UNIVERSIDAD AUTÓNOMA DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

-PROYECTO FIN DE CARRERA-

Sistema multicanal de captura y codificación de audio con micrófonos digitales y DSPs

Mario Gutiérrez Herzing

Septiembre 2011

Sistema multicanal de captura y codificación de audio con micrófonos digitales y DSPs

Autor: Mario Gutiérrez Herzing
Tutor: Doroteo Torre Toledano

ATVS Grupo de Reconocimiento Biométrico
(http://atvs.ii.uam.es)
Dpto. de Ingeniería Informática
Escuela Politécnica Superior
Universidad Autónoma de Madrid

Resumen

Este proyecto tiene como finalidad la realización de un dispositivo que adquiera audio de cuatro canales en paralelo a través de micrófonos digitales y/o analógicos los codifique a tiempo real reduciendo el número de bits por muestra y manteniendo la máxima calidad posible de la información.

El dispositivo va a constar de varios micrófonos digitales en paralelo para realizar la captura del audio, específicamente voz. Estos micrófonos se van a conectar a un CODEC que va a obtener el audio y lo va a transformar a una señal con formato PCM. Se tiene la capacidad desde el CODEC de poder variar la frecuencia de muestreo de la señal y la longitud de las palabras de formato PCM. El CODEC tiene un módulo periférico que permite transmitir el audio PCM a un DSP desde el cual se va a proceder a realizar la codificación ADPCM de todos los canales de manera independiente a tiempo real simultáneamente.
Para la codificación ADPCM a tiempo real se ha adaptado un código de licencia libre procedente del programa *SOX*. Dicho código se ha adaptado al entorno de programación y se ha ajustado para un funcionamiento a tiempo real.

El proyecto se divide en las partes de interconexión entre los dispositivos, de programación de control de los dispositivos, los canales de comunicación I2C e I2S y de programación del código de codificación a tiempo real de las señales de audio.

Palabras Clave

Micrófonos digitales, CODEC, codificación, decodificación, DSP, ADPCM

Abstract

This thesis has the object to be a device that acquires audio signals from four different channels through digital and/or analog microphones and encodes these signals at real time reducing the number of bits that describes a sample but keeping the maximum possible quality of de speech signal.

The device will have as components some digital microphone connected in parallel mode acquiring audio signals, especially speech signals. These microphones are going to be connected to a CODEC which will transform the audio signals to a digital audio standard format called PCM. The CODEC has the ability to change by programming the sampling frequency and the number of bits of every sampled word. The CODEC will be connected to a DSP which will receive the PCM signals and will encode every channel independently to an ADPCM signal in real time. The ADPCM encoder is taken from the *SOX* system. It has been necessary to adapt this code to the DSP environment and has been adjusted to work at real time for some channels at the same time.

This project has been divided to the interconnection of the devices, the programming of the control and the functionality and the encoding at real time of the audio signals.

Key Words

Digital microphone, CODEC, encoding, decoding, DSP, ADPCM

Agradecimientos

Quiero agradecer al grupo ATVS por darme la oportunidad de poder trabajar en con ellos y realizar mi proyecto fin de carrera, en especial a Doroteo Torre Toledano por tener paciencia conmigo y darme su apoyo en todo momento.
Quiero agradecer al resto de grupo el tiempo y la dedicación que han tenido conmigo y la ayuda que me han prestado cuando la necesitaba.

Quiero agradecer a los profesores de la Escuela Politécnica Superior la formación que me han brindado no solo en lo profesional, sino también en lo personal porque he aprendido tanto conocimientos técnicos como maneras de afrontar problemas en los ámbitos profesionales y personales.

Quiero agradecer a mis padres el apoyo que me han dado todos estos años para que no dejase nunca de aprovechar las oportunidades que me ha dado la vida e intentar siempre sacar lo mejor de cada momento.

Quiero agradecer a mis compañeros, que siempre me han apoyado y ayudado. Todos estos años he conseguido superar todas las barreras porque tenía el apoyo de todos ellos y hemos compartido muchos momentos bueno, que nos han unido para tener una amistad que durará para siempre.

Quiero también agradecer a mis amigos que no han estudiado conmigo como Paloma Sánchez, Juan Serrano, Manuel Gutiérrez, David Martín, Rodrigo Gómez, Santiago Cruz, Guillermo Pascual, Andy Feige, Enrique Gomáriz, Meriem Khatib, Juan Burgos, Daniel Röhrenbach y en especial a mi novia Eva del Olmo todos los momentos fuera de la escuela que han enriquecido y dado sentido a mi vida. También quiero hacer una mención especial a todos mis amigos con los he compartido muchas horas de biblioteca como son Almudena Gilperez, Sergio Sebastián, Álvaro Nussbaum, Daniel Hernández, Santiago Cruz, Juan Burgos, etc. porque he podido soportar todas las horas encerrado estudiando.
También han sido importantes mis amigos a la hora de poder desconectar de la carrera viendo juntos el baloncesto disfrutando y otras muchas veces sufriendo.

Muchas gracias a todos y perdón por haberme olvidado de alguien porque para mí todos mis amigos fuera y dentro de la escuela han sido estos años muy importantes y sé que os conservaré el resto de mi vida en mi corazón.

Índice de Contenidos

Índice de Figuras

Índice de Tablas

Glosario de Acrónimos

- **ADC:** Analog to digital converter
- **BCLK:** Bit Clock
- **CCStudio:** Code Composer Studio
- **CODEC:** Coder-Decoder
- **DAC:** Digital to analog converter
- **DARAM:** Dual Access RAM
- **DAX:** Digital Audio Transmitter
- **DSP:** Digital Signal Processor
- **ESAI:** Enhanced Serial Audio Interface
- **EMAC:** Ethernet Media Access Controller
- **I2C:** Inter IC bus
- **I2S:** Inter IC Sound bus
- **LSB:** Less Significant Bits
- **MMACS:** Million Multiply-Accumulates per Second
- **MIPS:** millones de instrucciones por segundo
- **MMR:** Memory-mapped Registers
- **MSB:** Most Significant Bits
- **PLL:** Phase-Locked Loop
- **RAM:** Random-Access Memory
- **ROM:** Read-Only Memory
- **SARAM:** Single Access RAM
- **SHI:** Serial Host Interface
- **WCLK:** Word Clock

1.INTRODUCCIÓN

1.1 Motivación

El proyecto de "Sistema multicanal de captura y codificación de audio con micrófonos digitales y DSPs" parte de una propuesta de una empresa que tiene como objetivo crear un dispositivo propietario que realice las tareas de adquirir audio por varios canales independientes, codificarlos a tiempo real y transferir los datos a un ordenador. El desarrollo de este dispositivo se divide en tres partes.

La primera parte consiste en evaluar los elementos físicos que pueden componer el diseño hardware de adquisición y codificación, seleccionarlos y crear un prototipo que realice una demostración de las características y propiedades.
La segunda parte del proyecto consiste en crear un interfaz que permita transferir el audio codificado del prototipo a un ordenador para guardarlo y reproducirlo.
La última parte del proyecto consiste en crear una tarjeta que permita alojar y encapsular todos los elementos extraídos de las tarjetas comerciales, seleccionadas para el primer desarrollo, en único dispositivo móvil.

El dispositivo debe ser capaz de poder realizar grabaciones con diferentes parámetros de configuración a tiempo real. Específicamente el interés de esta empresa se centra en la evaluación de los nuevos micrófonos digitales, en particular por una posible menor vulnerabilidad en ruido e interferencias electromagnéticas. Además de la evaluación de los micrófonos digitales se seleccionan dispositivos que sean capaces de adquirir audio de cuatro canales de forma independiente en paralelo, codificarlos y transferirlos a un ordenador a tiempo real. Con este dispositivo se quiere tener la capacidad de grabar audio de alta calidad de múltiples fuentes almacenándolo en equipos reduciendo el tamaño de las muestras de audio con una tasa de 4 a 1 gracias a la codificación ADPCM que se aplica. Esto permite conservar la información reduciendo de forma muy sutil la calidad pero ocupando mucho menos espacio.

1.2 Objetivos

El objetivo del proyecto "Sistema multicanal de captura y codificación de audio con micrófonos digitales y DSPs" es la construcción de la primera parte de un sistema que consiste en un desarrollo de un prototipo mediante la interconexión de distintas placas de pruebas comerciales y la programación de un DSP para realizar la adquisición del audio y la codificación del mismo a tiempo real.

Los elementos que compondrán el prototipo serán micrófonos digitales y/o analógicos que capturarán el audio, un códec que definirá los parámetros del audio a muestrear y un DSP que controlará la configuración del sistema y realizará la codificación a tiempo real.

En primer lugar, para lograr la construcción y el funcionamiento del prototipo se comenzará con la selección de los dispositivos y las tarjetas comerciales que puedan encajar en el sistema final. Se analizarán los productos (códecs y DSPs) comerciales disponibles en el mercado para evaluar los micrófonos digitales en comparación con los micrófonos analógicos. Los micrófonos digitales se presentan como dispositivos que tienen mejores características electromagnéticas que los analógicos frente al ruido ya que disponen de conversores analógico-digitales integrados en el propio chip de silicio junto con la membrana de transducción.
El códec que se seleccionará para el sistema tendrá como características principales la posibilidad de conectar 4 micrófonos digitales independientes en paralelo y de trasferir el audio en dos canales de información distintos. El DSP se escogerá porque deberá tener el mismo tipo de canales de transmisión que el códec y contará con al menos dos puertos para recibir la información de los cuatro micrófonos digitales en paralelo. La arquitectura y frecuencia de funcionamiento del DSP deberán ser suficientes para realizar la codificación de cuatro canales de audio a tiempo real.

En segundo lugar se generarán las interconexiones físicas entre las tarjetas comerciales adquiridas (códec y DSP) y entre los micrófonos digitales y el códec después de haber seleccionado los elementos que compondrán el dispositivo de prueba. También se crearán las interconexiones lógicas entre el códec y el DSP para poder controlar el códec desde el mismo y generar los canales de comunicación de datos que transfieran información tanto desde el códec hacia el DSP como viceversa.

Como último objetivo del desarrollo del sistema se programará un software que realice una demostración de las características de sistema final del proyecto. Esta parte incluye la codificación y decodificación a tiempo real de señales de audio. Para la codificación se empleará la implementación de Microsoft ADPCM. Se obtendrá un código de codificación del programa *SOX* que se adaptará al sistema embebido adaptándolo al entorno para que funcione a tiempo real.
El prototipo construido será capaz de adquirir audio de cuatro canales independientes, ya sean procedentes de micrófonos digitales o analógicos, transferir la información al DSP, donde se codificará a tiempo real y se retransmitirá el audio decodificado al códec para escucharlo por las salidas de audio analógicas.

1.3 Planificación

La duración del proyecto se planteaba como de un año, cosa que se ah cumplido. Durante ese año se han realizado diferentes tareas divididas en dos fases principales. Dichas tareas estaban acordadas así desde el comienzo del proyecto. La primera fase está relacionada con el desarrollo "hardware". En ella se deben hacer informes comparativos sobre los diferentes elementos a usar en el proyecto para proponer finalmente un dispositivo por cada elemento del sistema. También hay que definir la interconexión entre los distintos dispositivos. Por una parte hay que establecer la conexión entre los micrófonos digitales elegidos y el códec y por la otra establecer la conexión entre el códec y el DSP.
En la segunda fase hay que realizar el desarrollo "software". Esto consiste en realizar un programa en código "C" que controle desde el DSP todo el sistema y que realiza la adquisición y codificación del audio. Aquí se implementará la configuración del los dispositivos así cómo la adquisición y la codificación.

- **Primera entrega:** Evaluación "hardware" de los diferentes elementos del proyecto e interconexión de los mismos.
- **Segunda entrega:** Implementación del sistema en el DSP tanto para la configuración como para la adquisición y codificación de audio a diferentes frecuencias de muestreo y longitudes de palabra.

1.4 Organización memoria

La memoria está dividida en varias secciones claramente diferenciadas:

Estado del arte: Se describen los sistemas que actualmente están en el mercado y qué prestaciones tienen para la adquisición de audio. Se describirán los DSPs, CODECs y micrófonos que se pueden adquirir hoy en día para realizar este proyecto.

Diseño Hardware: En esta parte se describen los parámetros Hardware necesarios para la interconexión de los diferentes elementos y las características de los mismos para el funcionamiento final del sistema.

Diseño Software: En la última parte se describe la implementación del código para controlar el códec y el DSP. El código está diseñado para configurar los dispositivos adecuadamente, para adquirir el audio de todos los canales, codificarlo y decodificarlo a tiempo real. El código es configurable para poder realizar la adquisición y la codificación con diferentes longitudes de palabra y frecuencias de muestreo.

2.ESTADO DEL ARTE

2.1 Codificación de audio y voz

La codificación de audio es un procedimiento para representar el audio digitalizado usando el menor número de bits posible manteniendo al mismo tiempo un nivel de calidad aceptable.
Este apartado tiene como finalidad explicar los conceptos de los codificadores de audio y voz, así como sus características para poder compararlos y establecer una clasificación.

2.1.1 Introducción a los codificadores

La estructura un sistema de codificación de audio y voz se puede resumir en el siguiente diagrama de bloques.

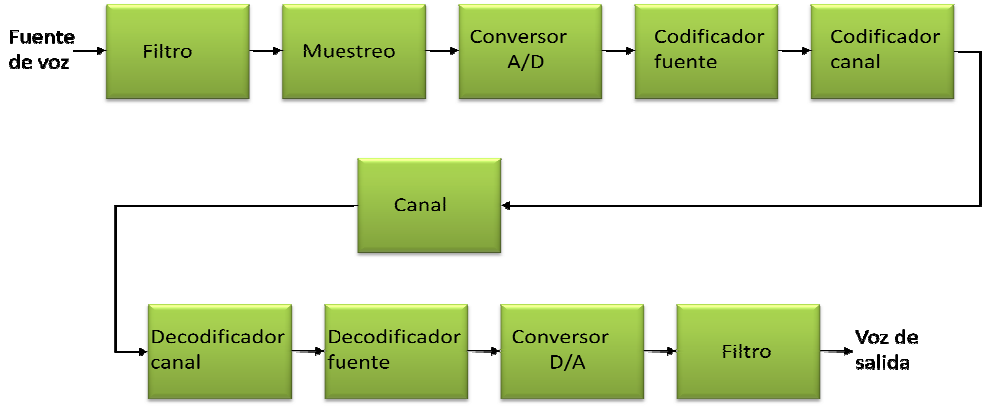


Figura 1. Esquema general de sistema de codificación y decodificación de voz.

Los bloques de filtro, muestro y conversor A/D permiten pasar el audio de analógico a digital. Esta parte se explicará un poco más adelante. El siguiente bloque es el codificador de fuente. Esta parte es la que se encarga de transformar un flujo de datos entrante en un flujo de datos con menor tasa binaria sin perder calidad. Este es el bloque en el que se hace la parte más importante de la codificación de audio. Existen muchos codificadores de audio con características distintas. Vamos a clasificar primero los codificadores de audio más importantes según sus propiedades.
El siguiente bloque es el codificador de canal que en nuestro sistema no será implementado debido a que en el sistema final el audio codificado será decodificado inmediatamente después para recuperarlo y así comprobar su funcionamiento. El bloque del decodificador de fuente también es parte del decodificador de audio. La señal de salida será llevada al códec de nuevo para

hacer primero la conversión digital-analógica y posteriormente filtrada para finalmente obtener una señal de voz que se puede escuchar a través de un altavoz.

Los sistemas de codificación de audio y voz han sido diseñados habitualmente para estar integrados dentro de sistemas de telecomunicaciones como por ejemplo la telefonía. Se considera que con un ancho de banda de la voz de entre 300 y 3400 Hz la inteligibilidad de la voz es casi perfecta. La ley de Nyquist nos dice que para muestrear una señal con un ancho de banda limitado sin sufrir "aliasing" debemos muestrear por lo menos al doble del ancho de banda la fuente. Por ello se ha elegido tradicionalmente una tasa de muestreo de 8 kHz para las señales de voz. Para convertir muestras analógicas a un formato digital usando cuantificación uniforme manteniendo la calidad de la señal es necesario tener más de 8 bits por muestra. Por ello se ha elegido también tradicionalmente una longitud de palabra de 16 bits por muestra. En consecuencia tenemos que la tasa binaria combinando estos dos números es de 128 kbps (Tasa binaria = tasa de muestreo * número de bits por muestra = 8 *16 = 128).
La finalidad de los codificadores de audio y voz es reducir la tasa binaria manteniendo la calidad de la señal o aumentar la calidad perceptible de la señal sin aumentar la tasa binaria.
La tasa binaria a la que se quiere transmitir o almacenar el audio y la voz depende del coste de transmisión o almacenaje, del coste computacional de la codificación y de los requisitos del uso de la información. En casi todos los codificadores de voz el audio decodificado es algo distinto al audio original. Las señales de audio y voz son representadas con una precisión inferior y se elimina la redundancia de las señales. La consecuencia de esto es la pérdida de datos muchas veces humano sea capaz de percibirlo.
Existen varias propiedades que hay que tener en consideración a la hora de caracterizar un codificador de audio.

- **Tasa binaria baja:** cuanto más baja sea la tasa binaria de la señal codificado, más bajo será el ancho de banda necesario para transmitir dicha señal. Al disminuir la tasa binaria dentro de un mismo esquema de codificación de audio y voz disminuimos también la calidad de la señal. Hay que encontrar un equilibrio entre ambas propiedades siempre teniendo en cuenta los requisitos del sistema.
- **Calidad de voz alta:** La calidad de la voz decodificada debe ser aceptable para la aplicación en la que se use la codificación. Existen muchas maneras de reconocer la calidad de una señal de voz. Se valoran por ejemplo la inteligibilidad, la naturalidad, etc. Siempre debe estar en equilibrio con la tasa binaria.
- **Robustez frente a diferentes idiomas:** La técnica con la que se codifica debe ser totalmente independiente del contenido de la información así

como de su forma gramatical o pronunciación. Debe poder funcionar igual para gente de diferentes edades, sexo e idiomas.

- **Buen funcionamiento con señales no habladas:** En comunicaciones telefónicas hay a veces señales que no son solo señales de voz, si no también ruidos o música. Los codificadores deben ser capaces de no generar ruidos extraños cuando codifican señales que no son de voz.
- **Tamaño en memoria reducido y complejidad computacional baja:** El coste de implementación no de ser alto para poder introducirlo en alguna aplicación. Los costes de implementación son el tamaño que ocupa en memoria el código del codificador y los dispositivos lógicos necesarios para poder realizar las operaciones de codificación.
- **Retardo de codificación bajo:** Al existir un procesado de las señales entrantes de voz, es inevitable que se generen retardos en las señales codificadas. Una cantidad de retardo grande no es aceptable en una aplicación a tiempo real donde las fuentes y los destinos de la información se encuentran en ambos extremos de la aplicación.

2.1.2 Clasificación de codificadores de voz

Existen dos criterios para clasificar los codificadores de voz. El primero es clasificarlos por la tasa binaria de las muestras codificadas.

Categoría	Rango tasa binaria
Tasa binaria alta	> 15 kbps
Tasa binaria media	5 a 15 kbps
Tasa binaria baja	2 a 5 kbps
Tasa binaria muy baja	< 2kbps

Tabla 1. Clasificación codificadores por tasas binarias. Tabla extraída de [Wai C. Chu, "*Speech coding algorithms: foundation and evolution of standardized coders*"].

El segundo criterio de clasificación es por técnica de codificación.

- **Codificadores por forma de onda:** Se intenta preservar con la máxima fidelidad posible la forma de onda original de la señal de voz. Se pueden aplicar a cualquier tipo de señal. Se puede medir la calidad de estos codificadores con la tasa señal-a-ruido (SNR). Los codificadores PCM y ADPCM son un ejemplo de este tipo.
- **Codificadores paramétricos:** Se asume que la señal de voz producida esta generada a partir de algún tipo de modelo controlado por unos parámetros. Los parámetros se estiman a partir de la señal de entrada. La calidad de la señal decodificada varía dependiendo del modelo usado. No tiene buenos resultados para señales que no sean de voz. Conservan propiedades estadísticas del espectro de voz. También reciben el nombre de vocoders. En esta categoría entran el codificador de predicción lineal (LPC) y el codificador de predicción lineal de excitación mezclada (MELP).

- **Codificadores híbridos:** Combina métodos de codificadores paramétricos y de forma de onda. En codificación se aplican técnicas de codificación paramétrica. Sin embargo el decodificador actúa como uno de forma de onda. Un ejemplo de este tipo son los codificadores de predicción lineal de excitación de código (CELP).

2.1.2.2 Codificadores de forma de onda

Los codificadores de forma de onda intentan reproducir la forma de onda de la señal de entrada. Se diseñan para poder reproducir una gran variedad de señales. No solo son válidas para señales de voz sino también de audio en general. Se puede realizar la codificación tanto en el dominio del tiempo cómo en el de la frecuencia.
Los codificadores de forma de onda en el dominio de tiempo son PCM (Pulse Code Modulation) y sus dos variantes diferenciales DPCM (Differential PCM) y ADPCM (Adaptive DPCM).

La codificación PCM se refiere al proceso de cuantificación de las muestras de una señal de tiempo continuo para que tanto en tiempo y en amplitud sean representadas en tiempo discreto.
Básicamente el codificador PCM transforma la señal analógica en una secuencia de bits mediante un muestreo uniforme, una cuantificación y una codificación.

El primero de los procesos es el muestreo uniforme, que es coger muestras cada cierto tiempo. El tiempo viene determinado por la tasa de muestreo que como mínimo será de 8 kHz para voz y 44 kHz para el audio en general. La cuantificación es un proceso donde a cada muestra se le asigna un valor discreto. Tendremos tantos valores donde elegir como dos elevado al número de bits por muestra que tengamos. Por ejemplo si tenemos 16 bits de resolución podremos escoger para cada muestra un valor entre dos elevado a 16 que es 65536. Por último codificamos cada muestra con su respectivo número en binario.
El concepto de entrada-salida de un cuantificador se puede describir matemáticamente como una escalera.

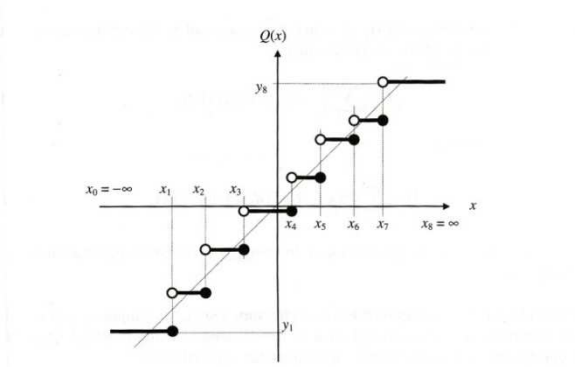


Figura 2. Función de transferencia cuantificador uniforme PCM. Tabla 1. Clasificación codificadores por tasas binarias. Imagen extraída de [Wai C. Chu, "*Speech coding algorithms: foundation and evolution of standardized coders*"].

Una muestra de entrada que se encuentra entre los puntos de referencia x_{k-1} y x_k se representa con el código I_k. La distancia entre los puntos de referencia se denomina valor de paso (Δ). Tendremos tantos puntos de referencia como resolución en bits tengamos.

El proceso de cuantificación es el más complejo de los tres. Existen dos métodos de cuantificación, el uniforme y el no uniforme.

- **Cuantificación uniforme:** El valor de paso es constante para todos los intervalos. Para audio, el número de muestras de amplitudes bajas es más alto que el número de muestras de amplitudes altas. Por ello representar las amplitudes bajas con el mismo número de bits que las amplitudes altas hace que el método de cuantificación uniforme no sea óptimo.
- **Cuantificación no uniforme:** El valor de paso no es constante para todos los intervalos. Se modifican las longitudes de los intervalos para adaptarse a la señal y tener una resolución eficiente.

La codificación PCM mediante la cuantificación no uniforme representan el núcleo de la recomendación ITU-T G.711. En este estándar primero se transforma la señal de entrada con una función sin memoria, monótona, no lineal. Luego se cuantifica uniformemente con los valores transformados y finalmente se realiza la transformación inversa. La función no lineal suele ser logarítmica y con las características de poca pendiente en la zona de amplitudes altas y con mucha pendiente en amplitudes bajas llevando a un efecto de compresión.

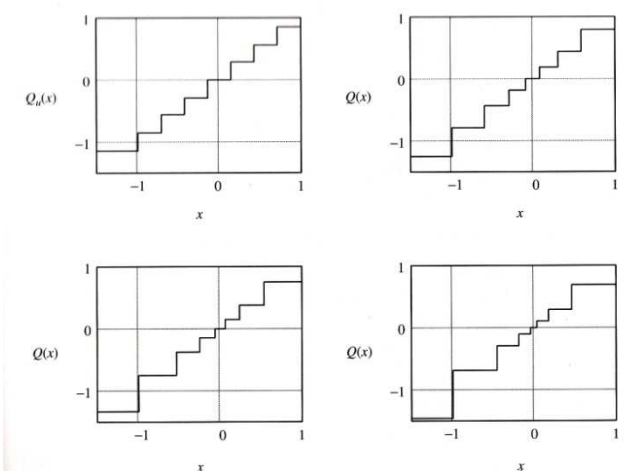


Figura 3. Función de transferencia de cuantificador uniforme de 8 bits (arriba izquierda) y funciones de transferencia de cuantificadores no uniformes. Imagen extraída de [Wai C. Chu, "*Speech coding algorithms: foundation and evolution of standardized coders*"].

En la recomendación G.711 se utilizan dos funciones que representan los dos métodos utilizados en la práctica en las comunicaciones mundiales:

- $\mu\text{-}law: f(x) = A \cdot \frac{\ln(1+\mu \cdot |x|/A)}{\ln(1+\mu)} \cdot sgn(x) \quad , |x| \leq A$

  A representa el valor de pico de la función de entrada y μ es una constante que controla el grado de compresión.

- $A\text{-}law: f(x) = \begin{cases} \frac{A_0 \cdot |x|}{1+\ln(A_0)} \cdot sgn(x) \ , |x| \leq \frac{A}{A_0} \\ \frac{A \cdot (1+\ln(A_0 \cdot |x|/A))}{1+\ln(A_0)} \cdot sgn(x) \ , \frac{A}{A_0} \leq |x| \leq A \end{cases}$

  A_0 es una constante que controla el grado de compresión.

La recomendación G.711 establece un estándar de uso de PCM usando la cuantificación no uniforme con aproximaciones a *μ-law* y *A-law* con valores μ igual a 255 y A_0 igual a 87,6 usando 8 bits por muestra. Así se llega a una tasa de 64 kbps a una frecuencia de muestro de 8 kHz.

Una evolución de la codificación PCM es la PCM diferencial (DPCM). Se basa en cuantificar la señal de predicción del error. Esta idea viene del concepto que las señales de audio y voz están fuertemente correladas ya que no existe una gran variación de valor entre muestras adyacentes. Se parte de la hipótesis de predecir las muestras futuras a partir de las actuales. Así conseguimos una señal de salida con una varianza y un rango dinámico mucho menor.

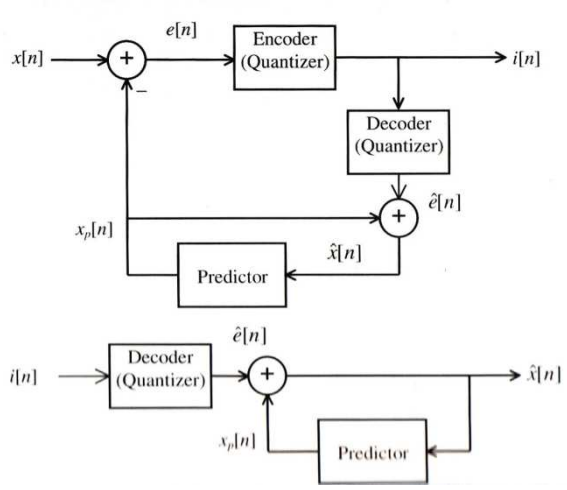


Figura 4. Diagrama de bloque funcional de codificador DPCM (arriba) y decodificador DPCM (abajo). Imagen extraída de [Wai C. Chu, "*Speech coding algorithms: foundation and evolution of standardized coders*"].

El error de predicción se calcula de manera muy sencilla: $e[n] = x[n] - x_p[n]$ donde x_p es la señal de predicción. Codificamos dicha señal para obtener el flujo de datos DPCM.

La predicción se puede realizar mediante dos técnicas muy sencillas. En la primera se decodifica primero el error de predicción. Luego se suma a la predicción de la muestra anterior y se realiza la predicción. La segunda utiliza el modelo AR de predicción que utiliza directamente el error de predicción decodificado como señal de entrada del predictor. Este método tiene un rendimiento más elevado en condiciones de canal ruidoso.

Para obtener un rendimiento óptimo cuando se trabaja con señales de audio no estacionarias, ya que cambian sus propiedades rápidamente con el tiempo, hay que tener un modelo de codificación adaptativo. Por ello se ha llegado a la evolución ADPCM (Adaptive DPCM). Existen dos modelos fundamentales de esta codificación:

- **Forward Gain-Adaptive Quantizer:** puede controlar con precisión el nivel de control de ganancia de la entrada pero también debe transmitir información secundaria al decodificador. La señal de entrada se divide en ventas con N muestras cada una. Con varias muestras de una ventana se puede calcular la ganancia que se aplicará a las muestras de la ventana completa. Se transmite tanto el índice de la ganancia codificado como los N índices de las muestras. Si existe un error en la transmisión, éste solo afectará a una venta, sin que exista propagación de error en el resto de las ventanas. El propósito de este esquema es normalizar la amplitud de las muestras en una ventana para obtener una cuantificación fija óptima.

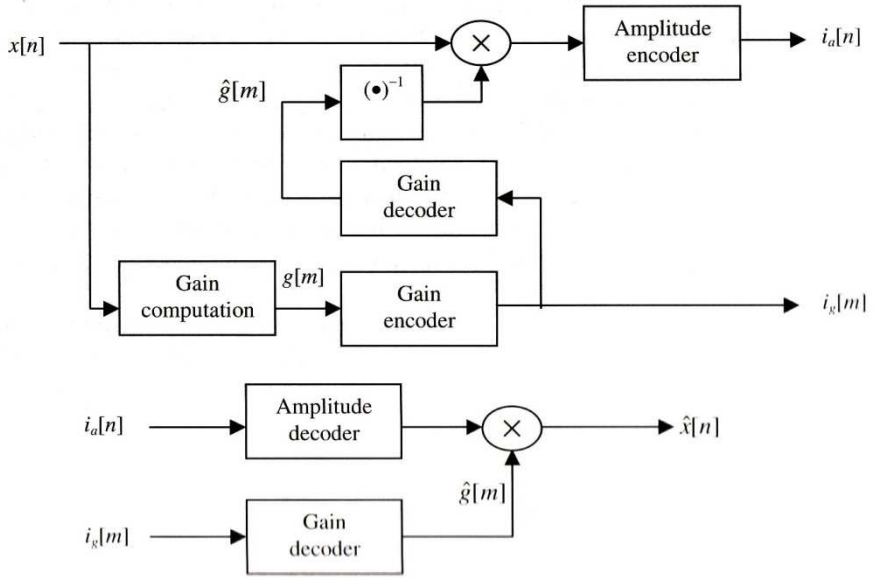


Figura 5. Diagrama de bloques funcional de codificador (arriba) y decodificador (abajo) ADPCM con esquema Forward Gain-Adaptive Quantizer. Imagen extraída de [Wai C. Chu, "*Speech coding algorithms: foundation and evolution of standardized coders*"].

- **Backward Gain-Adaptive Quantizer:** calcula la ganancia en la salida del cuantificador. En este esquema no hay necesidad de trasmitir información secundaria porque se puede derivar en el decodificador Tiene la desventaja que cuando existe un error en la transmisión, éste no solo afecta a la muestra actual, si no a la memoria del decodificador, teniendo una propagación de error en todas las muestras.

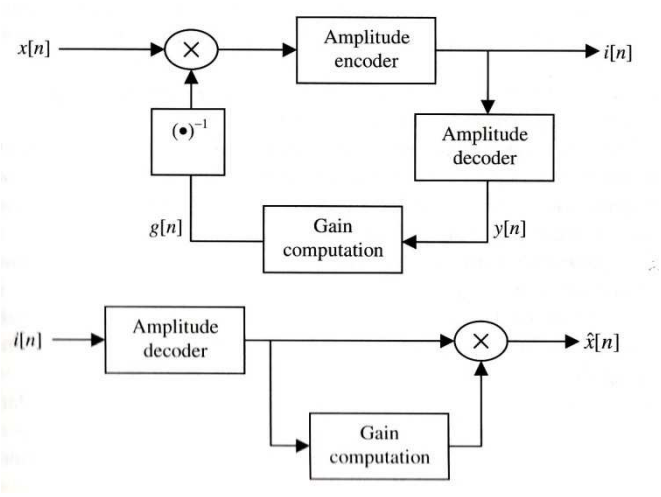


Figura 6. Diagrama de bloques funcional de codificador (arriba) y decodificador (abajo) ADPCM con esquema Backward Gain-Adaptive Quantizer. Imagen extraída de [Wai C. Chu, "*Speech coding algorithms: foundation and evolution of standardized coders*"].

El funcionamiento de un codificador ADPCM combina prestaciones de la predicción de DPCM con las características de adaptación explicada anteriormente. La codificación ADPCM puede realizar el proceso de adaptación en el predictor, en el cuantificador o en ambos.

Cuando se usa el método de Forward Gain-Adaptive Quantizer tendremos que transmitir en el flujo de datos de salida ADPCM tanto la información cuantificada como el índice de predicción y de ganancia. En el codificador necesitaremos un número de muestras de cada ventana para calcular los parámetros del predictor. Luego estos parámetros se cuantifican y se transmiten ($i_p[n]$). Cuando se hace uso del análisis de predicción lineal, cosa muy común en las implementaciones comerciales de ADPCM, los coeficientes generados se tendrán que cuantificar y transmitir. Como se trata de una codificación diferencial trabajaremos con ventanas de muestras de error de predicción en vez de con muestras reales. Haremos uso de algunas de estas muestras para calcular la ganancia de la ventana. Este parámetro también se cuantifica y se transmite ($i_g[n]$). Finalmente cuantificaremos las muestras de error de predicción, a las que previamente hemos aplicando la ganancia calculada previamente. Estas muestras serán las que transmitamos junto a los dos índices.

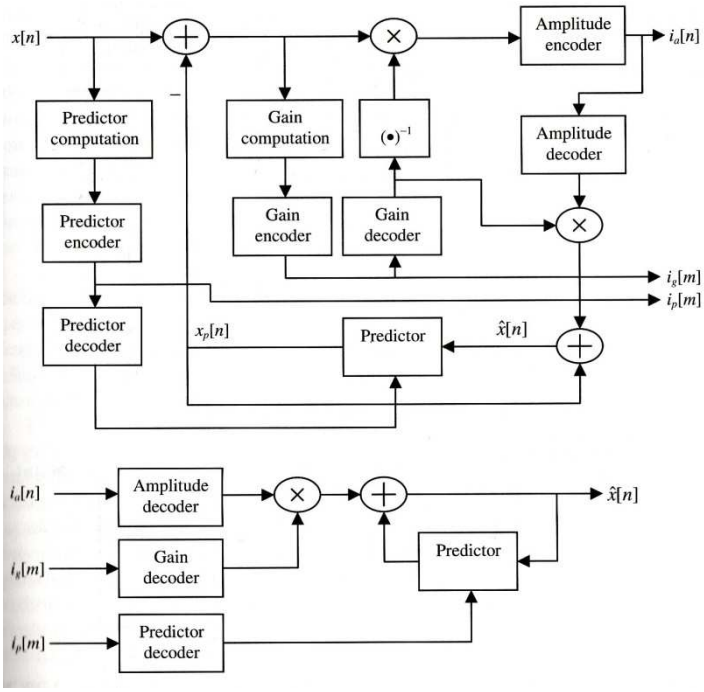


Figura 7. Codificador (arriba) y decodificador (abajo) de un cuantificador con esquema Forward-Adaptive ADPCM. Imagen extraída de [Wai C. Chu, "*Speech coding algorithms: foundation and evolution of standardized coders*"].

También se puede hacer uso del método Backward Gain-Adaptive Quantizer. El proceso de codificación es muy similar pero aquí no hay necesidad de transmitir los índices de ganancia ni de predicción porque se calculan una vez cuantificadas las muestras de error de predicción.

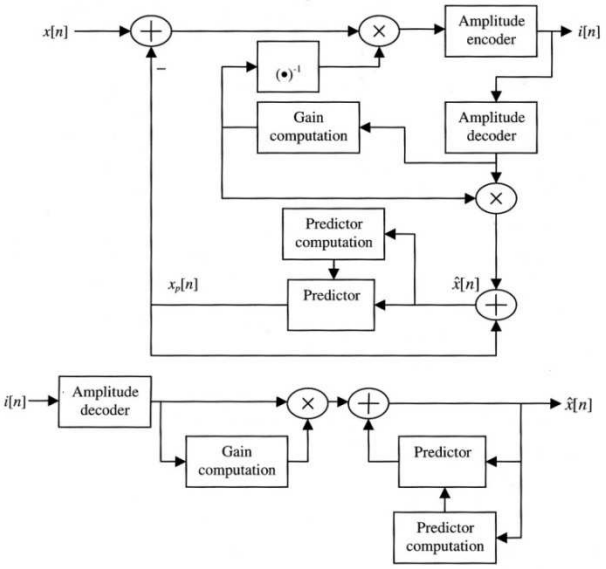


Figura 8. Codificador (arriba) y decodificador (abajo) de un cuantificador con esquema Backward-Adaptive ADPCM. Imagen extraída de [Wai C. Chu, *"Speech coding algorithms: foundation and evolution of standardized coders*"].

El retardo en este esquema es menor y suele implementar se sistemas donde el retardo sea el factor crítico. Tiene la gran desventaja que cuando existe un error de predicción éste se propaga a todas las muestras, siendo así un sistema de

codificación muy sensible a los errores de transmisión. La codificación ADPCM usando el método Forward Gain-Adaptive Quantizer tiene el problema de tener un retardo mayor pero que cuando existe un error en la transmisión, éste solo afecta a las muestras de una ventana. El tamaño de la ventana es el que determina la cantidad de retardo que sufre el sistema. Éste método es el más usado en las implementaciones reales como puede ser la recomendación ITU-T G.726 ADPCM.

2.1.2.1 Codificadores paramétricos

Los codificadores paramétricos de voz codifican una señal de audio destinada a ser escuchada por humanos. Por tanto se hace uso de los conocimientos de producción de voz humana basándose en un modelo de producción de la misma sin necesidad de usar información de la forma de onda de la señal acústica. La calidad de la voz no es tan alta como en la codificación de forma de onda pero se puede codificar a tasas mucho más bajas. Es muy útil para comunicaciones con baja tasa de transmisión. Los codificadores paramétricos están basados en modelos acústicos de generación de voz a través de la excitación del tracto vocal.

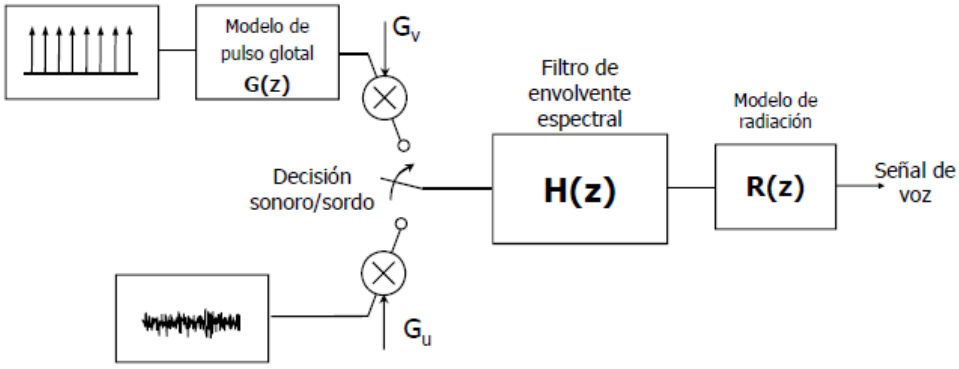


Figura 9. Modelo simplificado de producción de voz.

En la producción de voz existen dos situaciones, cuando tenemos sonidos sonoros y cuando hay sonidos sordos (ejemplo: fricantes como la s y la f). Durante la generación de sonidos sonoros se hace uso de la excitación periódica y en caso de que se generen sonidos sordos, se aplica la excitación de ruido blanco.
El modelo de pulso glotal, G(z), aplica una estimación de cómo son los pulsos glotales (en forma de onda). La salida de este sistema, a cuya entrada tenemos un tren de impulsos separados el periodo fundamental, será un tren de pulsos glotales a la frecuencia fundamental deseada.
A partir de este modelo se han realizado diversas codificaciones añadiéndole el modelo de predicción lineal para transmitir los parámetros de las señales de audio (coeficientes de filtro de síntesis, ganancia de la fuente de excitación, información sobre si los segmentos en los que está dividido son sordos o sonoros y el periodo fundamental).

2.1.3 Codificadores de voz estándar

Vamos a resumir brevemente los diferentes tipos de codificación de voz estandarizados y enunciar su ámbito de aplicación.

Año	Nombre	Tasa binaria [kbps]	Aplicación
1972	ITU-T G.711 PCM	64	Propósito general
1984	FS 1015 LPC	2.4	Comunicaciones seguras
1987	ETSI GSM 6.10	13	Radio móvil digital
1990	ITU-T G.726 ADPCM	16, 24, 32, 40	Propósito general
1990	TIA IS54 VSELP	7.95	USA TDMA telefonía móvil digital
1990	ETSI GSM 6.20 VSELP	5.6	Sistema GSM celular
1990	RCR STD-27B VSELP	6.7	Sistema celular japonés
1991	FS10106 CELP	4.8	Comunicaciones seguras
1992	ITU-T G.728	16	Propósito general
1993	TIA IS96	8.5, 4, 2, 0.8	USA CDMA telefonía móvil digital
1995	ITU-T G.723.1 ACELP	5.3, 6.3	Comunicaciones multimedia
1995	ITU-T G.729 CS-ACELP	8	Propósito general
1996	ETSI GSM EFR ACELP	12.2	Propósito general
1996	TIA IS641 ACELP	7.4	USA TDMA telefonía móvil digital
1997	FS MELP	2.4	Comunicaciones seguras
1999	ETSI AMR-ACELP	12.2 – 4.75	Propósito general telecomunicaciones

Tabla 2. Codificadores de voz estándar. Tabla extraída de [Wai C. Chu, "*Speech coding algorithms: foundation and evolution of standardized coders*"].

Para poder realizar una comparación mediante los parámetros característicos de los codificadores de voz presentamos una gráfica en el cual están representados varios de los estándares más importantes.

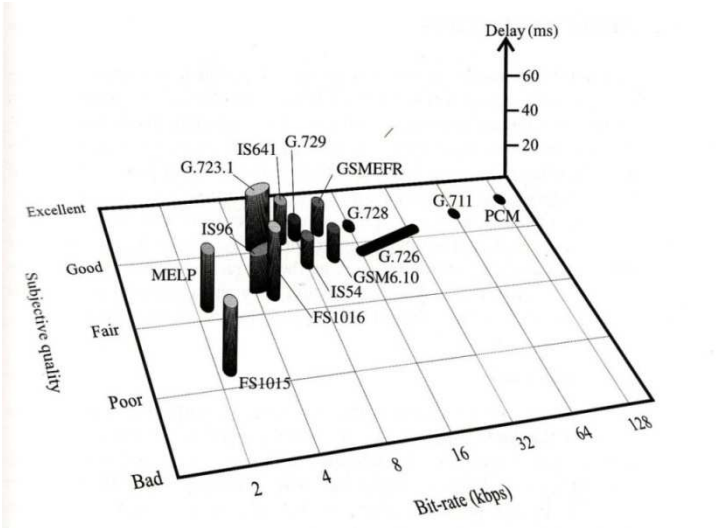


Figura 10. Comparativa entre codificadores de voz estándar. Imagen extraída de [Wai C. Chu, "*Speech coding algorithms: foundation and evolution of standardized coders*"].

Podemos observar como las codificaciones de voz basadas en forma de onda tienen calidad buena a excelente y con un retardo muy bajo. Esto es algo muy bueno para una aplicación a tiempo que quiere como propósito hacer grabaciones y guardarlas en un sistema de red. Por otra parte tienen como punto negativo las tasa binarias más altas de los codificadores de voz. Para la aplicación de este proyecto es secundario tener esta tasa binaria ya que no vamos a transmitir la información codificada por ningún canal de comunicaciones con ancho de banda muy limitado. Se transmitirá por un cable USB desde un DSP a un PC.

2.2 Micrófonos

El micrófono es un elemento que transforma una señal acústica de audio en una señal eléctrica. Existen muchos tipos diferentes de micrófonos que utilizan diferentes técnicas para realizar la transformación de la señal acústica en la señal eléctrica.
Edison y Hughes inventaron el micrófono moderno en el año 1878. Era un micrófono de carbono. Fue el primero micrófono robusto y significó un cambio drástico en el sistema de telefonía ya que se podía construir dispositivos telefónicos más robustos que sus predecesores.
Todos los micrófonos se basan en la misma propiedad de transducir una variación de presión a una variación de tensión y varían en el concepto físico del procedimiento.

Tipos de micrófonos:

- **Condensador o capacitivo:** Su funcionamiento se basa en dos placas metálicas paralelas que funcionan como un condensador. Al existir un cambio de presión acústica una de las placas metálicas se mueve y hace variar la capacitancia. Existe dos maneras de realizar la traducción de presión a voltaje. La primera consiste en conectar a las placas un voltaje constante y medir la variación de corriente inducida por la variación de la capacitancia. La segunda manera consiste en usar la variación de la capacitancia para generar una señal modulada a una determinada frecuencia en u oscilador. Este tipo de micrófonos están considerados los más baratos y comunes.

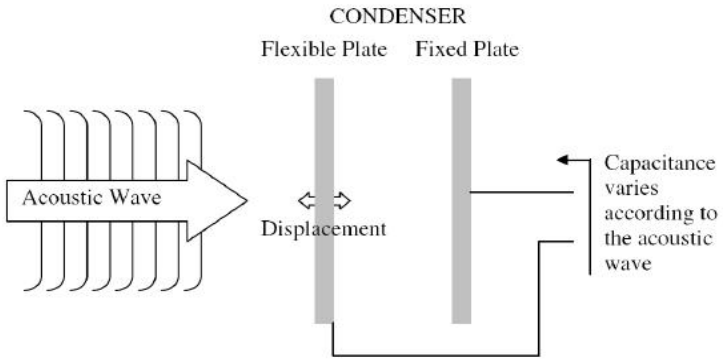


Figura 11. Esquema funcionamiento micrófono de capacitivo. Imagen extraída [Doroteo T. Toledano, Joaquin Gonzalez-Rodriguez and Javier Ortega-Garcia, "*Voice Device*"].

- **Dinámico o por inducción:** El principio físico de la transducción es diferente que en el micrófono de condensador o de capacitancia. Se aplica el principio físico de inducir un voltaje mediante la variación de un campo magnético. La estructura se constituye acoplando una pequeña bobina que está en un campo magnético generado por un imán permanente a un diafragma. Cuando hay sonido el diafragma se mueve y en consecuencia también se mueve la bobina. Ese movimiento de bobina induce un cambio del campo magnético y genera una tensión relacionada directamente con el sonido.
- **Carbono:** Se trata de un recipiente lleno de polvo de carbono cerrado por una membrana metálica a un lado y una placa metálica al otro. Su funcionamiento consiste en que la membrana vibra con el sonido aplicando más o menos presión al polvo de carbono. Esta sustancia cambia su resistencia eléctrica dependiendo de la presión aplicada. Al tener un elemento con una resistencia variable se puede obtener una tensión variable dependiendo de si era una resistencia con una corriente fija. Este tipo de micrófono se usaban mucho en teléfonos hasta que fueron reemplazados por los micrófonos capacitivos.
- **Piezoeléctrico:** Se basan en las propiedades de los materiales piezoeléctricos. Cuando se le aplica una presión, en nuestro caso acústica, producen un voltaje.
- **Silicio:** También se les conoce por MEMS (Micro Electrical-Mechanical System). Se trata de micrófonos donde la membrana está grabada directamente sobre un chip de silicio. La mayoría de este tipo son variaciones de los micrófonos de capacitancia. Pueden llegar a tener un tamaño muy reducido. Este tipo de micrófonos son muy prácticos para integrarlos en diseños de circuitos eléctricos.

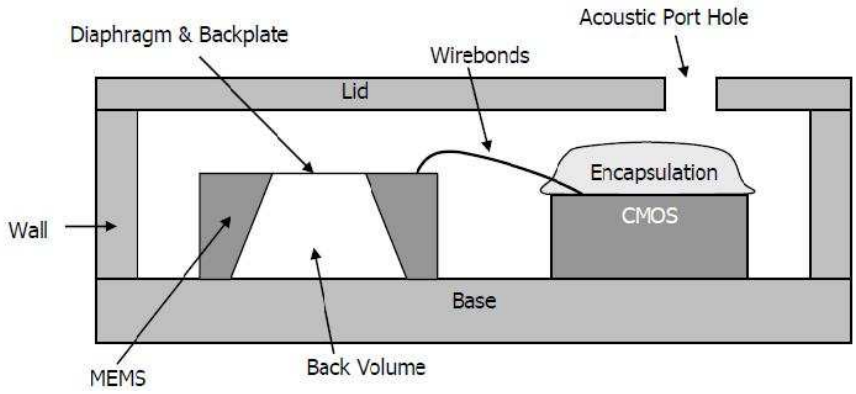


Figura 12. Esquema sección lateral micrófono digital. Imagen extraída [Knowles Acoustics, "*SiSonic Design Guide – Application Note 16*"].

Para poder realizar comparaciones entre micrófonos del mismo u otro tipo hay que definir las propiedades de más importantes con las que caracterizamos estos dispositivos. Se van a explicar de forma breve algunos de estos conceptos.

- **Sensibilidad:** mide la eficiencia en la transducción, es decir cuánto voltaje genera para una entrada de presión específica. Se mide en milivoltios por Pascal a 1 MHz.
- **Respuesta en frecuencia:** es la variación de la sensibilidad en función de la frecuencia de una señal.
- **Característica direccional:** La direccionalidad es la variación de la sensibilidad dependiendo de la dirección de la fuente de sonido. Se suele especificar en forma de modelo de direccionalidad.
- **No linealidad:** En el caso ideal, el micrófono debería ser un transductor lineal y por ello un tono puro de audio debería generar un voltaje sinusoidal a la misma frecuencia. Como los micrófonos no son exactamente lineales, no solo se produce un volteje sinusoidal si no también una serie de armónicos. La medida de no linealidad más extendida es la distorsión harmónica total (THD) que es la tasa entre la potencia de los armónicos producidos y la potencia del voltaje de la sinusoide producida a una cierta frecuencia de entrada.
- **Característica límite:** indica el nivel de presión de sonido (SPL) que se puede transducir sin distorsionar.
- **Ruido inherente:** Un micrófono produce voltaje aun cuando no hay ningún sonido debido al ruido inherente generado por el mismo.
- **Rango dinámico:** Son las tasas máximas y mínimas de nivel de sonido que el transductor puede convertir.

Los micrófonos de silicio son muy nuevos y todavía no están muy extendidos. Son económicos, de reducido tamaño y encapsulados para montajes electrónicos. Dentro de este tipo de micrófonos existen dos campos: los micrófonos con salida analógica y con salida digital. Los primeros son micrófonos de condensador pero

construidos sobre silicio. Así se pueden integrar con mayor facilidad dentro de circuitos eléctricos. Los segundos tienen como gran diferencia que la salida es digital. El proceso de conversión analógico a digital se realiza en el propio chip del micrófono evitando usar un códec posteriormente y así ahorrando tiempo y espacio físico para la conversión y reduciendo el ruido del sistema considerablemente.

2.2.1 Micrófonos analógicos

Los micrófonos analógicos que vamos a describir en esta sección son del tipo silicio, también llamados "MEMS". Suelen estar construidos en encapsulados para montaje superficial para mayor comodidad a la hora de integrarlos en un circuito impreso para alguna aplicación. Su reducido tamaño permite poder colocarlos en el mejor lugar acústico para la aplicación. Las dimensiones mínimas rondan los 3mm de largo, 2 mm de ancho y 0.88 mm de altura. La directividad de este tipo de micrófonos es omni-direccional para que no importe el ángulo de la fuente de sonido. Se alimentan con tensión entre 1.8 y 3.6 Voltios habitualmente. Las tasas de señal a ruido suelen rondar entre 50 dB y 60 dB.
Existen varias empresas importantes que fabrican estos productos para un uso comercial como son "Knowles", "Analog Devices", "Memstech" y "Akustica". Todas estas empresas tienen micrófonos con características muy similares. Lo más importante es que la respuesta en frecuencia sea lo más plana posible en el rango que se desee.

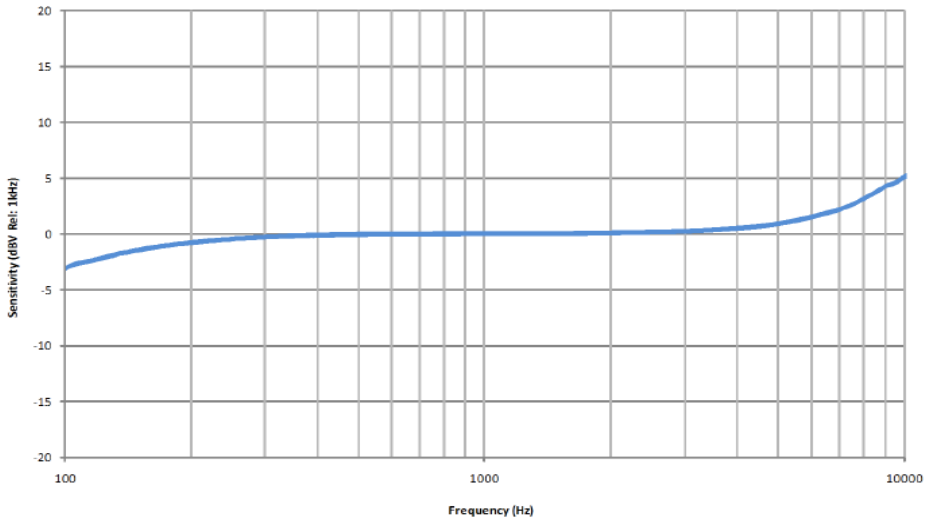


Figura 13. Respuesta en frecuencia de micrófono SPM0408HE5H de Knowles. Imagen extraída de [Knowles Acoustics, "*SPM0408HE5H Amplified "Mini" SiSonic Microphone Specification with enhanced RF protection- Halogen Free*"].

En la figura vemos una respuesta en frecuencia de un micrófono de la marca "Knowles". Vemos que es bastante plano desde los 100 Hz hasta los 1000Hz. Este rango abarca prácticamente todas las frecuencias del marco propio del audio hablado, es decir de la voz humana.

Las estructuras de todos los micrófonos de este tipo son iguales. Todos tienen un diafragma donde se realiza la transducción, un amplificador de salida y un adaptador de impedancias.

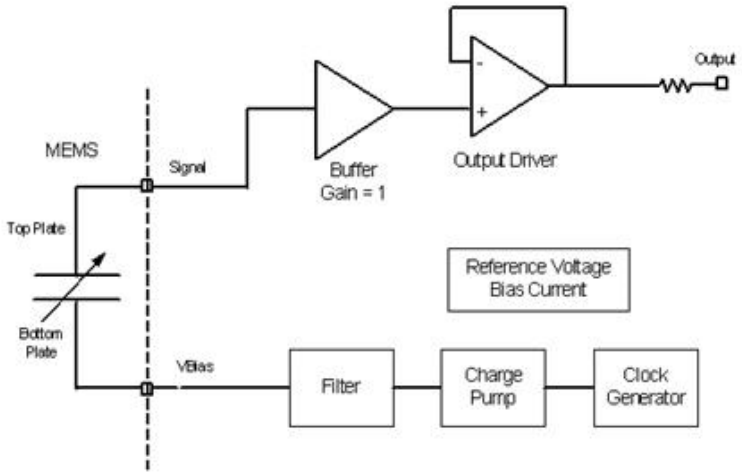


Figura 14. Diagrama de bloques de micrófono SPM0408HE5H de Knowles.

La mayoría de los micrófonos de silicio tienen un amplificador con ganancia unitaria. Esto hace que se suelan tener que utilizar amplificadores externos antes del codificador para tener una señal suficientemente energética. Existe un tipo de micrófono que tiene la etapa de amplificación incluida en el propio chip. La empresa "Knowles" fabrica este dispositivo con el nombre "SPM0408HE5H". Es un micrófono con salida analógica y un amplificador de ganancia variable integrado en el chip de silicio. Tiene las mismas dimensiones físicas y respuesta en frecuencia que los demás micrófonos. Al tener esta etapa de amplificación no es una etapa de amplificación antes del códec. La diferencia estructural es que el encapsulado se construye con 4 pines diferentes. Uno es para la alimentación, otro es para la tierra, uno para los datos de salida y el último es para determinar la ganancia del amplificador en función del valor de una resistencia y de un condensador que se conecten.

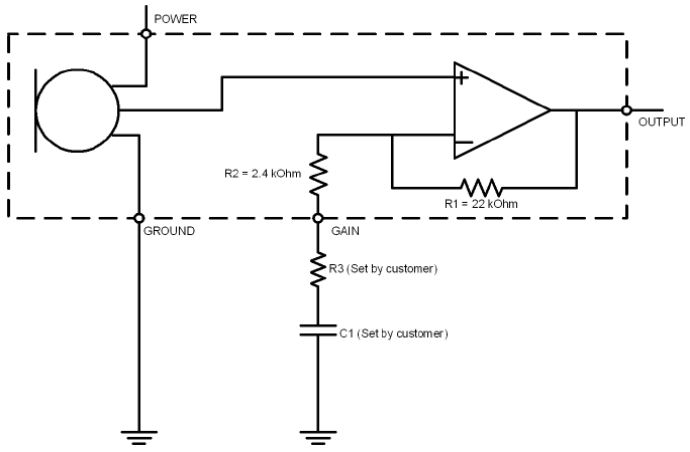


Figura 15. Circuito eléctrico micrófono digital SPM0408HE5H de Knowles. Imagen extraída de [Knowles Acoustics, "*SPM0408HE5H Amplified "Mini" SiSonic Microphone Specification with enhanced RF protection- Halogen Free*"].

La ganancia del amplificador depende de un condensador y una resistencia que se pueden calcular fácilmente.

Ganancia	Método terminación de pin ganancia
0 dB	Conectado directamente al pin de salida
20 dB	Conectado a tierra a través de un condensador de capacidad 0.47µF
Variable	Conectado a tierra a través de un condensador de capacidad C y una resistencia de resistividad R

Tabla 3. Posibilidad de ganancias en micrófonos analógicos SPM0408HE5H de Knowles. Tabla extraída de [Knowles Acoustics, "*SPM0408HE5H Amplified "Mini" SiSonic Microphone Specification with enhanced RF protection- Halogen Free*"].

Para calcularse la capacidad del condensador y la resistividad de la resistencia tenemos dos fórmulas:

$$G = 1 + \{R_1/(R_2 + R_3)\}$$

$$Ganancia(dB) = 20 * \log(G)$$

2.2.2 Micrófonos digitales

A diferencia de los micrófonos analógicos, los digitales tienen la característica de llevar un conversor analógico-digital en el propio dispositivo de transducción para tener en la salida una señal digitalizada y en formato PWM. Esta mejora de prestación en la estructura de los micrófonos de silicio permite reducir el ruido del sistema considerablemente ya que no hay ni cables ni conexiones entre el codificador y el micrófono y permite eliminar directamente dicho componente lo cual nos deja la oportunidad de crear dispositivos más pequeños y con menos consumo.
Existen varias empresas que venden estos dispositivos como son "Knowles", "Akustica" y "Analog Devices". Todos los modelos de las diferentes compañías tienen características similares.
La estructura en los todos los casos siempre es la misma.

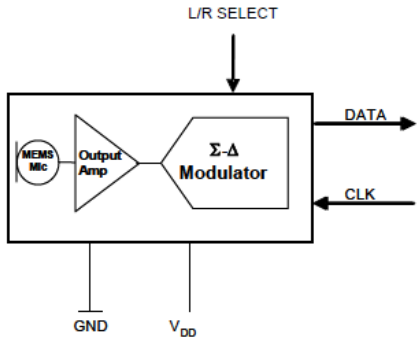
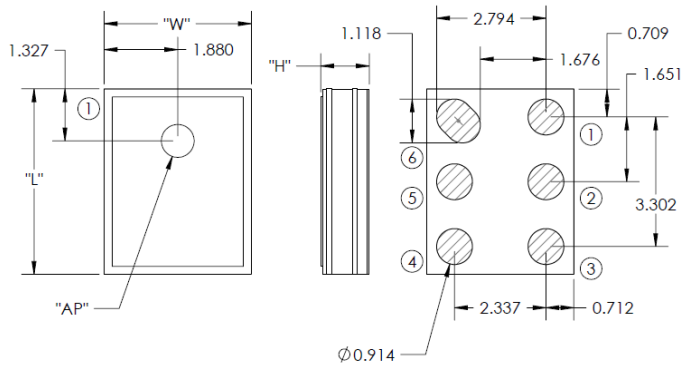


Figura 16. Diagrama conexiones micrófono digital.

Tenemos como primer elemento el diafragma fabricado en silicio representado por MEMS. A continuación siempre habrá un amplificador y justo después un modulador "sigma-delta", que permite transmitir los datos muestreados por un canal de un único bit.
Los pines de los micrófonos siempre son iguales aunque tienen diferentes distribuciones dependiendo del fabricante. Siempre hay un pin de alimentación (VDD), al cual se le puede aplicar una tensión entre 1.8 y 3.6 V. también hay dos pines de masa, u pin de datos y un pin de entrada de reloj para el modulador.

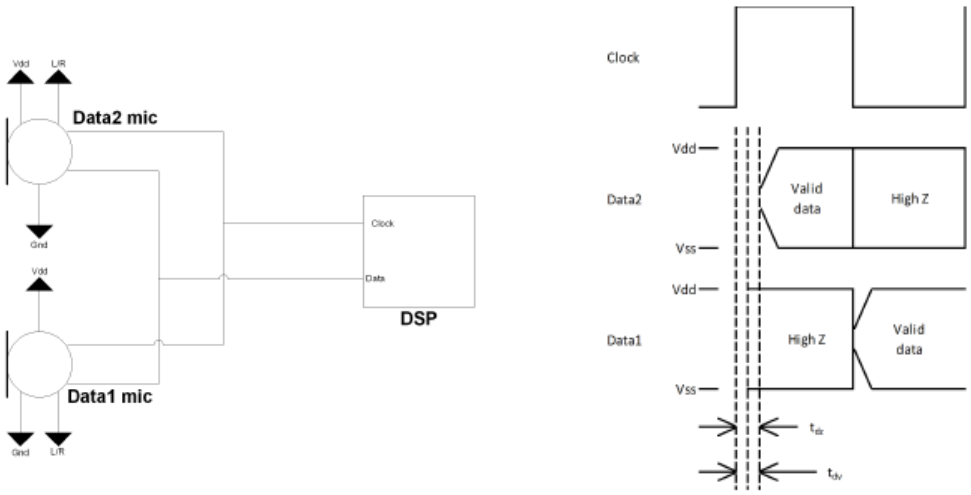


PIN OUTPUT	
PIN #	FUNCTION
1	GROUND
2	LEFT/RIGHT
3	GROUND
4	CLOCK
5	DATA
6	POWER (Vdd)

ITEM	DIMENSION	TOLERANCE	UNITS
LENGTH (L)	4.720	±0.100	mm
WIDTH (W)	3.760	±0.100	mm
HEIGHT (H)	1.250	±0.100	mm
ACOUSTIC PORT (AP)	Ø0.840	±0.100	mm

Figura 17. Diagrama conexiones físicas micrófono digital Knowles. Imagen extraída de [Knowles Acoustics, "*SPM0405HD4H-WB Topics Digital "Mini" SiSonicTM Microphone Specification - Halogen Free*"].

El último pin es de selección izquierda-derecha. Todos estos micrófonos están preparados para configuraciones de dos micrófonos en paralelo y tener audio estéreo. Uno de los pines izquierda-derecha se conecta al pin de masa y el otro al pin de alimentación. Así uno de los dos tendrá la información del canal derecho y el otro del canal izquierdo. La información en esta configuración se transmite enviando los datos de un canal en el flanco de bajada y los datos del otro canal en el flanco de subida.



LABEL	L/R	Drives data after	High-Z after
Data2	High	Rising clock edge	Falling clock edge
Data1	Low	Falling clock edge	Rising clock edge

Figura 18. Esquema interconexión micrófonos digitales con aparato receptor. Imagen extraída de [Knowles Acoustics, "*SPM0405HD4H-WB Topics Digital "Mini" SiSonicTM Microphone Specification - Halogen Free*"].

Las características de todos los micrófonos son muy similares. Son valores para condiciones de temperatura a 25ºC, a 1.8V y con un reloj a 2.4MHz:

Característica	Condición	Valor Min.	Valor Típico	Valor Max.	Unidades
Sensibilidad	1 kHz, 1 Pa	-29	-26	-23	dBFS
SNR	-	56	-	61	dB
Alimentación	-	1.8		3.6	V
Frecuencia Reloj	-	1		4	MHz

Tabla 4. Características generales de los micrófonos digitales.

La respuesta en frecuencia es una característica importante porque muestra la linealidad de la captura de audio en el rango de frecuencias. A continuación se muestra la respuesta en frecuencia de dos micrófonos con salida de datos digital. Podemos observar como todas las curvas son prácticamente planas en el rango entre 300 y 10.000Hz, es decir el rango de voz.

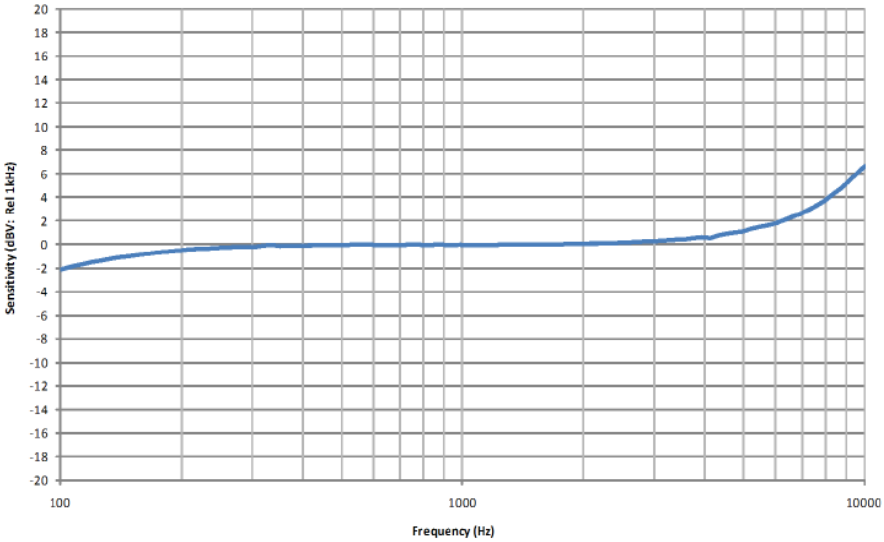


Figura 19. Respuesta en frecuencia de micrófono "Knowles". Imagen extraída de [Knowles Acoustics, "*SPM0405HD4H-WB Topics Digital "Mini" SiSonicTM Microphone Specification - Halogen Free*"].

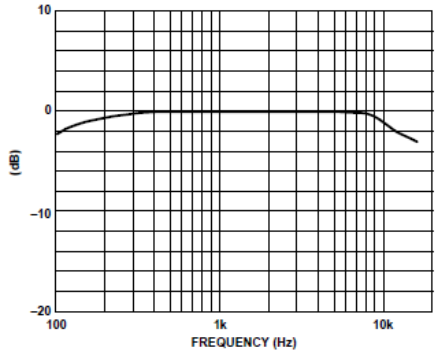


Figura 20. Respuesta en frecuencia de micrófonos de Analog Devices. Imagen extraída de [Analog Devices, "*Omnidirectional Microphone with Bottom Port and Digital Output ADMP421*"].

2.3 Códecs Audio

La palabra códec es una abreviatura de codificador-decodificador. El códec es un dispositivo electrónico que une las funciones de codificador y decodificador. Las funciones que realizan son muy simples. Convierten una señal analógica de audio o vídeo en una señal de audio o vídeo digital. Transforman la señal en primer lugar y luego la envían a través de un canal de comunicaciones y en un formato específico para que sea procesado en un dispositivo electrónico digital como por ejemplo un DSP. También reciben señales digitales de los dispositivos que manejan señales digitales y decodifican las señales para volver a convertirlas en analógicas y hacer visibles o audibles por los puertos pertinentes como salidas de línea de audio o de video. En definitiva un CODEC es un dispositivo digital que está compuesto por un codificador analógico-digital, un codificador digital-analógico, puertos de entrada y

salida específicos y una ruta de datos digital que une la salida del conversor analógico-digital con un puerto de salida y otra ruta de datos que une un puerto digital con el conversor digital-analógico.

2.3.1 Aplicaciones

Las aplicaciones más comunes de los CODECs de audio suelen ser en cámaras de vídeo, cámaras fotográficas, tabletas digitales, teléfonos móviles, ordenadores portátiles, dispositivos portátiles de videojuegos, dispositivos de navegación y sistemas de audio portátiles. En todos estos sistemas existe algún micrófono o sensor de alguna fuente analógica que ha de ser transformada para su almacenamiento, tratamiento o transmisión. Un teléfono por ejemplo recoge una señal de voz hablada, que ha de ser transformada a una señal digital para poder ser procesada y transmitida correctamente.

2.3.1 Clasificación

Como se ha dicho un poco más arriba existen CODECs tanto de audio como de vídeo. Sus funciones y estructuras son las mismas pero con la diferencia del tipo de señal.
En este apartado nos centraremos en describir un CODEC de audio. Éstos se pueden clasificar por el tipo de características. La primera y más importante es el número de canales. Tendremos CODECs de un solo canal, llamados "Mono CODECs", de dos canales, llamados "Stereo CODECs" y de múltiples canales, llamados "Multi channel CODECs".
Dentro de cada categoría tendremos varios parámetros con diversos valores. Los valores más comunes dentro de los dispositivos comerciales son los siguientes:

- **Rango frecuencias de muestreo:** desde 8kHz hasta 192kHz
- **Resolución salida ADC:** 16, 20, 24 o 32 bits.
- **Tipo y número de entradas y salidas analógicas:** entradas/salidas de línea, micrófonos analógicos y digitales, líneas de alta amplificación, entrada/salida específica de tipo "Jack", etc.
- **Tipo de señales de control:** canal de comunicaciones de control I2C o SPI.
- **Tipo de interfaz de comunicaciones con otros dispositivos:** I2S, TDM, etc.
- **Rango frecuencias de reloj interno:** creación de frecuencias internas a partir de osciladores desde 6 hasta 60MHz.
- **Tasas señal a ruido (SNR) de los ADC y DAC:** desde 85dB hasta 100dB.
- **Control de volumen digital o analógico en diferentes etapas:** amplificación de señal entrante previa a la conversión, control digital de volumen, control en las etapas de salidas posteriores al DAC y controles automáticos de ganancia.
- **Cantidad de mezcladores de canales:** mezcladores con amplificadores de diferentes canales de entrada y salida.

- **Sistema digital de introducción de efectos y filtros:** sistemas digitales de manipulación de señales para introducir efecto de filtrado paso alto, paso bajo, etc.

2.3.3 Características

Aparte de los parámetros configurables de los CODES existe una estructura común en todos ellos. Siempre tienen por lo menos una entrada analógica y algunos una entrada digital. Siempre existe un ADC, un DAC, sistema de control de ganancia a la entrada, sistema de volumen a la salida, interfaz de audio digital y un sistema de control digital. Estos son los elementos mínimos para poder definir un CODEC.
La mayoría de los CODECs están basados en registros, es decir que se pueden programar sus funciones escribiendo en los registros los valores adecuados. Por ello es fundamental el sistema de control. En esta sección se va a explicar el funcionamiento de los CODECs de audio multicanal de bajo consumo. Los CODECs de audio de un solo canal son idénticos pero con una sola ruta de audio de entrada y salida. Los CODECs multicanal son básicamente varios CODECs mono canal en paralelo con los canales de audio de entrada y salida multiplexados. El CODEC de audio se diferencia del CODEC de vídeo por la naturaleza la fuente. Todos los CODECs actuales tienen como ADC un modulador delta-sigma (Δ-Σ). Mediante este tipo de moduladores se pueden obtener conversores con la mayor resolución posible.
Los módulos de ADC están compuestos por un modulador delta-sigma y un filtro de diezmado. Antes del propio ADC tendremos un controlador de ganancia analógica de entrada.

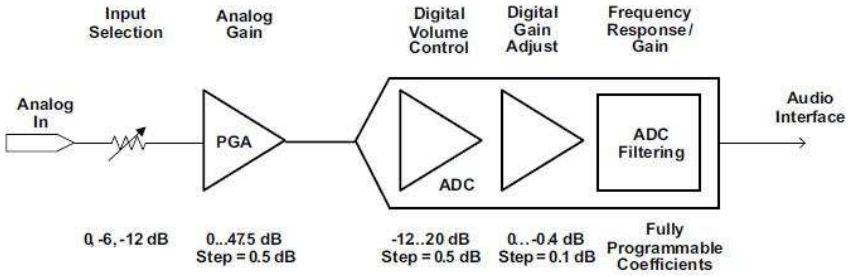


Figura 21. Diagrama de bloques de ruta de entrada de CODEC.

Tendremos dos tipos de rutas de datos de entrada: la de los datos analógicos (procedentes de micrófonos con salida analógica) y la de los datos procedentes de micrófonos digitales. La diferencia reside en que las entradas analógicas recorren toda la ruta de datos y las entradas procedentes de micrófonos digitales tienen acceso directo a filtro paso bajo del módulo ADC.

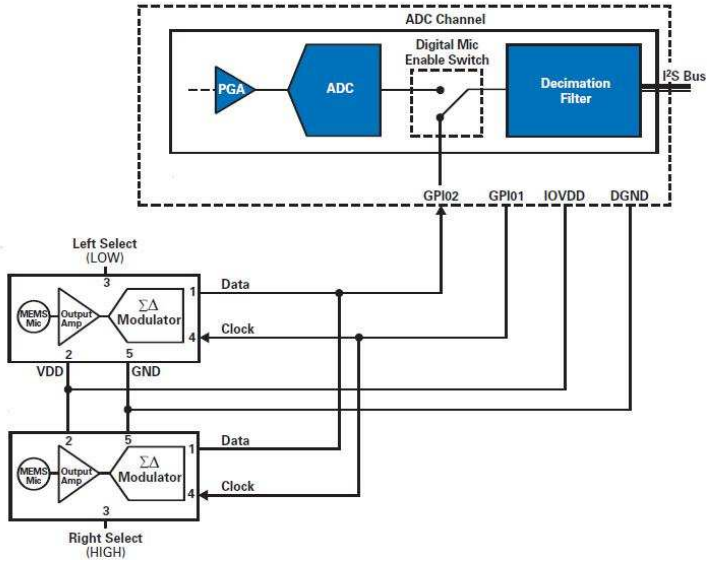


Figura 22. Interconexión micrófonos digitales con CODEC. Imagen extraída de [Texas Instruments, "*Product Bulletin TLV320AIC33/3106/34 Stereo Audio Converters*"].

Este modelo de ruta de datos es válido para todos los CODECs y micrófonos digitales.

En caso de tener micrófonos analógicos la señal de audio primero se transformará en una señal eléctrica en el propio micrófono, luego será amplificada en el preamplificador (PGA en la figura), después será convertida a una señal digital y finalmente será filtrada por un filtro de diezmado. En el caso de tener un micrófono digital la ruta varía un poco al comienzo. La señal acústica se transforma a una señal digital en el propio micrófono y después se lleva directamente al filtro de diezmado (diezmado para bajar la tasa de sobremuestreo del ADC del micrófono digital).

La salida será en ambos casos una señal digital que se transmitirá por el canal de comunicaciones I2S en cualquiera de los formatos disponibles. Todos los CODECs, que analizamos soportan los formatos I2S, "left-justifies", "right-justified" y DSP. Estos formatos simplemente determinan que canal se transmite en que flanco del reloj de sincronización de palabra y si se transmiten primero los bits más o menos significativos. El tratamiento posterior en cualquier dispositivo digital será para cualquiera de estos formatos. De la misma manera suele estar disponible la transmisión en modo TDM para transmitir al mismo tiempo varias señales a diferentes dispositivos.

El control sobre estos aspectos y otros parámetros configurables en los CODEC analizados se realiza a través de los canales I2C o SPI. El primero de ellos siempre está implementado en todos estos dispositivos y el segundo en la mayoría de ellos pero no en todos.

Los CODECs son dispositivos configurables mediante escritura en registros. Todos los parámetros configurables se pueden definir escribiendo en algún registro un valor por el canal de comunicaciones I2C.

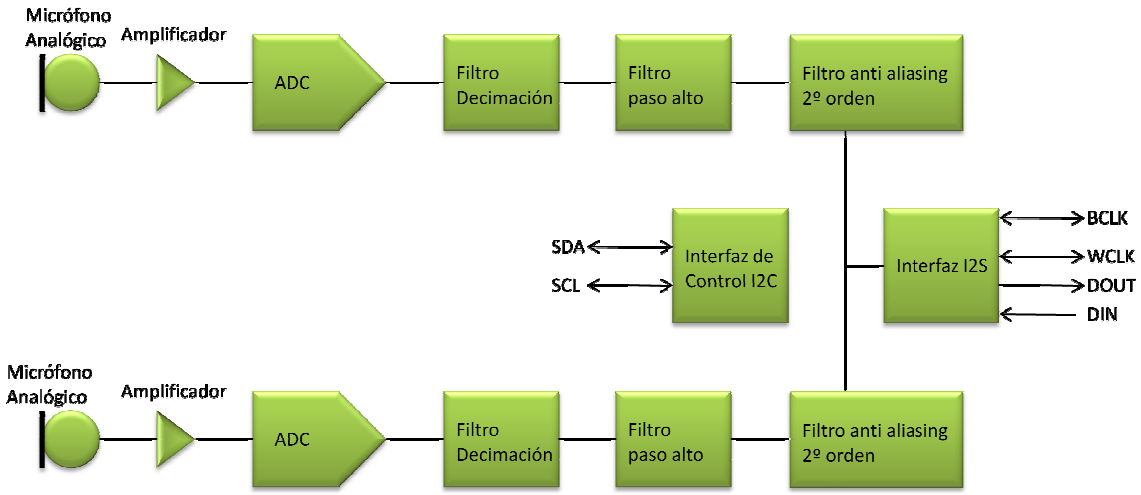


Figura 23. Diagrama de bloques de ruta completa de codificación en CODEC con micrófonos analógicos.

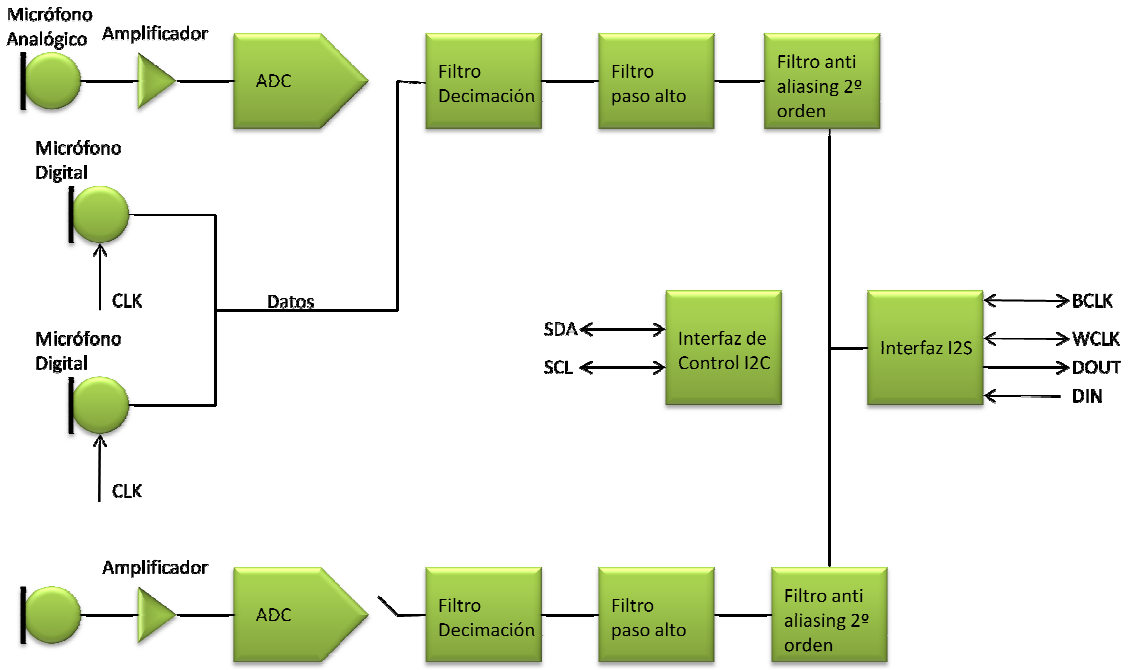


Figura 24. Diagrama de bloques de ruta completa de codificación en CODEC con micrófonos digitales.

Podemos apreciar que cuando usamos micrófonos analógicos cada micrófono está directamente conectado con un ADC. Sin embargo cuando usamos micrófonos digitales, las salidas de datos se unen en una línea multiplexada y los datos se transmiten por un solo canal transmitiendo los datos de cada canal en un flanco de reloj distinto. Aquí utilizamos solo la ruta de uno de los dos ADC. Teóricamente podríamos llegar a usar cuatro micrófonos digitales en un CODEC de dos canales pero la limitación reside en los puertos físicos disponibles de cada CODEC para poder utilizar micrófonos digitales. Ningún CODEC de los analizados tiene implementado pines suficientes para conectar dos micrófonos digitales en ambas rutas de ADC. El otro problema es el número de interfaces de comunicación I2S. Los CODECs actuales solo cuentan con un canal I2S por cada pareja de ADC/DAC.

Cada canal de I2S puede transmitir información de dos canales. Existe la posibilidad de transmitir en formato TDM (Time Divison Multiplexing) pero aquí nos volvemos a encontrar con el problema del número insuficiente de pines para los micrófonos digitales.

Todos los CODECs multicanal comerciales tienen esta estructura. Las únicas diferencias que existen entre los CODECs son las frecuencias que pueden aceptar sus PLLs, así como las frecuencias que pueden generar. También hay diferencias en las relaciones señal a ruido (SNR) de cada una de las rutas de audio.

2.3.4 CODECs comerciales

Como se ha comentado antes existen muchas empresas que fabrican este tipo de dispositivo para que luego se integren en dispositivos más complejos y con una aplicación concreta.

Fabricante	Modelo	ADC/DAC	ADC SNR	DAC SNR	Rango de muestreo	Resolución
Analog Devices	ADAU1381	2/2	95,6 dB	95,6 dB	8-96 kHz	16, 20, 24 bits
Maxim Integrated	MAX9867	2/2	85 dB	90 dB	8-48 kHz	16 bits
Cirrus Logic	CS42L73	2/4	91 dB	97 dB	8-48 kHz	16 bits
Wolfson Microelectronics	WM8903	2/2	95 dB	96 dB	8-96 kHz	24 bits
Wolfson Microelectronics	WM8904	2/2	91 dB	96dB	8-96 kHz	24 bits
Texas Instruments	AIC31	2/2	92 dB	100 dB	8-96 kHz	16, 20, 24, 32 bits
Texas Instruments	AIC3204	2/2	93 dB	100 dB	8-192 kHz	16, 20, 24, 32 bits
Texas Instruments	AIC3253	0/2	-	100 dB	8-192 kHz	16, 20, 24, 32 bits
Texas Instruments	AIC3254	2/2	93 dB	100 dB	8-192 kHz	16, 20, 24, 32 bits
Texas Instruments	AIC33	2/2	92 dB	100 dB	8-96 kHz	16, 20, 24, 32 bits
Texas Instruments	AIC34	4/4	92 dB	102 d	8-96 kHz	16, 20, 24, 32 bits

Tabla 5. Características CODECs actuales comerciales.

Se van a presentar las características especiales de los códecs seleccionados para poder evaluar la mejor elección para el sistema final:

- ADAU1381: solo puede funcionar como esclavo I2C. Tiene integrado un motor de sonido configurable para mejorar la calidad del audio entrante.

Este motor tiene implementado las siguientes funciones que se pueden aplicar a tiempo real en la ruta de grabación: reducción del sonido del viento, mejora de la captura de sonido estéreo, ecualizador de seis bandas y un procesador dinámico de banda dual.

- MAX9867: Tiene integrado un banco de filtro digitales. Lista de filtros: IIR y FIR para fuentes de audio de amplio espectro. Una de las entradas de audio puede ser usada para medir voltajes de voltaje en continua. Tiene gestor de consumo eficiente de energía.
  Diagrama del MAX9867:

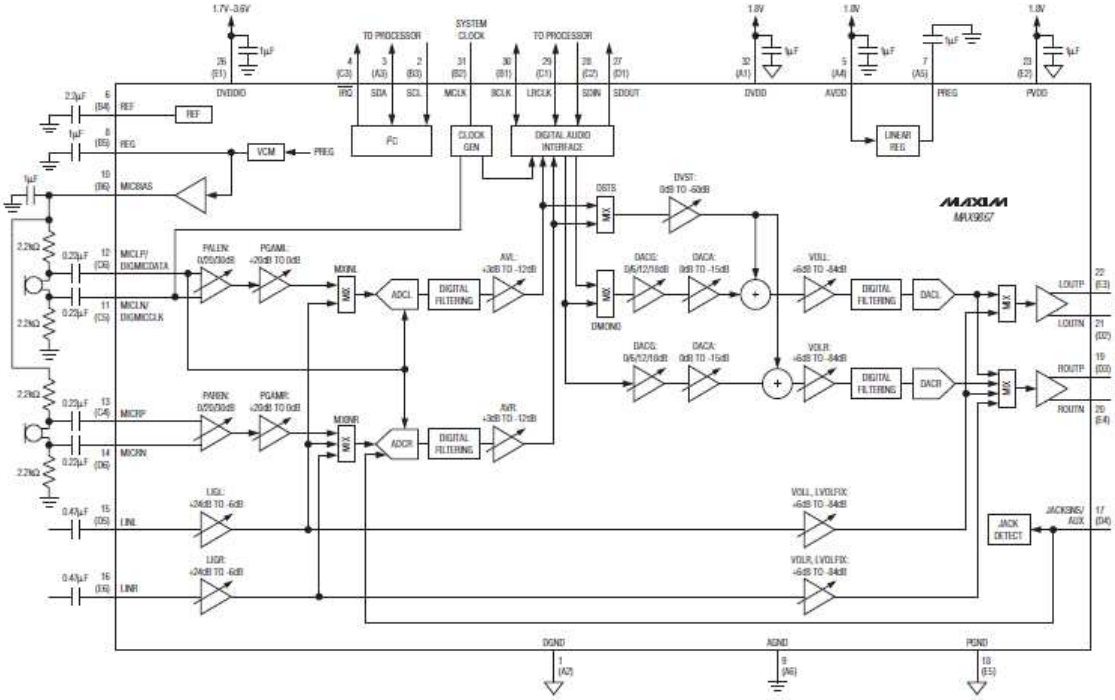


Figura 25. Esquema general de CODEC MAX9867 de MAXIM. Imagen extraída [MAXIM, "*19-4573 Ultra-Low Power Stereo Audio Codec MAX9867"].*

- CS42L73: Tiene PGA, gestor consumo energético de alta eficiencia. Puede tener 2 relojes internos diferentes de referencia, es decir se puede muestrear con dos tasas distintas en los ADC simultáneamente. Tiene un mezclador de entradas digitalizadas conectado con las salidas para simular una mesa de mezclas digital. Tiene 3 puertos independientes de comunicaciones serie: audio, voz y auxiliar.
- WM8903 y WM8904: Se pueden conectar hasta 3 canales de micrófonos en estéreo. Hay un mezclador a la entrada con un control de ganancia para elegir qué cantidad de sonido se quiere dejar pasar a los ADC. Tiene un módulo digital con banco de filtros y control de volumen. Puede realizar las siguientes operaciones: control de volumen dinámico (DRC), filtrado digital y control de volumen normal tanto a la salida de los ADC como a la entrada de los DAC. Tiene una tasa de sobresampleo (OSR) de 64 o de 128. Tiene

registro de configuración de frecuencia de corte del filtro paso alto para cada una de las frecuencias de muestreo.

Esquema interfaz de micrófonos digitales en comparación con los micrófonos analógicos:

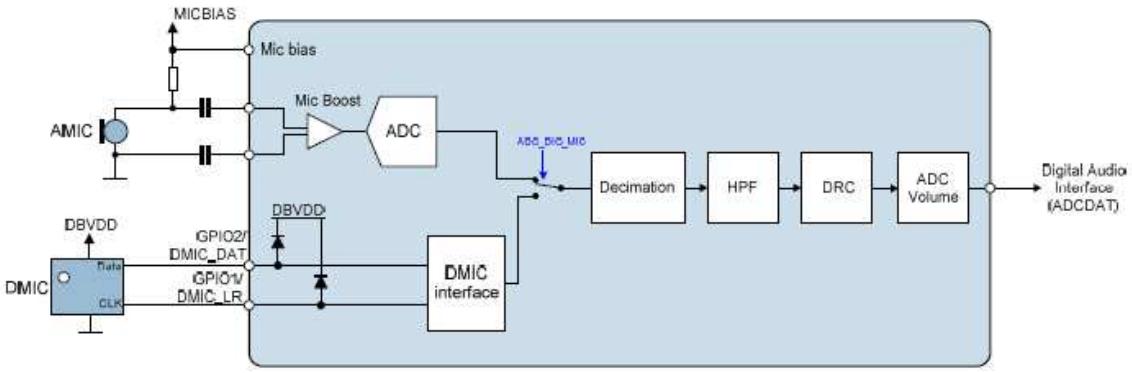


Figura 26. Esquema general de WM8904 de Wolfson Microelectronics. Imagen extraída de [Wolfson Microelectronics, "*Ultra Low Power CODEC for Portable Audio Applications WM8903*"].

- AIC31: AGC, PGA, mezclador de entradas y salidas y control de volumen y efectos en la ruta de salida (antes de los DAC). No tiene capacidad de conexión de micrófonos digitales.
- AIC3204: Tiene soporte par micrófonos digitales, PGA, AGC, canal auxiliar de comunicaciones I2S.
- AIC3253: Solo tiene entradas para micrófonos digitales. Solo existe la posibilidad de conectar dos entradas analógicas para que se sumen a la salida (después del DAC) para realizar mezclas de señales. Tiene incorporado dos núcleos de MiniDSPs, uno logado a la ruta de datos entrante y otro a la ruta de datos DAC, para realizar tareas de cancelación de ruido, eco, . Cada MiniDSP es capaz de procesar 1152 instrucciones por segundo para muestras e audio a 48kHz. Tiene una interfaz secundaria de I2S y DRC para control de volumen.
- AIC3254: Es muy similar al AIC3253, es decir con la característica fundamental de incorporar do MiniDSPs, pero con la diferencia que si tiene incorporados entradas para micrófonos analógicos, así como ADC y controles de volumen analógico a la entradas.
- AIC33: Tiene mezcladores de entradas con control de ganancia individual, control automático de volumen (AGC) y un procesador de efectos de audio digital.
- AIC34: Es muy similar a la estructura y funciones del AIC33 pero con la diferencia que son dos bloques de CODEC en uno. Se han adaptado dos AIC33 en un solo núcleo. Es decir es un CODEC con el doble de entradas y salidas que el AIC33, el doble de ADC, DAC e interfaces de comunicación I2S. El control sobre ellos es muy simple. Tiene para cada bloque una dirección I2C diferente. En definitiva con este CODEC podemos tener 4 micrófonos digitales y dos canales de comunicaciones I2S.

El único CODEC de los analizados que es capaz de tener 4 micrófonos digitales conectados, en una estructura de 2 canales estéreo, es el AIC34 de Texas Instrument. Esto se debe a que el AIC34 son dos CODECs en uno. Son dos bloques de CODEC juntados con 4 ADCs, 4 DACs, 2 canales I2S y 2 entradas diferentes de micrófonos digitales. Tiene la ventaja que con solo una línea de control I2C es posible programarlo. Cada uno de los bloques tiene su propia dirección I2C. Cada bloque se puede programa de manera independiente y ser utilizado para un propósito distinto**.**

2.4 Digital Signal Processors

Las siglas DSP representan en inglés "Digital Signal Processor" que significa procesado digital de señales. Los DSPs son microprocesadores que poseen un juego de instrucciones, un "hardware" y un "software" optimizados para aplicaciones que requieren operaciones numéricas a muy alta velocidad y son diseñados para procesador digital de señales. Un procesador digital de señales es un microprocesador especializado y diseñado para procesar señales digitales en tiempo real mediante secuencias de instrucciones iterativas a muy alta velocidad. La diferencia esencial entre un DSP y un microprocesador es que el DSP tiene características diseñadas para soportar tareas de altas prestaciones, repetitivas y numéricamente intensas. Los DSPs se utilizan habitualmente para medir, filtrar y/o comprimir señales analógicas como por ejemplo filtrar y convertir audio en diferentes formatos. El primer paso, para realizar la el tratamiento de señales, consiste en convertir las señales analógicas a través de un ADC en un flujo de muestras digitales.
Un ejemplo típico de una aplicación de un DSP es un filtro de respuesta impulsional finita (FIR) ha sido ampliamente utilizado en el entorno DSP, es quizás el más simple que permite ilustrar la necesidad de estas prestaciones en los DSP, las cuales permiten concebir muchas de las funciones de procesado en tiempo real.

2.4.1 Historia

El primer DSP fue inventado por la compañía Intel en el año 1978. Se le concedió el nombre "2920" y era un procesador analógico de señales con conversores ADC y DAC pero sin multiplicador "hardware". Hubo otros como compañías como los laboratorios Bell que comenzaron un año más tarde a fabricar también sus primeros DSPs. Pero fue en 1980 cuando la compañía NEC construyó el DSP "PD7710", que fue el primer DSP completo hasta el momento. Un poco más tarde Texas Instruments fabricó su primero DSP llamado TMS32010.
Existen varias empresas a nivel mundial que se encargan de diseñarlos y fabricarlos. En la actualidad las empresas más importantes son Texas Instruments,

Analog Devices y Freescale Semiconductors qué es una división de la compañía Motorola.

2.4.2 Categorización

Los DSP se pueden categorizar de varias maneras. La primera es por el tipo de operaciones que son capaces de realizar, es decir existen DSP que pueden realizar operaciones en coma flotante (floating point) y otros de coma fija (fixed point). Para un mismo tamaño en número de bits, el formato en coma fija proporciona una mejor resolución que el formato en coma flotante. Sin embargo, es este último quien posee un margen dinámico superior.

2.4.3 Arquitectura y Características

La arquitectura y las características de los DSP se diseñan y se construyen en función del propósito con el que se quieran usar y a la aplicación que se quiera destinar. Los DSPs están formados por un núcleo que realiza las operaciones aritméticas y lógicas, una memoria para los datos e instrucciones y elementos periféricos para la entrada y salida de datos.

Los DSP utilizan arquitecturas especiales para acelerar los cálculos matemáticos intensos implicados en la mayoría de sistemas de procesado de señal en tiempo real. Los elementos que componen la arquitectura son las unidades aritmético-lógicas (ALU), sumadores, multiplicadores en combinación con acumuladores (MAC), desplazadores, registros de propósito general, otros elementos específicos para generación de señales de reloj y buses de datos e instrucciones.

Los DSP cuentan con bancos de registros de propósito general para guardar tanto los datos como las direcciones de memoria. Las unidades funcionales que componen el núcleo de procesamiento de los DSP cuentan con los siguientes elementos aritméticos y lógicos:

- **Unidad aritmético-lógica (ALU):** posibilidad de procesamiento aritmético y lógico, incluso la división está soportada.
- **Unidad multiplicadora/acumuladora (MAC):** realiza operaciones en un ciclo de multiplicación, multiplicación y suma o multiplicación y resta o e varios ciclos multiplicaciones con número de mayor longitud de palabra que el bus de datos. Suelen tener varios registros de acumulación auxiliares.
- **Unidades comparativas:** realizan comparaciones a nivel de bits
- **Unidades aritméticas:** para la realización de varias operaciones aritméticas en un mismo ciclo de reloj con varios tamaños de palabra. Estas unidades suelen poder realizar al menos una operación aritmética en un

ciclo de reloj con datos que tengan ancho de palabra igual a los buses de datos. En las unidades aritméticas también se realizan los cálculos para determinar las direcciones de salto en el código, generación de constantes, comparaciones y transferencia de datos entre registros. Existen DSP que tienen varias unidades aritméticas para poder realizar sumas y saltos en un mismo ciclo de reloj.

- **Unidades de carga/guarda desde memoria**.
- **Unidad de desplazamiento:** realiza desplazamientos lógicos y aritméticos, normalizaciones y desnormalizaciones.
- **Secuenciador:** soporta saltos condicionales, saltos a subrutinas e interrupciones.

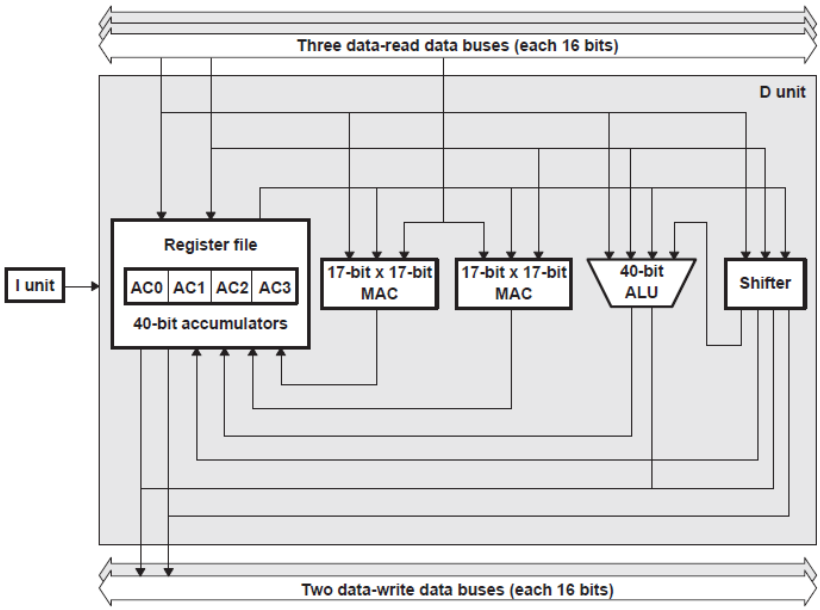


Figura 27. Diagrama de bloques de Unidad lógica D de C55x de Texas Instruments. Imagen extraída de [Texas Instruments, "*Reference Guide C55x v3.x CPU*"].

En la figura superior se muestra como ejemplo la unidad de ejecución del C55X de Texas Instruments donde se puede apreciar como cuenta con un banco de registros acumuladores, 2 unidades MAC, una ALU y un desplazador. Para que los datos accedan a las unidades funcionales se tienen implementados tres buses de lectura de datos y dos de escritura de datos de 16 bits para leer hasta tres operandos y escribir dos en memoria en un solo ciclo de reloj.

Las arquitecturas se diseñan en ciertos DSP en varios grupos para poder aumentar el grado de paralelismo y aumentar así la capacidad de procesado a tiempo real. Cada bloque de unidades funcionales completo tiene un camino de datos diferente. Se replican los bloques de unidades funcionales para lograr mayor grado de paralelismo. También se implementan unidades adicionales para registros auxiliares si no se replican los bloques. De esta forma las arquitecturas suelen permitir ejecución de dos instrucciones en paralelo al mismo tiempo en el mismo ciclo de reloj. Aparte de tener los elementos funcionales duplicados es necesario

tener varios buses de datos para poder realizar dos escrituras simultáneas, o una doble escritura reduciendo el tiempo de ciclo por tarea.

Aparte de núcleo aritmético y lógico del DSP, la memoria juego otro papel importante en el rendimiento de los DSP. A diferencia de los microprocesadores de propósito general, que se basan en una arquitectura de memoria de tipo Von Neunmann, los DSPs cuentan con memoria tipo Harvard, es decir que la memoria de datos y de instrucciones está separada en bloques diferentes, cada uno con sus propios buses de acceso permitiendo acceder de forma simultánea a ambos bloques.
Para una ejecución rápida de las instrucciones mediante los procesos aritméticos y lógicos se diseñan los DSP con múltiples buses de datos e instrucciones para poder cargar y/o guardar de forma simultánea varios operandos e instrucciones al mismo tiempo. Como ejemplo de buses de memoria se muestran con los que cuenta el TMS320VC5505 de Texas Instruments:

- 12 buses independientes:
  - 3 buses de lectura de datos
  - 2 buses de escritura de datos
  - 5 buses de direccionamiento de datos
  - 1 bus de lectura de código
  - 1 bus de direccionamiento de código

Esto permite poder cargar tres datos de memoria, una instrucción, ejecutar otra instrucción y guardar dos datos en memoria en un solo ciclo de reloj.
Las cantidades de memoria que soportan los chips de los DSP de forma interna son muy reducidas dado que no se quieren tener buses de direccionamiento muy grandes. Por ello se tienen puertos para memorias externas que posibilitan la ampliación de la misma.

Para que los DSP puedan funcionar de forma rápida y eficaz se diseñan las arquitecturas mediante segmentación ("Pipeline"). Esto consiste en dividir una secuencia de operaciones en otras de más sencillas y ejecutar en lo posible cada una de ellas en paralelo. En consecuencia se reduce el tiempo total requerido para completar un conjunto de operaciones. Casi todos los DSP del mercado incorporan el uso de la segmentación en mayor o menor medida:

1. Obtención de la instrucción de la memoria: acumula y decodifica las instrucciones que crean el programa de aplicación.
2. Ejecución del programa: hace seguimiento del punto de ejecución mientras el programa se ejecuta. Se proporcionan las direcciones de los operandos.
3. Leer o escribir un operando de la memoria.
4. Ejecutar la parte de la instrucción relacionada con la ALU o MAC.

Los periféricos tienen un peso muy importante dentro de la estructura de un DSP porque permiten recibir y/o transmitir información desde/hacia el mundo exterior. Para poder trabajar con datos digitales a tiempo es necesario que las tarjetas en las que se implementan los DSP tengan periféricos de control de comunicaciones, interfaces datos de entrada/salida y conversores analógico-digitales. También se implementan cristales oscilatorios para generar frecuencias de reloj que posteriormente serán amplificadas en el DSP mediante un circuito de generación de reloj basado en un PLL.

La medida clave para saber si un DSP es o no apropiado para una aplicación es su velocidad de ejecución. Existen varias formas para medir la velocidad de un procesador, aunque quizás el parámetro más usual es el tiempo de ciclo de instrucción. Se trata del tiempo necesario para ejecutar la instrucción más rápida del procesador. Su inverso dividido por un millón da lugar a la velocidad del procesador en millones de instrucciones por segundo o MIPS.

2.4.5 DSP comerciales

Las tres compañías más importantes de fabricación de DSPs a nivel mundial son Texas Instruments, Analog Devices y Freecom Semiconductors, que es una filial de Motorola. Los fabricantes diseñan diferentes categorías de DSP en función de la aplicación, velocidad y consumo que se requieren. En este apartado se van a analizar y comparar los DSP diseñados específicamente para aplicaciones de audio y voz. La manera de comparar los diferentes DSP es a través de la velocidad de cálculo (millones de instrucciones por segundo), la cantidad de memoria disponible y los sistemas de entrada/salida de datos para la aplicación. Se escoge aquel que se adapta de la manera más óptima a la aplicación que se quiere desarrollar.
Texas Instruments cuenta con dos categorías de DSP, la familia con procesadores C6000 de alto rendimiento y la familia de consumo bajo C5000. Freecom Semiconductors cuenta a su vez con dos categorías, una de bajo y otra de alto coste siendo ambas de alto rendimiento. Analog Devices presenta una gama más reducida de DSP de audio siendo muy similares a la C5000 de Texas Instruments. Se ha escogido una muestra de los DSP de coma fija para estudiar sus prestaciones y analizar cuál de los presentados es el más idóneo para implementar el sistema final.

Fabricante	Modelo	Frecuencia Máxima reloj	Velocidad MMAC	Memoria	Periféricos principales
Texas Instruments	TMS320DM6433	700 MHz	5200	240 kbytes	I2C,SPI,Synch Serial, Uart
	TMS320C6416T	1000 MHz	4000	1056 kbytes	I2C,SPI,Synch Serial, Uart
	TMS320C6474	1200 MHz	28800	4800 kbytes	I2C,McBSP
	TMS320C5504	100 MHz	240	256 kbytes	I2S,I2C,SPI,MMC/SD, Synch Serial, USB2.0
	TMS320C5505	100 MHz	240	320 kbytes	I2S,I2C,SPI,MMC/SD, Synch Serial, USB2.0
	TMS320C5514	120 MHz	240	256 kbytes	I2S,I2C,SPI,MMC/SD, Synch Serial, USB2.0
	TMS320C5515	120 MHz	240	320 kbytes	I2S,I2C,SPI,MMC/SD, Synch Serial, USB2.0
	TMS320VC5509A	200 MHz	400	320 kbytes	I2C,SPI,McBSP
Freecom Semiconductor	MSC7116	266 MHz	1000	408 kbytes	Ethernet, TDM (T1/E1, H.110), UART, I2C
	MSC7119	300 MHz	1200	472 kbytes	Ethernet, TDM (T1/E1, H.110), UART, I2C
	DSP56364	100 MHz	100 MIPS	25 k por 24 bits	ESAI(I2S,AC97), SHI(SPI,I2C),GPIO
	DSP56366	120 MHz	120 MIPS	110k por 24 bits	ESAI(I2S,AC97), SHI(SPI,I2C), GPIO, DAX(SPDIF,IEC958,…)
	DSP56374	150 MHz	150 MIPS	54k por 24 bits	ESAI(I2S,AC97), SHI(SPI,I2C), GPIO
Analog Devices	21990	80 MHz	160 MIPS	4k palabras de 24 bits (programa) 4k palabras de 16 bits (datos)	EMIF, SPI, Synchronous Serial(SPORT)
	21992	80 MHz	160 MIPS	32k palabras de 24 bits (programa) 16k palabras de 16 bits (datos)	EMIF, SPI, Synchronous Serial((SPORT))

Tabla 6. Comparación DSP comerciales.

Se van a presentar con un grado más alto de detalle las prestaciones de varios de los DSP de la tabla para mostrar las diferencias entre ellos y poder seleccionar el dispositivo que mejor se adapta al sistema final de codificación.
Los DSPs TMS320DM6433, TMS320C6416T y TMS320C6474 son de la familia de alto rendimiento de Texas Instruments que tienen un núcleo de tecnología C6000 diseñado para aplicaciones de audio, video y red (Ethernet, etc.):

- TMS320DM6433: Su diseño permite ejecutar ocho instrucciones de de 32 bits por ciclo de reloj teniendo en una tasa de 5600 MIPS. Es un dispositivo de bajo consumo ("Low-Power") con una arquitectura de ocho unidades funciones independientes que se componen de:
    - 6 ALUs de 32 o 40 bits que soportan operaciones aritméticas de operandos de 32 bits en un ciclo, dos de 16 bits o cuatro de 8 bits por ciclo
    - 2 Multiplicadores soportan cuatro multiplicaciones de 16 por 16 bits por ciclo de instrucción con resultado de 32 bits o 8 multiplicaciones de de 8 por 8 bits con resultado de 16 bits.

  La memoria interna con la que cuenta el dispositivo es de dos niveles de 128 kbytes cada uno de tipo SRAM. Además entre sus periféricos se encuentran:
    - 4 registros de 32 bit de propósito general
    - 1 puerto de video
    - 1 PCI de bits a 33 MHz
    - 1 puerto I2C
    - 1 puerto McBSP
    - 1 UART
    - 3 puerto VLYNQ (puerto serie full-duplex de alta velocidad para conexión con host)
    - 1 puerto McASP
    - 1 puerto HPI (puerto paralelo para acceso a memoria interna desde host)

- TMS320C6416T: es un procesador con un núcleo de instrucciones de muy larga longitud de palabra (VLIW) con un tiempo de ciclo de instrucción de máximo 1 ns. Es capaz de realizar 8 instrucciones de 32 bits por ciclo de reloj. Como característica añadida para el DSP, dispone de un rango de funcionamiento de temperatura extendido.
  La arquitectura está diseñada en 8 unidades funcionales independientes:
    - 6 ALUs que permiten operaciones en un ciclo de reloj una operación aritmética con operandos con 32 bits, dos de 16 bits o cuatro de 8 bits.

- 2 Multiplicadores soportan cuatro multiplicaciones de 16 por 16 bits por ciclo de intrucción con resultado de 32 bits o 8 multiplicacionesd de 8 por 8 bits con resultado de 16 bits

Se basa en una arquitectura no alineada de carga y guarda. Dispone de una memoria interna de dos niveles, cuyo primer nivel dispone de 32 kbytes de memoria distribuidos en 16 kbytes para código y 16 para datos. El segundo nivel dispone de 1024 kbytes espacio distribuidos por software. Los periféricos para la entrada/salida de los datos con los que cuenta el DSP son:

- 1 conector PCI de 32 bits a 33 MHz
- 3 puertos McBSP
- 1 puerto HPI
- 1 UART
- 1 PWM
- 3 puertos VLYNQ
- Aceleradores de hardware VCP y TCP

- TMS320C5505: Es un procesador de la familia de los C5000 de punto fijo diseñado para aplicaciones de bajo consumo. Los ciclos de instrucción son de 10 ns, es decir que tiene un reloj funcionando a 100 MHz. Puede ejecutar una o dos instrucciones por ciclo de reloj. Su arquitectura se compone de:
  - 2 Multiplicadores con capacidad de 200 MMACS
  - 2 ALUs
  - 3 buses internos de lectura de datos
  - 2 buses internos de escritura de datos

  La memoria de la que dispone es de 320 kBytes de tipo RAM compuesta de 64 kbytes de RAM de doble acceso y 256 kbytes de RAM de acceso único por ciclo de reloj. Dispone de los siguientes periféricos:
  - 1 puerto SPI
  - 1 puerto I2C
  - 4 puertos I2S
  - 1 UART
  - 1 puerto USB
  - 2 interfaces multimedia MMC/SD
  - 4 canales de acceso directo a memoria

- TMS320C5515: Es un procesador de la familia de los C5000 de punto fijo diseñado para aplicaciones de bajo consumo. Tiene una arquitectura muy similar a la delTMS320VC5505 pero con una mejora en el tiempo de los ciclos de instrucción. Funciona hasta a 120 MHz. Dispone prácticamente de los mismos periféricos incluyendo además un puerto USB para la comunicación y transmisión de datos con un sistema Host. Dispone de la misma capacidad de almacenamiento que el TMS320VC5505.

- TMS320VC5509A: Es un procesador de la familia de los C5000 de punto fijo diseñado para aplicaciones de bajo consumo. El tiempo de ciclo de instrucción es de 5 ns llegando hasta los 200 MHz de frecuencia de reloj. Puede ejecutar hasta 2 instrucciones por ciclo de reloj como todos los DSP de la familia C5000. La arquitectura interna del DSP se compone principalmente de:
    - 2 Multiplicadores con capacidad de 400 MMACs
    - 2 ALUs
    - 3 buses internos de lectura de datos y dos de escritura de datos.

  La memoria de la que dispone el DSP es 64 kbytes de RAM tipo DARAM y 192 kbytes de tipo SRAM divididos en bloques de 4 k de 16 bits. El DSP cuenta con una serie de periféricos internos para la comunicación con otros dispositivos:
    - 6 Canales de Acceso directo a memoria
    - 3 puertos serie que se podían combinar entre: 3 McBSP y 2 interfaces multimedia MMC/SD
    - Puerto I2C

- MSC7116: Es un DSP de la compañía Freescale Semiconductors con un núcleo modelo SC1400. Funciona con un reloj hasta a 266 MHz para el núcleo y 133 MHz para los elementos periféricos y puede realizar hasta seis instrucciones en un ciclo de reloj, por ello tiene una tasa de instrucciones por segundo de 1000 MIPS. Su arquitectura se compone principalmente de cuatro ALUs de trabajan con datos de 16 bits, 16 registros de 40 bits de tamaño y 27 registros de 32 bits de capacidad.

  Tiene una memoria interna de dos niveles. El primer nivel tiene una capacidad de 192 kbytes con una cache de instrucciones 16 kbytes de memoria. El segundo nivel se compone de 192 kbytes para datos críticos y temporales. Los periféricos de los que dispone son:
    - Interface Ethernet 10/100 Mbps compatible con estándares de comunicación IEEE 802.3, 802.3u, 802.3x y 802.3ac.
    - Puerto HDI de 16 bits para comunicaciones serie con un sistema Host, DSP y microcontroladores.
    - UART para comunicaciones full-duplex de hasta 5 Mbps.
    - 1 puerto I2C.
    - 41 puertos GPIO.
    - 1 puertos TDM.
    - 1 bus para memoria externa.

- MSC7119: Este dispositivo tiene las mismas características, arquitectura, periféricos y núcleo que el MSC7116. Sin embargo cuenta con dos mejoras respecto al MSC7116 que consisten en que el PLL puede generar una señal

de reloj para uso del núcleo de hasta 300 MHz y de 150 MHz para los periféricos. la memoria interna del DSP tiene una mayor capacidad. Dispone de 256 kbytes de memoria interna de nivel 1, 16 kbytes de cache de instrucciones y 192 kbytes de memoria de segundo nivel.
Ambos DSP están diseñados para ser utilizados en aplicaciones y sistemas de redes Ethernet por sus características.

- DSP56364: Es un DSP, al igual que el resto que están basado en este núcleo 56300, está diseñado para aplicaciones de audio que tiene un PLL que genera una señal de reloj de hasta 100 MHz que alimenta el núcleo y los periféricos. Solo puede ejecutar una instrucción de 24 bits por ciclo, con lo cual tiene un rendimiento de 100 MIPS. El núcleo dispone de una ALU que permite operandos de 24 bits y un multiplicador-acumulador de 24 por 24 bits con un resultado de 56 bits. Dispone de un controlador DMA para gestionar la memoria interna. La memoria está basada en una arquitectura Harvard que permite distribuir la cantidad de memoria de datos y programa de la siguiente manera: de 0,5 k a 1,25 k palabras de 24 bits de memoria de programa y 1,5 k a 0,75 palabras de 24 bits de memoria de datos.
  Los periféricos de los que dispone para la comunicación con otros dispositivos son:
  - Interface serie de audio ESAI (6 líneas de datos que pueden ser hasta 4 cuatro de transmisor y hasta dos de receptor) que permite tener los canales de comunicación I2S, protocolo Sony y AC97.
  - SPI
  - I2C

- DSP56374: Tiene una arquitectura similar al DSP56364 pero con un rendimiento de 150 MIPS funcionando a 150 MHz. También cuenta con una ALU y un multiplicador-acumulador de 24 por 24 bits con un resultado de 56 bits. La memoria es de tipo Harvard con una zona de entre 5 y 7 k por palabras de 24 bits de tipo RAM para los datos y de 2 a 10 k de palabras de 24 bits para el código. Cuenta con los siguientes periféricos:
  - 2 interfaces ESAI
  - SPI
  - I2C
  - Interface de puerto paralelo con sistema Host HDI08
  - Transmisor de audio digital que soporta diferentes formatos de audio

- 21990 y 21992: Son dos DSP con características similares diseñado para aplicaciones de control de motores industriales, etc. Los núcleos trabajan con operandos de 16 bits en punto fijo. Son para aplicaciones de consumo

bajoEl tiempo de instrucción es de 6m25 ns, es decir que funcionan con un reloj interno a 160 MHz. La arquitectura soporta operaciones en paralelo. La cache de instrucciones permite capturas duales de instrucciones en cada ciclo. La arquitectura se compone de :

- ALU
- Multiplicador/acumulador
- Registro de desplazamiento con acumuladores de 40 bits

El reloj para alimentar a los periféricos puede llegar a tener 80 MHz y dispone de los siguientes periféricos:

- ADC de alto rendimiento de 8 canales con resolución de 14 bits y 2 canales simultáneos.
- Controlador DMA conectado a interfaces periféricos de comunicaciones como SPORT (puerto de comunicaciones síncrono serie) y SPI.

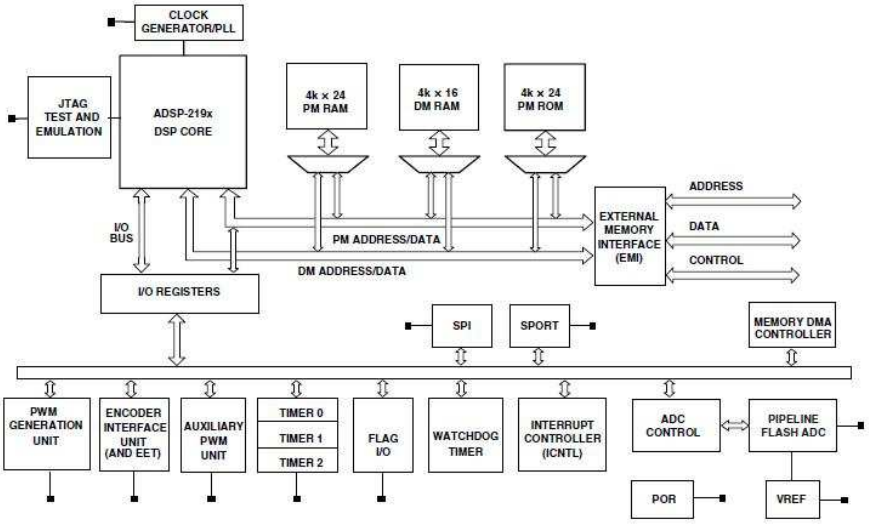


Figura 28. Diagrama bloque funcionamiento DSP Analog Devices serie 21990. Imagen extraída de [Analog Devices, "*Mixed-Signal DSP Controller ADSP-21990*"].

Se ha escogido el TMS320VC5505 debido a que tiene un modulo evaluación con un formato muy pequeño, un coste muy reducido en comparación con los demás DSP y para codificar a tiempo real 4 canales de audio bastan 100 MIPS a 100 MHz de frecuencia de reloj.

Los DSPs de la familia C6000 de Texas Instruments tienen unas características y unos puertos enfocados a las aplicaciones de comunicación de redes. El precio de estos dispositivos es más elevado que el del TMS320VC5505. El mismo problema aparece con los DSPs de la serie MSC de Freescale Semiconductors. Tienen unas características de alto rendimiento que no son necesarias para el desarrollo del prototipo. La serie DSP con núcleo 56300 de Freescale Semiconductors tiene unas características muy parecidas a las del DSP escogido, sin embargo su arquitectura interna no permite trabajar con operandos de 32 o más bits con el misma facilidad. Son necesarios dos ciclos de reloj para realizar operaciones que en TS320VC5505 solo requiere uno. Este margen puede crítico para el rendimiento del sistema final.

3.DISEÑO Y DESARROLLO DE HARDWARE

El primer paso para el desarrollo del sistema de adquisición de audio y codificación a tiempo real es conocer los elementos físicos que compondrán el sistema. La lista de material que se ha escogido, entre los dispositivos presentados, para la confección del sistema de adquisición es la siguiente:

- Micrófonos digitales *SPM0405HD4H-WB* de *Knowles.*
- Micrófonos analógicos con etapa de amplificación *SPM0408HE5H* de *Knowles*.
- CODEC TLV320AIC34 de Texas Instrumentes que va montado sobre el módulo de evaluación TLV320AIC34EVM. El módulo de evaluación contiene los pines físicos para realizar las conexiones con otros dispositivos. El paquete comercial se compone también de una segunda tarjeta llamada USB-MODEVM que permite montar el módulo de evaluación TLV320AIC34EVM sobre ella para que tenga alimentación y señales de reloj procedentes de un oscilador. Esta segunda tarjeta funciona a modo de interface.
- DSP TMS320VC5505 de Texas Instruments está montado sobre una tarjeta de evaluación llamada TMS320VC5505 eZdsp USB Stick también es de Texas Instruments. La tarjeta tiene integrados el DSP TMS320VC5505, un oscilador para obtener señales de reloj externas, un códec para obtener señales analógicas llamado TLV320AIC3204, un conectar USB para recibir alimentación procedente de un ordenador y pines de entrada/salida.

3.1 Propiedades de los dispositivos Hardware

En los siguientes subapartados se definirán las propiedades más importantes de los diferentes elementos involucrados en el sistema.

3.1.1 Propiedades de micrófonos digitales SPM0405HD4H-WB

La primera etapa de un sistema de adquisición de audio son los micrófonos. En esta ocasión se dispondrá de unos micrófonos digitales. Los elegidos son el modelo *SPM0405HD4H-WB.* Como se ha definido en el capítulo 2, los micrófonos digitales tienen la característica fundamental de tener una señal de salida digital, habiendo realizado la conversión en el propio chip de silicio.
Los micrófonos digitales tienen la característica de poder multiplexar dos de ellos en una sola línea de datos teniendo un micrófono para el canal izquierdo y otro para el derecho transmitiendo cada canal en flancos de reloj diferentes (Ver figura 18).

Otras características importantes de los *SPM0405HD4H-WB* son que su SNR es de 56 dB y soporta frecuencias de reloj de entrada entre 1 y 3,25 MHz. La respuesta en frecuencia es plana entre los 300 Hz y los 3 kHz.
Se ha elegido una conexión en la que dos micrófonos digitales están conectados a una misma señal de reloj y a línea de datos de forma conmutada. A través del códec TLV320AIC23EVM se pueden configurar los pines y los ADC para que los datos de cada micrófono digital conectado se capturen en flancos de reloj distintos.

3.1.2 Propiedades de micrófonos analógicos SPM0408HE5H

Para poder realizar una comparación con los micrófonos digitales se ha elegido un modelo de micrófonos analógicos con una etapa de amplificación integrada en el chip de silicio.
Los micrófonos analógicos *SPM0408HE5H* tienen la peculiaridad de que se les puede variar la ganancia que le aplican a la señal de salida. Esta ganancia está comprendida en el rango de 0 a 20dB. Si queremos aplicar una ganancia de 0db debemos conectar el pin número 2 a tierra sin más. Sin embargo si queremos aplicar una ganancia de 20dB debemos conectarlo al pin de tierra con un condensador de 0.47µF entre medias. Existe la posibilidad de ponerle otra ganancia según dos fórmulas:

$$G = 1 + \left\{\frac{R_1}{R_2 + R_3}\right\}; Ganancia\ (dB) = 20 \cdot \log\ (G)$$

La relación señal a ruido (SNR) tiene como valor mínimo los 55 dB y como valor máximo 59 dB. La respuesta en frecuencia tiene una zona plana entre los 300 y los 3000 Hz.

3.1.3 Propiedades de CODEC TLV320AIC34

El AIC34 es un CODEC de audio de cuatro canales. Está estructurado en dos bloques que dispone de dos canales de cada uno de ellos, es decir que cuenta con dos ADC y DAC por cada bloque. Tiene la posibilidad de configurar cada bloque de forma independiente para utilizarlos con propósitos distintos.
Cada bloque del CODEC cuenta con un canal I2S de audio, dos pines GPIO, dos ADC y dos DAC independientes, un banco de filtros digitales, un módulo de generación de frecuencias de reloj de muestreo y una serie de entradas y salidas de audio analógicas.

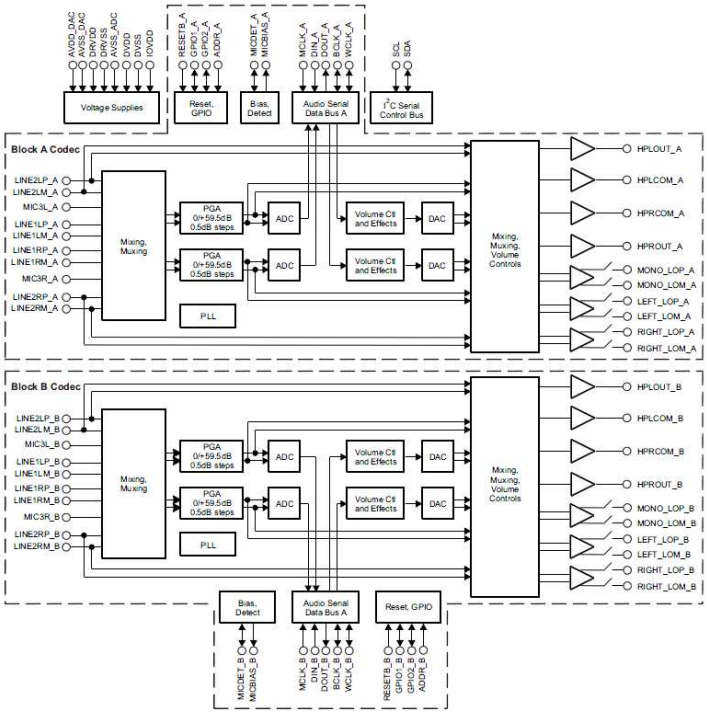


Figura 29. Diagrama de bloque AIC34. Imagen extraída [Texas Instruments, "*TLV320AIC34EVM Datasheet*", SLAS538A].

EL dispositivo es completamente configurable a través del bus de configuración I2C. Cada bloque tiene una dirección I2C distinta. Se va a describir en el punto 4.3.5.

El CODEC consta de diferentes opciones de rutas de datos de entrada con la posibilidad de habilitarlos todos y mezclarlos antes de ser transformados a señales digitales. Las entradas de los ADC cuentan con un control de ganancia llamado PGA que permite aumentar la energía de la señal hasta en 59 dB para las entradas a través de micrófonos analógicos. Se puede configurar el CODEC para que los puertos GPIO funcionen como interface de micrófonos digitales. Cada bloque permite una pareja de micrófonos digitales.

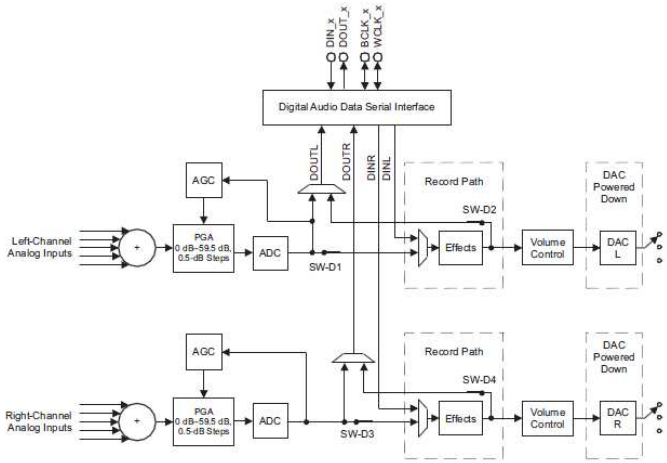


Figura 30. Rutas de datos. Imagen extraída [Texas Instruments, *"TLV320AIC34EVM Datasheet"*, SLAS538A].

EL CODEC con cuatro delta-sigma multibit ADC y cuatro delta-sigma multibit DAC que pueden funcionar con frecuencias de muestreo comprendidos entre 8y 96 kHz. Además par cada ADC existe un amplificador de ganancia programable dedicado con un hardware configurable de ganancia automática de control en todos los canales.
Como interfaces cuenta con buses de datos I2S programables en cuanto a la longitud de las palabras, programable en cuanto la decisión de si el CODEC funciona como maestro o esclavo en la comunicación y cuatro formatos de transmisión (I2S, left/right-justified, DSP) para cada bloque. El códec permite conectar en cada bloque dos micrófonos digitales o dos micrófonos analógicos, con lo que se podrán tener hasta 4 fuentes de audio independientes. Se puede muestrear cada pareja de fuentes de audio con tasas distintas y transmitir al DSP con propiedades de palabras digitales distintas. El módulo I2S para la recepción y transmisión de datos digitales puede funcionar como elemento maestro o esclavo en la comunicación. Se ha elegido que funcione como esclavo y que sea el DSP el que genere las señales de reloj WCLK y BCLK porque el códec no generaba las frecuencias adecuadas y no se podía realizar la comunicación de forma correcta.

3.1.4 Propiedades de DSP TMS320VC5505 eZdsp

El TMS320Vc5505 eZdsp se compone de un núcleo TMS320VC5505 montado sobre un chip de evaluación que contiene los periféricos, osciladores para generación de la fuente de reloj, los pines de entrada/salida de datos, el interface de alimentación y el interface de comunicación de programación del DSP.

3.1.4.1 Propiedades del núcleo del DSP C55x

El dispositivo TSM320VC5505 es un DSP de bajo consumo específico para el tratamiento de señales de audio que contiene el núcleo de procesamiento a una frecuencia máxima de funcionamiento de 100 MHz y un rendimiento de 100 MIPS
EL TMS320VC5505 es parte de la familia de procesadores de señales digitales de punto fijo y está diseñado para aplicaciones de consumo energético bajo. Se logra un rendimiento alto con un consumo bajo a través de un incremento del paralelismo en la arquitectura del núcleo y el ahorra energético.
La estructura interna del CPU del DSP está compuesta por:

- 1 cola de buffers de instrucciones de tamaño 32 x 16 bits.
- 2 unidades MAC de 17 bits x 17 bits
- 1 ALU de 40 bits
- 1 desplazador de 40 bits
- 1 ALU de 16 bits
- 4 acumuladores de 40 bits

- 12 buses independientes:
    - 2 buses de lectura de datos de 16 bits
    - 1 bus de lectura de datos de 32 bits
    - 2 buses de escritura de datos de 16 bits
    - 5 buses de direccionamiento de datos
    - 1 bus de lectura de código
    - 1 bus de direccionamiento de código
- Tamaño de palabras de código variables: 8, 16, 24, 32, 40, 48 bits
- Tamaño de palabras de datos: 16 bits
- 3 unidades aritmético-lógicas de registros auxiliares de 24 bits
- 8 registros auxiliares
- 4 registros de datos

El set de instrucciones incluye la sintaxis necesaria para ejecutar instrucciones en paralelo. Esto simplifica la tarea del programador y optimiza el tiempo y el coste de desarrollo de un programa. En conjunto se consigue un rendimiento de cuatro lecturas de datos de 16 bits y dos escrituras de datos de 16 bits en un mismo ciclo de reloj.
Se incluyen 4 controladores DMA, cada uno de cuatro canales, que proporcionan movimientos de datos para 16 canales independientes sin intervención del CPU. Cada controlador de DMA realiza una transferencia de 32 bits por ciclo.

La longitud de las instrucciones es variable para optimizar la densidad del código y la eficiencia de bus de datos.

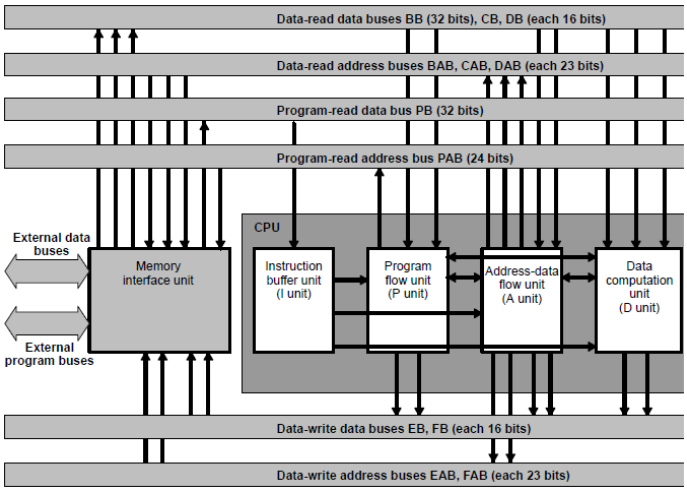


Figura 31. Esquema de buses de datos y módulos lógicos de núcleo de DSP C55x. Imagen extraída [Texas Instruments, "*Reference Guide C55x v3.x CPU*"].

La CPU se estructura en cuatro unidades funcionales segmentadas:

- Instrucction buffer unit: (unidad de instrucciones) Acumula y decodifica las instrucciones que crean el programa de aplicación.

- Program flow unit: (unidad de flujo de programa) Hace seguimiento del punto de ejecución mientras el programa se ejecuta. Incluye hardware para uso eficiente de los bucles, saltos condicionales, ejecución condicional y "pipeline."
- Address data fow unit: (unidad de flujo de direcciones de datos) Proporciona las direcciones de datos durante la ejecución del programa. Hay un hardware dedicado para el manejo de los cinco buses de datos. También incrementa el paralelismo del C55x introduciendo si fuera necesario una ALU para operaciones artiméticas.
- Data computation unit: (unidad de computación de datos) Es el corazón del DSP. Realiza las operaciones aritméticas sobre los datos procesados. Incluye las MACs, las ALU y los acumuladores de registros. Incluye también el desplazador, el control de saturación y redondeo y hardware especial dedicado para la realización del algoritmo de Viterbi.

El CPU del C55x soporta un set de instrucciones variable para mejorar la densidad del código. La unidad de instrucción (IU) realiza capturas de palabras de 32 bits de memorias internas o externas y encola las instrucciones para la unidad de programa (PU). Esta unidad decodifica las instrucciones y las envía a la unidad de direccionamiento (AU) y a la unidad de datos (DU) donde se realizan las operaciones aritméticas y lógicas.
Aparte de la CPU, el DSP dispone de una memoria de datos e instrucciones físicamente unida en un único chip pero segregada de forma lógica. La memoria cuenta con 320 kbytes de espacio de almacenamiento dividido en un bloque de 64 kytes de memoria tipo DARAM, que es una memoria RAM de acceso dual en un ciclo de reloj y 256 kbytes de memoria tipo SARAM, que es una memoria de acceso único en un ciclo de reloj.

3.1.4.2 Propiedades de los periféricos del TMS320VC5505 eZdsp

La tarjeta de evaluación contiene los periféricos del DSP. Mediante los registros de control de los buses serie y paralelo se pueden configurar los pines multiplexados, los drivers de salida, la habilitación/deshabilitación de los dispositivos pull-up y pull-down, los controladores DMA, el reset periférico y el interface de memoria externa EMIF.
Aparte del CPU y la memoria del C55x el chip dispone de un amplio set de periféricos, entre los que se incluyen:

- **Módulo con 4 canales Inter-IC Sound (I2S):** Permiten transferencia de datos en modo serie de forma full-duplex, normalmente datos de audio entre un dispositivo externo y el DSP.

- **1 interface de Puerto-serie (Serial-Port Interface, SPI):** El interface de puerto serie (SPI) es un modulo de de transferencia de datos síncrono de alta velocidad que permite un flujo de datos de longitud programable.
- **1 bus de comunicaciones I2C multimaestro y esclavo:** El modulo I2C proporciona un interface entre el VC5505 y otros dispositivos que tengan un módulo I2C compatible con las especificaciones del bus I2C de Phiips Semiconductors Inter-IC bus (I2C-bus™), versión 2.1 de dos pines.
- **1 Interface transmisor/receptor UART.**
- **1 Interface de memoria externa (EMIF):** Es un interface de memoria que aporta acceso directo a memorias asíncronas de tipo EPROM, NOR, NAND de tipo célula de nivel único (SCL) con 1 bit ECC y de tipo célula de múltiples niveles (MLC) con 4 bits de ECC y SDRAM, que está compuesto por un bus de dirección de 21 bits y un bus de datos de 16 bits.
- **Reloj a tiempo real (RTC).**
- **3 temporizadores de propósito general.**
- **4 controladores de acceso directo a memoria (DMA), cada uno con cuatro canales independientes:** Para permitir movimientos de datos en la memoria interna sin intervención de la CPU, mientras ésta realiza operaciones.
- **2 buses serie:** los buses serie agrupan el control de los módulos 2C, I2S, GPIO, etc.
- **1 bus paralelo:** Es configurable para soportar una línea de control del LCD, un interface del puerto serie (SPI), un I2S y/o las señales de un módulo UART (Universal Asynchronous Receivr/Transmitter).
- **GPIO:** Este periférico proporciona una serie de pines de propósito general que pueden ser configurados como entradas o salidas.

La tarjeta TMS320VC5505 eZdsp dispone a su vez de varios módulos periféricos que funcionan a modo de interface de audio e interfaces con otros dispositivos como son el conector de expansión para unir mediante un conector MEC1-130-02-S-D-NP-A de 40 pines, un controlador LED, un códec de audio modelo TLV320AIC3205 de Texas Instruments que tiene en sus puertos de entrada/salida un conector estéreo de audio tipo "Jack" y una entrada de micrófono. La tarjeta también dispone de un bus serie de alta velocidad (USB 2.0) que permite la conexión con un equipo host para transmitir a través del emulador embebido del XDS100 JTAG el programa de carga del DSP.

3.2 Interconexión

3.2.1 Esquema general de Interconexión

El sistema final tiene como objetivo la codificación de audio a tiempo real mediante la adquisición de audio a través de micrófonos digitales. Para ello se construye un

sistema físico interconectando las tarjetas comerciales de evaluación, es decir la tarjeta del DSP (TMS320VC5505 eZdsp) con la tarjeta del códec (TLV320AIC34EVM) y conectando los micrófonos digitales y/o los analógicos a la tarjeta del CODEC. Después de unir todos los elementos y se comprueba que la ruta de transferían de datos es correcta realizando una pequeña demostración de grabación y reproducción del audio a tiempo real. El audio es conducido desde los micrófonos hasta el DSP y de vuelta hacia las salidas para escucharlo y comprobar que funciona correctamente. El primero paso para la realización del sistema es la interconexión entre los micrófonos tanto digitales como analógicos con el CODEC AIC34 a través de los pines de la tarjeta de evaluación TLV320AIC34EVM. La tarjeta del códec TLV320AIC34EVM contiene todos los pines de conexión para realizar el sistema final. Los pines de la tarjeta USB-MODEVM están diseñados exclusivamente para proporcionar al TLV320AIC34EVM alimentación y una señal de reloj maestro. En segundo lugar se ha de conectar el DSP a través de los pines del conector de expansión de la tarjeta de evaluación TS320VC5505 eZdsp con el AIC34 a través de los pines del TLV320AIC34EVM.

3.2.2 Interconexión Micrófonos-CODEC

La primera etapa de interconexión se realiza entre los micrófonos y la tarjeta de evaluación del CODEC. Se tienen dos conexiones diferentes dependiendo del tipo de micrófono. En la primera se usará una estructura de 2 micrófonos digitales que utilizan una línea de datos multiplexada en el tiempo para ambos. Para el segundo caso se tiene que cada línea de datos de salida de los micrófonos analógicos se conecta a una línea de entrada de micrófono del TLC320AIC34EVM.
Los micrófonos tienen una línea de alimentación, una o dos de tierra, una de datos y para los micrófonos digitales existe además una línea elección de canal izquierdo o derecho y una línea de reloj. En este caso el micrófono analógico tiene además una línea para la configuración de la ganancia.

# Pines Impares	Señal	# Pines Pares	Señal
1	Ground	2	Left/Right
3	Ground	4	Clock
5	Data	6	Power (V_{dd})

Tabla 7. Conexiones micrófono SPM0405HD4H-WB. Tabla extraída de [Knowles Acoustics, "*SPM0405HD4H-WB Topics Digital "Mini" SiSonicTM Microphone Specification - Halogen Free*"].

# Pines Impares	Señal	# Pines Pares	Señal
1	Output	2	Gain
3	Ground	4	Power (V_{dd})

Tabla 8. Conexiones micrófono SPM0408HE5H. Tabla extraída [Knowles Acoustics, "*SPM0408HE5H Amplified "Mini" SiSonic Microphone Specification with enhanced RF protection- Halogen Free*"].

Por parte del TLC330AIC34EVM existe también una diferencia para conectar micrófonos digitales o analógicos. Para los digitales se hace uso de los pines de

GPIO1 y GPIO2 del conector J16 o J17 de la tarjeta TLV320AIC EVM. El pin GPIO1 sirve para transmitir la señal de reloj que se le suministrará al micrófono digital y e GPIO2 es el puerto de datos que se puede configurar para tener 1 o 2 micrófonos conectados. Las conexiones con los micrófonos analógicos se realizan a través de las entradas de línea de micrófonos.

# Pines Impares	Nombre Señal	# Pines Pares	Nombre Señal
1	No conectado	2	GPIO1_A
3	No conectado	4	DGND
5	No conectado	6	GPIO2_A
7	No conectado	8	Señal RESET entrada
9	No conectado	10	DGND
11	No conectado	12	No conectado
13	No conectado	14	RESET
15	No conectado	16	SCL
17	No conectado	18	DGND
19	No conectado	20	SDA

Tabla 9. Pines del conector J16 de la tarjeta TLV320AIC34EVM. Tabla extraída [Texas Instruments, "*TLV320AIC34EVM-K User's Guide*", SLAU232].

# Pines Impares	Señal	# Pines Pares	Señal
1	No conectado	2	No conectado
3	BCLK_A	4	DGND
5	BCLK_B	6	GPIO1_B
7	WCLK_A	8	GPIO2_B
9	WCLK_B	10	DGND
11	DIN_A	12	DIN_B
13	DOUT_A	14	DOUT_B
15	No conectado	16	SCL
17	MCLK_A	18	DGND
19	MCLK_B	20	SDA

Tabla 10. Pines del conector J17 de la tarjeta TLV320AIC34EVM. Tabla extraída [Texas Instruments, "*TLV320AIC34EVM-K User's Guide*", SLAU232].

Micrófonos Digitales		TLV320AIC34EVM				
# Pin	Nombre Señal	# Pin Left	# Pin Right	Nombre Conector	Nombre Señal	
1	GROUND	5	5	J.15	DGND	DGND
2	LEFT/RIGHT	5	7	J.15	DGND	+1.8VD
3	GROUND	5	5	J.15	DGND	DGND
4	CLOCK	2 / 6	2 / 6	J.16 / J.17	GPIO0_A	GPIO1_B
5	DATA	6 / 8	6 / 8	J.16 / J.17	GPIO1_A	GPIO2_B
6	POWER(V_{dd})	7	7	J.15	+1.8VD	+1.8VD

Tabla 11. Conexión entre micrófonos digitales SPM0405HD4H-WB y tarjeta TLV320AIC34EVM.

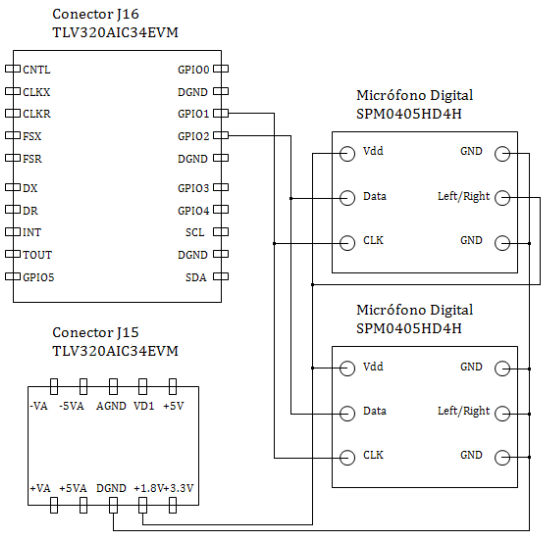


Figura 32. Esquema de conexión entre micrófonos digitales SPM0405HD4H-WB y TLV320AIC34 EVM.

Para realizar las conexiones entre los micrófonos digitales y la tarjeta TLV320AIC34 EVM montado sobre la tarjeta TLV320AIC34, hay que conectar los pines de alimentación de los micrófonos (pin 6: VCC) con el pin 7 del J15 (+1.8VD).
Los pines de tierra (pin 1 y 3: Ground) hay que conectarlos al pin 5 del J15 (DGND).
Los pines de datos de los micrófonos (pin 5: Data) van conectados al pin GPIO2 del J16 (pin 2 GPIO1).
Los pines de reloj de los micrófonos (pin 4: CLK) van conectados al pin GPIO1 del J16 (pin 6 GPIO2).
Uno de los pines de izquierda/derecha (pin 2: left /right) de los micrófonos va conectado a tierra, pin 5 del J15 o pin 10 del J16 y el otro pin, el del segundo micrófono, va conectado al pin de alimentación, pin 7 del J15 (+1.8VD). Para usar dos micrófonos al mismo tiempo siempre debemos conectar uno de los pines left/right a tierra (DGND) y el otro al voltaje de alimentación (DVDD=+1.8V).
Los pines de CLK de ambos micrófonos van juntos en una sola línea hasta su pin en la placa. Lo mismo sucede con los de datos.
Para la conexión entre los micrófonos analógicos y el códec se tiene una configuración ligeramente diferente. No se usan los pines GPIO1 ni GPIO2 para el reloj y los datos. Para alimentar los micrófonos se usarán los pines de MICBIAS y para la tierra los pines de AGND. Los datos irán por los pines MIC3R y MIC3L.

Micrófonos Analógicos		**TLV320AIC34EVM**			
# Pin	**Nombre Señal**	**#Pin Left**	**# Pin Right**	**Nombre Conector**	**Nombre Señal**
1	OUTPUT	12	10	J.13	MIC3R_x/ MIC3L_x
2	GAIN	13	13	J.13	AGND más Capacidad
3	GROUND	9	10	J.13	AGND
4	POWER(Vdd)	14	14	J.13	MICBIAS_x

Tabla 12. Conexión entre micrófonos analógicos SPM0408HE5H y TLV320AIC34 EVM.

Los micrófonos analógicos, que se han escogido, tienen además un pin que se debe conectar a tierra a través de una resistencia y un condensador para definir el valor

de la ganancia que se le aplica a la señal adquirida. Se ha se conectado el micrófono para que la ganancia sea máxima. Para ello se conecta el pin de ganancia a tierra intercalando solo un condensador de 0,47 µF y no colocando una resistencia en el lugar de R_3.

$$G = 1 + \left\{\frac{22k\Omega}{2{,}4k\Omega}\right\} = 10{,}167 \Rightarrow Ganancia\ (dB) = 20 \cdot \log(10{,}167) = 20dB$$

3.2.3 Interconexión CODEC-DSP

La conexión correcta entre a tarjeta TS320VC5505 eZdsp y la TLV320AIC34 EVM de Texas Instruments se realiza de la siguiente manera. Para tener cuatro canales de audio simultáneamente se necesitarán dos buses de audio I2S, un bus de control I2C, una señal GPIO para el control de "software reset" del códec desde el DSP y varias conexiones de tierra para realizar una captura de audio de dos micrófonos digitales de dos canales cada uno (estéreo).
Se usarán los canales I2S2 e I2S1 para la comunicación de audio entre el DSP y el códec. Los pines correspondientes del conector de expansión P1 de la tarjeta TMS320VC5505 eZdsp para los buses I2S son el 19, 21, 23 y 25 para I2S_2 y el 29, 31, 33 y 35 para I2S_1. Los pines 19 y 29 llevan las señales de los relojes, BCLK (I2Sn_CLK) de los dos buses, los pines 21 y 31 llevan los datos de recepción RX (I2Sn_DIN), los pines 23 y 33 llevan los datos de transmisión DX (I2S_DOUT) y los pines 25 y 35 llevan las señales de los relojes de sincronización WCLK (I2S_FS).

# Pin Top	Señal	# Pin Bottom	Señal
1	GND	2	GND
3	SPIO_CS1	4	GPIO13
5	SPIO_CLK	6	GPIO12
7	SPIO_DX	8	GPIO14
9	SPIO_RX	10	GPIO15
11	GND	12	GND
13	I2C_SDA	14	GPIO16
15	I2C_SCL	16	GPIO17
17	GND	18	GND
19	I2S2_CLK	20	GPIO11
21	I2S2_RX	22	GPIO10
23	I2S2_DX	24	GPIO5
25	I2S2_FS	26	GPIO4
27	GND	28	GND
29	I2S1_CLK	30	UART_RTS
31	I2S1_RX	32	UART_CTS
33	I2S1_DX	34	UART_RX
35	I2S1_FS	36	UART_TX
37	VCC_3V3	38	VCC_3V3
39	VCC_3V3	40	VCC_3V3

Tabla 13. Pines del conector de expansión P1 de la tarjeta TMS320VC5505 eZdsp. Tabla extraída de [Spectrum Digital, "*Technical Reference TMS320VC5505 eZdsp USB Stick*"].

Las señales de reloj I2S, I2Sn_CLK (pines 19 y 29), se conectan a los pines de la señales de reloj del códec AIC34, es decir a los pines J17.3 y J17.5 (BCLK A y B) de la tarjeta TLV320AIC34.
Las señales de recepción de datos, I2Sn_RX (pines 21 y 31), de los buses I2S del DSP se conectan a los pines J17.13 y J17.14 del códec (DOUT A y B), así como las señales de transmisión de datos, I2Sn_DX (pines 23 y 33), se conectan a los pines J17.11 y J17.12 (DIN A y B). Los últimos cables de los buses I2S son las señales de sincronización, I2Sn_FS (pines 25 y 35), que se conectan a los pines J17.5 y 17.7 (WCLK A y B).
Los pines 13 y 15 del conector de expansión son las dos señales del bus de control I2C. EL pin 13 es la señal de datos I2C_SDA. El pin 15 es la señal de reloj del I2C_SCL. La señal I2C_SDA va conectada al puerto J17.20 o al J16.20 (SDA). Ambos puertos están cortocircuitados. La señal I2C_SCL va conectada al puerto 17.18 o J16.18 (SCL). En principio se ha elegido el conector J17 para realizar las conexiones del bus de control I2C.

Como señal de reset hemos elegido el pin GPIO16 del conector P1 de la tarjeta TMS320VC5505 eZdsp que se encuentra en el pin 14. Esta señal se conecta al pin GPIO4 del conector J16 de la tarjeta TLV320AIC34 EVM que está unido a la señal de reset del códec, es decir al puerto J16.14.
Por último hemos realizado 3 conexiones de tierra entre amabas tarjetas. La unión de los planos de tierra es fundamental para la perfecta sincronización y nivel de voltaje de las señales. Para ello hemos unido los pines 11, 17 27 del conector de expansión P1 del la tarjeta de evaluación del DSP con los pines 4, 10 y 18 (puertos DGND) del conector J17 de la tarjeta TLV320AIC34 EVM.

TMS320VC5505 eZdsp			**TLV320AIC34EVM**		
# Pin	**Nombre Señal**	**Grupo Señales**	**# Pin**	**Nombre Conector**	**Nombre Señal**
19	I2S2_CLK	I2S_2	3	J.14	BCLK_A
21	I2S2_RX	I2S_2	13	J.14	DOUT_A
23	I2S2_DX	I2S_2	11	J.14	DIN_A
25	I2S2_FS	I2S_2	7	J.14	WCLK_A
29	I2S1_CLK	I2S_1	5	J.14	BCLK_B
31	I2S1_RX	I2S_1	14	J.14	DOUT_B
33	I2S1_DX	I2S_1	12	J.14	DIN_B
35	I2S1_FS	I2S_1	9	J.14	WCLK_B
13	I2C_SDA	I2C	20	J.14 / J.13	SDA
15	I2C_SCL	I2C	16	J.14 / J.13	SCL
14	GPIO16	RESET	8	J.13	GPIO2
16	GPIO17	RESET	14	J.13	GPIO4
11	GND	Tierra	4	J.14	DGND
17	GND	Tierra	10	J.14	DGND
27	GND	Tierra	18	J.14	DGND

Tabla 14. Conexión entre TLV320AIC34 EVM y conector P1 del DSP TMS320VC5505 eZdsp.

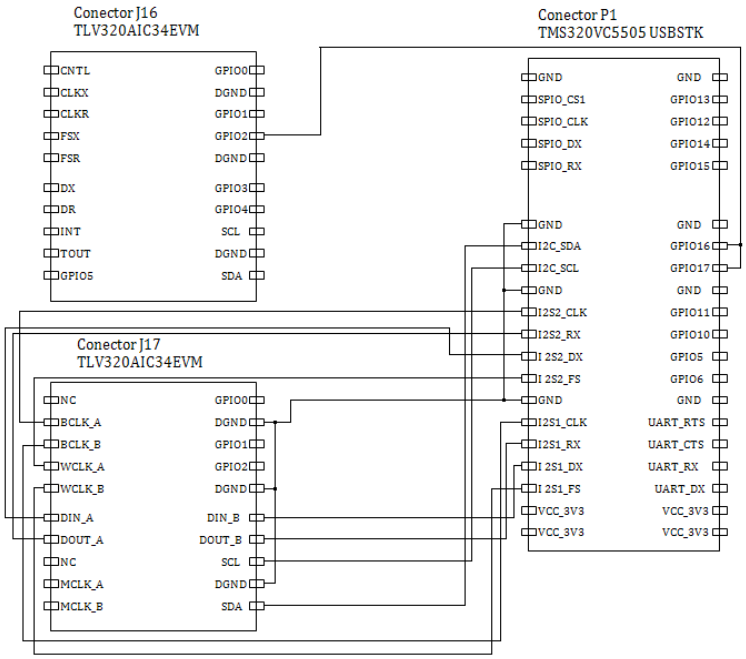


Figura 33. Esquema interconexión DSP y CODEC AIC34.

Se ha usado solo dos buses I2S de dos canales cada uno debido a las limitaciones del códec. Éste solo dispone de dos bloques de codificación de señales, es decir solo se es capaz de tener en un mismo sistema hasta 2 micrófonos estéreo. Esta tarjeta del DSP también tiene la limitación de solo tener disponible 3 buses de datos I2S para la comunicación de información de audio. Solo se podría añadir un micrófono de dos canales si se añadiera otro códec de las mismas características.

Se va a utilizar el conector "jack" de 3.5mm para poder escuchar directamente el audio saliente decodificado en el programa de demostración. Estos conectores irán directamente conectados al códec TLV320AIC34 a través de las salidas "HPLOUT" y "HPROUT" de cada bloque.

El CODEC se encuentra instalado en una tarjeta llamada TLV320AIC34EVM en la cual están disponibles los pines de entrada/salida, alimentación y masa. El códec cuenta con una segunda tarjeta llamada USB-MODE de evaluación que dispone de un cristal que genera una señal de reloj de 11,2867 MHz para alimentar la entrada de la señal de reloj del AIC34. Además del cristal, el códec está conectado mediante el canal de control I2C con la tarjeta USB-MODE para poder controlar desde un equipo a través del USB el códec. Se ha eliminado dicha conexión para que sea el DSP el que controle y configure el códec. Esa conexión no permitía poder usar otro dispositivo como elemento maestro de la comunicación I2C y en consecuencia no se podría configurar el códec desde el DSP. Por ello la configuración final del prototipo tendrá el USB-MODE conectado al TLV320AIC34EVM para entregar la señal de reloj maestro.

3.3 Canales de Interconexión

Para las comunicaciones entre los dispositivos se usan los buses de comunicación serie síncrona I2C (proviene de Inter-Integrated Circuit) e I2S (Inter-IC Sound). El bus I2C, un estándar que facilita la comunicación entre microcontroladores, memorias y otros dispositivos sólo requiriendo dos líneas de señal y un común o masa. El I2S es otro estándar que se usa habitualmente para las transferencias de datos tipo PCM. El bus I2C se utiliza en el sistema como canal de control y configuración de los dispositivos conectados al DSP. A través del canal I2C, el DSP configura el códec mediante la escritura de sus registros. Los canales I2S se usan para transmitir los datos de audio digitalizados tanto desde el códec hacia el DSP para codificarlos como desde el DSP hacia el códec para reproducirlos y comprobar que el sistema funciona correctamente.

3.3.1 I2C

3.3.1.1 Funcionamiento y propósito

El bus I2C es un bus de control. Permite desde cualquier máquina tener un control sobre los registros de otro dispositivo. Permite acceder a los registros del códec TLV320AIC34 y programarlo desde el programa que se carga en el DSP.
El bus I2C es muy simple y se compone de 2 líneas, una de datos y una de reloj. La línea de datos se llama “SDA” y la de reloj se llama “SCL”.
Su funcionamiento se basa en un con una estructura maestro-esclavo. El maestro y el esclavo pueden enviar y recibir datos pero solo el elemento maestro tiene la capacidad de comenzar una comunicación y transmitir datos o solicitarlos.
La tarjeta TLV320AIC34EVM actúa como elemento esclavo y el DSP como elemento maestro.
Todos los dispositivos conectados a esta línea tienen una dirección única I2C que se define en el TLV320AIC34EVM en unos interruptores y en el DSP de forma lógica. La tecnología I2C tiene además como ventajas el reducido consumo de energía, inmunidad alta frente a ruido, rango amplio de suministro de voltaje y un rango amplio de temperaturas de operación.

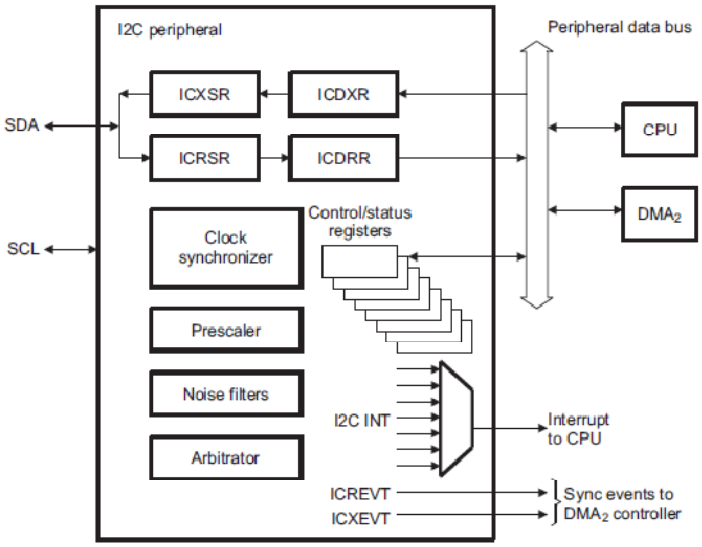


Figura 34. Diagrama módulo I2C. Imagen extraída de [Texas Instruments, "*TMS320VC5505/5504 DSP Inter-Integrated Circuit (I2C) Peripheral User's Guide*", SPRUFO1A].

3.3.1.2 Arquitectura y estructura del bus

El bus consta de dos líneas llamada SDA y SCL. Sobre la línea SDA se transfieren los datos y sobre la SCL se envía la señal de reloj de sincronización. Todos los datos transmitidos se hacen en grupo de 8 bits.
El dispositivo que actúa como maestro es el que genera y envía la señal de reloj por la línea SCL.
El elemento maestro es el que inicia las comunicaciones. En este caso el DSP tiene una serie de funciones que realizan el algoritmo de transmisión. Se comienza enviando una señal que representa una condición de "Start". A través del registro de modo del módulo periférico del DSP se puede enviar la señal de "Start". Este comando provoca, cuando la señal SCL está a nivel alto ('1') una transición de nivel alto ('1') a nivel bajo ('0') en la línea SDA. Del mismo modo, para terminar la comunicación, a través del registro de modo se puede solicitar el fin de la comunicación mediante el envío de la condición de "Stop". También es el maestro el que envía esta condición. Esta condición se refleja a nivel de señal que cuando la línea SCL está a nivel alto, la línea SDA realice una transición de nivel bajo a nivel alto.

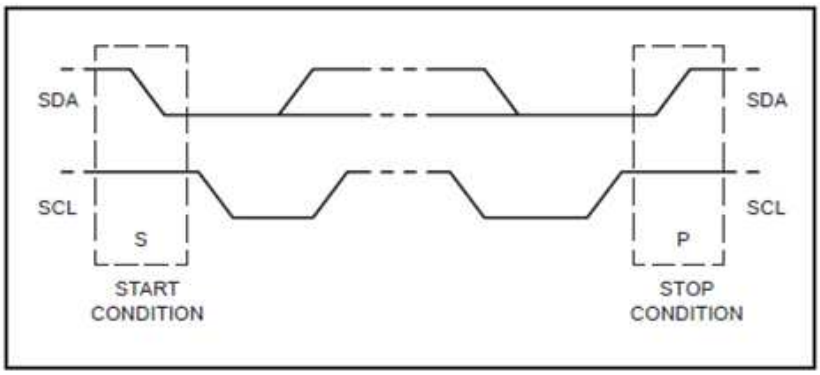


Figura 35. Condiciones START y STOP. Imagen extraída de [Philips Semiconductors, "*The I2C -bus and how to use it*"].

Después de que el maestro haya enviado la señal de comienzo (“Start”), envía una byte, de los cuales 7 bits son de dirección y el octavo bit es de lectura/escritura, con el que quiere comenzar una comunicación.
Para que los datos tengan validez, el dato de la señal SDA tiene que ser estable mientras estamos en el flanco de subida de la señal SCL. Solo se puede cambiar de dato cuando la señal SCL está en el flanco de bajada.

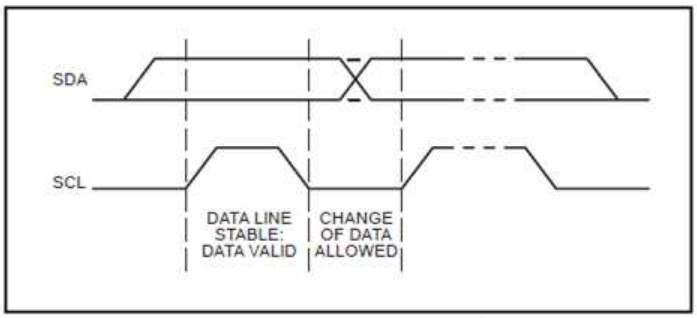


Figura 36. Transferencia de bit. Imagen extraída de [Philips Semiconductors “*The I2C-bus and how to use it”].*

Todos los datos que se envían tienen 8 bits y van seguidos de un asentimiento. La transferencia comienza con el bit más significativo. Es necesario que por cada dato enviado por el transmisor el receptor tenga que enviar un bit de asentimiento. Si el receptor no está listo para continuar con la transferencia de datos puede dejar la señal
SCL a nivel bajo terminar sus tareas y luego volver a poner SCL para la transmisión de datos.

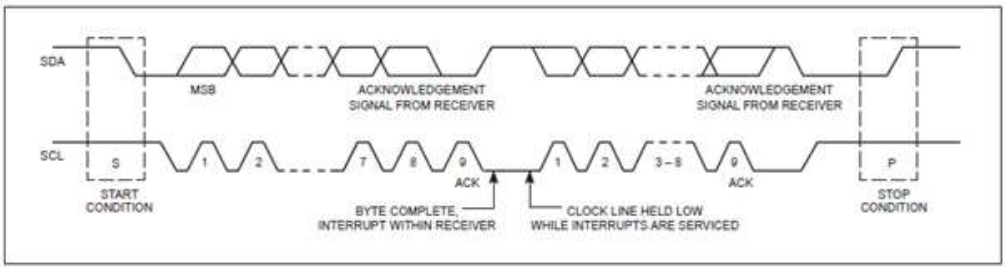


Figura 37. Transferencia de datos. Imagen extraída de [Philips Semiconductors, “*The I2C -bus and how to use it*”].

Esta comunicación se ha definido en el DSP mediante una serie de funciones que modifican los registros de modo del módulo periférico I2C del DSP para que se envíen las señales de “Start”, “Stop”, los datos a escribir y se reciban los datos desde el códec.
Existen dos modos de operación en el bus I2C. El modo estándar (“I2C Standard Mode”) que funciona a 100kbps y el modo rápido que funciona a 400kbps. (“I2C Fast Mode”).

3.3.2 I2S

3.3.2.1 Funcionamiento y propósito

I2S es un bus de datos que permite la transferencia de datos serie de modo "full-duplex" entre el códec TLV320AIC34EVM y el DSP. Este bus solo se encargará de la transferencia de datos de audio.

El módulo periférico I2S del DSP dispone de cuatro canales estéreo. Cada uno de estos canales se compone de dos señales de reloj, una funciona a modo de sincronización de segmento (WCLK) y la otra a modo de sincronización a nivel de bit (BCLK), una señal de recepción de datos (RX) y otra de envío de datos (DX). Este bus tiene una estructura de maestro-esclavo en la cual las señales de reloj son generadas por el elemento maestro. Las señales de transmisión y recepción de datos son unidireccionales y la primera es una señal de salida y la segunda de entrada. A su vez el maestro envía los datos por la línea de transmisión (DX) y los recibe por la línea de recepción (RX). También existe otro tipo de configuración en el cual el dispositivo que transmite información es el esclavo y el que recibe la información es el maestro. La diferencia reside en el que recibe la información es aquí el que define las frecuencias de reloj. Los datos se mueven en la misma dirección que en el otro esquema.

Las señales WCLK y BCLK se corresponden con las señales I2Sn_FS y I2Sn_CLK del DSP respectivamente. La primera de las señales transportará la frecuencia de muestreo del ADC o del DAC. En ambos extremos se deben conectar físicamente las líneas I2Sn_FS de ambos extremos entre sí así como las líneas I2Sn_CLK.
Las otras dos señales que hay son las que transfieren datos y los reciben. La primera de estas señales es el puerto que envía datos (DX) que corresponde con la línea I2Sn_DX.

La última línea es la que recibe datos (RX) que corresponde con I2Sn_RX. Las conexiones físicas se hacen uniendo el puerto I2Sn_Dx del extremo transmisor con el puerto I2Sn_RX del extremo receptor así como I2Sn_RX del transmisor con la señal I2Sn_DX del receptor.

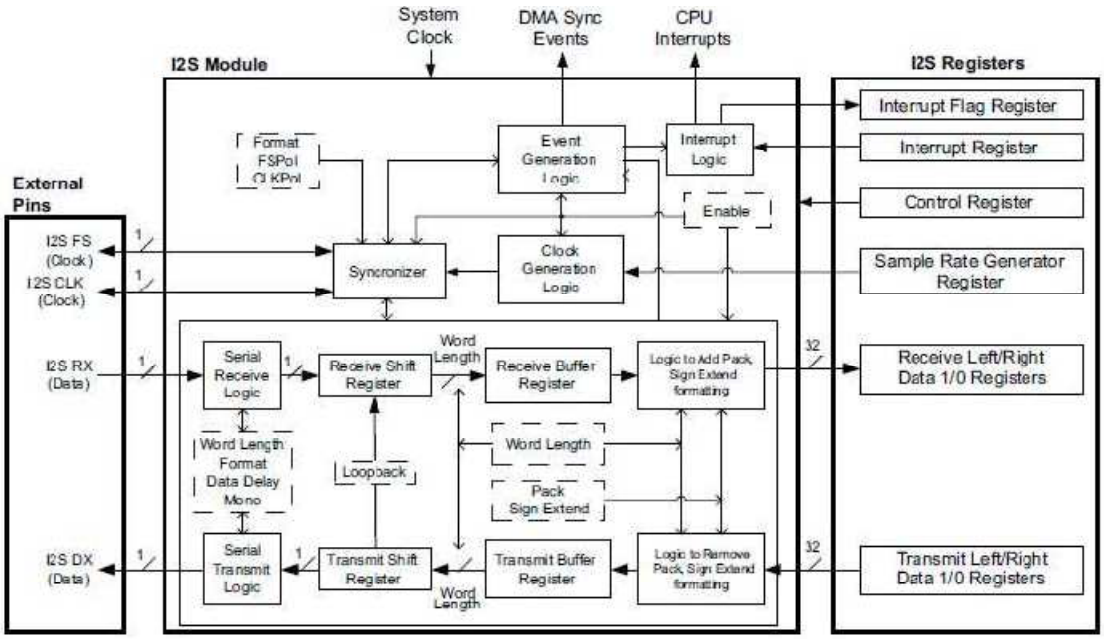


Figura 38. Bloque funcionamiento I2S en DSP. Imagen extraída de [Texas Instruments,"*TMS320VC5505 DSP Inter-IC Sound (I2S) Bus User's Guide*", SPRUFP0].

3.3.2.2 Protocolos de transmisión

Una palabra es un grupo de bits que forman un dato. La longitud de las palabras se define en el registro de control del I2S (I2SSCTRL) en el módulo periférico I2S del DSP. Un marco es un grupo de palabras (una palabra si la información es en "mono" y dos si es en "stereo") que forman un dato transmitido o recibido. La longitud de las palabras puede ser de 16, 20, 24 o 32 bits.

El bus de datos I2S tiene la posibilidad de comunicarse con otros dispositivos mediante cuatro formatos, "I2S", "DSP", "Right Justified" y "Left Justified", que se describen a continuación:

1. I2S: El bit más significativo del canal izquierdo es válido en el segundo flanco de la señal de reloj de bit (BCLK). El bit más significativo del canal derecho es válido en el segundo flanco de la señal de reloj de bit (BCLK) y cuando la señal de reloj de palabra cambie de estado.

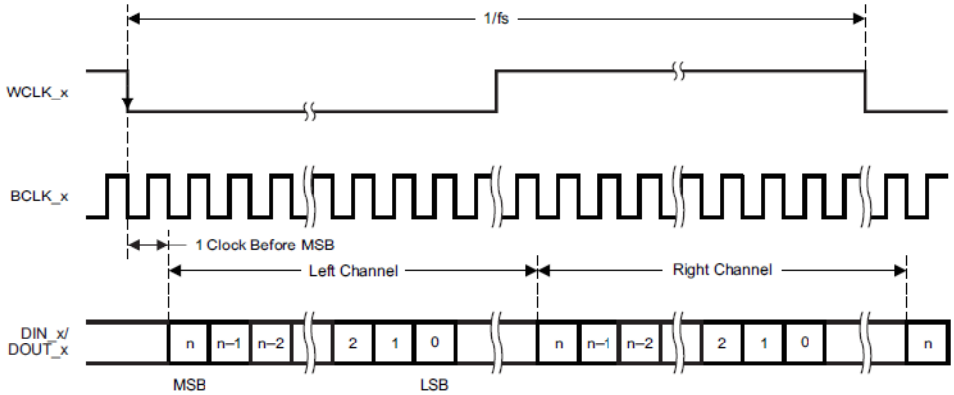


Figura 39. Esquema transmisión en modo I2S. Imagen extraída de [Texas Instruments,"*TMS320VC5505 DSP Inter-IC Sound (I2S) Bus User's Guide*", SPRUFP0].

2. DSP: La transferencia de datos comienza cuando hay un flanco de subida de la señal de reloj de palabra (WCLK) empezando por el canal izquierdo seguido inmediatamente por el canal derecho.

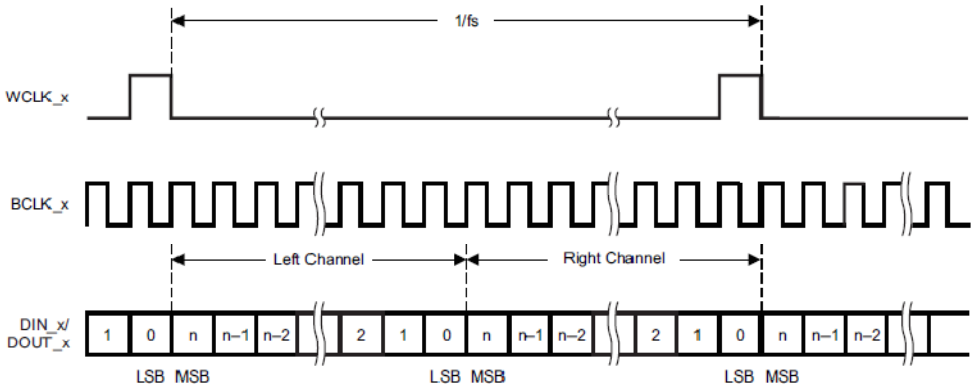


Figura 40. Esquema transmisión en modo DSP. Imagen extraída de [Texas Instruments,"*TMS320VC5505 DSP Inter-IC Sound (I2S) Bus User's Guide*", SPRUFP0].

3. Left Justified: El canal izquierdo es válido durante todo el flanco de subida de la señal de reloj de palabra (WCLK) y el canal derecho durante todo el flanco de bajada. Además lo primero que se transmite es el bit más significativo del canal. Si al final sobrase espacio de datos no se rellenaría con nada.

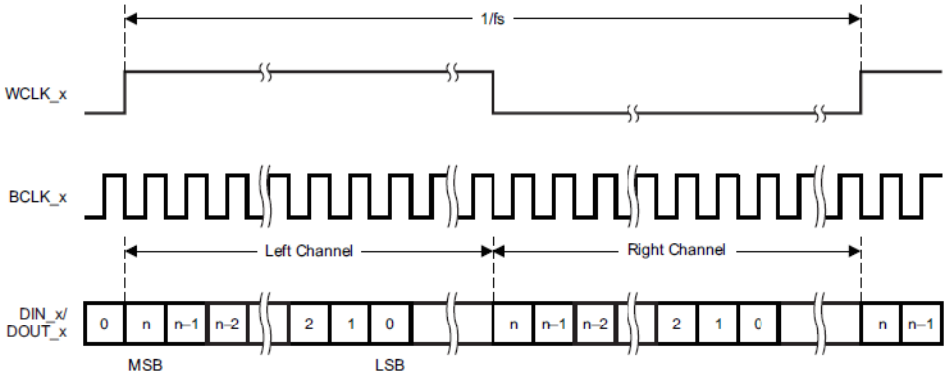


Figura 41. Esquema transmisión en modo Left-justified. Imagen extraída de [Texas Instruments,"*TMS320VC5505 DSP Inter-IC Sound (I2S) Bus User's Guide*", SPRUFP0].

4. Right Justified: Se transmite el canal izquierdo empezando por el bit más significativo con el flanco de la señal de reloj de palabra (WCLK).
Funciona exactamente igual que el modo anterior a excepción de que el bit más significativo no es el primero si no de relleno. Sin embargo el último bit transmitido de cada canal es el menos significativo del dato de ese canal.

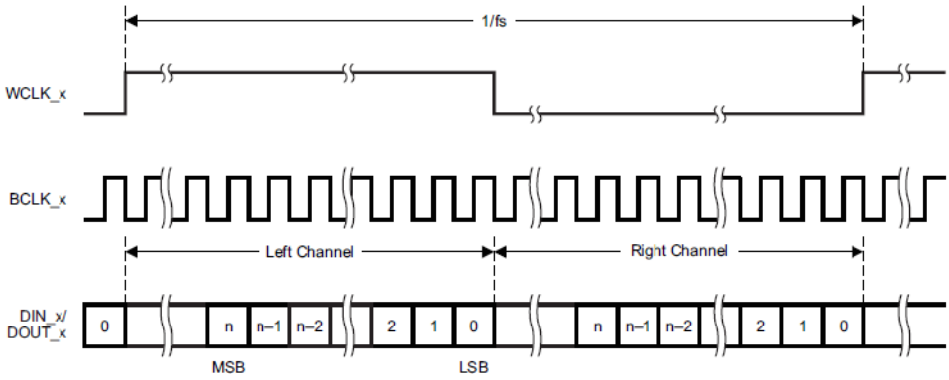


Figura 42. Esquema transmisión en modo Right-justified. Imagen extraída de [Texas Instruments,"*TMS320VC5505 DSP Inter-IC Sound (I2S) Bus User's Guide*", SPRUFP0].

Se ha escogido para el diseño de la transmisión de datos entre el CODEC AIC34 y el DSP 5505 el modo I2S. Se han detectado problemas a la hora de seleccionar al CODEC AIC34 como maestro en la comunicación I2S. Se ha decidido escoger al DSP como maestro de la comunicación. Para ello escoger el modo I2S es computacionalmente el más sencillo. Los micrófonos digitales capturan el audio del canal izquierdo en el flanco de bajada del reloj de sincronización y el canal derecho en el de subida. Debido al fallo en la elección del CÓDEC como dispositivo maestro en la comunicación I2S y a la arquitectura de generación de las frecuencias de reloj WCLK y BCLK en el DSP, lo más sencillo es generar a partir de la señal interna de reloj del DSP generar las señales WCLK BCLK a través de unos divisores específicos.

Como comprobación de la interconexión se realiza un programa a modo de demostración que en primer lugar configura desde el DSP el AIC34 definiendo la frecuencia de muestreo a 44,1 kHz y la longitud de palabras de 16 bits. Esta demostración solo permite la transmisión de datos desde un micrófono digital de dos canales desde el códec hacia el DSP y se devuelve el audio sin procesar hacia el códec de nuevo para que lo decodifique y se pueda escuchar a través de unos altavoces o unos auriculares el audio que se ha adquirido.

4.DISEÑO Y DESARROLLO SOFTWARE

El objetivo del proyecto es la realización de un prototipo que adquiera audio a tempo real a través de micrófonos digitales y codifique la información a tiempo real. Una vez elegidos los dispositivos que compondrán el sistema final, realizado la interconexión a nivel físico y seleccionado los modos de transmisión de datos se define el programa para el control del prototipo y la codificación a tiempo real.
El software que se desarrolla es una demostración de las capacidades del prototipo y debe poder realizar las tareas de configuración del CODEC, gestión del bus de control I2C y del bus de datos I2S, recepción de las muestras provenientes del CODEC, codificación a tiempo real de las muestras de audio y la configuración del propio DSP y sus módulos.
Se va a describir como se ha desarrollado el programa para que realice tanto el control sobre el CODEC como las transmisiones y la codificación. Para hacer funcionar el sistema completo, se debe primero configurar el DSP para que tenga los parámetros adecuados en cuanto a la frecuencia de reloj, como a la configuración de los buses de datos series y paralelos y la inclusión de todas las funciones necesarias. Esta aplicación de demostración se hace programando el DSP.

4.1 Entorno de desarrollo

El primer paso es la descripción del entorno de trabajo para el desarrollo de la aplicación de control de la lógica del DSP.

4.1.1 Introducción al entorno

El desarrollo del programa se realiza en el entorno de programación Code Composer Studio v.4.1.2. Se trata de un entorno de programación que cuenta con un compilador y un enlazador (linker) que permite programar en los lenguajes C, C++ o DSPIOS (lenguaje propietario de Texas Instruments específico para el entorno basado en C). Para ejecutar un programa en el DPS se debe realizar un proyecto que contenga todos los ficheros necesarios para la compilación. Un proyecto debe tener asociado los siguientes tipos de archivos:

- ccxml: archivo de configuración básico para definir la tarjeta específica como DSP y el tipo de emulador para la carga de archivos compilados. Para la aplicación del DPS TMS320VC5505 eZdsp hay una opción específica “USBSTK5505” que hay que combinar con el emulador XDS100 v1 USB.
- lnkx.cmd: mapa de memoria del sistema donde se definen las secciones de alojamiento de las variables, la memoria dinámica, las constantes, las tablas,

los buffer y las interrupciones dentro de la memoria así como las direcciones de los puertos de entrada/salida.

- asm: ficheros con información en código ensamblador para poder definir funciones a bajo nivel por criticidad de tiempo y declaración de direccionamiento de código de interrupciones.
- h: ficheros de cabecera con declaración de funciones en código C, declaraciones de estructuras de datos y direcciones de registros del DSP.
- c: fichero con definición de funciones y código en lenguaje C. Puede haber tantos ficheros de tipo .c para definir las funciones que realizará el código como sean necesarias.
- GEL: ficheros de arranque de valores de DSP. Cuando se conecta un dispositivo al equipo Host que le transferirá la aplicación el archivo tipo GEL inicializa valores del DSP.

El entorno permite agrupar todos los ficheros declarados en un proyecto. Primero se compila el código para transformar de código C a código máquina y luego se enlazan todos los ficheros de código máquina para unir todos los ficheros en un objeto que se transferirá al dispositivo.

4.1.2 Tipos de datos y registros

La tarjeta que contiene el DSP escogido para el prototipo es el TMS320VC5505 USBSTK eZdsp. Los núcleos de la serie C55x de *Texas Instruments* trabajan con tipos de datos de 16 bits por palabra. Para la implementación del código se ha elegido el lenguaje C. En consecuencia se pretende trabajar con tipos de datos y sintaxis propia de ese lenguaje. Gracias al compilador de CCStudio v.4.1.2 se puede definir todo el código salvo varias excepciones en ese lenguaje. Los tipos de datos usados en C en el CCStudio para usarlos en un DSP de la serie C55x son distintos de los usados en un compilador de lenguaje C de un sistema Host en procesador de propósito general.

Tipo de dato	Resolución estándar	Resolución en C55x
char	8 bits	16 bits
short	16 bits	16 bits
int	32 bits	16 bits
long	32 bits	32 bits
long long	32 bits	40 bits
float (coma flotante)	32 bits	32 bits
double (coma flotante)	64 bits	40 bits

Tabla 15. Tipos de datos.

Los tipos de datos tienen en cuenta si son de coma flotante o no. Por ejemplo un int y un double tiene la misma resolución, pero el primero define el número en coma fija y el segundo en coma flotante. A través de esta tabla se realizan las

conversiones de tipos de datos para poder trabajar con palabras de 16 bits en el entorno de desarrollo.

4.2 Esquema de funcionamiento del prototipo

El prototipo a nivel hardware ya se ha descrito y se va a definir como se ha realizado el programa de demostración que permita seleccionar una frecuencia de muestreo, elegir un tipo de micrófono y una longitud de palabra digital paraqué se codifique a formato MS ADPCM a tiempo real. El programa se divide en una parte de configuración de los elementos que integran el prototipo incluyendo el reloj del sistema, los periféricos del DSP, el AIC34 a través del I2C y el AIC3204 para poder escuchar dos canales de salida. Una vez configurados los periféricos del DSP, los códecs y el árbol de generación del reloj se comienza a adquirir audio y codificar a tiempo real las muestras. El proceso de configuración sigue el siguiente orden.

1. **Obtención de parámetros de audio:**
    - Obtención de frecuencia de muestreo : 48 kHz; 44,1 kHz; 22,05 kHz; 16 kHz; 8 kHz.
    - Obtención de la longitud de las palabras I2S: 16, 20, 24 o 32 bits
    - Definición tipos de micrófonos:
        - Bloque A y B micrófonos digitales.
        - Bloque A micrófonos digitales y Bloque B micrófonos analógicos
        - Bloque A micrófonos analógicos y Bloque B micrófonos digitales
        - Bloque A y B micrófonos analógicos.
2. **Inicialización parámetros DSP:**
    - Configuración del árbol de reloj en función de la frecuencia de muestreo obtenida.
    - Configuración de los buses serie y paralelo para habilitar los puertos I2S0, I2S1, I2S2, GPIO16 GPIO17 e I2C.
3. **Configuración GPIO:**
    - Definición puertos GPIO16 y GPIO17 como salida.
    - Escritura de '1' en GPIO16 y GPIO17 para realizar RESET de software en bloques A y B del AIC34.
4. **Configuración I2C:**
    - Definición de reloj interno.
    - Definición de frecuencia de funcionamiento de la línea transmisora.
5. **Configuración AIC34 Bloque A:**
    - Definición de los registros en función de los parámetros obtenidos (frecuencia de muestreo, longitud de palabras y tipo de micrófonos) para adquisición de audio, transmisión al DSP y reproducción del audio decodificado.
6. **Configuración AIC34 Bloque B:**

- Definición de los registros en función de los parámetros obtenidos (frecuencia de muestreo, longitud de palabras y tipo de micrófonos) para adquisición de audio y transmisión al DSP.

7. **Configuración AIC3204:**
   - Definición de los registros en función de parámetros obtenidos (longitud de palabras) para la reproducción del audio decodificado.
8. **Configuración I2S:**
   - Definición de las señales BCLK y WCLK en función de la frecuencia de muestreo.
   - Definición del modo de funcionamiento en función de la longitud de palabras.

La segunda parte del software del prototipo realiza la adquisición de audio de los cuatro canales y su codificación y decodificación MS ADPCM a tiempo real. Para ello primero se configuran interrupciones software que correspondientes a los canales I2S de recepción y transmisión de los datos de audio a codificar y decodificado. De esta forma cuando un dato llega o está listo para ser enviado al códec para reproducirlo, se interrumpe el proceso activo y se pasa transmite o recibe el dato. Los datos que se van recibiendo se guardan en buffers específicos que se crean de un determinado tamaño. Se generan buffers de igual tamaño al de los bloques que codifica el algoritmo MS ADPCM que viene determinado por una función que se explicará más adelante. Una vez que los buffers de los datos de entrada están llenos, se comienza a codificar el bloque completo. Una vez codificado el bloque, se decodifica y se transfiere a un buffer circular que tiene una vez y media el tamaño del buffer de entrada. De esta forma se consigue evitar la pérdida de datos si no hay una sincronización entre la cantidad de datos entrantes y salientes. Cada bloque que se captura se codifica e inmediatamente después se decodifica para ser escuchado y comprobar que el sistema funciona correctamente.

4.3 Desarrollo del control y periféricos del DSP

Para la programación del sistema de captura de audio hace falta programar el DSP para que adquiera el audio procedente del CODEC, codifique los diferentes canales a tiempo real de manera paralela y configure los buses de datos, de control y el CODEC AIC34. Para la realización del sistema es necesario habilitar los módulos periféricos I2C, I2S y GPIO del TMS320VC55005.

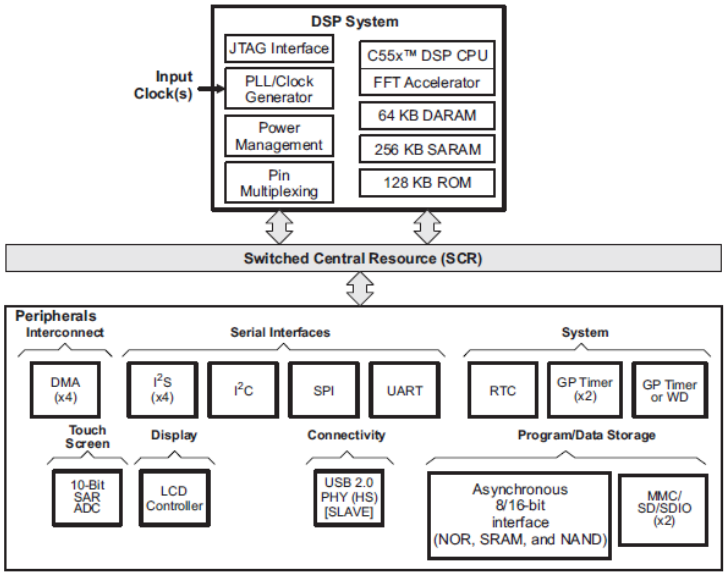


Figura 43. Diagrama funcional de DSP. Imagen extraída de [Texas Instruments, "*System User's Guide TMS320VC5505*", SPRUFP0].

4.3.1 Configuración parámetros DSP

En primer lugar se configuran son los parámetros propios del DSP. El TMSVC5505 es un dispositivo digital que se configura mediante la escritura de registros. Por ello se define en un fichero de cabecera las direcciones de los registros del sistema así como los tipos de datos que se van a usar en el programa en un fichero de cabecera. Estas direcciones se pueden encontrar en el manual del usuario del TMS320VC5505 eZdsp. Al no usar todos los periféricos disponibles en el DSP solo se enumeran aquellos a los que se va a acceder (Ver anexo 7.2).

El registro EBSR (Ver Anexo 7.2) controla los dos puertos serie (0 y 1) y el puerto paralelo. Los puertos del DSP están multiplexados y se habilitan los que se van a usar. Se analizan las posibles combinaciones que permiten los puertos serie (tres posibles modos cada puerto) y paralelo (seis posibles modos) y se selecciona el modo 1 tanto en el puerto serie 0 como en el 1 para habilitar los canales de comunicación I2S0 e I2S1 respectivamente. El canal de comunicaciones I2S0 es un bus de datos exclusivo y dedicado para la comunicación entre el DSP TMS320VC5505 y el CODEC AIC3204 montado en la propia tarjeta TMS320VC5505 eZdsp. Para el programa de demostración se requiere del uso CODEC AIC3204 para la salida de audio y comprobar el correcto funcionamiento de la decodificación. El bus I2S1 será uno de los canales de transmisión de información de audio entre el códec AIC34 y el DSP. Del puerto paralelo, se selecciona el modo 6 para habilitar el canal I2S2 que será el segundo canal de transmisión de información de audio entre el AIC34 y TS320VC5505 y para habilitar los pines GPIO16 y GPIO17 que se usarán como señales de reset software de los bloques lógicos del códec AIC34.

El siguiente paso es la configuración del árbol de reloj del DSP con el propósito de que la señal de reloj llegue a todos los periféricos que se van a usar. Para ello hay que definir los valores de los registros *PCGCR1* y *PCGCR2* (Ver. Anexo 7.1).
Se habilita el paso de reloj a los módulos que van a ser utilizados para no generar un consumo eléctrico mayor del necesario. Se configura el árbol de reloj para que los módulos periféricos I2C, I2S0, I2S1, I2S2 reciban la señal de reloj maestro y se habilita para que el reloj del sistema también alimente a la CPU del DSP. El resto de periféricos se dejan deshabilitados por motivos de consumo energético. Para ello el registro *PCGCR1* se define con el valor 0x3CBE y el registro *PCGCR2* con el valor 0x003B. Cada campo de los registros corresponde con un módulo de DSP y es un registro que escribiendo un '0' se habilita y un '1' se deshabilita.

4.3.2 Configuración de periféricos

Se va a hacer uso de 3 módulos periféricos del DSP en el sistema final con las funcionalidades de buses de datos, bus de control y señal de reset del AIC34. Se va a comenzar explicando la configuración de módulo I2S porque a partir de sus valores se configurar el resto de módulos.

4.3.2.2 I2S

El bus de datos I2S se usará para la recepción y transmisión de información entre el CODEC AIC34 y el DSP y entre el códec embebido AIC3205 y el DSP. Se dispone de 4 buses de datos I2S, que son I2S0, I2S1, I2S2 e I2S3. El canal I2S0 está dedicado a la comunicación entre el códec AIC3205 y el DSP. Se usarán los canales I2S1 e I2S2 para la transmisión de información entre el DSP y el AIC34.
Para poder trabajar con los canales I2S hay que configurar las señales de reloj del módulo. El bus I2S funciona con una señal de sincronización de palabras (WCLK) y otra de sincronización de bits (BCLK). Ambas señales se definen a partir de la señal del reloj del sistema SYSCLK. Mediante unos divisores, llamados CLKDIV y FSDIV se generan las señales WLCK y BCLK.

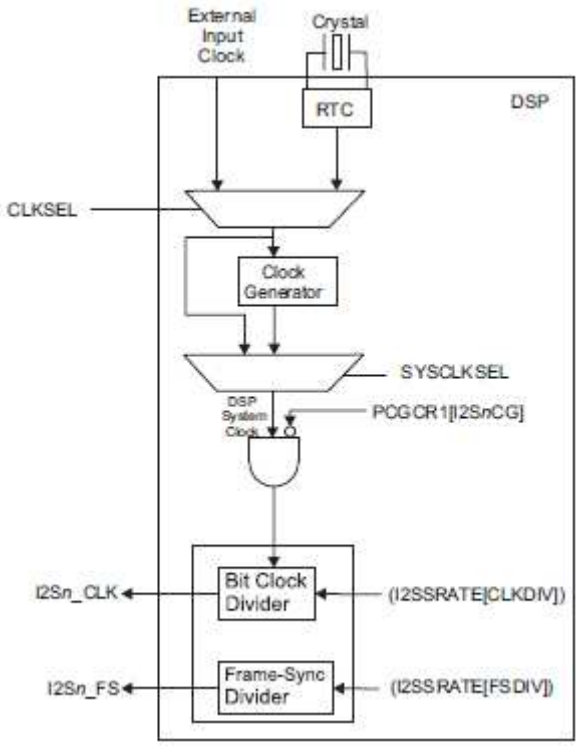


Figura 44. Generación de WCLK y BCLK. [Texas Insrtuments, "*TMS320VC5505/5504 DSP Inter-IC Sound (I2S) Peripheral User's Guide*", SPRUFP4].

Se ha seleccionado al DSP TMS320VC5505 cómo elemento maestro en la comunicación dado que cuando se ha hecho pruebas para que el CODEC AIC34 fuera el elemento maestro, ha dado muchos problemas porque no genera las señales de reloj forma correcta. El programa de demostración final permite escoger entre varias frecuencias de muestreo (48 kHz, 44,1kHz, 22,5kHz 16kHz o 8 kHz) y trasmitir las muestras adquiridas con diferentes longitudes de palabra (32, 24, 20 o 16 bits). Dependiendo de la frecuencia de muestreo y de la longitud de las palabras, los divisores tendrán diferentes valores. El maestro es el que genera las señales de reloj pero puede transmitir y recibir información.
La relación entre las frecuencias de las señales WCLK y BCLK con el reloj del sistema (SYSCLK) es:

$$BCLK = SYSCLK/(2^{(CLKDIV+1)})$$

$$WCLK = BCLK/(2^{(FSDIV+3)})$$

Donde CLKDIV y FSDIV son los divisores que hay que calcular e introducir en el registro I2SRATE. Para ello se aplica el siguiente algoritmo:
1º Cálculo de FDIV:

$$2^{FSDIV+3} \geq 2 \cdot Longitud\ de\ palabra\ (para\ canales\ estéreo)$$

⇨ $FSDIV \geq (\log_2(2 \cdot Longitud\ de\ palabra)) - 3$

2º Como conocemos la frecuencia de muestreo que coincide con la señal WCLK, podemos calcular la frecuencia de BCLK:

$$BCLK = WCLK \cdot (2^{(FSDIV+3)})$$

3º Cálculo de CLKDIV:

$$SYSCLK \geq BCLK \cdot (2^{(CLKDIV+1)})$$

Sabiendo que SYSCLK no puede mayor de 100 MHz, se obtiene la siguiente ecuación:

$$CLKDIV \leq (\log_2(SYSCLK/BCLK)) - 1$$

4º Ahora se calcula la frecuencia justa del reloj del sistema para obtener las frecuencias exactas de BCLK y WCLK con los divisores calculados:

$$SYSCLK = BCLK \cdot (2^{(CLKDIV+1)})$$

Se han realizado los cálculos para todas las posibles combinaciones de frecuencias de muestreo y se muestran en la tabla a continuación. El prototipo está diseñado de tal forma que cuando se elige una frecuencia de muestreo, el reloj del DSP se define como en la columna SYSCLK:

Tasa de muestreo	Longitud de palabra	FSDIV	BCLK	CLKDIV	SYSCLK	I2SRATE
48 kHz	16 bits	≥ 2	1,536 MHZ	5	98,304 MHZ	0x0015
	32 bits	≥ 3	3,072 MHZ	4	98,304 MHZ	0x001C
	24 bits	≥ 3	3,072 MHZ	4	98,304 MHZ	0x001C
	20 bits	≥ 3	3,072 MHZ	4	98,304 MHZ	0x001C
44,1 kHz	16 bits	≥ 2	1,4112 MHZ	5	90,3168 MHZ	0x0015
	32 bits	≥ 3	2,8224 MHZ	4	90,3168 MHZ	0x001C
	24 bits	≥ 3	2,8224 MHZ	4	90,3168 MHZ	0x001C
	20 bits	≥ 3	2,8224 MHZ	4	90,3168 MHZ	0x001C
22,05 kHz	16 bits	≥ 2	0,7056 MHZ	6	90,3168 MHZ	0x0016
	32 bits	≥ 3	1,4112 MHZ	5	90,3168 MHZ	0x001D
	24 bits	≥ 3	1,4112 MHZ	5	90,3168 MHZ	0x001D
	20 bits	≥ 3	1,4112 MHZ	5	90,3168 MHZ	0x001D
16 kHz	16 bits	≥ 2	0,512 MHz	6	65,536 MHz	0x0016
	32 bits	≥ 3	1,024 MHz	5	65,536 MHz	0x001D
	24 bits	≥ 3	1,024 MHz	5	65,536 MHz	0x001D
	20 bits	≥ 3	1,024 MHz	5	65,536 MHz	0x001D
8 kHz	16 bits	≥ 2	0,256 MHz	7	65,536 MHz	0x0017
	32 bits	≥ 3	0,512 MHz	6	65,536 MHz	0x001E
	24 bits	≥ 3	0,512 MHz	6	65,536 MHz	0x001E
	20 bits	≥ 3	0,512 MHz	6	65,536 MHz	0x001E

Tabla 16. Valores de configuración de I2S.

La configuración del reloj del sistema (SYSCLK) se describirá en el capítulo 4.2.3.2.

La segunda parte es la de la configuración del modo de funcionamiento de módulo I2S. Primero se define el registro I2SRATE con los valores de los divisores, después el modo de funcionamiento en el registro I2SSCTRL. Para cada frecuencia de muestreo que se quiere seleccionar existe un valor I2SRATE que define el valor adecuado de los divisores para generar las señales de reloj BCLK y WCLK adecuados.
Para definir el modo de funcionamiento del módulo I2S se describen los valores del registro de modo. A través de este registro realizamos las siguientes acciones:

- Habilitamos el módulo I2S escribiendo un '1' en el campo "ENABLE".
- Definimos que habrá dos canales de transmisión poniendo el valor '0' en el campo "MONO".
- Definimos la polaridad de la ventana para el modo I2S de transmisión escribiendo un '0' en el campo "FSPOL".
- Definimos la longitud de las palabras de los datos: 16, 20, 24 32 bits.
- Se define el modo de funcionamiento como modo maestro escribiendo un '1' en el campo "MODE".

La recepción y transmisión de los datos se hace mediante interrupciones. Cuando un dato está listo para ser recibido o transmitido, una bandera de interrupción salta, se deja de realizar el proceso en uso y se transmite o recibe el dato que está listo. Para ello se activan las opciones de interrupción de transmisión estéreo "XMITST" y recepción estéreo "RCVST" del registro de control de interrupciones I2SINTMASK (Ver. Anexo 7.1).

4.3.2.1 I2C

El bus I2C se usará para realizar la configuración del CODEC AIC34 desde el DSP. Primero se configura el módulo en el DSP para que funcione a la frecuencia correcta y con los parámetros de transmisión adecuados. Las direcciones de los registros de control del módulo I2C se han definido previamente en el punto 4.2.1.
Para que el módulo I2C funcione se definen las frecuencias de la señal de sincronización (SCL), del funcionamiento del módulo periférico y el modo de operación. Ambas señales de reloj se definen a partir del reloj del sistema.

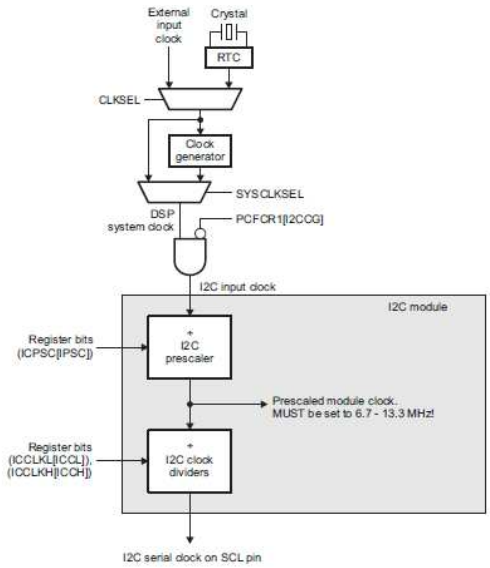


Figura 45. Estructura de creación de las señales de reloj del modulo I2C. [Texas Instruments, "*TMS320VC5505/5504 DSP Inter-Integrated Circuit (I2C) Peripheral User's Guide*", SPRUFO1A].

Para que el módulo I2C funcione, la frecuencia de la señal de reloj debe estar comprendida entre 6,7 y 13,3 MHz. A través de un divisor se define la señal en función del reloj del sistema del DSP. El valor del divisor se define en el registro ICPSC y se calcula de la siguiente manera:

$$Frecuencia\ de\ reloj\ I2C = {Señal\ de\ entrada}/{(IPSC + 1)}$$

El valor de IPSC puede estar comprendido entre 0 y 255. Como se ha visto en el punto anterior la señal del reloj del sistema está definida por la frecuencia de muestreo en la adquisición del audio. Se ha calculado un valor adecuado de IPSC en función de las diferentes posibilidades de la señal de entrada.

Frecuencia señal de muestreo	Frecuencia de reloj del sistema	Frecuencia reloj I2C	IPSC
48 kHz	98,304 MHz	9,8304 MHz	9
44, kHz	90,3168 MHz	9,8304 MHz	9
22,05 kHz	90,3168 MHz	9,03168 MHz	9
16kHz	65, 536 MHz	10,93 MHz	5
8 kHz	65, 536 MHz	10,93 MHz	5

Tabla 17. Parámetros I2C.

El siguiente paso en la configuración de los módulos periféricos del DSP es la definición de la frecuencia de la señal SCL (señal de sincronización de comunicación entre los dispositivos conectados) a través de los registros ICCLKL e ICCLKH, que representan las frecuencias del flanco de bajada y de subida de la señal de sincronización. Para el correcto funcionamiento de las transmisiones I2C se decide definir el la señal SCL a unos 50 kHz Esta frecuencia es óptima para que el códec tenga la capacidad de leer los datos. Definimos ambos flancos de reloj con la misma frecuencia a través de la siguiente ecuación:

$Frecuencia\ SCL = {Frecuencia\ reloj\ I2C}\Big/{((ICCLKL + d) + (ICCLKH + d))}$, donde

"d" tiene el siguiente valor en función de IPSC:

Valor IPSC	Valor "d"
0	7
1	6
2 a 255	5

Tabla 18. Valor constante "d". Tabla extraída de [Texas Instruments, "*TMS320VC5505/5504 DSP Inter-Integrated Circuit (I2C) Peripheral User's Guide*", SPRUFO1A].

El valor de IPSC para todos los casos que se aplican en el sistema hace que "d" sea siempre 5. El valor que se obtiene al aplicar la fórmula para los distintos casos se muestra en la siguiente tabla:

Frecuencia reloj I2C	Frecuencia SCL	"d"	ICCLKL	ICCLKH
9,8304 MHz	49,15 kHz	5	95	95
9,03168 MHz	45,16 kHz	5	95	95
10,93 MHz	43,7 kHz	5	120	120

Tabla 19. Factores cálculo divisores I2C.

El último paso es la configuración del modo de operación se define en el registro ICMDR (Ver Anexo 7.1). Solo se define en este registro que el DSP debe ser el elemento transmisor y por ello el único que puede comenzar con a comunicación. El registro de modo se cambia a medida que se requiere para comenzar, finalizar, transmitir y recibir los datos.

Para cada escritura o lectura se modifica el registro ICMDR para adaptarse al modo de funcionamiento requerido. En modo de escritura siempre se definirá que se lance una señal de "Start", se selecciona que el módulo I2C del DSP será el maestro, se habilita el I2C y que funcione sin detenerse ante un breakpoint. Al finalizar la transmisión de los datos se envía una señal de STOP. En este caso se enviarán dos datos, el primero será la dirección de 8 bits del registros y el segundo será el valor que queremos que tenga ese registros.
En el modo de lectura se realiza primero una escritura pero con un solo dato, que será la dirección de registro del que se quiere leer y a continuación se procederá a realizar la lectura donde se vuelve a comenzar con una señal de "Start" indicando de nuevo la dirección del registro que se quiere leer pero ahora el dispositivo esclavo envía al maestro el dato del registro indicado. Este mecanismo es el que se utiliza para leer y escribir registros entre el DSP y el AIC34.

4.3.2.3 GPIO

El bus de datos GPIO (General Purpose Input Output) se va a utilizar como señales de RESET del CODEC AIC34. El CODEC tiene para cada bloque un puerto de señal de reset.

Se han elegido los puertos GPIO16 y GPIO17 para realizar la función de RESET del CODEC. Primero se configuran los puertos 16 y 17 como salidas. Para ello se escribe en la posiciones 0 y 1 del registro IODIR2, que controla el modo de entrada o salida de los puertos 16 a 31, el valor "1".
Una vez que los módulos y el reloj del sistema, que se definirá como se configura en el siguiente apartado, se envía a través de los pines GPIO16 y GPIO17 una señal de reset escribiendo un '0' en las posiciones 0 y 1 del registro IOOUTDATA2.

4.3.3 Configuración de los relojes del sistema

Una vez elegidos los parámetros de configuración de los periféricos sistema final hay que definir como se generan las diferentes opciones de señales del reloj del sistema del DSP.

4.3.3.1 Estructura de generación del reloj del sistema

El sistema de generación del reloj del DSP se compone de un PLL, un multiplicador y varios divisores programables. A la entrada del módulo de generación existe un multiplexor que permite tener como señal de entrada una señal de reloj que uno desee o por el contrario una señal ya definida proveniente del periférico RTC (Real Time Clock) que aporta una señal a 32,768 kHz. Dado que el módulo ya se encuentra integrado en la tarjeta del TMS320VC5505 eZdsp USBSTK se elige la segunda opción como fuente de reloj. En el sistema se hace uso como oscilador generador de señal de reloj el que se encuentra en la tarjeta del TMS32VC5505 eZdsp y por ello se escribe un '0' en el registro PCCR1 (Ver Anexo 7.1)

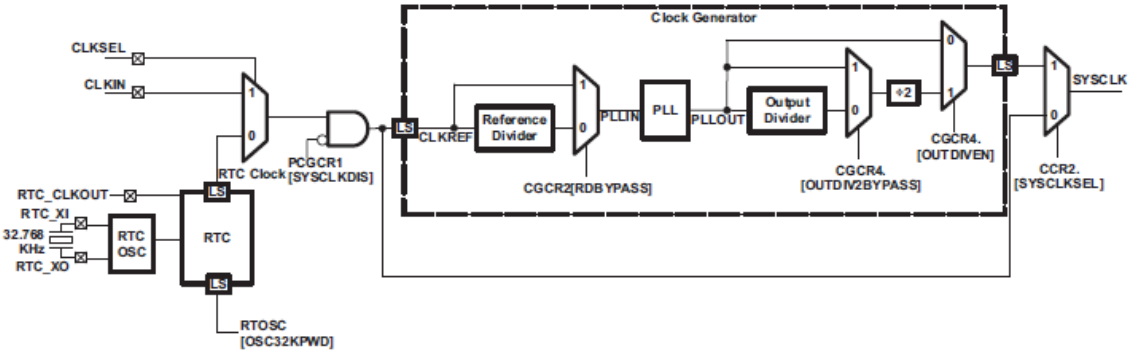


Figura 46. Diagrama de bloques funcional del sistema generador de reloj del 5505. Imagen extraída de [Texas Instruments, "*System User's Guide TMS320VC5505*", SPRUFP0].

A partir de la señal procedente del RTC mediante los componentes del sistema generador se van a crear las frecuencias de reloj que se requieren para poder adquirir audio en todas las frecuencias propuestas. En la tabla se muestran las frecuencias que hay que generar como reloj del sistema para lograr generar las frecuencias de muestreo deseadas como se ha mencionado en el punto 4.2.2.2:

Frecuencia de muestreo	Frecuencia del sistema
48 kHz	98,304 MHZ
44,1 kHz	90,3168 MHZ
22,05 kHz	90,3168 MHZ
16 kHz	65,536MHz
8 kHz	65,536MHz

Tabla 20. Correspondencia entre frecuencia de muestreo y reloj del sistema.

A través de los registros CGCR1, CGCR2, CGCR3 y CGCR4 se controla el paso de las señales por los diferentes elementos del generador de señales para crear la frecuencia del reloj del sistema SYSCLK deseada a partir de la frecuencia de entrada con valor 32,768 kHz.

En primer lugar se ha calculado la relación que existe entre las frecuencias del sistema que se deben de generar y la señal de entrada al generador de reloj. Se divide la frecuencia del sistema entre la señal de entrada para comprobar si solo con un factor multiplicativo se pueden generar las frecuencias del sistema deseadas.

Frecuencia del sistema	Frecuencia del sistema / señal de entrada
98,304 MHZ	3000
90,3168 MHZ	2755,25
65,536MHz	2000

Tabla 21. Relación Frecuencias del sistema con señal de entrada.

Para el primer y tercer caso se puede observar que la relación es por un factor entero. Simplemente se multiplicará la señal de entrada por un factor 3000 para el primero caso y por un factor de 2000 en el segundo mediante el PLL y se generará la señal de 98,304 MHz y 65,536 MHz respectivamente. En el segundo caso la relación no es de número entero. El PLL se describe mediante dos elementos para generar las frecuencias de reloj *SYSCLK*.

$$Frecuencia\ SYSCLK = 32{,}768\ kHz \cdot ((4 \cdot MH) + ML + 4)$$

Se ha calculado para las frecuencias del sistema que valores han de tener “MH” y “ML” para mediante la ecuación obtener la frecuencia del sistema. Dado que el segundo caso tiene una relación de factor entero entre la frecuencia final y la entrada al sistema, existe un pequeño error del 0,001%.

Frecuencia SYSCLK	Multiplicadores		Frecuencia obtenida	Diferencia
	MH	ML		
98,304 MHz	749	0	98,304 MHz	0
90,3168 MHz	688	0	90,308608	99,991%
65,536MHz	499	0	65,536MHz	0

Tabla 22. Frecuencias de reloj de sistema generadas.

Existen una serie de pasos para configurar el valor del reloj y definir los valores MH y ML. No hace falta usar ningún divisor para la generación de la señal de reloj. Por ello todos los valores de divisores serán cero.
En primer lugar se habilita el modo bypass del sistema de generación de reloj escribiendo un '0' en el campo SYSCLKSEL del registro CCR2 (Ver Anexo 7.1).
En segundo lugar se borran los valores de los Flip-Flops del sistema escribiendo un '0' en el campo CLR_CNTL y se apaga el PLL escribiendo un '1' en los campos PLL_STANDBY y PLL_PWRDN del registro CGCR1.
Se programan los valores de RDRATIO, ML, y RDBYPASS el registro CGCR2 de manera acorde a las necesidades del sistema, es decir que todos los valores serán 0 para este registro.
El siguiente paso es el cálculo de los parámetros ODRATIO, OUTDIV2BYPASS y OUTDIVEN que se definen en el registro CGCR4. Para la obtención de las diferentes frecuencias de sistema, todos estos parámetros del registro CGCR4 son 0. Luego se define el campo INIT del registro CGCR3 con el valor 0806_{hex}. Siempre se define este registro con este valor.
En el registro CGCR1 hay que especificar los campos PLL_PWRDN = 0, PLL_STANDBY = 0, CLR_CNTL = 1 e introducir el valor del campo MH que corresponde.

Por último se esperar un tiempo para la fijación del PLL que tarda 4 ms comprobando el registro CGCR3. Se cambia el modo del PLL a encendido para que la señal del reloj del sistema provenga del PLL y no del circuito de bypass escribiendo un '1' en el campo SYSCLKSEL del registro CCR2.

4.3.4 Configuración de la memoria

La memoria del DSP TMS320VC5505 es un recurso muy escaso. El DSP solo cuenta con tres bloques de memoria en los cuales se unifican la memoria de datos y de programa. Los dos primeros bloques están basados en memorias tipo RAM. El primero bloque de memoria es del tipo DARAM (Dual-Access RAM) y tiene 64 kbytes de capacidad y la segunda es de tipo SARAM y tiene 256 kbytes de capacidad. El tercer bloque de memoria es de tipo ROM (Read Only Memory) y tiene 128 kbytes de capacidad. La diferencia entre ambas memorias basadas en tecnología RAM es que la memoria DARAM es de doble acceso en un ciclo de reloj y la SARAM de acceso único en un ciclo de reloj. Se pueden acceder a datos en un ciclo de reloj, ya sean dos lecturas, dos escrituras o una lectura y una escritura en un mismo ciclo de reloj.

A su vez el bloque de memoria DARAM está dividido en 8 bloques de 4k palabras y el bloque SARAM en 64 bloques de 4k palabras también.

En total se encuentran 448 kbytes de memoria en la tarjeta distribuidos en 160 kbytes de palabras de 16 bits en memoria RAM y 64 kbytes de palabras de 16 bits en memoria ROM. Existe la posibilidad de agrandar el mapa de memoria a través del módulo EMIF con memorias tipo SRAM, NOR FLASH, NAND FLASH y SRAM hasta 8 Mbyte.

Tipo memoria	Longitud	Tamaño	Tipo de accesos
MMR	0xC0	192 bytes	RWIX
DARAM0	0xff40	64kbytes -192 bytes	RWIX
SARAM0	0x030000	64 kbytes	RWIX
SARAM1	0x010000	64 kbytes	RWIX
PDROM	0x008000	32 kbytes	RIX
IOPORT	0x020000	128 kbytes	RWI

Tabla 23. Distribución del mapa de memoria.

A través del fichero lnkx.cmd se definen las secciones en la memoria. Primero hay que definir los tamaños del *stack* (zona de memoria donde se guardan las variables de paso de las funciones), del *systack* (stack secundario) y del *heap* (zona de memoria para variables que se guardan toda la ejecución del programa). Se decide que el stack debe tener X tamaño, el systack Y tamño y el heap Z tamaño. Debido a que es necesario pasar un bloque completo de datos adquiridos para codificarlos, se requiere un stack de tamaño X para que no se genere un error.

Se definen las secciones lógicas para distribuirlas en las zonas de memoria como muestra la tabla. La distribución se ha realizado de esta forma para facilitar el traspase de bloques de datos de audio de una función a otra. Se ha dado prioridad a que estos bloques se encuentren en la memoria de tipo DARAM que tiene un doble acceso en un ciclo de reloj.

Sección	Zonas de memoria	Descripción
text	SARAM1 y SARAM0	código
stack	DARAM0	stack primario
systack	DARAM0	stack secundario
data	DARAM0	Variables inicializadas
bss	SARAM1	Variables globales y estáticas
const	SARAM1	Datos constantes
sysmem	SARAM0	Memoria dinámica
switch	SARAM1	Tablas de asignación de switch
cinit	SARAM1	Tablas auto inicializadas
pinit	SARAM1	Inicialización de tablas de tipo *fn*
cio	SARAM1	Buffers entrada/salida
args	SARAM1	Argumentos de *main*
Vectores	DARAM0	Vectores de interrupciones
ioport	IOPORT	Variables globales y estáticas de I/O

Tabla 24. Mapeo de zonas lógicas en la memoria física.

El segundo motivo de dicho mapeo es para optimizar el espacio en la memoria y obtener los accesos más rápidos a memoria de datos y programas en las partes

críticas ejecución que son la codificación. El algoritmo de codificación aplicado al sistema trabaja con datos que provienen de bloques de muestras.

4.3.5 Configuración AIC34

El CODEC AIC34 cuanta con 2 páginas de 128 registros cada uno. La configuración del códec se realiza a través del bus I2C. Para ello se han implementado unas funciones específicas para la escritura y lectura de los valores en los registros. De esta forma después de cada escritura se realiza una lectura de un registro comprobando que el valor que contiene es el correcto. Cada bloque del CODEC AIC34 se configura de forma independiente, ya que tienen direcciones I2C distintas. Mediante el interruptor SW2 que se encuentra en la placa TLV320AIC34EVM se puede definir la dirección I2C que tendrá cada bloque del códec AIC34. Mediante la siguiente tabla se muestran las opciones para definir las direcciones de los bloques del códec.

Posición interruptor SW2	Dirección bloque A	Dirección bloque B
"LOW"	0x18	0x19
"HIGH"	0x1A	0x1B

Tabla 25. Direcciones I2C códec AIC34 Bloque A y Bloque B.

La tarjeta de evaluación TMS320VC5505 eZdsp tiene el códec AIC3204 integrado y también se controla y configura mediante el bus I2C. Dado que la dirección I2C de este códec es 0x18 se ha decidido colocar el interruptor SW2 en posición "LOW" para que las direcciones del bloque A sea 0x1A y del bloque B 0x1B.
La rutina de configuración del códec realiza las escrituras en los registros para que el códec adquiera audio a la frecuencia deseada desde los micrófonos elegidos y transfiere la información a través de datos con formato I2S con longitud de palabra definidas previamente. El prototipo está preparado para que por pantalla se introduzca la frecuencia de muestreo entre las seleccionables, la longitud de palabra de los datos I2S y el tipo de micrófono a usar para adquirir los datos. Hay que calcular primero conociendo el árbol de generación de reloj las frecuencias de muestreo que se han introducido por pantalla. A través de él se permite la posibilidad de programar el CODEC para tener cualquier frecuencia de muestreo entre 8 y 96 kHz. Los elementos que componen el árbol de generación de los relojes son un PLL que consta de dos multiplicadores, un divisor programable y un constante.

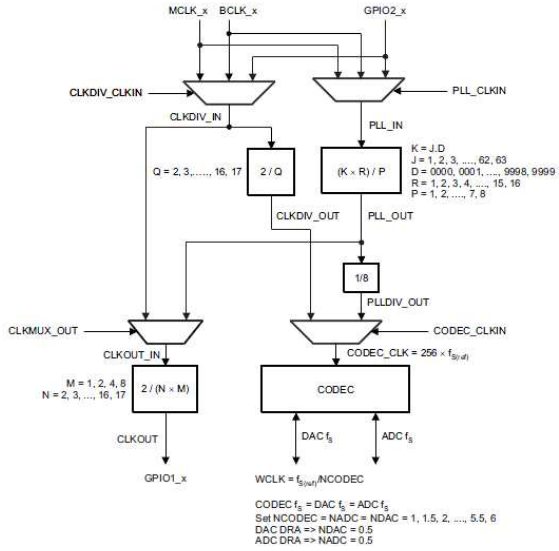


Figura 47. Generador de relojes de audio. Imagen extraída de [Texas Instruments, "*TLV320AIC34EVM Datasheet*", SLAS538A].

Para la generación de la frecuencia de muestreo se puede utilizar el PLL, o elegir una señal de entrada propicia que genere a través de un divisor (factor Q) la señal deseada.

Se va a tener como fuente de reloj el puerto MCLK que está conectado a un oscilador que se encuentra en la tarjeta de evaluación USB-MODEVM. A partir de esta señal se van a obtener las frecuencias de muestreo mediante el PLL o los divisores.

Para el cálculo de las frecuencias de muestreo se tienen las siguientes ecuaciones:

- Si PLL apagado:

$$f_{s(ref)} = CLKDIV\_IN/(128 \cdot Q),$$

donde *CLKDIV_IN* puede será *MCLK_x*,.

- Si PLL habilitado:

$$f_{s(ref)} = (PLLCLK\_IN \cdot K \cdot R)/(2048 \cdot P),$$

donde P puede ser 1, 2, 3,..., 8, *R* puede ser 1, 2,..., 16, K es un número decimal definido a través de la relación entre *J* y *D*: *J,D*. J puede ser 1, 2, 3, ..., 63 y *D* puede ser 0000, 0001, 0002, 0003, ..., 9998, 9999. *PLLCLK_IN* puede será *MCLK_x*.

Los parámetros K, R y P se calcularán en función de cada frecuencia de muestreo.

El sistema configura los pines de entrada, que están conectados a los micrófonos de diferente forma en función del tipo de micrófonos. Los registros a los que se hacen referencia en esta rutina de configuración se encuentran plasmados en el anexo 7.2. Se enuncia el flujo de configuración:

- Micrófonos digitales:

- Se definen el registro 98 para que el pin GPIO1 funcione como puerto para el reloj modulador del micrófono digital. El registro 99 se define para que el pin GPIO2 funcione como puerto de entrada de datos muestreando datos en el flanco de subida para un canal y en el flanco de bajada para el segundo canal.
- El registro 8 es de control del interface de audio serie y se configura para que las señales BCLK y WCLK sean de entrada. También se define la tasa de sobremuestreo debe ser de 64. Esta tasa se ha elegido entre las tres opciones (32, 64 y 128), dado que si se aplica el factor 128, cuando se quieren tener datos a 48 kHz y en los micrófonos digitales se sobremuestrea con 48 kHz por 128, se sale de la escala permitida por el micrófono.
- El registro 25 es el de control del MICBIAS y se define para deshabilitar esta opción ya que no se hará uso de los micrófonos analógicos.
- Se deshabilita en los registros 15 y 16 la ganancia de entrada de los ADC izquierdo y derecho para no tener señal de entrada procedente de micrófonos analógicos.

➢ Micrófonos analógicos:
- Se habilitan los pines MIC3L y MIC3R para conectarlos con los ADC izquierdo y derecho respectivamente en los registros 17 y 18.
- En el registro 8 se define que las señales WCLK y BCLK son de entrada. Se deshabilitan las funcionalidades de los micrófonos digitales.
- En el registro 25 se habilita el MICBIAS para alimentar los micrófonos analógicos.
- registros 15 y 16 se programa la ganancia de entrada deseada y se habilita la sonoridad ("*unmute*"). Se escoge como ganancia de entrada el valor de 5 dB.

➢ Frecuencias de muestreo: 44,1 kHz
- Se define en el registro 7 la configuración del camino de datos del códec. Se establece 44,1 kHz de frecuencia de referencia, se vincula el ADC izquierdo con el camino de datos del canal izquierdo y el ADC derecho con el camino de datos derecho.
- Se configura el registro 2, que es la definición de la frecuencia de muestro en función de la frecuencia de referencia. Se definen los divisores para generar la frecuencia deseada. Se establece que los divisores tienen el valor 1.

➢ Frecuencias de muestreo: 22,05 kHz
- Se define en el registro 7 como en el caso de 44, 1 kHz.

- Se configura el registro 2 definiendo los divisores con el valor 2 para que desde la frecuencia de referencia se genere 22,05 kHz de frecuencia de muestreo.

➢ Frecuencias de muestreo: 48 kHz, 16 kHz y 8 kHz

Para las frecuencias de muestreo 48, 16 y 8 kHz hay que configurar el árbol de reloj del CODEC. Para las otras dos frecuencias no es necesario dado que la señal de entrada de reloj maestro del CODEC está conectada a un oscilador que aporta una frecuencia de 11,2896 MHz.

Cuando el PLL no está encendido la frecuencia de referencia se calcula de la siguiente manera:

$$Frecuencia\ de\ referencia = CLKDIV\_IN/(128 \cdot Q)$$

Teniendo en cuenta que $CLKDIV\_IN$ = $MCLK$ = 11,2896 MHz y que Q = 2, se obtiene una frecuencia de referencia de 44,1kHz.

Sin embargo para los otros tres valores de frecuencias de muestreo hay que transformar dicha frecuencia para poder obtener una de referencia de 48 kHz mediante el uso del PLL del CODEC.

Cuando se habilita el PLL la frecuencia de referencia se calcula de la siguiente manera:

$$Frecuencia\ de\ referencia = (PLLCLK\_IN \cdot K \cdot R)/(2048 \cdot P)$$

Se ha calculado que para las tres frecuencias de muestro, solo es necesario una configuración del PLL, para la de la frecuencia de referencia que será 48 kHz. A través de los divisores del ADC y DAC se pueden obtener las frecuencias de muestreo de 16 y 8 kHz. Para obtener los 48 kHz se obtienen los valores J =8, D = 6500, R =1 y P = 1. Se configuran a través de los siguientes registros:

- Se define el registro 4 con el valor de J.
- Se define el registro 5 con los 8 bits más significativos del valor D.
- Se define el registro 6 con los 6 bits menos significativos del valor D.
- El registro 11 se define con el valor de R.
- En el registro 7 se define la opción de los 48 kHz como frecuencia de referencia se vincula el ADC izquierdo con el camino de datos del canal izquierdo y el ADC derecho con el camino de datos derecho.
- Se configura el registro 3 para que el valor de Q sea igual a 2 y se pone el valor de P obtenido. Además este registro enciende el PLL con los valores definidos.
- En el registro 2 se definen los divisores para la obtención de la frecuencia de muestreo. En el caso de tener 48 kHz, los divisores serán 1, en el caso de 16 kHz los divisores tendrán en valor de 3 y en el caso de 8 kHz el valor de 6.

- Definición de la procedencia del reloj maestro en el registro 102. El valor será el mismo para todos los casos. Se tiene como entrada del reloj maestro la señal procedente del oscilador de la tarjeta MODEVM.
- Longitud de palabras de datos I2S
    - EN el registro 9 se definen la longitud de las palabras de la transmisión I2S. También se define en este registro el modo de transmisión que es el I2S.
- Configuración ADC
    - A través del registro 19 se enciende el ADC izquierdo y se deshabilita la conexión entre el ADC y la línea de entrada analógica LINE1L.
    - Se define el registro 22 para encender el ADC derecho y deshabilitar la entrada analógica LINE1R.
- Configuración DAC
    - Se define el registro 37 para encender os DAC izquierdo y derecho.
    - Se define el registro 43 para configurar el nivel de amplificación de salida de los datos del DAC izquierdo. Se elige el valor máximo de 0dB.
    - Se define el registro 44 para configurar el nivel de amplificación de salida de los datos del DAC derecho. Se elige el valor máximo de 0dB.
- Configuración salida de audio
    - Se define el registro 47 para conectar la salida del DAC izquierdo con el canal de salida de audio HPLOUT con un nivel de ganancia de 0 dB.
    - Se define el registro 64 para conectar la salida del DAC derecho con el canal de salida de audio HPROUT con un nivel de ganancia de 0 dB.
    - Se define el registro 51 con el volumen de salida del canal HPLOUT que se configurará con un valor de 2 dB para escuchar el audio del canal izquierdo decodificado.
    - Se define el registro 65 con el volumen de salida del canal HPROUT que se configurará con un valor de 2 dB para escuchar el audio del canal derecho decodificado.
- Etapa de comprobación
    - Después de cada escritura se realiza una lectura del registro comprobando que el valor que contiene es correcto. Además se hace lectura de dos registros.
    - Con el registro 36 se puede comprobar si los ADC están encendidos además de ver si las ganancias aplicadas a los ADC y DAC son las programadas.
    - Con el registro 94 se comprueba que los DAC, el HPLOUT y el HPROUT están encendidos. Los valores de ambos registros se imprimen en pantalla para mostrar al usuario si ha existido algún error y es necesario reiniciar el sistema.

4.3.6 Configuración AIC3204

El códec TLV320AIC3204 se encuentra integrado en la tarjeta del TMS320VC5505 eZdsp y se utiliza en el sistema como salida de dos canales de audio ya que la tarjeta dispone de un conector tipo Jack de 3,5 mm para poder escuchar la decodificación del audio. El códec se configura mediante la escritura de registros a través del canal I2C. Como se ha definido en el punto anterior, la dirección I2C del AIC3204 será 0x18. Este códec tiene canal I2S0 dedicado para la transferencia de datos de audio con el DSP TMS320VC5505. A través de dicho canal se enviará la información decodificada en el DSP para que se pueda escuchar a través de unos altavoces y comprobar tiempo real que se ha producido correctamente tanto la codificación como decodificación del audio. El códec se configura a través de una rutina de escritura de registros como el AIC34. Como este códec se quiere únicamente para decodificar la señal de dos canales de salida, la configuración se centra en generar las frecuencias de funcionamiento del DAC en función de la frecuencia de muestreo seleccionada y en conectar la salida del DAC con el puerto de salida Jack a través de las salidas de potencia altas llamadas HPL y HPR.

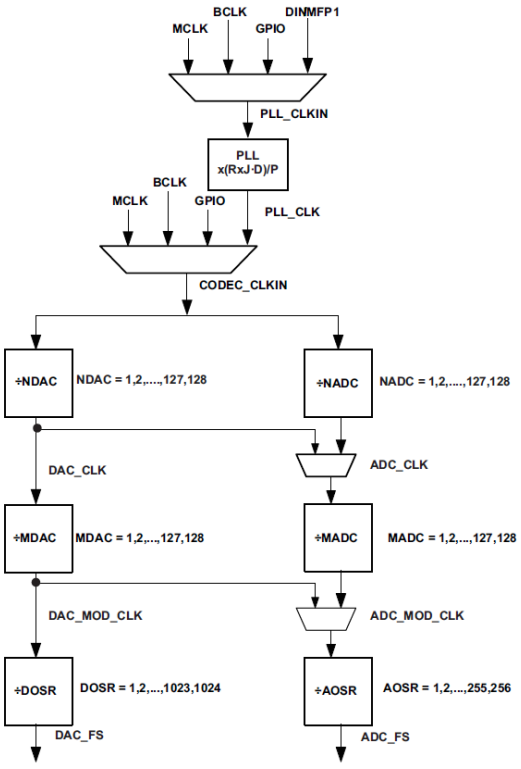


Figura 48. Árbol de reloj del TLV320AIC3204. Imagen extraída de [Texas Instruments,"*TLV320AIC3204 Ultra Low Power Stereo Audio Codec*", SLOS602A].

La tarjeta dispone de un oscilador que funcione a 12 MHz y está conectada con el puerto de entrada MCLK. A partir de esta señal de reloj se generan las frecuencias de muestreo del DAC a través de las siguientes ecuaciones:

$$DAC\_FS = {CODEC\_CLKIN}/{(NDAC \cdot MDAC \cdot DOSR)}$$

$$DAC\_MOD\_CLK = {CODEC\_CLKIN}/{(NDAC \cdot MDAC)}$$

$$PLL\_CLK = {PLL\_CLKIN \cdot R \cdot J, D}/{P}$$

La señal PLL_CLKIN será la misma que el MCLK, es decir tendrá el valor de 12 MHz. Para la programación del códec se tiene en primer lugar dos casos de generación de la señal de muestreo del DAC. El primero es para las frecuencias de muestreo de 44,1kHz y de 22,05 kHz. Se define el valor de D como 560, el valor NDAC como 5 y el valor de DOSR igual a 128. Estos valores se definen en los registros 7, 8, 11, 13 y 14. Para el caso de 44,1 kHz de frecuencia de muestreo, se define el valor de MDAC igual a 6 en el registro 12. Para el caso de 22,05 kHz, MDAC tendrá el valor 6.
El segundo caso es para las frecuencias de muestreo de 48, 16 y 8 kHz. Se define el valor de D como 1680, el valor MDAC como 7 y el valor de DOSR igual a 128. Para el caso de 48 kHz, NDAC se define igual a 2, para generar los 16 kHz NDAC es 6 y para los 8 kHz NDAC tiene el valor 12.

A través del registro 27 se define la longitud de palabras de datos I2S que será acorde al seleccionado al comienzo de la configuración del sistema.

El resto del proceso de configuración del códec es común para todos los casos. El registro número cuatro se define para que la señal CODEC_CLK sea la misma que el PLL_CLK y que la señal de PLL_CLKIN provenga del puerto de entrada MCLK. El registro número 5 se configura para que se encienda el PLL y los valores de P y R sean igual a 1. El registro seis contiene el valor J que en todos los casos es igual a 7.
Por último se genera la ruta de datos de salida desde el DAC. A través del registro 12 de la página 1 del mapa de registros se conecta la salida del DAC izquierdo con la salida de potencia alta izquierda HPL y a través del registro 13 de la página 1 del mapa de registros se conecta la salida del DAC derecho con la salida de potencia derecha HPR. Los registros 16 y 17 se definen para que el HPL y activen el sonido. La ganancia que se les aplica es de 18 dB. En el registro 9 se encienden las salida de sonido HPL y HPR. Los registros 64 y 65 son de configuración de los DAC. Primero se activa el sonido y a continuación se le aplica una ganancia de 0 dB en la entrada de los DAC. Por último a través del registro 63 se encienden los DAC. Los registros que se enuncian en este apartado se encuentran definidos en el anexo 7.3.

4.4 Implementación ADPCM

Para la implementación de la codificación ADPCM se parte de un código abierto existente desarrollado para el programa *SOX*. Por las exigencias del proyecto, se ha pedido explícitamente la implementación del Microsoft ADPCM, también denominado MS ADPCM, que es uno de los tres estándares comerciales del codificador ADPMC. La codificación del MS ADPCM es de tipo "Forward Gain-Adaptive" con coeficientes y ajuste de ganancia establecidos.
El sonido se codifica como una sucesión de paquetes de 4 bits. Cada paquete representa la diferencia entre la muestra actual y la siguiente muestra. La tasa de compresión que se obtiene es de 4 a 1, pasando de muestras de 16 bits a muestras de 4 bits.
Se han obtenido las librearías de código abierto de Microsoft ADPCM del programa *Sound Exchange* (*SOX*). El código está vinculado a una licencia de GNU (General Public LIcence) que permite y garantiza la libertad para compartir y modificar el código. Una vez adquirido el código y adaptado al entorno de desarrollo, se añaden comentarios sobre las modificaciones respecto a la librería original para que cualquier desarrollador que quiera utilizar la librería pueda libremente partir del código modificado conociendo las variaciones respecto al original.

4.4.1 Formato ADPCM

El formato de los datos MS ADPCM es una categoría de archivos WAV. A su vez el formato de archivo WAV es un subconjunto de la especificación de datos multimedia RIFF (Resource interchange file format) de Microsoft. Los archivos RIFF están formados en su totalidad por "bloques", cada uno de ellos con su propia cabecera y bytes de datos. Antes de comenzar a codificar las muestras se genera una cabecera que contiene los datos característicos de la codificación. La cabecera se divide en varios bloques que contienen información sobre la frecuencia de muestreo, el número de segmentos codificados, etc.
Todos los bloques tienen la siguiente estructura:

- 4 bytes: un identificador ASCII del bloque, por ejemplo "fmt " o "data".
- 4 bytes: un entero sin signo de 32 bits, little-endian, con la longitud del bloque (sin contar este mismo campo y el identificador del bloque).
- Campo de longitud variable: la información del bloque, del tamaño especificado en el campo anterior.
- Un byte de relleno, si la longitud del bloque no es par.

Dos identificadores, "RIFF" y "LIST", introducen un bloque que puede a su vez contener subbloques (por ejemplo, los bloques "fmt" y "data" son en realidad

subbloques del bloque “RIFF”). Su información de bloque, tras el identificador y la longitud, tiene la siguiente composición:

- 4 bytes: un identificador ASCII para este bloque en particular, en el caso del bloque RIFF: para el archivo completo
- Resto: subbloques

La cabecera de los ficheros WAV sigue el estándar RIFF. Los primeros 12 bytes los componen la cabecera del bloque RIFF que contiene el identificador “RIFF, un tamaño de bloque igual al tamaño total del archivo menos los 8 bytes utilizados en la cabecera y cuatro bytes que contienen la palabra “WAVE”.

Descripción	Offset	Tamaño	Valor
ID de bloque	0	4	“RIFF”
Tamaño de boque	0x04	4	Tamaño de boque - 8
ID tipo de RIFF	0x08	4	“WAVE”

Tabla 26. Bloques de cabecera WAV.

Tras éste identificador vendrán los distintos bloques del fichero WAV que definen la forma de onda. El formato “WAVE” consiste en dos partes, la “fmt”, que describe el formato de los datos de audio y los datos, que se compone de información sobre el tamaño, número de muestras canales de los datos y los datos de audio en sí.
En la zona de definición del formato la cabecera tendrá toda la información acerca de la codificación ADPCM que se utiliza.

Descripción	Offset	Tamaño	Valor
ID de bloque	0	4	“fmt”
Fmtsize	0x04	4	16 + bytes de formato adicionales
Código de formato	0x08	2	0x02
Número de canales	0x0A	2	2
Frecuencia de muestreo	0x0C	4	*Frecuencia de muestreo*
Tasa de bytes por segundo	0x10	4	*AvgBytesPerSec*
Alineamiento de bloques	0x14	2	*BlockAlign*
Bits por muestra	0x16	2	4
Tamaño de parámetros extra	0x18	2	50
Muestras por bloque	0x1A	2	4
Número de coeficientes	0x1C	2	7
Coeficientes ADPCM	0x1E	14	Coeficientes MS ADPCM
ID de bloque	0x4A	4	“fact”
Tamaño del fact	0x50	4	4
Número de muestras escritas	0x54	4	

Tabla 27. Zona “fmt” del bloque de cabecera WAV.

La estructura de la cabecera es particular para un archivo WAV MS ADPCM. El bloque “fmt” tiene unos valores estándar comunes a todos los formatos de codificación y una zona para los parámetros adicionales que depende de dicha codificación En el caso de MS ADPCM tendremos que añadir los coeficientes de codificación.

Código	Descripción
0 (0x0000)	Desconocido

1 (0x0001)	PCM / sin compresión
2 (0x0002)	Microsoft ADPCM
6 (0x0006)	ITU G.711 a-law
7 (0x0007)	ITU G.711 μ-law
17 (0x0011)	IMA ADPCM
20 (0x0016)	ITU G.723 ADPCM (Yamaha)
49 (0x0031)	GSM 6.10
64 (0x0040)	ITU G.721 ADPCM
80 (0x0050)	MPEG
0 (0x0000)	Desconocido

Tabla 28. Códigos comunes de formato.

Los valores introducidos se obtienen de las siguientes reglas:

- "Fmtsize": siempre es el valor 16 más el tamaño de los parámetros adicionales que dependen del formato de codificación. En este caso al tener MS ADPCM tendremos como un tamaño adiciona que sumar 4 bytes del campo más el espacio que ocupan los 7 coeficientes MS ADPCM:

$$Fmtsize = 16 + Extrasize = 16 + 4 + (4 \cdot 7) = 50$$

- "Extrasize": es el tamaño que ocuparán los datos de tipo de codificación. En el caso de tener MS ADPCM se tienen cuatro bytes más 7 veces 4 bytes de los coeficientes:

$$Extratsize = 4 + (4 \cdot 7) = 32$$

- Número de canales: Especifica cuántas señales de audio separadas van codificadas en el bloque WAV. Un valor de "1" indica una señal "mono", "2" indica "estéreo", etc. En nuestro caso se definirá el número canales igual a 2.
- Frecuencia de muestreo: Indica la tasa de muestreo en muestras por segundo (Hercios). Los valores que van a ser posibles son 8 kHz, 16 kHz, 22.05 kHz, 44.1 kHz y 48 kHz.
- Tasa de bytes por segundo: Se calcula la tasa de bytes por segundo de la siguiente manera:

$$AvgBytesPerSec = \frac{BlockAlign \cdot Tasa\ de\ muestreo}{Muestras\ por\ bloque + 0{,}5}$$

- Alineamiento de bloques: parámetro dado por el MS ADPCM:

$$BlockAlign = \frac{Tasa\ de\ muestreo}{11008} \cdot Número\ de\ canales \cdot 256$$

El sistema dispone de varias opciones de tasas de muestreo. El número canales es constante. Por ello se muestra en una tabla los valores de BlockAlign en función de la tasa de muestreo.

Frecuencia de Muestreo	*BlockAlign*
48 kHz	2048
44,1 kHz	2048
22,05 kHz	1024
16 kHz	512
8 kHz	512

Tabla 29. Valores *BlockAlign*.

- Bits por muestra: es el número de bits por muestra en la codificación MS ADPCM, que son 4.
- Muestras por bloque: Se calcula en función del número de canales y del alineamiento de bloques:

$$SamplesPerBlock = \frac{(BlockAlign - (7 \cdot Número\ de\ canales)) \cdot 2}{Número\ de\ canales} + 2$$

- Número de coeficientes: es el número de coeficientes para el cálculo de las muestras MS ADCPM en codificación.
- Coeficientes MS ADPCM: son 7 parejas de coeficientes dados por la codificación ya establecidos:
  {256,0}, {512,-256}, {0, 0}, {192,64}, {240,0}, {460,-208}, {392,-232}
- Número de muestras: se calculan el número de muestras escritas multiplicando los boques escritos con de las muestras por bloque.

A partir de este punto hay que calcular la zona de datos. Es la última parte de la cabecera WAV. Consta de un campo con el valor “data”, un segundo campo con el valor del tamaño de los datos y por último se escriben en el fichero los datos codificados.

Descripción	Offset	Tamaño	Valor
ID de bloque	0	4	“data”
Longitud de datos	0x04	4	Tamaño de boque - 8
Bloque de datos	0x08	-	-

Tabla 30. Zona “data” del bloque de cabecera WAV.

Una vez que se tiene la cabecera WAV lista se comienza la codificación MS ADPCM. Como se ha expuesto anteriormente los formatos WAV se basan en datos por bloques. MS ADPCM codifica el audio en bloques. Para ello hay que acumular en un buffer de entrada por cada canal I2S las muestras procedentes del CODEC AIC34. El número de muestras a acumular viene determinado por el parámetro *SamplesPerBlock* (muestras por bloque).

Frecuencia de Muestreo	*BlockAlign*	*SamplesPerBlock*
48 kHz	2048	2036
44,1 kHz	2048	2036
22,05 kHz	1024	1012
16 kHz	512	500
8 kHz	512	500

Tabla 31. Valores *SamplesPerBlock*.

El parámetro *SamplesPerBlock* indica el número de muestras de un canal en un bloque para codificar. *BlockAlign* indica el tamaño del bloque codificado. Se observa que es ligeramente superior debido a que para cada bloque tenemos una pequeña cabecera de 14 bytes. El tamaño en bytes es menor ya que el tamaño de las muestras en los bloques de entrada es de 16 bits y en el de salida es de 4 bits.
La estructura de un bloque codificado consiste en tener una cabecera que contiene el índice de los coeficientes escogidos para el bloque, el valor de paso y las dos primeras muestras del bloque en PCM. Con estos datos, los coeficientes, que se encuentran en la cabecera WAV y los valores de ajuste de ganancia se puede decodificar cualquier bloque a PCM.

Índice de la cabecera	Valor de la cabecera
0	Índice de coeficientes MS ADPCM del canal 0
1	Índice de coeficientes MS ADPCM del canal 1
2	LSB del valor de paso del canal 0
3	MSB del valor de paso del canal 0
4	LSB del valor de paso del canal 1
5	MSB del valor de paso del canal 1
6	LSB de la segunda muestra del bloque del canal 0 sin codificar
7	MSB de la segunda muestra del bloque del canal 0 sin codificar
8	LSB de la segunda muestra del bloque del canal 1 sin codificar
9	MSB de la segunda muestra del bloque del canal 1 sin codificar
10	LSB de la primera muestra del bloque del canal 0 sin codificar
11	MSB de la primera muestra del bloque del canal 0 sin codificar
12	LSB de la primera muestra del bloque del canal 0 sin codificar
13	MSB de la primera muestra del bloque del canal 0 sin codificar

Tabla 32. Cabecera de un bloque codificado.

Cada bloque contiene datos de los canales izquierdo y derecho. Los datos están entrelazados, es decir, que en la primera posición se encuentra un dato del canal izquierdo y en segunda posición el primer dato del canal derecho y así sucesivamente. Esto ocurre tanto para los bloques codificados como para los bloques con datos PCM.

4.4.2 Implementación ADPCM

La codificación MS ADPCM se basa en la codificación de bloques de tamaño fijo para muestras obtenidas con una frecuencia de muestreo constante de manera independiente.
La codificación siempre tiene muestras de entrada de 16 bits y 4 bits de salida. Esta compresión implica pérdidas imperceptibles para el oído humano en señales de audio.
La implementación del código de codificación MS ADPCM se estructura en varias fases. La primera de ellas es la adaptación de los coeficientes de codificación al bloque de muestras. Para ello primero hay que encontrar para cada bloque de datos entrante la pareja de coeficientes óptima. Se realiza una pre-codificación con una parte de las muestras y se calcula la diferencia de energía entre las muestras codificadas con cada coeficiente. El número de muestras elegido para la realización de la búsqueda de la pareja de coeficientes óptima es de un 20% del número de muestras de *SamplesPerBlock.* Se ha elegido este porcentaje después de realizar una serie de pruebas y comprobar que los resultados son óptimos. Luego se comparan las energías obtenidas y la resultante más baja será la que se aplique a todo el bloque de forma definitiva. El coeficiente obtenido se escribe en las primeras posiciones de la cabecera del boque codificado.
Con esto se adapta el nivel energético del bloque a la codificación. El rango dinámico es reducido y la codificación resulta óptima. La búsqueda del valor adecuado se realiza de manera independiente para todos los canales.
Esta parte es la de adaptación de la ganancia al bloque de forma anticipada.
La segunda parte es la codificación del bloque completo con el coeficiente definitivo. El algoritmo calcula primero las muestras a través de la predicción lineal:
1º predicción lineal con muestras actuales
2º calculo de diferencia entre predicción lineal y la muestra actual
3º se le suma a la diferencia el valor del paso multiplicado por 8 y dividido entre 2.
4º cálculo del valor de paso de las muestras
5º Se guarda como valor de salida el valor del paso entre la diferencia de la muestra actual y la predicción lineal.

4.4.3 Adaptación al sistema en tiempo real

Se obtuvo un código MS ADPCM del paquete de software *SOX* ("*Sound Excahnge*"). Dicho código ha sido adaptado al entorno del DSP. El programa original está diseñado para funcionar en un sistema Host, como por ejemplo un ordenador que funcione con un procesador de propósito general y sin restricciones temporales. El entorno en el cual debe funcionar, el DSP TMS320VC5505 tiene limitaciones en

memoria, capacidad de procesamiento y la premisa del funcionamiento a tiempo real de 4 canales en paralelo.
El código obtenido desde *SOX* está diseñado para funcionar en un PC que tenga un procesador de varios GHz de rendimiento con una memoria amplia y sin limitación temporal de funcionamiento. Los tipos utilizados de datos están diseñados para un entorno de procesador de palabras de 32 bits.
Al adaptar el código al entorno embebido han aparecido una serie de problemas por las limitaciones y diferencias de condiciones.

1. Mapeo de tipos de datos:

La primera diferencia reside en el cambio del tipo de variables. A través de la tabla 28 "Tipos de datos" se han cambiado los tipos de datos para conseguir que el código realice la codificación de manera correcta.
El primer problema por consecuencia de este cambio reside en el tiempo de ejecución de las instrucciones. La mayoría de las variables están definidas en el entorno de procedencia como enteros de 32 bits. El procesador de DSP trabaja con palabras de 16 bits. Hay que buscar las variables que pueden estar definidas como variables de enteros de 16 bits. El resto de las variables deben quedar definidas en el DSP como tipo "double". Esto implica que cada instrucción con ese tipo de datos necesita dos ciclos de ejecución. Se prolonga de esta manera el tiempo de ejecución del codificador que ha de funcionar a tiempo real.

2. Definición de memoria:

Como primer elemento de ayuda se pueden definir las zonas de memoria donde se quieren establecer las variables y los buffers. En el punto "4.2.4 Gestión de la memoria" se ha habla sobre la cantidad de memoria disponible y la capacidad de poder distribuirla entre las secciones lógicas del sistema. Es necesario reservar memoria para los buffers de entrada y de salida, los buffers con bloques codificados y para las variables.
El sistema final funcionará para 4 canales en paralelo. La limitación del número de canales a codificar proviene del número de interfaces I2S existentes. Solo se dispone de un CODEC AIC34 que tiene 2 bloques que permiten 4 canales de datos simultáneamente. Por ello, como buffers de entrada es necesario tener 2 buffers de 1,5 veces el tamaño del parámetro "*SamplesPerBlock*".
El sistema funciona capturando muestras de manera continua y trasmitiéndolas al DSP. Una vez que el buffer de entrada de un canal se completa, es decir que se tiene "*SamplesPerBlock*" número de muestras nuevas, éstas se copian a un segundo buffer que es utilizado por el algoritmo de codificación MS ADPCM. Dado que es necesario el ir acumulando muestras debido a la codificación por bloques hay que reservar más espacio para los buffer de entrada. Se ha optado por añadir un 50% más de tamaño al los buffer de entrada para que durante el proceso de transferencia no se pierdan muestras. Así quedan los dos buffer de entrada con un tamaño de 3 veces el tamaño "*SamplesPerBlock*". Las muestras se van acumulando en los buffers de entrada cuando el bus I2S advierte de un datos entrante y

mediante una interrupción se para el proceso activo, es decir la codificación o decodificación para guardar el dato.
Una vez obtenidas las muestras se transfieren los bloques de datos al codificador. Aquí es necesario realizar una transferencia de las muestras a unos buffer con el tamaño exacto de dos veces "*SamplesPerBlock*". Se necesita un buffer de entrada de dicho tamaño por cada dos canales y un buffer de salida de tamaño dos veces "*BlockAlign*" por cada dos canales.
Una vez terminado el proceso de codificación, el bloque de salida, se utiliza para realizar la decodificación y posteriormente reproducir los datos y comprobar mediante su escucha que son correctos.
Para el proceso de salida se son necesarios 2 buffers circulares de tamaño 2 *SamplesPerBlock* por canal para los datos decodificados que pasarán. Se utilizan buffers circulares para los datos de salida para el caso en el que los datos decodificados están listos y no se han terminado de transmitir los datos a los códecs del bloque anterior para su reproducción, éstos no se pierdan escribiendo el bloque justo detrás para que exista una continuidad en la reproducción.

Frecuencia de muestreo	*SamplesPerBlock*	*BlockAlign*	Buffer entrada	Buffer salida	Datos Codificados	Datos Decodificados	Memoria Total
48 kHz	2048	2036	12kbytes	8kbytes	4kbytes	8kbytes	56 kbytes
44,1 kHz	2048	2036	12kbytes	8kbytes	4kbytes	8kbytes	56 kbytes
22,05 kHz	1024	1012	6 kbytes	4kbytes	2kbytes	4kbytes	28 kbytes
16 kHz	512	500	3 kbytes	2kbytes	1kbytes	2kbytes	16 kbytes
8 kHz	512	500	3 kbytes	2kbytes	1kbytes	2kbytes	16 kbytes

Tabla 33. Memoria de buffers.

- Cálculo del buffer de entrada: 2048 muestras/canal *1,5 extra *2 canales*16bits/8=12kbytes
  Para cada línea es *SamplesPerBlock* muestras/canal *1,5 extra *2 canales*16bits/8 = *SamplesPerBlock* muestras/canal * 6 (entre 1024 por kbytes)
- Cálculo del buffer de salida: 2048 muestras/canal *2 canales*16bits/8=8kbytes
  Para cada línea es *SamplesPerBlock* muestras/canal *2 canales*16bits/8 = *SamplesPerBlock* muestras/canal * 4 (entre 1024 por kbytes)
- Memoria total: 2 Buffers de entrada (para 4 canales)+2 Buffers de salida + 2 Buffers de datos codificados +2 Buffers con datos

Teniendo en cuenta que se dispone de 64 kbytes – 192 bytes de memoria tipo DARAM y 128 kbytes de tipo SARAM. Es necesario alojar el *stack* y el *systack* en la zona de memoria DARAM con lo cual hay que elegir las zonas de memoria SARAM0 o SARAM1.
Esto tiene como consecuencia el tiempo de ejecución del codificador. El código original no está diseñado para funcionar a tiempo real. El problema reside en que el código MS ADPCM en el sistema original está diseñado para leer de un fichero acabado con todos los datos finales preparados. La idea del sistema es capturar los

bloques a tiempo e ir codificando a tiempo real sin conocer a priori la cantidad de datos que se van a obtener. El tiempo necesario para capturar un bloque completo es de SamplesPerBlock muestras por Frecuencia de muestreo.

Frecuencia de muestreo	*SamplesPerBlock*	Tiempo de captura de un bloque
48 kHz	2048	42,67 ms
44,1 kHz	2048	46,44 ms
22,05 kHz	1024	42,67 ms
16 kHz	512	32 ms
8 kHz	512	64 ms

Tabla 34. Tiempos de captura de bloques completos.

Esto implica que un bloque ha de ser codificado y decodificado en menos de 32 ms en el peor de los casos para poder ser a tiempo real. La primera ejecución del código adaptado en cuanto a tipo de variables y memoria ejecutó 17 veces más lento que el tiempo de captura de un bloque. Se hizo un análisis más exhaustivo en cuanto a los valores que podían tomar las variables de las funciones de codificación y decodificación y las instrucciones. Había que adaptar las expresiones de las funciones para poder lograr un grado mayor de optimización de instrucciones acorde a la arquitectura del DSP 5505. Se realizaron cambios de instrucciones para lograr un menor tiempo de ejecución.

3. Herramienta de análisis de número de instrucciones:

Una vez habiendo logrado reducir el tiempo de ejecución por debajo de los tiempos de captura de un bloque había que conseguir reducirlos un 50% más para poder ejecutar los dos codificadores y decodificadores en paralelo para los 4 canales. Estas reducciones de tiempo han consistido en reformular el código del algoritmo y adaptarlo a la arquitectura del DSP. Se han modificado las instrucciones para reducir el tiempo de ejecución de las mismas de forma individual. Esto se ha realizado mediante una herramienta de análisis que tiene el programa de desarrollo CCStudio integrado. Esta herramienta permite ejecutar código contando el número de instrucciones y tiempo que requiere. Mediante esta herramienta se han analizado las diferentes formas de ejecutar el algoritmo y se buscado siempre aquella que reduzca el tiempo de ejecución.

La primera comprobación del funcionamiento de la adaptación del código obtenido se ha hecho ejecutando por una parte el código MS ADPCM en un PC y guardando los bloques de codificación de una fuente de audio. Se ha aplicado la misma fuente de audio en el sistema, se ha codificado dicha fuente. Para cada bloque generado se ha parado el sistema y se ha comprobado que las salidas de datos de ambas ejecuciones coinciden. De esta forma se ha comprobado en primera instancia sin tener en cuenta el factor de codificación a tiempo real que el sistema está realizando la codificación de forma correcta. A partir de este punto se han utilizado

las técnicas descritas para reducir el tiempo de ejecución y conseguir que el sistema funcione a tiempo real para cuatro canales en paralelo.

Finalmente se cumplen los requisitos establecido por el cliente ya que los tiempos de ejecución máximos están por debajo de los tiempos de captura de un bloque y permite mantener dos codificaciones y decodificaciones simultáneas en tiempo real en el sistema embebido. Mediante la escucha del audio decodificado se comprueba que los datos que entran son muy similares al audio humano en comparación con los entrantes.

5.CONCLUSIONES

El objetivo del proyecto es realizar un sistema de captura de audio multicanal que codifique a tiempo real. El resultado del proyecto es una prototipo que captura audio de cuatro canales en paralelo, codifica y decodifica los cuatro canales en tiempo real y se reproduce para comprobar el funcionamiento.
En primer lugar se ha recogido información sobre los diferentes elementos comerciales existentes para el desarrollo del sistema, se ha hecho un análisis de los elementos y se han seleccionado aquellos que mejor se adaptan a las necesidades del prototipo.
Se ha construido físicamente el sistema final a partir de los elementos adquiridos interconectándolos. Se han hecho pruebas del funcionamiento unitario de los dispositivos y después se han realizado pruebas de comunicación y de control de los elementos interconectados. A continuación se ha desarrollado un programa que controle la configuración del sistema final para todas las opciones de muestreo (48 kHz, 44,1 kHz, 22,05 kHz, 16 kHz y 8 kHz), transmisión de datos digitales I2S entre el códec y el DSP con diferentes longitudes de palabras (16, 20, 24 y 32 bits), la codificación a tiempo real y las opciones de tipos de micrófono para la adquisición de audio.
Se ha obtenido un software de codificación y decodificación MS ADPCM y se ha adaptado dicho código al sistema embebido. El resultado del proceso es un sistema que codifica y decodifique a tiempo real fuentes de audio adquiridas mediante micrófonos digitales y/o analógicos con ganancia integrada de 4 canales en paralelo de forma simultánea. El prototipo permite la comprobación del correcto funcionamiento del sistema ya que se puede escuchar el audio decodificado a través de altavoces. El sistema final codifica datos de audio a alta, media o baja calidad de adquisición reduciendo el tamaño de almacenamiento del audio con un ratio de cuatro a uno perdiendo muy poca calidad de sonido.

Existe una continuación del desarrollo de este prototipo que forma parte de un proyecto de mayor envergadura. Primero se va a realizar una interface entre el sistema y un equipo host para guardar a través de un canal de datos (USB) a tiempo real los datos codificados. Estos datos se podrán reproducir en el equipo host.
La última parte del proyecto consiste en integrar todos los elementos del prototipo en una única tarjeta de tamaño reducido que se podrá conectar a cualquier ordenador para poder realizar adquisiciones de audio de varios canales de forma independiente grabándolos a tiempo real en un sistema host reduciendo el espacio de almacenamiento cuatro veces.

6.Referencias

[1] Wai C. Chu, **"***Speech coding algorithms: foundation and evolution of standardized coders*", J. Wiley, 2003.

[2] Ogunfunmi Tokunbo, *"Principles of speech coding"*, 2010

[3] Knowles Acoustics, "*SPM0405HD4H-WB Topics Digital "Mini" SiSonicTM Microphone Specification - Halogen Free".*

[4] Burr-Brown Products from Texas Instruments, *"TLV320AIC34 FOUR-CHANNEL, LOW-POWER AUDIO CODEC FOR PORTABLE AUDIO/TELEPHONY Datasheet"*, SLAS538A, 2007.

[5] Texas Instruments, *"TLV320AIC34EVM-K User's Guide",* SLAU232, 2007.

[6] Texas Instruments, *"TMS320VC5505 DSP System User's Guide",* SPRUFP0, 2009.

[7] Texas Instruments, *"TMS320VC5505 Fixed-Point Digital Signal Processor",* SPRS503A, 2009.

[8] Spectrum Digital, *"TMS320VC5505 eZdsp USB Stick Technical Reference",* 512325-0001 Rev. A, 2009.

[9] Philips Semiconductors, *"I2S bus specification",* 1996.

[10] Philips Semiconductors, *"The I_2C -bus and how to use it",* 1995.

[11] Texas Instruments, *"TMS320VC5501/5502/5503/5507/5509/5510 DSP Multichannel Buffered Serial Port Reference Guide", 200*5.

[12] Texas Instruments*,"TMS320VC5505 DSP Inter-IC Sound (I2S) Bus User's Guide",* SPRUFP0, 2009

[13] Texas Instruments*,"TLV320AIC3204 Ultra Low Power Stereo Audio Codec***",** SLOS602A, 2008

[14] Doroteo T. Toledano, Joaquin Gonzalez-Rodriguez and Javier Ortega-Garcia, "*Voice Device*".

[15] Texas Instruments, "*TMS320C55x Optimizing C/C++ Compiler User's Guide*", SPRU281, 2003

[16] Texas Instruments, *"TLV320AIC34EVM Datasheet",* SLAS538A, 2007

[17] Texas Instruments, "TMS320VC5505 Fixed-Point Digital Signal Processor", SPRS503A, 2009

[18] Knowles Acoustics, "*SPM0408HE5H Amplified "Mini" SiSonicTM Microphone Specification with enhanced RF protection- Halogen Free" 2009,*

[19] Knowles Acoustics, *"SiSonic Design Guide –Application Note 16"*

[22] MAXIM, "*19-4573 Ultra-Low Power Stereo Audio Codec MAX9867,* 2010

[21] Wolfson Microelectronics, "*Ultra Low Power CODEC for Portable Audio Applications WM8903,* 2010

[22] Analog Devices, "*Mixed-Signal DSP Controller ADSP-21990*"

7.Anexos

7.1 Registros del TMS320VC5505 eZdsp

Dirección registros	Acrónimo	Descripción
1C00	EBSR	External Bus Selection Register
1C02	PCGCR1	Peripheral Clock Gating Control Register 1
1C03	PCGCR2	Peripheral Clock Gating Control Register 2
1C06	IODIR1	GPIO Direction Register 1
1C07	IODIR2	GPIO Direction Register 2
1C08	IOINDATA1	GPIO Data In Register 1
1C09	IOINDATA2	GPIO Data In Register 2
1C0A	IODATAOUT1	GPIO Data Out Register 1
1C0B	IODATAOUT2	GPIO Data Out Register 2
1C0C	IOINTEDG1	GPIO Interrupt Edge Trigger Enable Register 1
1C0D	IOINTEDG2	GPIO Interrupt Edge Trigger Enable Register 2
1C0E	IOINTEN1	GPIO Interrupt Enable Register 1
1C0F	IOINTEN2	GPIO Interrupt Enable Register 2
1C10	IOINTFLG1	GPIO Interrupt Flag Register 1
1C11	IOINTFLG2	GPIO Interrupt Flag Register 2
1C20	CGCR1	Clock Generator Control Register 1
1C21	CGCR2	Clock Generator Control Register 2
1C22	CGCR3	Clock Generator Control Register 3
1C23	CGCR4	Clock Generator Control Register 4
1C24	CCSSR	CLKOUT Control Source Select Register
1C1E	CCR1	Clock Configuration Register 1
1C1F	CCR2	Clock Configuration Register 2
1A00	ICOAR	I2C Own Address Register
1A04	ICIMR	I2C Interrupt Mask Register
1A08	ICSTR	I2C Interrupt Status Register
1A0C	ICCLKL	I2C Clock Low-Time Divider Register
1A10	ICCLKH	I2C Clock High-Time Divider Register
1A14	ICCNT	I2C Data Count Register
1A18	ICDRR	I2C Data Receive Register
1A1C	ICSAR	I2C Slave Address Register
1A20	ICDXR	I2C Data Transmit Register
1A24	ICMDR	I2C Mode Register
1A28	ICIVR	I2C Interrupt Vector Register
1A2C	ICEMDR	I2C Extended Mode Register
1A30	ICPSC	I2C Prescaler Register
1A34	ICPID1	I2C Peripheral Identification Register 1
1A38	ICPID2	I2C Peripheral Identification Register 2
2800	I2SSCTRL	I2S Serializer Control Register
2804	I2SSRATE	I2S Sample Rate Generator Register
2808	I2STXLT0	I2S Transmit Left Data 0 Register
2809	I2STXLT1	I2S Transmit Left Data 1 Register
280C	I2STXRT0	I2S Transmit Right Data 0 Register
280	I2STXRT1	I2S Transmit Right Data 1 Register
2810	I2SINTFL	I2S Interrupt Flag Register
2814	I2SINTMASK	I2S Interrupt Mask Register
2828	I2SRXLT0	I2S Receive Left Data 0 Register
2829	I2SRXLT1	I2S Receive Left Data 1 Register
282C	I2SRXRT0	I2S Receive Right Data 0 Register
282D	I2SRXRT1	I2S Receive Right Data 1 Register

Las direcciones de los registros están definidas en formato hexadecimal.

7.1.1 Descripción de los campos de los registros del DSP TMS320VC5505

- **EBSR:**

Bit	Field	Value	Description
15	Reserved	0	Reserved
14-12	PPMODE		Parallel port mode bits. These bits control the pin muxing on the parallel port.
		0	· Mode 0: (16-bit LCD controller. All 21 signals of the LCD controller module are routed to the 21 external signals of the parallel port.
		1h	· Mode 1: GPIO, SPI, UART, and I2S2: 6 GPIO signals, 7 signals of the SPI module, 4 signals of the UART module and 4 signals of the I2S2 module are routed to the 21 external signals of the parallel port.
		2h	· Mode 2: 8-bit LCD controller and GPIO: 8-bits of pixel data of the LCD controller module and 8 bits of GPIO are routed to 21 external signals on the parallel port.
		3h	· Mode 3: 3 8-bit LCD controller, SPI, and I2S3: 8-bits of pixel data of the LCD controller module, 4 signals of the SPI module, and 4 signals of the I2S3 module are routed to the 21 external signals on the parallel port.
		4h	· Mode 4: 4 8-bit LCD controller, I2S2, and UART: 8-bits of pixel data of the LCD controller module, 4 signals of the I2S2 module, and 4 signals of the UART module are routed to the 21 external signals on the parallel port.
		5h	· Mode 5: 8-bit LCD controller, UART, and SPI: 8-bits of pixel data of the LCD controller module, 4 signals of the U ART module, and 4 signals of the SPI module are routed to the 21 external signals on the parallel port.
		6h	· Mode 6: GPIO, SPI, and I2S3: 6 PIO, 7 signals of the SPI module, and two sets of 4 signals of the I2S2 and I2S3 module are routed to the 21 external signals of the parallel port. The mode selection of the parallel port is controlled.
		7h	· Reserved, do not use.
11-10	SP1MODE		Serial port 1 mode bits. These bits control the pin muxing on serial port 1.
		0	· Mode0: (MMC/SD) All 6 signals of the MMC/SD port are routed to the 6 external signals of the serial port.
		1h	· Mode 1: (I2S) All 4 signals of the I2S module and 2 GPIO signals are routed to the 6 external signals of the serial port.
		2h	· Mode 2: (GPIO) 6 GPIO signals are routed to the 6 external signals of the serial port.
		3h	· Mode 3: This mode selection of serial port 0 and serial port 1 is controlled through the
9-8	SP0MODE	0	Serial port 0 mode bits. These bits control the pin muxing on serial port 0. Mode 0
		1h	Mode 1
		2h	Mode 2
		3h	Mode 3
7-6	Reserved	0	Reserved
5	A20_MODE		This bit controls the pin multiplexing on EM_A[20].
		0	Pin behaves as EM_A[20].
		1	Pin behaves as GP[26].
4	A19_MODE		This bit controls the pin multiplexing on EM_A[19].
		0	Pin behaves as EM_A[19].
		1	Pin behaves as GP[25].
3	A18_MODE		This bit controls the pin multiplexing on EM_A[18].
		0	Pin behaves as EM_A[18].
		1	Pin behaves as GP[24].
2	A17_MODE		This bit controls the pin multiplexing on EM_A[17].
		0	Pin behaves as EM_A[17].
		1	Pin behaves as GP[23].
1	A16_MODE		This bit controls the pin multiplexing on EM_A[16].
		0	Pin behaves as EM_A[16].
		1	Pin behaves as GP[22].
0	A15_MODE		This bit controls the pin multiplexing on EM_A[15].
		0	Pin behaves as EM_A[15].
		1	Pin behaves as GP[21].

- **PRCR:**

Bit	Field	Value	Description

15-8	Reserved	0	Reserved. Always write 0 to these bits.
7	PG4_RST		Peripheral group 4 software reset bit. Drives the LCD, I2S2, I2S3, UART, and SPI reset signal.
		0	Reset deasserted, modules out of reset.
		1	Reset asserted, modules in reset.
6	Reserved	0	Reserved, always write 0 to this bit.
5	PG3_RST		Peripheral group 3 software reset bit. Drives the MMC/SD0, MMC/SD1, I2S0, and I2S1 reset signal.
		0	Reset deasserted, modules out of reset.
		1	Reset asserted, modules in reset.
4	DMA_RST		DMA software reset bit. Drives the reset signal to the DMA controllers.
		0	Reset deasserted, modules out of reset.
		1	Reset asserted, modules in reset.
3	USB_RST		USB software reset bit. Drives the USB reset signal.
		0	Reset deasserted, module out of reset.
		1	Reset asserted, module in reset.
2	Reserved	0	Reserved. Always write 0 to these bits.
1	PG1_RST		Peripheral group 1 software reset bit. Drives the EMIF and timers reset signal.
		0	Reset deasserted, modules out of reset.
		1	Reset asserted, modules in reset.
0	I2C_RST		I2C software reset bit. Drives the I2C reset signal.
		0	Reset deasserted, module out of reset.
		1	Reset asserted, module in reset.

- **PCGCR1:**

Bit	Field	Value	Description
15	SYSCLKDIS		System clock disable bit. This bit can be used to turn off the system clock. Setting the WAKEUP pin high enables the system clock. Since the WAKEUP pin is used to re-enable the system clock, the WAKEUP pin must be low to disable the system clock. Note: Disabling the system clock disables the clock to most parts of the DSP, including the CPU.
		0	System clock is active.
		1	System clock is disabled.
14	I2S2CG		I2S2 clock gate control bit. This bit is used to enable and disable the I2S2 peripheral clock.
		0	Peripheral clock is active.
		1	Peripheral clock is disabled.
13	TMR2CG		Timer 2 clock gate control bit. This bit is used to enable and disable the Timer 2 peripheral clock.
		0	Peripheral clock is active.
		1	Peripheral clock is disabled.
12	TMR1CG		Timer 1 clock gate control bit. This bit is used to enable and disable the Timer 1 peripheral clock.
		0	Peripheral clock is active.
		1	Peripheral clock is disabled.
11	EMIFCG		EMIF clock gate control bit. This bit is used to enable and disable the EMIF peripheral clock. NOTE: You must request permission before stopping the EMIF clock through the peripheral clock stop request/acknowledge register (CLKSTOP).
		0	Peripheral clock is active.
		1	Peripheral clock is disabled.
10	TMR0CG		Timer 0 clock gate control bit. This bit is used to enable and disable the Timer 0 peripheral clock.
		0	Peripheral clock is active.
		1	Peripheral clock is disabled.
9	I2S1CG		I2S1 clock gate control bit. This bit is used to enable and disable the I2S1 peripheral clock.
		0	Peripheral clock is active.
		1	Peripheral clock is disabled.
8	I2S0CG		I2S0 clock gate control bit. This bit is used to enable and disable the I2S0 peripheral clock.
		0	Peripheral clock is active.
		1	Peripheral clock is disabled.
7	MMCSD1CG		MMC/SD1 clock gate control bit. This bit is used to enable and disable the MMC/SD1 peripheral clock.
		0	Peripheral clock is active.
		1	Peripheral clock is disabled.
6	I2CCG		I2C clock gate control bit. This bit is used to enable and disable the I2C peripheral clock.
		0	Peripheral clock is active.
		1	Peripheral clock is disabled.
5	Reserved	0	Reserved, you must always write 1 to this bit.
4	MMCSD0CG		MMC/SD0 clock gate control bit. This bit is used to enable and disable the MMC/SD0 peripheral clock.
		0	Peripheral clock is active.
		1	Peripheral clock is disabled.
3	DMA0CG		DMA controller 0 clock gate control bit. This bit is used to enable and disable the
		0	peripheral clock the DMA controller 0.
		1	Peripheral clock is active.

			Peripheral clock is disabled.
2	UARTCG		UART clock gate control bit. This bit is used to enable and disable the UART peripheral clock. NOTE: You must request permission before stopping the UART clock through the peripheral clock stop request/acknowledge register (CLKSTOP).
		0	Peripheral clock is active.
		1	Peripheral clock is disabled.
1	SPICG		SPI clock gate control bit. This bit is used to enable and disable the SPI controller peripheral clock.
		0	Peripheral clock is active.
		1	Peripheral clock is disabled.
0	I2S3CG		I2S3 clock gate control bit. This bit is used to enable and disable the I2S3 peripheral clock.
		0	Peripheral clock is active.
		1	Peripheral clock is disabled.

- **PCGCR2:**

Bit	Field	Value	Description
15-7	Reserved	o	Reserved.
6	ANAREGCG		Analog registers clock gate control bit. This bit is used to enable and disable the clock to the registers that control the analog domain of the device, i.e. registers in the 7000h-70FFh I/O space address range. NOTE: When SARCG = 0, the clocks to the analog domain registers are enabled regardless of the ANAREGCG setting.
		0	Clock is active.
		1	Clock is disabled.
5	DMA3CG		DMA controller 3 clock gate control bit. This bit is used to enable and disable the DMA controller 3 peripheral clock.
		0	Peripheral clock is active.
		1	Peripheral clock is disabled.
4	DMA2CG		DMA controller 2 clock gate control bit. This bit is used to enable and disable the DMA controller 2 peripheral clock.
		0	Peripheral clock is active.
		1	Peripheral clock is disabled.
3	DMA1CG		DMA controller 1 clock gate control bit. This bit is used to enable and disable the DMA controller 1 peripheral clock.
		0	Peripheral clock is active.
		1	Peripheral clock is disabled.
2	USBCG		USB clock gate control bit. This bit is used to enable and disable the USB controller peripheral clock. NOTE: You must request permission before stopping the USB clock through the peripheral clock stop request/acknowledge register (CLKSTOP). Peripheral clock is active.
		0	
		1	Peripheral clock is disabled.
1	SARCG		SAR clock gate control bit. This bit is used to enable and disable the SAR peripheral clock. NOTE: When SARCG = 0, the clock to the analog domain registers is enabled regardless of the ANAREGCG setting.
		0	Peripheral clock is active.
		1	Peripheral clock is disabled.
0	LCDCG		LCD controller clock gate control bit. This bit is used to enable and disable the LCD controller peripheral clock.
		0	Peripheral clock is active.
		1	Peripheral clock is disabled.

- **CCR2:**

Bit	Field	Value	Description
15-6	Reserved	0	Reserved.
5-4	SYSCLKSRC		System clock source bits. These read-only bits reflect the source for the system clock.
		0	The system clock generator is in bypass mode; SYSCLK is driven by the RTC oscillator output.
		1h	The system clock generator is in PLL mode; the RTC oscillator output provides the input clock.
		2h	The system clock generator is in bypass mode; SYSCLK is driven by CLKIN.
		3h	The system clock generator is in PLL mode; the CLKIN pin provides the input clock.
3	TIMER0CLKSEL		Timer 0 clock source select bit. This bit specifies the input clock to Timer 0.
		0	Timer 0 uses SYSCLK as input clock.
		1	Timer 0 uses PLL output clock (PLLOUT) as input clock.
2	CLKSELSTAT		CLK_SEL pin status bit. This reflects the state of the CLK_SEL pin.
		0	CLK_SEL pin is low (RTC input clock selected).
		1	CLK_SEL pin is high (CLKIN input clock selected).
1	Reserved	0	Reserved.
0	SYSCLKSEL	0	System clock source select bit. This bit is used to select between the two main clocking modes for the DSP: bypass and PLL mode. In bypass mode, the DSP
		1	

			clock generator is bypassed and the system clock is set to either CLKIN or the RTC output (as determined by the CLKSEL pin). In PLL mode, the system clock is set to the output of the DSP clock generator. Logic in the system clock generator prevents switching from bypass mode to PLL mode if the PLL is powered down. Bypass mode is selected. PLL mode is selected.

➢ CGCR1:

Bit	Field	Value	Description
15	CLR_CNTL	 0 1	Clear control bit. This bit is used to clear digital flip flops within the clock generator. This bit must be cleared to 0 while changing the PLL settings. Digital flip-flops cleared. Digital flip-flops not cleared (normal operation).
14	Reserved	0	Reserved.
13	PLL_STANDBY	 0 1	PLL standby bit. This bit is used to place the PLL in standby mode. This bit is used for test purposes only and must always be cleared to 0. PLL is active (normal operation). PLL is in standby mode.
12	PLL_PWRDN	 0 1	PLL power down bit. This bit is used to power down the PLL when it is not being used. PLL is powered up. PLL is powered down.
11-10	Reserved	0	Reserved.
9-0	MH	0-3FFh	PLL multiplier value bits. These bits along with the ML bits in CGCR2 define the PLL multiplier value. Multiplier value = MH x 4 + ML + 4. For example, MH = 23Bh and ML = 1h means multiply by 2289 (8F1h).

➢ CGCR2:

Bit	Field	Value	Description
15	RDBYPASS	 0 1	Reference divider bypass bit. When this bit is set to 1 the reference divider in the DSP clock generator is bypassed. When this bit is set to 0, the reference clock to the DSP clock generator is divided by the reference divider. The RD bits specify the divider value. Use the reference divider. Bypass the reference divider.
14	Reserved	0	Reserved.
13-12	ML	0-3h	PLL multiplier value bits. These bits along with the MH bits in CGCR1 define the PLL multiplier value. Multiplier value = MH x 4 + ML + 4. For example, MH = 23Bh and ML = 1h means multiply by 2289 (8F1h).
11-0	RDRATIO	0-FFFh	Divider ratio bits for the reference divider. Divider value = RD_DIV + 4. For example, setting RD_DIV = 0 means divide by 4.

➢ CGCR3:

Bit	Field	Value	Description
15-2	INIT	0-FFFFh	Initialization bits for DSP clock generator. These bits must be initialized with 0x806 during PLL configuration.
3-0	Lock Status Monitor	0 1	PLL not locked PLL locked

➢ CGCR4:

Bit	Field	Value	Description
15-10	Reserved	0	Reserved.
9	OUTDIVEN	 0 1	Output divider and fixed /2 divider enable bit. This bit determines whether the output divider and the fixed /2 divider are enabled or bypassed. The output divider and the fixed /2 divider are bypassed. The output divider and the fixed /2 divider are enabled.
8	OUTDIV2BYPASS	 0 1	Output divider bypass bit. This bit determines whether the output divider is bypassed. The fixed /2 divider is used regardless of the setting of this bit. The output divider is bypassed. The output divider is enabled.
7-6	Reserved	0	Reserved.
5-0	ODRATIO	0-3Fh	Divider ratio bits for the output divider. Divider value = ODRATIO + 4. For example, setting ODRATIO = 2 means divide by 6. Note: The divider output is further divided by the fixed /2 divider.

➢ I2S Control Register:

Bit	Field	Value	Description

15	ENABLE	 0 1	Resets or enables the serializer transmission or reception. The I2S Bus (Data XFR, clock generation, and event gen logic) is disabled and held in reset state. I2S enabled.
14-13	Reserved	0	Reserved.
12	MONO	 0 1	Sets I2S into mono or Stereo mode. Stereo mode. Mono mode. Valid only when bit 0, FRMT=1 (DSP Format).
11	LOOPBACK	 0 1	Routes data from transmit shift register back to receive shift register for internal digital loopback. Normal operation, no loopback. Digital Loopback mode enabled.
10	FSPOL	0 1	Inverts I2S frame-synchronization polarity. The following: FRMT (bit 0): Function 0: (I2S/LJ) Frame-synchronization pulse is low for left word and high for right word 1: (DSP) Frame-synchronization is pulsed high The following: FRMT (bit 0): Function 0: (I2S/LJ) Frame-synchronization pulse is high for left word and low for right word 1: (DSP)Frame-synchronization is pulsed low
9	CLKPOL	 0 1	Controls I2S clock polarity. Receive data is sampled on the rising edge and transmit data shifted on the falling edge of the bit clock. Receive data is sampled on the falling edge and transmit data shifted on the rising edge of the bit clock.
8	DATADLY	 0 1	Sets the I2S receive/transmit data delay. 1-bit data delay. 2-bit data delay.
7	PACK	 0 1	Enable data packing. Divides down the generation of interrupts so that data is packed into the 32-bit receive/transmit word registers for each channel (left/right). Data packing mode disabled. Data packing mode enabled. For more information about data packing, see Section 2.8.
6	SIGN_EXT	 0 1	Enable sign extension of words. No sign extension. Received data is sign extended. Transmit data is expected to be sign extended.
5-2	WDLNGTH	 0 1h 2h 3h 4h 5h 6h 7h 8h 9h-Fh	Choose serializer word length. 8-bit data word. 10-bit data word. 12-bit data word. 14-bit data word. 16-bit data word. 18-bit data word. 20-bit data word. 24-bit data word. 32-bit data word. Reserved.
1	MODE	 0 1	Sets the serializer in master or slave mode. Serializer is configured as a slave. I2Sn_CLK and I2Sn_FS pins are configured as inputs. The bit-clock and frame-synchronization signals are derived from an external source and are provided directly to the I2S synchronizer without being further divided. Serializer is configured as a master. I2Sn_CLK and I2Sn_FS pins are configured as outputs and driven by the clock generators. The bit-clock and frame-synchronization signals are derived from the internal CPU clock.
0	FRMT	 0 1	Sets the serializer data format. I2S/left-justified format. DSP format.

- **I2S Rate:**

Bit	Field	Value	Description
15-6	Reserved	0	Reserved.
5-3	FSDIV	 0 1h 2h 3h 4h 5h 6h 7h	Divider to generate I2S*n*_FS (frame-synchronization clock). The I2S*n*_CLK is divided down by the configured value to generate the frame-synchronization clock. (Has no effect when I2S is configured as slave device). Divide by 8 Divide by 16 Divide by 32 Divide by 64 Divide by 128 Divide by 256 Reserved Reserved
2-0	CLKDIV	 0 1h	Divider to generate I2S*n*_CLK (bit-clock). The system clock (or DSP clock) to the I2S is divided down by the configured value to generate the bit clock. (Has no effect when I2S is configured as slave device). Divide by 2. Divide by 4.

		2h 3h 4h 5h 6h 7h	Divide by 8. Divide by 16. Divide by 32. Divide by 64. Divide by 128. Divide by 256.

- **I2SINTFL:**

Bit	Field	Value	Description
15-6	Reserved	0	Reserved.
5	XMITSTFL	 0 1	Stereo data transmit flag. Used only when the MONO bit 12 in the I2SSCTRL register = 0 (Stereo mode). This bit is cleared on read. No pending stereo transmit interrupt. Stereo transmit interrupt pending. Write new data value to I2S Transmit Left and Right data 0 and 1 registers.
4	XMITMONFL	 0 1	Mono data transmit flag. Used only when the MONO bit 12 in the I2SSCTRL register = 1 (Mono mode). This bit is cleared on read. No pending mono transmit interrupt. Mono transmit interrupt pending. Write new data value to Transmit Left Data 0 and 1 registers.
3	RCVSTFL	 0 1	Stereo data receive flag. Used only when the MONO bit 12 in the I2SSCTRL register = 0 (Stereo mode). This bit is cleared on read. No pending stereo receive interrupt. Stereo receive interrupt pending. Read Receive Left and Right data 0 and 1 registers.
2	RCVMONFL	 0 1	Mono data receive flag. Used only when the MONO bit 12 in the I2SSCTRL register = 1 (Mono mode). This bit is cleared on read. No pending mono receive interrupt. Mono receive interrupt pending. Read Receive Left data 0 and 1 registers.
1	FERRFL	 0 1	Frame-synchronization error flag. This bit is cleared on read. No frame-synchronization errors. Frame-synchronization error(s) occurred.
0	OUERRFL	 0 1	Overrun or Underrun condition. This bit is cleared on read. No overrun/under-run errors. The data registers were not read from or written to before the receive/transmit buffer was overwritten.

- **I2SINTMASK:**

Bit	Field	Value	Description
15-6	Reserved	0	Reserved.
5	XMITST	 0 1	Enable stereo left/right transmit data interrupt. Used only when the MONO bit 12 in the I2SSCTRL register = 0 (Stereo mode). This bit is cleared on read. Disable stereo TX data interrupt. Enable stereo TX data interrupt.
4	XMITMON	 0 1	Enable mono left transmit data interrupt. Used only when the MONO bit 12 in the I2SSCTRL register = 1 (Mono mode). This bit is cleared on read. Disable mono TX data interrupt. Enable mono TX data interrupt.
3	RCVST	 0 1	Enable stereo left/right receive data interrupt. Used only when the MONO bit 12 in the I2SSCTRL register = 0 (Stereo mode). This bit is cleared on read. Disable stereo RX data interrupt. Enable stereo RX data interrupt.
2	RCVMON	 0 1	Enable mono left receive data interrupt. Used only when the MONO bit 12 in the I2SSCTRL register = 1 (Mono mode). This bit is cleared on read. Disable mono RX data interrupt. Enable mono RX data interrupt.
1	FERR	 0 1	Enable frame sync error. Disable frame-synchronization error interrupt. Enable frame-synchronization error interrupt.
0	OUERR	 0 1	Enable overrun or underrun condition. Disable overrun/underrun error interrupt. Enable overrun/underrun error interrupt.

- **ICOAR:**

Bit	Field	Value	Description
15-10	Reserved	0	These reserved bit locations are always read as zeros. A value written to this field has no effect.
9-0	OADDR	0-3FFh	Own slave address. Provides the slave address of the I2C. In 7-bit addressing mode (XA = 0 in ICMDR): bits 6-0 provide the 7-bit slave address of the I2C. Bits 9-7 are ignored. In 10-bit addressing mode (XA = 1 in ICMDR): bits 9-0 provide the 10-bit slave address of the I2C.

- **ICSTR:**

Bit	Field	Value	Description
15	Reserved	0	These reserved bit locations are always read as zeros. A value written to this field has no effect.
14	SDIR	0 1	Slave direction bit. In digital-loopback mode (DLB), the SDIR bit is cleared to I2C is acting as a master-transmitter/receiver or a slave-receiver. SDIR is cleared by one of the following events: • A STOP or a START condition. • SDIR is manually cleared. To clear this bit, write a 1 to it. I2C is acting as a slave-transmitter.
13	NACKSNT	0 1	No-acknowledgment sent bit. NACKSNT bit is used when the I2C is in the receiver mode. NACK is not sent. NACKSNT is cleared by one of the following events: • It is manually cleared. To clear this bit, write a 1 to it. • The I2C is reset (either when 0 is written to the IRS bit of ICMDR or when the processor is reset). NACK is sent. A no-acknowledge bit was sent during the acknowledge cycle on the I2C-bus.
12	BB	0 1	Bus busy bit. BB bit indicates whether the I2C-bus is busy or is free for another data transfer. In the master mode, BB is controlled by the software. Bus is free. BB is cleared by one of the following events: • The I2C receives or transmits a STOP bit (bus free). • BB is manually cleared. To clear this bit, write a 1 to it. • The I2C is reset (either when 0 is written to the IRS bit of ICMDR or when the processor is reset). Bus is busy. When the STT bit in ICMDR is set to 1, a restart condition is generated. BB is set by one of the following events: • The I2C has received or transmitted a START bit on the bus. • SCL is in a low state and the IRS bit in ICMDR is 0.
11	RSFULL	0 1	Receive shift register full bit. RSFULL indicates an overrun condition during reception. Overrun occurs when the receive shift register (ICRSR) is full with new data but the previous data has not been read from the data receive register (ICDRR). The new data will not be copied to ICDRR until the previous data is read. As new bits arrive from the SDA pin, they overwrite the bits in ICRSR. No overrun is detected. RSFULL is cleared by one of the following events: • ICDRR is read. • The I2C is reset (either when 0 is written to the IRS bit of ICMDR or when the processor is reset). Overrun is detected.
10	XSMT	0 1	Transmit shift register empty bit. XSMT indicates that the transmitter has experienced underflow. Underflow occurs when the transmit shift register (ICXSR) is empty but the data transmit register (ICDXR) has not been loaded since the last ICDXR-to-ICXSR transfer. The next ICDXR-to-ICXSR transfer will not occur until new data is in ICDXR. If new data is not transferred in time, the previous data may be re-transmitted on the SDA pin. Underflow is detected. No underflow is detected. XSMT is set by one of the following events: • Data is written to ICDXR. • The I2C is reset (either when 0 is written to the IRS bit of ICMDR or when the processor is reset).
9	AAS	0 1	Addressed-as-slave bit. The AAS bit has been cleared by a repeated START condition or by a STOP condition. AAS is set by one of the following events: • I2C has recognized its own slave address or an address of all zeros (general call). • The first data word has been received in the free data format (FDF = 1 in ICMDR).
8	AD0	0 1	Address 0 bit. AD0 has been cleared by a START or STOP condition. An address of all zeros (general call) is detected.
7-6	Reserved	0	These reserved bit locations are always read as zeros. A value written to this field has no effect.
5	SCD	0 1	Stop condition detected bit. SCD indicates when a STOP condition has been detected on the I2C bus. The STOP condition could be generated by the I2C or by another I2C device connected to the bus. No STOP condition has been detected. SCD is cleared by one of the following events: • By reading the INTCODE bits in ICIVR as 110b. • SCD is manually cleared. To clear this bit, write a 1 to it. A STOP condition has been detected.
4	ICXRDY	0 1	Transmit-data-ready interrupt flag bit. ICXRDY indicates that the data transmit register (ICDXR) is ready to accept new data because the previous data has been copied from ICDXR to the transmit shift register (ICXSR). The CPU can poll ICXRDY or use the XRDY interrupt request. ICDXR is not ready. ICXRDY is cleared by one of the following events: • Data is written to ICDXR. • ICXRDY is manually cleared. To clear this bit, write a 1 to it. ICDXR is ready. Data has been copied from ICDXR to ICXSR. ICXRDY is forced to 1 when the I2C is reset.

3	ICRRDY		Receive-data-ready interrupt flag bit. ICRRDY indicates that the data receive register (ICDRR) is ready to be read because data has been copied from the receive shift register (ICRSR) to ICDRR. The CPU can poll ICRRDY or use the RRDY interrupt request.
		0	ICDRR is not ready. ICRRDY is cleared by one of the following events: • ICDRR is read. • ICRRDY is manually cleared. To clear this bit, write a 1 to it. • The I2C is reset (either when 0 is written to the IRS bit of ICMDR or when the processor is reset).
		1	ICDRR is ready. Data has been copied from ICRSR to ICDRR.
2	ARDY		Register-access-ready interrupt flag bit (only applicable when the I2C is in the master mode). ARDY indicates that the I2C registers are ready to be accessed because the previously programmed address, data, and command values have been used. The CPU can poll ARDY or use the ARDY interrupt request.
		0	The registers are not ready to be accessed. ARDY is cleared by one of the following events: • The I2C starts using the current register contents. • ARDY is manually cleared. To clear this bit, write a 1 to it. • The I2C is reset (either when 0 is written to the IRS bit of ICMDR or when the processor is reset).
		1	The registers are ready to be accessed. This bit is set after the slave address appears on the I2C bus. • In the nonrepeat mode (RM = 0 in ICMDR): If STP = 0 in ICMDR, ARDY is set when the internal data counter counts down to 0. If STP = 1, ARDY is not affected (instead, the I2C generates a STOP condition when the counter reaches 0). • In the repeat mode (RM = 1): ARDY is set at the end of each data word transmitted from ICDXR.
1	NACK		No-acknowledgment interrupt flag bit. NACK applies when the I2C is a transmitter (master or slave). NACK indicates whether the I2C has detected an acknowledge bit (ACK) or a no-acknowledge bit (NACK) from the receiver. The CPU can poll NACK or use the NACK interrupt request.
		0	ACK received/NACK is not received. NACK is cleared by one of the following events: • An acknowledge bit (ACK) has been sent by the receiver. • NACK is manually cleared. To clear this bit, write a 1 to it. • The CPU reads the interrupt source register (ICISR) when the register contains the code for a NACK interrupt.
		1	• The I2C is reset (either when 0 is written to the IRS bit of ICMDR or when the processor is reset). NACK bit is received. The hardware detects that a no-acknowledge (NACK) bit has been received. Note: While the I2C performs a general call transfer, NACK is 1, even if one or more slaves send acknowledgment.
0	AL		Arbitration-lost interrupt flag bit (only applicable when the I2C is a master-transmitter). AL primarily indicates when the I2C has lost an arbitration contest with another master-transmitter. The CPU can poll AL or use the AL interrupt request.
		0	Arbitration is not lost. AL is cleared by one of the following events: • AL is manually cleared. To clear this bit, write a 1 to it. • The CPU reads the interrupt source register (ICISR) when the register contains the code for an AL interrupt.
		1	• The I2C is reset (either when 0 is written to the IRS bit of ICMDR or when the processor is reset). Arbitration is lost. AL is set by one of the following events: • The I2C senses that it has lost an arbitration with two or more competing transmitters that started a transmission almost simultaneously. • The I2C attempts to start a transfer while the BB (bus busy) bit is set to 1. When AL is set to 1, the MST and STP bits of ICMDR are cleared, and the I2C becomes a slave-receiver.

- **ICCLKL:**

Bit	Field	Value	Description
15-0	ICCL	0-FFFFh	Clock low-time divide-down value of 1-65536. The period of the module clock is multiplied by (ICCL + d) to produce the low-time duration of the I2C serial on the SCL pin.

- **ICCLKH:**

Bit	Field	Value	Description
15-0	ICCH	0-FFFFh	Clock high-time divide-down value of 1-65536. The period of the module clock is multiplied by (ICCH + d) to produce the high-time duration of the I2C serial on the SCL pin.

- **ICSAR:**

Bit	Field	Value	Description
15-10	Reserved	0	These reserved bit locations are always read as zeros. A value written to this field has no effect.

9-0	SADDR	0-3FFh	Slave address. Provides the slave address that the I2C transmits when it is in master-transmitter mode. In 7-bit addressing mode (XA = 0 in ICMDR): bits 6-0 provide the 7-bit slave address that the I2C transmits when it is in the master-transmitter mode. Bits 9-7 are ignored. In 10-bit addressing mode (XA = 1 in ICMDR): Bits 9-0 provide the 10-bit slave address that the I2C transmits when it is in the master-transmitter mode.

- **ICMDR:**

Bit	Field	Value	Description
15	NACKMOD		No-acknowledge (NACK) mode bit (only applicable when the I2C is a receiver).
		0	In slave-receiver mode: The I2C sends an acknowledge (ACK) bit to the transmitter during the each acknowledge cycle on the bus. The I2C only sends a no-acknowledge (NACK) bit if you set the NACKMOD bit. In master-receiver mode: The I2C sends an ACK bit during each acknowledge cycle until the internal data counter counts down to 0. When the counter reaches 0, the I2C sends a NACK bit to the transmitter. To have a NACK bit sent earlier, you
		1	must set the NACKMOD bit.
			In either slave-receiver or master-receiver mode: The I2C sends a NACK bit to the transmitter during the next acknowledge cycle on the bus. Once the NACK bit has been sent, NACKMOD is cleared. To send a NACK bit in the next acknowledge cycle, you must set NACKMOD before the rising edge of the last data bit.
14	FREE		This emulation mode bit is used to determine the state of the I2C when a breakpoint is encountered in the high-level language debugger.
		0	When I2C is master: If SCL is low when the breakpoint occurs, the I2C stops immediately and keeps driving SCL low, whether the I2C is the transmitter or the receiver. If SCL is high, the I2C waits until SCL becomes low and then stops. When I2C is slave: A breakpoint forces the I2C to stop when the current transmission/reception is complete.
		1	The I2C runs free; that is, it continues to operate when a breakpoint occurs.
13	STT		START condition bit (only applicable when the I2C is a master). The RM, STT, and STP bits determine when the I2C starts and stops data transmissions (see Table 15). Note that the STT and STP bits can be used to terminate the repeat mode.
		0	In master mode, STT is automatically cleared after the START condition has been generated. In slave mode, if STT is 0, the I2C does not monitor the bus for commands from a master. As a result, the I2C performs no data transfers.
		1	In master mode, setting STT to 1 causes the I2C to generate a START condition on the I2C-bus. In slave mode, if STT is 1, the I2C monitors the bus and transmits/receives data in response to commands from a master.
12	Reserved	0	These reserved bit locations are always read as zeros. A value written to this field has no effect.
11	STP		STOP condition bit (only applicable when the I2C is a master). The RM, STT, and STP bits determine when the I2C starts and stops data transmissions (see Table 15). Note that the STT and STP bits can be used to terminate the repeat mode.
		0	STP is automatically cleared after the STOP condition has been generated.
		1	STP has been set to generate a STOP condition when the internal data counter of the I2C counts down to 0.
10	MST		Master mode bit. MST determines whether the I2C is in the slave mode or the master mode. MST is automatically changed from 1 to 0 when the I2C master generates a STOP condition.
		0	See Table 16. Slave mode. The I2C is a slave and receives the serial clock from the master.
		1	Master mode. The I2C is a master and generates the serial clock on the SCL pin.
9	TRX		Transmitter mode bit. When relevant, TRX selects whether the I2C is in the transmitter mode or the receiver mode. Table 16 summarizes when TRX is used and when it is a don't
		0	care.
		1	Receiver mode. The I2C is a receiver and receives data on the SDA pin.
			Transmitter mode. The I2C is a transmitter and transmits data on the SDA pin.
8	XA		Expanded address enable bit.
		0	7-bit addressing mode (normal address mode). The I2C transmits 7-bit slave addresses (from bits 6-0 of ICSAR), and its own slave address has 7 bits (bits 6-0 of ICOAR).
		1	10-bit addressing mode (expanded address mode). The I2C transmits 10-bit slave addresses (from bits 9-0 of ICSAR), and its own slave address has 10 bits (bits 9-0 of ICOAR).
7	RM		Repeat mode bit (only applicable when the I2C is a master-transmitter). The RM, STT, and STP bits determine when the I2C starts and stops data transmissions (see Table 15). If the I2C is configured in slave mode, the RM bit is don't care.
		0	Nonrepeat mode. The value in the data count register (ICCNT) determines how many data words are received/transmitted by the I2C.
		1	Repeat mode. Data words are continuously received/transmitted by the I2C regardless of the value in ICCNT until the STP bit is manually set to 1.
6	DLB		Digital loopback mode bit (only applicable when the I2C is a master-transmitter). This bit disables or enables the digital loopback mode of the I2C. Note that DLB mode in the free data format mode (DLB = 1 and FDF = 1) is not supported.
		0	Digital loopback mode is disabled.
		1	Digital loopback mode is enabled. In this mode, the MST bit must be set to 1 and data transmitted out of ICDXR is received in ICDRR after n clock cycles by an internal path, where: n = ((I2C input clock frequency/prescaled module clock frequency) × 8) The transmit clock is also the receive clock. The address transmitted on the SDA pin is the

			address in ICOAR.
5	IRS		I2C reset bit. Note that if IRS is reset during a transfer, it can cause the I2C bus to hang (SDA and SCL are in a high-impedance state).
		0	The I2C is in reset/disabled. When this bit is cleared to 0, all status bits (in ICSTR) are set to their default values.
		1	The I2C is enabled.
4	STB	0 1	START byte mode bit (only applicable when the I2C is a master). As described in version 2.1 of the Philips I2C-bus specification, the START byte can be used to help a slave that needs extra time to detect a START condition. When the I2C is a slave, the I2C ignores a START byte from a master, regardless of the value of the STB bit. The I2C is not in the START byte mode. The I2C is in the START byte mode. When you set the START condition bit (STT), the I2C begins the transfer with more than just a START condition. Specifically, it generates: 1. A START condition 2. A START byte (0000 0001b) 3. A dummy acknowledge clock pulse 4. A repeated START condition The I2C sends the slave address that is in ICSAR.
3	FDF		Free data format mode bit. Note that DLB mode in the free data format mode (DLB = 1 and FDF = 1) is not supported. See Table 16.
		0	Free data format mode is disabled. Transfers use the 7-/10-bit addressing format selected by the XA bit.
		1	Free data format mode is enabled.
2-0	BC	0-7h	Bit count bits. BC defines the number of bits (2 to 8) in the next data word that is to be received or transmitted by the I2C. The number of bits selected with BC must match the data size of the other device. Note that when BC = 0, a data word has 8 bits. If the bit count is less than 8, receive data is right aligned in the D bits of ICDRR and the remaining D bits are undefined. Also, transmit data written to ICDXR must be right aligned.
		0	8 bits per data word
		1h	Reserved
		2h	2 bits per data word
		3h	3 bits per data word
		4h	4 bits per data word
		5h	5 bits per data word
		6h	6 bits per data word
		7h	7 bits per data word

- **ICPSC:**

Bit	Field	Value	Description
15-8	Reserved	0	These reserved bit locations are always read as zeros. A value written to this field has no effect.
7-0	IPSC	0-FFh	I2C prescaler divide-down value. IPSC determines how much the I2C input clock is divided to create the I2C prescaled module clock: I2C clock frequency = I2C input clock frequency/(IPSC + 1) Note: IPSC must be initialized while the I2C is in reset (IRS = 0 in ICMDR).

- **IODIR1 e IODIR2:**

Bit	Field	Value	Description
15-0	DIR	0-FFFFh	Data direction bits that configure the general-purpose IO pins as either inputs or outputs. Bit 0 of IODIR1 corresponds to GPIO 0 and Bit 0 of IODIR2
		0	corresponds to GPIO 16. Configure corresponding pin as an INPUT.
		1	Configure corresponding pin as an OUTPUT.

- **IODATAOUT1 e IODATAOUT2:**

Bit	Field	Value	Description
15-0	OUT	0-FFFFh	Data bits that are used to control the level of the GPIO pins configured as general-purpose output pins. If DIR = 0, then: • X = value stored on register but not reflected on the output pin. If DIR = 1, then: • 0 = set corresponding I/O pin to LOW. • 1 = set corresponding I/O pin to HIGH.

7.2 Registros del TLV320AIC34

Los registros del códec TLV320AIC34 se dividen en dos páginas de 128 registros. Para el funcionamiento del sistema final solo es necesario usar los registros de la página 0 de ambos bloques.

- **Registro 2:** Selección de frecuencia de muestreo del códec

Bit	Read/Write	Reset Value	Description
D7–D4	R/W	0000	ADC Sample Rate Select 0000: ADC fS = fS(ref)/1 0001: ADC fS = fS(ref)/1.5 0010: ADC fS = fS(ref)/2 0011: ADC fS = fS(ref)/2.5 0100: ADC fS = fS(ref)/3 0101: ADC fS = fS(ref)/3.5 0110: ADC fS = fS(ref)/4 0111: ADC fS = fS(ref)/4.5 1000: ADC fS = fS(ref)/5 1001: ADC fS = fS(ref)/5.5 1010: ADC fS = fS(ref)/6 1011–1111: Reserved.
D3–D0	R/W	0000	DAC Sample Rate Select 0000 : DAC fS = fS(ref)/1 0001 : DAC fS = fS(ref)/1.5 0010 : DAC fS = fS(ref)/2 0011 : DAC fS = fS(ref)/2.5 0100 : DAC fS = fS(ref)/3 0101 : DAC fS = fS(ref)/3.5 0110 : DAC fS = fS(ref)/4 0111 : DAC fS = fS(ref)/4.5 1000 : DAC fS = fS(ref)/5 1001: DAC fS = fS(ref)/5.5 1010: DAC fS = fS(ref) / 6 1011–1111 : Reserved.

- **Registros 3:** Programación PLL Registro A

Bit	Read/Write	Reset Value	Description
D7	R/W	0	PLL Control Bit 0: PLL is disabled. 1: PLL is enabled.
D6–D3	R/W	0010	PLL Q Value 0000: Q = 16 0001: Q = 17 0010: Q = 2 0011: Q = 3 0100: Q = 4 … 1110: Q = 14 1111: Q = 15
D2–D0	R/W	000	PLL P Value 000: P = 8 001: P = 1 010: P = 2 011: P = 3 100: P = 4 101: P = 5 110: P = 6 111: P = 7

- **Registro 4:** Programación PLL Registro B

Bit	Read/Write	Reset Value	Description
D7–D2	R/W	0000 01	PLL J Value 0000 00: Reserved. 0000 01: J = 1 0000 10: J = 2 0000 11: J = 3 … 1111 10: J = 62 1111 11: J = 63
D1–D0	R/W	00	Reserved. Write only zeros to these bits.

- **Registro 5:** Programación PLL Registro C

Bit	Read/Write	Reset Value	Description
D7–D0	R/W	0000 0000	PLL D value – Eight most-significant bits of a 14-bit unsigned integer. Valid values for D are from zero to 9999, represented by a 14-bit integer located in page 0,

			registers 5–6. Values should not be written into these registers that would result in a D value outside the valid range.

- **Registro 6:** Programación PLL Registro D

Bit	Read/Write	Reset Value	Description
D7–D2	R/W	0000 00	PLL D value – Six least-significant bits of a 14-bit unsigned integer. Valid values for D are from zero to 9999, represented by a 14-bit integer located in page 0, registers 5–6. Values should not be written into these registers that would result in a D value outside the valid range.
D1–D0	R	00	Reserved. Write only zeros to these bits.

- **Registro 7:** Configuración de la ruta de datos del códec

Bit	Read/Write	Reset Value	Description
D7	R/W	0	fS(ref) Setting This register setting controls timers related to the AGC time constants. 0: fS(ref) = 48 kHz 1: fS(ref) = 44.1 kHz
D6	R/W	0	ADC Dual Rate Control 0: ADC dual-rate mode is disabled. 1: ADC dual-rate mode is enabled. Note: ADC dual-rate mode must match DAC dual-rate mode.
D5	R/W	0	DAC Dual Rate Control 0: DAC dual rate mode is disabled. 1: DAC dual rate mode is enabled.
D4–D3	R/W	00	Left-DAC Data-Path Control 00: Left-DAC data path is off (muted). 01: Left-DAC data path plays left-channel input data. 10: Left-DAC data path plays right-channel input data. 11: Left-DAC data path plays mono mix of left-and right-channel input data.
D2–D1	R/W	00	Right-DAC Data Path Control 00: Right-DAC data path is off (muted). 01: Right-DAC data path plays right-channel input data. 10: Right-DAC data path plays left-channel input data. 11: Right-DAC data path plays mono mix of left-and right-channel input data.
D0	R/W	0	Reserved. Write only zero to this bit.

- **Registro 8:** Interface serie de datos de audio A

Bit	Read/Write	Reset Value	Description
D7	R/W	0	Bit Clock Directional Control 0: BCLK_x (or GPIO2_x if programmed as BCLK_x) is an input (slave mode). 1: BCLK_x (or GPIO2_x if programmed as BCLK_x) is an output (master mode).
D6	R/W	0	Word Clock Directional Control 0: WCLK_x (or GPIO1_x if programmed as WCLK_x) is an input (slave mode). 1: WCLK_x (or GPIO1_x if programmed as WCLK_x) is an output (master mode).
D5	R/W	0	Serial Output Data Driver (DOUT_x) 3-State Control 0: Do not place DOUT_x in high-impedance state when valid data is not being sent. 1: Place DOUT_x in high-impedance state when valid data is not being sent.
D4	R/W	0	Bit/ Word Clock Drive Control 0: BCLK_x (or GPIO2_x if programmed as BCLK_x) / WCLK_x (or GPIO1_x if programmed as WCLK_x) does not continue to be transmitted when running in master mode if codec is powered down. 1: BCLK_x (or GPIO2_x if programmed as BCLK_x) / WCLK_x (or GPIO1_x if programmed as WCLK_x) continues to be transmitted when running in master mode, even if codec is powered down.
D3	R/W	0	Reserved. Do not write to this register bit.
D2	R/W	0	3-D Effect Control 0: Disable 3-D digital effect processing. 1: Enable 3-D digital effect processing.
D1–D0	R/W	00	Digital Microphone Functionality Control 00: Digital microphone support is disabled. 01: Digital microphone support is enabled with an oversampling rate of 128. 10: Digital microphone support is enabled with an oversampling rate of 64. 11: Digital microphone support is enabled with an oversampling rate of 32.

- **Registro 9:** Interface serie de datos de audio B

Bit	Read/Write	Reset Value	Description
D7–D6	R/W	00	Audio Serial Data Interface Transfer Mode 00: Serial data bus uses I2S mode.

			01: Serial data bus uses DSP mode. 10: Serial data bus uses right-justified mode. 11: Serial data bus uses left-justified mode.
D5–D4	R/W	00	Audio Serial Data Word Length Control 00: Audio data word length = 16 bits 01: Audio data word length = 20 bits 10: Audio data word length = 24 bits 11: Audio data word length = 32 bits
D3	R/W	0	Bit Clock Rate Control This register only has effect when bit clock is programmed as an output. 0: Continuous-transfer mode used to determine master-mode bit clock rate 1: 256-clock transfer mode used, resulting in 256 bit clocks per frame
D2	R/W	0	DAC Re-Sync 0: Don't care 1: Re-sync stereo DAC with codec interface if the group delay changes by more than ±DAC (fS/4).
D1	R/W	0	ADC Re-Sync 0: Don't care 1: Re-sync stereo ADC with codec interface if the group delay changes by more than ±ADC (fS/4).
D0	R/W	0	Re-Sync Mute Behavior 0: Re-sync is done without soft-muting the channel. (ADC/DAC) 1: Re-sync is done by internally soft-muting the channel. (ADC/DAC)

- **Registro 11:** Registro de desbordamiento de audio del códec

Bit	Read/Write	Reset Value	Description
D7	R	0	Left-ADC Overflow Flag This is a sticky bit, so it stays set if an overflow occurs, even if the overflow condition is removed. The register bit is reset to 0 after it is read. 0: No overflow has occurred. 1: An overflow has occurred.
D6	R	0	Right-ADC Overflow Flag This is a sticky bit, so it stays set if an overflow occurs, even if the overflow condition is removed. The register bit is reset to 0 after it is read. 0: No overflow has occurred. 1: An overflow has occurred.
D5	R	0	Left-DAC Overflow Flag This is a sticky bit, so it stays set if an overflow occurs, even if the overflow condition is removed. The register bit is reset to 0 after it is read. 0: No overflow has occurred. 1: An overflow has occurred.
D4	R	0	Right DAC Overflow Flag This is a sticky bit, so it stays set if an overflow occurs, even if the overflow condition is removed. The register bit is reset to 0 after it is read. 0: No overflow has occurred. 1: An overflow has occurred.
D3–D0	R/W	0001	PLL R Value 0000: R = 16 0001 : R = 1 0010 : R = 2 0011 : R = 3 0100 : R = 4 … 1110: R = 14 1111: R = 15

- **Registro 15:** Controlador de ganancia PGA del canal izquierdo del ADC

Bit	Read/Write	Reset Value	Description
D7	R/W	1	Left-ADC PGA Mute 0: The left-ADC PGA is not muted. 1: The left-ADC PGA is muted.
D6–D0	R/W	000 0000	Left-ADC PGA Gain Setting 000 0000: Gain = 0 dB 000 0001: Gain = 0.5 dB 000 0010: Gain = 1 dB … 111 0110: Gain = 59 dB 111 0111: Gain = 59.5 dB 111 1000: Gain = 59.5 dB … 111 1111: Gain = 59.5 dB

- **Registro 16:** Controlador de ganancia PGA del canal derecho del ADC

Bit	Read/Write	Reset Value	Description

D7	R/W	1	Right-ADC PGA Mute 0: The left-ADC PGA is not muted. 1: The right-ADC PGA is muted.
D6–D0	R/W	000 0000	Right-ADC PGA Gain Setting 000 0000: Gain = 0 dB 000 0001: Gain = 0.5 dB 000 0010: Gain = 1 dB … 111 0110: Gain = 59 dB 111 0111: Gain = 59.5 dB 111 1000: Gain = 59.5 dB … 111 1111: Gain = 59.5 dB

- **Registro 17:** Control de MIC3L_x y MIC3R_x del ADC izquierdo

Bit	Read/Write	Reset Value	Description
D7–D4	R/W	1111	MIC3L_x Input Level Control for Left-ADC PGA Mix Setting the input level control to one of the following gains automatically connects MIC3L_x to the left-ADC PGA mix. 0000: Input level control gain = 0 dB 0001: Input level control gain = –1.5 dB 0010: Input level control gain = –3 dB 0011: Input level control gain = –4.5 dB 0100: Input level control gain = –6 dB 0101: Input level control gain = –7.5 dB 0110: Input level control gain = –9 dB 0111: Input level control gain = –10.5 dB 1000: Input level control gain = –12 dB 1001–1110: Reserved. Do not write these sequences to these register bits. 1111: MIC3L_x is not connected to the left-ADC PGA.
D3–D0	R/W	1111	MIC3R_x Input Level Control for Left-ADC PGA Mix Setting the input level control to one of the following gains automatically connects MIC3R_x to the left-ADC PGA mix. 0000: Input level control gain = 0 dB 0001: Input level control gain = –1.5 dB 0010: Input level control gain = –3 dB 0011: Input level control gain = –4.5 dB 0100: Input level control gain = –6 dB 0101: Input level control gain = –7.5 dB 0110: Input level control gain = –9 dB 0111: Input level control gain = –10.5 dB 1000: Input level control gain = –12 dB 1001–1110: Reserved. Do not write these sequences to these register bits. 1111: MIC3R_x is not connected to the left-ADC PGA.

- **Registro 18:** Control de MIC3L_x y MIC3R_x del ADC derecho

Bit	Read/Write	Reset Value	Description
D7–D4	R/W	1111	MIC3L_x Input Level Control for Right-ADC PGA Mix Setting the input level control to one of the following gains automatically connects MIC3L_x to the right-ADC PGA mix. 0000: Input level control gain = 0 dB 0001: Input level control gain = –1.5 dB 0010: Input level control gain = –3 dB 0011: Input level control gain = –4.5 dB 0100: Input level control gain = –6 dB 0101: Input level control gain = –7.5 dB 0110: Input level control gain = –9 dB 0111: Input level control gain = –10.5 dB 1000: Input level control gain = –12 dB 1001–1110: Reserved. Do not write these sequences to these register bits. 1111: MIC3L_x is not connected to the right-ADC PGA.
D3–D0	R/W	1111	MIC3R_x Input Level Control for Right-ADC PGA Mix Setting the input level control to one of the following gains automatically connects MIC3R_x to the right-ADC PGA mix. 0000: Input level control gain = 0 dB 0001: Input level control gain = –1.5 dB 0010: Input level control gain = –3 dB 0011: Input level control gain = –4.5 dB 0100: Input level control gain = –6 dB 0101: Input level control gain = –7.5 dB 0110: Input level control gain = –9 dB 0111: Input level control gain = –10.5 dB 1000: Input level control gain = –12 dB 1001–1110: Reserved. Do not write these sequences to these register bits. 1111: MIC3R_x is not connected to the right-ADC PGA.

- **Registro 19:** Controlador ADC izquierdo

Bit	Read/Write	Reset Value	Description
D7	R/W	0	LINE1L Single-Ended vs Fully Differential Control If LINE1L is selected to both left- and right-ADC channels, both connections must use the same configuration (single-ended or fully differential mode). 0: LINE1L is configured in single-ended mode. 1: LINE1L is configured in fully differential mode.
D6–D3	R/W	1111	LINE1L Input Level Control for Left-ADC PGA Mix Setting the input level control to one of the following gains automatically connects LINE1L to the left-ADC PGA mix. 0000: Input level control gain = 0 dB 0001: Input level control gain = –1.5 dB 0010: Input level control gain = –3 dB 0011: Input level control gain = –4.5 dB 0100: Input level control gain = –6 dB 0101: Input level control gain = –7.5 dB 0110: Input level control gain = –9 dB 0111: Input level control gain = –10.5 dB 1000: Input level control gain = –12 dB 1001–1110: Reserved. Do not write these sequences to these register bits. 1111: LINE1L is not connected to the left-ADC PGA.
D2	R/W	0	Left-ADC Channel Power Control 0: Left-ADC channel is powered down. 1: Left-ADC channel is powered up.
D1–D0	R/W	00	Left-ADC PGA Soft-Stepping Control 00: Left-ADC PGA soft-stepping at once per sample period 01: Left-ADC PGA soft-stepping at once per two ηe periods 10–11: Left-ADC PGA soft-stepping is disabled.

- **Registro 22:** Controlador ADC izquierdo

Bit	Read/Write	Reset Value	Description
D7	R/W	0	LINE1R Single-Ended vs Fully Differential Control If LINE1R is selected to both left- and right-ADC channels, both connections must use the same configuration (single-ended or fully differential mode). 0: LINE1R is configured in single-ended mode. 1: LINE1LRis configured in fully differential mode.
D6–D3	R/W	1111	LINE1L Input Level Control for Right-ADC PGA Mix Setting the input level control to one of the following gains automatically connects LINE1R to the left-ADC PGA mix. 0000: Input level control gain = 0 dB 0001: Input level control gain = –1.5 dB 0010: Input level control gain = –3 dB 0011: Input level control gain = –4.5 dB 0100: Input level control gain = –6 dB 0101: Input level control gain = –7.5 dB 0110: Input level control gain = –9 dB 0111: Input level control gain = –10.5 dB 1000: Input level control gain = –12 dB 1001–1110: Reserved. Do not write these sequences to these register bits. 1111: LINE1L is not connected to the left-ADC PGA.
D2	R/W	0	Right-ADC Channel Power Control 0: Right-ADC channel is powered down. 1: Right-ADC channel is powered up.
D1–D0	R/W	00	Right-ADC PGA Soft-Stepping Control 00: Right-ADC PGA soft-stepping at once per sample period 01: Right-ADC PGA soft-stepping at once per two ηe periods 10–11: Right-ADC PGA soft-stepping is disabled.

- **Registro 25:** Controlador MICBIAS_x

Bit	Read/Write	Reset Value	Description
D7–D6	R/W	00	MICBIAS_x Level Control 00: MICBIAS_x output is powered down. 01: MICBIAS_x output is powered to 2 V. 10: MICBIAS_x output is powered to 2.5 V. 11: MICBIAS_x output is connected to AVDD.
D5–D4	R/W	00	Digital Microphone Control 00: If digital MIC is enabled, both left and right digital MICs are available. 01: If digital MIC is enabled, left digital MIC and right ADC are available. 10: If digital MIC is enabled, left ADC and right digital MIC are available. 11: Reserved. Do not write this sequence to these register bits.
D3	R	0	Reserved. Do not write to this register bit.
D2–D0	R	XXX	Reserved. Write only zeros to these register bits.

- **Registro 36:** Flags ADC

Bit	Read/Write	Reset Value	Description
D7	R	0	Left-ADC PGA Status 0: Applied gain and programmed gain are not the same. 1: Applied gain = programmed gain
D6	R	0	Left-ADC Power Status 0: Left ADC is in a power-down state. 1: Left ADC is in a power-up state.
D5	R	0	Left-AGC Signal Detection Status 0: Signal power is greater than noise threshold. 1: Signal power is less than noise threshold.
D4	R	0	Left-AGC Saturation Flag 0: Left AGC is not saturated. 1: Left-AGC gain applied = maximum allowed gain for left AGC
D3	R	0	Right-ADC PGA Status 0: Applied gain and programmed gain are not the same. 1: Applied gain = programmed gain
D2	R	0	Right-ADC Power Status 0: Right ADC is in a power-down state. 1: Right ADC is in a power-up state.
D1	R	0	Right-AGC Signal Detection Status 0: Signal power is greater than noise threshold. 1: Signal power is less than noise threshold.
D0	R	0	Right-AGC Saturation Flag 0: Right AGC is not saturated. 1: Right-AGC gain applied = maximum allowed gain for right AGC

- **Registro 37:** Controlador de DAC

Bit	Read/Write	Reset Value	Description
D7	R/W	0	Left-DAC Power Control 0: Left DAC is not powered up. 1: Left DAC is powered up.
D6	R/W	0	Right-DAC Power Control 0: Right DAC is not powered up. 1: Right DAC is powered up.
D5–D4	R/W	00	HPLCOM_x Output Driver Configuration Control 00: HPLCOM_x configured as differential of HPLOUT_x 01: HPLCOM_x configured as constant VCM output 10: HPLCOM_x configured as independent single-ended output 11: Reserved. Do not write this sequence to these register bits.
D3–D0	R	0000	Reserved. Write only zeros to these register bits.

- **Registro 38:** Control de salida de gran potencia

Bit	Read/Write	Reset Value	Description
D7–D6	R	00	Reserved. Write only zeros to these register bits.
D5–D3	R/W	000	HPRCOM_x Output Driver Configuration Control 000: HPRCOM_x configured as differential of HPROUT_x 001: HPRCOM_x configured as constant VCM output 010: HPRCOM_x configured as independent single-ended output 011: HPRCOM_x configured as differential of HPLCOM_x 100: HPRCOM_x configured as external feedback with HPLCOM_x as constant VCM output 101–111: Reserved.
D2	R/W	0	Short-Circuit Protection Control 0: Short-circuit protection on all high-power output drivers is disabled. 1: Short-circuit protection on all high-power output drivers is enabled.
D1	R/W	0	Short-Circuit Protection-Mode Control 0: If short-circuit protection is enabled, it limits the maximum current to the load. 1: If short-circuit protection is enabled, it powers down the output driver automatically when a short is detected.
D0	R	0	Reserved. Write only zero to this register bit.

- **Registro 43:** Control de volume del DAC izquierdo

Bit	Read/Write	Reset Value	Description
D7	R/W	1	Left-DAC Digital Mute 0: The left-DAC channel is not muted. 1: The left-DAC channel is muted.

D6–D0	R/W	000 0000	Left-DAC Digital Volume Control Setting 000 0000: Gain = 0 dB 000 0001: Gain = –0.5 dB 000 0010: Gain = –1 dB … 111 1101: Gain = –62.5 dB 111 1110: Gain = –63 dB 111 1111: Gain = –63.5 dB

- **Registro 44:** Control de volume del DAC derecho

Bit	Read/Write	Reset Value	Description
D7	R/W	1	Right-DAC Digital Mute 0: The right-DAC channel is not muted. 1: The right-DAC channel is muted.
D6–D0	R/W	000 0000	Right-DAC Digital Volume Control Setting 000 0000: Gain = 0 dB 000 0001: Gain = –0.5 dB 000 0010: Gain = –1 dB … 111 1101: Gain = –62.5 dB 111 1110: Gain = –63 dB 111 1111: Gain = –63.5 dB

- **Registro 47:** Control volumen de salida L1 de DAC izquierdo

Bit	Read/Write	Reset Value	Description
D7	R/W	0	DAC_L1 Output Routing Control 0: DAC_L1 is not routed to HPLOUT_x. 1: DAC_L1 is routed to HPLOUT_x.
D6–D0	R/W	000 0000	DAC_L1 to HPLOUT_x Analog Volume Control For 7-bit register setting versus analog gain values.

- **Registro 51:** Control de nivel de HPLOUT_x

Bit	Read/Write	Reset Value	Description
D7–D4	R/W	0000	HPLOUT_x Output Level Control 0000: Output level control = 0 dB 0001: Output level control = 1 dB 0010: Output level control = 2 dB ... 1000: Output level control = 8 dB 1001: Output level control = 9 dB 1010–1111: Reserved. Do not write these sequences to these register bits.
D3	R/W	0	HPLOUT_x Mute 0: HPLOUT_x is muted. 1: HPLOUT_x is not muted.
D2	R/W	1	HPLOUT_x Power Down Drive Control 0: HPLOUT_x is weakly driven to a common mode when powered down. 1: HPLOUT_x is high-impedance when powered down.
D1	R	1	HPLOUT_x Volume Control Status 0: All programmed gains to HPLOUT_x have been applied. 1: Not all programmed gains to HPLOUT_x have been applied yet.
D0	R/W	0	HPLOUT_x Power Control 0: HPLOUT_x is not fully powered up. 1: HPLOUT_x is fully powered up.

- **Registro 64:** Control volumen de salida R1 de DAC derecho

Bit	Read/Write	Reset Value	Description
D7	R/W	0	DAC_R1 Output Routing Control 0: DAC_R1 is not routed to HPROUT_x. 1: DAC_R1 is routed to HPROUT_x.
D6–D0	R/W	000 0000	DAC_R1 to HPROUT_x Analog Volume Control For 7-bit register setting

- **Registro 65:** Control de nivel de HPROUT_x

Bit	Read/Write	Reset Value	Description
D7–D4	R/W	0000	HPROUT_x Output Level Control 0000: Output level control = 0 dB 0001: Output level control = 1 dB 0010: Output level control = 2 dB ...

			1000: Output level control = 8 dB 1001: Output level control = 9 dB 1010–1111: Reserved.
D3	R/W	0	HPROUT_x Mute 0: HPROUT_x is muted. 1: HPROUT_x is not muted.
D2	R/W	1	HPROUT_x Power-Down Drive Control 0: HPROUT_x is weakly driven to a common mode when powered down. 1: HPROUT_x is high-impedance when powered down.
D1	R	1	HPROUT_x Volume Control Status 0: All programmed gains to HPROUT_x have been applied. 1: Not all programmed gains to HPROUT_x have been applied yet.
D0	R/W	0	HPROUT_x Power Control 0: HPROUT_x is not fully powered up. 1: HPROUT_x is fully powered up.

- **Registro 94:** Módulo estado de potencia

Bit	Read/Write	Reset Value	Description
D7	R	0	Left-DAC Power Status 0:Left DAC is not fully powered up. 1: Left DAC is fully powered up.
D6	R	0	Right-DAC Power Status 0: Right DAC is not fully powered up. 1: Right DAC is fully powered up.
D5	R	0	MONO_LOP_x and MONO_LOM_x Power Status 0: MONO_LOP_x and MONO_LOM_x output driver is powered down. 1: MONO_LOP_x and MONO_LOM_x output driver is powered up.
D4	R	0	LEFT_LOP_x and LEFT_LOM_x Power Status 0: LEFT_LOP_x and LEFT_LOM_x output driver is powered down. 1: LEFT_LOP/M_ output driver is powered up.
D3	R	0	RIGHT_LOP_x and RIGHT_LOM_x Power Status 0:RIGHT_LOP_x and RIGHT_LOM_x is not fully powered up. 1: RIGHT_LOP_x and RIGHT_LOM_x is fully powered up.
D2	R	0	HPLOUT_x Driver Power Status 0: HPLOUT_x driver is not fully powered up. 1: HPLOUT_x driver is fully powered up.
D1	R	0	HPROUT_x Driver Power Status 0: HPROUT_x Driver is not fully powered up. 1: HPROUT_x Driver is fully powered up.
D0	R	0	Reserved. Do not write to this register bit.

- **Registro 98:** Control GPIO1_x

Bit	Read/Write	Reset Value	Description
D7–D4	R/W	0000	GPIO1_x Output Control 0000: GPIO1_x is disabled. 0001: GPIO1_x used for audio serial data bus ADC word clock 0010: GPIO1_x output = clock mux output divided by 1 (M = 1) 0011: GPIO1_x output = clock mux output divided by 2 (M = 2) 0100: GPIO1_x output = clock mux output divided by 4 (M = 4) 0101: GPIO1_x output = clock mux output divided by 8 (M = 8) 0110: GPIO1_x output = short-circuit interrupt 0111: GPIO1_x output = AGC noise interrupt 1000: GPIO1_x = general-purpose input 1001: GPIO1_x = general-purpose output 1010: GPIO1_x output = digital microphone modulator clock 1011: GPIO1_x = word clock for audio serial data bus (programmable as input or output) 1100: GPIO1_x output = hook-switch/button-press interrupt (interrupt polarity: active-high, typical interrupt duration: button pressed time + clock resolution. Clock resolution depends on debounce programmability. Typical interrupt delay from button: debounce duration + 0.5 ms) 1101: GPIO1_x output = jack/headset detection interrupt 1110: GPIO1_x output = jack/headset detection interrupt OR button-press interrupt 1111: GPIO1_x output = jack/headset detection OR button press OR short-circuit detection OR AGC noise-detection interrupt
D3	R/W	0	GPIO1_x Clock Mux Output Control 0: GPIO1_x clock mux output = PLL output 1: GPIO1_x clock mux output = clock divider mux output
D2	R/W	0	GPIO1_x Interrupt Duration Control 0: GPIO1_x interrupt occurs as a single active-high pulse of typical 2-ms duration.

			1: GPIO1_x interrupt occurs as continuous pulses until the interrupt flags register (register 96) is read by the host.
D1	R	0	GPIO1_x General-Purpose Input Value 0: A logic-low level is input to GPIO1_x. 1: A logic-high level is input to GPIO1_x.
D0	R/W	0	GPIO1_x General-Purpose Output Value 0: GPIO1_x outputs a logic-low level. 1: GPIO1_x outputs a logic-high level.

- **Registro 99:** Control GPIO2_x

Bit	Read/Write	Reset Value	Description
D7–D4	R/W	0000	GPIO2_x Output Control 0000: GPIO2_x is disabled. 0001: Reserved. Do not write this sequence to these register bits. 0010: GPIO2_x output = jack/headset detect interrupt (interrupt polarity: active-high. Typical interrupt duration: 1.75 ms) 0011: GPIO2_x = general-purpose input 0100: GPIO2_x = general-purpose output 0101–0111: GPIO2_x input = digital microphone input, data sampled on clock rising and falling edges 1000: GPIO2_x = bit clock for audio serial data bus (programmable as input or output) 1001: GPIO2_x output = headset detect OR button-press interrupt 1010: GPIO2_x output = headset detect OR button press OR short-circuit detect OR AGC noise-detect interrupt 1011: GPIO2_x output = short-circuit detect OR AGC noise-detect interrupt 1100: GPIO2_x output = headset detect OR button press or short-circuit detect interrupt 1101: GPIO2_x output = short-circuit detect interrupt 1110: GPIO2_x output = AGC noise-detect interrupt 1111: GPIO2_x output = button-press/hookswitch interrupt
D3	R/W	0	GPIO2_x General-Purpose Output Value 0: GPIO2_x_x outputs a logic-low level. 1: GPIO2_x outputs a logic-high level.
D2	R	0	GPIO2_x General-Purpose Input Value 0: A logic-low level is input to GPIO2_x. 1: A logic-high level is input to GPIO2_x.
D1	R/W	0	GPIO2_x Interrupt Duration Control 0: GPIO2_x interrupt occurs as a single active-high pulse of typical 2-ms duration. 1: GPIO2_x interrupt occurs as continuous pulses until the interrupt flags register (register 96) is read by the host.
D0	R	0	Reserved. Do not write to this register bit.

- **Registro 102:** Control de reloj

Bit	Read/Write	Reset Value	Description
D7–D6	R/W	00	CLKDIV_IN Source Selection 00: CLKDIV_IN uses MCLK_x. 01: CLKDIV_IN uses GPIO2_x. 10: CLKDIV_IN uses BCLK_x. 1 1: Reserved. Do not write this sequence to these register bits.
D5–D4	R/W	00	PLLCLK_IN Source Selection 00: PLLCLK_IN uses MCLK_x. 01: PLLCLK_IN uses GPIO2_x. 10: PLLCLK _IN uses BCLK_x. 11: Reserved. Do not write this sequence to these register bits.
D3–D0	R/W	0010	PLL Clock Divider N Value 0000: N = 16 0001: N = 17 0010: N = 2 0011: N = 3 … 1111: N = 15

7.3 Registros del TLV320AIC3204

EL códec TLV320AIC3204 dispone de dos páginas con 128 registros cada uno. Para la configuración del códec se hace uso de amabas páginas.

1. Página 0 del banco de registros:

- **Registro 5:** Configuración del reloj mediante PLL con valores P y R

Bit	Read/Write	Reset Value	Descripcion
D7	R/W	0	PLL Power Up 0: PLL is powered down 1: PLL is powered up
D6–D4	R/W	001	PLL divider P Value 000: P=8 001: P=1 010: P=2 … 110: P=6 111: P=7
D3–D0	R/W	0001	PLL divider R Value 000: Reserved, do not use 001: R=1 010: R=2 011: R=3 100: R=4 101…111: Reserved, do not use

- **Registro 7:** Configuración del reloj mediante PLL con valor D (MSB)

Bit	Read/Write	Reset Value	Descripcion
D7-D6	R	00	Reserved. Write only default values any value other than default
D5–D0	R/W	00 0000	PLL divider D value (MSB) PLL divider D value(MSB) & PLL divider D value(LSB) 00 0000 0000 0000: D=0000 00 0000 0000 0001: D=0001 … 10 0111 0000 1110: D=9998 10 0111 0000 1111: D=9999 10 0111 0001 0000 … 11 1111 1111 1111: Do not use Note: This register will be updated only when the register 8 is written immediately after.

- **Registro 8:** Configuración del reloj mediante PLL con valor D (LSB)

Bit	Read/Write	Reset Value	Descripcion
D7–D0	R/W	0000 0000	PLL divider D value (LSB) PLL divider D value(MSB) & PLL divider D value(LSB) 00 0000 0000 0000: D=0000 00 0000 0000 0001: D=0001 … 10 0111 0000 1110: D=9998 10 0111 0000 1111: D=9999 10 0111 0001 0000 … 11 1111 1111 1111: Do not use Note: Page-0, Reg-8 should be written immediately after register 7.

- **Registro 11:** Configuración del reloj mediante valores NDAC

Bit	Read/Write	Reset Value	Descripcion
D7	R/W	0	NDAC Divider Power Control 0: NDAC divider powered down 1: NDAC divider powered up
D6–D0	R/W	000 0001	NDAC Value 000 0000: NDAC=128 000 0001: NDAC=1 000 0010: NDAC=2 … 111 1110: NDAC=126 111 1111: NDAC=127 Note: Please check the clock frequency requirements in the Overview section

- **Registro 12:** Configuración del reloj mediante valores MDAC

Bit	Read/Write	Reset Value	Descripcion
D7	R/W	0	MDAC Divider Power Control 0: MDAC divider powered down 1: MDAC divider powered up
D6–D0	R/W	000 0001	MDAC Value 000 0000: MDAC=128

			000 0001: MDAC=1 000 0010: MDAC=2 … 111 1110: MDAC=126 111 1111: MDAC=127 Note: Please check the clock frequency requirements in the Overview section

- **Registro 13:** Configuración DAC valor OSR (MSB)

Bit	Read/Write	Reset Value	Descripcion
D7-D2	R/W	0000 00	Reserved. Write only default values
D1–D0	R/W	00	DAC OSR (DOSR) Setting DAC OSR(MSB) & DAC OSR(LSB) 00 0000 0000: DOSR=1024 00 0000 0001: DOSR=1 00 0000 0010: DOSR=2 … 11 1111 1110: DOSR=1022 11 1111 1111: DOSR=1023 Note: This register is updated when register 14 is written to immediately after register 13

- **Registro 14:** Configuración DAC valor OSR (LSB)

Bit	Read/Write	Reset Value	Descripcion
D7-D2	R	0000 00	Reserved. Write only default values
D1–D0	R/W	00	DAC OSR (DOSR) Setting DAC OSR(MSB) & DAC OSR(LSB) 00 0000 0000: DOSR=1024 00 0000 0001: Reserved. Do not use 00 0000 0010: DOSR=2 … 11 1111 1110: DOSR=1022 11 1111 1111: Reserved. Do not use Note: This register should be written immediately after register 14

- **Registro 27:** Configuración de interface de audio

Bit	Read/Write	Reset Value	Descripcion
D7-D6	R/W	00	Audio Interface Selection 00: Audio Interface = I2S 01: Audio Interface = DSP 10: Audio Interface = RJF 11: Audio Interface = LJF
D5–D4	R/W	00	Audio Data Word length 00: Data Word length = 16 bits 01: Data Word length = 20 bits 10: Data Word length = 24 bits 11: Data Word length = 32 bits
D3	R/W	00	BCLK Direction Control 0: BCLK is input to the device 1: BCLK is output from the device
D2	R/W	0	WCLK Direction Control 0: WCLK is input to the device 1: WCLK is output from the device
D1	R	0	Reserved. Write only default value
D0	R/W	0	DOUT High Impendance Output Control 0: DOUT will not be high impedance while Audio Interface is active 1: DOUT will be high impedance after data has been transferred

- **Registro 63:** Configuración DAC

Bit	Read/Write	Reset Value	Descripcion
D7	R/W	0	Left DAC Channel Power Control 0: Left DAC Channel Powered Down 1: Left DAC Channel Powered Up
D6	R/W	0	Right DAC Channel Power Control 0: Right DAC Channel Powered Down 1: Right DAC Channel Powered Up
D5-D4	R/W	01	Left DAC Data path Control 00: Left DAC data is disabled 01: Left DAC data Left Channel Audio Interface Data

			10: Left DAC data is Right Channel Audio Interface Data 11: Left DAC data is Mono Mix of Left and Right Channel Audio Interface Data
D3-D2	R/W	01	Right DAC Data path Control 00: Right DAC data is disabled 01: Right DAC data Right Channel Audio Interface Data 10: Right DAC data is Left Channel Audio Interface Data 11: Right DAC data is Mono Mix of Left and Right Channel Audio Interface Data
D1-D0	R/W	00	DAC Channel Volume Control's Soft-Step control 00: Soft-Stepping is 1 step per 1 DAC Word Clock 01: Soft-Stepping is 1 step per 2 DAC Word Clocks 10: Soft-Stepping is disabled 11: Reserved. Do not use

- **Registro 64:** Configuración DAC

Bit	Read/Write	Reset Value	Descripcion
D7	R/W	0	Right Modulator Output Control 0: When Right DAC Channel is powered down, the data is zero. 1: When Right DAC Channel is powered down, the Right DAC Modulator output data is inverted version of Left DAC Modulator Output. Can be used when differential mono output is used
D6-D4	R/W	000	DAC Auto Mute Control 000: Auto Mute disabled 001: DAC is auto muted if input data is DC for more than 100 consecutive inputs 010: DAC is auto muted if input data is DC for more than 200 consecutive inputs 011: DAC is auto muted if input data is DC for more than 400 consecutive inputs 100: DAC is auto muted if input data is DC for more than 800 consecutive inputs 101: DAC is auto muted if input data is DC for more than 1600 consecutive inputs 110: DAC is auto muted if input data is DC for more than 3200 consecutive inputs 111: DAC is auto muted if input data is DC for more than 6400 consecutive inputs
D3	R/W	1	Left DAC Channel Mute Control 0: Left DAC Channel not muted 1: Left DAC Channel muted
D2	R/W	1	Right DAC Channel Mute Control 0: Right DAC Channel not muted 1: Right DAC Channel muted
D1-D0	R/W	00	DAC Master Volume Control 00: Left and Right Channel have independent volume control 01: Left Channel Volume is controlled by Right Channel Volume Control setting 10: Right Channel Volume is controlled by Left Channel Volume Control setting 11: Reserved. Do not use

- **Registro 65:** Control de volumen de DAC izqueirdo

Bit	Read/Write	Reset Value	Descripcion
D7-D0	R/W	0000 0000	Left DAC Channel Digital Volume Control Setting 0111 1111-0011 0001: Reserved. Do not use 0011 0000: Digital Volume Control = +24dB 0010 1111: Digital Volume Control = +23.5dB … 0000 0001: Digital Volume Control = +0.5dB 0000 0000: Digital Volume Control = 0.0dB 1111 1111: Digital Volume Control = -0.5dB … 1000 0010: Digital Volume Control = -63dB 1000 0001: Digital Volume Control = -63.5dB 1000 0000: Reserved. Do not use

2. Página 1 del banco de registros:

- **Registro 9:** Control de potencia de salida

Bit	Read/Write	Reset Value	Descripcion
D7-D6	R/W	00	Reserved. Write only default value
D5	R/W	0	0: HPL is powered down 1: HPL is powered up
D4	R/W	0	0: HPR is powered down 1: HPR is powered up
D3	R/W	0	0: LOL is powered down 1: LOL is powered up
D2	R/W		0: LOR is powered down 1: LOR is powered up
D1	R/W	0	0: Left Mixer Amplifier(MAL) is powered down

			1: Left Mixer Amplifier(MAL) is powered up
D0	R/W	0	0: Right Mixer Amplifier(MAR) is powered down 1: Right Mixer Amplifier(MAR) is powered up

- **Registro 12:** Selección ruta del HPL

Bit	Read/Write	Reset Value	Descripcion
D7-D4	R	0000	Reserved. Write only default values
D3	R/W	0	0: Left Channel DAC reconstruction filter's positive terminal is not routed to HPL 1: Left Channel DAC reconstruction filter's positive terminal is routed to HPL
D2	R/W	0	0: IN1L is not routed to HPL 1: IN1L is routed to HPL
D1	R/W	0	Ri 0: MAL output is not routed to HPL 1: MAL output is routed to HPL ght DAC Channel Mute Control 0: Right DAC Channel not muted 1: Right DAC Channel muted
D0	R/W	0	0: MAR output is not routed to HPL 1: MAR output is routed to HPL

- **Registro 13:** Selección ruta del HPR

Bit	Read/Write	Reset Value	Descripcion
D7-D5	R	000	Reserved. Write only default values
D4	R/W	0	0: Left Channel DAC reconstruction filter's negative terminal is not routed to HPR 1: Left Channel DAC reconstruction filter's negative terminal is routed to HPR
D3	R/W	0	0: Right Channel DAC reconstruction filter's positive terminal is not routed to HPR 1: Right Channel DAC reconstruction filter's positive terminal is routed to HPR
D2	R/W	0	0: IN1R is not routed to HPR 1: IN1R is routed to HPR
D1	R/W	0	0: MAR output is not routed to HPR 1: MAR output is routed to HPR
D0	R/W	0	0: HPL output is not routed to HPR 1: HPL output is routed to HPR (use when HPL&HPR output is powered by AVDD)

7.4 GNU LESSER GENERAL PUBLIC LICENCE

El código MS ADPCM que se aplica en el sistema final parte de unas librerías que son adquiridas a través de la entidad Sound Exchange. Las librerías que se han obtenido son “adpcm.c” y “adpmc.h”. Ambas han sido modificadas para funcionar en un DSP para cuatro canales en paralelo y a tiempo real. La licencia bajo la que se encuentra el software permite modificar y compartir de forma gratuita el código fuente.

GNU LESSER GENERAL PUBLIC LICENSE
Version 2.1, February 1999

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public Licenses are intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users.

This license, the Lesser General Public License, applies to some specially designated software packages--typically libraries--of the Free Software Foundation and other authors who decide to use it. You can use it too, but we suggest you first think carefully about whether this license or the ordinary General Public License is the better strategy to use in any particular case, based on the explanations below.

When we speak of free software, we are referring to freedom of use, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish); that you receive source code or can get it if you want it; that you can change the software and use pieces of it in new free programs; and that you are informed that you can do these things.

To protect your rights, we need to make restrictions that forbid distributors to deny you these rights or to ask you to surrender these rights. These restrictions translate to certain responsibilities for you if you distribute copies of the library or if you modify it.

For example, if you distribute copies of the library, whether gratis or for a fee, you must give the recipients all the rights that we gave you. You must make sure that they, too, receive or can get the source code. If you link other code with the library, you must provide complete object files to the recipients, so that they can relink them with the library after making changes to the library and recompiling it. And you must show them these terms so they know their rights.

We protect your rights with a two-step method: (1) we copyright the library, and (2) we offer you this license, which gives you legal permission to copy, distribute and/or modify the library.

To protect each distributor, we want to make it very clear that there is no warranty for the free library. Also, if the library is modified by someone else and passed on, the recipients should know that what they have is not the original version, so that the original author's reputation will not be affected by problems that might be introduced by others.

Finally, software patents pose a constant threat to the existence of any free program. We wish to make sure that a company cannot effectively restrict the users of a free program by obtaining a restrictive license from a patent holder. Therefore, we insist that any patent license obtained for a version of the library must be consistent with the full freedom of use specified in this license.

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License. This license, the GNU Lesser General Public License, applies to certain designated libraries, and is quite different from the ordinary General Public License. We use this license for certain libraries in order to permit linking those libraries into non-free programs.

When a program is linked with a library, whether statically or using a shared library, the combination of the two is legally speaking a combined work, a derivative of the original library. The ordinary General Public License therefore permits such linking only if the entire combination fits its criteria of freedom. The Lesser General Public License permits more lax criteria for linking other code with the library.

We call this license the "Lesser" General Public License because it does Less to protect the user's freedom than the ordinary General Public License. It also provides other free software developers less of an advantage over competing non-free programs. These disadvantages are the reason we use the ordinary General Public License for many libraries. However, the Lesser license provides advantages in certain special circumstances.

For example, on rare occasions, there may be a special need to encourage the widest possible use of a certain library, so that it becomes a de-facto standard. To achieve this, non-free programs must be allowed to use the library. A more frequent case is that a free library does the same job as widely used non-free libraries. In this case, there is little to gain by limiting the free library to free software only, so we use the Lesser General Public License.

In other cases, permission to use a particular library in non-free programs enables a greater number of people to use a large body of free software. For example, permission to use the GNU C Library in non-free programs enables many more people to use the whole GNU operating system, as well as its variant, the GNU/Linux operating system.

Although the Lesser General Public License is Less protective of the users' freedom, it does ensure that the user of a program that is linked with the Library has the freedom and the wherewithal to run that program using a modified version of the Library.

The precise terms and conditions for copying, distribution and modification follow. Pay close attention to the difference between a "work based on the library" and a "work that uses the library". The former contains code derived from the library, whereas the latter must be combined with the library in order to run.

GNU LESSER GENERAL PUBLIC LICENSE
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License Agreement applies to any software library or other program which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Lesser General Public License (also called "this License"). Each licensee is addressed as "you".

A "library" means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables.

The "Library", below, refers to any such software library or work which has been distributed under these terms. A "work based on the Library" means either the Library or any derivative work under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term "modification".)

"Source code" for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

1. You may copy and distribute verbatim copies of the Library's complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the Library.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) The modified work must itself be a software library.

b) You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.

c) You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.

d) If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that use the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful. (For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the application. Therefore, Subsection 2d requires that any application-supplied function or table used by this function must be optional: if the application does not supply it, the square root function must still compute square roots.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Library, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Library, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Library.

In addition, mere aggregation of another work not based on the Library with the Library (or with a work based on the Library) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. To do this, you must alter all

the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2, instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices.

Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy.

This option is useful when you wish to copy part of the code of the Library into a program that is not a library.

4. You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange.

If distribution of object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place satisfies the requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object code.

5. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a "work that uses the Library". Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

However, linking a "work that uses the Library" with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a "work that uses the library". The executable is therefore covered by this License. Section 6 states terms for distribution of such executables.

When a "work that uses the Library" uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length),

then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.)

Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.

6. As an exception to the Sections above, you may also combine or link a "work that uses the Library" with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer's own use and reverse engineering for debugging such modifications.

You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

a) Accompany the work with the complete corresponding achine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable "work that uses the Library", as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)

b) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (1) uses at run time a copy of the library already present on the user's computer system, rather than copying library functions into the executable, and (2) will operate properly with a modified version of the library, if the user installs one, as long as the modified version is interface-compatible with the version that the work was made with.

c) Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.

d) If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.

e) Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

For an executable, the required form of the "work that uses the Library" must include any data and utility programs needed for reproducing the executable from it. However, as a special exception, the materials to be distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

7. You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:

a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.

b) Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

8. You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Library is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

9. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Library or

its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.

10. Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties with this License.

11. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all. For example, if a patent license would not permit royalty-free redistribution of the Library by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Library.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply, and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

12. If the distribution and/or use of the Library is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Library under this License may add an explicit geographical distribution

limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

13. The Free Software Foundation may publish revised and/or new versions of the Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Library does not specify a license version number, you may choose any version ever published by the Free Software Foundation.

14. If you wish to incorporate parts of the Library into other free programs whose distribution conditions are incompatible with these, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free
Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

15. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE LIBRARY "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE

LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

7.5 Descripción entregables

A lo largo del desarrollo del proyecto y de cara a satisfacer las dos entregas planificadas con el cliente, se han generado los siguientes entregables:
Entrega 1:

- Conexionado de micrófonos y configuración de la tarjeta de evaluación con un ordenador personal
- Informe técnico de conexionado del chip de codificación de audio y del chip DSP
- Informe técnico de conexionado del chip de codificación de audio y los micrófonos analógicos y digitales

Entrega 2:

- Herramientas y documentación del desarrollo de algoritmos de tratamiento de audio, depuración e integración en el DSP
- Requisitos diseño HW

PRESUPUESTO

1. Ejecución Material

➢ Compra de ordenador personal (Software incluido)	1000 €
➢ Micrófonos digitales *SPM0405HD4H-WB*	2,6 €
➢ Micrófonos analógicos *SPM0408HE5H*	2,3 €
➢ CODEC TLV320AIC34 EVM-K	149 €
➢ DSP TMS320V5505 eZdsp USB STICK	49 €
➢ Material Conexiones	50 €
➢ Material de oficina	80 €
➢ Total de ejecución material	1332,9 €

2. Ejecución Material

➢ 16 % sobre Ejecución Material	213,3 €

3. Beneficio Industrial

➢ 6 % sobre Ejecución Material	80 €

4. Honorarios Proyecto

➢ 640 horas a 15 € / hora	9600 €

5. Material fungible

➢ Gastos de impresión	100 €
➢ Encuadernación	50 €

6. Subtotal del presupuesto

➢ Subtotal Presupuesto	11376 €

7. I.V.A. aplicable

➢ 18% Subtotal Presupuesto	2048 €

8. Total presupuesto

➢ Total Presupuesto	13424 €

Madrid, Septiembre 2011

El Ingeniero Jefe de Proyecto Fdo.: Mario Gutiérrez Herzing, Ingeniero Superior de Telecomunicación

PLIEGO DE CONDICIONES

Este documento contiene las condiciones legales que guiarán la realización, en este proyecto, de un sistema de captura y codificación de audio. En lo que sigue, se supondrá que el proyecto ha sido encargado por una empresa cliente a una empresa consultora con la finalidad de realizar dicho sistema. Dicha empresa ha debido desarrollar una línea de investigación con objeto de elaborar el proyecto. Esta línea de investigación, junto con el posterior desarrollo de los programas está amparada por las condiciones particulares del siguiente pliego.

Supuesto que la utilización industrial de los métodos recogidos en el presente proyecto ha sido decidida por parte de la empresa cliente o de otras, la obra a realizar se regulará por las siguientes:

Condiciones generales

1. La modalidad de contratación será el concurso. La adjudicación se hará, por tanto, a la proposición más favorable sin atender exclusivamente al valor económico, dependiendo de las mayores garantías ofrecidas. La empresa que somete el proyecto a concurso se reserva el derecho a declararlo desierto.

2. El montaje y mecanización completa de los equipos que intervengan será realizado totalmente por la empresa licitadora.

3. En la oferta, se hará constar el precio total por el que se compromete a realizar la obra y el tanto por ciento de baja que supone este precio en relación con un importe límite si este se hubiera fijado.

4. La obra se realizará bajo la dirección técnica de un Ingeniero Superior de Telecomunicación, auxiliado por el número de Ingenieros Técnicos y Programadores que se estime preciso para el desarrollo de la misma.

5. Aparte del Ingeniero Director, el contratista tendrá derecho a contratar al resto del personal, pudiendo ceder esta prerrogativa a favor del Ingeniero Director, quien no estará obligado a aceptarla.

6. El contratista tiene derecho a sacar copias a su costa de los planos, pliego de condiciones y presupuestos. El Ingeniero autor del proyecto autorizará con su firma las copias solicitadas por el contratista después de confrontarlas.

7. Se abonará al contratista la obra que realmente ejecute con sujeción al proyecto que sirvió de base para la contratación, a las modificaciones autorizadas por la superioridad o a las órdenes que con arreglo a sus facultades le hayan comunicado por escrito al Ingeniero Director de obras siempre que dicha obra se haya ajustado a los preceptos de los pliegos de condiciones, con arreglo a los cuales, se harán las modificaciones y la valoración de las diversas unidades sin que el importe total pueda exceder de los presupuestos aprobados. Por consiguiente, el número de unidades que se

consignan en el proyecto o en el presupuesto, no podrá servirle de fundamento para entablar reclamaciones de ninguna clase, salvo en los casos de rescisión.

8. Tanto en las certificaciones de obras como en la liquidación final, se abonarán los trabajos realizados por el contratista a los precios de ejecución material que figuran en el presupuesto para cada unidad de la obra.

9. Si excepcionalmente se hubiera ejecutado algún trabajo que no se ajustase a las condiciones de la contrata pero que sin embargo es admisible a juicio del Ingeniero Director de obras, se dará conocimiento a la Dirección, proponiendo a la vez la rebaja de precios que el Ingeniero estime justa y si la Dirección resolviera aceptar la obra, quedará el contratista obligado a conformarse con la rebaja acordada.

10. Cuando se juzgue necesario emplear materiales o ejecutar obras que no figuren en el presupuesto de la contrata, se evaluará su importe a los precios asignados a otras obras o materiales análogos si los hubiere y cuando no, se discutirán entre el Ingeniero Director y el contratista, sometiéndolos a la aprobación de la Dirección. Los nuevos precios convenidos por uno u otro procedimiento, se sujetarán siempre al establecido en el punto anterior.

11. Cuando el contratista, con autorización del Ingeniero Director de obras, emplee materiales de calidad más elevada o de mayores dimensiones de lo estipulado en el proyecto, o sustituya una clase de fabricación por otra que tenga asignado mayor precio o ejecute con mayores dimensiones cualquier otra parte de las obras, o en general, introduzca en ellas cualquier modificación que sea beneficiosa a juicio del Ingeniero Director de obras, no tendrá derecho sin embargo, sino a lo que le correspondería si hubiera realizado la obra con estricta sujeción a lo proyectado y contratado.

12. Las cantidades calculadas para obras accesorias, aunque figuren por partida alzada en el presupuesto final (general), no serán abonadas sino a los precios de la contrata, según las condiciones de la misma y los proyectos particulares que para ellas se formen, o en su defecto, por lo que resulte de su medición final.

13. El contratista queda obligado a abonar al Ingeniero autor del proyecto y director de obras así como a los Ingenieros Técnicos, el importe de sus respectivos honorarios facultativos por formación del proyecto, dirección técnica y administración en su caso, con arreglo a las tarifas y honorarios vigentes.

14. Concluida la ejecución de la obra, será reconocida por el Ingeniero Director que a tal efecto designe la empresa.

15. La garantía definitiva será del 4% del presupuesto y la provisional del 2%.

16. La forma de pago será por certificaciones mensuales de la obra ejecutada, de acuerdo con los precios del presupuesto, deducida la baja si la hubiera.

17. La fecha de comienzo de las obras será a partir de los 15 días naturales del replanteo oficial de las mismas y la definitiva, al año de haber ejecutado la provisional, procediéndose si no existe reclamación alguna, a la reclamación de la fianza.

18. Si el contratista al efectuar el replanteo, observase algún error en el proyecto, deberá comunicarlo en el plazo de quince días al Ingeniero Director de obras, pues transcurrido ese plazo será responsable de la exactitud del proyecto.

19. El contratista está obligado a designar una persona responsable que se entenderá con el Ingeniero Director de obras, o con el delegado que éste designe, para todo relacionado con ella. Al ser el Ingeniero Director de obras el que interpreta el proyecto, el contratista deberá consultarle cualquier duda que surja en su realización.

20. Durante la realización de la obra, se girarán visitas de inspección por personal facultativo de la empresa cliente, para hacer las comprobaciones que se crean oportunas. Es obligación del contratista, la conservación de la obra ya ejecutada hasta la recepción de la misma, por lo que el deterioro parcial o total de ella, aunque sea por agentes atmosféricos u otras causas, deberá ser reparado o reconstruido por su cuenta.

21. El contratista, deberá realizar la obra en el plazo mencionado a partir de la fecha del contrato, incurriendo en multa, por retraso de la ejecución siempre que éste no sea debido a causas de fuerza mayor. A la terminación de la obra, se hará una recepción provisional previo reconocimiento y examen por la dirección técnica, el depositario de efectos, el interventor y el jefe de servicio o un representante, estampando su conformidad el contratista.

22. Hecha la recepción provisional, se certificará al contratista el resto de la obra, reservándose la administración el importe de los gastos de conservación de la misma hasta su recepción definitiva y la fianza durante el tiempo señalado como plazo de garantía. La recepción definitiva se hará en las mismas condiciones que la provisional, extendiéndose el acta correspondiente. El Director Técnico propondrá a la Junta Económica la devolución de la fianza al contratista de acuerdo con las condiciones económicas legales establecidas.

23. Las tarifas para la determinación de honorarios, reguladas por orden de la Presidencia del Gobierno el 19 de Octubre de 1961, se aplicarán sobre el denominado en la actualidad “Presupuesto de Ejecución de Contrata” y anteriormente llamado ”Presupuesto de Ejecución Material” que hoy designa otro concepto.

Condiciones particulares

La empresa consultora, que ha desarrollado el presente proyecto, lo entregará a la empresa cliente bajo las condiciones generales ya formuladas, debiendo añadirse las siguientes condiciones particulares:

1. La propiedad intelectual de los procesos descritos y analizados en el presente trabajo, pertenece por entero a la empresa consultora representada por el Ingeniero Director del Proyecto.

2. La empresa consultora se reserva el derecho a la utilización total o parcial de los resultados de la investigación realizada para desarrollar el siguiente proyecto, bien para su publicación o bien para su uso en trabajos o proyectos posteriores, para la misma empresa cliente o para otra.

3. Cualquier tipo de reproducción aparte de las reseñadas en las condiciones generales, bien sea para uso particular de la empresa cliente, o para cualquier otra aplicación, contará con autorización expresa y por escrito del Ingeniero Director del Proyecto, que actuará en representación de la empresa consultora.

4. En la autorización se ha de hacer constar la aplicación a que se destinan sus reproducciones así como su cantidad.

5. En todas las reproducciones se indicará su procedencia, explicitando el nombre del proyecto, nombre del Ingeniero Director y de la empresa consultora.

6. Si el proyecto pasa la etapa de desarrollo, cualquier modificación que se realice sobre él, deberá ser notificada al Ingeniero Director del Proyecto y a criterio de éste, la empresa consultora decidirá aceptar o no la modificación propuesta.

7. Si la modificación se acepta, la empresa consultora se hará responsable al mismo nivel que el proyecto inicial del que resulta el añadirla.

8. Si la modificación no es aceptada, por el contrario, la empresa consultora declinará toda responsabilidad que se derive de la aplicación o influencia de la misma.

9. Si la empresa cliente decide desarrollar industrialmente uno o varios productos en los que resulte parcial o totalmente aplicable el estudio de este proyecto, deberá comunicarlo a la empresa consultora.

10. La empresa consultora no se responsabiliza de los efectos laterales que se puedan producir en el momento en que se utilice la herramienta objeto del presente proyecto para la realización de otras aplicaciones.

11. La empresa consultora tendrá prioridad respecto a otras en la elaboración de los proyectos auxiliares que fuese necesario desarrollar para dicha aplicación industrial, siempre que no haga explícita renuncia a este hecho. En este caso, deberá autorizar expresamente los proyectos presentados por otros.

12. El Ingeniero Director del presente proyecto, será el responsable de la dirección de la aplicación industrial siempre que la empresa consultora lo estime oportuno. En caso contrario, la persona designada deberá contar con la autorización del mismo, quien delegará en él las responsabilidades que ostente.