MODULO PARA ANÁLISIS, VISUALIZACIÓN Y CARACTERIZACIÓN DE SEÑALES Y SISTEMAS APOYADO DEL DSP TMS320VC5402 TEXAS INSTRUMENTS Y MATLAB.

RAFAEL LEONARDO CORZO TORRES

UNIVERSIDAD PONTIFICIA BOLIVARIANA

FACULTAD DE INGENIERÍA ELECTRÓNICA

BUCARAMANGA

2008

MODULO PARA ANÁLISIS, VISUALIZACIÓN Y CARACTERIZACIÓN DE SEÑALES Y SISTEMAS APOYADO DEL DSP TMS320VC5402 TEXAS INSTRUMENTS Y MATLAB.

RAFAEL LEONARDO CORZO TORRES.

Director

JESUS ANTONIO VEGA URIBE.

Trabajo de grado para optar al titulo de

Ingeniero electrónico

UNIVERSIDAD PONTIFICIA BOLIVARIANA

FACULTAD DE INGENIERIA ELECTRONICA

BUCARAMANGA

2008

DEDICATORIA

Este proyecto está dedicado con mucho cariño a mi madre, mi abuela JUANITA (Q.E.P.D.) y mis tías JULIA, CECILIA e ISABEL. Quienes con su apoyo permitieron que se lleve a cabo este proyecto.

A Dios por que me ha Cobijado toda la vida y agradezco la familia que me ha dado, quienes han realizado muchos sacrificios para que sea una persona útil a la sociedad, agradezco su apoyo, protección y entrega.

AGRADECIMIENTOS

El autor presenta agradecimiento a:

Jesús Antonio Vega Uribe, ingeniero eléctrico y director del proyecto de grado, por sus valiosos aportes y orientaciones. Y a todas aquellas personas que de una u otra forma aportaron para culminar el presente proyecto de grado.

CONTENIDO

LISTA DE ANEXOS

LISTA DE TABLAS

Pág.

LISTA DE FIGURAS

Pág.

GLOSARIO

Algoritmo: Fórmula o conjunto de pasos que sirven para resolver un problema particular. Para ser un algoritmo, el conjunto de reglas que lo compongan, debe evitar ser enlazado y tener un punto final claro.

Analizador de espectro: Instrumento o dispositivo, capaz de representar señales en el dominio de la frecuencia, la representación suele ser en forma discreta con bandas representando cada una un valor de amplitud para una frecuencia o rango de frecuencias componentes de la señal de entrada que está siendo analizada.

Analógico(a) - La principal característica de las representaciones analógicas es que estas son continuas, es decir, que entre dos puntos que representan valores existe una infinidad de valores intermedios.

Ancho de Banda - La cantidad de datos que pueden ser transmitidos en un intervalo fijo de tiempo. Para dispositivos digitales, el ancho de banda es usualmente expresado en bits por segundo (bps) o bytes por segundo. Para dispositivos analógicos, el ancho de banda se expresa en ciclos por segundo, o Hertz (Hz.

Aplicación O programa de aplicación. Un programa o grupo de programas diseñados para usuarios finales. Las aplicaciones se encuentran en la cima de los sistemas de software a causa de que son incapaces de correr sin el sistema operativo o las utilerías del sistema.

Apuntador: En programación, un apuntador es un tipo especial de variable que contiene una dirección de memoria (esto es, apunta a una localidad de memoria.

Archivo: Una colección de datos o i ıación que tiene un nombre, llamado nombre de archivo. Diferentes tipos de archivo almacenan diferente tipo de información.

Archivo.OUT Archivo ejecutable de salida para los DSP de Texas Instruments. Se obtiene a la salida del proceso de encadenado y es código que corre de los DSP TMS320.

Archivo Binario Archivo que contiene datos o instrucciones en formato binario.

Archivo CMD Archivo de comandos de encadenamiento. Identifica los nombres de los archivos de entrada y salida del proceso de encadenado, los medios disponibles en el TMS320, y donde han de localizarse las secciones dentro de cada módulo. El encadenador concatena cada sección de los archivos de entrada,

reserva memoria a cada sección con su longitud y localización tal como se especifica en los comandos del archivo de comandos de encadenamiento.

Archivo Objeto Archivo que contiene código objeto.

Arquitectura Harvard Caracterizada por el hecho de que los buses de programas y de datos están físicamente separados.

Bandera: Una marca de software que señaliza una condición o estado particular. Una bandera es como un interruptor que puede estar apagado o encendido.

Bit: Unidad mínima de información en sistemas computacionales, sólo puede tomar el valor de 0 o 1.

Bit Más Significativo (MSB) - En un conjunto conformado por más de un bit, el bit más significativo es el que tiene el valor más alto en el sistema binario.

Bit Menos Significativo (LSB) - En un conjunto conformado por mas de un bit, el bit menos significativo es aquel que conlleva el valor más bajo en el sistema binario. Éste sería siempre el bit que representa los números decimales 0 y 1.

Bloque: Conjunto de datos organizados de manera subsiguiente en el orden de direcciones.

Buffer: Es un área de almacenamiento temporal, usualmente en RAM. El propósito de la mayoría de los buffer es el actuar como un área de retención, permitiendo que la unidad de procesamiento pueda manipular los datos antes de transferirlos a un dispositivo.

También se utiliza para definir a un dispositivo de hardware que refuerza los niveles de las señales que pasan a través de él sin hacerles ninguna otra modificación. Conocidos también como drivers.

Bus: Es una colección de alambres a través de los cuales se transmiten datos de una parte de una computadora a otra.

Byte: Se refiere a un tamaño específico de datos, siendo un byte equivalente a un conjunto de ocho bits. Un byte es la unidad de datos básica.

C2X: Se refiere a los procesadores de segunda generación de la familia de procesadores digitales de señales TMS320CXX de Texas Instruments o a alguna de las características especificas de esa familia.

C5X: Se refiere a los procesadores de quinta generación de la familia de procesadores digitales de señales TMS320CXX de Texas Instruments o a alguna de sus características especificas.

C6X: Se refiere a los procesadores de sexta generación de la familia de procesadores digitales de señales TMS320CXX de Texas Instruments.

Chip ó Pastilla- Una pequeña pieza de material semiconductor (usualmente Silicio) sobre la cual ha sido interconstruido un circuito integrado. Frecuentemente referido al paquete completo del circuito integrado incluyendo el encapsulado y terminales.

Ciclo Máquina - Conjunto de pasos ocupados por una unidad de procesamiento desde tomar una instrucción hasta tomar la siguiente instrucción. Un ciclo de máquina común consiste de los pasos: Búsqueda, Decodificación, Ejecución y Almacenamiento.

Código Fuente - Instrucciones de programa en su forma original. Inicialmente, un programador escribe un programa en un lenguaje de programación en particular. Esta forma del programa se llama programa fuente, o de manera más general, código fuente.

Código Máquina - Lenguaje Máquina. El lenguaje de programación de más bajo nivel (excepto por las computadoras que utilizan microcódigo programable). Los lenguajes Máquina, son los únicos lenguajes que son entendidos por las computadoras. Los programadores utilizan programación ya sea en lenguajes de alto nivel o lenguaje ensamblador. Cada CPU tiene su propio y único lenguaje

máquina. Por ello los programas deben ser reescritos o recompilados, para correr en diferentes tipos de computadoras.

Código Objeto - El código producido por un compilador. El compilador produce una forma de código intermedia entre el código fuente y el código máquina llamado código objeto. El código objeto es frecuentemente el mismo o similar al lenguaje máquina de una computadora. El último paso al producir código ejecutable, es el convertir código objeto en lenguaje máquina, si aun no se encuentra en esta forma.

Compilador Un programa que transfiere de código fuente a código objeto.

Computadora Anfitrión - Es aquella que recibe elementos secundarios tales como tarjetas de expansión o dispositivos externos, que permiten conformar un sistema de cómputo especializado.

Convolución Aplicar una función a un arreglo de elementos o matriz de algún tipo. También puede definirse como la suma de muestras vecinas multiplicadas por un factor de peso.

CPU: Abreviación de Unidad de Procesamiento Central. Algunas veces referido simplemente como el procesador o el procesador central, el CPU es donde la

mayoría de los cálculos toman lugar. En términos de poder de cómputo, el CPU es el elemento más importante de un sistema de cómputo.

DAQ: Abreviación de Adquisición de Datos, en su acepción más frecuente se refiere a la adquisición de datos en formato digital desde señales de tipo analógico.

DFT o Transformada Discreta de Fourier - En sistemas discretos en tiempo, la Transformada Discreta de Fourier (DFT) es la equivalente de la transformada de Fourier continua en tiempo. Ya que la DFT requiere computación intensiva, tiene muy pocas aplicaciones, aún con las computadoras modernas.

Digital: Describe cualquier sistema basado en datos o eventos discontinuos. Las representaciones digitales consisten de valores medidos a intervalos discretos.

DSP TMS320C5402 - Procesador de señales digitales perteneciente a la familia TMS320CXX de Texas Instruments.

DSPHEX: Programa de utilería asociado a los procesadores digitales de señales de Texas Instruments, el cual transfiere archivos de tipo OUT a archivos hexadecimales que pueden ser programados directamente en un dispositivo programable por medio de un programador de dispositivos. El archivo hexadecimal contiene el código de máquina que será interpretado por el DSP.

Dword: Double Word, Palabra Doble. Se refiere al tamaño de los datos, siendo en este caso un Word equivalente a dos bytes, un double Word es equivalente a 4 bytes, o 32 bits.

Encadenador - Linker. También llamado editor de encadenamiento o enlazador, un encadenador es un programa que combina módulos objeto para formar un programa ejecutable. Además de combinar módulos, un encadenador también reemplaza direcciones simbólicas con direcciones reales. Por ello, en ocasiones es necesario encadenar un programa aunque éste contenga solo un módulo.

Encadenamiento - En programación este término se refiere a la ejecución de un encadenador.

Ensamblador - Un programa que transfiere programas de lenguaje ensamblador a lenguaje máquina.

FFT o Transformada Rápida de Fourier - Nombre genérico para una clase de algoritmos computacionalmente eficientes que implementan la Transformada Discreta de Fourier y son ampliamente utilizados en el campo del procesamiento digital de señales (DSP).

Filtro Digital - Un programa que acepta cierto tipo de datos como entrada, de algún modo los transforma, para después entregar los datos transformados a la salida.

Frecuencia Fundamental - La frecuencia de interés con el valor mas bajo en el análisis de frecuencia. Múltiplos enteros de esta frecuencia son llamados armónicos.

Full-Duplex- Se refiere a la transmisión de datos en dos direcciones de manera simultánea.

Hardware: De "ferretería" en inglés. En sistemas de cómputo se refiere a los objetos físicos que conforman el sistema, tales como tarjetas de circuito impreso, procesadores, monitores, teclado, etc.

Hexadecimal Se refiere al sistema numérico de base 16, el cual consiste de 16 símbolos únicos: los números del 0 al 9 y las letras A hasta la F. Los números hexadecimales tienen ya sea un prefijo 0x o un sufijo h.

INT: Abreviatura de interrupción. Usualmente se acompaña de un número el cual define un número de servicio específico para una señal de interrupción.

Interface: Medio que permite la conexión de dos entidades separadas.

Interface Gráfica - Un programa de interfase que toma ventajas de las capacidades gráficas de una computadora para hacer al programa más sencillo de utilizar.

Interrupción - Señal que informa a un programa o un dispositivo de que un evento ha ocurrido. Cuando un programa recibe una señal de interrupción éste toma una acción específica (la cual puede ser ignorar la señal). Las interrupciones pueden causar que un programa sea suspendido para dar servicio a esa interrupción.

IRQ: Abreviación de Interrupt Request Line, o Línea de solicitud de interrupción. Las IRQ son generalmente líneas de hardware en un bus de expansión sobre las cuales dispositivos pueden enviar señales de interrupción a un microprocesador.

JTAG: Siglas en inglés de Grupo de Acción de Pruebas Conjuntas (Joint Test Action Group), este es un concepto que está orientado a pruebas de bajo costo sobre componentes electrónicos y es un estándar documentado en el IEE 1149.1. Este estándar requiere el uso de circuitería de prueba especial en las entradas y salidas de componentes semiconductores seleccionados, junto con una lógica para controlar tal circuitería de prueba.

Kernel: De "núcleo" en alemán. Se refiere al módulo central de una plataforma de programa que provee servicios esenciales a otras partes de la plataforma o aplicaciones.

Kit de Desarrollo Sistema integral de características definidas que permite realizar pruebas de código de programa sobre un dispositivo físico con el fin de simplificar el desarrollo de una aplicación o probar configuraciones de un sistema en desarrollo.

Lenguaje C Un lenguaje de programación de alto nivel desarrollado por Dennis Ritchie y Brian Kernighan en los laboratorios Bell a mediados de los 70s. Aunque diseñado originalmente como lenguaje de programación de sistemas, C ha probado ser un lenguaje flexible y popular que puede ser utilizado para una gran variedad de aplicaciones. C es un lenguaje popular para los programadores de computadoras personales a causa de que es relativamente pequeño, y requiere menos memoria que otros lenguajes.

Lenguaje Ensamblador - Lenguaje de programación de bajo nivel, los lenguajes ensamblador tienen la misma estructura y conjunto de comandos que los lenguajes máquina, pero permiten al programado utilizar nombres en vez de números. El lenguaje ensamblador se utiliza principalmente, cuando la velocidad es esencial o cuando es necesario realizar una operación que no es posible en un lenguaje de alto nivel.

MAC: Operación de multiplicación acumulación.

Macro: Un símbolo, nombre o clave, que representa una lista de comandos, acciones, o golpes de tecla. Los macros son susceptibles de ser repetidos un número predeterminado de veces.

Mapa de Memoria - Distribución de localidades de memoria desde el punto de vista de una entidad de procesamiento.

Mapear: Hacer conexiones lógicas entre dos entidades.

Memoria de Datos - Memoria utilizada principalmente para el almacenamiento de datos fijos o variables.

Memoria de Programa - Memoria utilizada principalmente para el almacenamiento de Instrucciones o código de programa.

Memoria Global - Es memoria compartida por más de un procesador. Por ello el acceso a esta memoria debe ser arbitrado. Cuando se utiliza memoria global, el espacio de direcciones del procesador lividido en secciones local y global. La sección local es utilizada por el procesador para realizar sus funciones individuales, y la sección global es utilizada para comunicarse con otros procesadores.

Memoria Mapeada Técnica de coprocesamiento que consiste en compartir un área de direcciones de memoria entre entidades de procesamiento, haciéndose presente en la distribución de memoria de ambas, pero sin estar necesariamente duplicada.

Memoria RAM de Acceso Dual (DARAM)- Memoria tipo RAM capaz de ser acezada dos veces por ciclo de máquina, por ejemplo, una escritura y una lectura.

Memoria RAM de Acceso Sencillo (SARAM) - Memoria tipo RAM, la cual requiere de un ciclo de máquina completo para efectuar una operación de lectura o escritura.

Muestra: Se refiere a la adquisición de un valor de una señal continua por medio de un sistema discreto de tiempo.

Palabra: Se refiere al tamaño de datos utilizado, en este caso una palabra es equivalente a un word o a dos bytes (16 bits).

PC: (1)Abreviación del inglés computadora personal o PC de cualquier marca. La primera computadora personal pro da por IBM fue llamada PC, e incrementalmente el término PC paso a significar computadora personal de IBM o computadora personal compatible con IBM. (2) Abreviación de circuito impreso.

PCI: Acrónimo de ***P**eripheral **C**omponent **I**nterconnect,* un bus local estándar desarrollado por la corporación Intel. El PCI es un bus de 64 bits, frecuentemente implementado como bus de 32 bits. Puede operar a velocidades de reloj de 33 o 66 MHz. A 32 bits y 33 MHz se obtiene una tasa de transferencia de 133 MBps.

Procesador de Señales Digitales (DSP)- Del inglés Digital Signal Processor. Microprocesador cuya estructura está especializada en la resolución de algoritmos relacionados al procesamiento de señales convertidas a un formato digital.

Programa: O Código de Programa. Lista organizada de instrucciones, que cuando son ejecutadas, causan que la entidad de procesamiento se comporte de un modo predeterminado.

Programador de Dispositivos - Dispositivo que escribe un programa en pastillas de circuito integrado programables.

Puerto: Interface o medio de conexión por medio del cual puede conectarse un dispositivo a una entidad computacional.

Puerto Serie Un puerto o Interface que puede ser utilizado para comunicación de tipo serial, en la cual sólo un bit es transmitido a la vez.

RAM: Del ingles Random Acces Memory. Memoria de acceso aleatorio. Se conforma por un conjunto de localidades capaces de almacenar datos o programa por largos periodos de tiempo. En su acepción más usual se refiere a unidades de memoria que no pierden la información sino hasta que sé desenergiza el circuito de memoria.

Registro: Un área especial de almacenamiento de alta velocidad dentro del CPU. Cualquier dato debe ser presentado en un registro antes de ser procesado.

Reset: Reposicionamiento. Señal que obliga a un dispositivo a entrar en procedimiento de inicialización, parecido al hecho de eliminar la alimentación y reactivarla.

ROM: De las siglas en inglés Read Only Memory, Memoria de sólo lectura. Memoria en la cual los datos han sido pregrabados. Una vez que los datos han sido escritos en un dispositivo ROM, no pueden ser removidos y sólo pueden ser leídos.

Sistema de Desarrollo - Sistema cuya finalidad es simplificar el diseño y desarrollo de aplicaciones, compuesto generalmente por una base de hardware y un kernel de software, los cuales brindan herramientas para facilitar pruebas de sistemas en desarrollo.

Sistema Operativo - El programa más importante que corre en una computadora. Cualquier computadora de propósito general debe tener un sistema operativo que corra otros programas. Los sistemas operativos realizan tareas básicas, tales como reconocer entrada desde teclado, mandar salida a la pantalla de despliegue, llevar registro de archivos y directorios en el disco, y controlar dispositivos periféricos tales como manejadores de disco e impresoras. Los sistemas operativos proveen una plataforma de software sobre la cual pueden correr otros programas, llamados programas de aplicación. Los programas de aplicación deben ser escritos para correr encima de un sistema operativo en particular.

Sistemas de Tiempo Real Estricto - Son sistemas donde las computaciones deben ser completadas dentro de un límite dado de tiempo (el periodo de muestreo).

Software: Se refiere a instrucciones o datos de computadora, existe como ideas conceptos y símbolos, pero carece de sustancia, cualquier cosa que pueda ser almacenada electrónicamente es software.

Tabla de Vectores de Interrupción - Una tabla usualmente en memoria que está conformada por vectores de interrupción.

Tablas: Se refiere a datos almacenados en un formato de renglones y columnas.

Tarjeta de adquisición de datos - Subsistema cuya tarea es convertir señales de voltaje o corriente acondicionadas a un formato digital el cual puede ser interpretado por la computadora personal.

Temporizador - El temporizador o Timer es un contador descendente que puede ser utilizado para generar periódicamente interrupciones al CPU.

Tiempo Real - De ocurrencia inmediata. "Un sistema computacional de tiempo real puede ser definido como uno que controla un medio ambiente al recibir datos, procesarlos, y regresar los resultados suficientemente rápido como para afectar el medio ambiente en ese tiempo". "Perte ente al procesamiento de datos por una computadora en conjunción con algún proceso fuera de la misma, en concordancia con requerimientos de tiempo impuestos por el proceso externo." [IEEE89].

Vector de Interrupción - Apuntador a una rutina que maneja señales de interrupción.

Virtual: No real. El término virtual es popular entre científicos de la computación y es usado en una amplia variedad de situaciones. En general, distingue algo que es meramente conceptual de algo que es físicamente real. Por ejemplo, memoria virtual se refiere a un conjunto de localidades, o direcciones, donde se pueden almacenar datos. Es imaginaria en el sentido de que el área de memoria no es la

misma que la memoria física real compuesta de transistores. Lo opuesto a virtual es real, absoluto o físico.

Word: "palabra" en inglés. Se refiere al tamaño del dato en bits, en este caso un word es equivalente a dos bytes, ó 16 bits.

RESUMEN

MODULO PARA ANÁLISIS, VISUALIZACIÓN Y CARACTERIZACIÓN DE SEÑALES Y SISTEMAS APOYADO DEL DSP TMS320VC5402 TEXAS INSTRUMENTS Y MATLAB.

AUTOR: CORZO TORRES, Rafael Leonardo.

PALABRAS CLAVES: Digital, matemáticamente, DSP, Code Composer Studio, Tarjeta DSK, tendencias contemporáneas.

DESCRIPCIÓN: Las tendencias contemporáneas demandan a alumnos y docentes nuevos niveles de relación con los recursos que apoyan los procesos de enseñanza y aprendizaje, para así mejorar el proceso educativo. Se hizo el presente trabajo como un base para introducir fundamentos básicos de los sistemas de desarrollo (DSP'S) ya que es una tecnología que esta incursionando actualmente en el área de la ingeniería, teniendo como principio que los procesadores DSP son microprocesadores diseñados para realizar procesamiento de señales, para así manipular matemáticamente y representar digitalmente una señal.

Se desarrolló un manual de entrenamiento para programar el DSP TMS320VC5402 junto con el *code composer Studio*, por lo que el estudiante puede consultar y desarrollar los diferentes laboratorios para el proceso de aprendizaje en el manejo de la tarjeta DSK TMS320VC5402.

El presente documento muestra, las bases de programación y utilización del DSP, así como descripción y documentación detallada de las actividades necesarias para su realización, finalizando con las recomendaciones y conclusiones.

ABSTRACT

MODULE FOR ANALYSIS, VISUALIZATION AND CHARACTERIZATION OF SIGNS AND SYSTEMS SUPPORTED OF THE DSP TMS320VC5402 TEXAS INSTRUMENTS AND MATLAB.

AUTHOR: CORZO TORRES, Rafael Leonardo

KEY WORDS: Digital, mathematically, DSP, Code Composer Studio, DSK board, contemporary tendencies.

DESCRIPTION: The contemporary tendencies request the students and teachers new levels of relationship with the resources that support the learning and teaching process, to improve the teaching process, we did this project as a base to introduce basic foundations Of the systems of development (DSP'S), Since it is a technology that this penetrating nowadays into the area of the engineering, having as beginning that the processors DSP are microprocessors designed to realize processing sign, for manipulate mathematically and to represent a digital sign.

We developed a manual of training to programme the DSP TMS320VC5402 together with the code composer Studio, For what the student can consult and develop the different laboratories for the learning process in the managing of the board DSK TMS320VC5402.

The present document shows, the bases of programming and utilization of the DSP, and a detailed description and documentation of the necessaries activities for its making , finishing with the recommendations and conclusions.

INTRODUCCIÓN

El trabajo de grado titulado MODULO PARA ANÁLISIS, VISUALIZACIÓN Y CARACTERIZACIÓN DE SEÑALES Y SISTEMAS APOYADO DEL DSP TMS320VC5402 TEXAS INSTRUMENT Y MATLAB., está basado en la tecnología de procesadores en donde se darán unas pautas de la utilización del código y el procesamiento que se puede realizar.

Esta tecnología de circuitos electrónicos que han desarrollado ordenadores digitales pequeños y rápidos, han hecho posible la construcción de sistemas altamente sofisticados, capaces de realizar funciones y tareas de procesado digital de señales.

Se ha producido un crecimiento a nivel mundial en la teoría y aplicaciones del procesado de señales digitales. En el caso de Colombia la teoría ha existido pero no se ha explotado, en la UNIVERSIDAD PONTIFICIA BOLIVARIANA no se ha trabajado con este tipo de tecnología, por eso se quieren dar unas pautas, ya que no se han implementado conceptos y aplicaciones para el procesamiento de señales.

El conocimiento y aplicación de esas nuevas tendencias tecnológicas, se tornan igualmente importantes como objeto de estudio y conocimiento dentro del proceso de formación académica de un ingeniero electrónico, haciendo necesaria la implementación de herramientas que faciliten la adquisición de dicho conocimiento. Debido a que la Asignatura de Señales y Sistemas tiene un componente de análisis de señales en tiempo discreto, es importante contar con una herramienta como el DSP para realizar adquisición, pruebas de algoritmos y visualización.

OBJETIVOS

OBJETIVO GENERAL

- Manejar un dispositivo que permita el procesamiento de señales (DSP) TMS320VC5402, mostrando de manera práctica y eficiente los conceptos básicos y el desarrollo de aplicaciones, y de esta forma dar una pauta para utilizar el software de aplicación y el kit de desarrollo.

OBJETIVOS ESPECÍFICOS

- Desarrollar laboratorios para el aprendizaje y manejo del dispositivo DSP en la asignatura de señales y sistemas.
- Transmitir los conceptos básicos del procesamiento de señales.
- Interpretar la instalación del kit de desarrollo y la utilización del software Code Composer Estudio.

.

JUSTIFICACIÓN

Este proyecto ayudará a reforzar los conocimientos en el curso de señales y sistemas; ya que se trabajará señales análogas, y se analizará el comportamiento visto en la teoría de señales.

Se hará una integración de software a un dispositivo existente como lo es el DSP TMS320VC5402, este software (MATLAB) tendrá la gran ventaja de validar las señales procesadas en el DSP, y así poder analizar su comportamiento.

Se enriquecerá el conocimiento en los estudiantes de ingeniería electrónica, ya que con la implementación de nuevas tecnologías, los estudiantes despiertan mucho más sus talentos y se motivan a investigar más.

En este proyecto se implementará una herramienta versátil, en cuanto al avance tecnológico, y sus aplicaciones en todas las ramas de la ingeniería eléctrica y electrónica, ya que es un dispositivo abierto para la aplicación que se le quiera dar.

La culminación de este proyecto es importante para la contribución de conocimientos y el trabajo con este tipo de electrónica ya que se tendrá una guía de consulta, y de la forma más ilustrativa se explicaran ejemplos que manejen la programación de la tarjeta. Así mismo la utilización del software Code Composer Studio.

ETAPA 1

PRELIMINARES

1. MARCO TEÓRICO

1.1 INTRODUCCIÓN A LOS PROCESADORES DIGITALES DE SEÑALES

Los procesadores DSP son microprocesadores diseñados para realizar procesamiento de señales digitales, donde se dividen de la siguiente manera:

1.1.1 ¿A QUE SE LLAMA DSP? "A LA DISCIPLINA: se llama DSP por ***Digital Signal Processing*** **(Procesamiento digital de señal)**.

Comprende a los fundamentos matemáticos y algorítmicos que describen como procesar, en un ambiente de cómputo digital, información asociada a señales provenientes del mundo real.

• ***Digital*** **(Digital)***:* cuando las operaciones se hacen usando un sistema digital

• ***Signal*** **(señal)***:* sobre algo que transporta información

• ***Processing*** **(Procesamiento)***:* realizando operaciones sobre esa señal para modificarla o extraerle información

Métodos algorítmicos:

- La transformada rápida de Fourier (FFT)

- Diseño de filtros digitales
- Ecuaciones en diferencia.
- Convolución de señales discretas.

AL DISPOSITIVO: se llama DSP por ***Digital Signal Processor*** **(Procesamiento digital de señal)**.

Comprende a ciertas soluciones especializadas de hardware que facilitan la ejecución de algoritmos de DSP; incluye tanto a CPUs como a hardware a medida.

Diferencia entre un DSP y una CPU tradicional: ***Diferencias por sus aplicaciones de software:***

- Aplicaciones cíclicas, de duración acotada, donde se requiere altísima eficiencia de ejecución
- Uso de Assembler y dialectos especiales del lenguaje C para optimizar el código
- Algoritmos usuales:
  - Filtrado
  - Convolución (interacción de dos señales)
  - Transformación (alinealidades, rectificación,..)

1.1.2 *Características del hardware en un DSP*

- "Arquitecturas tipo HARVARD con mapas de datos e instrucciones separados.
- Dos o más mapas de memoria de datos que permiten leer concurrentemente operandos y coeficientes
- Manejo especializado de punteros de direcciones a través de unidades de cálculo dedicadas
- Opciones para la digitalización y captura de señales con intervalos regulares (DMA)
- Recursos internos o dispositivos periféricos especializados para la conversión A/D y D/A de señales, así como para el filtrado anti-alias y la reconstrucción.
- Elevada capacidad de procesamiento aritmético de datos en tiempo real, con elevada precisión, para evitar problemas de redondeo y truncamiento.
- Etapas Multiplicadora/Acumuladora (MAC) apta para resolver ecuaciones del tipo A = A + (B x C) en un único ciclo
- Una ALU operando en forma independiente al MAC.
- Códigos de operación para controlar al MAC, ALU en una única instrucción (varias operaciones concurrentes)"[1].

[1] ROBLES PRIETO, Paula Andrea y SANCHEZ CARDOZO, John Haumer. Tesis: Análisis espectral de una señal utilizando el DSP TMS320C6211. Universidad Manuela Beltrán. Bogota DC 2005

- **Porqué usar Procesamiento Digital de Señales?**

Tabla 1.Porque usar procesamiento digital de señales.

CARACTERÍSTICAS	PROCEDIMIENTO ANALÓGICO DE SEÑALES	PROCEDIMIENTO DIGITAL DE SEÑALES
Precisión Obtenible	1% al 10%	2^(-16)... 2^(-24)
Efectos Espureos	Temperatura, Humedad, Envejecimiento, ruido.	Redondeo o Truncamiento
Costo y Tamaño	Elevado	Bajo / medio
Confiabilidad	Medida	Elevada
Relación señal / ruido	50 a 80 dB	100 o más dB
Calibración	Manual	No necesaria o digital
Adaptabilidad	Compleja	Simple
Señales 2D o 3D	No procesable	Procesable
Actualización	Cambio de hadware	Software
Consumo de Potencia	Elevado	Bajo

Referencia: ROBLES PRIETO, Paula Andrea y SANCHEZ CARDOZO, John Haumer. Tesis: Análisis espectral de una señal utilizando el DSP TMS320C6211. Universidad Manuela Beltrán. Bogota DC 2005

1.1.3 Áreas de aplicación.

Comunicaciones

- "Filtrado y compresión de audio y video, cancelación de ruido: ecualización y tratamiento alineal para mejorar la relación señal / ruido o el uso del ancho de banda (Ej: ADPCM, MPEG2, MP3, FAX)

- Modems: métodos de modulación y demodulación digital de datos sobre un canal de ancho de banda y ruido propio dado. P.Ej: (ASK, FSK, PSK, DPSK, QAM, TCM)
- Reconocimiento de la palabra: para automatización de interfaz a usuario (Ej: sistemas IVR)".
- Señalización: envío y detección de información de control sobre un canal de voz o datos (P.Ej: DTMF, R2, CallerID)
- Cancelación de eco: para compensar ecos en sistemas de elevado tiempo de propagación (Ej: VOIP: Voice Over IP) o con elevado tono local (telefonía de manos libres)
- Encriptado: para comunicaciones seguras
- Detección y corrección de errores: agregado de datos a la información transmitida para detectar y corregir eventuales errores de recepción.
- Paquetizado/de paquetizado de mensajes: en aplicaciones como ATM y Frame Relay
- Telefonía Celular: manejo dinámico de frecuencias y potencias en estaciones base
- Multiplexores T1: para uso combinado de datos y voz
- Switches PBX: para centrales telefónicas digitales
- Síntesis digital directa: para estaciones de broadcast totalmente digitales
- Tratamiento de señales de RF: telefonía celular, modulación y demodulación digital, spread-spectrum.

Instrumentación:

- Medicina: tomografía, MNR, ecografía, scanner, electrocardiograma, electroencefalograma, diagnóstico asistido
- Visión artificial y OCR: Óptica Character Recognition
- Telemetría: monitoreo satelital de recursos, prospección petrolera/minera/submarina
- Sonar y Radar: radares de apertura sintética, arrays de antenas, detección de blancos móviles, detección doppler, navegación, oceanografía
- Instrumental: analizadores de red, de espectro, etc...

Industria:

- Control de motores: robótica, sistemas de transporte, sistemas de impresión, control de cabezales en sistemas de almacenamiento masivo de datos (discos rígidos, DVD, etc.).
- Control de procesos: controladores PID, control adaptativo
- Análisis de vibraciones: detección preventiva de fallas por análisis del espectro de vibraciones.
- Sistemas de navegación: GPS, piloto automático, sistemas de guía de mísiles, etc.
- Manejo de potencia: corrección del factor de potencia, inversores, variadores de frecuencia, fuentes de alimentación.

Consumo:

- telefonía: Caller ID, generación DTMF, detección de DTMF, Call Progress y Pulsos de Tarificación (16kHz)
- automotriz: AirBags, control de combustión, inyección y emisiones, ABS, etc.
- Electrodomésticos inteligentes: neveras, lavadoras, heladeras, lavarropas, aire acondicionado.
- Audio hogareño sem-profesional: sistemas surround.
- Equipos de música: órganos, sintetizadores
- Radio digital y televisión: Set-Top boxes
- Domótica y sistemas de seguridad.

1.2 PROCESADORES DE SEÑAL TMS320xx DE TEXAS INSTRUMENTS

1.2.1 Generalidades de las familias TMS320Cxx. La línea de DSP de TEXAS INSTRUMENTS, (TI) englobada bajo el código TMS320, consiste en múltiples familias de DSPs, de punto fijo o flotante, de procesadores simples o múltiples, y orientadas a distintas aplicaciones, tanto en el área de comunicaciones como de control. Desde 1982, se introdujo el TMS32010, han aparecido muchas generaciones, de las cuales algunas son hoy obsoletas y otras siguen vigentes"[2]:

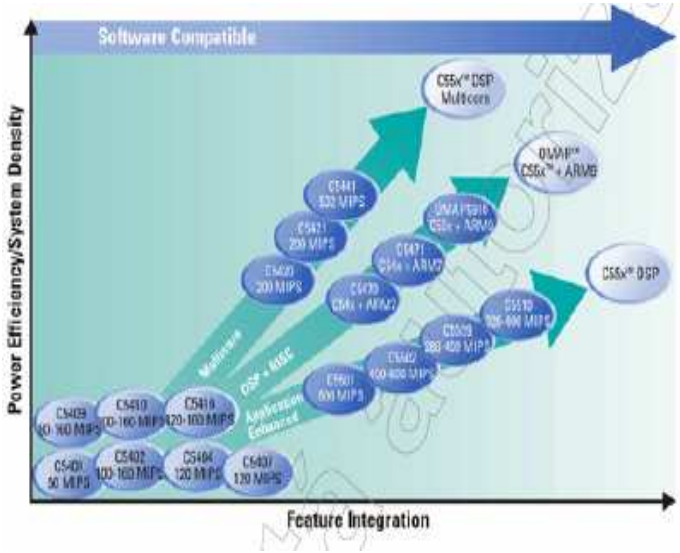
[2] ROBLES PRIETO, Paula Andrea y SANCHEZ CARDOZO, John Haumer. Tesis: Análisis espectral de una señal utilizando el DSP TMS320C6211. Universidad Manuela Beltrán. Bogota DC 2005

- De punto fijo: C1x, C2x, C24x, C5x, C54x, C55x, C62x, y C64x
- De punto flotante: C3x, C4x, C67x
- Multiprocesadores: C8x
- Combinaciones DSP+CPU: OMAP

A la par que la creación de nuevas soluciones de hardware, TI ha creado toda una tecnología de desarrollo de software bajo el nombre DSPTM que incluye a:

- Un ambiente integrado de desarrollo, llamado Code Composer StudioTM.
- Un kernel de tiempo real, llamado DSP/BiosTM.

Figura 1. Mapa de ruta familia tms320c5000



Fuente:

http://focus.ti.com/dsp/docs/dspsupporttechdocs.tsp?sectionId=3&tabId=1620&techDoc=6&familyId=324&documentCategoryId=6&Input3=Go

1.2.2Plataformas De Desarrollo Para Familia TMS320C54X .

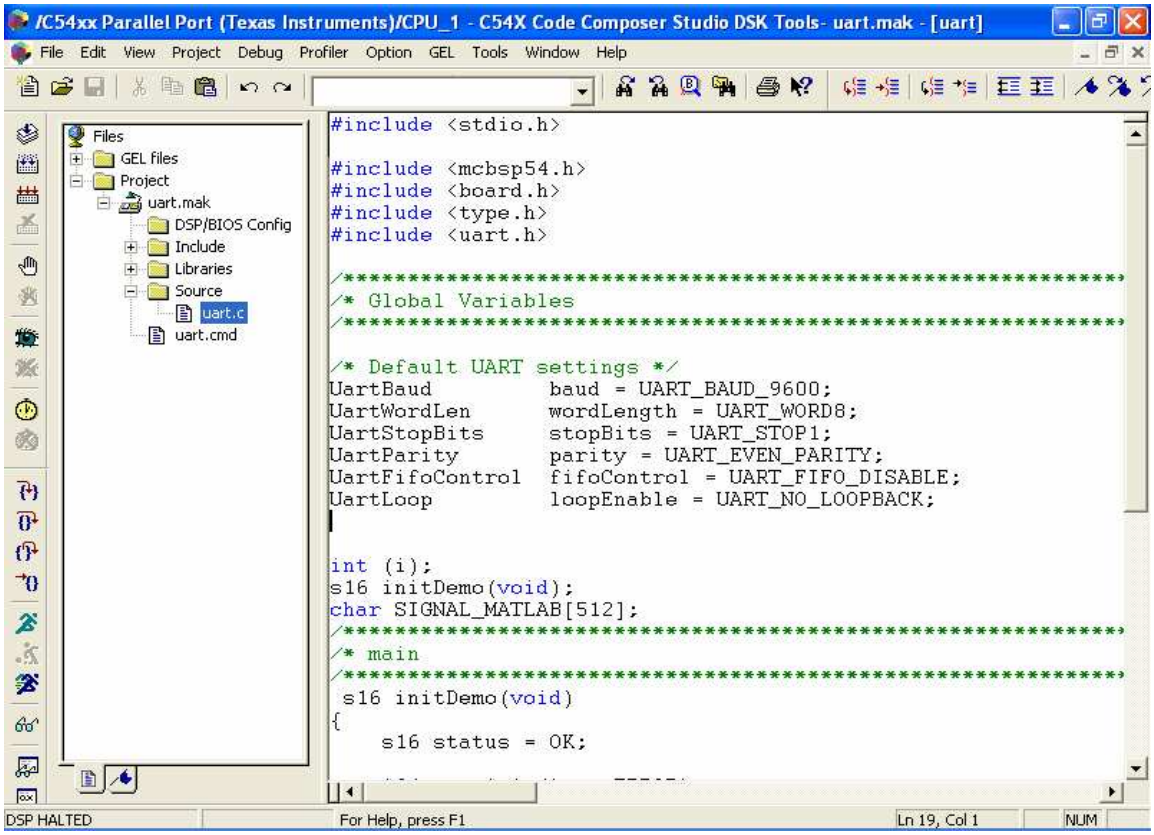
- DSK 5402: Este es un Starter kit orientado a aplicaciones de comunicaciones digitales.
- El DSK 5402 incluye
- TMS320VC5402, DSP 100MIPS, 16 bits punto fijo, 16K ram interna, 4K rom interna, 64K/1M ram/programa externa, 2 McBSP, 6 canales DMA, HPI/8 bits, ALU 40 bits, 2 timer 16 bits Emulación JTAG vía puerto paralelo hasta 32KW
- Entrada y salida de audio vía jack 3.5mm, asociado a AIC TLC320AD50 (DAC + DAC Sigma-Delta 16 bits, 22Ksps)
- Entrada / salida por conector RJ11, asociado al TLC320AD50 para aplicaciones de telefonía
- Conectores de expansión de buses de datos, I/O, programa y doughter boards
- Ambiente de desarrollo DSK Code Composer Studio IDE específico para esta plataformas que incluye editor, debugger, compilador C limitado a 32KW, assembler, linker, visualizador gráfico para estudio en el dominio del tiempo y la frecuencia.
- Interfase 14 pines para conectar emulador JTAG externo (no incluido)
- Emulación vía puerto paralelo conectado a controlador JTAG en laplaca.[3]

[3] CD ROOM CODE COMPOSER STUDIO TMS32054X.

Figura 2. Kit de desarrollo.

1.2.3 Software. Para poder realizar la programación del DSP es necesario contar con las herramientas apropiadas. El DSK es vendido junto al programa Code Composer Studio el cual es un ambiente de trabajo que permite escribir, compilar, simular y realizar *debug* de los códigos que se crean. En sí, Code Composer Studio es una interfaz estándar tipo windows que posee menús, barras de herramientas que ayudan a construir, revisar (*debug*) aplicaciones en tiempo real.

Figura 3. Code Composer Studio.



La figura (3) muestra una vista general del Code Composer Studio. En ella se observa a la izquierda una ventana en la que se ha declarado el proyecto en el que actualmente se está trabajando y los diferentes módulos que éste posee. Los módulos están formados por el o los archivos en C, C++ o Assembler, librerías de funciones, archivos “include” y otros más específicos a la aplicación que se esté realizando.

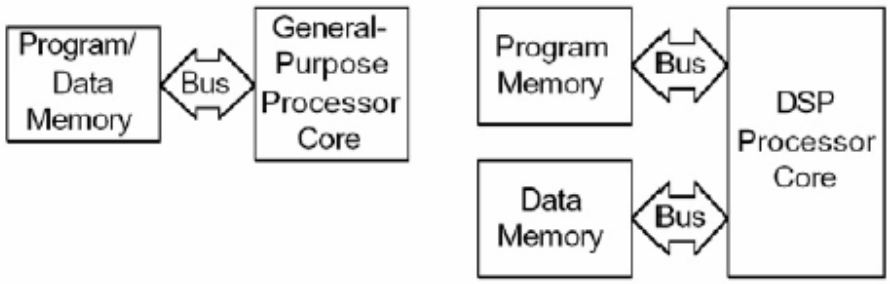
La ventana principal puede mostrar el editor de texto que se utiliza para escribir el código, además puede mostrar gráficos de variables utilizadas en el DSP, un mapa de la memoria de programa I/O datos, etc. En el desarrollo de los laboratorios se deberá adquirir el manejo apropiado de este software para así acelerar el proceso de programación.

1.2.4 Arquitectura Harvard. En la arquitectura clásica de Neumann la ALU y la unidad de control están conectadas a una sólo unidad de memoria que almacena tanto instrucciones de programa como datos. Durante la ejecución de un programa, una instrucción es leída desde la memoria y decodificada, los operandos necesarios son obtenidos (fetched) desde la memoria, y finalmente, la instrucción es ejecutada. La principal desventaja es que la memoria se transforma en el cuello de botella de esa arquitectura.

La instrucción que con más frecuencia realiza un DSP estándar es la multiplicación y acumulación. Ésta debe ser realizada con eficiencia, y para ello debería ser completada en un ciclo de instrucción. Esto implica que dos valores deben ser leídos desde memoria y (dependiendo de la organización) un valor debe ser escrito, o dos o más registros de direcciones deben ser actualizados, en ese ciclo. Por lo tanto, una longitud grande en la memoria es tan importante como la operación de multiplicación–acumulación.

Varios buses y memorias incluidas en el chip son utilizadas de forma que lecturas y escrituras a diferentes unidades de memoria pueden ser hechas a la vez. Dos memorias son utilizadas en la arquitectura Harvard clásica. Una de ellas es utilizada exclusivamente para datos, mientras que la otra es utilizada para instrucciones. Esta arquitectura alcanza un alto grado de concurrencia (lecturas y escrituras simultáneas). Los DSP's actuales usan varios buses y unidades de ejecución para alcanzar niveles incluso más altos de concurrencia. Chips con múltiples DSP y procesadores RISC existen hoy en día.

Figura 4. Arquitectura Harvard



Referencia: ROBLES PRIETO, Paula Andrea y SANCHEZ CARDOZO, John Haumer. Tesis: Análisis espectral de una señal utilizando el DSP TMS320C6211. Universidad Manuela Beltrán. Bogota DC 2005

1.3 CODE COMPOSER STUDIO

1.3.1 Características del code composer Studio. Utilizando el Code Composer Studio: Todas las ventanas (excepto la ventana Edit) y todos los

dispositivos son acoplados con el Code Composer Studio. Esto significa que usted puede mover o alinear una ventana. También se puede mover una ventana fuera de la ventana principal, haciendo Click derecho en la ventana y seleccionando Allow Docking desde el menú de contexto.

El Menú de Contexto: En el menú de contexto proviene una lista de opciones y/o comandos que pueden ser aplicados a una ventana particular. Por ejemplo se puede manipular proyectos (*add/remove source/GEL files*, *Set build options*, etc.), haciendo Click derecho en el archivo *proyect* en *Proyect View* y seleccionar la acción apropiada.

Utilizando la Ventana (Dis – Assembly): Cuando se carga un programa en la tarjeta o simulador DSP, el Code Composer Studio se ejecuta automáticamente y abre una ventana *Dis_assembly*. Esta ventana muestra las instrucciones y símbolos en lenguaje assembler necesarios para la ejecución. Para cada instrucción en lenguaje assembler, la ventana *Dis_Assembly* representa la instrucción en Assembler, la dirección la cual la instrucción es localizada y el correspondiente código (código maquina que representa la instrucción). Al producir el listado en Disassembly, el código ejecutado se lee desde la tarjeta board y adiciona información simbólica comenzando en la localización indicada por el contador de programa (PC) activado. La línea contiene la corriente PC delineada en amarillo. Puede ver su programa paso a paso usando el comando

stepping, el PC avanza a la apropiada instrucción. Para visualizar su programa escrito en lenguaje C, usted puede seleccionar el código assembler.

Usando la Ventana Memoria: El Code Composer Studio ejecuta siempre el contenido de memoria en una localización específica: para Ver el contenido de memoria: Seleccione *View -> Memoria,* desde el menú. La ventana *Memory Options* aparece, esta ventana especifica varias características de la memoria. Para modificar alguna característica de la memoria, hacer Click derecho en la ventana *Memory* y seleccione *Properties* y para editar el contenido de una localización de memoria, hacer doble clic en la dirección apropiada o seleccionar *Edit -> Memory -> Edit.*

Registros de la CPU: La CPU y los registros periféricos del procesador pueden ser visualizados usando la ventana *Register* y su contenido lo vemos en la ventana *Edit Register.*

Cargando un Archivo (COFF): "Para cargar un archivo seleccionamos *File -> Load*, aparece la ventana *Load Program* (cargar Programa), seleccione el archivo diseñado.COFF y teclee OK. Este comando carga tanto los datos como los símbolos de información desde el archivo COFF. Para cargar información solamente seleccione File -> *Load Symbol,* esta función es usada en un archivo ejecutado, donde no se necesita cargar el codigo object, como cuando el codigo

está en ROM. Este comando limpia la tabla de símbolos existentes antes de cargar uno nuevo pero sin modificar la memoria"[4].

1.3.2 Compilación de un programa. Por medio de los siguientes botones que aparecen en la barra de herramienta *Debug*, se puede observar paso a paso la compilación del programa:

***STEP INTO (*Paso Sencillo)*:** Seleccionando *Debug -> Stepinto* desde el menú, si usted está trabajando en lenguaje C, este comando pasa a través de una sencilla instrucción en C; de otra manera, hace el paso a través de una instrucción en assembler.

Reseteando la tarjeta procesadora. Los siguientes comandos son habilitados para resetear su tarjeta procesadora:

- ***Reset DSP:*** El comando *Debug -> Reset* inicializa todos los contenidos de los registros a su estado de encendido y detiene la ejecución de su programa. Si la tarjeta no responde a este comando y usted esta usando un aparato (Kernel), en este caso, usted debe recargar el kernel nuevamente.

[4] CD ROM. Code Composer Studio. TMS320C5000.

- ***Load Kernel:*** Si usted esta usando una configuración Kernel y no JTAG, entonces el kernel DSP es el responsable en la comunicación al host del computador. Si es modificado, el aparato driver no debe entablar comunicación con el DSP; en este caso, usted debe ejecutar el comando *Debug -> Load Kernel*.

- ***Restart:*** El comando *Debug - > Restart* restaura el PC en el punto de corriente para cargar el programa. Este comando no comienza la ejecución del programa.

- ***Go Main:*** El comando *Debug - > Go Main,* pasa un breakpoint (punto de ruptura) temporalmente a un símbolo principal en su programa y comienza la ejecución. El breakpoint es removido cuando en la ejecución se detiene.

Copiando valores de datos: Para copiar un bloque de memoria a una nueva localización, seleccionamos *Edit -> Memory -> Copiar* desde el menú. Aparece él dialogo: *Address* -> El comienzo de la dirección del bloque de memoria a copiar. *Length* -> La longitud del bloque de memoria a copiar. Luego entramos a la información de destino:

Address: La dirección en la cual el bloque de memoria será copiado y ejecutamos la copia (OK.)

Refrescando la ventana: Todas las ventanas muestran usualmente la corriente de estado de la tarjeta. Si usted conecta una ventana a un punto de prueba (Probe point), la ventana no debe contener información. Para actualizar una ventana seleccionamos *Window -> Refresh* desde el menú. Esta ventana actualiza los datos de la tarjeta en una ventana activa.

1.3.3 Conociendo la barra de herramientas. "Cuando el Code Composer Studio es instalado automáticamente se representa una barra de herramientas estándar. A continuación se observan algunos botones de la barra de herramienta"[5]:

 **New**. Crear un Nuevo archivo.

 **Open.** Abrir un archivo existente.

 **Save**. Guardar un archivo en la ventana activada.

 **Cut**. Cortar texto seleccionado al portapapeles.

 **Copy**. Copiar texto seleccionado al portapapeles.

 **Paste**. Pegar texto en la posición del cursor desde el portapapeles.

 **Undo**. Deshacer la última acción realizada.

 **Redo**. Retroceder a la última escritura.

 **Find Next**. Encontrar información de la palabra a buscar.

[5] C54X Code Composer Studio DSK Tools

 **Find Previous**. Encontrar información de la palabra anterior

 **Search Word**. Seleccione la palabra para hacer la búsqueda o seleccione toda la línea de oración para buscar el texto. Haciendo Click en este botón mueve la ventana al siguiente renglón de búsqueda.

 **Find in Files**. Buscar múltiples archivos para un texto específico.

 **Print**. Imprimir el archivo fuente activado.

 **Help**. Hacemos Click en este botón para ver información del Composer Studio

1.3.4 Breakpoints y probe points

 **Breakpoints.**

El Breakpoints (punto de ruptura) detiene la ejecución de un programa. Cuando el programa es detenido, usted puede examinar el estado de su programa, ó modificar variables. En *Debug -> Breakpoints* desde el menú. Se habilita o deshabilita el *Breakpoint.*

Notas de Diseño *(Code Composer Studio basado en Kernel):* Los siguientes enunciados son sugerencias de procesos pipeline:

- No seleccionar un *Breakpoint* en alguna instrucción ejecutada como parte de un delay
- No seleccionar el *breakpoint* en una o dos instrucciones antes de terminar un bloque de instrucción repetido.

Adicionando y borrando un breakpoint:

- ***Adicionando un breakpoint desde la ventana de dialogo:*** Para adicionar un *breakpoint* usando la ventana de dialogo, seleccionamos desde el menú *Debug -> Breakpoint*. Aquí aparecen en la ventana *Break/Probe/Profile Point.* El *breakpoint* es seleccionado y especificamos el tipo de campo. Seleccionando también *Break at Location* para la localización del punto de ruptura que desee. También puede seleccionar la localización del *breakpoint* en su código en C, esto es conveniente cuando usted no sabe en donde termina la instrucción en C en el ejecutable. El formato para entrar en la localización en su código en C, es: *FileName* línea *LineNumber.* Presione el botón *Add* para crear un nuevo *Breakpoint* ya habilitado.

- ***Adicionando un Breakpoint desde la barra de herramientas****:* Usé también el botón *Breakpoint* en la barra de herramientas, es un camino fácil para seleccionar y limpiar *breakpoint* en alguna localización en el programa colocando el cursor en la línea donde usted desee seleccionar el *Breakpoint*, usted puede

seleccionarlo también desde *Edit* en la ventana *Dis-Assembly* ó desde el código en C. La línea seleccionada queda de color azul.

- ***Removiendo un Breakpoint:*** Para remover el *breakpoint* coloque el cursor en la línea que contiene el *breakpoint*.

- ***Cambiando y borrando un Breakpoint existente:*** Desde el menú *Debug -> Breakpoint* en la ventana de *breakpoint*, localización y campo de Expresión son actualizados. Presione *Replace* para cambiar un *breakpoint* seleccionado. Para borrar un *breakpoint* existente, selecccionamos el *breakpoint* y en la ventana presionamos el botón *delete* para borrar un breakpoint.

Probe Point: El *Probe Point* (punto de prueba), permite que se haga una actualización de una ventana en particular ó lea y escriba muestras desde un archivo en un punto específico en su algoritmo. Estas conexiones prueban una señal a un punto en su algoritmo. Cuando el punto de prueba es seleccionado, se puede habilitar o deshabilitar como desee el *breakpoints.* Cuando una ventana es creada, por defecto, actualiza todos los *Breakpoint,* se puede cambiar esto, entonces.

Adicionando y borrando un probe point: Se crea un *Probe Point* colocando el cursor en la línea en el archivo fuente o en la ventana *Dis- Assembly* y haciendo click en la barra de herramientas en *Probe Point*. Seleccionando *Debug -> Probe Point* desde el menú permite que usted seleccione un Punto de Prueba complejo.

Borrando un Probe Point existente: Seleccione *Debug -> Probe Point*, aparecerá la ventana *Probe Point* y seleccione *Delete* (Borrar) para desactivar el *Probe Point.* También puede presionar el botón *Delete All Probepoint* para borrar todo el Punto de Prueba ó desde la barra de herramientas también *Remove All Probepoints* para remover el Probe Point.

1.3.5Formato de archivo de datos. Los comandos *File -> Data -> Load*, *File -> Data -> Store,* y *File -> File I/O* todos usan el formato COFF y archivos de datos del Code Composer Studio.

COFF: Archivo en binario que usa el formato de archivo de Objeto común (COFF) "Common Objetc File Format"[6].

Code Composer Studio proporciona una dirección del programa integrada para los proyectos. El gestor del proyecto debe guardar:

- Los archivos fuente y las librerías*.object* necesarias para construir un programa en el DSP; es decir que el gestor del proyecto identifica los archivos por su extensión de archivo. La tabla siguiente muestra las Extensiones hechas basadas en la extensión del archivo, como por ejemplo:

[6] CD ROM. Code Composer Studio. TMS320C5000, spru352a.pdf

Extensión

Descripción

. ó .c*

Archivo fuente en C.

.a* ó .s*

Archivo fuente en Assembler.

.o* ó .lib

Archivo perteneciente a las librerías ó al object.

.cmd

Archivo de comando.

Otros

Archivo Irreconocible. No le permite agregar este archivo al proyecto.

- Debe guardar el Compilador en assembler usado para ejecutar el programa.
- Los archivos dependientes incluidos en su programa.
- Se guarda la información para cada proyecto en un archivo separado. Este archivo puede ser un *makefile* (la extensión de .mak) o puede ser parte de su archivo de (la extensión de .cmd).

La nota: Estos archivos se agregan automáticamente a un proyecto examinando los archivos fuente dependiente. Sólo un archivo de orden de montador puede especificarse para un proyecto. No hay ningún límite, en el número de archivos que pueden agregarse al proyecto. Ellos se guardan, sin embargo, con los

nombres de la ruta para que el proyecto pueda moverse fácilmente a un directorio diferente. Los nombres de la ruta son determinados cada vez que el proyecto se abre.

1.4 COMUNICACIÓN SERIE

Cuando hablamos con alguien, en primer lugar llamamos su atención y entonces se transmite el mensaje, una palabra cada vez. Cuando terminamos, realizamos una pausa para indicar que hemos concluido. Lo mismo se cumple con la lectura o la escritura, se comienza una oración con la letra mayúscula, y lee o escribe una palabra cada vez, con intervalos de cierto período. Estas formas de comunicación humanas son serie, no paralelas.

Los sistemas microprogramables basados en CPU internamente están diseñados para la transferencia de datos en buses o líneas de 8 bits o múltiplos de 8. Así el bus de datos está optimizado para el tratamiento de datos en paralelo lo cual es mucho más rápido que el tratamiento serie.

Si la velocidad de transferencia de datos en paralelo es mucho más rápida, ¿por qué se utiliza la transmisión de datos serie? Algunas respuestas se dan a continuación:

a. Para realizar la comunicación de datos en paralelo se requiere gran cantidad de hilos conductores, pues debe ser establecido un hilo para cada bit de datos, además de las señales de control. Esto encarece notablemente la comunicación en función de la distancia. La comunicación serie requiere 2, 3 ó 4 hilos.

b. Una entrada salida/serie puede ser transmitida a través de pares de cobre, cable coaxial, fibra óptica, vía radio o vía satélite, lo que proporciona comunicación con equipos remotos (redes locales) o muy remotos (Internet a través de las redes telefónicas y de datos).

c. La comunicación paralelo no posee el alto grado de estandarización que ha alcanzado la comunicación serie, lo que permite la intercominicación entre equipos, por ejemplo mediante RS232, USB o FireWire.

1.4.1 RS232. Es una de las normas más populares empleadas en la comunicación serie (su inserción en el PC incremento su popularidad). Fue desarrollada en la década de los 60 para gobernar la interconexión de terminales y MODEM. Está patrocinada por la EIA (Asociación de Industrias Eléctricas).

Norma RS232: Es una de las más populares que se utilizan en la comunicación serie, y es la que se utiliza en los PC's, si bien hoy día está ampliamente superada por la transmisión serie a través de USB, de manera que está remitiendo su uso

(por ejemplo, ya no se implementa en ordenadores portátiles). Se desarrolló en la década de los 60 para gobernar la interconexión de terminales y MODEM.

La norma RS232 resuelve tres aspectos en la comunicación que se establece entre el **DTE**, (Equipo Terminal de Datos), por ejemplo un PC y el **DCE**, (Equipo para la comunicación de datos), Tarjeta DSK TMS3205402.

- **Características eléctricas de la señal:** Se establece que la longitud máxima entre el DTE y el DCE no debe ser superior a los 15 metros y la velocidad máxima de transmisión es de 20.000 bps. Los niveles lógicos no son compatibles TTL, considerando:

a. 1 lógico entre -3V y -15V
b. 0 lógico entre +3V y +15V

- **Características mecánicas de los conectores:** Se utiliza un conector 25 patillas, DB 25, o de 9 patillas, DB 9, donde el conector macho identifica al DTE y el conector hembra al DCE.

- **Descripción funcional de las señales usadas:** Las señales están básicamente divididas en dos grupos:

a. Señales primarias, que son normalmente utilizadas para la transferencias de datos

b. Señales secundarias, utilizadas para el control de la información que será transferida.

La norma RS232 está definida tanto para la transmisión síncrona como para la asíncrona, pero cuando se utiliza esta última, sólo un conjunto de terminales (de los 25), es utilizado.

1.4.2 Consideraciones de la comunicación serie. Cuando se transmite información a través de una línea serie es necesario utilizar un sistema de codificación que permita resolver los siguientes problemas:

a. **Sincronización de bits:** El receptor necesita saber donde comienza y donde termina cada bit en la señal recibida para efectuar el muestreo de la misma en el centro del intervalo de cada símbolo (bit para señales binarias).

b. **Sincronización del carácter:** La información serie se transmite por definición bit a bit, pero la misma tiene sentido en palabras o bytes.

c. **Sincronización del mensaje:** Es necesario conocer el inicio y fin de una cadena de caracteres por parte del receptor para, por ejemplo, detectar algún error en la comunicación de un mensaje.

1.4.3 Velocidad de transmisión. La velocidad de transmisión de datos es expresada en bits por segundo o baudios. El baudio es un concepto más general que bit por segundo. El primero queda definido como el número de estados de la señal por segundo, si sólo existe dos estados (que pueden ser representados por un bit, que identifica dos unidades de información) entonces baudio es equivalente a bit por segundo. Baudio y bit por segundo se diferencian cuando es necesario más de un bit para representar más de dos estados de la señal.

La velocidad de transmisión queda limitada por el ancho de banda, potencia de señal y ruido en el conductor de señal. La velocidad de transmisión queda básicamente establecida por el reloj. Su misión es examinar o muestrear continuamente la línea para detectar la presencia o ausencia de los niveles de señal ya predefinidos. El reloj sincroniza además todos los componentes internos.

1.4.4 Modos de transmisión. Existen dos modos básicos para realizar la transmisión de datos y son:

- Modo asíncrono.
- Modo síncrono.

Las transmisiones asíncronas son aquellas en que los bits que constituyen el código de un caracter se emiten con la ayuda de impulsos suplementarios que permiten mantener en sincronismo los dos extremos.

En las transmisiones síncronas los caracteres se transmiten consecutivamente, no existiendo ni bit de inicio ni bit de parada entre los caracteres, estando dividida la corriente de caracteres en bloques, enviándose una secuencia de sincronización al inicio de cada bloque.

Transmisión Asíncrona. En este modo se comunica la tarjeta DSK TMS3205402 con MATLAB. Cuando se opera en modo asíncrono no existe una línea de reloj común que establezca la duración de un bit y el caracter puede ser enviado en cualquier momento. Esto conlleva que cada dispositivo tiene su propio reloj y que previamente se ha acordado que ambos dispositivos transmitirán datos a la misma velocidad.

No obstante, en un sistema digital, un reloj es normalmente utilizado para sincronizar la transferencia de datos entre las diferentes partes del sistema. El reloj definirá el inicio y fin de cada unidad de información así como la velocidad de transmisión. Si no existe reloj común, algún modo debe ser utilizado para sincronizar el mensaje.

En realidad, la frecuencia con que el reloj muestrea la línea de comunicación es mucho mayor que la cadencia con que llegan los datos. Por ejemplo, si los datos están llegando a una cadencia de 2400 bps, el reloj examinará la línea unas 19200 veces por segundo, es decir, ocho veces la cadencia binaria. La gran rapidez con que el reloj muestrea la línea, permite al dispositivo receptor detectar una

transmisión de 1 a 0 o de 0 a 1 muy rápidamente, y mantener así la mejor sincronización entre los dispositivos emisor y receptor.

El tiempo por bit en una línea en que se transfiere la información a 2400 bps es de unos 416 microsegundos (1 seg/2400). Una frecuencia de muestreo de 2400 veces por segundo nos permitirá muestrear el principio o el final del bit. En ambos casos detectaremos el bit, sin embargo, no es extraño que la señal cambie ligeramente, y permanezca la línea con una duración un poco más larga o más corta de lo normal. Por todo ello, una frecuencia de muestreo lenta no sería capaz de detectar el cambio de estado de la señal a su debido tiempo, y esto daría lugar a que la estación terminal no recibiera los bits correctamente.[7]

- **Reglas de transmisión asíncrona:** La transmisión asíncrona que vamos a ver es la definida por la norma RS232, en la que profundizaremos más adelante y que se basa en las siguientes reglas:

a. Cuando no se envían datos por la línea, ésta se mantiene en estado alto (1).

b. Cuando se desea transmitir un carácter, se envía primero un bit de inicio que pone la línea a estado bajo (0) durante el tiempo de un bit.

[7] http://perso.wanadoo.es/pictob/comserie.htm#rs232_en_el_pc

c. Durante la transmisión, si la línea está a nivel bajo, se envía un 0 y si está a nivel alto se envía un 1.

d. A continuación se envían todos los bits del mensaje a transmitir con los intervalos que marca el reloj de transmisión. Por convenio se transmiten entre 5 y 8 bits.

e. Se envía primero el bit menos significativo, siendo el más significativo el último en enviarse.

f. A continuación del último bit del mensaje se envía el bit (o los bits) del final que hace que la línea se ponga a 1 por lo menos durante el tiempo mínimo de un bit. Estos bits pueden ser un bit de paridad para detectar errores y el bit o bits de stop, que indican el fin de la transmisión de un carácter.

Los datos codificados por esta regla, pueden ser recibidos siguiendo los pasos siguientes:

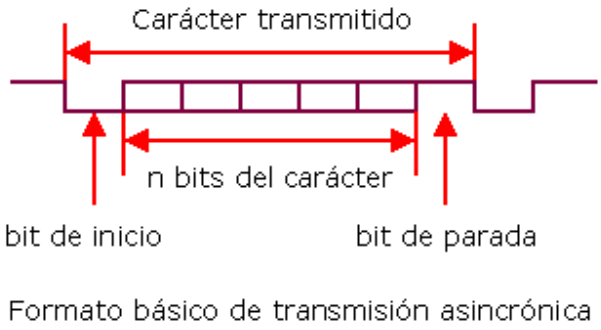
a. Esperar la transición 1 a 0 en la señal recibida.

b. Activar el reloj con una frecuencia igual a la del transmisor.

c. Muestrear la señal recibida al ritmo de ese reloj para formar el mensaje.

d. Leer un bit más de la línea y comprobar si es 1 para confirmar que no ha habido error en la sincronización.

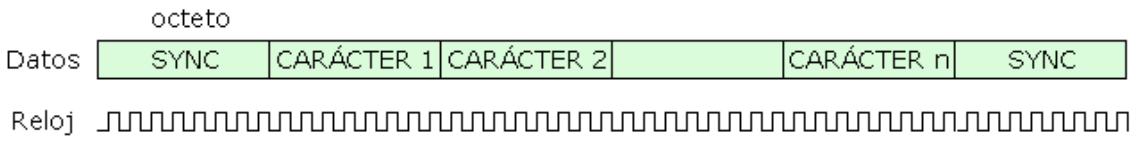
Figura 5. Comunicación asíncrona.



Formato básico de transmisión asincrónica

- **Transmisión sincronía:** Es un método más eficiente de comunicación en cuanto a velocidad de transmisión. Ello viene dado porque no existe ningún tipo de información adicional entre los caracteres a ser transmitidos.

Figura6. Transmisión sincrona.



Transmision sincrónica

Cuando se transmite de manera síncrona lo primero que se envía es un octeto de sincronismo ("sync"). El octeto de sincronismo realiza la misma función que el bit de inicio en la transmisión asíncrona, indicando al receptor que va ha ser enviado un mensaje. Este carácter, además, utiliza la señal local de reloj para determinar cuándo y con qué frecuencia será muestreada la señal, es decir, permite sincronizar los relojes de los dispositivos transmisor y receptor. La mayoría de los

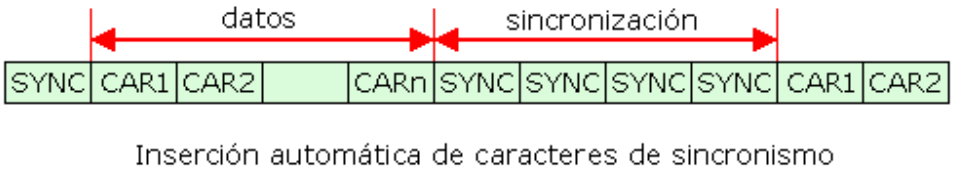
dispositivos de comunicación llevan a cabo una resincronización contra posibles desviaciones del reloj, cada uno o dos segundos, insertando para ello caracteres del tipo "sync" periódicamente dentro del mensaje.

Los caracteres de sincronismo deben diferenciarse de los datos del usuario para permitir al receptor detectar los caracteres "sync". Por ejemplo, el código ASCII utiliza el octeto 10010110.

Existen ocasiones en que son definidos dos caracteres de sincronismo, ello puede ser necesario si, por cualquier motivo el carácter "sync" original se desvirtuara, el siguiente permitirá la reinicialización del receptor. En segundo lugar, puede ocurrir que el equipo receptor necesite un tiempo adicional para adaptarse a la señal entrante.

Cuando se transmite de forma síncrona, es necesario mantener el sincronismo entre el transmisor y el receptor cuando no se envían caracteres, para ello son insertados caracteres de sincronismo de manera automática por el dispositivo que realiza la comunicación.

Figura 7. Caracteres de sincronismo.



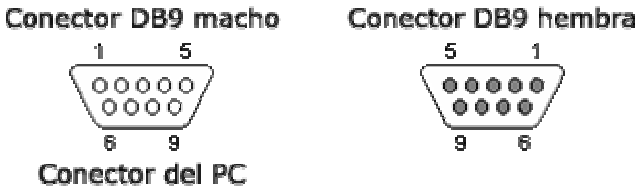
1.4.5 RS232 en el PC. El puerto serie de un ordenador trabaja en modo asincrónico. En puerto serie recibe y envía información fuera del ordenador mediante un determinado software de comunicación o un driver del puerto serie. La información se envía al puerto carácter a carácter. Cuando se ha recibido un carácter, el puerto serie envía una señal por medio de una interrupción indicando que el carácter está listo. Cuando el ordenador ve la señal, los servicios del puerto serie leen el carácter.

Existen dos tipos de interfaces RS232 puesto que la norma fue diseñada para dos tipos de equipos, el DTE (Equipo Terminal de Datos) y el DCE (Equipo de Comunicación de Datos). Existen entonces dos tipos de interfaz RS232, la DTE (conector macho) y la DCE (conector hembra):

- Interfaz DTE (macho) en el PC.
- Interfaz DCE (hembra) en los modem, ratones y otros dispositivos.

Por tanto en un PC se utilizan conectores DB9 macho, de 9 patillas, por los que se conectan los dispositivos al puerto serie. Los conectores hembra que se enchufan tienen una colocación de patillas diferente, de manera que se conectan la patilla 1 del macho con la patilla 1 del hembra, la patilla 2 con el 2, etc.

Figura 8. Conector DB9.



RS232 no admite comunicaciones a más de 15 metros y 20 Kbps (se puede utilizar mayor distancia y velocidad, pero no es el estándar). La comunicación es efectuada con 25 terminales diferentes, cada uno con su función. RS232 está definida tanto para la comunicación síncrona como asíncrona, pero cuando se utiliza esta última sólo se utiliza un conjunto de los 25 terminales.

Normalmente, las comunicaciones serie en el PC tienen los siguientes parámetros: 9.600 baudios, 1 bit de Start, 8 bits de Datos, 1 bit de Stop y sin paridad.

- **Direcciones e IRQ de los puertos serie:** El puerto serie utiliza direcciones I/O y una interrupción para llamar la atención del procesador. Además el software de control debe conocer la dirección.

La mayoría de los puertos series utilizan direcciones estándar predefinidas. Éstas están descritas normalmente en base hexadecimal. Las direcciones I/O e IRQ pueden seleccionarse en la BIOS o bajo Windows.

Las señales son:

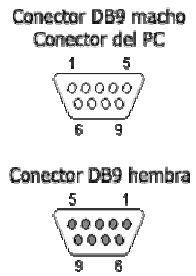
Tabla 2. Señales del PC para comunicación serial.

Puerto	Dir. I/O	IRQ
COM1	3F8-3FF	4
COM2	2F8-2FF	3
COM3	3E8-3EF	4
COM4	2E8-2EF	3

Las direcciones e IRQ usadas por los puertos serie fueron definidas al diseñar el PC, sin embargo, las del COM3 y COM4 no se han definido oficialmente, aunque están aceptadas por convenios.

Tabla 3. Terminales conector DB9.

PIN	Nombre	Dir	Descripción
1	CD	←	*Carrier Detect*, detección de portadora
2	RXD	←	*Receive Data*, recepción de datos
3	TXD	→	*Transmit Data*, transmisión de datos
4	DTR	→	*Data Terminal Ready*, terminal de datos preparado
5	GND	—	*System Ground* ó Signal Ground, tierra de señal
6	DSR	←	*Data Set Ready*, dispositivo preparado
7	RTS	→	*Request to Send*, petición de envío
8	CTS	←	*Clear to Send*, preparado para transmitir
9	RI	←	*Ring* Indicator, indicador de llamada entrante



La dirección (Dir) es DTE (PC) relativa a DCE (Dispositivo).

- DTE (PC) ⟵ DCE (Dispositivo), entrada en el DTE (PC).
- DTE (PC) ⟶ DCE (Dispositivo), salida en el DTE (PC).

Figura 9. Configuración de puerto serie.

- **Bit por segundo:** Define la velocidad máxima, en bits por segundo (bps), a la que se transmiten los datos a través del puerto. Normalmente, se establece a la velocidad máxima admitida por el equipo o dispositivo con el que se está comunicando.

- **Bits de datos:** Cambia el número de bits de datos a utilizar para cada carácter transmitido y recibido. El equipo o dispositivo con el que comunica debe tener la

misma configuración que aquí. La mayor parte de los caracteres se transmiten con siete u ocho bits de datos.

- **Paridad:** Cambia el tipo de comprobación de errores a utilizar para el puerto seleccionado. El equipo o dispositivo con el que se comunica debe tener la misma configuración que aquí. Se debe elegir una de las siguientes:

a. **Ninguna:** significa que no se agregará ningún bit de paridad a los bits de datos enviados desde este puerto. Esto deshabilitará la comprobación de errores.

b. **Par:** significa que el bit de paridad se establece a 1 si se necesita para que el número de unos (1) de los bits de datos sea par. Esto habilitará la comprobación de errores.

c. **Impar:** significa que se agrega un bit de paridad si se necesita para que el número de unos (1) de los bits de datos sea impar. Esto habilitará la comprobación de errores.

d. **Marca:** significa que se agrega un bit de paridad, pero siempre está establecido a 0.

e. **Espacio:** significa que se agrega un bit de paridad, pero siempre está establecido a 1.

- **Bit de parada:** Cambia el tiempo entre cada carácter que se transmite (cuando el tiempo se mide en bits por segundo).

- **Control de flujo:** Cambia la forma en que se controla el flujo de datos.

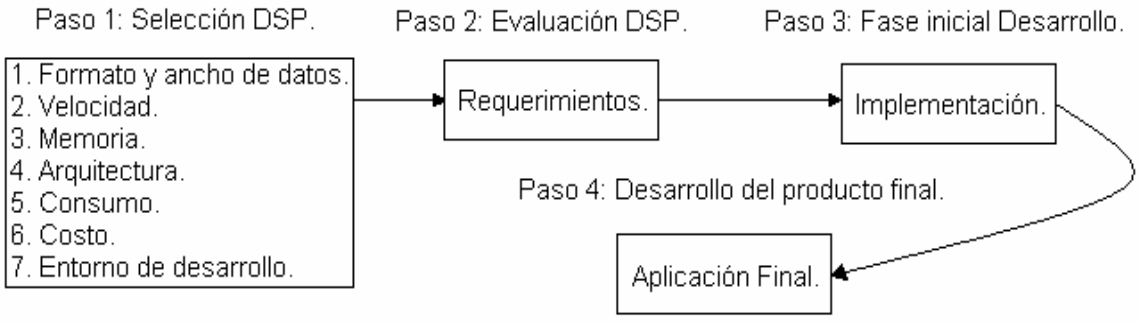
a. Ninguno

b. Xon/Xoff, llamado en ocasiones protocolo de enlace software, es el método de software estándar para controlar el flujo de datos entre dos módems.

c. Control de flujo Hardware, llamado en ocasiones protocolo de enlace hardware, es el método estándar de controlar el flujo de datos entre un equipo y un dispositivo serie.

1.5 PROCESO DE SELECCIÓN, PROCESAMIENTO Y APLICACIÓN DE UN DSP

Figura 10. Selección de un DSP.



PASO 1: Selección del DSP.

- **Formato de datos:** los DSP se clasifican según el tipo de aritmética que usen para realizar los cálculos, los **procesadores de punto fijo**, representan los números como enteros o como fracciones en un rango fijo (-1 a 1) utilizando un determinado número de bits. Requieren un hardware mas simple, por su costo unitario, consumo y el tamaño es menor lo que resulta idóneo para aplicaciones en telefonías móvil. Con los **procesadores de punto flotante** se obtiene un rango dinámico mayor entre el valor más grande y el valor más pequeño que puede ser representado. Los números se representan usando una mantisa y un exponente. Esto simplifica lo programación de algoritmos aunque a expensas de un mayor costo, consumo y tamaño.

- **Ancho de datos:** el tamaño de los buses de datos dan una medida del flujo de información entre la CPU y los periféricos. Buses de mayor ancho permiten que los flujos de datos sean más rápidos, aunque a costa de complicar la arquitectura, el tamaño del chip, el número de pines del encapsulado y el tamaño de los dispositivos de la memoria.

- **Velocidad:** determina el numero de instrucciones ejecutadas por segundo (MIPS) o millones de operaciones flotantes por segundo (MFLOPS), o millones de

operaciones de acumulación por segundo (MMACS). Sirven como patrón de comparación entre otros procesadores.

- **Organización de la memoria:** Jerarquías de memoria con memorias caché de programa y/o de datos que aceleraran la ejecución pues aprovechan de manera más eficiente la arquitectura interna del DSP.

- **Arquitectura del procesador:** Determina la estructura de ejecución de las instrucciones y el grado de paralelismo.

- **Consumo:** las tensiones de los DSPs son cada vez menores, (3.3, 2.5, o 1.8V)

- **Costo:** La mejor elección no es el procesador mas potente sino aquel que nos permite cumplir con las exigencias de la aplicación al menor costo.

- **Entorno de desarrollo:** Determina las herramientas software (ensambladores, simuladores, compiladores, sistemas operativos en tiempo real), las herramientas hardware (tarjetas de desarrollo y evaluación, emuladores).

- **PASO 2: Evaluación del DSP seleccionado.** Incluye el aprendizaje de las herramientas de desarrollo, que debe cumplir con los requerimientos de la aplicación.

- **PASO 3: Fase inicial del desarrollo.** Utiliza un sistema de desarrollo completo de la aplicación para probar las posibilidades que ofrece el sistema para implementar algunas partes del proyecto final o evaluar los algoritmos de procesamiento de diseño.

- **PASO 4: Desarrollo del producto final.** Ultima fase de diseño y desarrollo de un soporte hardware prototipo que permita, junto con las herramientas de desarrollo como software de prueba, muestras gratuitas y demás ayudas que pueda proporcionar el fabricante para implementar, y depurar la aplicación final.

1.5.1 Principales fabricantes de DSPs.

a. *Analog Device.*
b. *Hitachi.*
c. *Motorola.*
d. *NEC.*
e. *SGS – Thomson.*
f. *Texas Instruments.*
g. *Zilog.*

ETAPA 2

DESARROLLO

2. SISTEMA PROPUESTO

Procesar una señal significa realizar una operación matemática sobre ella con el fin de transformarla o de extraer información enviada.

Un estudiante de Ingeniería Electrónica debe tener una formación sólida en el área del procesamiento digital para aplicarlo en áreas tan importantes como las telecomunicaciones, sistemas de control, instrumentación y sistemas digitales.

A pesar de que el procesamiento digital es un campo dinámico y de crecimiento rápido sus fundamentos no lo son dentro de la universidad. Por ello se propone hacer énfasis en los conceptos fundamentales, para la implementación y la simulación empleando software Code Composer Studio y MATLAB manejando la tarjeta de procesamiento de señal digital (DSP) TMS320VC5402.

Un propósito importante es lograr la base para el entendimiento de procesos más complejos desarrollados en sistemas de control avanzado, y teoría de la información.

Para ello se realiza un documento con conocimientos básicos para la investigación e implementación de algoritmos basados en procesamiento digital de señales entre ellos están:

- Utilización del software Code Composer Studio.
- Configurar e instalar el software de aplicación y conectar la tarjeta TMS320VC5402.
- Comprender y programar microprocesadores DSP para la implementación algoritmos básicos.
- Desarrollo de filtros pasa bajos, pasa altos, pasa banda, rechaza banda, y un filtro no lineal Mediana Móvil.

Utilización del software Code Composer Studio: Se explica en forma resumida la utilización del software Code Composer Studio, donde se muestra la utilización de iconos y la función que realiza cada uno de ellos, para así poder construir e implementa proyectos con la tarjeta TMS 320VC5402, también se explicara las extensiones, que genera el programa cuando se crea, se compila y se carga una grafica.

Configurar e instalar el software de aplicación y conectar la tarjeta TMS320VC5402: Se explicara de una manera didáctica la tarjeta de procesamiento de señal digital TMS320VC5402, junto con el Code Composer Studio para las personas que requieran utilizar este tipo de tecnología, tengan la capacidad de configurarla y conectarla.

3. DESARROLLO DEL PROYECTO

Como primera medida para el entendimiento del desarrollo de los algoritmos a trabajar los estudiantes deben tener nociones previas del concepto de representación en frecuencia de una señal. Básicamente la Transformada de Fourier se encarga de transformar una señal del dominio del tiempo, al dominio de la frecuencia. El trabajo con la señal en frecuencia, no solo sirve como información, sino que se puede modificar, de forma que es ampliamente utilizada en filtros, procesado de la imagen y el sonido, comunicaciones (modulaciones, líneas de transmisión, etc.). El entorno de desarrollo del Code Composer Studio permite desarrollar esto mediante la visualización de la transformada de FOURIER ante una señal de entrada.

Se acopla el DSP con MATLAB, enviando una señal por el puerto serial desde MATLAB, visualizando en el DSP la señal recibida, y viendo su respectiva grafica de la FFT, así mismo se le realizara un algoritmo de convolución con otra señal discreta en el DSP.

Se realiza un filtro no lineal Mediana Móvil, los filtros no lineales MATLAB no los representa, se utiliza el DSP para hacer dicho algoritmo de representación.

3.1 DESCRIPCIÓN DEL SISTEMA A DESARROLLAR

- **Programación del DSP:** Hay que tener en cuenta una propiedad que determinará condiciones importantes de la programación de los algoritmos:

En cuanto al hardware se refiere, el espacio de memoria es limitado y si además se considera que el programa a ejecutar también se almacenará en memoria RAM entonces este espacio se verá más reducido. Esto implica que los datos a capturar y sobre los cuáles se ejecutará el análisis no pueden ser demasiados.

Para realizar la comunicación serial del DSP con el computador se realizarón las siguientes pruebas:

- La tarjeta DSK se acopló al PC mediante el cable serial DB-9.
- Se verifico la comunicación serial con el hyper Terminal del PC. (corriendo el proyecto uart.mak del DSP en CCS).
- Se desarrollo el siguiente programa en MATLAB para enviar la señal por el puerto serial:

```
s = serial('COM1');
set(s, 'BaudRate', 9600, 'StopBits', 1);
set(s, 'Terminator', 'LF', 'Parity', 'even');
```

```
set(s, 'FlowControl', 'hardware');
fopen(s);
get(s, 'Status')
fclose(s);
set(s, 'OutputBufferSize',3000);
fopen(s);
get(s, 'OutputBufferSize')
t = (0:80) .* 2 * pi / 80;
data = round((sin(t)*90)+120)
fwrite(s, data, 'char');
fclose(s)
delete(s)
clear s
plot(t,data)
```

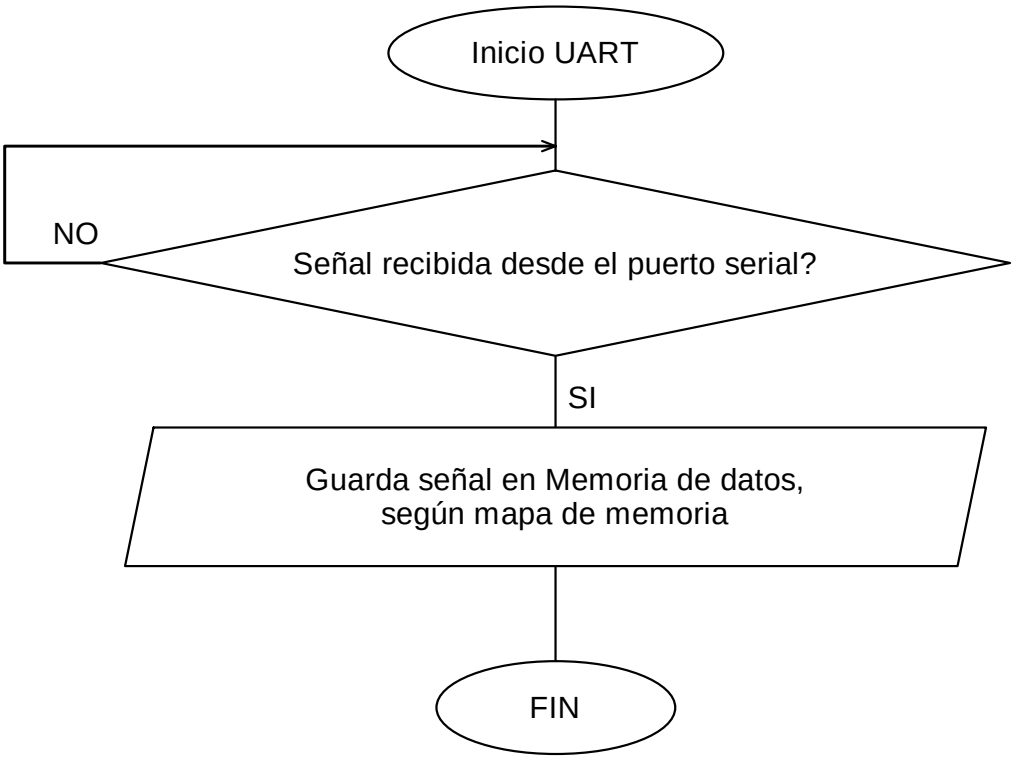
Se desarrolló un algoritmo en el DSP para recibir la señal por el puerto serial desde MATLAB:

```
#include <stdio.h>
#include <mcbsp54.h>
#include <board.h>
#include <type.h>
#include <uart.h>
s16 initDemo(void);
```

```
char  SIGNAL_MATLAB[512];
UartBaud           baud = UART_BAUD_9600;
UartWordLen      wordLength = UART_WORD8;
UartStopBits       stopBits = UART_STOP1;
UartParity          parity = UART_EVEN_PARITY;
UartFifoControl  fifoControl = UART_FIFO_DISABLE;
UartLoop            loopEnable = UART_NO_LOOPBACK;
/*****************************************************************************/
/* main                                                                      */
/*****************************************************************************/
 s16 initDemo(void)
{
       s16 status = OK;

   if(uart_init() == ERROR)
   {
      status = ERROR;
   }
   return status;
}
void main()
{
       brd_init(100);
    {
    uart_fgets(&SIGNAL_MATLAB[0],324);
     }
 }
```

Figura 11. Diagrama de flujo del proyecto UART.mak



Una vez enviada la señal por el Puerto serial se verificó, la entrada de esta señal en la EXRAM de la tarjeta DSK.

Figura 12. Señal de entrada del DSP

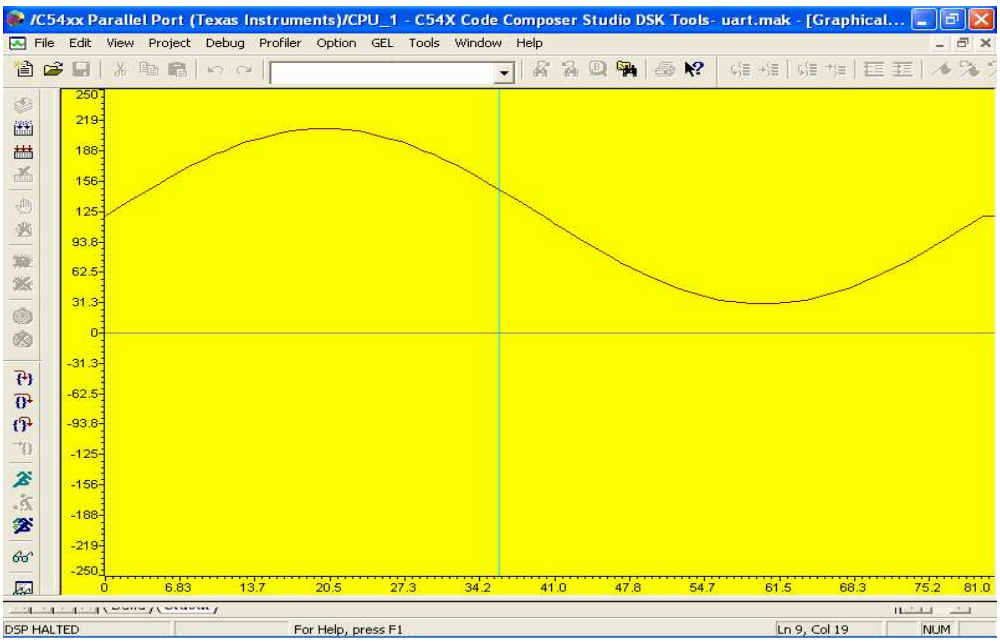
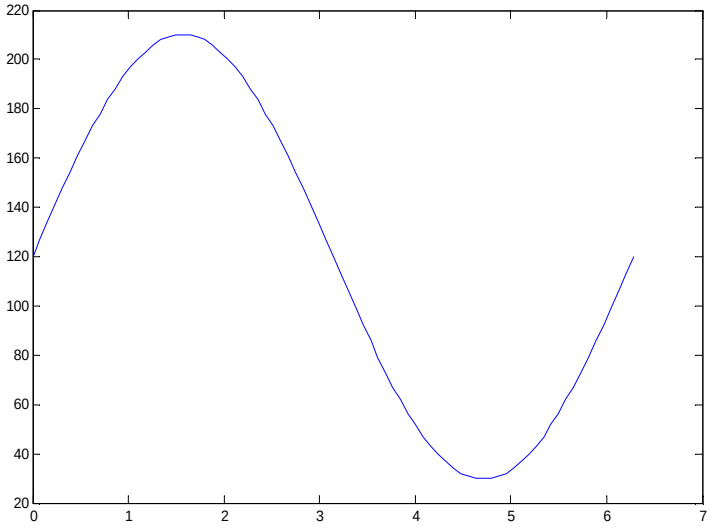


Figura 13. Señal visualizada en MATLAB



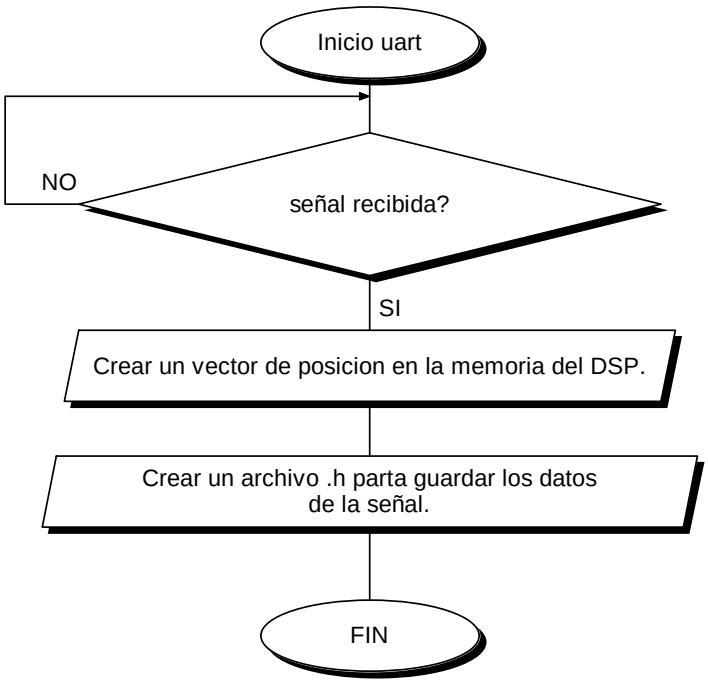
Se puede analizar la señal en el dominio de la frecuencia hacienda click derecho sobre la grafica en el DSP y ajustando valores de la FFT.

- **Programa del DSP aplicando un algoritmo matemático utilizando señal del puerto serial.**

Algoritmo:

$$r[j] = \sum_{k=0}^{nh} h[k]x[j-k] \quad 0 <= j <= nr$$

Figura 14. Diagrama de flujo proyecto serie.mak



- **Programa en el DSP:**

```
#include <stdio.h>
#include <mcbsp54.h>
#include <board.h>
#include <type.h>
#include <uart.h>
#include <dsplib.h>
#include <stdio.h>

UartBaud         baud = UART_BAUD_9600;
UartWordLen      wordLength = UART_WORD6;
UartStopBits     stopBits = UART_STOP1;
UartParity       parity = UART_EVEN_PARITY;
UartFifoControl  fifoControl = UART_FIFO_DISABLE;
UartLoop         loopEnable = UART_NO_LOOPBACK;

char  SIGNAL_MATLAB[50];
s16 initDemo(void);
int i;
int n;
int j;
int k;
FILE * pFile;
short datos [30]={0};
void delay(s16);
int a;
s16 initDemo(void)
{
```

```
        s16 status = OK;

    if(uart_init() == ERROR)
    {
        status = ERROR;
    }
    return status;
}
 void main(void)
{
        brd_init(100);
     {

        uart_fgets(&SIGNAL_MATLAB[0],21);

        for (i= 0; i<21 ; i++)

   datos[i+5] =  SIGNAL_MATLAB[i];

  pFile = fopen ("señal.h","w");

   fprintf (pFile, "//señal.h señal desde MATLAB\n");

   for (n=0 ; n<1 ; n++)
   {

    fprintf (pFile, "#define  vector  %d, ",n,datos);
   for (j=1 ; j<30 ; j++)
   {
    fprintf (pFile, "%d,",datos[j]);
```

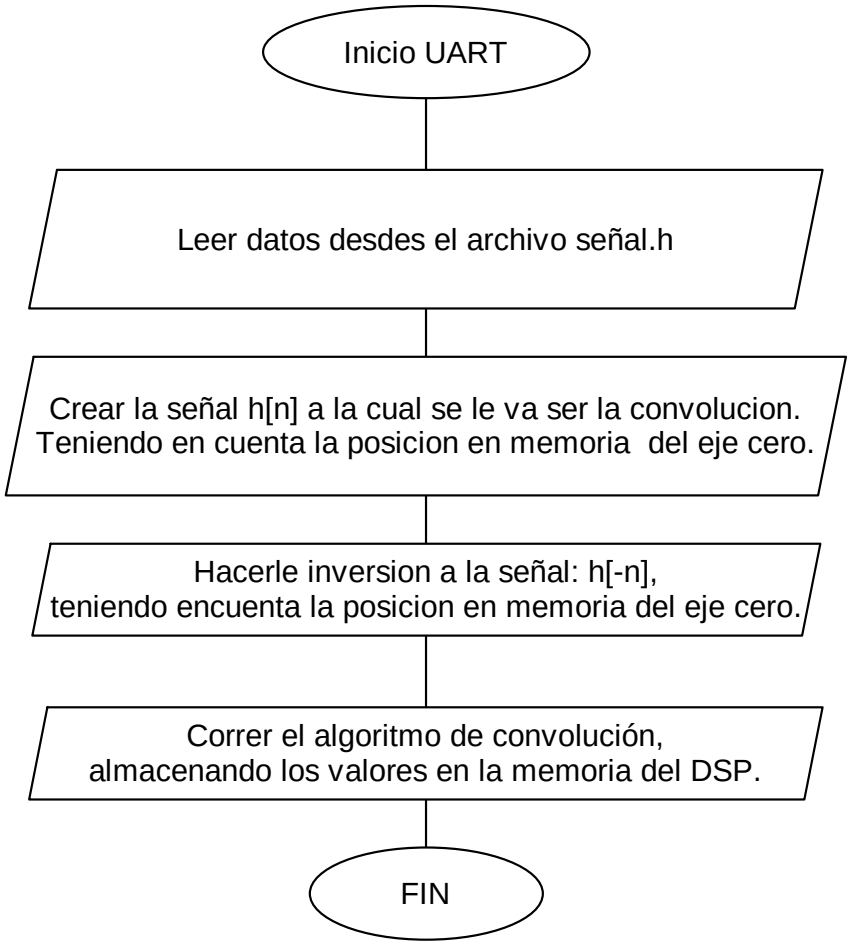
```
  }
  }
    fclose (pFile);

      }
      }
void delay(s16 period)
{
   int l, m;

   for(l=0; l<period; l++)
   {
      for(m=0; m<period>>1; m++);
   }
}
```

- Una vez creado el archivo con los valores de la señal enviada por el puerto serie, se le aplica el algoritmo respectivo.

Figura 15. Diagrama de flujo programa convol.mak.



- **Programa en C del proyecto**

```
# include <stdio.h>
# include <type.h>
# include "señal.h"
#define count 50
```

```
int i;
int dotp(short *a, short *b, int ncount);
short hn[20]={0,0,0,1,1,1,0,0,0};
short hr[50]={0};
short convolucion[20]={0};
short datos[30] = {vector};
short Mediana[50]={0};
short b[20]={0,0,0,0,0,0,0,0,0,0,0,800,0,0,0,0,0,0,0,0}; //señal de ruido..
int elementos=19;
int temp=0;
int j;
int n;

/*************************************************************************/
/* main                                                                  */
/*************************************************************************/
void main()
{
hr[0]=hn[8];    ;  h[-n]
 hr[1]=hn[7];
 hr[2]=hn[6];
 hr[3]=hn[5];
 hr[4]=hn[4];
 hr[5]=hn[3];
 hr[6]=hn[2];
  hr[7]=hn[1];
  hr[8]=hn[0];

 n=0;
```

```
convolucion[0]= dotp(datos,hr,count);
n=-1;
convolucion[1]= dotp(datos,hr,count);
n=-2;
convolucion[2]= dotp(datos,hr,count);
n=-3;
convolucion[3]= dotp(datos,hr,count);
n=-4;
convolucion[4]= dotp(datos,hr,count);
n=-5;
convolucion[5]= dotp(datos,hr,count);
n=-6;
convolucion[6]= dotp(datos,hr,count);
n=-7;
convolucion[7]= dotp(datos,hr,count);
n=-8;
convolucion[8]= dotp(datos,hr,count);
n=-9;
convolucion[9]= dotp(datos,hr,count);
n=-10;
convolucion[10]= dotp(datos,hr,count);
n=-11;
convolucion[11]= dotp(datos,hr,count);
n=-12;
convolucion[12]= dotp(datos,hr,count);
 n=-13;
convolucion[13]= dotp(datos,hr,count);
 n=-14;
convolucion[14]= dotp(datos,hr,count);
 n=-15;
```

```
 convolucion[15]= dotp(datos,hr,count);
  n=-16;
 convolucion[16]= dotp(datos,hr,count);
 n=-17;
 convolucion[17]= dotp(datos,hr,count);
  n=-18;
 convolucion[18]= dotp(datos,hr,count);
  n=-19;
 convolucion[19]= dotp(datos,hr,count);

 for(j=0;j<15;j++)
 Mediana[j]= convolucion[j]+b[j];

 for(i=0;i<=elementos-1;i++)
    {
        for(j=i+1;j<=elementos;j++)
        {
            if(Mediana[i]>Mediana[j])
            {
               temp = Mediana[i];
               Mediana[i] = Mediana[j];
               Mediana[j] = temp;
            }
        }
     }
}
int dotp(short *a, short*b, int ncount)
  {
int sum = 0;
```

```
int i;

for (i= 0; i<ncount ; i++)
sum += datos[i] * hr[i+n];
        //suma de productos

return (sum);

}
```

Una vez creado este algoritmo se puede implementar un filtro **mediana *móvil***, el cual no se puede montar en MATLAB. Ya que este filtro es un filtro no lineal.
Se le aplica el filtro a la señal de convolución simulando un ruido. Creado en otra señal.

A continuación se verán las graficas hechas en Matlab y hechas por el DSP comparando resultados.

Figura 16. Respuesta de convolución en el DSP.

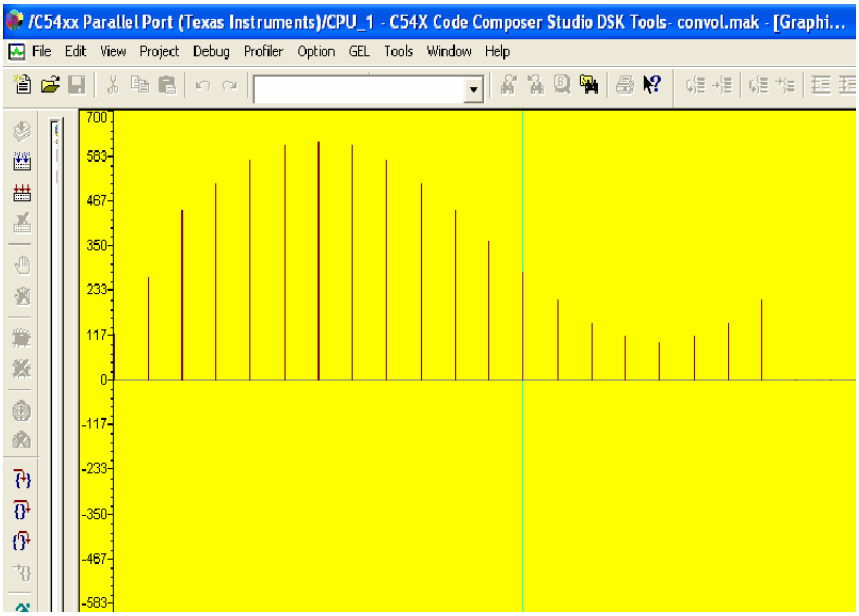


Figura 17. Señal en MATLAB de entrada.

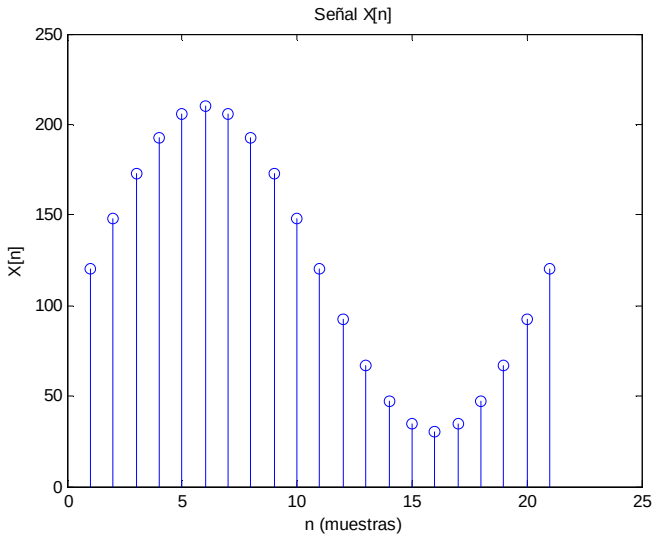


Figura 18. Señal h[n] Representada en MATLAB.

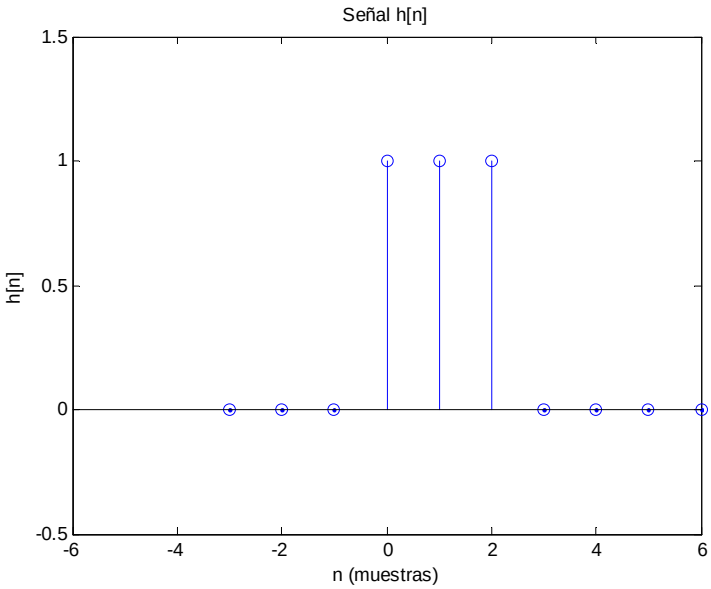


Figura 19. Señal de convolución.

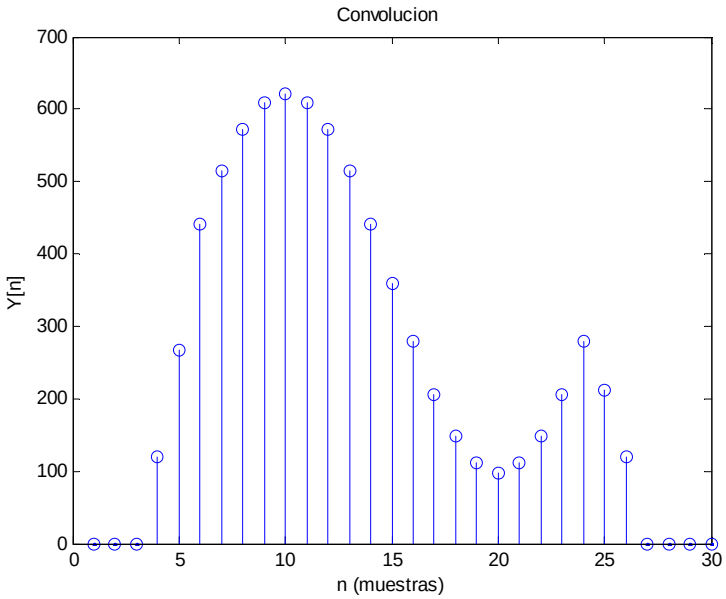
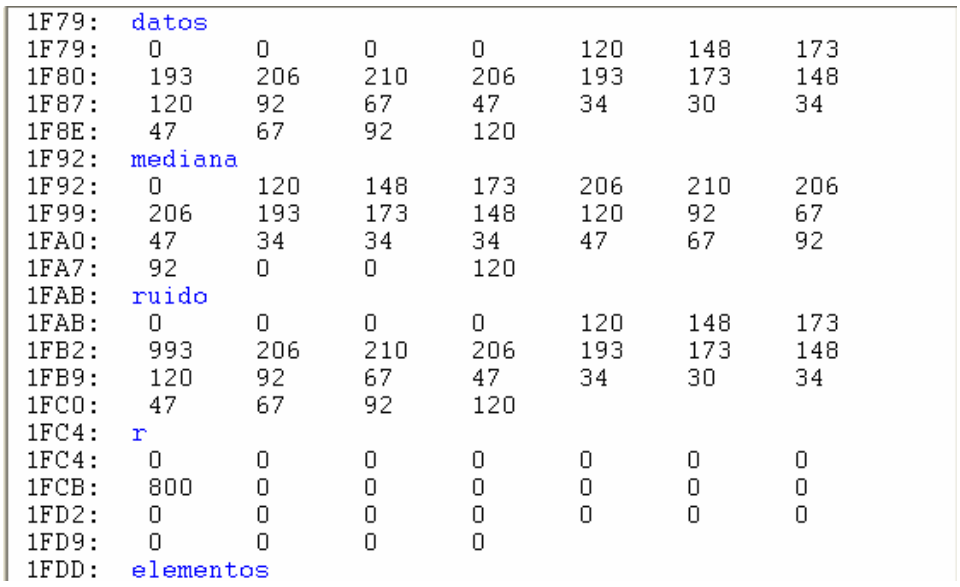


Figura 20. Señal filtro mediana Móvil en el DSP



```
1F79:  datos
1F79:   0       0       0       0       120     148     173
1F80:   193     206     210     206     193     173     148
1F87:   120     92      67      47      34      30      34
1F8E:   47      67      92      120
1F92:  mediana
1F92:   0       120     148     173     206     210     206
1F99:   206     193     173     148     120     92      67
1FA0:   47      34      34      34      47      67      92
1FA7:   92      0       0       120
1FAB:  ruido
1FAB:   0       0       0       0       120     148     173
1FB2:   993     206     210     206     193     173     148
1FB9:   120     92      67      47      34      30      34
1FC0:   47      67      92      120
1FC4:  r
1FC4:   0       0       0       0       0       0       0
1FCB:   800     0       0       0       0       0       0
1FD2:   0       0       0       0       0       0       0
1FD9:   0       0       0       0
1FDD:  elementos
```

En la memoria de datos se puede ver la señal con ruido, y en el vector mediana tenemos la misma señal sin ruido. El vector r, representa la señal ruido.

Este algoritmo es usado en procesamiento de imágenes para eliminar ruido en imágenes.

RESULTADOS

- Se consiguió explicar y desarrollar programas en la tarjeta DSP TMS 320VC5402 junto con el software de aplicación CODE COMPOSER ESTUDIO, se logro hacer laboratorios los cuales ayudan a familiarizar a los estudiantes de una manera mas didáctica en el desarrollo y aplicaciones a este tipo de tecnología, altamente configurable y adecuado a las necesidades del estudiante.

- Los laboratorios diseñados se realizan en primera medida para dejar bases teórico prácticas con el DSP el TMS320C5402.

- Se utilizo algunos periféricos del DSP, haciendo el trabajo práctico y con resultados visibles.

- Se deja una base teórica-practica en el manejo del DSP TMS320VC5402 DE TEXAS INSTRUMENTS.

- Con base en el manejo del kit de desarrollo y el software CODE COMPOSER STUDIO, el trabajo queda abierto para aplicaciones que requieran algoritmos mas complicados en aplicaciones de comunicaciones.

CONCLUSIONES

Los procesadores de señales digitales brindan muchas ventajas en cuanto a las cantidades de información procesada, a la velocidad del tratamiento y a la calidad del procesamiento sobre esta información en vista de que su arquitectura se orienta al propósito de resolución de algoritmos de tratamiento de señales digitales.

Una tarjeta de adquisición inteligente como la TMS320VC5402 forma un sistema de procesamiento muy eficiente, veloz y muy potente, que brinda una opción económica y con ventajas evidentes para la aplicación y desarrollos de software de una manera didáctica.

El desarrollo de este manual de entrenamiento para programar el DSP TMS32054X de Texas Instruments de punto fijo, favorecen a los estudiantes de ingeniería electrónica, ya que se tiene una guía en el manejo del mismo, no solo por la tecnología sino por hacer aplicaciones en DSP en la Universidad Pontificia Bolivariana.

Teniendo en cuenta la versatilidad de las nuevas tecnologías (específicamente los DSPs) y las enormes posibilidades que brinda este tipo de tecnología, casi las únicas restricciones que se tienen en su implementación son generalmente de

orden administrativo o de recursos, sin embargo la pregunta no es que enseñar con el apoyo de ellas, sino como lograr que el ambiente de aprendizaje tenga calidad pedagógica.

Para que la UPB pueda decir que esta realmente integrando las tecnologías en su escenario educativo, se requiere que los profesores en el área de señales conozcan, como utilizar dichas tecnologías y que se logre implementar estos dispositivos como un área de PROCESAMIENTO DIGITAL DE SEÑALES en la facultad de ingeniería electrónica.

RECOMENDACIONES

- Se recomienda para el uso de la herramienta, poseer un manejo teórico del procesamiento de señales digitales para así tener unas bases y poder implementar un buen diseño a la hora de realizar una aplicación.

- Realizar un diseño y tener bien definida la aplicación a desarrollar antes de empezar a trabajar en el DSP.

- Se recomienda, para una Futura implementación, el diseñar e implementar filtros no lineales con imágenes, ya que se hizo una base de filtros no lineales

BIBLIOGRAFÍA

ROBLES PRIETO, Paula Andrea y SANCHEZ CARDOZO, John Haumer. Tesis: Análisis espectral de una señal utilizando el DSP TMS320C6211. Universidad Manuela Beltrán. Bogota DC 2005.

CD ROM. Code Composer Studio. TMS320C54X. Texas Instrument. 1999. Archivo: SPRU 328b.pdf

CD ROM. Code Composer Studio. TMS320C54X. Texas Instrument. 1999. Archivo SPRU 328.pdf

CD ROM. Code Composer Studio. TMS320C54X. Texas Instrument. 1999. Archivo SPRU 327.pdf

CD ROM. Code Composer Studio. TMS320C54X. Texas Instrument. 1999. Archivo SPRU103.pdf

KUMAR B. Preetham. Digital Signal Processing Laboratory. California State University. 2005.

POULARIKAS. Alexander D. Formulas and Tables for Signal Processing. United States of America. Electrical engineering handbook series. 1998.

HAYES, Monson H. Schaum's Outline of Theory and Problems of Digital Signal Processing. United States of America. McGraw-Hill. 1999.

KARRIS, Steven T. Signals and Systems with MATLAB Applications Second Edition. United States of America. Orchard Publications. 2003.

TEXAS INSTRUMENTS. TMS320C54X Code composer Studio Tutorial. United Status of America. 2000.

THE MATH WORKS Inc. Instrument Control Toolbox Serial Port Tutorials. United States of America. 2006

ANEXO A

MANUAL DE ENTRENAMIENTO PARA PROGRAMAR EL DSP TMS320C54X DE TEXAS INSTRUMENT DE PUNTO FIJO

1. CONECTAR EL DSK C5402 AL COMPUTADOR

a. Apague el PC.

b. Conecte la tarjeta TMS320VC5402 al puerto paralelo LPT y conectar los cables de poder y de comunicaciones.

c. Prenda el PC

d. Inserte el CD Code Composer y haga Click en el icono install.

e. Cuando conecte la tarjeta, el Usuario puede observar cuando se activa el LED de color verde que está en la tarjeta DSP.

Nota : Antes de que usted instale el software de DSK, asegúrese que el puerto paralelo LPT del PC este configurado para ECP y modo de EPP.

1.2 INSTALANDO EL SOFTWARE

La instalación del Code Composer Studio consiste en dos pasos: Instalar el software en computador y ejecutar la aplicación del Code Composer Studio. Donde los requisitos mínimos para ejecutar son los siguientes:

• Un PC .486

• Microsoft Windows 95, 98 o NT 4.0

• 100 megabytes de espacio del disco-duro

• 16 megabytes de RAM

• Pantalla SVGA (640 x 480)

1.3 RECOMENDACIONES MINIMAS:

Pentium de

• 133 MHZ como mínimo

• Netscape Navigator 3.0 o Internet Explorer 3.0

• 32 Mega-bytes de RAM

• Pantalla SVGA (1024 x 768)

1.4. INSTALANDO EL CODE COMPOSER ESTUDIO A WINDOWS.

Nota: Para WINDOWS , usted debe instalar el Código Compositor Studio según los privilegios del administrador.

a. Inserte el CD de la instalación en la unidad de CD ROM, en un instante en la pantalla debe aparecerle un ejecutable en pantalla azul haga Click en Install, si no aparece este pantallaso, vaya al explorador de Windows y busque en la unidad de su CD ROM setup.exe.

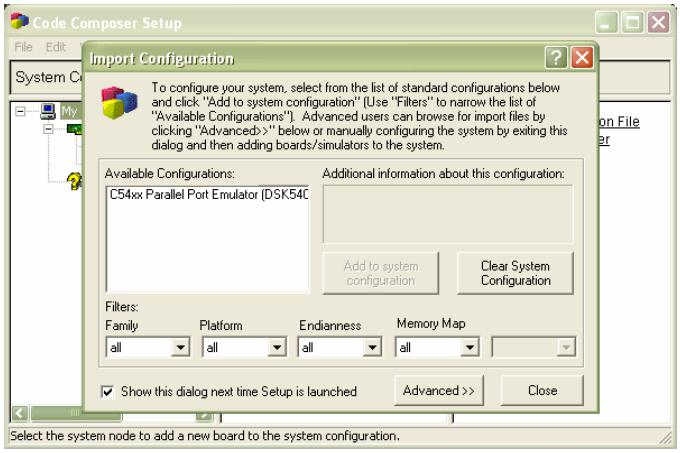
b. Escoja la opción para instalar el Code Composer Studio.

c. Responda los diálogos que aparecen mientras se corre el programa de instalación.

El procedimiento de la instalación crea dos iconos del programa, en el escritorio llamados Setup CCStudio y el otro CCStudio

d. Haga doble Click en Setup CCStudio

e. Le aparece una pantalla con el nombre de **Import Configuration**, en esta pantalla se hace la configuración de la tarjeta al PC, se configura el puerto a utilizar.

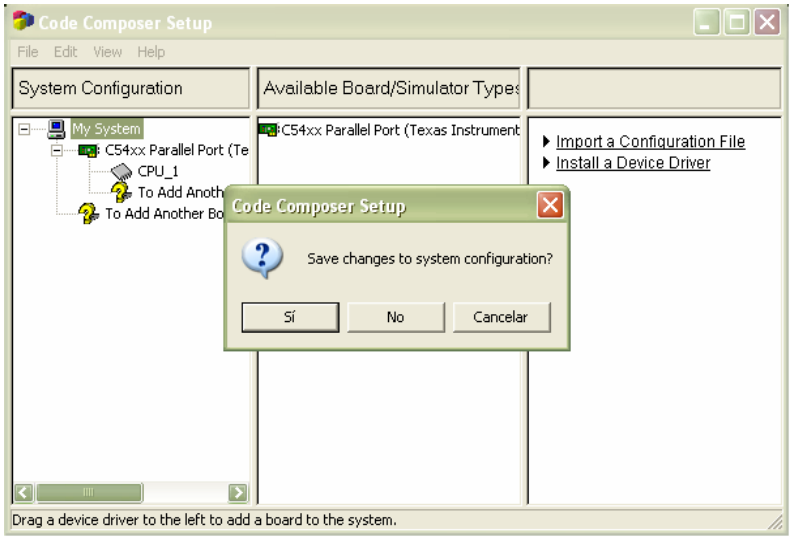


a. Haga doble Click en C5402 Parallel port emulador

b. Haga doble Click en el icono (Add System Configuration), para que cargue esta configuración al sistema.

c. Haga Click en el icono de cerrar (Close)

d. valla hacia FILE → y seleccione EXIT, Le aparece una pantalla donde le dice que si quiere salvar los cambios de la configuración y hace Click en **SÍ**



a. En este momento ya se encuentra configurado el DSP TMS320VC5402, por lo tanto puede iniciar trabajos en el programa de Code Composer Estudio Haciendo doble Click en el icono que tiene en el escritorio llamado CCStudio.

Nota: Si al abrir este icono le aparece un mensaje como este:

Puede significar lo siguiente:

- Que el cable de comunicaciones del DSP no esta conectado al puerto LPT del computador o este se encuentra apagado.

- Que el DSP no esta bien configurado, y tiene que volver al iniciar la configuración tal como aparece en numeral 4, repita los pasos y vuelva a intentarlo.

- Si hace Click en **Anular** el DSP no lo dejara entrar al programa Code Composer Estudio.

- Si hace Click en **Reintentar** el programa puede que inicie o vuelva a aparecer este mismo mensaje.

- Si hace Click en **Omitir** el programa inicia pero no habilita todas las funciones para que pueda trabajar correctamente.

LAB1: TUTORIAL CODE COMPOSER STUDIO.

Objetivo: familiarizarse con el uso del code componer Studio.

En este ejercicio se hará la familiarización del CCS cubriendo las siguientes secciones: crear proyectos, corriendo, analizando y reseteando el proyecto construido.

Paso 1: **Creando un "*Project*"**

- Abra el CCS.
- Cargue en GEL C54X y C5402_DSK_INIT, esto para resetear la tarjeta dsk.
- Vaya a projects > new.
- Escriba el nombre del proyecto y la ubicación donde será guardado.
- Debe guardar como tipo .mak.
- Cuando se cree el proyecto se vera en la ventana el nombre del proyecto con la extensión .mak.

Paso 2: **Crear un archivo fuente**

- En **File > New > Source File**.

- Abra el editor de ventana.
- Copie el siguiente código en asembler, declarando 10 valores:

```
.data
coeff:          .short 10
                .short -63
                .short 20
                .short 4
                .short -9
                .short 52
                .short -8
                .short 7
                .short 10
                .short -1
```

- En **File > Save As.**
- Guarde el archivo con el nombre que quiera, con extensión .asm
- Cree otro archivo fuente guárdelo como main.c en el proyecto.

//main.c C program

```
#include <stdio.h>
void main()
{
printf(“DSP\n”);
printf(“HELLO WORLD\n”);
}
```

Paso 3: **creando un archivo .cmd para mapear la memoria. (Llamado command file).**

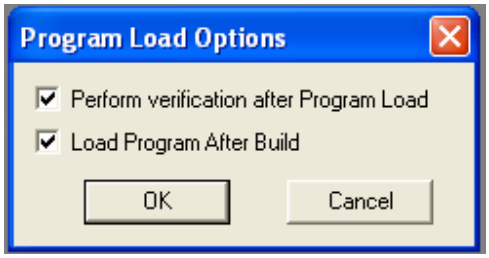
- Localicé el archivo hello.cmd y agréguelo al proyecto**.**
- Abra el archivo y localice en la línea Sections { .data
- Los datos serán guardados en la memoria externa EXRAM que empieza en la localidad de datos 0x1400
- Guarde el archivo como lab1.cmd.

Paso 4: **adicionando archivos al proyecto.**

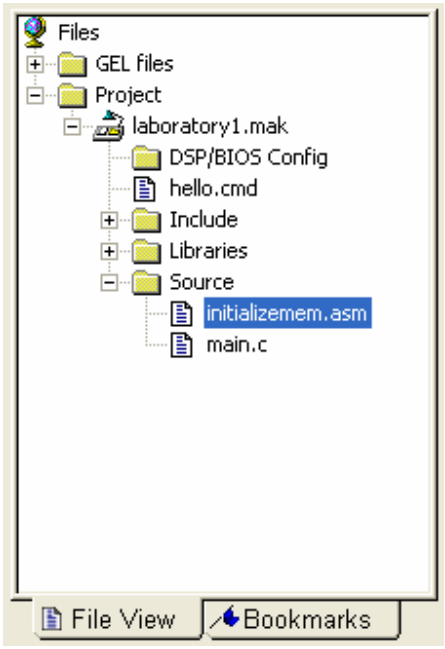
- Seleccione *Project* ***> Add Files to Project***.
- Abra el archivo .asm creado inicialmente que contiene el código ejecutable.
- Repita lo anterior para el archivo lab1.cmd y main.c
- Adicione el archivo ubicado en C:\ti\c5400\cgtools\lib****rts.lib. este archivo aparecerá en las librerías del proyecto a ejecutar.

Paso 5: **Crear el archivo ejecutable. lab1.out**

- En el menú **“option”** haga click en **“program load”** se mostrara la siguiente ventana:



- Una vez creado el proyecto se pueden desplegar las carpetas y ver en la parte izquierda la siguiente ventana:



- En el menu **Project > Rebuild All** compila, ensambla y carga todos los archivos del proyecto y produce el archivo ejecutable lb1.out. en la ventana inferior se muestran los errores o advertencias del proyecto a ejecutar.

- En el menu **Project > Build** puede ser usado cuando se ha hecho algún cambio en el proyecto, y vuelve a compilar ensamblar y cargar el proyecto, con los cambios efectuados.

Paso 6: **Ejecutando el programa:**

Para correr el programa primero tendremos que cargar el programa en la memoria del DSP.

- Si ejecutamos en el paso 5 la primera opción cuando carguemos todo el programa (**Rebuild All)** se cargará el programa automáticamente en la memoria del DSP.
- Para correr el programa seleccione **Debug > Run.**
- Aparecerá la ventana **Disassembly.** Con el programa en asembler a ejecutar.

PASO 7: **Mirando el contenido de la memoria.**

En este paso miraremos el contenido de la memoria que se guardo en el archivo **inicio.asm** mapeado en la sección .data del mapa de memoria **uart.cmd.**

Si miramos el archivo **uart.cmd** en SECTIONS { .data > EXRAM PAGE 1**,** y el mapa de memoria indica: EDATA: origin = 0x1F10, len = 0x4000, lo cual

quiere decir que los datos estan en la localidad 1400h, y se tiene una longitud de 80000h, para guardar los datos que se quieran en este espacio.

- Seleccione **View** > **memory**.
- En la posición **Address memory,** coloca 0x1F81, donde se localiza coeff:
- Seleccione **Format**: **16bit Signed Int**.
- En la ventana de memoria aparecerá el contenido en cada localidad de memoria.
- Compare los primeros diez valores con los registrados en el archivo **inicio.asm.**

PASO 8**: Mostrar grafica de los datos.**

Complete las siguientes instrucciones para ver la grafica de los datos en la memoria:

- Seleccione View > Graph > Time/frequency.
- En Start Address: 0x1F81
- En Acquisition Buffer Size: 10
- En Display Data Size: 10.
- En DSP Data Type: 16-bit signed integer.
- Compare los valores de la grafica con los del archivo **initializemen.asm.**

PASO9 : Modificacion del programa Main.c

- Haga doble click en Main.c en la ventana del proyecto.
- Haga las siguientes modificaciones:

//main.c C program –Modification 1

```
#include <stdio.h>
void main()
{
int i;
short *point;
point = (short *) 0x1f81;
printf("NUMEROS EN MEMORIA\n");
for (i=0;i<10;i++)
{
printf("[%d]%d\n,"i, point[i]);
}
printf("FIN\n");
}
```

Se asigna un puntero en los datos que tenemos en la memoria**.** Esto permite traer los datos e imprimirlos en la ventana **Stdout.**

PASO 10: Chequear una variable durante la ejecución del programa:

Breakpoints y **watch Windows** son usadas para mirar las variables mientras se corre el programa.

- Haga doble click en main.c
- Ponga el cursor en la linea : **point = (short*) 0x1F10**.
- Haga click derecho y vaya a **Toggle Breakpoint**.
- Haga click en la ventana **Stdout para** ver los datos de la memoria.
- Repita el anterior procedimiento en la linea **printf("[%d]%d\n",",i, point[i]);**

Para adicionar variables **a watch window:**

- Use el mouse para resaltar la variable point al principio de la linea **point = (short *).**
- Haga click derecho y seleccione **Add to Watch window**.un **watch window** aparecerá con la variable point.
- Repita el procedimiento para la variable **i** comenzando la linea **printf("[%d]**.
- Selecione **Debug > run.**

El programa para en el **Breakpoint** y en la ventana **watch window** aparecen las variables point y i; copie los valores que tiene cada variable: _____

Puede continuar mirando el programa en **Debug > animate**, para correrlo todo, cuando utiliza este comando, debe parar el sistema con Debug > Halt.

Si se quiere repetir el ejercicio seleccione **Debug** > **Restart, Run**, **Step over.**

Para remover los breakpoints, seleccione Debug > breakpoints, delete all.

- Haga doble click en main.c, agregue al programa lo siguiente:

```
//main.c C programa -Modificacion 2

#include <stdio.h>
void main()
{
int i, ret;
short *point;
point = (short *) 0x1f81;
printf("DATOS Y SUMA DE DATOS\n");
for (i=0;i<10;i++) {
                        printf("[%d]%d\n," ,i, point[i]);
                        }
ret = ret_sum(point,10);
    printf("Sum =%d\n," ,ret);
printf("FIN\n");
}
int ret_sum(const short* array, int N)
{
int count, sum;
sum=0;
for(count=0; count<N; count++)
sum += array[count];
return(sum);
}
```

- Guarde, reconstruya y cargue el programa en el memoria del DSP.
- Corra el programa y pruebe lo anterior mirando en que localidades de memoria se encuentran los números, y comprobando la operación hecha por el DSP.

LAB2: IMPLEMENTANDO OPERACIONES MATEMÁTICAS EN C5402 DSK. (MAC)

Objetivo: manejar algunas funciones del lenguaje C.

En este laboratorio se simulará una calculadora, multiplicando dos arrays, y después sumando el contenido de todas las multiplicaciones. El Algoritmo se hará en lenguaje C.

Operaciones como adición, sustracción, y multiplicación, son las bases en un Procesador Digital de Señal. Una muy importante aplicación es la multiplicación/acumulación, ya que se implementan en filtro digitales, correlación, y análisis espectral.

PASO1: crear el archivo donde alojamos los números para aplicarle el algoritmo.

//dotp5.h

```
#define x_array 1, 2, 3, 4,1
#define y_array 0, 2, 4, 6, 1
```

Copie el siguiente algoritmo:

```
//dotp5.c Multiplica dos arrays, cada uno con cinco numeros.

int dotp(short *a, short*b, int ncount);          //función

# include <stdio.h>                               //para imprimir
```

```
# include "dotp5.h"                                    //archivo donde están los datos

# define count 5                                      //# de datos de cada fila

short x[count] = {x_array};                                  //declara primera fila

short y[count] = {y_array};                                  //declara segunda fila

main ()

{
int result = 0;                                  //resultado de la suma de los productos

result = dotp(x,y,count);              //call a la función dotp
printf("resultado =% d (decimal),", result);        //imprime resultado
 }
int dotp(short *a, short*b, int ncount)
  {
int sum = 0;                                        //init sum
int i;
for (i= 0; i<ncount ; i++)
sum += a[i] * b[i];                                 //suma de productos
return (sum);                                       //retorna la suma como resultado
 }
```

PASO 2: Corra el programa y mire que hace cada función, haciendo las mismos pasos y pruebas como en el laboratorio1.

LAB3: ARITMÉTICA EN PUNTO FIJO

Objetivo: Familiarizarse con el uso de números fraccionarios en aritmética de punto fijo: fracciones binarias (formato Qn); haciendo operaciones de adición y multiplicación, también conocer acerca del C54X extensión de signo (SXM), modo overflow (OVF), y números fraccionarios en formato binario (FRCT).

FUNDAMENTO TEORICO:

Aritmética fraccional en el DSP C54X

Todos los números son tratados en Formato Q15, el MSB se utiliza como bit de signo los otros quince bits de la derecha son los números en punto binario.

El producto de dos números en formato Q15 es Q30.

De los 16 bits, los quince menos significativos representan la fracción, más el bit 16 que representa el signo de la fracción.

Overflow: (desbordamiento). En general, overflow hace referencia a un exceso de datos que pueden ser perdidos o transferidos.

Normalizando las fracciones:

32K = 32768

Fracciones	Espacio	Numero Hex.
~ 1	32k – 1	7FFFh
½	16K	4000h
0	0	0000h
- ½	-16K	C000h
- 1	-32K	8000h

Para representar números fraccionarios en el DSP se debe tener en cuenta que se tiene 16 bits, para hacer la respectiva representación.

Ejemplo:

Copie el siguiente programa en asembler en el cual se representan números fraccionarios, en formato Q15. Analice que hace el programa.

```
X:  .int 32768*325/1000     ; 0.325
Y:  .int 32768*625/1000     ; 0.625
Z:  .int 0                  ; x*y espacio resultado  multiplicación.

    LD #X,   DP             ; carga los datos en la pagina de datos.
    LD X,    T              ; carga el registro T con el contenido de X
    MPY      Y,    A              ; multiplica el registro T con el contenido de Y,
                            ; y pone el resultado en el acumulador A.
```

```
STH     A, 1, Z                 ; transforma de Q30 a Q15.
```

Overflow

Overflow ocurre cuando el resultado de 32 bits excede el máximo valor de número positivo (7FFF FFFFh) o cuando el resultado es mas negativo que el mas negativo del DSP un numero de 16-bit (8000 0000h).

El overflow puede ocurrir cuando se hace una operación matemática y excede el número máximo permitido, entonces ocurrirá un overflow.

Puede ocurrir un averflow durante un intermedio de una operación, pero el resultado se produce de forma correcta.

Notación Q15.

- Notación fraccionaria con signo en 16 bits
- Utilizada por los DSPs de 16 bits y coma fija, el producto no produce overflow.
- Escala: *j* =15.
- Rango y resolución:

[0, 99997,-1] ([132767, -32768])

2^15 = 0,000030517578125

Ejemplos:

Decimal	**Q15 = Decimal x 2^15**	**Q15 (entero)**
0.5	0.5 x 32767	16384 (0100000000000000)
0.05	0.05 x 32767	1638 (0000011001100110)
0.0012	0.0012 x 32767	39 (0000000000100111)

Cuando la operación en fraccionarios es > 1 se produce overflow.

Copie el siguiente programa y corralo paso a paso, indicando en cada linea que se hace.

```
//main.c C programa -Modificacion 2

#include <stdio.h>
 #include "dsplib.h"
 float yf[3] = { 0 , 0, 0};
 float xf[1] = { 0 };
 short x[3];
void main()
{
int i, ret;
short *point;
point = (short *) 0x1f81;
for (i=0; i<3; i++) ;
 q15tofl(point,yf,3);
printf("DATOS Y SUMA DE DATOS\n");
for (i=0;i<3;i++) {
```

```
                        printf("[%d]%f\n," ,i, yf[i]);
                        }
fltoq15(yf,x,3);
ret = ret_sum(x,3) ;
*(x)=ret;
q15tofl(x,xf,1);
    printf("Sum =%f\n," ,xf[0]);
printf("FIN\n");
}
int ret_sum(const short* array, int N)
{
int count, sum;
sum=0;
for(count=0; count<N; count++)
sum += array[count];
return(sum);
}
```

LAB4: IMPLEMENTACION DE FILTROS EN SEÑAL DE AUDIO.

Objetivo: comprender el funcionamiento de los periféricos de audio de la tarjeta DSK TMS320VC5402, aplicando filtros digitales a señales de entrada, enviadas por el micrófono.

Descripción:

· Lectura y escritura en el socket de audio.
· Realización de un filtro de FIR.

Se necesitará un micrófono y un altavoz/auriculares para conectarlos a la tarjeta DSK.

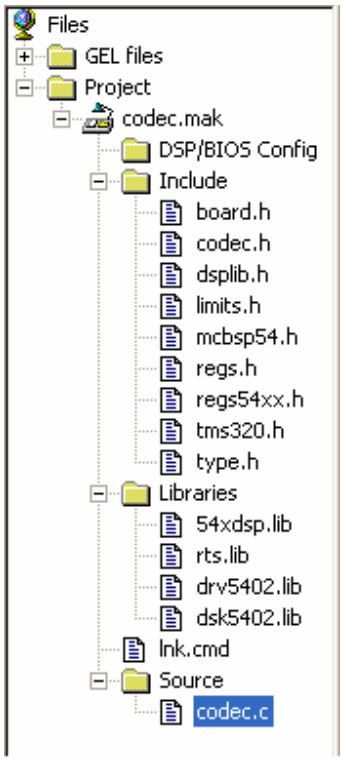
Este laboratorio implicará crear un proyecto de audio guarda muestras de un micrófono, los procesa añadiendo una ganancia y un filtro FIR, enviando la señal filtrada a los auriculares.

PASO 1:

Conecte la tarjeta a la fuente de energía y encienda el PC. copie el archivo codec.c a la carpeta myprojects/codec. Inicie el code composer Studio.

Paso 2:

Haga *Click* en el menu ***Project*** □ ***Open Project.*** Si no ha creado el proyecto creelo, como **codec.mak**. abra cada carpeta e incluya los siguientes archivos:



En codec.c se tendra:

```
/*************************************************************************/
// Codec.c
//
// Digital Loopback example
//
```

```
/*****************************************************************************/

#include <type.h>
#include <board.h>
#include <codec.h>
#include <mcbsp54.h>
#include <dsplib.h>

/*****************************************************************************/
// Function Prototypes
/*****************************************************************************/

void delay(s16 period);
int x;
int j;
int k;

/*****************************************************************************/
// Global Variables                                                 */
/*****************************************************************************/
#define GAIN 5
HANDLE hHandset;

short data[20]={0};
short salida[16]={0};
short delaybuff[16]={0};
DATA *delayptr1 = &(delaybuff[0]);
short coeffs[5]=
        {-1299,-994,-182,1067,2567};
        //Coefficientes de filtro pasabajo.
```

```
/*****************************************************************************/
// MAIN
/*****************************************************************************/
void main()
{
   s16 cnt=2;
   if (brd_init(100))
      return;
    */los leds encienden cada uno con un retardo.
    while ( cnt-- )
    {
        brd_led_toggle(BRD_LED0);
        /* brd_delay_msec(1000); */
        delay(1000);
        brd_led_toggle(BRD_LED1);
        delay(1000);
        brd_led_toggle(BRD_LED2);
        delay(1000);
    }
hHandset = codec_open(HANDSET_CODEC);
codec_dac_mode(hHandset, CODEC_DAC_15BIT);
codec_adc_mode(hHandset, CODEC_ADC_15BIT);
codec_ain_gain(hHandset, CODEC_AIN_6dB);
codec_aout_gain(hHandset, CODEC_AOUT_MINUS_6dB);
codec_sample_rate(hHandset,SR_16000);

while (1)
{
   /* Read sample from handset codec */
   data = *(volatile u16*)DRR1_ADDR(HANDSET_CODEC);
```

```
    //Filter sample
    fir(&data,&coeffs[0],&data,&delayptr1,16,1);
    // (input,filter,output,delay buffer,filter length,output length)
    /* Write back to handset codec */
    *(volatile u16*)DXR1_ADDR(HANDSET_CODEC) = GAIN*data;
}
}
void delay(s16 period)
{
    int i, j;
    for(i=0; i<period; i++)
    {
        for(j=0; j<period>>1; j++);
    }
}
```

PASO 3:

Haga *Click* en ***Project*** □ ***Rebuild All, Debug*** □ ***Run*** despues de que se hayan encendido y apagado los leds.de la tarjeta DSK, habla por el microfono; en los auriculares se escuchara el filtrado de la señal de audio. En ***Debug*** □ ***Halt***. Se para el programa.

Código en MATLAB para generar los coeficientes en MATLAB:

Filtro pasa banda:

```
N = 16;
fs = 8000;
f = [1400 1600];
wn = 2*pi*f/fs;
wn = wn/pi;
h = fir1(N, wn)
```

Filtro pasa baja:

```
N = 16;
fs = 8000;
f = 1400;
wn = 2*pi*f/fs;
wn = wn/pi;
h = fir1(N, wn)
```

Filtro pasa alta:

```
N = 16;
fs = 8000;
f = 2000;
wn = 2*pi*f/fs;
wn = wn/pi;
h = fir1(N, wn, 'hig')
```

Filtro rechaza bada:

```
N = 16;
fs = 8000;
f = [1400 1600];
wn = 2*pi*f/fs;
wn = wn/pi;
h = fir1(N, wn,’stop’)
```