IMPLEMENTACIÓN DE ALGORITMOS PARA EL PROCESAMIENTO DE IMÁGENES DE RADAR SAR USANDO DSPs

ENDER ARDILA SÁNCHEZ
LUIS FABEL PUERTO CIPAGAUTA

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FÍSICO- MECÁNICAS
ESCUELA DE INGENIERÍAS DE ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES
BUCARAMANGA
2008

IMPLEMENTACIÓN DE ALGORITMOS PARA EL PROCESAMIENTO DE IMÁGENES DE RADAR SAR USANDO DSPs

ENDER ARDILA SÁNCHEZ
LUIS FABEL PUERTO CIPAGAUTA

Este proyecto se presenta como requisito para optar al titulo de ingeniero electrónico.

Director
M.Sc. ANA BEATRIZ RAMIREZ

Codirector
Ph.D DOMINGO RODRÍGUEZ.

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FÍSICO- MECÁNICAS
ESCUELA DE INGENIERÍAS DE ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES
BUCARAMANGA
2008

AGRADECIMIENTOS

Agradecemos a nuestros padres y familiares por el apoyo brindado durante todo este tiempo de vida universitaria.

A Ana Beatriz Ramírez Silva por todo su apoyo como directora y consejera en el desarrollo de este proyecto de grado.

A Domingo Rodríguez por su colaboración prestada para la realización de este trabajo.

Al profesor César Duarte por su colaboración en la aprobación del plan del proyecto desarrollado.

Al grupo de investigación CPS por la oportunidad de trabajar en la realización de este tipo de proyectos de investigación.

A la universidad de Puerto Rico y el departamento de Ingeniería Eléctrica y Computadores por facilitar las herramientas hardware utilizadas para el desarrollo del proyecto.

CONTENIDO

LISTA DE TABLAS

LISTA DE FIGURAS

LISTA DE ANEXOS

RESUMEN

TÍTULO: IMPLEMENTACIÓN DE ALGORITMOS PARA EL PROCESAMIENTO DE IMÁGENES DE RADAR SAR USANDO DSPs.

AUTORES: ENDER ARDILA SÁNCHEZ Y LUIS FABEL PUERTO CIPAGAUTA.

PALABRAS CALVES: Radar de apertura sintética, función de ambigüedad, transferencia de datos, respuesta al impulso del sistema.

Los sistemas de radares de apertura sintética son dispositivos de tipo activos que recopilan información de la superficie terrestre mediante generación de señales de cierta longitud de onda y recepción de los ecos resultantes, el tipo de información obtenida por estos sistemas permite la formación de imágenes bajo condiciones desfavorables o inadecuadas en relación con otros sistemas. Los datos recogidos por un radar de apertura sintética deben ser procesados para extraer información que pueda ser interpretada y utilizada con diferentes fines, sin embargo este procesado puede consumir tiempo valioso y retrasar la toma de decisiones importantes con base a esta información. Es por eso que se implemento una combinación de hardware y software que agiliza el procesado y la transferencia de estos datos entre un sistema embebido y un dispositivo con gran capacidad de almacenamiento de información. En este trabajo se presenta una descripción de las funciones con las que se modela este tipo de sistemas como es el caso de la función de ambigüedad la cual es utilizada como respuesta al impulso del sistema de radar, además realizando su implementación mediante diferentes algoritmos que permiten aumentar el rendimiento del hardware utilizado para reducir el tiempo de cómputo total. Así mismo se presenta una guía que describe los diferentes procedimientos para la correcta utilización del hardware y software utilizado en este trabajo.

---

* Proyecto de Grado

**Facultad de Ingenierías Físico-Mecánicas, Escuela de Ingeniería Eléctrica, Electrónica y Telecomunicaciones, Director M.Sc. ANA BEATRIZ RAMIREZ

ABSTRACT

TITLE: IMPLEMENTATION OF ALGORITMOS FOR THE IMAGE PROCESSING OF RADAR SAR USING DSPs.

AUTHOR: ENDER ARDILA SÁNCHEZ Y LUIS FABEL PUERTO CIPAGAUTA.

KEY WORDS: Synthetic Aperture Radar, function of ambiguity, Data transfer, The impulse response of the system.

Synthetic aperture radar systems are active-type devices that compile information from the terrestrial surface by means of the generation of signals of certain length-wave and the reception of the resulting echoes. The kind of information obtained by these systems allows the formation of images under unfavorable or inadequate conditions compared to other systems. The data collected by a synthetic aperture radar must be processed to extract information that can be interpreted and used to several purposes. Nevertheless, this processing can be highly time-consuming and delay important decision making based on this information. That is the reason why a combination of software and hardware was implemented, which nimbles the processing and the data transfer between an embebide system and a device with great information storage capacity. In this work, a description of the functions that model this type of systems is presented as it is the case of the function of ambiguity, which is utilized as response to the impulse of the radar system. Besides, its implementation was carried out by means of different algorithms that allow to enhance the performance of the hardware employed to reduce the total computation time. Similarly, a guide that describes the different procedures for the correct usage of the hardware and software utilized in this work is presented.

---

*Project of Grade

**Ability of Physical-Mechanical Engineerings, Electric, Electronic School of Engineerings and Telecommunications, Managing M.Sc. ANA BEATRIZ RAMIREZ

INTRODUCCIÓN

El radar SAR (Synthetic Aperture Radar) es un dispositivo de tipo activo que envía señales electromagnéticas y recibe las señales reflejadas por el objeto con el fin de obtener información para generar una imagen usando procesamiento digital. La aplicación de estas tecnologías se puede encontrar en diferentes ámbitos como reconocimiento geográfico, monitoreo ambiental y diferentes aplicaciones militares como búsqueda de objetivos. Una ventaja que presenta la adquisición de imágenes por medio de esta tecnología es que las imágenes pueden ser adquiridas en condiciones desfavorables, como condiciones climáticas adversas o condiciones de iluminación inadecuadas sin que se vea afectada.
Sin embargo el radar SAR solo obtiene una información preliminar conocida como datos crudos que requiere un procesamiento para poder obtener las imágenes de interés y debido al gran volumen de datos adquiridos para la formación de una imagen el procesamiento se vuelve demorado. Es por eso que la implementación tanto del Software como el Hardware juegan un papel importante si se quieren obtener resultados satisfactorios y mas aún si se desea obtener imágenes tiempo real.

En este trabajo se utiliza un DSP (Digital Signal Proccessor) como herramienta para el procesamiento de imágenes tipo SAR, junto con el emulador USB560 para transmisión de datos entre el DSP y un PC, además de la ayuda de MATLAB que permitirá la respectiva visualización y validación de los resultados obtenidos.

En el primer capítulo se presentan los conceptos básicos sobre radares SAR, Transformada de Fourier en 1D y 2D, Convolución en 2D y función de Ambigüedad.

En el segundo capítulo se presenta la descripción del software y hardware usado en el procesamiento de imágenes tipo SAR.

En el tercer capitulo se muestran los resultados obtenidos junto con una comparación con los resultados obtenidos en trabajos previos relacionados con este trabajo.

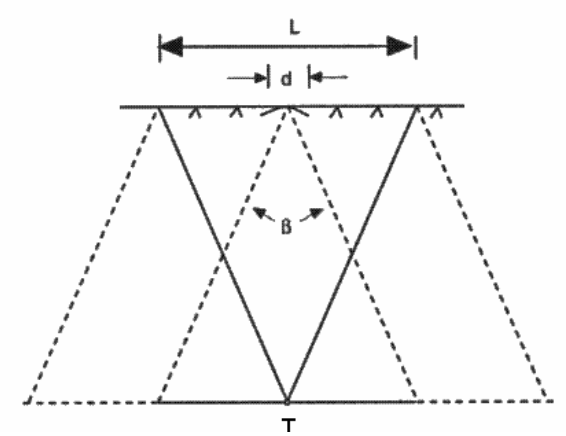
Finalmente se presentan las recomendaciones junto con las conclusiones del desarrollo de este proyecto; además se incluye una guía de usuario del DSK y el USB560 JTAG Emulator que contiene aspectos importantes para el funcionamiento adecuado de las herramientas Hardware y Software.

1. CONCEPTOS BÁSICOS

1.1. RADAR SAR

Los radares SAR son radares (Synthetic Aperture Radar) de tipo activo que emiten un haz electromagnético en el rango de las microondas, separados en intervalos pequeños de tiempo y reciben los ecos provenientes de las reflexiones de la señal al chocar con la superficie terrestre [6]. Una abertura sintética o antena virtual, consiste en un extenso arreglo de sucesivas y coherentes señales de radar que son transmitidas y recibidas por una pequeña antena que se mueve a lo largo de un determinado recorrido de vuelo u órbita, simulando una antena de longitud real de mayor tamaño [6].

Figura 1. Apertura sintética



Fuente: autores del proyecto

L: longitud de la antena sintética.

d: longitud antena real.

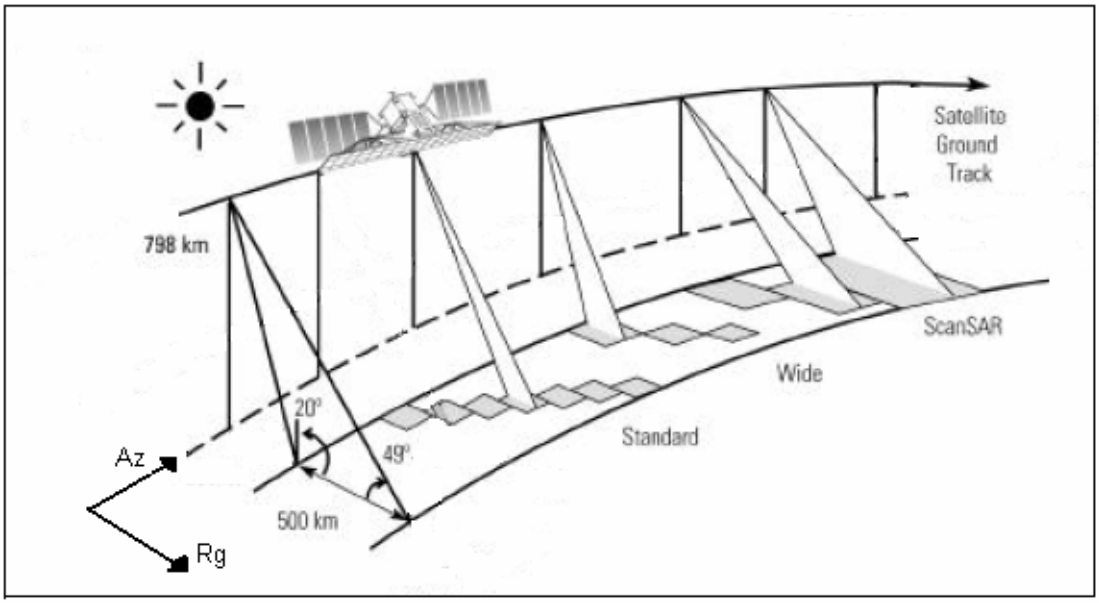
β: ancho del haz electromagnético

T: blanco

Por consiguiente para generar la imagen SAR se envían pulsos de señal a los mismos puntos de la superficie terrestre en dos o más momentos distintos de la trayectoria del radar [3].

La resolución de las imágenes obtenidas por un radar SAR es mejor que la resolución de las imágenes obtenidas por un radar tipo RAR (Real Apertura Radar).

Figura 2. RADARSAT modos de operación del SAR



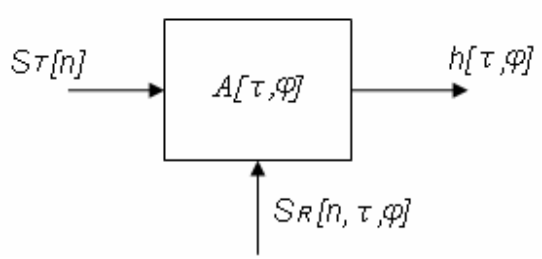
Fuente: referencia [7]

1.2. MODELO PARA LA FORMACIÓN DE IMÁGENES

El proceso requerido para la formación de imágenes con este tipo de radares se puede separar en tres etapas, las cuales son descritas a continuación.

1.2.1. Generación de la repuesta al impulso de un radar SAR. Cuando el radar ha almacenado tanto la señales transmitidas $S_T[n]$ como la señales recibidas $S_R[n,\tau,\varphi]$, los datos son enviados a una estación en tierra para ser procesados. Estos datos son utilizados para generar la función de ambigüedad que es utilizada como respuesta al impulso del sistema (ver figura 3).

Figura 3. Respuesta al impulso del sistema SAR

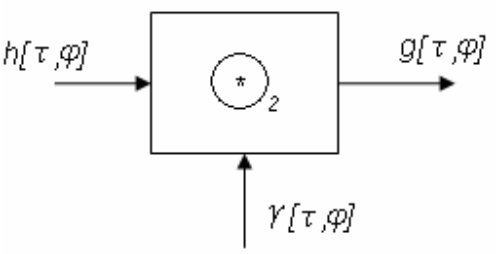


Fuente: autores del proyecto

Donde τ corresponde al tiempo de atraso de la señal recibida y φ al cambio de frecuencia o frecuencia Doppler de las señales recibidas.

1.2.2. Generación de datos crudos. Para generar de los datos crudos es necesario utilizar la matriz de la respuesta al impulso del sistema $h[\tau,\varphi]$, junto con la matriz de datos de la reflexión de la señal $\gamma[\tau,\varphi]$, realizando una convolución en dos dimensiones de estas matrices, se podrá obtener dichos datos. A continuación se muestra el diagrama de bloques para la obtención de los datos crudos.

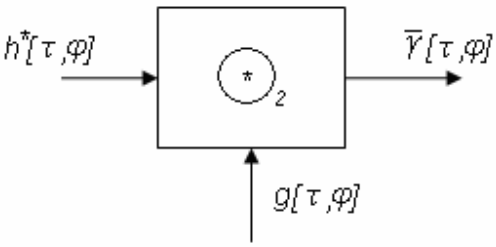
Figura 4. Generación de datos crudos



Fuente: autores del proyecto

1.2.3. Formación de la imagen. Realizando un proceso inverso al de generación de datos crudos, pero esta vez se utiliza el conjugado de la matriz de la respuesta al impulso *h*[τ , φ],* para ser convolucionada con los datos crudos *g[τ , φ],* y obtener la imagen [8].

Figura 5. Formación de imágenes



Fuente: autores del proyecto

1.3. TRANSFORMADA RÁPIDA DE FOURIER (FFT)

Muchas técnicas de procesamiento de señales se hacen en un dominio diferente al tiempo, debido a la ventaja que esto representa ya sea para disminuir la complejidad y cantidad de operaciones o por facilidad de análisis.

La base del algoritmo de formación de imágenes SAR y generación de datos crudos, es la operación de convolución, la cual se puede implementar a partir de la transformada de Fourier. Una aproximación de la Trasformada de Fourier se puede obtener a partir del algoritmo obtenido por Cooley Tukey de la Transformada Rápida de Fourier (FFT por sus siglas en ingles).

La DFT es definida como:

$$F(k) = \sum_{n=0}^{N-1} x[n] e^{\frac{-j2\pi kn}{N}} \qquad \text{Para k = 0, 1, 2,…., N-1} \qquad (1.1)$$

Si se considera el cálculo de la DFT para N puntos, el número de multiplicaciones complejas para implementar la ecuación (1.1) sería de N^2, pues por cada uno de los valores que toma k, la sumatoria necesita N multiplicaciones complejas con el termino $e^{\frac{-j2\pi kn}{N}}$ y N-1 sumas de resultado. Empleando la FFT el número de multiplicaciones será $N/2 * Log_{10}(N)$.

El algoritmo de la FFT esta basado en la implementación de DFTs de menor tamaño y usando la redundancia de términos W_N debido a su periodicidad y simetría.

$$W_N^{k+N} = W_N^k \qquad \text{Periodicidad} \qquad (1.2)$$

$$W_N^{k+\frac{N}{2}} = -W_N^k \qquad \text{Simetría} \qquad (1.3)$$

Además

$$W_N^{Nk} = 1 \qquad W_N^{n+N} = W_{\frac{N}{2}} \qquad (1.4)$$

Donde

$$W_N = e^{\frac{-j2\pi}{N}} \qquad (1.5)$$

FFT Si se elige el valor de N de forma tal que $N=r^m$ donde r se le denomina radix o base del algoritmo, siendo 2 el utilizado en este proyecto, de modo que $N=2^m$, por tanto, el algoritmo será denominado radix -2.

Entonces se expresa la ecuación (1.1) de la forma

$$F(k)=\sum_{n=0}^{N-1}x[n]W_N^{kn} \tag{1.6}$$

Si se expresa la FFT como suma de dos secuencias de FFT, la primera con la secuencias de 0 hasta el N/2-1 dato y la segunda desde N/2 hasta N-1, se tiene:

$$F(k)=\sum_{n=0}^{\frac{N}{2}-1}x[n]W_N^{kn}+\sum_{n=\frac{N}{2}}^{N-1}x[n]W_N^{kn} \tag{1.7}$$

Redefiniendo los límites de las sumatorias

$$F(k)=\sum_{n=0}^{\frac{N}{2}-1}x[n]W_N^{kn}+\sum_{n=0}^{\frac{N}{2}-1}x\left[n+\frac{N}{2}\right]W_N^{k\left(n+\frac{N}{2}\right)} \tag{1.8}$$

Como $W_N^{k\left(n+\frac{N}{2}\right)}=W_N^{k\frac{N}{2}}*W_N^{kn}$ y $W_N^{k\frac{N}{2}}=e^{\frac{-j2\pi}{N}*\frac{kN}{2}}=e^{-j2\pi k}=(-1)^k$

$$F(k)=\sum_{n=0}^{\frac{N}{2}-1}x[n]W_N^{kn}+(-1)^k\sum_{n=0}^{\frac{N}{2}-1}x\left[n+\frac{N}{2}\right]W_N^{kn} \tag{1.9}$$

Agrupando las sumatorias

$$F(k)=\sum_{n=0}^{\frac{N}{2}-1}\left[x[n]+(-1)^k x\left[n+\frac{N}{2}\right]\right]W_N^{kn} \qquad \text{Para k=0, 1, 2,. . ., N-1} \qquad (1.10)$$

Dividiendo (1.10) en dos secuencias:

Secuencia par:

$$F(2k)=\sum_{n=0}^{\frac{N}{2}-1}\left[x[n]+x\left[n+\frac{N}{2}\right]\right]W_N^{2kn} \qquad (1.11)$$

Reescribiendo el término W_N

$$F(2k)=\sum_{n=0}^{\frac{N}{2}-1}\left[x[n]+x\left[n+\frac{N}{2}\right]\right]W_{\frac{N}{2}}^{kn} \qquad \text{Para k=0, 1, 2,. . ., N/2-1} \qquad (1.12)$$

Secuencia impar.

$$F(2k+1)=\sum_{n=0}^{\frac{N}{2}-1}\left[x[n]-x\left[n+\frac{N}{2}\right]\right]W_N^{n(2k+1)} \qquad (1.13)$$

Reescribiendo el término W_N

$$F(2k+1)=\sum_{n=0}^{\frac{N}{2}-1}\left[\left[x[n]-x\left[n+\frac{N}{2}\right]\right]W_N^{n}\right]W_{\frac{N}{2}}^{nk} \qquad \text{Para k=0, 1, 2,. . ., N/2-1} \qquad (1.14)$$

De esta forma la DFT es implementada reduciendo la cantidad de operaciones, a continuación se presenta una comparación entre el cómputo directo de DFT y FFT radix-2.

Tabla 1. Comparación entre el cómputo directo de DFT con la FFT radix-2.

	DFT		FFT radix-2	
Numero de muestra	Productos complejos	Sumas complejas	Productos complejos	Sumas complejas
4	16	12	4	8
16	256	240	32	64
64	4096	4032	192	384
256	65536	65280	1024	2048
1024	1048576	1047552	5120	10240
N	N^2	N^2-N	(N/2) $\log_2 N$	N $\log_2 N$

Fuente: referencia [11]

Un ejemplo de la ventaja de la implementación de la FFT es que para el cálculo de una transformada de 256 muestras, este algoritmo es aproximadamente 32 veces más rápida que usar el algoritmo de la DFT y la diferencia será más significativa a medida que el número de muestras aumenta [4].

A continuación se presenta la implementación del algoritmo del la FFT tomando como ejemplo una secuencia con N=8.

Con la ecuación (1.12) secuencia par

Para k=0

$$F(0) = [x[0] + x[4]]W_4^0 + [x[1] + x[5]]W_4^0 + [x[2] + x[6]]W_4^0 + [x[3] + x[7]]W_4^0$$

Para k=1

$$F(2) = [x[0] + x[4]]W_4^0 + [x[1] + x[5]]W_4^1 + [x[2] + x[6]]W_4^2 + [x[3] + x[7]]W_4^3$$

Para k=2

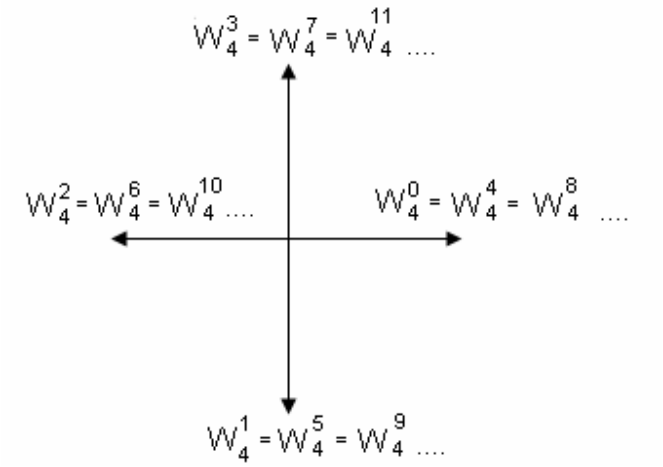
$$F(4) = [x[0] + x[4]]W_4^0 + [x[1] + x[5]]W_4^2 + [x[2] + x[6]]W_4^4 + [x[3] + x[7]]W_4^6$$

Para k=3

$$F(6) = [x[0] + x[4]]W_4^0 + [x[1] + x[5]]W_4^3 + [x[2] + x[6]]W_4^6 + [x[3] + x[7]]W_4^9$$

Por simetría y periodicidad de W_N (ver figura 6), se pueden reemplazar términos, en la figura 6 se muestra la relación de dichos términos cuando se tiene una sub-secuencia de N=4.

Figura 6. Simetría y periodicidad de W_N



Fuente: autores del proyecto

Las expresiones se puede reescribir:

$$F(0) = [x[0] + x[4]]W_4^0 + [x[1] + x[5]]W_4^0 + [x[2] + x[6]]W_4^0 + [x[3] + x[7]]W_4^0$$

$$F(2) = [x[0] + x[4]]W_4^0 + [x[1] + x[5]]W_4^1 + [x[2] + x[6]]W_4^0 - [x[3] + x[7]]W_4^1$$

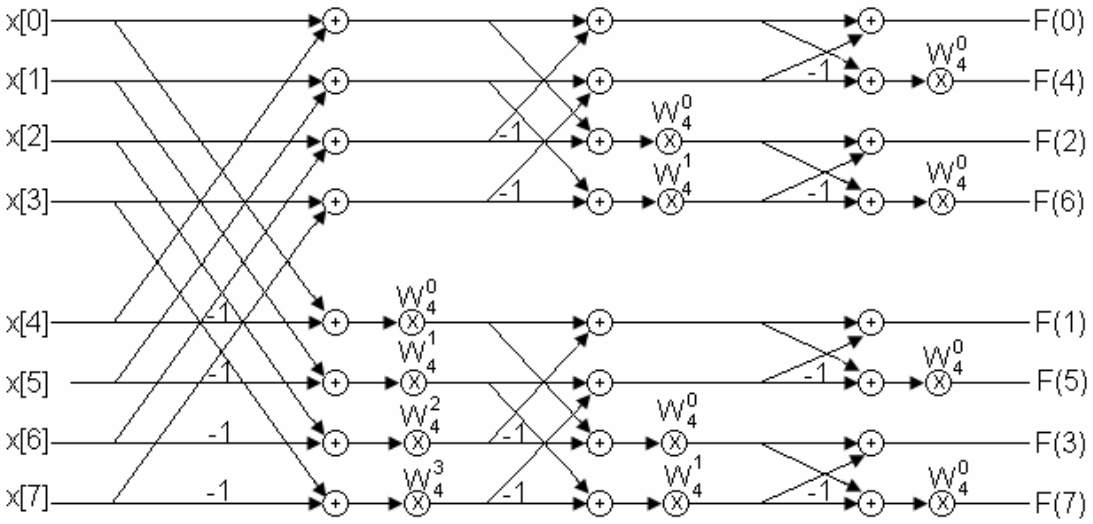
$$F(4) = [x[0] + x[4]]W_4^0 - [x[1] + x[5]]W_4^0 + [x[2] + x[6]]W_4^0 - [x[3] + x[7]]W_4^0$$

$$F(6) = [x[0] + x[4]]W_4^0 - [x[1] + x[5]]W_4^1 - [x[2] - x[6]]W_4^0 + [x[3] + x[7]]W_4^1$$

De igual forma se realiza el procedimiento para los términos impares con la ecuación (1.14).

Con estás nuevas expresiones se puede ver el orden y la forma que toman las secuencias, pudiendo representar el comportamiento de las ecuaciones (1.12) y (1.14) de la siguiente manera.

Figura 7. Representación operacional de una FFT



Fuente: autores del proyecto

Como la salida F(k) no se encuentra organizada es necesario implementar un proceso de reacomodación, este proceso se encuentra en aplicaciones en los DSPs, se conoce como direccionamiento con inversión de bit (bit-reversed adressing), el cual consiste en invertir la dirección de memoria de la FFT.

En la tabla 2 se muestra el funcionamiento de *bit-reversed adressing* para una FFT con N=8.

Tabla 2. Inversión de bits de memoria

FFT	Dirección de memoria	Inversión de bits de memoria	FFT ordenada
F(0)	000	000	F(0)
F(4)	001	100	F(1)
F(2)	010	010	F(2)
F(6)	011	110	F(3)
F(1)	100	100	F(4)
F(5)	101	101	F(5)
F(3)	110	011	F(6)
F(7)	111	111	F(7)

Fuente: autores del proyecto

1.4. TRANSFORMADA RÁPIDA DE FOURIER BIDIMENSIONAL (FFT 2-D)

Para el cálculo de la transformada en dos dimensiones, esta se puede encontrar usando el método la transformada unidimensional a las columnas y filas de la matriz de datos [2].

Dada una secuencia de datos x [n_1, n_2] la DFT-2D se define:

$$F(k_1,k_2)=\sum_{n_2=0}^{N_2-1}\sum_{n_1=0}^{N_1-1}x[n_1,n_2]e^{\frac{-j2\pi k_2 n_2}{N_2}}e^{\frac{-j2\pi k_1 n_1}{N_1}} \tag{1.15}$$

Reescribiendo la ecuación (1.15)

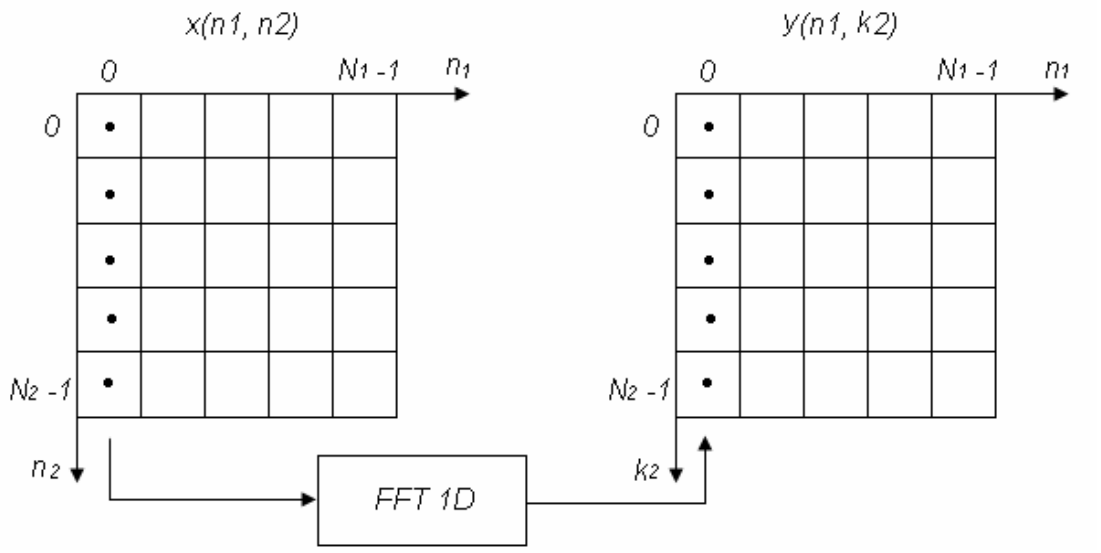
$$F(k_1,k_2)=\sum_{n_2=0}^{N_2-1}y[n_1,k_2]e^{\frac{-j2\pi k_1 n_1}{N_1}} \tag{1.16}$$

Donde

$$y[n_1,k_2]=\sum_{n_1=0}^{N_1-1}x(n_1,n_2)e^{\frac{-j2\pi k_2 n_2}{N_2}} \tag{1.17}$$

Si se considera n_1 fijo por ejemplo si $n_1=0$, entonces $y(n_1, k_2)$ será la DFT unidimensional de $x(n_1, n_2)|_{n1=0}$ con respecto a la variable n_1.

Figura 8. Transformada unidimensional por columnas



Fuente: autores del proyecto

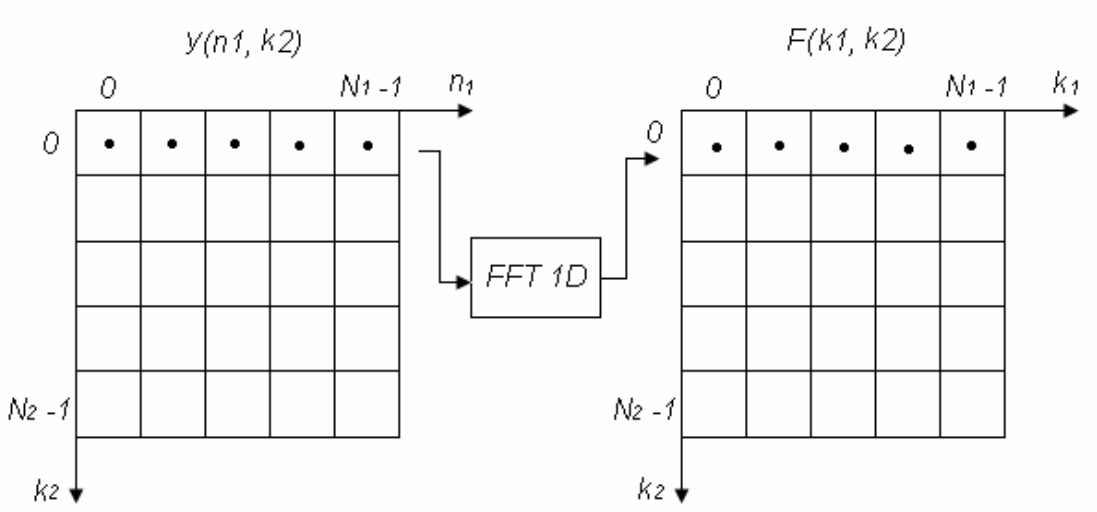
De esta manera se obtiene una matriz $y(n_1, k_2)$ para todos los posibles valores de n_1.

Para obtener $F(k_1, k_2)$ a partir de la matriz $y(n_1, k_2)$

$$F(k_1, k_2) = \sum_{n_2=0}^{N_2-1} y[n_1, k_2] e^{\frac{-j2\pi k_1 n_1}{N_1}} \qquad (1.18)$$

Igual que en el caso anterior pero esta vez se fija k_2, por ejemplo $k_2=0$ el cual corresponde a una fila de la matriz $y(n_1, k_2)$, a la cual se le aplica la transformada unidimensional, realizando este proceso para todos los posibles valores de k_2 .

Figura 9. Transformada unidimensional por filas



Fuente: autores del proyecto

1.5. TRANSFORMADA INVERSA RÁPIDA DE FOURIER (IFFT)

Definida como

$$x[n] = \frac{1}{N}\sum_{k=0}^{N-1} F(k) W_N^{-kn} \qquad (1.19)$$

Dado que tanto la DFT como la IDFT implican el mismo tipo de operaciones, es necesario implementar un algoritmo eficiente para dichos proceso.

Implementar la IFFT consistirá en seguir los mismos pasos que en la FFT, realizando un escalamiento por el factor 1/N y cambiando el factor W_N^{kn} por W_N^{-kn}.

El proceso seguirá la dirección de derecha a izquierda (recorrido inverso a la FFT).

Se debe tener en cuenta la reorganización del vector de entrada F (k), para que en la salida x[n] quede ordenada. De lo contrario hay que obtener los valores a la salida del cómputo de la IFFT para obtener el vector x[n].

1.6. TRANSFORMADA INVERSA RÁPIDA DE FOURIER BIDIMENSIONAL (IFFT 2-D)

Tomada como referencia la IFFT unidimensional, ecuación (1.19), se redefine para dos dimensiones así:

$$x[n_1, n_2] = \frac{1}{N_2 N_1} \sum_{k_2=0}^{N_2-1} \sum_{k_1=0}^{N_1-1} F(k_1, k_2) e^{\frac{j2\pi k_2 n_2}{N_2}} e^{\frac{j2\pi k_1 n_1}{N_1}} \quad (1.20)$$

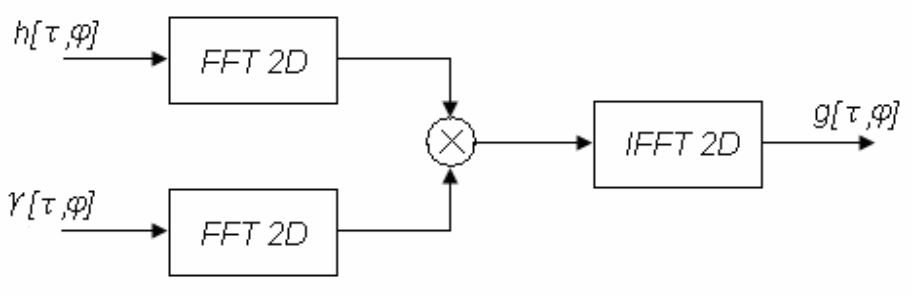
El proceso para el desarrollo de IFFT 2-D es similar al presentado en la FFT 2-D, el cual consiste en aplicar la IFFT 1-D a las columnas y al resultado obtenido aplicarle la IFFT 1-D a las filas.

1.7. CONVOLUCIÓN BIDIMENSIONAL

Empleando las propiedades de la transformada de Fourier, la convolución comprende en el dominio de la frecuencia al producto elemento a elemento de dos matrices.

Para este proyecto la convolución se realizará en forma indirecta como se muestra en la figura 10, ver numera 1.2.

Figura 10. Convolución método indirecto



Fuente: autores del proyecto

1.8. FUNCIÓN DE AMBIGÜEDAD

La función de ambigüedad se puede considerar una herramienta tiempo-frecuencia, se utiliza para obtener una estimación simultánea de los cambios de tiempo y frecuencia de los ecos recibidos de las señales transmitidas, dando una estimación más eficaz que si se utilizará representaciones en el dominio del tiempo o de frecuencia por separado.

Esta función se obtiene empleando la señal transmitida por el radar y los ecos recibidos generando un espacio bidimensional en el cual sus dimensiones están representadas por el tiempo y la frecuencia [5] [9].

Si la señal trasmitida modulada esta definida por

$$S_T[n] = U_T[n]e^{-j\omega_c n} \tag{1.21}$$

La señal recibida tendrá la forma

$$S_R[n] = U_R[n - n_d]e^{-j(\omega_c - \omega_d)(n - n_d)} \tag{1.22}$$

Reescribiendo la exponencial

$$S_R[n] = U_R[n - n_d]e^{j\omega_d(n - n_d)}e^{-j\omega_c(n - n_d)} \tag{1.23}$$

Realizando el proceso de demodulación $S_R[n]$

$$S_R[n] = U_R[n - n_d]e^{j\omega_d(n - n_d)} \tag{1.24}$$

Ahora se puede encontrar n_d realizando la correlación entre la señal $S_R[n]$ y $S_T[n]$ para cuando no existe cambio de frecuencia (frecuencia doppler ω_d=0).

$$C[\tau] = \sum_{n=-\infty}^{\infty} U_T[n]U_R^*[n - n_d + \tau] \tag{1.25}$$

La magnitud de la correlación será máxima cuando τ = n_d

Ahora si existe ω_d

$$C[\tau] = \sum_{n=-\infty}^{\infty} U_T[n]U_R[n - n_d + \tau]e^{j\omega_d[n - n_d + \tau]} \tag{1.26}$$

Como ecuación (1.26) es una expresión compleja, por lo tanto la ecuación se modifica así:

$$A[\tau,\omega]=\sum_{n=-\infty}^{\infty}U_T[n]U_R[n-n_d+\tau]e^{j\omega_d[n-n_d+\tau]}e^{-j\omega n} \quad (1.27)$$

Separando el primer término de la exponencial y agrupándolo con el segundo

$$A[\tau,\omega]=e^{j\omega_d[\tau-n_d]}\sum_{n=-\infty}^{\infty}U_T[n]U_R[n-n_d+\tau]e^{jn[\omega_d-\omega]} \quad (1.28)$$

De esta forma si $\omega=\omega_d$ la magnitud de $A[\tau,\omega]$ es máxima con $\tau = n_d$.

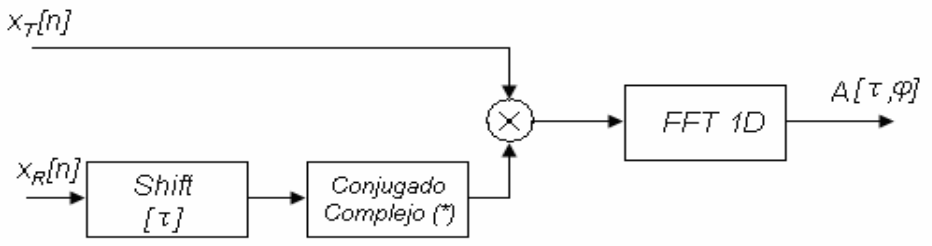
La función de Ambigüedad permite determinar el tiempo de atraso n_d y el cambio de frecuencia de la señal recibida ω_d, encontrando τ y ω que hagan la magnitud de $A[\tau,\omega]$ un máximo.

Para una secuencia de N muestras y aproximando ω a $\frac{2\pi\varphi}{N}$ la función de ambigüedad queda definida como:

$$A_{\{S_T,S_R\}}[\tau,\varphi]=\sum_{n=0}^{N-1}S_T[n]S_R^*[n+\tau]e^{\frac{-j2\pi\varphi n}{N}} \quad (1.29)$$

La función de ambigüedad se usa como respuesta al impulso del sistema, la cual se implementó en este proyecto de la siguiente manera.

Figura 11. Implementación de la función de ambigüedad



Fuente: autores del proyecto

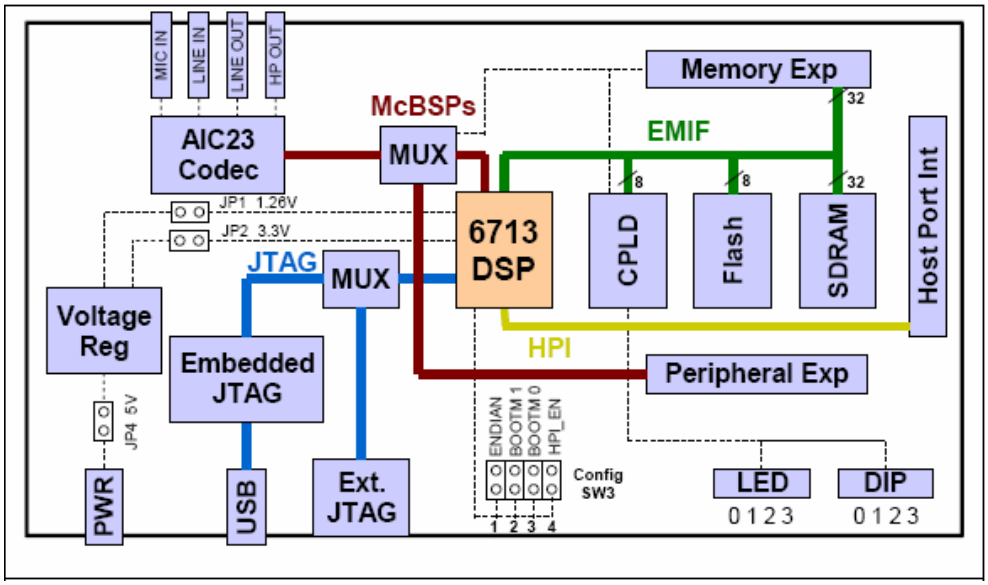
2. DESCRIPCIÓN DE HARDWARE Y SOFTWARE

2.1. DESCRIPCIÓN HARDWARE

2.1.1. DSK TMS320C6713 El DSK TMS320C6713 es una tarjeta de desarrollo "Texas instrument inc" de bajo costo, usada para el procesamiento de señales en tiempo real y alta velocidad de transferencia de datos. Entre las características mas importantes se destacan:

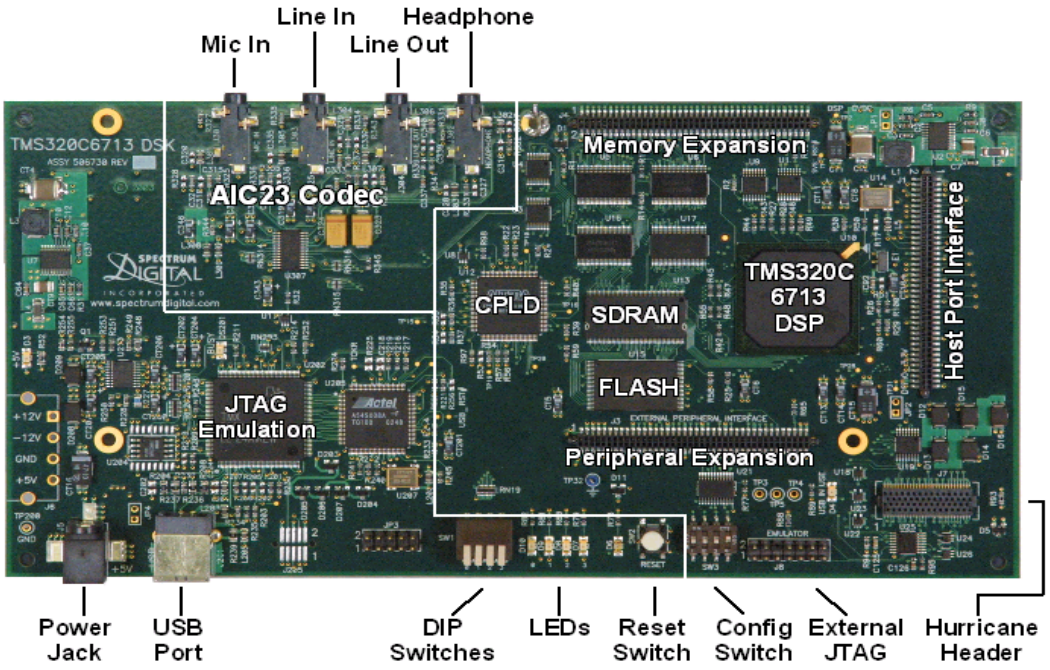
- Un DSP (Digital Signal Processing) C6713 de punto flotante con frecuencia de reloj de 225Mhz [1].
- 2 salidas y 2 entradas de audio Stereo de 32-bit TLV320AIC23 (AIC23) que funciona con un reloj de 12MHz y una frecuencia de muestreo que varia de 8 a 96KHz ajustable [1].
- Memoria SDRAM (Synchronous Dynamic Random Access Memory) de 16Mb. Además tiene la posibilidad de expandir la memoria SDRAM por medio del EMIF (External Memory Inter Face) con un bus de 32-bit para aumentar la capacidad del DSK,
- 512Kb de memoria Flash reservando 256Kb para configuración del sistema.
- CPLD (Complex Programmable Logia Device) Altera EPM3128TC100, posee 4 registros para el control interno del DSK y se encarga de manejar la lógica que controla los periféricos externos de la tarjeta.
- Un conector HPI (Host Port Int) y una interfaz de alta velocidad permite conectar y comunicar varios DSPs para agilizar el procesamiento de datos.
- Un puerto de expansión al cual se pueden conectar periféricos tales como una pantalla LCD para visualizar espectros de señales [10].

Figura 12. Diagrama de bloques del DSK



Fuente: referencia [12]

Figura 13. DSK TMS320C6713

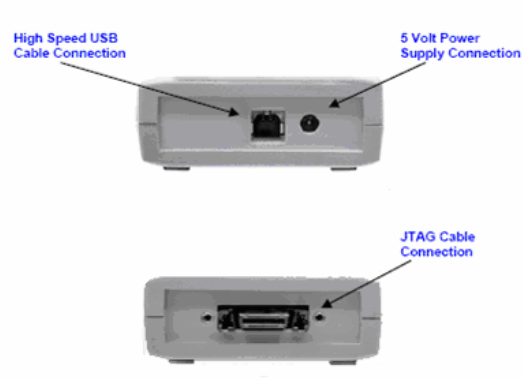


Fuente: referenciada [12]

El TMS320C6713 (C6713) está diseñado para algoritmos numéricos de gran exigencia. El interior de la memoria del programa está estructurado con una arquitectura VLIW (Very-Long-Instruction-Word) de modo que un total de ocho instrucciones se descarguen y ejecuten en cada ciclo. Por ejemplo, Con una frecuencia de reloj de 225MHz, el C6713 es capaz de ir a buscar 8 de 32 instrucciones cada 1/225MHz, lo que es equivalente a realizar un promedio de 1800 millones de operaciones en punto flotante por segundo. Esto es ideal para implementar operaciones como la FFT de matrices de datos de más de 10 millones de elementos como es el caso de los datos obtenidos por radar SAR.

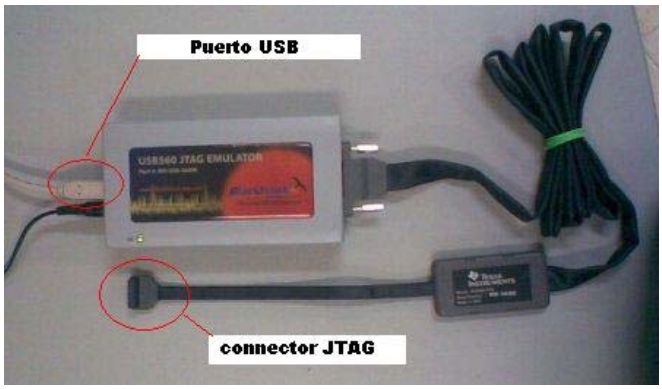
2.1.2. JTAG emulator.(Blackhawk USB560) El Blackhawk USB560 es una herramienta desarrollada para mejorar el desempeño del DSK TMS320C6713 de Texas Instrument, esto equivale a un aumento de la velocidad de transferencias de datos en tiempo real superior a 2Mb por segundo, con el uso de este emulador externo es posible habilitar la unidad HSRTDX que posee el DSP C6713. Se utiliza con un cable flexible de 5.5' high-speed especial para ser conectado al puerto externo JTAG del DSK y de este al PC por medio de un puerto USB de alta velocidad (480Mb por segundo) [15].

Figura 14. Puertos de conexión del blackhawk USB560



Fuente: EWA technologies, Inc. (www.blackhawk-dsp.com)

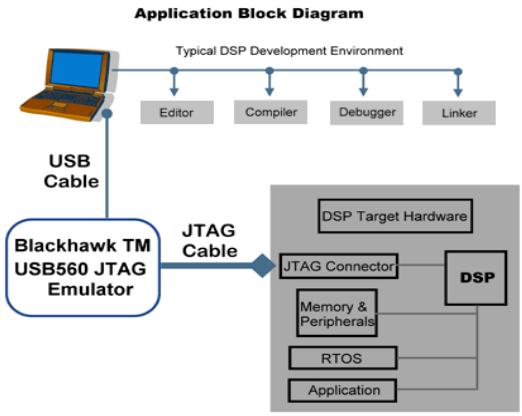
Figura 15. Conector JTAG y USB del blackhawk USB560



Fuente: autores del proyecto

En el siguiente diagrama se especifica las conexiones entre el DSK, el blackhawk USB560 y un PC para la implementación de las diferentes aplicaciones.

Figura 16 Diagrama de bloques de conexión



Fuente: EWA technologies, Inc. (www.blackhawk-dsp.com)

En la figura 17, se observa la configuración que se utilizo para las pruebas y la obtención de los resultados durante el desarrollo del proyecto.

Figura 17. Conexión implementada en laboratorio

Fuente: autores del proyecto

2.2. DESCRIPCIÓN SOFTWARE

2.2.1. Code Composer Studio IDE (CCS) El code composer studio (CCS) es una herramienta software desarrollada por Texas Instrument para diferentes aplicaciones, que se quieran implementar en el DSP Starter Kit (DSK). El CCS permite descargar a la tarjeta código desarrollado en C o en lenguaje Ensamblador. Estos códigos son compilados y depurados por medio de esta herramienta, para luego ser cargados y simulados directamente en el DSK, en la guía de usuario anexada en este trabajo se especifican las pautas y procedimiento a seguir para un correcto funcionamiento de la herramienta.

Debido a que los datos que se manejan del radar SAR son un conjunto de matrices de diferentes tamaños, es de interés que la transferencia de datos se realice en el menor tiempo posible, por esto se utilizó la herramienta RTDX (Real-

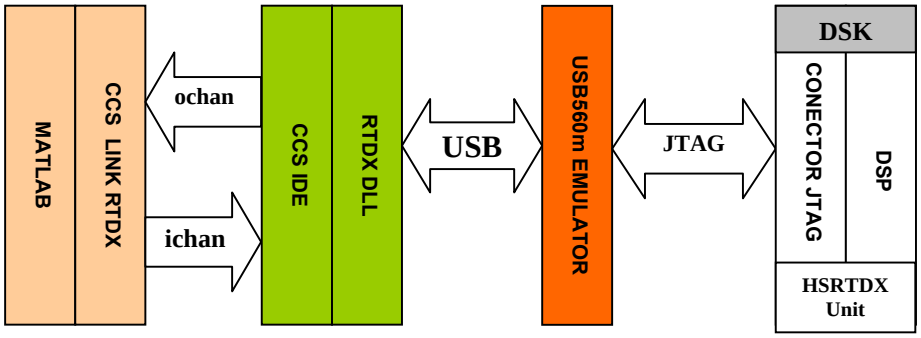
Time Data Exchange) que se puede configurar por medio del CCS. La RTDX básicamente consiste en un canal de comunicación de alta velocidad entre la tarjeta DSK y el un equipo anfitrión en este caso un PC convencional, para la recepción y envío de datos. Este intercambio de información se controla por medio de un API (Application Programming Interface) o interfaz de programación de aplicaciones, además de una serie de funciones contenidas en el DSK y configurables desde el CCS. Estos datos pueden ser almacenados o visualizados por interfaz como MatLab, National Instruments' LabVIEW™, Quinn-Curtis' Real-Time Graphics Tools o Microsoft Excel. Además de esto es posible utilizar un canal de comunicación aun más rápido denominado HSRTDX (High-Speed RTDX), pues se cuenta con una unidad física dentro del DSP C6713 que proporciona un control de interrupciones DMA (Direct memory access) entre el EMU (emulador USB560) y la memoria de la tarjeta. La ventaja de esta configuración es la rapidez de transmisión de los datos (alrededor de 2343KBits/seg) en comparación con trabajos previos. A continuación se describen las funciones de RTDX utilizadas:

RTDX_CreateInputChannel y RTDX_CreateOutputChanne: con estas funciones se declaran los canales de entrada y salida de datos respectivamente de la RTDX, cuando el canal ha sido creado se encuentra desactivado por defecto, por lo tanto debe ser habilitado, esto se puede llevar acabo desde MatLab.

RTDX_read: si el canal de entrada está habilitado esta función se encarga de recibir los datos y almacenarlos en una variable específica, la cual ha sido declarada previamente con un formato y tamaño adecuado para la recepción de los datos.

RTDX_write: si el canal de salida está habilitado la función se encarga de enviar los datos almacenándolos en una variable específica, la cual ha sido creada con un formato (doble-precisión) y un tamaño determinado.

Figura 18. Diagrama de bloques del funcionamiento Hardware y Software



Fuente: autores del proyecto

2.2.2. MATLAB

2.2.2.1 CCSLINK Debido a que la manipulación de los datos y visualización de los mismos se hace desde MatLab, fue necesario el uso de esta herramienta. Básicamente es un enlace que permite la comunicación del CCS IDE y MatLab a través de la manipulación de objetos. Mediante este enlace es posible utilizar funciones de MatLab para acceder a información contenida en el CCS IDE. Entre las funciones utilizas para este propósito están:

CCSBOARDINFO: esta función ejecutada desde MatLab inicialmente se encarga de asegurarse que existe un dispositivo configurado (en este caso el Blackhawk USB560m y el DSK) y previamente reconocido por le CCS IDE mostrando información acerca del tipo de procesador o tarjeta utilizada.

Board Num	Board Name	Proc Num	Processor Name	Processor Type
0	Blackhawk USB560m - C6713 DSK ...	0	TMS320C6710_0	TMS320C6710

CCSDSP: esta función se encarga de crear el objeto para establecer la comunicación entre MatLab y la tarjeta, por tanto es necesario que la tarjeta se encuentre previamente conectada y configurada.

RESET: una vez creado el objeto es posible utilizar funciones de MatLab directamente sobre la tarjeta, el reset se utiliza para detener cualquier aplicación que pueda estar corriendo en el DSK. El reset se bebe realizar al objeto previamente creado por ccsdsp. Por ejemplo si cc=ccsdsp entones el objeto es cc y el *reset (cc).*

VISIBLE: esta función muestra la ventana principal del CCS IDE en su PC o la mantiene invisible cuando MatLab realiza la comunicación con la tarjeta; sin embargo puede ser más útil visualizar la ventana principal del CCS IDE para monitorear o hacer cambios en la configuración de los dispositivos.

ENABLE: esta función se encarga de habilitar uno o más canales para la operación de RTDX, el objeto debe haberse creado previamente para poder habilitar el canal.

ISENABLE: esta función simplemente comprueba que el canal fue habilitado o no, devolviendo un 1 o 0 respectivamente, puede ser útil para detectar errores si el canal no está habilitado.

SET: con esta función es posible establecer un tiempo de espera de datos antes de deshabilitar el canal, si ocurre un problema con la tarjeta o el CCS IDE el programa en MatLab envía un mensaje de error si el tiempo programado ya transcurrió.

OPEN: esta función es utilizada para cargar un proyecto (file.pjt) en el CCS IDE. Este proyecto fue creado y compilado previamente para que se pueda ejecutar desde MatLab. Es conveniente que el archivo *file.m* creado en MatLab esté en el directorio de trabajo o de lo contrario se debe redireccionar cuando se ejecute el programa. Además está función crea y habilita los canales de escritura (w) y lectura (r) de datos para MatLab con la siguiente sintaxis:
open(cc.rtdx,'ichan','w'); open(cc.rtdx,'ochan','r');

LOAD: una vez abierto el proyecto (file.pjt) en el CCS IDE esta función cargar el archivo *file.out* directamente en la tarjeta (DSK), el cual fue generado después de compilar el proyecto si errores en el CCS IDE, está compilación se hace previamente para depurar el programa de cualquier error que pueda presentar.

RUN: con el archivo *file.out* cargado en la tarjeta el siguiente paso es ejecutar el programa en el DSK, que es precisamente lo que realiza está función.

CONFIGURE: es posible configurar las dimensiones del buffer (*size*) y el número de canales de la RTDX, esto se hace en base a los datos o mensajes que se van a transmitir. En este caso se tiene en cuenta el tamaño de la matriz que se quería enviar y recibir, por ejemplo para la matriz de 256x256 se habilita un buffer de 1024 bytes que es el tamaño mas pequeño que se puede configurar, de esta forma se envían mensajes desde MatLab que corresponden a cada fila de la matriz con size=1024 bytes con formato de doble precisión (double). Para está aplicación se configuraron siempre dos canales uno de lectura de datos (ochan) y otro de escritura de datos (ichan).

WRITEMSG: una vez es configurado y habilitado un canal de comunicación, es posible enviar mensajes de MatLab al DSK que en realidad son las filas de la

matriz en formato de doble precisión, los mensajes son enviados por el canal *ichan* que es la entrada al DSK.

READMSG esta función se encarga de recibir los mensajes o datos procesados por el DSK por medio del canal *ochan.* Este es el canal de salida del DSK y los datos son recibidos por MatLab para su posterior almacenamiento y visualización.

DISABLE Y CLOSE una vez terminada la transferencia y recepción de datos es necesario deshabilitar la función RTDX con la función *disable* se deben cerrar. Los respectivos canales *inchan* y *ochan* con la función *close.*

Las anteriores funciones fueron descritas según su implementación en el código de MatLab. En total la versión 7.4 de MatLab cuenta con 67 funciones para el link del CCS, dividida en cuatro grupos:

- Operations on, Links for CCS IDE: configuration del link CCS IDE.
- Operations on Links for RTDX: configuración del link con la herramienta RTDX.
- Manipulate Data: manejo de datos en la tarjeta (DSK) desde MatLab.
- Hardware-in-the-Loop Processing: simulación con el DSK.

3. RESULTADOS DE LA IMPLEMENTACIÓN

A continuación se presentan los resultados de la implementación del hardware y software para el procesamiento de imágenes SAR. Los resultados consisten en los tiempos de ejecución de los diferentes algoritmos.

Es indispensable resaltar que el computador utilizado para la toma de tiempos y la ejecución de los algoritmos posee un procesador de Intel 4 de 2.5GHz con memoria RAM de 512MB.

3.1. RESULTADOS DE LA FUNCIÓN DE AMBIGÜEDAD

Los tiempos de cómputo de los algoritmos se hicieron en base de los diferentes tamaños de señales que fueron aportados por la Universidad de Puerto Rico, en este caso matrices. Debido al tamaño de las señales que se manejan, es necesario almacenar las matrices de datos en una memoria externa, y en este caso son almacenados en un computador para luego ser enviadas a través del emulador al DSK y luego retornadas al PC. La tabla 3 presenta los tiempos de ejecución del algoritmo y transferencia de datos para 5 diferentes longitudes.

Tabla 3. Tiempos de cómputo para la función de ambigüedad.

Longitud de las señales	Transferencia de datos por fila [Seg.]	Tiempo de procesado por fila [Seg.]	Tiempo por fila [Seg.]	Tiempo total [Seg.]
256x256	0,31727	0.02773	0.3450	88.32
512x512	0,32968	0.06012	0.3898	199.57
1024x1024	0.4043	0.1300	0.5343	547.12
2048x2048	0.7061	0.2795	0.9856	2018.51
4096x4096	1.58	0.6098	2.19	8970.24

Fuente: autores del proyecto

A continuación se muestran los tiempos obtenidos previamente en otros trabajos utilizando la misma tarjeta de desarrollo [8] (ver tabla 4), los cuales fueron tomados bajo un algoritmo similar pero sin ningún tipo de emulador.

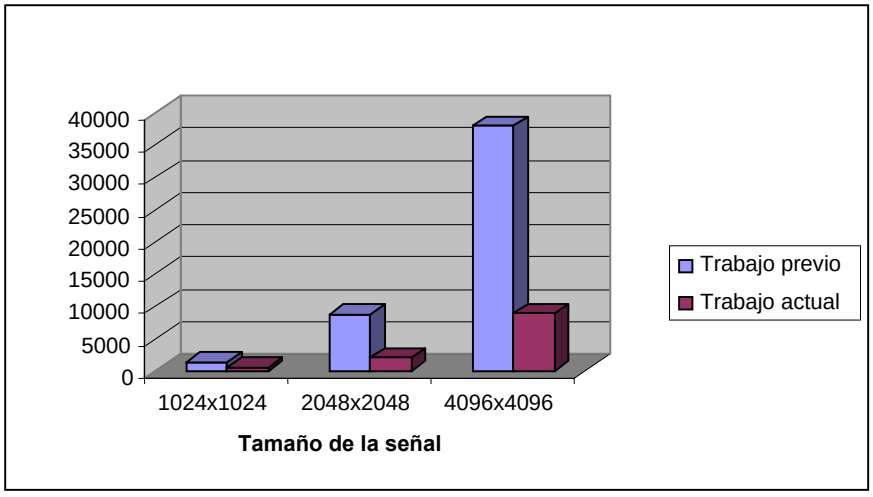
Tabla 4. Tiempos de cómputo para la función de ambigüedad trabajo previo

Length Signals	Transfer Data Times secs.	Processing Time secs	Total Computation Time secs
32x32	N7A	0.057	0.057
64x64	N/A	0.294	0.294
128x128	N/A	1.307	1.307
256x256	N/A	5.762	5.762
512x512	N/A	25.145	25.145
1024x1024	1245.44	112.320	1357.76
2048x2048	8277.39	481.890	8759.28
4096x4096	35832.40	2061.520	37893.90

Fuente: referencia [8].

Realizando una comparación entre las tablas 3 y 4 se tiene la siguiente figura.

Figura 19. Comparación de tiempos de cómputo de la función de ambigüedad



Fuente: autores del proyecto

Cabe notar que el aumento de tiempo en el procesado se debe a que la implementación de la FFT en éste trabajo fue realizado en código C, mientras que en el trabajo previo ésta fue implementada en código ensamblador.

3.2. RESULTADOS DE LA FFT 2D

Teniendo almacenada la matriz de señales en el PC se envía columna por columna al DSP, en donde se le realiza la FFT unidimensional, luego son enviadas y almacenadas en el PC, una vez terminado el proceso para esta primera matriz esta misma se envía fila por fila y se realiza nuevamente la FFT unidimensional, y se retorna el resultado al PC (ver numeral 1.4.).

Los tiempos obtenidos para la FFT 2D se presentan en la siguiente tabla, además, también se presentaran los resultados obtenidos en el trabajo previo.

Tabla 5. Tiempos de cómputo para la FFT 2D

Longitud de las señales	Tiempo columnas [Seg.]	Tiempo por filas [Seg.]	Tiempo total [Seg.]
256x256	121.36	63.87	185.23
512x512	308.48	173.62	482.10
1024x1024	720.99	426.80	1147.79
2048x2048	1975.29	1312.15	3287.44

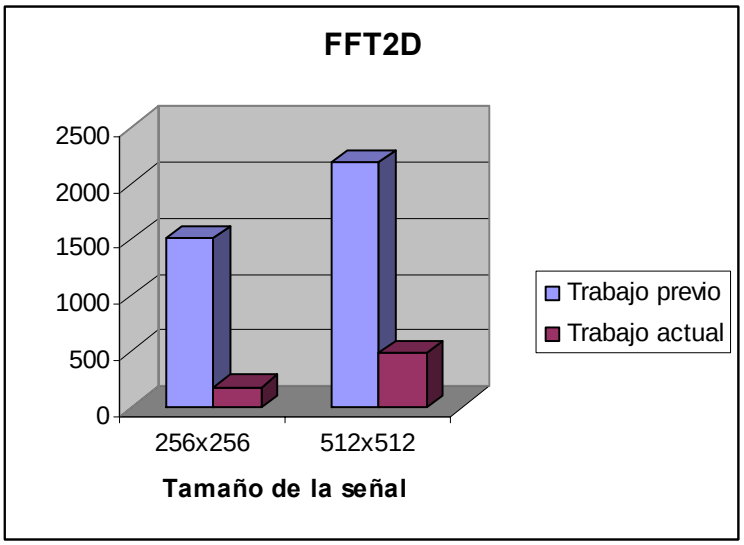
Fuente: autores del proyecto

Tabla 6. Tiempos de cómputo de la FFT 2D trabajo previo

Length Signals	2D FFT Secs.
16x16	66
32x32	145
64x64	290
128x128	600
256x256	1510
512x512	2186

Fuente: referencia [8]

Figura 20. Comparación de tiempos para la FFT 2D



Fuente: autores del proyecto

3.3. RESULTADOS DE LA CONVOLUCIÓN EN 2D

La convolución en dos dimensiones fue realizada con el método indirecto presentado en el capitulo 1, para la ejecución de esta función se es necesario la aplicación de FFT 2D a las matrices a convolucionar, una vez se tengan las matrices en el dominio de la frecuencia en la memoria del PC, se envía una columna de una matriz junto con una columna de la otra matriz al DSP para ser multiplicados elemento a elemento y a este resultado realizarle la IFFT unidimensional, para luego ser enviados y almacenados en el PC, terminado el proceso de multiplicación y transformación para todas las columnas de las matrices, se procede a enviar nuevamente al DSP la matriz de resultado fila por fila donde se les aplicara la IFFT unidimensional para luego ser retornadas al PC.
A continuación se presentan los resultados de este proceso para diferentes longitudes de señales.

Tabla 7. Tiempos de cómputo de la convolución en 2D

Longitud de las señales	Tiempo columnas [Seg.]	Tiempo por filas [Seg.]	Tiempo total [Seg.]
256x256	119.93	94.54	214.47
512x512	313.24	284.26	561.51
1024x1024	1517.16	600.47	2117.63
2048x2048	2053.32	1734.45	3787.77

Fuente: autores del proyecto

Para los tiempos de cómputo de la convolución no se tiene registros de trabajos previos para realizar una comparación, sin embargo se estima que estos resultados fueron menores en proporciones similares a los de la FFT 2-D.

El método utilizando en los algoritmo para el envió y la recepción de datos, ayudó a la eliminación de procesos en matlab así mismo contribuyendo con el mejoramiento de los tiempos de cómputo, como fue en el caso de la FFT 2-D el cual se evito realizar una transpuesta, además se utilizó a matlab exclusivamente para comunicación, envío y almacenamiento de datos.

4. CONCLUSIONES

Se demostró la ventaja de la utilización del BLACKHAWK USB560 en la transferencia de datos entre el DSP TMS320C6713 y un PC, al contrastarse con trabajos previos, donde se utiliza el DSK para el procesado de datos y manejo de los canales de transmisión. En este trabajo se evidenciaron las ventajas de la implementación de algoritmos usando esta herramienta, lo cual reduce considerablemente los tiempos de transferencia de datos alrededor del 80% para tamaños de 2048x2048 y 4096x4096 en la función de ambigüedad , con lo cual se logra un acercamiento significativo al procesamiento en tiempo real de información obtenida por un radar tipo SAR.

Se implementaron algoritmos tanto en MatLab como en el CCS utilizados para la transferencia de datos y para el procesado, estos algoritmos consisten básicamente de tres etapas, envío de datos al DSP, procesamiento de los datos en el DSP y envío de datos del DSP al PC. La Función propia de Texas Instruments, Inc., la FFT fue utilizada para obtener el mayor rendimiento del algoritmo.

Se implementó la convolución en dos dimensiones y la FFT2D para los diferentes tamaños de señales utilizadas, obteniendo los tiempos de cómputo total. A si mismo se evidenciaron importantes descensos en el tiempo total de cómputo que eran mas significativos a medida que se hacían mas grandes los tamaños de los datos procesados.

La ventaja de utilizar el DSK TMS320C6713 se evidenció en este tipo de aplicaciones, el sistema esta diseñado de forma que el procesado de la

información sea eficiente debido a la estructuración de la memoria del DSP y a los algoritmos optimizados para estas aplicaciones.

Se depuró el manual del usuario anexado al final de este trabajo. En este manual se incluye un procedimiento para una correcta conexión y manejo del Blackhawk USB560. Estos cambios y mejoras fueron realizados en base de la experiencia de los procedimientos realizados y en trabajos previos.

5. RECOMENDACIONES

Es importante tener en cuenta, las limitaciones que se presentaron a nivel de hardware, concretamente en cuánto al PC que se utiliza para la comunicación del CCS IDE con el DSP. Es conveniente contar con una PC que tenga un procesador superior a 3GHz de frecuencia de reloj y memoria RAM superior a 1GB. A pesar de que las operaciones críticas como la FFT no dependen del PC sino del DSP, la transferencia de los datos y la memoria suficiente para el almacenado de los mismos se ve influenciado por las características de este dispositivo.

Como se mencionó en el capitulo 3, MatLab no es la única plataforma para la comunicación con el CCS IDE y el DSK, existen por lo menos otras 3 herramienta software que permiten canales de comunicación directo con el DSP. Un trabajo futuro podría utilizar aplicaciones desarrollados en estos ambientes de programación (National Instruments' LabVIEW™, Quinn-Curtis' Real-Time Graphics Tools o Microsoft Excel) y desarrollar guías de usuarios similares a la depurada en este trabajo.

REFERENCIAS BIBLIOGRÀFICAS

[1] Cooley, J. W., and Tukey, J. W. An Algorithm for the Machine Calculation of Complex Fourier Series.Math. of Computation 19 (1965), pags. 297-301.

[2] Duhamel, P. & Vetterli, M. 1990 Fast Fourier Transforms: a tutorial review and a state of the art. Signal Process. 19, 259–299. 1984.

[3] Franceschetti,G and others. An Electromagnetic Model for SAR Raw Signal Simulation of Urban Areas. *IEEE/ISPRS Joint Workshop in Remote Sensing and Data Fusion over Urban Areas.* pags.10-14. Nov. 2001.

[4] Matusiak Robert. Implementing fast Fourier transforms algorithms of real-valued sequences with the TMS320 DSP platform. Digital Signal Processing Solutions.2002. 75 p. Texas Instruments Incorporated.

[5] Nava Valles Hilaura Raquel. Modeling and simulation of point spread functions for advanced SAR systems. Mayagüez. 2004.107 p. Trabajo de maestría. Universidad de Puerto Rico. Departamento de ingeniería eléctrica y computadores.

[6] Platónov Alexéi. Aplicación de imágenes de satélite SAR en los estudios de contaminación marina y de dinámica de las aguas en el mediterráneo noroccidental. Barcelona. Junio de 2002.144 p. Obtención del grado de Doctor (oceanógrafo). Universidad Politécnica de Cataluña, Departamento de Ingeniería Hidráulica, Marítima y Ambiental.

[7] RADARSAT International Data Products Specifications, reporte escrito por radarsat international y recibido por RSI. Canada May 8, 2000.

[8] Ramírez Silva Ana B. Implementación de representaciones tempo-frecuencia sobre estructuras computacionales hardware/software para aplicaciones SAR. Mayagüez. Junio 2006. Trabajo de maestría. Universidad de Puerto Rico, departamento de ingeniería eléctrica y computadores.

[9] Ramírez Ana B., Rivera Ivan J., Rodriguez Domingo. SAR image processing algorithms based on the ambiguity function. Circuits and Systems 48th Midwest Symposium on. 7-10 Aug 2005.

[10] Rodríguez Domingo, Rueda Dilia, Nava Hilaura, Quinchanegua Alberto, High Performance Artificial SAR Raw Data Generation Algorithms for Remote-sensed Imaging Applications, Mayagüez, 3 p, Universidad de Puerto Rico. Departamento de ingeniería eléctrica y computadores.

[11] Texas Instruments, Implementing Fast Fourier Transform Algorithms of Real-Valued Sequences With the TMS320 DSP Platform, Application Report, August 2001.

[12] Texas Instruments, TMS320C6713 DSK Technical Reference, Spectrum Digital, INC, mayo 2003.

[13] Texas Instruments,TMS320C6713, TMS320C6713B, Floatting-Point Digital Signal Processors, Houston 2004.

[14] Texas Instruments. Code Composer Studio IDE, version 3.1.0, 2005.

[15] EWA Technologies, Inc. Blackhwak USB560 JTAG Emulator, 2007.

ANEXOS

ANEXOS A
CÓDIGOS IMPLEMENTADOS EN CCS Y MATLAB

Algoritmo de la función de ambigüedad en CCS para longitud de señales de 256 datos

```
//Recibe y envía datos de MATLAB para calculo de la función de ambigüedad para
//longitud 256.
#include <rtdx.h>
#include "target.h"
#include "math.h"
#include "Sgtx256.h"
#include "dataTW256.h"
RTDX_CreateInputChannel(ichan);
RTDX_CreateOutputChannel(ochan);
far double Srgrx[512];
far double Srgrxs[512];
short N=256;

void Shift_Signal (double Srx[],int m)
{
int n,i;
    for (n=0; n<N;n+=1)
        {
          i=(n+m)%(N);
         Srgrxs[2*n]= (Srx[2*i]);
         Srgrxs[2*n+1]= (Srx[2*i+1]);
         }
}
```

```
void Haddamart (double Stx[])
{
int n;
    for (n=0; n<N;n+=1)
        {
         Srgrxs[2*n]= (Srgrxs[2*n]*Stx[n]);
         Srgrxs[2*n+1]= (Srgrxs[2*n+1]*Stx[n]);
        }
}

void FFT_TI (void)
{
  DSPF_sp_cfftr2_dit((double*) Srgrxs,(float*) w, N);
  bit_rev((double *)Srgrxs, (2*N)>>1);
}

void main(void)
{
 int k;
     TARGET_INITIALIZE();
     for(k=0;k<256;k++)
       {
        while(!RTDX_isInputEnabled(&ichan))
        puts("\n\n Waiting to read ");
        RTDX_read(&ichan,Srgrx,sizeof(Srgrx));
        puts("\n\n Read Completed 2");

         Shift_Signal (Srgrx,k);
         Haddamart (Sigtx);
```

```
        FFT_TI();

        while(!RTDX_isOutputEnabled(&ochan))
        puts("\n\n Waiting to write ");
        while(RTDX_writing);
        while(!RTDX_write( &ochan, Srgrxs, sizeof(Srgrxs)));
        puts("\n\n Write Completed");
         }
}
```

Algoritmo de la función de ambigüedad en MATLAB para longitud de señales de 256 datos

```
%Transfiere datos desde MATLAB->DSK y DSK->MATLAB
load('256_256_imaging.mat');
Re=real(data);
Im=imag(data);
for i=1:256
   for j=1:256
      datar(i,2*j-1)=Re(i,j);
      datar(i,2*j)=Im(i,j);
   end
end
datar=double(datar);
ccsboardinfo
cc = ccsdsp('boardnum',0);
reset(cc)
visible(cc,1);
enable(cc.rtdx);
```

```
cc.rtdx.set('timeout', 20);
open(cc,'rtdx_matlab_sim.pjt');
load(cc,'./debug/rtdx_matlab_sim.out');
run(cc);
configure(cc.rtdx,2048,2);
open(cc.rtdx,'ichan','w');
open(cc.rtdx,'ochan','r');

for m=1:256
   enable(cc.rtdx,'ichan');
   writemsg(cc.rtdx,'ichan', double(datar(m,:)))

   enable(cc.rtdx,'ochan');
   outdata(m,:) = readmsg(cc.rtdx,'ochan','double');
end
disable(cc.rtdx);
close(cc.rtdx,'ichan');
close(cc.rtdx,'ochan');
```

Algoritmo para la implementación de transformada de fourier en dos dimensiones en CCS para tamaño de 256x256.

```
//Calcula la FFT 2D de matrices de 256x256.
#include <rtdx.h>
#include "target.h"
#include "math.h"
#include "dataTW256.h"
RTDX_CreateInputChannel(ichan);
```

```
RTDX_CreateInputChannel(ichan2);
RTDX_CreateInputChannel(ichan3);
RTDX_CreateOutputChannel(ochan);
RTDX_CreateOutputChannel(ochan2);
RTDX_CreateOutputChannel(ochan3);
double buffer[256];
double buffer2[256];
double buffer3[256];
far double Srgrxs[512];
far double Srgrxs1[256];
far double Srgrxs2[256];
short N=256;

void FFT_TI (void)
{
   DSPF_sp_cfftr2_dit((double*) Srgrxs,(float*) w, N);
   bit_rev((double *)Srgrxs, (2*N)>>1);
}

void main(void)
{
  int k,i,j;

 TARGET_INITIALIZE();

  for(k=0;k<256;k++)
     {
      while(!RTDX_isInputEnabled(&ichan))
      puts("\n\n Waiting to read ");
```

```
    RTDX_read(&ichan,Srgrxs,sizeof(Srgrxs));
    puts("\n\n Read Completed 1");

    FFT_TI();                                          .

    while(!RTDX_isOutputEnabled(&ochan))
    puts("\n\n Waiting to write ");
    while(RTDX_writing);
    while(!RTDX_write( &ochan, Srgrxs, sizeof(Srgrxs)));
    puts("\n\n Write Completed ");
   }

for(i=0;i<256;i++)
   {
    while(!RTDX_isInputEnabled(&ichan2))
    puts("\n\n Waiting to read ");
    RTDX_read(&ichan2,buffer2,sizeof(buffer2));
    puts("\n\n Read Completed 3");

    while(!RTDX_isInputEnabled(&ichan3))
    puts("\n\n Waiting to read ");
    RTDX_read(&ichan3,buffer3,sizeof(buffer3));
    puts("\n\n Read Completed 3");

    for(j=0;j<256;j++)
       {
        Srgrxs[2*j]=buffer2[j];
        Srgrxs[2*j+1]=buffer3[j];
       }
```

```
      FFT_TI();

      for(j=0;j<256;j++)
         {
          Srgrxs1[j]=Srgrxs[2*j];
          Srgrxs2[j]=Srgrxs[2*j+1];
         }

      while(!RTDX_isOutputEnabled(&ochan2))
      puts("\n\n Waiting to write ");
      while(RTDX_writing);
      while(!RTDX_write( &ochan2, Srgrxs1, sizeof(Srgrxs1)));
      puts("\n\n Write Completed");

      while(!RTDX_isOutputEnabled(&ochan3))
      puts("\n\n Waiting to write ");
      while(RTDX_writing);
      while(!RTDX_write( &ochan3, Srgrxs2, sizeof(Srgrxs2)));
      puts("\n\n Write Completed");
     }
}
```

Algoritmo de la FFT 2D en MATLAB para matrices 256x256.

```
%Transfiere datos desde MATLAB->DSK y DSK->MATLAB
load('256_256_imaging.mat');
Re=real(data);
Im=imag(data);
```

```
for i=1:256
   for j=1:256
      datar(i,2*j-1)=Re(i,j);
      datar(i,2*j)=Im(i,j);
   end
end
ccsboardinfo
cc = ccsdsp('boardnum',0);
reset(cc)
visible(cc,1);
enable(cc.rtdx);
cc.rtdx.set('timeout', 20);
open(cc,'rtdx_matlab_sim.pjt');
load(cc,'./debug/rtdx_matlab_sim.out');
run(cc);
configure(cc.rtdx,2048,5);
open(cc.rtdx,'ichan','w');
open(cc.rtdx,'ochan','r');
%%%%%%fft para filas
for m=1:256
   enable(cc.rtdx,'ichan');
   writemsg(cc.rtdx,'ichan', double(datar(m,:)))

   enable(cc.rtdx,'ochan');
   outdata(m,:)=readmsg(cc.rtdx,'ochan','double');
end
close(cc.rtdx,'ichan');
close(cc.rtdx,'ochan');
%%%%%%fft para columnas
```

```
open(cc.rtdx,'ichan2','w');
open(cc.rtdx,'ichan3','w');
open(cc.rtdx,'ochan2','r');
open(cc.rtdx,'ochan3','r');
for i=1:256
   enable(cc.rtdx,'ichan2');
   writemsg(cc.rtdx,'ichan2', double(outdata(:,2*i-1)))

   enable(cc.rtdx,'ichan3');
   writemsg(cc.rtdx,'ichan3', double(outdata(:,2*i)))

   enable(cc.rtdx,'ochan2');
   outdata2(:,2*i-1)=readmsg(cc.rtdx,'ochan2','double');

   enable(cc.rtdx,'ochan3');
   outdata2(:,2*i)=readmsg(cc.rtdx,'ochan3','double');
end
disable(cc.rtdx);
close(cc.rtdx,'ichan3');
close(cc.rtdx,'ochan3');
close(cc.rtdx,'ochan2');
```

Algoritmo para la implementación de la convolución en dos dimensiones en CCS para tamaños de matrices de 256x256.

```
//Calcula la convolución  2D método indirecto de matrices de 256x256.
#include <rtdx.h>
#include "target.h"
#include "math.h"
```

```
#include "dataTW256.h"

RTDX_CreateInputChannel(ichan);
RTDX_CreateOutputChannel(ochan);
RTDX_CreateInputChannel(ichan2);
RTDX_CreateOutputChannel(ochan2);
RTDX_CreateInputChannel(ichan3);
RTDX_CreateInputChannel(ichan4);
RTDX_CreateOutputChannel(ochan3);
double buffer3[256];
double buffer4[256];
far double Srgrx[512];
far double Srgrx1[512];
far double Srgrx2[512];
far double Srgrxs[512];
far double Srgrxs1[256];
far double Srgrxs2[256];
short N=256;


void PRODUCTO (void)
{
int i;
  for(i=0;i<256;i++)
     {
          Srgrxs[2*i]= Srgrx1[2*i] * Srgrx2[2*i]- Srgrx1[2*i+1] * Srgrx2[2*i+1];
          Srgrxs[2*i+1]= Srgrx1[2*i] * Srgrx2[2*i+1]+ Srgrx1[2*i+1] * Srgrx2[2*i];
     }
}
```

```
void IFFT_TI (void)
{
int i;
   In_DSPF_sp_cfftr2_dit((double*) Srgrxs,(float*) w, N);
   bit_rev((double *)Srgrxs, (2 * N)>>1);
   for(i=0;i<512;i++)
     {
           Srgrxs[i]= Srgrxs[i]/N;
     }
}

void main(void)
{
 int k,i,j;

 TARGET_INITIALIZE();

 for(k=0;k<256;k++)
   {
    while(!RTDX_isInputEnabled(&ichan))
     puts("\n\n Waiting to read ");
         RTDX_read(&ichan,Srgrx1,sizeof(Srgrx1));
         puts("\n\n Read Completed 1");

         while(!RTDX_isInputEnabled(&ichan2))
         puts("\n\n Waiting to read ");
         RTDX_read(&ichan2,Srgrx2,sizeof(Srgrx2));
         puts("\n\n Read Completed 2");
```

```
        PRODUCTO();
        IFFT_TI();

        while(!RTDX_isOutputEnabled(&ochan))
        puts("\n\n Waiting to write ");
        while(RTDX_writing);
        while(!RTDX_write( &ochan, Srgrxs, sizeof(Srgrxs)));
        puts("\n\n Write Completed ");
  }


for(i=0;i<256;i++)
  {
        while(!RTDX_isInputEnabled(&ichan3))
        puts("\n\n Waiting to read ");
        RTDX_read(&ichan3,buffer3,sizeof(buffer3));
        puts("\n\n Read Completed 3");

        while(!RTDX_isInputEnabled(&ichan4))
        puts("\n\n Waiting to read ");
        RTDX_read(&ichan4,buffer4,sizeof(buffer4));
        puts("\n\n Read Completed 3");

   for(j=0;j<256;j++)
     {
      Srgrxs[2*j]=buffer3[j];
      Srgrxs[2*j+1]=buffer4[j];
     }
```

```
        IFFT_TI();

        for(j=0;j<256;j++)
      {
       Srgrxs1[j]=Srgrxs[2*j];
       Srgrxs2[j]=Srgrxs[2*j+1];
      }

        while(!RTDX_isOutputEnabled(&ochan2))
       //for MATLAB to enable RTDX
        puts("\n\n Waiting to write ");
        while(RTDX_writing);
        while(!RTDX_write( &ochan2, Srgrxs1, sizeof(Srgrxs1)));
        puts("\n\n Write Completed");

        while(!RTDX_isOutputEnabled(&ochan3))
       //for MATLAB to enable RTDX
        puts("\n\n Waiting to write ");
        while(RTDX_writing);
        while(!RTDX_write( &ochan3, Srgrxs2, sizeof(Srgrxs2)));
        puts("\n\n Write Completed");
   }
}
```

Algoritmo de convolución 2D en MATLAB para matrices 256x256.

```
%Transfiere datos desde MATLAB->DSK y DSK->MATLAB
load('256_256_imaging.mat');
```

```
Re=real(data);
Im=imag(data);
for i=1:256
   for j=1:256
      datar(i,2*j-1)=Re(i,j);
      datar(i,2*j)=Im(i,j);
   end
end
load('Resp_impulso.mat');
Re=real(data);
Im=imag(data);
for i=1:256
   for j=1:256
      datar2(i,2*j-1)=Re(i,j);
      datar2(i,2*j)=Im(i,j);
   end
end
ccsboardinfo
cc = ccsdsp('boardnum',0);
reset(cc)
visible(cc,1);
enable(cc.rtdx);
cc.rtdx.set('timeout', 20);
open(cc,'rtdx_matlab_sim.pjt');
load(cc,'./debug/rtdx_matlab_sim.out');
run(cc);
configure(cc.rtdx,2048,6);
open(cc.rtdx,'ichan','w');
open(cc.rtdx,'ichan2','w');
```

```
open(cc.rtdx,'ochan','r');

%%%%%% proceso para filas
for m=1:256
   enable(cc.rtdx,'ichan');
   writemsg(cc.rtdx,'ichan', double(datar(m,:)))

   enable(cc.rtdx,'ichan2');
   writemsg(cc.rtdx,'ichan2', double(datar2(m,:)))

   enable(cc.rtdx,'ochan');
   outdata(m,:)=readmsg(cc.rtdx,'ochan','double');
end
close(cc.rtdx,'ichan');
close(cc.rtdx,'ichan2');
close(cc.rtdx,'ochan');

%%%%%% proceso para columnas
open(cc.rtdx,'ichan3','w');
open(cc.rtdx,'ichan4','w');
open(cc.rtdx,'ochan2','r');
open(cc.rtdx,'ochan3','r');
for i=1:256
   enable(cc.rtdx,'ichan3');
   writemsg(cc.rtdx,'ichan3',double(outdata(:,2*i-1)))

   enable(cc.rtdx,'ichan4');
   writemsg(cc.rtdx,'ichan4', double(outdata(:,2*i)))
```

```
    enable(cc.rtdx,'ochan2');
    outdata2(:,2*i-1)=readmsg(cc.rtdx,'ochan2','double');

    enable(cc.rtdx,'ochan3');
    outdata2(:,2*i) = readmsg(cc.rtdx,'ochan3','double');
end
disable(cc.rtdx);
close(cc.rtdx,'ichan3');
close(cc.rtdx,'ichan4');
close(cc.rtdx,'ochan2');
close(cc.rtdx,'ochan3');
```

Función de la FFT en una dimensión.

```
//x = señal a transformar.
//w = vector de los coeficientes W.
//n = longitud de la señal.
//La longitud del vector de los coeficientes W  debe tener n/2 de elementos de la señal a //transformar.
void DSPF_sp_cfftr2_dit(double* x, float* w, short n)
      {
         short n2, ie, ia, i, j, k, m;
         float rtemp, itemp, c, s ;

         n2 = n;
         ie = 1;

         for(k=n; k > 1; k >>= 1)
```

```
        {
          n2 >>= 1;
          ia = 0;
          for(j=0; j < ie; j++)
              {
               c = w[2*j];
               s = w[2*j+1];
               for(i=0; i < n2; i++)
                   {
                     m = ia + n2;
                     rtemp     = c* x[2*m]   + s * x[2*m+1];
                     itemp     = c* x[2*m+1] - s * x[2*m];
                     x[2*m]    = x[2*ia]   - rtemp;
                     x[2*m+1]  = x[2*ia+1] - itemp;
                     x[2*ia]   = x[2*ia]   + rtemp;
                     x[2*ia+1] = x[2*ia+1] + itemp;
                     ia++;
                    }
                ia += n2;
              }
          ie <<= 1;
        }
    }
```

Función de la IFFT en una dimensión.

```
//x = señal a transformar.
//w = vector de los coeficientes W.
```

```
//n = longitud de la señal.
//La longitud del vector de los coeficientes W  debe tener n/2 de elementos de la
señal a //transformar.

void In_DSPF_sp_cfftr2_dit(double* x, float* w, short n)
      {
         short n2, ie, ia, i, j, k, m;
         float rtemp, itemp, c, s ;

         n2 = n;
         ie = 1;

         for(k=n; k > 1; k >>= 1)
            {
             n2 >>= 1;
             ia = 0;
             for(j=0; j < ie; j++)
                {
                 c = w[2*j];
                 s = -1*w[2*j+1];
                 for(i=0; i < n2; i++)
                    {
                     m = ia + n2;
                     rtemp     = c* x[2*m]   + s * x[2*m+1];
                     itemp     = c* x[2*m+1] - s * x[2*m];
                     x[2*m]    = (x[2*ia]   - rtemp);
                     x[2*m+1]  = (x[2*ia+1] - itemp);
                     x[2*ia]   = (x[2*ia]   + rtemp);
                     x[2*ia+1] = (x[2*ia+1] + itemp);
```

```
                ia++;
               }
            ia += n2;
           }
        ie <<= 1;
       }
   }
```

Función de ordenamiento de vectores

```
//x=señal a ordenar
//n=longitud de la señal
void bit_rev(double* x, int n)
   {
     int i, j, k;
     double rtemp, itemp;
      j = 0;
      for(i=1; i < (n-1); i++)
        {
          k = n >> 1;
          while(k <= j)
                {
                j -= k;
                k >>= 1;
                 }
           j += k;
           if(i < j)
             {
```

```
            rtemp    = x[j*2];
            x[j*2]   = x[i*2];
            x[i*2]   = rtemp;
            itemp    = x[j*2+1];
            x[j*2+1] = x[i*2+1];
            x[i*2+1] = itemp;
           }
       }
}
```

ANEXO B
GUIA DE USUARIO

Este documento contiene una guía de programación básica en el desarrollo de algoritmos en lenguaje C y MatLab, para ser implementados en el DSK TMS320C6713, con la posibilidad de utilizar la herramienta “USB560 JTAG Emulator - Blackhawk”

Esta guía esta organizada del la siguiente forma:

- Introducción al TI-TMS320C6713 DSK.
- Ambiente de desarrollo para el DSP Code Compose Studio (CCS) y Matlab.
- Instalación y funcionamiento del DSK.
- Ejemplos de programación para probar las herramientas DSK
- Algunas implementaciones de algoritmos para el procesamiento de señales digitales en el TMS320C6713 DSP. Descripción con detalles de los algoritmos, requisitos especiales y notas de implementación.
- USB560 JTAG Emulator – Blackhawk.
- Instalación del USB560 JTAG Emulator.

Algunos ejemplos fueron tomados del libro “Digital Signal Processing and Applications with the C6713 and C6416” de Rulph Chassaing. También los proyectos relacionados en esta guía del usuario se encuentran en el CD anexo a esta guía.

(a)

DOCUMENTACIÓN RELACIONADA

Los siguientes libros y guías describen herramientas de apoyo relacionadas con este trabajo. Esta guía presenta algunos ejemplos tomados de estas referencias.

'Digital Signal Processing and Applications with the C6713 and C6416 DSK': Esta es la versión mas reciente del libro escrito por Rulph Chassaing. El libro tiene ejemplos del uso de DSP's con lenguaje C, Matlab, Labview, Visual Basic y Visual C++, además de poseer la descripción teórica necesaria para el desarrollo de los ejemplos.

'TMS320C67X DSP library programmer's reference guide (Texas Instruments, Inc.)': Este documento presenta una colección de funciones optimizadas para DSP's específicamente para los dispositivos TMS320C67X. Las librerías incluyen código en C (compatibles con ANSI-C) para los procesamientos matemáticos de señales y funciones de vectores.

'How to Begin Development Today With the TMS320C6713-Floating-Point DSP (Texas Instruments, Inc.)': Este documento hace un repaso de los periféricos del TMS320C6713. También presenta las herramientas de software para el DSP de la serie C6000 y las plataformas de desarrollo que pueden ser utilizadas para la creación de proyectos.

'USB560 JTAG Emulator Installation Guide': Este documento indica los pasos necesarios y recomendaciones para una correctas instalación del **USB560m,** también se puede encontrar una descripción del hardware.

‘Code Composer Studio-User’s Guide’: Esta guía explica como se debe utilizar el ambiente de desarrollo “Code Composer Studio” para crear y realizar “debug” en aplicaciones de software que requieren tiempo real.

‘Matlab Help for Link for Code Composer Studio’: Este documento presenta algunas funciones de Matlab que son útiles para establecer enlace con el “Code Composer Studio”.

TABLA DE CONTENIDO

1. INTRODUCCION AL DSK TMS320C6713

Los Procesadores de Señales Digitales (DSP, por sus siglas en ingles) son utilizados en muchas aplicaciones tales como procesamiento de imágenes, reconocimiento de voz, control, medicina, espectrografía, comunicaciones, sismología etc. Algunas de las ventajas de utilizar DSP se deben a que son menos afectadas por las condiciones ambientales, son fáciles de utilizar en comparación con otros dispositivos digitales, son flexibles y son económicos todo esto en comparación con los dispositivos análogos.

La herramienta principal para diseñar un programa de aplicación de DSP es el "Digital Starter Kit" (DSK) de Texas Instruments, Inc. Este paquete consta básicamente de un software y un hardware que son el "Code Composer Studio (CCS)" y una tarjeta de desarrollo (TMS320C6713 DSK).

Este kit de inicio es de utilidad para desarrolladores ya que permite probar el funcionamiento de los algoritmos implementados antes de la producción en serie de dispositivos para aplicaciones específicas. Además de que posee conexiones para periféricos (Audio, memoria o dispositivos que requieran conectores JTAG por ejemplo) para simular las señales de entrada y salida al procesador. Esta herramienta es compatible con los PCs y necesita una conexión de USB para programarla.

1.1. CARACTERÍSTICAS DEL TMS320C6713 DSK

En la siguiente tabla hay algunos atributos básicos del TMS320C6713 DSK.

Característica	**Valor**
Frecuencia del Reloj	225 MHz
Memoria SDRAM	16 MB
Memoria FLASH	256 KB
Arquitectura	VLIW (Very-Long-Instruction-Word)
I/O Señales de Audio	2 para entrada y 2 para salida

Otras características especiales disponibles para el DSK son:

- La tarjeta tiene un convertido de análogo a digital (ADC) y un convertidor de digital a análogo (DAC).
- Los canales McASP tienen un filtro en la entrada especial para "anti-aliasing" el cual elimina señales erróneas y un filtro en la salida para suavizar o reconstruir la señal de salida procesada.
- Una espacio para tarjetas de expansión con un conector de 80 pines proveído para realizar interfase con periféricos externos y memoria externa.
- Cuatro switches tipo DIP.
- Reguladores de voltaje que proveen 1.26V para el DSP y 3.3V para la memoria y los periféricos.

1.2. LA ARQUITECTURA DEL TMS320C6713 DSP

La memoria interna del TMS320C6713 DSP tiene una arquitectura de dos niveles de memoria caché. El primer nivel tiene 4KB para programa y 4KB de memoria para datos. El segundo nivel tiene 256KB compartidos entre programa y datos.

Hay dos bancos diferentes con dos buses de 32 bits para poder ser accedidos independientemente.

La CPU del DSP tiene ocho unidades funcionales independientes las cuales están divididas en dos rutas y son útiles para operaciones de multiplicación (.M), operaciones de lógica y aritmética (.L), para manipulaciones de bits (.S) y carga y descarga de datos (.D).

2. SISTEMA DE DESARROLLO DSP

Las posibilidades de desarrollar una aplicación en el DSP C6713 son variadas. Las herramientas utilizadas para compilar y bajar programas al DSP son Matlab, Labview, Visual Basic y Visual C++. Estas herramientas pueden comunicarse con el DSP por medio del uso RTDX (Real Time Data Exchange) según la aplicación lo requiera.

2.1. CODE COMPOSER STUDIO IDE (CCS)

Este es un Ambiente de Desarrollo Integrado de Texas Instruments Inc., utilizado para crear y hacer “debug” a aplicaciones desarrolladas en C o en lenguaje de ensamblador. Algunas de las características especiales de este ambiente son la posibilidad de revisar variables y registros desde el DSK y su utilidad para intercambiar data entre la tarjeta y otros lenguajes de programación así como Matlab.

En este trabajo se utiliza la versión Code Composer Studio IDE Platinum Edition como plataforma de programación de los DSPs de Texas Instruments, Inc. y

también se utiliza para calcular la cantidad de operaciones de punto flotante ejecutadas durante cualquier proceso, y así evaluar el rendimiento de una implementación para un algoritmo. El CCS IDE puede ser utilizado para revisar los resultados de una implementación debido a la posibilidad de revisar la memoria de la tarjeta.

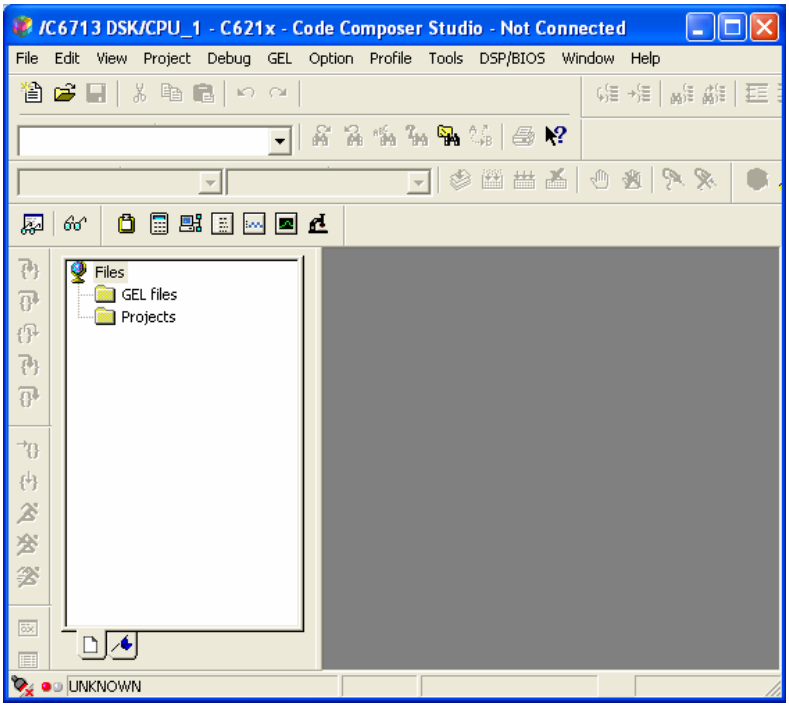


Figura 1. Ejemplo del Ambiente de programación: CCS.

2.1.1. Instalación del CCS e instalación de Drivers El ambiente de desarrollo esta provisto por Texas Instruments. Instale el Code Composer Studio IDE Platinum Edition desde el CD, La instalación crea un archivo con la dirección C:\ CCStudio_v3.1, seguidamente instale los drivers del DSK del CD que viene con la tarjeta TMS320C6713 estos driver se instalaran en la carpeta anteriormente nombrada.

Al terminar la instalación, en su escritorio aparecerán los siguientes accesos directos.

Con este link se puede seleccionar la tarjeta de desarrollo con la que usted desee trabajar que para este caso será la C6713 DSK.

Este link accede al CCS con la configuración establecida por última vez.

Con este link se accede al CCS con la configuración para la tarjeta 6713 DSK.

Este link me permite revisar el estado de la tarjeta 6713 DSK.

Asegúrese que el DSK este alimentado a 5V y conectado con el PC por medio del un cable USB. Windows le informara que encontró un nuevo hardware, usted puede dar le la opción para buscar el driver de una lista específica o búsqueda automática. Windows le informara cunado el dispositivo este listo para usar.

Ingrese al CCS por medio del acceso directo “6713 DSK CCStudio v3.” Aparecer una ventana como la mostrada en la figura 1.

A continuación evaluaremos los siguientes aspectos que hay que tener en cuenta a la hora de realizar un proyecto en el CCS.

1. Construir el proyecto → 
2. Cargar el proyecto a la tarjeta.

3. Correr el proyecto → 
4. Evaluar los resultados y corregir los errores.
5. Si hay errores en los resultados volver al paso 2.

2.1.2. Tipos de archivos útiles Cada programa que se construye utilizando CCS estará trabajando con una serie de archivos con extensiones diferentes.

- Namefile.pjt: Para crear y construir el proyecto.
- Namefile.c: Programa en lenguaje C creado por el usuario. Puede ser uno o varios según la aplicación.
- Namefile.asm: Programa en lenguaje de ensamblador creado por el usuario. Pueden ser uno o varios dependiendo de la aplicación.
- Namefile.h: Archivos de soporte de los programas en C o Ensamblador.
- Namefile.lib: Librerías de soporte.
- Namefile.cmd: Archivo con la descripción de la distribución de memoria en el DSP.
- Namefile.obj: Archivos creados después de la compilación del proyecto.
- Namefile.out: Archivo ejecutable creado por el “linker” para ser cargado en el procesador.
- Namefile.cdb: Archivo de configuraciones cuando se utiliza la herramienta DSP/BIOS en la aplicación.

2.2. HERRAMIENTAS DE APOYO DEL DSK

Los siguientes archivos de apoyo son utilizados frecuentemente cuando se crea un proyecto:

- **C6713dskinit.c:** Contiene funciones que inicializan el DSK, los codecs para los puertos seriales y las I/O de la tarjeta.
- **C6713dskinit.h:** Directorio con la descripción de las funciones para inicializar la tarjeta.
- **C6713dsk.cmd:** Archivo con una distribución útil para organizar la memoria del DSP.
- **Vectors_intr.asm:** Archivo en lenguaje de ensamblador que permite manejar las interrupciones.
- **Vectors_poll.asm:** Archivo en lenguaje de ensamblador que permite manejar el acceso a los puertos haciendo "polling".
- **Rts6700.lib: dsk6713bsl.lib; csl6713.lib; rtdx.lib:** Estas librerías son de apoyo para la tarjeta, el chip y el intercambio de datos en "tiempo real".

2.3. EJEMPLOS DE PROGRAMACIÓN PARA PROBAR LAS HERRAMIENTAS DEL DSK

El siguiente programa ilustra las características del CCS y de la tarjeta DSK. Este ejemplo muestra paso a paso, como crear un proyecto para compilarlo y bajarlo al DSK TMS320C6713. Asegúrese de colocar los archivos que viene con esta guía en la carpeta C:\ CCStudio_v3.1\Myprojects antes de comenzar a trabajar en los ejemplos.

Ejemplo 1: Hello World!

2.3.1. Creando el Proyecto En esta sección se muestra como crear un proyecto, añadiendo los archivos necesarios para construirlo utilizando CCS.

1. Seleccione *Project* ⟶ *New*. Como nombre del proyecto escriba “hello” y luego haga clic en “Save”.

Este archivo del proyecto (.pjt) se guarda en la carpeta “hello” (C:\CCStudio_v3.1\MyProjects\hello). La figura 2 y 3 muestran como crear un proyecto nuevo y las carpetas que se deben observar al crear el proyecto.

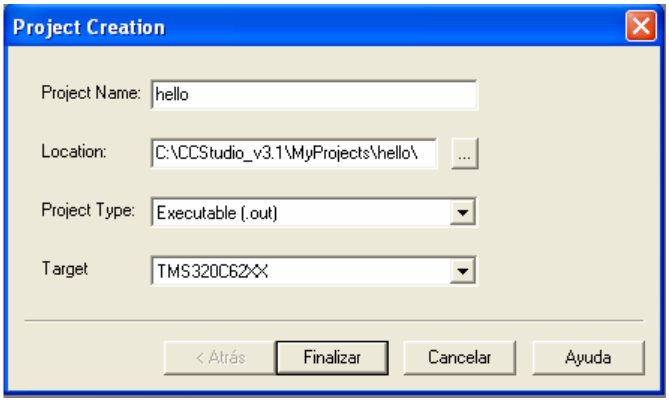


Figura 2: Ventana para la creación de un proyecto nuevo

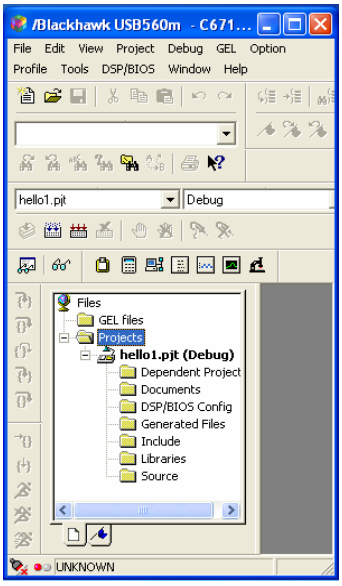


Figura 3: Carpetas del proyecto.

2. Seleccione *File* → *New*, copie el siguiente código en C (.C) y haga clic en “Save As” y guarde el archivo como “hello.c”.

```
#include <std.h>
// ======== main ========
void main()
{
puts("hello world!\n");
return;
}
```

3. Seleccione *Project* → *Add files to project*. Añada el archivo “hello.c” creado en el paso anterior.

4. Repita el paso 3 para añadir al proyecto el archivo “.asm” vectors_poll.asm. Repita otra vez y seleccione el archivo “.cmd”, C6713dsk.cmd para añadirlo al proyecto, el archivo se localiza en la carpeta “Support”.

5. De manera similar al paso previo, los siguientes archivos “.lib” deben ser añadidos: rts6700.lib localizado en la siguiente ruta C:\CCStudio_v3.1\c6000\cgtools\lib; dsk6713bsl.lib localizado en la ruta C:\CCStudio_v3.1\c6000\dsk6713\lib; y las librerías de soporte del chip, csl6713.lib localizado en la dirección C:\CCStudio_v3.1\C6000\csl\lib.

6. Seleccione *Project* → *Scan All Files Dependencies*. Verifique que todos los archivos que se muestran en la figura 4 hayan sido añadidos al proyecto.

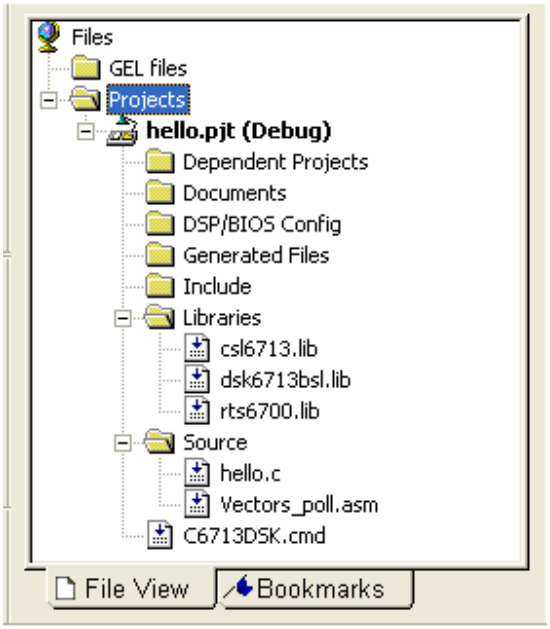


Figura 4: Archivos del proyecto

7. Seleccione Project → Build Options. En la opción "compiler" seleccione "preprocessor" y en la opción "Define Symbols" (-d), escriba CHIP_6713. En la opción "Linker", seleccione "Include Libraries" y escriba rts6700.lib; dsk6713bsl.lib; csl6713.lib. Finalmente haga clic en OK para guardar las opciones.

2.3.2. Compilando y haciendo "debug" al Proyecto En este paso se realizan la compilación y creación del ejecutable del proyecto:

1. Haga clic en el botón 'rebuild all'  que esta localizado en la parte superior del ambiente CCS y verifique que obtenga 0 errores, 0 advertencias y 0 comentarios.

2. Seleccione *File* → *Load Program*. Cargue el archivo “hello.out” localizado en el siguiente directorio: C:\CCStudio_v3.1\MyProjects\hello\Debug.

3. Haga clic en el botón ‘run’  localizado en la parte izquierda del ambiente CCS.

2.3.3. Observando los resultados El mensaje “hello world” aparece en el “Stdout” (Parte inferior del ambiente CCS) y entonces el programa termina.
Si muestra algún error revise cuidadosamente que haya hecho correctamente todos los pasos.

3. EJEMPLOS DE ALGUNAS IMPLEMENTACIONES

En esta sección se presentan algunos algoritmos (Código C y archivos asociados) y proyectos necesarios para entender las herramientas de software y de hardware para la tarjeta.

Nota: asegúrese de guardar los proyectos de la carpeta support que viene en el CD adjunto a esta guía en la dirección C:\CCStudio_v3.1\MyProjects.

3.1. EJEMPLO 2. GENERAR Y GRAFICAR UNA SINUSOIDE CON CCS

El propósito de este ejemplo es mostrar el funcionamiento del canal de salida “McASP” generando una onda sinusoidal utilizando el ambiente del CCS.

3.1.1. Creando el Proyecto

1. Haga clic en *Project* → *Open*. Busque y seleccione el archivo "Sine2sliders" en el siguiente directorio: C:\CCStudio_v3.1\MyProjects\Sine2sliders.
2. Haga doble clic en "Sine2sliders.pjt" localizado en el lado izquierdo y haga clic en *Source* para ver los archivos. Luego haga clic en "Sine2sliders.c". Su ambiente debe verse como en la figura 5 y debe tener todos los archivos que se encuentran al lado izquierdo de la ventana.

   Nota: Verifique que en la opción Project → Build Options., la opción compiler → preprocessor → Pre-Define Symbols (-d), esta escrito CHIP_6713 lo cual selecciona este chip. También, verifique que en la opción Linker → Include Libraries este escrito rts6700.lib; dsk6713bsl.lib; csl6713.lib. Finalmente haga clic en Aceptar para guardar las opciones.
3. Haga clic en el botón 'rebuild all'  que esta localizado en la parte superior del ambiente CCS y verifique que obtenga 0 errores, 0 advertencias y 0 comentarios.
4. Haga clic en *File* → *Load Program*, luego haga doble clic en "Debug" y abra el archivo "Sine2sliders.out" que debe estar en es siguiente: C:\CCStudio_v3.1\Myprojects\ Sine2sliders \Debug.
5. Haga clic en el botón 'run'  localizado en la parte izquierda del ambiente CCS.

3.1.2. Resultados

6. Conecte la salida J302. Revise la salida de la tarjeta marcada como "HEADPHONE" de la tarjeta a la bocina.

7. Seleccione la opción *File* → *Load Gel*. Escoja la opción “Sine2sliders.gel” que se encuentra en el directorio: C:\ CCStudio_v3.1\Myprojects\ Sine2sliders.

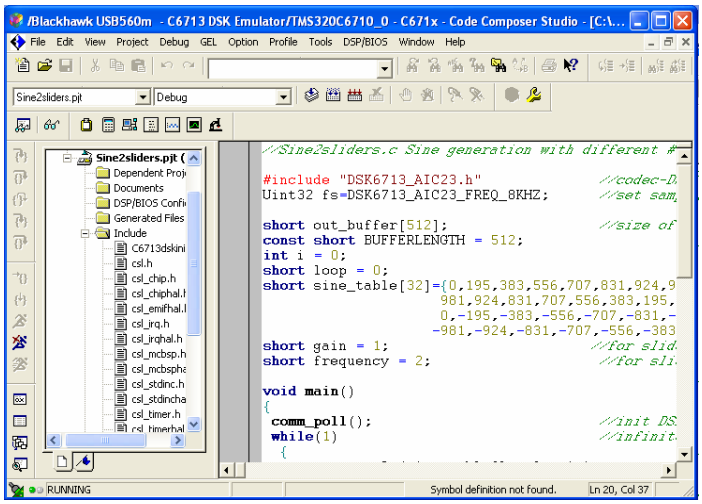


Figura 5: Ventana del ambiente para el ejemplo 2.

8. Seleccione *Gel* → *Sine Parameters* y finalmente haga clic en “Gain” y “Frequency”. Debe aparecer una ventana como en la figura 5

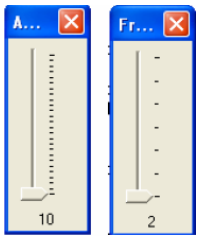


Figura 6: Ventanas con deslizadores para controlar la amplitud y la frecuencia.

9. Presione la tecla de subir o bajar para incrementar o bajar los valores de la amplitud y la frecuencia hasta el máximo, El cambio se notará en las ventanas de la figura 6. Verifique en la bocinas que la amplitud de la onda

sinusoidal generada se ha aumentado y la frecuencia de la misma ha cambiado.

10. Seleccione *View* → *Graph* → *time/frequency.*

11. Cambie el Graph Property Dialog tal que las opciones en la figura 6 estén seleccionadas para la grafica en el dominio del tiempo. La dirección inicial del buffer de salida es “outputbuffer”.

12. Seleccione *View* → *Graph* → *time/frequency*.

13. Cambie el “Graph Property Dialog” tal que las opciones en la figura 7 estén seleccionadas. En la opción “Display Type” seleccione FFT “Magnitude.” Presione OK y verifique que la gráfica de la magnitud de la FFT es un pico en una frecuencia específica.

14. Si cambia los parámetros de frecuencia o amplitud, debe hacer un clic derecho en la grafica y seleccionar “Refresh”; la magnitud y la frecuencia de la onda sinusoidal cambiarán.

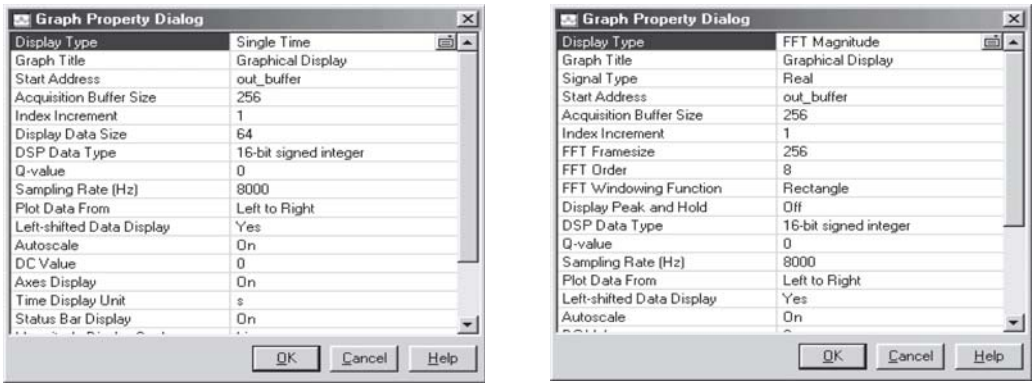


Figura 7: Cuadro de diálogo con las propiedades para graficar.

3.2. Ejemplo 3. Filtrando Señales de Audio.

Requisitos:

- Conector de Audio stereo-stereo.
- Audífonos.

El propósito de este ejemplo es mostrar el funcionamiento de los canales de entrada y salida McASP. Esta aplicación es utilizada para filtrar una señal de audio en tres diferentes frecuencias: 600Hz, 1500Hz o 3000Hz, utilizando el ambiente del "Code Composer Studio". Para este ejemplo es necesario tomar los coeficientes de los tres filtros diferentes utilizando Matlab, guardar los datos en un archivo (.h) y añadirlos al proyecto. En los archivos adjuntos los datos de los coeficientes fueron llamados: lp600.h, lp1500.h y lp3000.h.

3.2.1. Creando el Proyecto

1. Haga clic en *Project* → *Open*. Busque y seleccione el archivo "FIR3LP.pjt" que se encuentra en la siguiente ruta: C:\ CCStudio-_v3.1\Myprojects\ FIR3LP.

2. Haga doble clic en "FIR3LP.pjt" localizado en el lado izquierdo y haga clic en "Source" para ver los archivos. Luego haga clic en "filter.c". Su ambiente debe verse como en la figura 8 y debe tener todos los archivos que se encuentran al lado izquierdo en la misma.

Nota: Verifique que en la opción *Project* → *Build Options*., la opción *compiler* → *preprocessor* → *Pre-Define Symbols* (-d), esta escrito la opción CHIP_6713. También, verifique que en la opción *Linker* → *Include*

Libraries este escrito rts6700.lib; dsk6713bsl.lib; csl6713.lib finalmente haga clic en Aceptar para guardar las opciones.

3. Haga clic en el botón 'rebuild all'  que esta localizado en la parte superior del ambiente CCS y verifique que obtenga 0 errores, 0 advertencias y 0 comentarios.

4. Haga clic en *File* → *Load Program*, luego haga doble clic en "Debug" y abra el archivo "FIR3LP.out" que debe estar en el siguiente directorio: C:\ CCStudio_v3.1\Myprojects\ FIR3LP\Debug.

5. Haga clic en el botón 'run'  localizado en la parte izquierda del ambiente CCS.

3.2.2. Resultados

6. Conecte la salida de audio del PC a la tarjeta en la entrada J7. Abra el programa de Media Placer de Windows y haga clic en cualquier archivo de extensión file.mp3.

7. Conecte la salida J304 de la tarjeta a los audífonos.

8. Seleccione la opción *File* → *Load gel*. Escoja la opción "filter.gel" que se encuentra en la siguiente ruta: C:\ CCStudio v3.1\Myprojects\ FIR3LP.

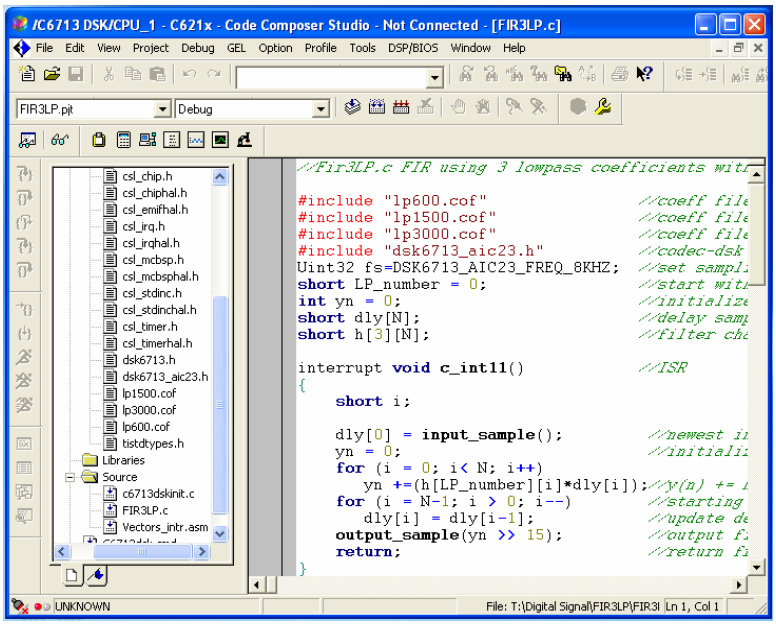


Figura 8. Ventana del ambiente para el ejemplo 3.

9. Seleccione GEL → Filter characteristics y finalmente haga clic en “filter”.

10. Modifique los Valores de las ventanas deslizantes para cambiar la frecuencia de corte del filtro. Cuando el valor de la ventana se encuentra en 0, la frecuencia de corte es 600Hz, en 1 la frecuencia de corte es de 1500 Hz y cuando esta en 2 la frecuencia de corte es de 3000Hz.

4. MATLAB COMO INTERFACE PARA DSP’S

Matlab puede utilizarse como un lenguaje de programación para unir un DSP con una PC con el propósito de intercambiar datos entre la memoria de la PC y el DSP, y viceversa. Matlab puede verse como una herramienta para visualizar los resultados procesados por la tarjeta con el objetivo de evaluar el funcionamiento correcto del algoritmo implementado. Por otra parte, algunos desarrolladores

utilizan Matlab para modelar sistemas y evaluar algoritmos, luego los bajan directamente desde Matlab sin traducir a C o a lenguaje de ensamblador.

Las siguientes herramientas deben estar instaladas en la computadora para el funcionamiento correcto de la familia de DSPs TMSC6X:

- Link for Code Composer Studio®
- Embedded target for TI C6000 DSP.

Esta última es requerida para implementar algoritmos del DSP utilizando simulink. Si instalo la versión Code Composer Studio IDE Platinum Edition estas herramientas ya fueron incluidas durante la instalación.
A lo largo de este trabajo los ejemplos pretenden demostrar la transmisión de datos entre Matlab y la tarjeta DSP.

Desde Matlab se puede abrir CCS, bajar y correr proyectos; y se puede detener la ejecución de un programa, cerrar proyectos, y hacerle *reset* a la tarjeta. En la Tabla 1 se encuentran las funciones para manejar CCS desde Matlab. Estas funciones se pueden utilizar mientras el procesador no esta corriendo.
Para un mejor entendimiento de esta guía, las instrucciones de Matlab estarán escritas en *verde* y las instrucciones del DSP estarán escritas en *azul*.

Función	Descripción
Open	Este comando abre un proyecto de CCS localizado en la ruta especificado en directorio actual en Matlab. Ej. :open(cc,' filename.pjt')
Load	Con esta opción es posible bajar un archivo de output (.out) a la tarjeta. Ej. : load(cc,'/debug/ filename.out')
Run	Comienza a ejecutar el programa que este cargado en la tarjeta y al cual hace referencia *cc*. Ej. : run(cc)
Halt	Detiene la ejecución del programa que se esté corriendo en la tarjeta. También permite configurar una opción para detener el objetivo en un tiempo específico. Ej. : halt(cc,timeout)
Read	Lee de la dirección de memoria (*dir*) especificada en la instrucción. Se puede configurar un tiempo de espera para permitir que se complete la operación de lectura (*timeout*). También es posible definir el tipo de data (*datatype*) y el tamaño de la data a leer (*size*). Ej. : read(cc,dir,'datatype',size,timeout)
Write	Permite enviar un bloque de datos al CCS que puede ser utilizado cuando el procesador esta corriendo. Se envía a una dirección específica (dir) en un tiempo de espera definido (timeout). Ej. : write(cc,dir,data,timeout)
Restart	Detiene el procesador y reinicia el programa que este cargado en la tarjeta. Ej. : restart(cc)
Close	Permite cerrar un proyecto abierto en CCS. Para cerrar el proyecto, 'type' se define como 'project' y 'filename' es el nombre del proyecto que se va a cerrar. Ej. : close(cc,'filename','type')

Tabla 1. Funciones para manejar CCS desde Matlab

4.1. Ejemplos de DSP utilizando la interfase de Matlab

Ejemplo 1

1. Cree un archivo .m con las siguientes instrucciones y asegúrese que se encuentra en la carpeta *work* de Matlab

extract.m

```
ccsboardinfo                                    % información de la tarjeta
cc=ccsdsp;                                      % configura el objeto CCS
cd C:\ti\Myprojects\ Sine2sliders               % cambia el directorio actual
reset(cc);                                      % hace “reset” a la tarjeta
enable(cc.rtdx);                                % habilita RTDX
open(cc,' Sine2sliders.pjt');                   % abre el proyecto
load(cc,'./debug/ Sine2sliders.out');           %carga el archivo ejecutable
run(cc);                                        % corre el programa cargado
a = address(cc,'out_buffer');                   % Encuentra la dirección
                                                % de la variable
data=read(cc,address(cc,'out_buffer'),'int16',256);  %Lee la variable
plot(data)                                        % grafica la variable de la
                                                  % tarjeta
close(cc, ' Sine2sliders.pjt','project')        %Cierra el proyecto actual
```

2. El proyecto “Sine2sliders” del ejemplo 2, explicado previamente, tiene que estar en la carpeta: C:\CCStudio_v3.1\Myprojects y debe estar compilado antes de comenzar a correr la función de Matlab.

3. Escriba *extract* en el prompt de Matlab y asegurándose que el directorio actual es el de la carpeta *work* de Matlab.
4. Este ejemplo abre el programa CCS y carga y corre el proyecto. Cuando el programa termina la data que esta guardada en la variable out_buffer del proyecto se guarda en la variable data del workspace de Matlab.

5. Debe aparecer una gráfica sinusoidal en Matlab.

Ejemplo 2:

1. Cree un archivo file.m con las siguientes instrucciones y asegúrese que se encuentra en la carpeta *work* de Matlab.

writting.m

```
cc = ccsdsp;
cd C:\ti\myprojects\ Sine2sliders        % cambia el directorio actual
reset(cc);                               % hace “reset” a la tarjeta
enable(cc.rtdx);                         % habilita RTDX
open(cc,' Sine2sliders.pjt');            % abre el proyecto
load(cc,'./debug/ Sine2sliders.out');    % carga el archivo ejecutable
run(cc);                                 % corre el proyecto cargado
indata = 1:25;
write(cc,0,indata,30);
outdata=read(cc,0,'double',25)
```

2. El proyecto “Sine2sliders” del ejemplo 2, explicado previamente, tiene que estar en la carpeta: C:\CCStudio_v3.1\myprojects y debe estar compilado antes de comenzar a correr la función de Matlab.

3. Escriba *writting* en el “prompt” de Matlab asegurándose que el directorio actual es el de la carpeta *work* de Matlab.
4. Este ejemplo abre el programa CCS y carga y corre el proyecto. Mientras el programa esta corriendo, escribe la variable “indata” en la dirección “0” del DSP. Luego la data es regresada a Matlab en la variable “outdata”.

5. Los siguientes resultados deben aparecer en Matlab:

```
outdata =
  Columns 1 through 12

    1    2    3    4    5    6    7    8    9   10   11   12

  Columns 13 through 24

   13   14   15   16   17   18   19   20   21   22   23   24
  Column 25

    25
```

4.1.1. Transmisión de data PC → DSP La implementación del algoritmo para la transmisión de data del PC al DSP consiste en crear uno o más canales de entrada para transferir dada y por tal razón la siguiente instrucción son añadidas al programa del DSP.

```
RTDX_CreateInputChannel(ichan)
```

En este caso el canal llamado “ichan” es creado para la transmisión de data.

En el menú principal del programa del DSP se colocan las instrucciones para recibir la data desde Matlab. Cuando el canal es habilitado comienza la transmisión y continua hasta que el canal es deshabilitado.

```
TARGET_INITIALIZE()
while(!RTDX_isInputEnabled(&ichan))
RTDX_read(&ichan,buffer,sizeof(buffer))
```

Aquí se muestra que la data es recibida mientras “ichan” esta habilitada y se almacena en la variable “buffer” que ha sido definida al inicio con un tamaño específico.

Por otra parte las siguientes instrucciones en Matlab permiten dar inicio a la comunicación:

```
cc = ccsdsp('boardnum',0)
enable(cc.rtdx)
open(cc.rtdx,'ichan','w')
enable(cc.rtdx,'ichan')
writemsg(cc.rtdx,'ichan', ‘variable’)
```

Con las primeras dos instrucciones el canal es creado y el RTDX es habilitado. Luego se abre y habilita el canal de entrada y finalmente la data localizada en el registro “variable” es enviada al DSP. Si la transmisión termina el canal es deshabilitado desde Matlab utilizando:

```
disable(cc.rtdx,'ichan')
```

4.1.2. Transmisión de data DSP → PC La implementación del algoritmo para la transmisión de data del DSP a la PC consiste en crear uno o más canales de output para transferir dada y por tal razón la siguiente instrucción son añadidas al programa del DSP.

```
RTDX_CreateOutputChannel(ochan)
```

En este caso el canal de output llamado “ochan” es creado para la transmisión de data.

En el menú principal del programa del DSP se colocan las instrucciones para enviar la data a Matlab. Cuando el canal es habilitado comienza la transmisión y continua hasta que el canal es deshabilitado.

```
TARGET_INITIALIZE()
RTDX_enableOutput(&ochan)
RTDX_write( &ochan, message, sizeof(message))
```

Aquí se muestra que la data es transmitida cuando “ochan” esta habilitado y está localizada en la variable *“message”* que a sido definida al inicio con un tamaño específico.

Por otra parte las siguientes instrucciones en Matlab permiten el almacenamiento de data

```
enable(cc.rtdx,'ochan')
output = readmsg(cc.rtdx,'ochan',)
```

La primera instrucción habilita el canal de output y la ultima instrucción guarda la data del DSP en la variable *“output”*. Si la transmisión termina el canal es deshabilitado desde Matlab utilizando:

```
disable(cc.rtdx,'ochan')
```

Ejemplo 3:

Este ejemplo muestra una secuencia de instrucciones para transmitir data del DSP a la PC:

1. Cree un archivo file.m con las siguientes instrucciones y asegúrese que se encuentra en la carpeta C:\CCStudio_v3.1\MyProjects\rtdx_matlab_sim.

Transfer.m

```
indata(1:10) = [1:10];                  % data a ser enviada al DSK
ccsboardinfo                            % información de la tarjeta
cc = ccsdsp('boardnum',0);              % configura el objeto CCS
reset(cc)                               % hace “reset” a la tarjeta
visible(cc,1);                          % para la ventana CCS
enable(cc.rtdx);                        % habilita RTDX
if ~isenabled(cc.rtdx)                  % verifica que le canal este listo
error('RTDX is not enabled')
end
cc.rtdx.set('timeout', 20);              % fija 20seg de espera
open(cc,'rtdx_matlab_sim.pjt');          % abre el proyecto
load(cc,'./debug/rtdx_matlab_sim.out');  % carga el archivo ejecutable
run(cc);                                 % corre el programa cargado
configure(cc.rtdx,1024,2);               % configura dos canales RTDX
open(cc.rtdx,'ichan','w');               % abre el canal de input
open(cc.rtdx,'ochan','r');              % abre el canal de output
pause(3)                                % pausa de 3seg
enable(cc.rtdx,'ichan');                % habilita el canal ichan
if isenabled(cc.rtdx,'ichan')           % verifica que el canal este listo
writemsg(cc.rtdx,'ichan', int16(indata))  % envía data de 16 bits al DSK
pause(3)
else
error('Channel ''ichan'' is not enabled')
```

```
end
enable(cc.rtdx,'ochan');                    % habilita el canal  ochan
if isenabled(cc.rtdx,'ochan')
 outdata = readmsg(cc.rtdx,'ochan','int16');    % lee data de 16bits del DSK
  pause(3)
  else
 error('Channel ''ochan'' is not enabled')
end
if isrunning(cc), halt(cc);                 % de tiene el procesador
end
disable(cc.rtdx);                           % deshabilita RTDX
close(cc.rtdx,'ichan');                     % cierra el canal de input
close(cc.rtdx,'ochan');                     % cierra el canal de output
```

2. Copie la carpeta "rtdx_matlab_sim" localizada en C:\CCStudio_v3.1\MyProjects\rtdx_matlab_sim y compílela antes de comenzar a ejecutar la función de Matlab.

3. Escriba rtdx_matlab_sim en el prompt de Matlab asegurándose que el directorio actual es C:\CCStudio_v3.1\MyProjects\rtdx_matlab_sim.

4. Este ejemplo abre el programa CCS y carga y corre el proyecto. Abre dos canales para la transmisión de data: "ichan" es creado para enviar data desde el "*workspace"* de Matlab al DSP y "ochan" es creado para enviar data desde el DSP al "*workspace* "de Matlab.

5. La variable "indata" es enviada al DSP. Revise que la variable "outdata" es creada en el *workspace* de Matlab cuando el programa termina. Las dos variables deben ser iguales.

6. Si uno de los siguientes errores aparece durante el proceso, usted debe detener el programa Matlab, reconectar la tarjeta y comenzar el procedimiento nuevamente.

   ?? Error using ==> ccs.ccsdsp.open, ?? Error using ==> transmission

4.2. MANEJO DEL BUFFER Y MEMORIA VIRTUAL DEL PC

Según las aplicaciones que se vaya a trabajar y la cantidad de datos a transferir, es posible configurar el buffer que se utiliza para la comunicación de la tarjeta con el PC. El tamaño del buffer por defecto es de 1024 bytes con un máximo de 65536 bytes, equivalente a transferir 8190 datos en formato double. Para configurar el tamaño del buffer del PC siga los siguientes pasos:

- Seleccione tools → RTDX → configuration control → configure.

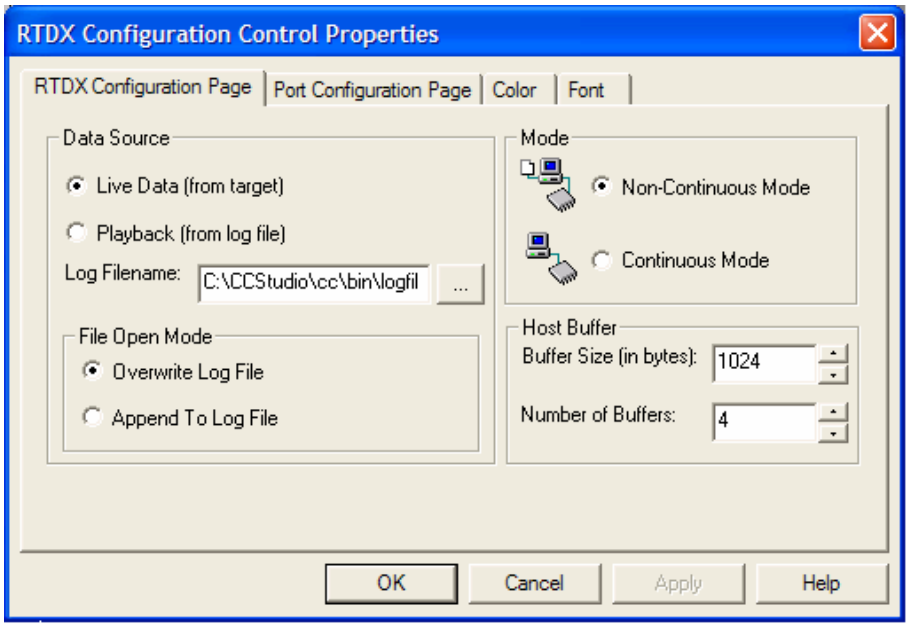


Figura 9: ventana de configuración de buffer

Para configurar el buffer de la tarjeta realice los siguientes pasos:

- Agregue el siguiente archivo a su proyecto rtdx_buf.c que esta en la dirección C:\CCStudio_v3.1\C6000\rtdx\lib.
- Abra el archivo y en la línea 12 del código, modifique el tamaño del buffer igual al tamaño del buffer configurado para el PC.

También es importante configurar la memoria virtual del PC que se este utilizando, pues si la cantidad de datos utilizados es muy grande el tamaño que este configurado por defecto puede ser insuficiente y generar errores.

Para configurar la memoria virtual se sigue el siguiente procedimiento:

- Vaya a inicio → panel de control → sistema → opciones avanzadas y donde dice uso de memoria virtual de clic en configurar.
- Una vez se le da configurar de clic en la pestaña de opciones avanzas, en memoria virtual aparece una opción de cambiar (figura 10).
- En esta ventana podrá ver la memoria virtual por defecto del PC, seleccione la opción de tamaño personalizado y amplié el tamaño, si es posible al límite permitido que se indica en la parte superior.
- Una vez realizado este cambio correctamente el PC se reiniciara para salvar los cambios.

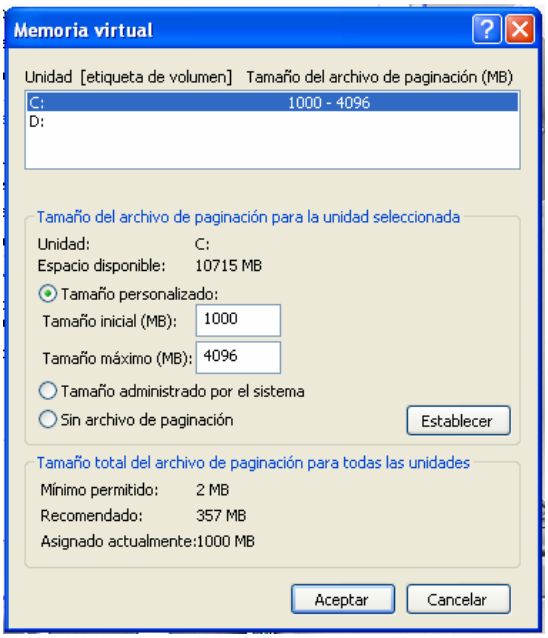


Figura 10: configuración de memoria virtual del PC

5. ERRORES TÍPICOS TRABAJANDO CON CCS

Errores de compilación

1. si este mensaje de error (figura 11) aparece en la ventana inferior derecha asegúrese de incluir la librería dentro de la carpeta del proyecto, estas librerías se puede encontrar en la carpeta support.

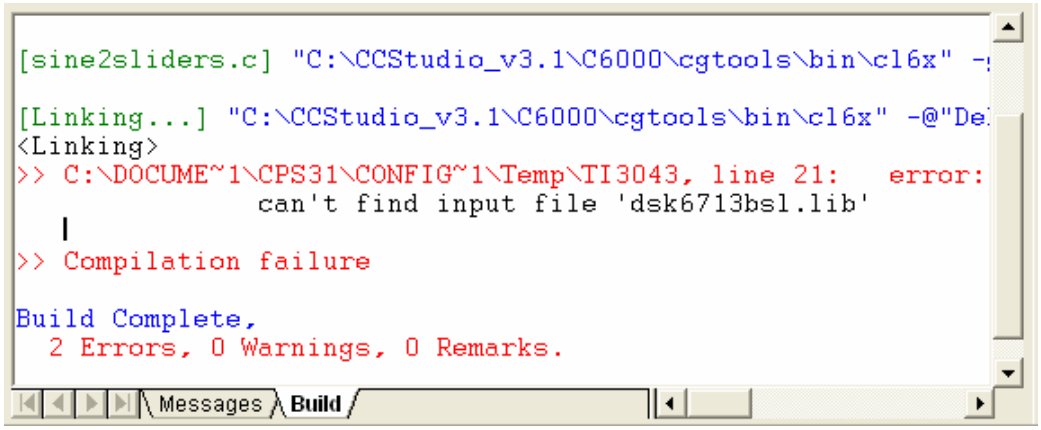


Figura 11: Error de compilación

2. Otro tipo de mensaje común es el que vemos en la figura 12, hace referencia a la memoria del DSK, diríjase a Project⟶Build Options: Compiler⟶Advanced⟶Memory Models: y selecione la opción ⟶ Far[--mem model:data=far] de clic en Aceptar y compile de nuevo el proyecto.

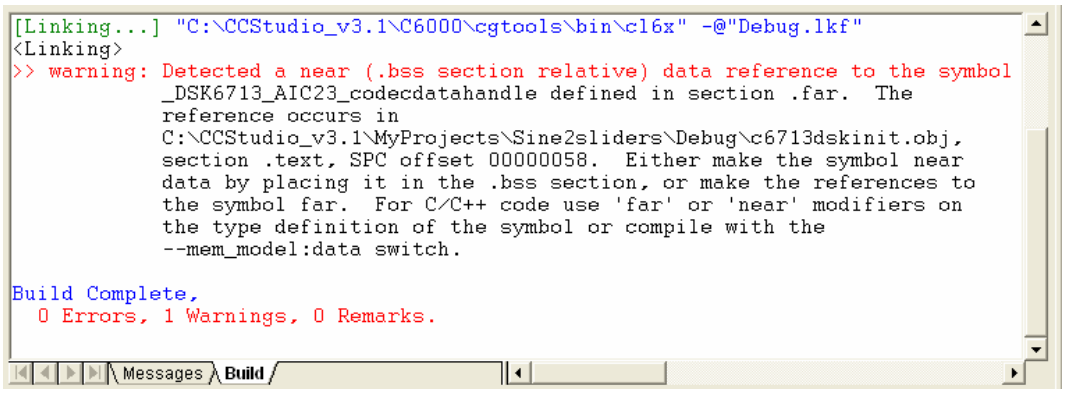


Figura 12: Warning de memoria.

6. USB560 JTAG EMULATOR - BLACKHAWK

Una herramienta adicional para utilizar junto con el DSK en el USB560 JTAG Emulator Blackhawk esta herramienta permite una transferencia de data mucho más rápida al DSP.

6.1 CARACTERÍSTICAS DEL USB560 JTAG EMULADOR

- Tasa de transferencia de datos superior a 2MB/seg.
- Puerto USB de alta velocidad 2.0 (480MB/seg).
- Hasta 100 veces más rápido que su antecesor USB510.
- Compatible con versiones del code componer estudio (CCS) v1.2 hasta v4.1x.

Para la instalación y funcionamiento del emulador se requieren los siguientes elementos:

1. CCS versión platinum. (Este CCS se requiere debido a que permite manejar emuladores, lo cual no es posible simplemente con el CCS que viene con las tarjetas de desarrollo esta es la versión completa del CCS para programación de los procesadores).

2. CCS-DSK version 3.0 o mas alta. (Este CCS se requiere debido a que contiene los drivers para el manejo de los periféricos de la tarjeta. Este CD viene con las tarjetas de desarrollo de los DSP).

3. Blackhawk drivers. (Este CD viene con el blackhawk)

4. DSK TMS320C6713 y Blackhawk Emulator.

6.2. INSTALACIÓN.

1. Instale los drivers que vienen en el DC del Blackhawk Emulator y conecte el dispositivo al PC por medio de su puerto USB, escoja instalar los drivers para la versión platinum del Code Componer Studio versión 3.1. Al terminar la instalación, en su escritorio aparecerá el siguiente acceso directo del Blackhawk. Con este link se podrá verificar la conexión del blackhawk con el sistema. Al hacer doble click sobre el icono se abrirá una ventana como la mostrada en la figura 13.

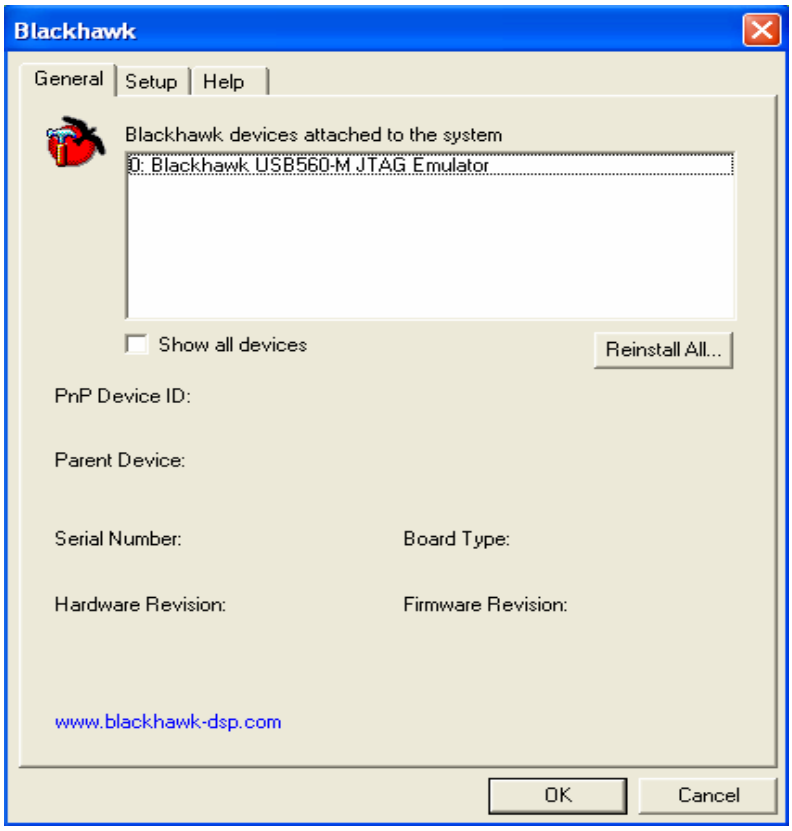


Figura 13: Blackhawk Emulator

2. Haga doble clic en la opción “Setup CCStudio v. 3.1”.  En la lista de “Available Factory Boards” (ver figura 14) elija la opción “Blackhawk USB560m – C6713 DSK Emulator”. Luego haga clic en “Add” y finalmente en “Save & Quit”. Esto configura el método de comunicación entre el PC y el DSK. Si no se desea trabajar con el Emulador se usa la opción “C6713 DSK” en la lista de “Available Factory Boards” y se desconecta el emulador de la tarjeta conectando el DSK directamente al PC por el puerto USB.

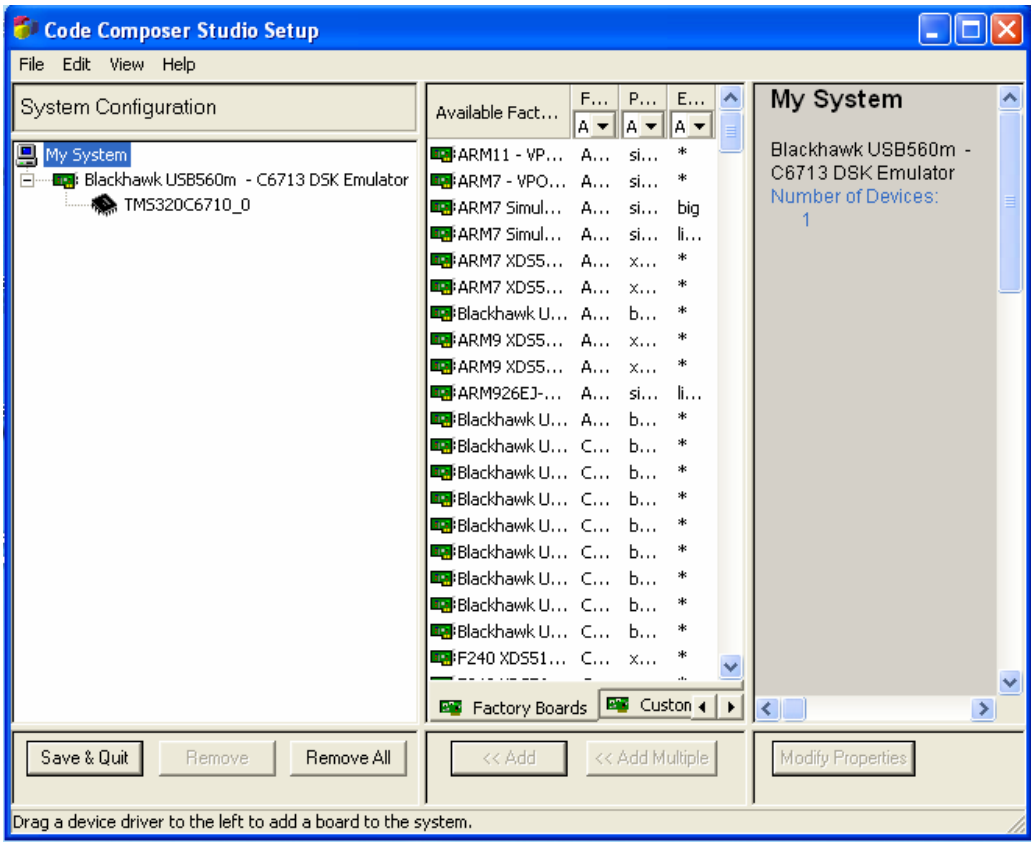
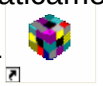


Figura 14: Blackhawk Control Panel

Si el dispositivo fue correctamente instalado y esta conectado al PC deberá aparecer la referencia del dispositivo en esta ventana.

3. Una vez configurado los dispositivos y salvada la configuración el CCS se inicia automáticamente, o puede utilizar el siguiente acceso directo CCStudio 3.1 

6.3. CONFIGURACIÓN PARA COMPILACIÓN UTILIZANDO EMULADOR

Para utilizar emulador USB560 es necesario hacer unos cambios en el archivo intvecs.asm, con el cual se puede habilitar la unidad HSRTDX (Higth Speed

RTDX). Busque en el directorio de archivos del proyecto el archivo intvecs.asm ábralo y realice los siguientes cambios:

- La línea 21 del código colóquela como comentario JTAGRTDX .set 1
- En la línea 14 digite el siguiente código: HSRTDX .set 1
- Cambien la librería rtdx.lib por rtdxhs.lib que esta en la dirección C:\CCStudio_v3.1\C6000\rtdx\lib.

Depuse de realizar estos cambios queda configurado el dispositivo para transferir datos a una tasa promedio de 2MB por segundo, estos cambios se deben realizar siempre que cree un nuevo proyecto.