina que es el ancho de banda

2.-¿Cual es el ancho de banda de ?
1) Voz
2) Vídeo
3) Teléfono

3.-Enuncie el teorema de desplazamiento de frecuencias y como se relaciona con el teorema de modulación

4.-Enuncie el teorema de reciprocidad de la convolución y como se relaciona con la modulación AM

5.-¿Cual es la diferencia entre bit y baudio?

6.-Explique que es la modulación AM con doble lateral y portadora suprimida DSB-SC

7.-Defina la modulación AM con doble banda lateral y gran portadora DSB-LC.

Conclusiones y expectativas

Cuando planteamos nuestro tema de tesis, lo hicimos con la idea de desarrollar algún tema que no se hubiera cubierto al terminar nuestros estudios de la carrera de Ingeniero Mecánico Electricista en la FES Cuautitlán, entonces decidimos enfocarnos en los *procesadores digitales de señales*, ya que, aunque en la vida cotidiana estos se usan frecuentemente; ese tema no está incluido en el plan de estudios de la carrera. Para poder cubrir tanto la teoría como la práctica sobre estos dispositivos, junto con nuestro asesor de tesis, decidimos estructurar el trabajo con el objetivo de poder realizar prácticas sobre procesamiento digital de señales con algún DSP. A través de una investigación encontramos que el DSP TMS320C50 de Texas Instruments era apropiado para nuestros fines, por lo tanto decidimos adquirir el material necesario y comenzar con nuestro trabajo.

Decidimos comenzar nuestro trabajo con la teoría sobre filtros digitales, ya que el filtrado es una de las operaciones fundamentales en el procesamiento de señales. Necesitábamos entender esa teoría para después implementar los filtros con el DSP. También nos interesamos en observar las ventajas de los filtros digitales sobre los analógicos. Después pasamos a la teoría sobre los procesadores digitales de señales, esto es; que son, que hacen, como funcionan, etc. Luego nos enfocamos en el TMS320C50 que es un DSP de la cuarta generación producido por Texas Instruments, necesitábamos entender su estructura, funcionamiento y programación para poder trabajar con el.

Contando con los puntos anteriores, tuvimos la información necesaria para poder realizar prácticas con este dispositivo. Nuestra primera práctica fue diseñada para entender el funcionamiento de las principales instrucciones con que cuenta el DSP, aquí podemos decir que nuestra práctica cumple con los requerimientos necesarios para que cualquier persona interesada en trabajar con DSP aprenda a programar el TMS320C50 y pueda entender el funcionamiento de los DSP's en general, ya que todos trabajan de una forma similar, entonces se puede emigrar a otros dispositivos más avanzados.

Sobre las prácticas en tiempo real, podemos decir que una ves que obtuvimos el código fuente para implementar algún filtro digital, ya sea de tipo FIR o IIR, pudimos implementar cualquier tipo de filtro, ya sea pasa bajas, pasa bandas, pasa altas o supresor de banda, debido a que para cambiar de un diseño

de filtro a otro diferente, sólo es necesarios cambiar los coeficientes del filtro y el resto del programa es el mismo. Esa es una característica que nos interesaba demostrar al inicio de nuestro trabajo y lo logramos exitosamente, ya que con los programas desarrollados se pode implementar cualquier tipo de filtro, solo será necesario cambiar los coeficientes en el programa, en cambio para lograr el mismo resultado en el caso de filtros analógicos sería necesario cambiar muchos de los componentes eléctricos de nuestro circuito (capacitares, inductores, resistencias, etc.), por lo tanto podemos decir que en ese sentido, los sistemas digitales son más flexibles que los analógicos. También implementamos un modulador en amplitud, el cuál es muy ilustrativo para enseñar la parte práctica sobre este tema.

Podemos concluir que terminamos nuestro trabajo exitosamente, ya que cumplimos con nuestro objetivo principal que fue el desarrollo del tema de procesadores digitales y lo hicimos tanto en forma teórica como práctica. También complementamos nuestro conocimiento sobre el teme de filtros digitales, ya que este tema no se desarrolla de una forma completa durante el estudio de la carrera.

Una de nuestras expectativas es que nuestro trabajo sirva como complemento para futuros pasantes de ingeniería que se interesen en este tema, además, como la materia de procesamiento digital de señales de la FES Cuautitlán no cuenta con un laboratorio, el desarrollo de este tipo de prácticas sirve para complementar el conocimiento teórico con la práctica, para ello donamos nuestro material adquirido con la idea de que se pueda plantear el inicio de un laboratorio para esa materia.

Apéndice *A* Uso de instrucciones de MATLAB

MATLAB(Matrix LABoratory) es un programa que trabaja con datos en forma vectorial o matricial si trabajamos con escalares los expresamos de la forma matricial 1 x1

Formatos de salida

Los formatos de salida se escogen con la instrucción ***format*** , esta instrucción debe aplicarse antes de realizar cualquier calculo para indicar el tipo de formato deseado .

» format short Adapta el resultado de salida a cinco dígitos
» format long Adapta el resultado de salida a 15 dígitos
» format short e Adapta el resultado con un base de 5 dígitos más el exponente
» format long e Adapta el resultado con base de 15 dígitos más el exponente

Diseño de filtros usando plantillas analógicas.

Las siguientes instrucciones indican como calcular los coeficientes de los filtros con plantilla Butterworth y Chevychev.

El significado de las instrucciones es:

N	*Orden del filtro*
WN	*Frecuencia de corte /frecuencia de muestreo/2*
T	*Frecuencia de muestreo*
high	*Pasa altas*
low	*Pasa bajas*
stop	*Supresor de banda*
bandpass	*Pasa banda*
Wm=[w1 w2];	*w1 = Frecuencia baja w2= Frecuencia alta*
R	*Riso en la banda paso*
A	*Atenuación*
[b,a]=butter(N,WN, 'low') *[b,a]=butter(N,Wm/T, 'stop')* *[b,a]=cheby1(N,R,WN, 'high')* *[b,a]=cheby1(N,R,Wm, 'bandpass')* *[b,a]=cheby2(N,A,WM, 'low')*	

Para mayor información teclear el comando help para cada instrucción por ejemplo
» help butter

Si desea visualizar su función de transferencia teclee el comando printsys a,b,'z')

En algunos casos es necesario implementar filtros en cascada para reducir el error por efectos del largo de la palabra en los filtros recursivos, los filtros se descomponen en funciones de transferencia de segundo grado.

$$H(z)=\prod_{k=1}^{L}H(z)=\prod_{k=1}^{L}\frac{b_{0k}+b_{1k}z^{-1}+b_{2k}z^{-2}}{a_{0k}+a_{1k}z^{-1}+a_{2k}z^{-2}}$$

Por lo que usamos la instrucción:

sos =ss2sos(A,B,C,D) esta función nos devuelve los coeficientes de los polinomios de segundo grado correspondiente a un sistema

Diseño de filtros FIR.

Las siguientes instrucciones indican como diseñar filtros del tipo respuesta finita al impulso.

N	*Grado del filtro N+1*
W	*Frecuencia de corte digital*
Boxcar(N)	*Ventana rectangular o sin ventana el mismo efecto que noscale*
Blackman(N+1)	*Ventana de Blackman con N+ 1 coeficientes*
Hanning(N+1)	*Ventan de Hanning con N+1 coeficientes*
noscale	*Sin ventana ó con ventana rectangular*
Wm=[w1 w2]	*w1= frecuencia baja w2= frecuencia alta*
'high'	*Pasa altas*
'stop'	*Supresor de banda*
'dc-0'	*Pasa banda*
'dc-1'	*Pasa baja*
b=fir1(N,'W,'high',blackman(N+1)) *b=fir1(N,Wm,'DC-0',hanning(N+1))*	

Los valores obtenidos con la función ***fir1*** devuelve valores de los coeficientes de un filtro usando el método de ventanas, ***fir1*** utiliza la ventana de hamming, con esta ventana entrega resultados si se quiere cambiar el tipo de la ventan es necesario especificar el tipo de ventana usa, El Tamaño de la ventana para es de N+1 veces el número de coeficientes.

Para mayor información sobre la función ***fir1*** invoque la instrucción help dentro del ambiente de matlab >>***help fir1.***

Graficas de filtros

La instrucción *freqz(b,a,m,Fs),* nos da la gráfica de la respuesta en frecuencia de un sistema donde b es el numerador y a es el denominador, Fs es la frecuencia de muestreo, los puntos de la gráfica para mayor información invocar la función help freqz

Ejemplo 1. A continuación se presenta un programa realizado en Matlab para diseñar un filtro supresor de banda usando un polinomio normalizado de orden tres, con una frecuencia suprimida de 500 Hz a 1000 Hz, dando su respuesta en frecuencia y los coeficientes del sistema en forma directa y en cascada.

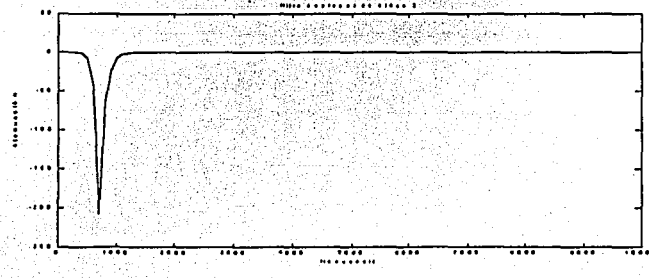
```
» W=[500 1000];
» [b,a]=butter(3,W/10000,'stop');
» printsys(b,a,'z')
```

Los resultados son;

```
num/den =

  0.8545 z^6 - 5.0007 z^5 + 12.3187 z^4 - 16.3448 z^3 + 12.3187 z^2

  - 5.0007 z + 0.8545
  ---------------------------------------------------
  z^6 - 5.5461 z^5 + 12.9532 z^4 - 16.3035 z^3 + 11.663 z^2 - 4.4966 z

  + 0.73017
» [c,d,e,f]=butter(3,W/10000,'stop');
» sos=ss2sos(c,d,e,f)
sos =

  0.9270  -1.8084   0.9270   1.0000  -1.8084   0.8541
  0.7659  -1.4940   0.7659   1.0000  -1.8177   0.9037
  1.2035  -2.3478   1.2035   1.0000  -1.9200   0.9460
```

Las tres primeras columnas corresponden a los coeficientes de los polinomios en el numerador, las tres ultimas columnas a los polinomios en el denominador.

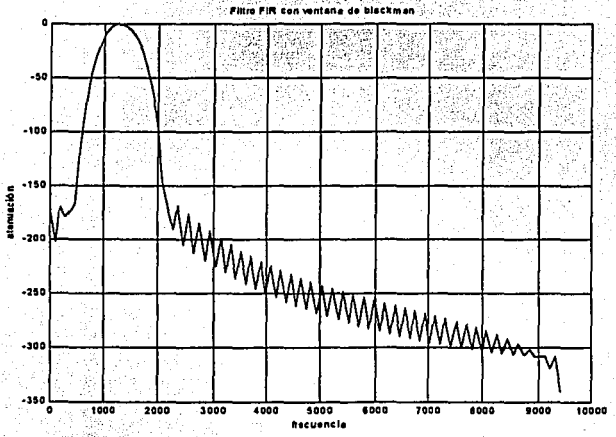
```
» [h1,w1]=freqz(b,a,100);
» f=w1*10000/pi;
» plot(f,20*log(abs(h1))),title('filtro supresor de orden 3'),xlabel('frecuencia'),ylabel('atenuación')
```



Diseño de un filtro FIR pasa banda usando ventana de Blackman, la banda de paso de 1000 a 1600 Hz con un a ventana de tamaño de 100 y periodo de muestreo de 19000 hz

```
» plot(f,20*log(abs(h1)))
» W=[1000 1600];
» b=fir1(100,W/9500,'dc-0',blackman(101));
```

```
» [h1,w1]=freqz(b,1,100);
» f=w1*9500/pi;
» plot(f,20*log(abs(h1)),'k'),grid,title('Filtro FIR con ventana de
blackman'),xlabel('frecuencia'),ylabel('atenuación')
»
```



Apéndice *B* Lista de las principales instrucciones del TMS320C50

Mnemónico	Descripción
ABS	Da el valor absoluto del acumulador, si el acumulador es menor que cero este será reemplazado por el complemento a dos del valor anterior. En caso contrario, esta instrucción no afecta al acumulador. No utiliza operadores
ADD	Suma al acumulador el contenido de una dirección de memoria. El resultado se almacena en el acumulador
ADDB	El contenido del acumulador buffer (ACCB) es sumado al acumulador. El bit C se pone en uno si el resultado de la suma genera acarreo. No utiliza operadores
ADC	El contenido de la localidad de memoria y el valor del bit de acarreo son sumados al acumulador con extensión del signo .El bit de acarreo es afectado de manera normal
ADDS	Suma a la parte baja del acumulador el contenido de una dirección de memoria sin tomar en cuenta el signo. El resultado se almacena en el acumulador
ADDT	El valor en la memoria de dato es corrido a la izquierda de 0 a 15 bits especificado por TREG1 y sumado al acumulador, reemplazando el contenido del acumulador . El dato es tratado como un número no signado de 16 bits. C es afectado de manera normal
ADRK	El valor de una constante inmediata de 8 bits es sumada, justificado, a la derecha, en el actual registro auxiliar (como se especifica en al actual ARP) reemplazando el contenido del registro auxiliar con el resultado, la suma se lleva acabo en la unidad ARUA con el valor inmediato tratado como un entero positivo de 8 bits observe que todas las operaciones del registro auxiliar son iguales
AND	Se realiza una operación lógica AND en la parte baja del acumulador con el contenido de la dirección especificado de memoria o una constante de 16 bits, la parte alta del acumulador se llena de ceros.
ANDB	Se realiza una operación lógica AND entre el valor del acumulador y el acumulador buffer (ACCB) el resultado es colocado en el acumulador mientras el acumulador buffer no se ve afectado. No utiliza operadores
APAC	Suma el acumulador el contenido del registro producto P el resultado se almacena en el acumulador: El registro P es corrido antes de la suma tal y como se especifica en el registro PM. No utiliza operadores
APL	Si es especificada una constante inmediata larga, se hace AND de ésta con el valor de memoria de datos dma. DE otra manera se aplica un AND el valor de la memoria de dato y el contenido del registro de manipulación dinámico de bits (DBMR). En los dos casos. El resultado es escrito en una nueva localidad de memoria de dato, mientras que el contenido del acumulador no se ve afectado, si el resultado de la operación lógica AND es 0, entonces el bit TC se pone en uno si sucede lo contrario el bit TC se pone en cero
B	Salta incondicionalmente a una localidad especificada del programa

Mnemonico	Descripción
BLLD	Una palabra en memoria de dato apuntado por el src (fuente) es copiada a un espacio de memoria que es apuntado por el dst (destino)
BLDP	Una palabra en la memoria de dato es copiada a una palabra en el espacio de memoria de programa que apunta el registro BMAR
CALA	La instrucción CALA es utilizada para realizar llamadas computadas a subrurrutinas: El PC es incrementado y almacenado en el stack. Luego el contenido de los 16 bits menos significativos del acumulador se carga al aPC
CALL	Llamada a subrutina. El contador de programa es incrementado y almacenado en el STACK Entonces la dirección especificada de los 16 bits menos significativos del acumulador se carga en el PC
CC	Llamada condicional a subrutina. El control se transfiere al "pma" si se cumple las condiciones especificadas, Se utilizan las mismas condiciones de la instrucción BNCD
CLRC	El bit de control especificado e puesto a cero lógico, Observe que la instrucción LST puede ser utilizada para cargar ST0 y ST1, los bits de control son: C, CNF, HM, OVM, SXM, TC y XF
CMPL	Complemento del acumulador. El contenido del acumulador es reemplazado con la inversión lógica (Complemento a uno). El bit de acarreo no se ve afectado. No utiliza operadores
CPL	Compara constantes de 16 bits con una localidad en memoria: Si las dos cantidades involucradas en una comparación son iguales, El bit TC se pone a 1 en cualquier otro caso , el bit TC se pone a cero
CRGT	El contenido del acumulador (ACC) es comparador con el contendido del acumulador (ACCB). El valor mayor (signado) es cargado en ambos registros. Si el contenido del acumulador buffer, el bit de acarreo se pone en uno. El bit de acarreo se pone en cero en cualquier otro caso. No utiliza operadores
BANZ	Salta a la localidad especificada si el contenido del registro auxiliar en uso no es cero. Cuando se ejecuta esta instrucción, el registro auxiliar es decrementado. Si el registro auxiliaren uso es cero el programa ejecuta la siguiente instrucción
BNCD	Se realiza un salto a la dirección de memoria del programa "pma" si se cumple las condiciones especificadas. Condiciones EQ=0 ACC=0 LT ACC<0 GT ACC>0 C C=1 OV OV=1 BIO BIO=0 NTC TC=0 NEQ ACC=0 LEQ ACC<0 GEQ ACC>0 NC C=0 NOV OV=0 TC TC=1 UNC sin condición

Mnemónico	Descripción
CRLT	El contenido del acumulador (ACC) es comparado con el contenido del acumulador buffer (ACCB). El valor menor (signado) es cargada en ambos registros. Si el contenido del acumulador es menor que el contenido del acumulador buffer, el bit de acarreo se pone en 1. El bit de acarreo se pone en cero en cualquie otro caso. No utiliza operadores
DMOV	El contenido de la dirección de memoria especificada es copiada en el contendido de la siguiente dirección más alta
EXAR	El contenido del acumulador es cambiado (switchado) con el contenido del acumulador buffer (ACCB) y viceversa. No utiliza operadores
IDLE	La instrucción obliga a que el programa permanezca en estado de espera hasta que ocurra una interrupción no mascarable o se presente un reset. El contador de programa PC es incrementado sólo una vez, y el dispositivo permanece en estado de espera hasta que se presente una interrupción. El puerto serial y el timer permanecen activos. No utiliza operadores
IDLE2	Similar a IDLE solo que el puerto serial y el timer no están activos no utiliza operadores
IN	La instrucción IN lee datos de un periférico y coloca estos en memoria de datos, esto toma dos ciclos de instrucción
INTR	Es la instrucción por software que transfiere el control del programa a la dirección de memoria de programa especificada por el vector de interrupción "k"
LACB	El acumulador es cargado en el contenido del acumulador buffer ACCB. No utiliza operadores
LACC	El contenido de una localidad en memoria dato o una constante de 16 bits y efectúa el número de corrimiento a la izquierda especificados durante el corrimiento, los bits de orden bajo son llenados con cero y los bits de orden alto son iguales si el bit SXM = 1
LACL	El contenido de la localidad de memoria de la dirección de memoria o una constate de 8 bits es cargado en el ACCL, El ACCH es llenado con ceros, Esta instrucción no utiliza la extención de signo (no importa el valor del bit SXM)
LAMM	La parte baja del acumulador (ACCL) es cargado con el contenido del registro en memoria mapeada. La parte alta del acumulador (ACCH) es cargada con ceros. Los 9 bits más significativos de la dirección de memoria son llenados con cero independientemente del valor del registro apuntador de página (DP)
LAR	Carga el registro auxiliar con el contenido de la localidad en memoria o una constante de 16 bits
LDP	Carga el registro apuntador de pagina con una constante de 9 bits o el contenido de la localidad en memoria
LAMMR	El regisro de memoria mapeado es apuntado por los 7 bits más bajos del valor de la dirección en memoria del dato, es cargado en el contendido de la localidad de memoria de datos direcci0onada por la dirección de 16 bits

Mnemónico	Descripción
LTA	El contenido de la dirección de datos en memoria especificado es cargado en el

	registro TRG0 y el registro P que contiene el producto previo es sumado al acumulador
LTD	El registro TREG es cargado con el contenido de la dirección de datos especificada , el contenido del registro P es sumado al acumulador y el contenido de la dirección de memoria de datos y el registro producto corrido como lo define PM es almacenado en el acumulador
LTP	TREG0 es cargado con el contenido de la localidad de dirección de memoria de dato y el registro producto corrido como lo define PM es almacenado en el acumulador
LTS	TREG0 es cargado con el contenido de la localidad en memoria de dato, El registro de dato corrido como lo define P, es restado del acumulador. El resultado es colocado en el acumulador
MAC	Multiplica el valor de memoria de adatos (Especificada por el dma) por el valor de memoria de programa (especificado por el pma): También es sumado el producto previo corrido como lo define PM el acumulador
MACD	Multiplica el valor de memoria de dato (especificado por el dma) por un valor de memoria de programa (¿especificado por el pma). También suma el producto anterior con un corrimiento especificado por PM al acumulador. El contenido de memoria de dato especificado es copiado en la próxima dirección de memoria de dato
MDD	La instrucción multiplica un valor de memoria de dato (especificado por el dma) por un valor de memoria de programa (especificado por el pma). La dirección de memoria de programa está contenida en el registro de memoria BMAR. Esto facilita un direccionamiento dinámico de tablas de coeficientes. También suma el producto anterior con un corrimiento especificado por PM al acumulador. Además es contenido de memoria, dato especificado es copiado en la próxima dirección de memoria
MADS	La instrucción multiplica un valor de memoria de datos(especificado por el dma) por un valor de memoria de programas especificado por el pma) .También suma el producto anterior con un corrimiento especificado por el PM del acumulador, El pma es especificado por el contenido del registro BMA, en vez de una constante inmediata largo, Esto permite el direccionamiento dinámico de tablas de coeficientes
MAR	Esta instrucción utiliza direccionamiento de modo indirecto para incrementar o decrementar el registro auxiliar en uso y cambiar el apuntador de registros auxiliares
MPY	El contenido del registro TREG0 es multiplicado por el contenido de la dirección de datos de memoria, el resultado es almacenado en el registro P, El direccionamiento inmediato corto multiplica el registro por una constante de 13 bits con signo
MPYA	El contenido del registro TREG0 es multiplicado por el contenido de la localidad de memoria de dato, El resultado se almacena en el registro P, el producto previo corrido como lo define PM es sumado al acumulador

Mnemónico	Descripción
RPT	Repite n+1 veces la siguiente instrucción. El contador de repetición (RPTC) es carado con la localidad de memoria de dato si es usado el modo de

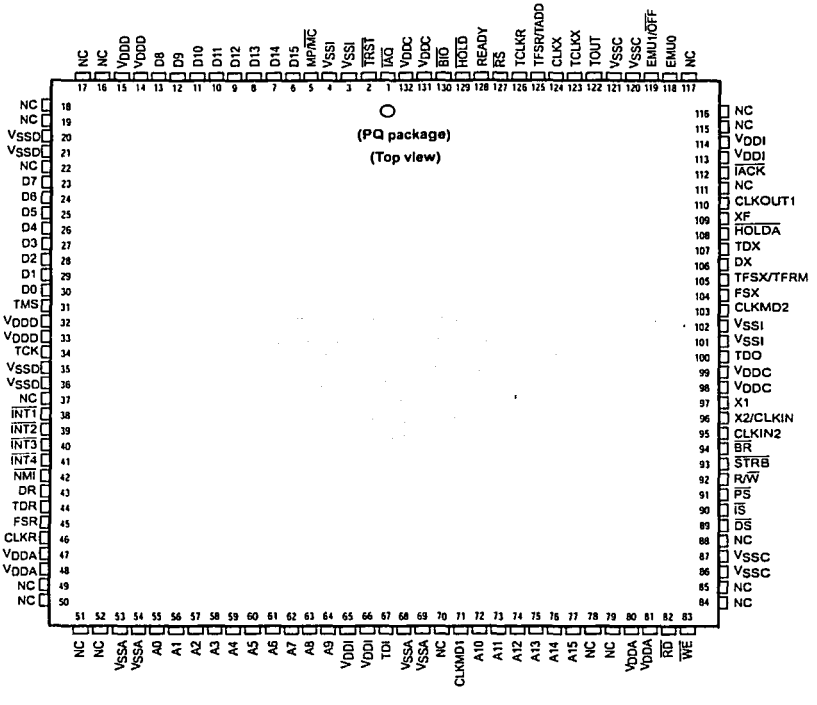
	direccionamiento directo o el modo indirecto, un valor inmediato de 8 bits es usado si se usa direccionamiento inmediato corto, o un valor inmediato largo, La instrucción siguiente RPT es repetida n+1 veces, donde n es la constante especificada en la instrucción, La instrucción RPT no es interrumpible
RPTB	Permite que un bloque de instrucciones pueda repetirse un número de veces especificado por el registro contador de repetición de bloque (BRC)
RPTZ	Limpia el acumulador y el registro de productos y repite la instrucción que sigue n +1 veces
SACB	El contenido del acumulador es copiado al acumulador buffer. No utiliza operadores
SACH	Almacena la parte alta del acumulador en memoria de dato. Con corrimiento a la izquierda (0-7) especificado en la instrucción. Durante el corrimiento de los bits altos ACC son perdidos
SACL	Almacena la parte alta del acumulador en memoria de dato. Con corrimiento a la izquierda (0-7) especificando en la instrucción. Durante el corrimiento los bits bajos son llenados con ceros
SAMM	La parte baja del acumulador (ACCL) es copiada a al registro de memoria mapaeda
SAR	Guarda el registro auxiliar especificado en la localidad de memoria
SBB	El contenido del acumulador buffer (ACCB) es restado del contenido del acumulador, y el acumulador buffer no se ve afectado, el bit de acarreo es puesto a cero si el resultado de las instrucciones genera un préstamo
SBBB	El contenido del acumulador buffer (ACCB) y la inversión lógica del bit de acarreo son restados del acumulador (ACC) los resultados son almacenados en el acumulador, y el acumulador buffer no se ve afectado. El bit de acarreo es puesto a cero si el resultado genera un préstamo
SBRK	El valor inmediato de una constante de 8 bits es restado, justificado a la derecha, al actual registro auxiliar en uso, La sustracción se realiza en el ARAU, tratando la constante inmediata como un entero de 8 bits
SETC	Un bit de control especificado es puesto a uno. Los bits de control pueden ser: C, CNF, INTM, OVM, TX y XF
SFL	El contenido del ACC es corrido por un a la izquierda, el MSB del ACC es corrido al bit de carry, los bits LSB del ACC son llenados con ceros. No es afectado por el SXM
SFR	El contenido de ACC es corrido un bit a al derecha. El tipo de corrimiento es determinado por el bit SSXM = 0 entonces el corrimiento lógico, el MSB del ACC es llenado con ceros, el LSB del ACCC es corrido al bit C, Si SXM = 1,entonces se produce un corrimiento aritmético, el MSB no cambia y e copiado al bit 30 del ACC. El LSB se acopia en el bit C
SFRB	El ACC y el ACCB son corridos un bit a la derecha. El LSB del ACC se copia en el MSB del ACCB, el LSB del a CCB se copia en el bit C el MSB del ACC es corrido dependiendo del valor del SXM de manera similar que en la instrucción SFR
SMMR	El valor del registro de memoria mapeada es almacenada en la localidad de memoria dato especificado
Mnemónico	Descripción
SPAC	El contenido del registro P corrido como lo especifica PM se resta al acumulador, el resultado se almacena en el mismo acumulador
SPH	Los bits de mayor orden del registro P son almacenados en memoria de dato

	Afectado por PM
SPL	Los bits de menor orden del registro P son almacenados en memoria de data Afectado por PM
SPLK	Permite almacenar una constante de 16 bits en cualquier localidad de memoria
SPM	Una constante de dos bits es copiada en el campo PM de registro de estado ST1
SQRA	El contenido del registro TREG0 es elevado al cuadrado y el contenido el registro P es sumado al acumulador con corrimiento especificado por PM
SST	Los registros de estado (ST0 o ST1) son almacenados en la dirección especificada de memoria dato
SUB	El contenido de la dirección específica de memoria de dato o una constatnte corrida a la izquierda se resta al acumulador. Durante el corrimiento, los bits de orden bajo se llenan con ceros y se coloca el signo del bit más significativo SXM =1
SUBB	El contendido de la localidad de memoria dato y la inversión lógica de bits de acarreo es restada al acumulador con extensión de signo comprimido
SUBS	El contenido especificado en memoria de datos se resta al acumulador sin considerar el bit de signo. El contenido de la dirección especificada de memoria dato se considera como un entero positivo de 16 bits
SUBT	El valor de la dirección especificada de memoria el dato se corre a la izquierda de 0 a 15 bits especificado por TREG1 y es restado al acumulador
TBLR	Transfiere una palabra desde cualquier lugar de memoria d programa a la localidad especificada de memoria de dato. La dirección pma es especificada en la parte baja del ACC
TBLW	Transfiere una palabra desde la localidad especificada de memoria de dato a la localidad RAM externa del programa, la dirección pma por la parte baja del ACC
TRAP	Interrupción por software que transfiere el control a la localidad de memoria dato 22h
XC	Las siguientes dos instrucciones de una sola palabra o la siguiente instrucción de dos palabras se ejecuta si se cumple cierta condición. El número de palabra (instrucción de una palabra) a ejecutar se especifican en la instrucción. Utilizan la misma condición que BCND
XOR	Efectúa el OR exclusivo del contenido de la localidad memoria de dato especificada o una constante de 16 bits con corrimiento de 16 bits con corrimiento con la parte baja del acumulador
XORB	Se realiza la operación OR exclusiva entre el contendido del cumulador buffer y el contenido del acumulador. El resultado se escribe en el ACC y el ACCB no se ve afectad. No utiliza operadores
XPL	XOR entre el valor de memoria de datos y la constante larga especificada. El resultado es escrito de nuevo en la localidad de memoria especificada
ZALR	Carga el valor de memoria de dato en la parte alta del acumulador la instrucción redondea el resultado en el ACC, es decir, suma un uno al bit 15 en el ACC y llena con ceros los bits 0-14 del ACC
ZAP	El acumulador y el registro de producto se cargan con ceros. No utiliza operandos
ZPR	El registro P se carga con ceros. No utiliza operandos

A.4 132-Pin BQFP Pinout ('C50, 'C51, and 'C53)

Refer to Figure A–4 and Table A–4 for pin/signal assignments of the 'C50, 'C51, and 'C53 in the 132-pin BQFP package.

Figure A–4. Pin/Signal Assignments for the 'C50, 'C51, and 'C53 in 132-Pin BQFP



Note: NC These pins are not connected (reserved).

Table A–4. Signal/Pin Assignments for the 'C50, 'C51, and 'C53 in 132-Pin BQFP

Signal	Pin	Signal	Pin	Signal	Pin	Signal	Pin	Signal	Pin
A0	55	D6	24	$\overline{RD}$	82	V_{DDI}	65	†	51
A1	56	D7	23	$\overline{RS}$	127	V_{DDI}	66	†	52
A2	57	D8	13	R/$\overline{W}$	92	V_{SSA}	53	†	70
A3	58	D9	12	$\overline{STRB}$	93	V_{SSA}	54	†	78
A4	59	D10	11	TCK	34	V_{SSA}	68	†	79
A5	60	D11	10	TCLKR	126	V_{SSA}	69	†	84
A6	61	D12	9	TCLKX	123	V_{SSC}	86	†	85
A7	62	D13	8	TDI	67	V_{SSC}	87	†	88
A8	63	D14	7	TDO	100	V_{SSC}	121	†	111
A9	64	D15	6	TDR	44	V_{SSC}	120	†	115
A10	72	DR	43	TDX	107	V_{SSD}	20	†	116
A11	73	$\overline{DS}$	89	TMS	31	V_{SSD}	21	†	117
A12	74	DX	106	TOUT	122	V_{SSD}	35		
A13	75	EMU0	118	$\overline{TRST}$	2	V_{SSD}	36		
A14	76	EMU1/$\overline{OFF}$	119	TFSR/TADD	125	V_{SSI}	3		
A15	77	FSR	45	TFSX/TFRM	105	V_{SSI}	4		
$\overline{BR}$	94	FSX	104	V_{DDC}	131	V_{SSI}	101		
$\overline{BIO}$	130	$\overline{HOLD}$	129	V_{DDC}	132	V_{SSI}	102		
CLKIN2	95	$\overline{HOLDA}$	108	V_{DDA}	47	$\overline{WE}$	83		
CLKMD1	71	$\overline{IACK}$	112	V_{DDA}	48	X1	97		
CLKMD2	103	$\overline{IAQ}$	1	V_{DDA}	80	X2/CLKIN	96		
CLKOUT1	110	$\overline{INT1}$	38	V_{DDA}	81	XF	109		
CLKR	46	$\overline{INT2}$	39	V_{DDC}	98	†	16		
CLKX	124	$\overline{INT3}$	40	V_{DDC}	99	†	17		
D0	30	$\overline{INT4}$	41	V_{DDD}	14	†	18		
D1	29	$\overline{IS}$	90	V_{DDD}	15	†	19		
D2	28	MP/$\overline{MC}$	5	V_{DDD}	32	†	22		
D3	27	$\overline{NMI}$	42	V_{DDD}	33	†	37		
D4	26	$\overline{PS}$	91	V_{DDI}	113	†	49		
D5	25	READY	128	V_{DDI}	114	†	50		

† These pins are not connected (reserved).

SLAS014E – SEPTEMBER 1987 – REVISED MAY 1995

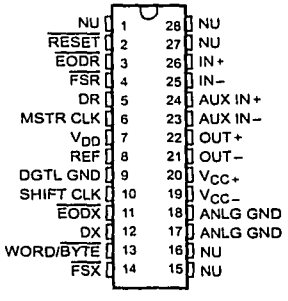
- **14-Bit Dynamic Range ADC and DAC**
- **Variable ADC and DAC Sampling Rate Up to 19,200 Samples per Second**
- **Switched-Capacitor Antialiasing Input Filter and Output-Reconstruction Filter**
- **Serial Port for Direct Interface to TMS32011, TMS320C17, TMS32020, and TMS320C25 Digital Signal Process**
- **Synchronous or Asynchronous ADC and DAC Conversion Rate With Programmable Incremental ADC and DAC Conversion Timing Adjustments**
- **Serial Port Interface to SN74299 Serial-to-Parallel Shift Register for Parallel Interface to TMS32010, TMS320C15, or Other Digital Processors**
- **600-Mil Wide N Package (C_L to C_L)**
- **2s Complement Format**
- **CMOS Technology**

PART NUMBER	DESCRIPTION
TLC32040	Analog interface circuit with internal reference. Also a plug-in replacement for TLC32041.
TLC32041	Analog interface circuit without internal reference

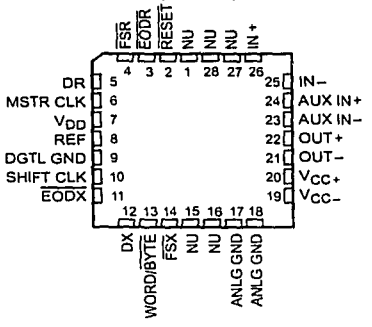
description

The TLC32040 and TLC32041 are complete analog-to-digital and digital-to-analog input/output systems, each on a single monolithic CMOS chip. This device integrates a bandpass switched-capacitor antialiasing input filter, a 14-bit-resolution A/D converter, four microprocessor-compatible serial port modes, a 14-bit-resolution D/A converter, and a low-pass switched-capacitor output-reconstruction filter.

N PACKAGE
(TOP VIEW)



FN PACKAGE
(TOP VIEW)



NU – Nonusable; no external connection should be made to these terminals.

AVAILABLE OPTIONS

T_A	PACKAGE	
	PLASTIC CHIP CARRIER (FN)	PLASTIC DIP (N)
0°C to 70°C	TLC32040CFN TLC32041CFN	TLC32040CN TLC32041CN
–40°C to 85°C		TLC32040IN TLC32041IN

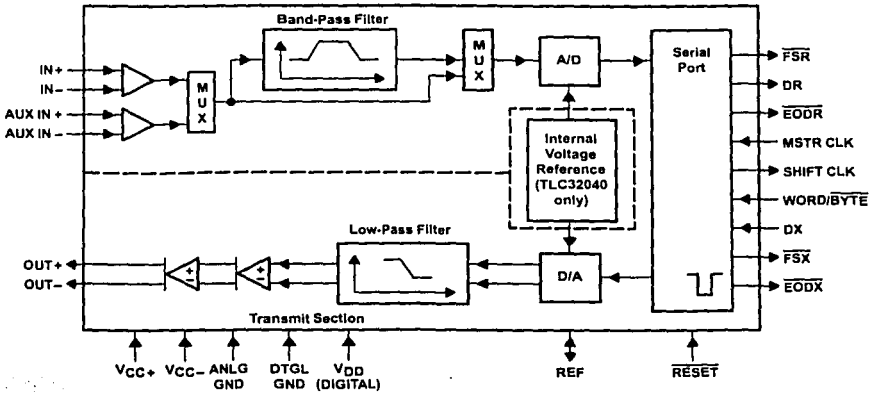
Please be aware that an important notice concerning availability, standard warranty, and use in critical applications of Texas Instruments semiconductor products and disclaimers thereto appears at the end of this data sheet.

FALTA

PÁGINA

2

functional block diagram



Terminal Functions

TERMINAL NAME	TERMINAL NO.	I/O	DESCRIPTION
ANLG GND	17,18		Analog ground return for all internal analog circuits. Not internally connected to DGTL GND.
AUX IN+	24	I	Noninverting auxiliary analog input state. This input can be switched into the bandpass filter and A/D converter path via software control. If the appropriate bit in the control register is a 1, the auxiliary inputs replace the IN+ and IN– inputs. If the bit is a 0, the IN+ and IN– inputs are used (see the AIC DX data word format section).
AUX IN–	23	I	Inverting auxiliary analog input (see the above AUX IN+ description)
DGTL GND	9		Digital ground for all internal logic circuits. Not internally connected to ANLG GND.
DR	5	O	DR is used to transmit the ADC output bits from the AIC to the TMS320 serial port. This transmission of bits from the AIC to the TMS320 serial port is synchronized with the SHIFT CLK signal.
DX	12	I	DX is used to receive the DAC input bits and timing and control information from the TMS320. This serial transmission from the TMS320 serial port to the AIC is synchronized with the SHIFT CLK signal.
$\overline{\text{EODR}}$	3	O	End of data receive. See the WORD/$\overline{\text{BYTE}}$ description and the Serial Port Timing diagrams. During the word-mode timing, $\overline{\text{EODR}}$ is a low-going pulse that occurs immediately after the 16 bits of A/D information have been transmitted from the AIC to the TMS320 serial port. $\overline{\text{EODR}}$ can be used to interrupt a microprocessor upon completion of serial communications. Also, $\overline{\text{EODR}}$ can be used to strobe and enable external serial-to-parallel shift registers, latches, or external FIFO RAM, and to facilitate parallel data bus communications between the AIC and the serial-to-parallel shift registers. During the byte-mode timing, $\overline{\text{EODR}}$ goes low after the first byte has been transmitted from the AIC to the TMS320 serial port and is kept low until the second byte has been transmitted. The TMS32011 or TMS320C17 can use this low-going signal to differentiate between the two bytes as to which is first and which is second. $\overline{\text{EODR}}$ does not occur after secondary communication.

Terminal Functions (continued)

TERMINAL NAME	TERMINAL NO.	I/O	DESCRIPTION
$\overline{EODX}$	11	O	End of data transmit. See the WORD/$\overline{BYTE}$ description and the Serial Port Timing diagram. During the word-mode timing, $\overline{EODX}$ is a low-going pulse that occurs immediately after the 16 bits of D/A converter and control or register information have been transmitted from the TMS320 serial port to the AIC. $\overline{EODX}$ can be used to interrupt a microprocessor upon the completion of serial communications. Also, $\overline{EODX}$ can be used to strobe and enable external serial-to-parallel shift registers, latches, or an external FIFO RAM, and to facilitate parallel data-bus communications between the AIC and the serial-to-parallel shift registers. During the byte-mode timing, $\overline{EODX}$ goes low after the first byte has been transmitted from the TMS320 serial port to the AIC and is kept low until the second byte has been transmitted. The TMS32011 or TMS320C17 can use this low-going signal to differentiate between the two bytes as to which is first and which is second.
$\overline{FSR}$	4	O	Frame sync receive. In the serial transmission modes, which are described in the WORD/$\overline{BYTE}$ description, $\overline{FSR}$ is held low during bit transmission. When $\overline{FSR}$ goes low, the TMS320 serial port begins receiving bits from the AIC via DR of the AIC. The most significant DR bit is present on DR before $\overline{FSR}$ goes low. (See Serial Port Timing and Internal Timing Configuration diagrams.) $\overline{FSR}$ does not occur after secondary communication.
FSX	14	O	Frame sync transmit. When FSX goes low, the TMS320 serial port begins transmitting bits to the AIC via DX of the AIC. In all serial transmission modes, which are described in the WORD/$\overline{BYTE}$ description, $\overline{FSX}$ is held low during bit transmission (see the Serial Port Timing and Internal Timing Configuration diagrams).
IN+	26	I	Noninverting input to analog input amplifier stage
IN–	25	I	Inverting input to analog input amplifier stage
MSTR CLK	6	I	Master clock. MSTR CLK is used to derive all the key logic signals of the AIC, such as the shift clock, the switched-capacitor filter clocks, and the A/D and D/A timing signals. The Internal Timing Configuration diagram shows how these key signals are derived. The frequencies of these key signals are synchronous submultiples of the master clock frequency to eliminate unwanted aliasing when the sampled analog signals are transferred between the switched-capacitor filters and the A/D and D/A converters (see the Internal Timing Configuration).
OUT+	22	O	Noninverting output of analog output power amplifier. OUT+ can drive transformer hybrids or high-impedance loads directly in either a differential or a single-ended configuration.
OUT–	21	O	Inverting output of analog output power amplifier. OUT– is functionally identical with and complementary to OUT+.
REF	8	I/O	Internal voltage reference for the TLC32040. For the TLC32040 and TLC32041 an external voltage reference can be applied to this terminal.
$\overline{RESET}$	2	I	Reset. A reset function is provided to initialize the TA, TA', TB, RA, RA', RB, and control registers. This reset function initiates serial communications between the AIC and DSP. The reset function initializes all AIC registers including the control register. After a negative-going pulse on $\overline{RESET}$, the AIC registers are initialized to provide an 8-kHz data conversion rate for a 5.184-MHz master clock input signal. The conversion rate adjust registers, TA' and RA', are reset to 1. The control register bits are reset as follows (see AIC DX data word format section): d7 = 1, d6 = 1, d5 = 1, d4 = 0, d3 = 0, d2 = 1 This initialization allows normal serial-port communication to occur between AIC and DSP.
SHIFT CLK	10	O	Shift clock. SHIFT CLK is obtained by dividing the master clock signal frequency by four. SHIFT CLK is used to clock the serial data transfers of the AIC, described in the WORD/$\overline{BYTE}$ description below (see the Serial Port Timing and Internal Timing Configuration diagrams).
V_{DD}	7		Digital supply voltage, 5 V ±5%
V_{CC+}	20		Positive analog supply voltage, 5 V ±5%
V_{CC-}	19		Negative analog supply voltage, –5 V ±5%

Terminal Functions (continued)

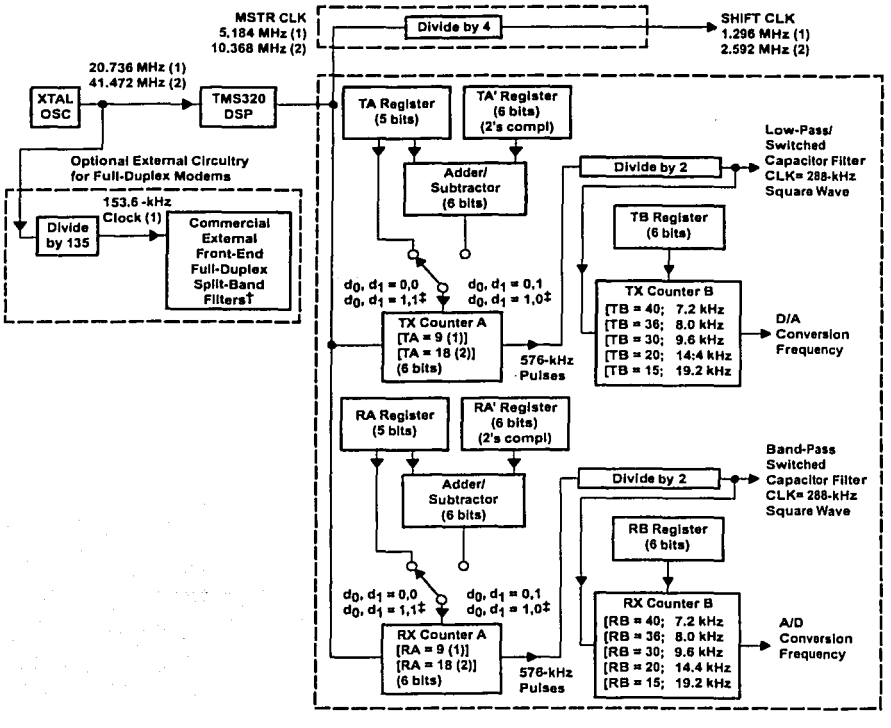
TERMINAL NAME	TERMINAL NO.	I/O	DESCRIPTION
WORD/$\overline{\text{BYTE}}$	13	I	WORD/$\overline{\text{BYTE}}$, in conjunction with a bit in the control register, is used to establish one of four serial modes. These four serial modes are described below. *AIC transmit and receive sections are operated asynchronously.* The following description applies when the AIC is configured to have asynchronous transmit and receive sections. If the appropriate data bit in the control register is a 0 (see the AIC DX data word format section), the transmit and receive sections are asynchronous. L Serial port directly interfaces with the serial port of the TMS32011 or TMS320C17 and communicates in two 8-bit bytes. The operation sequence is as follows (see Serial Port Timing diagrams). 1. $\overline{\text{FSX}}$ or $\overline{\text{FSR}}$ is brought low. 2. One 8-bit byte is transmitted or one 8-bit byte is received. 3. $\overline{\text{EODX}}$ or $\overline{\text{EODR}}$ is brought low. 4. $\overline{\text{FSX}}$ or $\overline{\text{FSR}}$ emits a positive frame-sync pulse that is four shift clock cycles wide. 5. One 8-bit byte is transmitted or one 8-bit byte is received. 6. $\overline{\text{EODX}}$ or $\overline{\text{EODR}}$ is brought high. 7. $\overline{\text{FSX}}$ or $\overline{\text{FSR}}$ is brought high. H Serial port directly interfaces with the serial port of the TMS32020, TMS320C25, or TMS320C30 and communicates in one 16-bit word. The operation sequence is as follows (see Serial Port Timing diagrams): 1. $\overline{\text{FSX}}$ or $\overline{\text{FSR}}$ is brought low. 2. One 16-bit word is transmitted or one 16-bit word is received. 3. $\overline{\text{FSX}}$ or $\overline{\text{FSR}}$ is brought high. 4. $\overline{\text{EODX}}$ or $\overline{\text{EODR}}$ emits a low-going pulse. *AIC transmit and receive sections are operated synchronously.* If the appropriate data bit in the control register is a 1, the transmit and receive sections are configured to be synchronous. In this case, the bandpass switched-capacitor filter and the A/D conversion timing are derived from the TX counter A, TX counter B, and TA, TA', and TB registers, rather than the RX counter A, RX counter B, and RA, RA', and RB registers. In this case, the AIC $\overline{\text{FSX}}$ and $\overline{\text{FSR}}$ timing are identical during primary data communication; however, $\overline{\text{FSR}}$ is not asserted during secondary data communication since there is no new A/D conversion result. The synchronous operation sequences are as follows (see Serial Port Timing diagrams). L Serial port directly interfaces with the serial port of the TMS32011 or TMS320C17 and communicates in two 8-bit bytes. The operation sequence is as follows (see Serial Port Timing diagrams): 1. $\overline{\text{FSX}}$ and $\overline{\text{FSR}}$ are brought low. 2. One 8-bit byte is transmitted and one 8-bit byte is received. 3. $\overline{\text{EODX}}$ and $\overline{\text{EODR}}$ are brought low. 4. $\overline{\text{FSX}}$ and $\overline{\text{FSR}}$ emit positive frame-sync pulses that are four shift clock cycles wide 5. One 8-bit byte is transmitted and one 8-bit byte is received. 6. $\overline{\text{EODX}}$ and $\overline{\text{EODR}}$ are brought high. 7. $\overline{\text{FSX}}$ and $\overline{\text{FSR}}$ are brought high. H Serial port directly interfaces with the serial port of the TMS32020, TMS320C25, or TMS320C30 and communicates in one 16-bit word. The operation sequence is as follows (see Serial Port Timing diagrams): 1. $\overline{\text{FSX}}$ and $\overline{\text{FSR}}$ are brought low. 2. One 16-bit word is transmitted and one 16-bit word is received. 3. $\overline{\text{FSX}}$ and $\overline{\text{FSR}}$ are brought high. 4. $\overline{\text{EODX}}$ or $\overline{\text{EODR}}$ emit low-going pulses. Since the transmit and receive sections of the AIC are now synchronous, the AIC serial port with additional NOR and AND gates will interface to two SN74299 serial-to-parallel shift registers. Interfacing the AIC to the SN74299 shift register allows the AIC to interface to an external FIFO RAM and facilitates parallel data bus communications between the AIC and the digital signal processor. The operation sequence is the same as the above sequence (see Serial Port Timing diagrams).

FALTAN

LAS

PÁGINAS

6 A 9



$$\text{SCF Clock Frequency} = \frac{\text{Master Clock Frequency}}{2 \times \text{Contents of Counter A}}$$

† Split-band filtering can alternatively be performed after the analog input function via software in the TMS320.
‡ These control bits are described in the AIC DX data word format section.

NOTE A: Frequency 1 (20.736 MHz) is used to show how 153.6 kHz (for commercially available modem split-band filter clock), popular speech and modem sampling signal frequencies, and an internal 288-kHz switched-capacitor filter clock can be derived synchronously and as submultiples of the crystal oscillator frequency. Since these derived frequencies are synchronous submultiples of the crystal frequency, aliasing does not occur as the sampled analog signal passes between the analog converter and switched-capacitor filter stages. Frequency 2 (41.472 MHz) is used to show that the AIC can work with high-frequency signals, which are used by high-speed digital signal processors.

Figure 1. Internal Timing Configuration

Bibliografía

1) Texas Instruments. *User's Guide* , SPRU056
2) Texas Instruments., *User's Guide Starter Kit* ., SPRU101A
3) Texas Instruments.,*TMS320C5x General-Purpose Aplications User's Guide.*,SPRU164
4) Texas Instruments., *Initialing the TLC 320C40 AIC* on TMS320C5x DSK
5) Texas Instruments., *TLC32040 Manual del usuario*
6) Texas Instruments., *Implementation of FIR / IIR with TMS32010 / TMS32020*,SPRA003 1997
7) Texas Instruments ., *The TMS320XX Sum- of- Product methodology.*, SPRA068 1996
8) Escobar Salguero Larry., *Manual de Procesamiento digital de señales.* Facultad de Ingeniería , UNAM Agosto del 2000
9) Escobar Salguedo Larry., Arquitecturas de DSPs, familia TMS320 t TMS320C50., Facultad de Ingeniería agosto 2001
10) Ifeachor Emmanuel C. Jervis Barrie., *Digital Signal Processing a practical aproach* ., Addison-wesley 1998
11) Mahamed E. S., Real time digital processing aplications with motorolás DSP56000 family, Prentice Hall U. S. A 1990
12) Pšenička Bohumil., *Procesamiento digital de señales*, Facultad de Ingeniería UNAM, Facultad de ingeniería 1998
13) Pšenička y Escobar., *Procesamiento digital de señales segunda parte microcontroladores y realización de los filtros con TMS320CXX* ., Facultad de Ingeniería UNAM julio de 1998
14) Oppenhein A. V. Aplications of digital processing ., Prentice Hall 1978
15) Proakis John Manolakys Dimitris G., *Tratamiento digital de señales principios algoritmos y aplicaciones*., Prentice hall,2001
16) Strum Robert D., Kirk Donald E., *First Principle of discrete systems and digital signal processing.* Adisson Wesley 1989
17) Wanhammar Lars., *DSP Integrated Circuits*., Academic Press. Julio 2000

Paginas de internet
www.ti.com
www.dsptutor.com
www.bores.com
www.techonline.com