Índice:

Índice de figuras:

Índice de tablas:

CAPÍTULO 1: INTRODUCCIÓN.

1.1. BREVE HISTORIA DE LOS DSP.

El procesado digital de señal se refiere al análisis matemático, mediante herramientas digitales, de unas señales analógicas reales. Este análisis puede ser para estudiar como se comportan estas señales, o para hacer una simulación de cómo actuarían las señales analógicas ante diversos sistemas, y así poder prever una salida.

Curiosamente en los años 50 y 60 el desarrollo del procesado digital de señal llegó a causa de la necesidad de simular el comportamiento de sistemas analógicos, antes de llevar a cabo costosos prototipos. Como era de esperar, la herramienta de simulación fue un PC. Este fue el inicio del desarrollo del procesado digital de señal.

Posteriormente, en los años 80 se empezó a buscar una arquitectura de procesador más eficaz para implementar los algoritmos ya desarrollados en los años 50 y 60. En ese momento se empiezan a usar computadores digitales para procesar señales analógicas reales.

Los algoritmos para DSP se basan en un modelo matemático básico para las señales continuas. Este modelo matemático esta basado en las transformadas de Fourier y Laplace. Más recientes, son los desarrollos de algoritmos difusos o neuronales. Por ejemplo, la transformada Z, basada en la transformada de Laplace, es el pilar básico de la construcción de filtros digitales.

Las arquitecturas usadas para DSP provienen de la arquitectura tipo Harvard para procesadores y microprocesadores. Su característica principal es la separación en dos de la memoria para datos y programa. Esta arquitectura no se desarrolló con mucho éxito al inicio de los procesadores, siendo desbancada por la arquitectura de Von Neuman.

La arquitectura Von Neuman simplifica el diseño del computador al entender un solo espacio de memoria, y dividir las instrucciones en código de operación y dirección del operando. Pero esto hacia a operaciones como la multiplicación extremadamente lentas. A los dispositivos con esta arquitectura se les conocen como dispositivos CISC (*Complex Instruction Set Computers*), y ejecutan las operaciones complejas como la multiplicación

con una cadena de operaciones sencillas como la suma, siendo este el motivo de la lentitud de ejecución de una multiplicación.

El procesado digital tiene como objetivo principal: procesar las señales en tiempo real y completar todas las operaciones antes de la llegada de un nuevo dato. En los DSP antiguos mediante el uso de la arquitectura de Von Neuman, se comprobó que el factor que relentizaba todo el procesado era la multiplicación. Se avanzó en su diseño mediante técnicas de procesado paralelo, lográndose el primer multiplicador en un ciclo de reloj en los años 70.

Mediante la arquitectura de Von Neuman se lograron procesadores como EL FDP (*Fast Digital Processor*) de Lincoln, que conseguía ejecutar una multiplicación en 600ns, pero eran muy complejos y necesitaban mucho espacio.

El LSP/2 de Lincoln ya se decantó por usar arquitectura Harvard a mediados de los 70, logrando cuatro veces más velocidad con un tercio de dimensiones, sentando así una de las bases de la arquitectura de los procesadores de señal, el uso de una arquitectura Harvard.

Con la llegada de los CI en los años 70, se lograron reducir sobre todo tamaños en los diseños de los DSP ya existentes.

Posteriormente en los años 80 se crearon procesadores digitales monocircuito como el S2811 de AMI, el Intel 2920, el NEC mPD7720, y el TMS32010 de Texas Instrument con el que realmente llegó el DSP monopastilla.

Ya usan la arquitectura Harvard y logran separar la memoria de datos de la memoria de programa, permitiendo acceso simultáneo a instrucción de programa y datos. El punto crítico es que el flujo de datos no necesita interrumpirse a causa de la lectura de instrucciones.

Se añadieron modificaciones posteriores a la arquitectura Harvard como unir las dos memorias, siendo esta la arquitectura Harvard modificada, muy usada actualmente.

Los computadores y la industria de los ordenadores personales ha avanzado enormemente desde entonces y los DSP no han sido una excepción. Se ha reducido la geometría de los CI y aumentado la densidad de transistores, por tanto se produjo un aumento de las velocidades de reloj y de las potencias de cálculo. Actualmente existen DSPs con tiempos de multiplicación de palabras de 32 bits de 40 ns o inferiores.

1.2. OBJETIVOS.

El objetivo de este PFC es crear un curso para el desarrollo de aplicaciones de audio mediante un dsPIC, para ello se usa el lenguaje de programación de alto nivel C.

Como objetivos secundarios están aprender a usar la placa de desarrollo UIB-PC104, conocer las características principales de los PIC de Microchip, y del dsPIC30F4011 más concretamente, aprender a programar dicho dsPIC, y por las necesidades del PFC, crear un pequeño PCB, y por último, aprender a configurar un CI mediante un microcontrolador, en este caso, el Si3000 a través del dsPIC30F4011.

1.3. CONTENIDO.

La organización de la memoria del PFC consta de 5 capítulos:

- Capítulo 1. Introducción:

Hace una breve introducción a la historia de los DSP, describe los objetivos del PFC, y enumera los capítulos que constan en el PFC y el contenido de cada uno de ellos.

- Capítulo 2. El dsPIC30F4011:

Este segundo capítulo describe al bloque principal de la parte hardware del PFC. El dsPIC30F4011 es el dsPIC que posee la placa de desarrollo UIB-PC104. Este dsPIC es realmente un microcontrolador pero con funciones y bloques DSP añadidos.

Se detallan en este capítulo las familias de microcontroladores del fabricante Microchip, el dsPIC30F4011 mas a fondo, la placa de desarrollo UIB-PC104 y otras alternativas evaluadas del mercado de las placas de desarrollo.

- Capítulo 3. El Si3000.

Debido a la falta de un códec en la placa de desarrollo UIB-PC104, se necesito añadir uno a ella. Para ello, se hizo un estudio del mercado de los códec, se detalla el códec Si3000 elegido, y se detalla la PCB creada para añadir el Si3000.

- Capitulo 4. Prácticas.

Se llevaron a cabo 4 prácticas en el dsPIC30F4011:

- Práctica 1: Efectos de Audio.
- Práctica 2: Generación de ondas.
- Práctica 3: Filtros digitales.
- Práctica 4: Ecualizador.

A parte de las prácticas se añadió un menú para configurar los parámetros más interesantes del Si3000 en tiempo de ejecución. En algunas prácticas se añade otro menú para configurar parámetros de las aplicaciones, como en la reverberación y el ecualizador.

Cada práctica se inicia con una pequeña parte de teoría, una explicación de la solución en el dsPIC30F4l011, y como parte más importante las limitaciones de usar un dsPIC para hacer cada una de las prácticas.

❖ Capítulo 5. Conclusiones.

En este capítulo se enumeran las conclusiones y las posibles líneas futuras de este PFC. Las líneas futuras se dividen en las mejoras del sistema UIB-PC104 junto al Si3000, y en las mejoras obtenidas si se usase otra placa de desarrollo y no la UIB-PC104.

CAPÍTULO 2: EL DSPIC30F4011.

2.1. INTRODUCCIÓN.

En el PFC que nos ocupa, el microcontrolador es una parte básica, si no la más importante. Es el motor de todo el procesado a realizar, de toda la comunicación con el códec de audio, en definitiva, es el eje del PFC aunque no se use el 100 % de su capacidad.

Por ello es necesario conocer qué es un microcontrolador, cuáles son las características de las familias más importantes de la marca usada, en este caso Microchip, y cuáles son más a fondo las características del microcontrolador elegido, que venía incorporado a la placa de desarrollos UIB-PC104 y se trata del dsPIC30F4011.

Como se trata de un microcontrolador especial al poseer características de DSP, se debe conocer también qué es un DSP, y por qué éste es más adecuado para el procesado digital que interesa en este PFC.

Por usar una placa de desarrollo y no directamente el microcontrolador por comodidad y facilidad de diseñar las aplicaciones, también es necesario conocer qué alternativas ofrece el mercado, cuáles son las características de la placa de desarrollo que poseemos, en concreto la UIB-PC104.

2.2. MICROCONTROLADORES.

2.2.1. ¿QUÉ ES UN MICROCONTROLADOR?

Los microcontroladores aparecen en los años 80. Se tratan básicamente de un circuito integrado programable, que posee una estructura de una microcomputadora. Con esto, se puede imaginar que dentro de un microcontrolador se encontrarán los siguientes bloques: CPU, Memoria RAM, Memoria ROM, Memoria EEPROM, Puertos E/S.

Hoy día se añaden otros módulos como conversores analógico/digital, módulos PWM y módulos para comunicaciones serie y paralelo, entre muchos otros módulos, dependiendo de la complejidad del microcontrolador, como por ejemplo, un motor DSP, para hacer procesamiento digital de señal.

Cada vez son más los productos que usan microcontroladores para aumentar sus prestaciones, reducir el tamaño y coste, así como reducir el consumo de energía.

Las aplicaciones donde se usan los microcontroladores son ilimitadas, puesto que, hay una variedad de microcontroladores muy amplia, desde unos sencillos y económicos hasta otros complejos y caros.

Teniendo en cuenta la gran variedad de microcontroladores que nos ofrece el mercado, lo realmente complicado no es usar un microcontrolador, sino elegir el adecuado para nuestra aplicación. Esto es debido a la gran cantidad de periféricos, módulos o utilidades que puede añadir un microcontrolador, o simplemente a la cantidad de algunas de esas características, como por ejemplo, la cantidad de memoria EEPROM, la cantidad de bloques para usar una interfaz digital mediante el bus SPI.

Por ejemplo, en la aplicación de este PFC, se usó un microcontrolador de la familia PIC de Microchip pero más concretamente un dsPIC30F4011, que posee el módulo de comunicación serie para comunicarnos con un códec de voz, y a su vez, el motor DSP necesario para el posterior procesado de muestras de audio.

2.2.2. ¿QUÉ ES UN DSP?

DSP proviene del inglés y su traducción es (*Digital Signal Processor)*, que significa Procesador Digital de Señal.

Un DSP, como cualquier otro tipo de microcontrolador, es un sistema basado en un microprocesador, que posee un juego de instrucciones, un hardware y un software preparado para aplicaciones, que requieran operaciones numéricas a muy alta velocidad.

Es especialmente útil para el procesado y representación de señales analógicas en tiempo real como el audio, y aplicaciones como la compresión de voz en telefonía móvil, filtros complejos de sonido, líneas de retardo, generadores de eco, decodificación de canales en telefonía celular (GSM)…

Como se ha comentado, normalmente trabaja con señales analógicas, pero es un sistema digital, por lo que necesitará un conversor analógico/digital a su entrada, y otro digital/analógico en la salida. Como todo sistema basado en un procesador programable necesita una memoria donde almacenar los datos con los que trabajará y el programa que ejecuta.

Si se tiene en cuenta que un DSP puede trabajar con varios datos en paralelo y que posee un diseño e instrucciones específicas para el procesado digital de señal, se puede apreciar la enorme potencia para este tipo de aplicaciones. Estas características, constituyen la principal diferencia de un DSP y otros tipos de procesadores.

Las aplicaciones básicas de un DSP son todas aquellas que requieran hacer procesado de señales analógicas en tiempo real, por ejemplo, como se dijo antes las señales de video y voz, y cualquier tipo de procesado con ellas.

Un DSP se puede programar tanto en ensamblador, como en lenguaje C, siendo como es de esperar, más eficiente el lenguaje ensamblador, y más sencillo de programar el lenguaje C. Cada familia de DSP tiene su propio lenguaje ensamblador y sus propias herramientas suministradas por el fabricante.

En la figura 2.1 se muestra un ejemplo del flujo de datos analógicos que se procesan en el DSP.

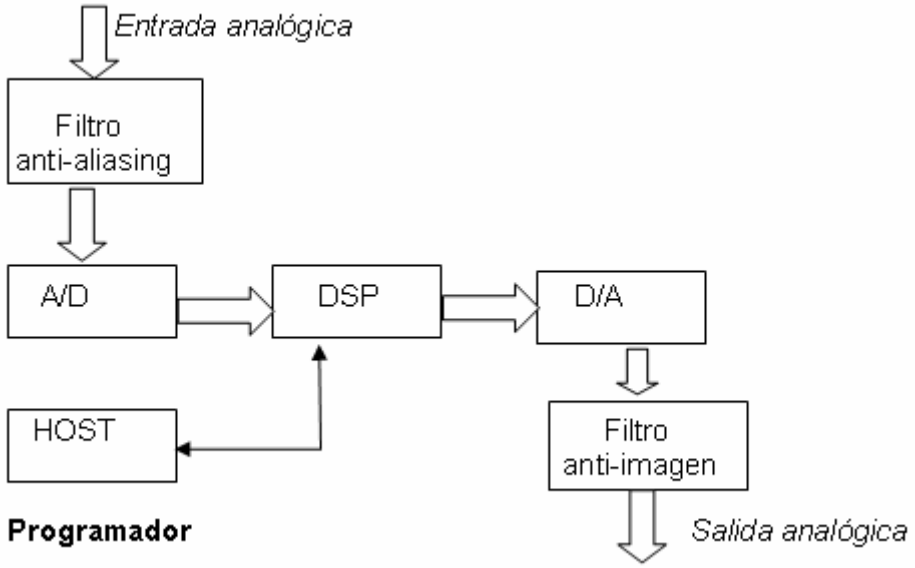

Figura 2.1. Flujo de la señal analógica procesada [6].

2.2.3. ESTRUCTURA DE LOS DSP.

Un DSP está diseñado teniendo en cuenta las tareas más habituales del procesado digital: sumas, multiplicaciones y retrasos (almacenar en memoria). Para ello cuentan con una arquitectura Harvard.

La arquitectura Harvard, consiste básicamente en la división física de las memorias de datos y memoria de programa, y se empezó a usar debido a que las CPU son más rápidas que las memorias y así poder reducir el número de accesos a memoria.

Desde el punto de vista de la arquitectura interna, se puede decir, que un DSP es un microcontrolador optimizado internamente, para realizar los cálculos necesarios en la implementación de algoritmos de procesado de señal. Esta optimización se consigue con la arquitectura Harvard y otros aspectos principales como:

- Implementación de operaciones por hardware.
- Instrucciones poco comunes que ejecutan varias operaciones en un solo ciclo.
- Modos de direccionamiento especiales.
- Memoria de programa ancha, con más de 8 bits.
- Pueden manejar números con coma flotante.

Los elementos básicos que componen un DSP son:

- Conversores en las entradas y salidas
- Memoria de datos, memoria de programa y DMA.
- Desplazador de barril.
- MACs: multiplicadores y acumuladores.
- ALU: Unidad aritmético-lógica.
- Registros.

2.3. MICROCONTROLADORES DE MICROCHIP.

Microchip es un gran proveedor de semiconductores, con sus líneas de productos analógicos, microcontroladores y memorias EEPROM. Su sede esta en Chandler, Arizona, y fue fundada en 1989. Ahora tiene 4600 empleados en el planeta y más de 45 oficinas de ventas, así como lugares de manufactura y diseño desde América (Arizona) hasta Asia (Tailandia e India) pasando por Europa (Suiza).

En el año 2006 logró el liderazgo mundial de venta de microcontroladores de 8 bits, los famosos PIC (*Peripheral Interface Controller*), desbancando a Motorola, ahora llamado Freescale [11].

Pero no solo se ha limitado a los microcontroladores de 8 bits, aunque en estos ya se logró gran capacidad de cálculo, sino que lanzó tanto microcontroladores de 16 y de 32 bits. Además, con los microcontroladores de 16 bits añadió un nuevo concepto, los DSC (*Digital Signal Controller*) que se detallarán más adelante y que básicamente es una combinación de las mejores características de los DSP y los microcontroladores.

La última tecnología anunciada por Microchip son unos microcontroladores USB, tanto de 8, 16 y 32 bits. Estos se basan en los antiguos PIC18 para los nuevos microcontroladores de 8 bits, y para 16 bits han hecho unos nuevos los PIC24F USB. Ambos son compatibles con la familia de microcontroladores USB PIC32 de 32 bits.

2.3.1. FAMILIAS DE 8 BITS.

Con 8 bits, Microchip desarrolló 5 familias, cada una de ellas, es en base la anterior mejorada. Serán expuestas a continuación para ver la evolución que sufrieron con el paso del tiempo, debido a la mejora de las tecnologías para diseñarlas y las necesidades del mercado.

Las 5 familias, como se ve en la figura 2.2, poseen cada vez mejor rendimiento y mejor funcionalidad, pero además, se aprecia en el hecho de que las familias están solapadas que cada familia tiene que ver con la anterior, es decir, mantiene bloques funcionales, o los mejora.

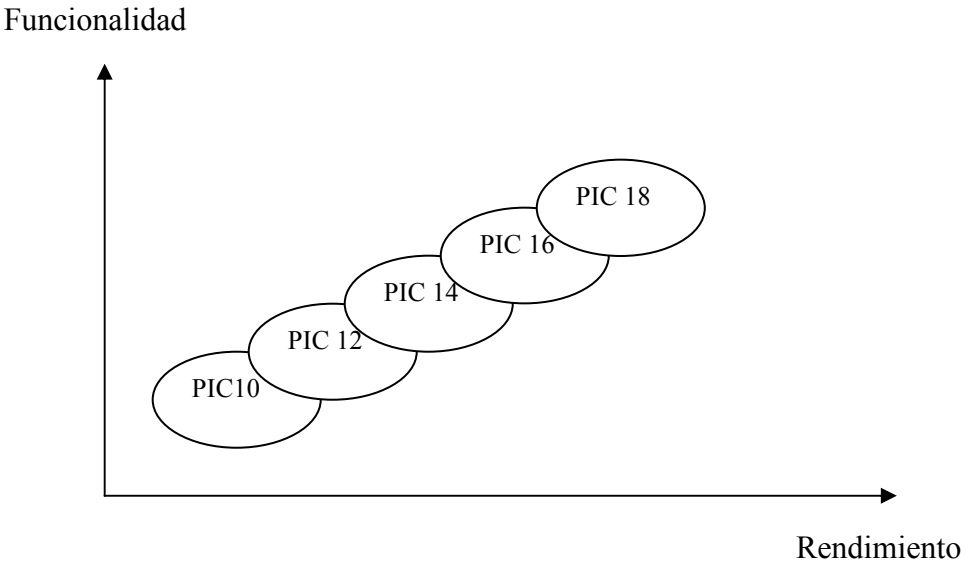

Figura 2.2. Las familias de 8 bits de Microchip.

2.3.1.1. PIC10 MCU.

Es la familia de más bajas prestaciones de Microchip, y por tanto la gama más económica, pero no es nada simple y sus 33 instrucciones lo convierten en un PIC sencillo para aprender.

Sus características principales son:

- Memoria Flash de programa entre 512 bytes y 1024 bytes.
- Memoria RAM de entre 16-24 bytes.
- Memoria EEPROM no posee.
- Velocidad de CPU de 4-8Mhz.
- Conversores A/D. No toda la familia.
- No poseen interfaces SPI (*Serial Peripheral Interface*), I2C (*Inter - Integrated Circuit Serial Port*), UART (*Universal Asinchronous Receiver Transmitter*), etc.
- Pines del encapsulado: 6 u 8 pines.
- Su precio no supera el medio dólar [19].

2.3.1.2. PIC12 MCU.

Microchip dio un pequeño avance en la complejidad del microcontrolador, pero básicamente solo se mejoran las características anteriores:

Sus características principales son:

- Memoria Flash de programa entre 1024 y 4096 bytes.
- Memoria RAM de entre 25-128 bytes.
- Memoria EEPROM entre 0 y 256 bytes.
- Velocidad de CPU de 4-20Mhz.
- 1 o 2 temporizadores de 8 bits e incluso a veces 1 temporizador de 16 bits.
- Conversores A/D. No toda la familia.
- No poseen interfaces SPI, I2C, UART, etc.
- Pines del encapsulado: 8 pines.
- Su precio no supera el dólar [19].

2.3.1.3. PIC14 MCU.

Solo existe un PIC en esta familia, el PIC14000. Este PIC ya posee avances notables y no se limita a mejorar las capacidades de la familia anterior al introducir cambios como la memoria OTP y la interfaz I2C.

Éstas son sus características principales:

- Memoria OTP (*One Time Programmable*) de programa hasta 7 Kbytes.
- Memoria RAM de entre 16-24 bytes.
- Memoria EEPROM no posee.
- Velocidad de CPU de 20Mhz.
- 1 temporizador de 8 bits y 1 temporizador de 16 bits.
- 8 Conversores A/D.
- Posee una interfaz I2C.
- Encapsulado de 28 pines.
- Su precio es de 5,50 dólares [19].

2.3.1.4. PIC16 MCU.

Es la familia de Microchip que mayor número de microcontroladores posee. Se pueden distinguir varias subfamilias: PIC16C, PIC16CR, PIC16F y PIC16HV. Cada una de ellas es especifica para unas especificaciones o poseen alguna peculiaridad, como el tipo de memoria de programa.

Estas son sus características principales:

- Memoria de programa OTP para la familia PIC16C, ROM para la familia PIC16CR y Flash para el resto de entre 0,75 y 14 Kbytes.
- Memoria RAM de entre 14-368 bytes.
- Memoria EEPROM entre 0 y 256 bytes.
- Velocidad de CPU de 20Mhz normalmente, con excepciones a 24 y 40 MHz.
- 1 o 2 temporizadores de 8 bits y 0 o 1 temporizador de 16 bits.
- Hasta 14 Conversores A/D. Algunos no poseen ninguno.

- Posee en algunos microcontroladores interfaz de comunicaciones A/E/USART o SSP que es un módulo con SPI e I2C, a veces posee las dos interfaces.
- El encapsulado tiene una variedad extensa en cuanto a número de pines como tipo de encapsulado.
- Su precio no es superior a 3,5 dólares, pero muchos microcontroladores no superan el dólar.
- Las tensiones con que trabaja estas familias sufren modificaciones, se reduce en la familia PIC16LF a los 3,6 V y se incrementa a 15V en la familia PIC16HV.
- Algunos PICs ya empiezan a ser reprogramables [19].

2.3.1.5. PIC18 MCU.

Con estos PIC, Microchip llega al final de la evolución de los PIC de 8 bits, el tamaño de la instrucción ya posee 16 bits, la arquitectura es de altas prestaciones, ofrece un compilador C eficiente.

Sus características principales son las siguientes:

- Memoria de programa OTP para la familia PIC18C, y para la familia PIC18F usa Flash de entre 4 y 128 Kbytes.
- Memoria RAM de hasta 3968 bytes.
- Memoria EEPROM entre 0 y 1024 bytes.
- Velocidad de CPU de 25, 40, 42,48 e incluso 64MHz.
- 1 o 2 temporizadores de 8 bits y 3 temporizadores de 16 bits.
- Hasta 16 Conversores A/D.
- Posee interfaz de comunicaciones A/E/USART o MSSP, es un módulo con SPI e I2C, a veces posee las dos interfaces incluso por partida doble.
- El encapsulado tiene una variedad extensa en cuanto número de pines como tipo de encapsulado, esto se debe a la gran variedad de la familia.
- Su precio no es superior a 3,5 dólares, pero muchos microcontroladores no superan el dólar.
- Las tensiones con que trabaja esta familia es 5,5 V o 3,6 V.
- La gran mayoría de los PIC de la familia son reprogramables [19].

2.3.2. FAMILIAS DE 16 BITS.

Como sucediera con la familia de 8 bits, Microchip desarrollo varias familias de 16 bits como se observa en la figura 2.3.

Microchip, tras el éxito de los PIC de 8 bits, sigue con su evolución hacia los microcontroladores de 16 bits y esto conlleva una potencia de cálculo mayor a la generación de 8 bits.

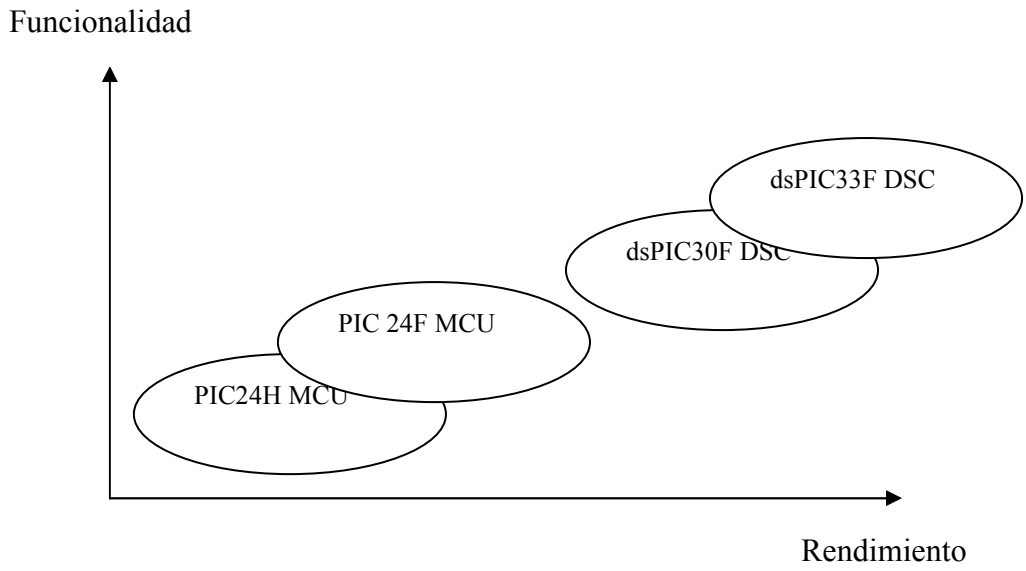

Figura 2.3. Las familias de 16 bits de Microchip.

2.3.2.1. PIC24 MCU.

Se mejora notablemente sobre todo la frecuencia de trabajo, se empieza a usar el bus CAN *(Controller Area Network*), se añaden novedades como el CRC (*Cyclic Redundancy Check*), los comparadores, puerto USB, temporizadores de 16 bits. Sobre todo ahora las interfaces de comunicación están presentes en todos los modelos, ya que se hacen imprescindibles.

Estos PIC incrementan las prestaciones, pero no sacrifican la flexibilidad de interrupciones, su sencillo uso ni la eficiencia del código C.

Se dividen en dos familias que difieren en las prestaciones, siendo la gama PIC24H la gama alta de los PIC de 16 bits.

Se detallan a continuación, las características de las dos familias de 16 bits que no poseen motor DSP, estas son las familias PIC24F MCU y PIC24H MCU.

2.3.2.1.1. PIC 24F MCU.

Características principales de la familia PIC24F MCU:

- Arquitectura de 16 bits.
- Velocidad de CPU de 16MIPS (millones de instrucciones por segundo), con dos osciladores de 8Mhz y 32Khz.
- Memoria de programa tipo FLASH de entre 16 y 256 Kbytes.
- Memoria RAM de entre 4096 y 16384 bytes.
- ALU *(Arithmetic Logic Unit*) de 16 bits.
- Multiplicador de 17 x 17 bits.
- Array de registros de 16 x 16 bits.
- Unidad generadora de direcciones.
- Tensión de trabajo de entre 2 V y 3.6 V.
- 2 o 3 comparadores.
- 1 conversor A/D de 16 entradas x 10 bits.
- 5 temporizadores de 16 bits.
- Algunos microcontroladores incorporan interfaz USB.
- CRC Programable.
- Interfaces de comunicacionais SPI , I2C y UART.
- Transceptores LIN, IrDA (*Infrared Data Association*) para algunos microcontroladores.
- Hardware RTCC (*Hardware Real-Time Clock/Calendar*).
- Puerto paralelo PMP (*Parallel Master Port*).
- JTAG – *Boundary Scan*.
- ICSP. (*In-Circuit Serial Programming*).
- Su precio ronda entre los 1,66 y 4,60 dólares [20].

2.3.2.1.2 PIC24H MCU.

Características principales de la familia PIC24H MCU:

- Arquitectura de 16 bits.
- Velocidad de hasta 40MIPS, con dos osciladores de 7,37Mhz y 512Khz.
- Memoria de programa tipo FLASH de entre 12 y 256 Kbytes.
- Memoria RAM de entre 1024 y 16384 bytes.
- ALU de 16 bits.
- Multiplicador de 17 x 17 bits.
- Array de registros de 16 x 16 bits.
- Unidad generadora de direcciones.
- Tensión de trabajo de entre 3 V y 3.6 V.
- 0 o 2 comparadores.
- Hasta 8 canales DMA (*Direct Memory Access)*, es una gran novedad.
- Añade bloques PWM (*Pulse-Width Modulation*), pudiendo haber hasta 2 bloques de 16 bits de resolución cada uno.
- 1 conversor A/D de 32 entradas x 12 bits.
- Hasta 9 temporizadores de 16 bits y 4 de 32 bits.
- Algunos microcontroladores incorporan interfaz CAN, es la mayor novedad.
- CRC en algunos microcontroladores de la família.
- Interfaces de comunicaciones SPI , I2C y UART.
- IrDA para algunos microcontroladores.
- Hardware RTCC.
- Puerto paralelo PMP o GPIO (*General Purpose Input/Output*).
- JTAG – *Boundary Scan*.
- ICSP.
- Su precio ronda entre los 1,99 y 5,1 dólares [20].

Como se ha podido apreciar, entre estas dos familias, aunque poseen muchas similitudes, hay multitud de diferencias. Sobre todo en familia PIC24H MCU, por ser mucho más completa y potente que la familia PIC24F MCU. Aunque hoy día, esa

diferencia no se aprecie económicamente, debido a ser una tecnología con unos cuantos años y una fase de fabricación similar.

2.3.2.2. dsPIC30F DSC y dsPIC33F DSC.

Tras el gran éxito de las familias PIC24 de 16 bits, Microchip marcó una nueva línea de trabajo: mejorar las características de sus PIC en el cada vez más usado mundo del procesado digital de señal, así se puede mejorar el procesado de señales comunes digitales como el audio.

Se prestará más atención a estas dos familias de 16 bits, se debe a su novedad en el mercado ya que como se ha comentado, se añadió un DSP a un microcontrolador, y por ser la familia que se usa en nuestra aplicación.

Un controlador digital de señal (DSC) es un controlador que integra en un mismo circuito integrado las capacidades de control de un microcontrolador (MCU), con las capacidades de computación y rendimiento de un procesador digital de señal (DSP).

La familia dsPIC30F fue un acercamiento al mundo del DSP para los usuarios que ya usaban los PICs de 8 o 16 bits, manteniendo toda su arquitectura y sus juegos de instrucciones de los MCU, y la segunda familia dsPIC33F es un paso adelante potenciando las capacidades y número los posibles periféricos.

2.3.2.2.1. Características de la idea DSC de Microchip.

Los dsPIC de Microchip ofrecen las mejores características de un poderoso MCU de 16-bits tales como:

- Gestión de interrupciones rápida, flexible y sofisticada.
- Una amplia variedad de periféricos analógicos y digitales.
- Gestión del consumo.
- Opciones de reloj flexibles.
- *Power-on-reset* (POR).
- *Brown-out Reset* (BOR).
- *Watchdog* o Perro guardián.
- Seguridad en código.
- Emulación en tiempo real a plena velocidad.

- Soluciones de depuración en circuito a plena velocidad.

Además añaden las capacidades de un DSP:

- Ejecutar la mayor parte de sus instrucciones en un solo ciclo (33ns a 30MIPS).
- *Zero overhead looping*, hardware e instrucciones específicas de repetición, que no requieren comprobación, ni actualización de contadores de iteración.
- Multiplicación de 16-bit en un único ciclo.
- Acumuladores de 40-bit duales.
- Desplazador de barril de 40 bits.
- Búsqueda de dos operandos de modo simultáneo.
- Bucles DO y REPEAT.

Con esto se consigue el MCU de 16-bits más poderoso del momento. Esta nueva tecnología DSC es realmente útil debido a que la mayoría de los usuarios de MCUs que buscan añadir características de DSP a su sistema se encuentran que añadir un chip DSP al microcontrolador puede ser realmente costoso y complicado.

2.3.2.2.2. Características de los dsPIC30F DSC.

La arquitectura dsPIC30F, fue desarrollada en colaboración con un equipo de compilador C. El resultado es un código en C de alta eficiencia, que comparado a cualquier otro compilador de un MCU o DSP de 16-bits, requieren estos otros compiladores, hasta un 70% más de espacio de código de programa para el mismo programa de aplicación escrito en C.

Características principales de los dsPIC30F:

- Arquitectura de 16 bits.
- Hasta 30 MIPS con osciladores de 7,37 MHz y 512 KHz.
- Memoria de programa Flash de entre 6 y 144 Kbytes.
- Memoria RAM de entre 256 y 8196 bytes.
- Memoria EEPROM de entre 0 y 4096 bytes.
- Tensión de trabajo de entre 2,5 V y 5,5 V.
- Muy pocos dsPIC de la familia mantienen los comparadores.
- Hasta 13 canales de entrada del bloque A/D de 10 o 12 bits.

- Interfaces para códec como I2S (*Inter – IC Sound*) o AC97 para algunos dsPIC.
- Hasta dos módulos CAN.
- Bloques de comunicación analógica SPI, I2C y UART. Hasta dos por cada tipo.
- Hasta 8 bloques PWM.
- Hasta 5 temporizadores de 16 bits y 2 de 32 bits.
- Algunos incorporan el bloque QEI (*Quadrature Encoder Interface*).
- Puerto paralelo GPIO.
- ICSP.
- Precio entre 2,5 y 7,40 dólares [20].

2.3.2.2.3. Características de los dsPIC33F DSC.

Características de los dsPIC33F:

- Arquitectura de 16 bits.
- Hasta 40 MIPs con osciladores de 7,37 MHz y 512 Khz.
- Memoria de programa Flash de entre 612y 256 Kbytes.
- Memoria RAM de entre 1024 y 32768 bytes.
- Tensión de trabajo de entre 3 V y 3,6 V.
- Hasta 8 canales DMA.
- Muy pocos dsPIC mantienen los comparadores.
- Bloque A/D de 10, 12 o 16 bits con muchos canales de entrada, de hasta 1100Ksps (kilo símbolos por segundo).
- Interfaces para códec como I2S o AC97 para algunos dsPIC.
- Hasta dos módulos ECAN (*Enhanced CAN*).
- Bloques de comunicación analógica SPI, I2C y UART, hasta dos por cada tipo.
- Hasta 8 bloques PWM.
- Hasta 5 temporizadores de 16 bits y 2 de 32 bits.
- Algunos incorporan el bloque QEI. *(Quadrature* Encoder Interface).
- Puerto paralelo GPIO.
- ICSP.
- Precio entre 2 y 5,64 dólares [20].

2.3.3. FAMILIA DE 32 BITS, PIC32.

Con la llegada de los PIC de 32 bits, Microchip ha dado un gran salto de calidad en sus productos. Para ello ha escogido el núcleo MIPS32 M4K en lugar de optar por desarrollar o ampliar su propia tecnología. Dicho núcleo MIPS. Ofrece un gran capacidad de cálculo en su categoría con una velocidad de 1,5DMIPS/MHz, gracias a su probada arquitectura segmentada (*pipeline*) de 5 etapas. A la vez, ofrece soporte a la tecnología compresión de código MIPS16e, que permite alcanzar reducciones del tamaño de código de hasta el 40%. Además, posee características de capacidad de memoria de datos, memoria de programa, y periféricos que mejoran los antiguos PIC24, como contrapartida, no ofrecen el motor DSP [21].

2.4. dsPIC30F4011.

2.4.1 CARACTERÍSTICAS GENERALES.

Por las características de nuestra aplicación, hemos usado un dsPIC30F4011 de la familia dsPIC30F, puesto que incluye el motor DSP fundamental para el procesado digital de señal, y en nuestro caso fundamental para el procesado de audio.

Pero no solo el motor DSP es la característica interesante de este dsPIC, siendo la interfaz digital de comunicación SPI otra parte muy importante para nuestra futura aplicación.

Aquí se enumeran las características más relevantes del dsPIC30F4011:

CPU RISC modificada.

- Estructura Harvard modificada.
- Arquitectura de instrucciones optimizada para el compilador de C, con modos de direccionamiento flexibles.
- 83 instrucciones base.
- Instrucciones de 24 bits, y datos de 16 bits.
- Memoria de programa: 48 Kbytes de memoria Flash on-chip.
- Memoria de datos: 2Kbyte RAM y 1Kbyte no volátil EEPROM.
- 30 MIPS, reloj externo de 40MHz, y otro de 4-10MHz con PLL (4x, 8x, 16x).

- 30 fuentes de interrupción, con 8 niveles de prioridad.
- 16 x 16 registros de trabajo.

Motor DSP.

- Obtención de datos simultáneos.
- Acumulador para operaciones DSP.
- Módulo generador de direcciones de bit-invertido.
- Dos acumuladores de 40 bits.
- Multiplicador hardware de 17 x 17 bits.
- Las instrucciones del motor DSP duran un ciclo.

Características de los Periféricos.

- Relojes de 32 y 16 bits.
- 16 bits de captura de entrada.
- 16 bits de de comparador o PWM.
- Módulo SPI de 3 hilos.
- Módulo I2C.
- 2 UART con buffers FIFO.
- Módulo CAN.

Características Módulo Motor PWM.

- 6 canales de salida PWM complementarios o independientes con 3 ciclos de trabajo posibles.
- Reloj programable, con polaridad programable.
- Disparo para conversiones A/D.

Características Módulo QEI. Sensor de posición.

- Cuenta los pulsos ascendentes o descendentes hasta una palabra de 16 bits.
- Cuenta en resolución simple (x2) o doble (x4).
- Filtrar la señal de entrada con filtros digitales programables.
- Generar una interrupción según determinados eventos.
- Los contadores pueden usarse como temporizadores.

Características Analógicas.

- Conversor Analógico/Digital de 10 bits.
- Velocidad de muestreo de 1Msps (millones de símbolos por segundo), mediante 9 canales de entrada distintos.

Características especiales del control digital de señal.

- Memoria de programa Flash: 48 Kbyte.
- Memoria de datos EEPROM: 1Kbyte.
- Memoria de datos en chip RAM: 2 Kbyte.
- Reprogramable por control software.
- Distintos modos de energía, POR, PWRT, OST.
- *Watchdog* o perro guardián flexible.
- Protección de código programable.
- ICSP, Programador serie *In-Circuit*.
- Modos de energía seleccionables: *Sleep, Idle*, distintos modos de reloj.

Tecnología CMOS.

- Tecnología flash de alta velocidad y bajo consumo.
- Rango de voltajes 2.5 V – 5.5 V.
- Rango de temperaturas Industrial y Extendido [3].

2.4.2 Encapsulado.

Existen tres variedades de encapsulado para el dsPIC30F4011, son las siguientes:

- 40-pin PDIP (*Plastic Dual In-Line*).
- 44-pin TQFP (*Plastic Thin Quad Flatpack*).
- 44-Pin QFN (*Plastic Quad Flat*).

La placa de desarrollo UIB-PC104 usa el dsPIC30F4011 con encapsulado 44-Pin TQFP; se muestra el patillaje del dsPIC30F4011 en la figura 2.4:

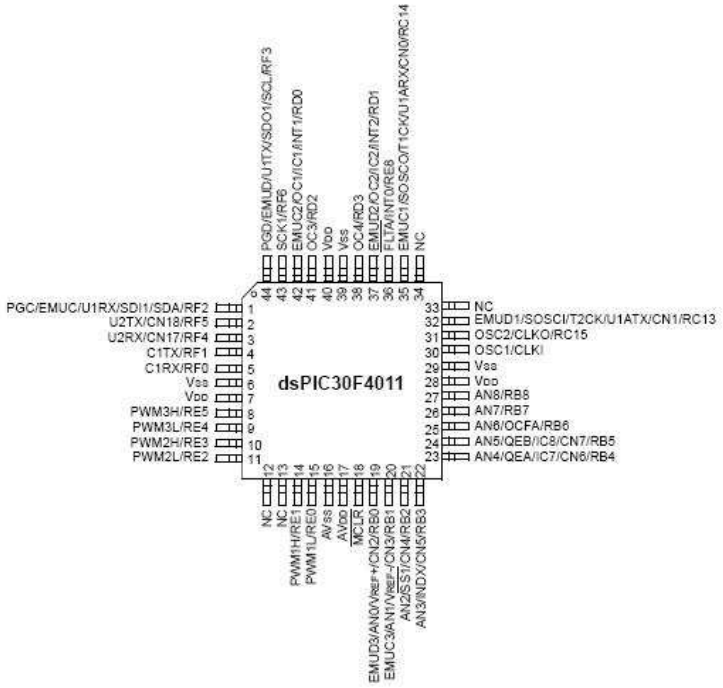

Figura 2.4. Patillaje y encapsulado del dsPIC30F4011 [3].

2.4.3. ARQUITECTURA.

En la figura 2.5 de la página siguiente se muestra el diagrama de bloques de la arquitectura interna del dsPIC30F4011.

En él se aprecian los buses de datos, el motor DSP con sus multiplicadores y acumuladores, los distintos módulos para interfaces digitales como SPI o I2C, y otras utilidades como los temporizadores. También se aprecia el generador de direcciones y la memoria de datos. En definitiva, se ve la organización de todos los bloques que se describirán a continuación.

El primer bloque importante corresponde a la memoria de datos RAM. Esta dividida en dos espacios X e Y. Cada espacio posee su propio generador de direcciones (AGU) permitiendo de este modo acceso simultáneo de datos. Esta memoria es de 2 Kbytes, cada posición de memoria corresponde a 16 bits. También hay una pequeña EEPROM de datos de 1 Kbyte no volátil. En instrucciones MCU los espacios de memoria no se consideran separados

La memoria de programa es de tipo FLASH y las posiciones son de 24 bits, esto se debe a que la arquitectura Harvard mejorada soporta un repertorio de instrucciones elevado. Dicha memoria tiene una capacidad de 48 Kbytes. El direccionamiento de la memoria se

hace mediante el contador de programa (PC) de 23 bits, con los que se puede direccionar 4 millones de palabras de instrucción de 24 bits.

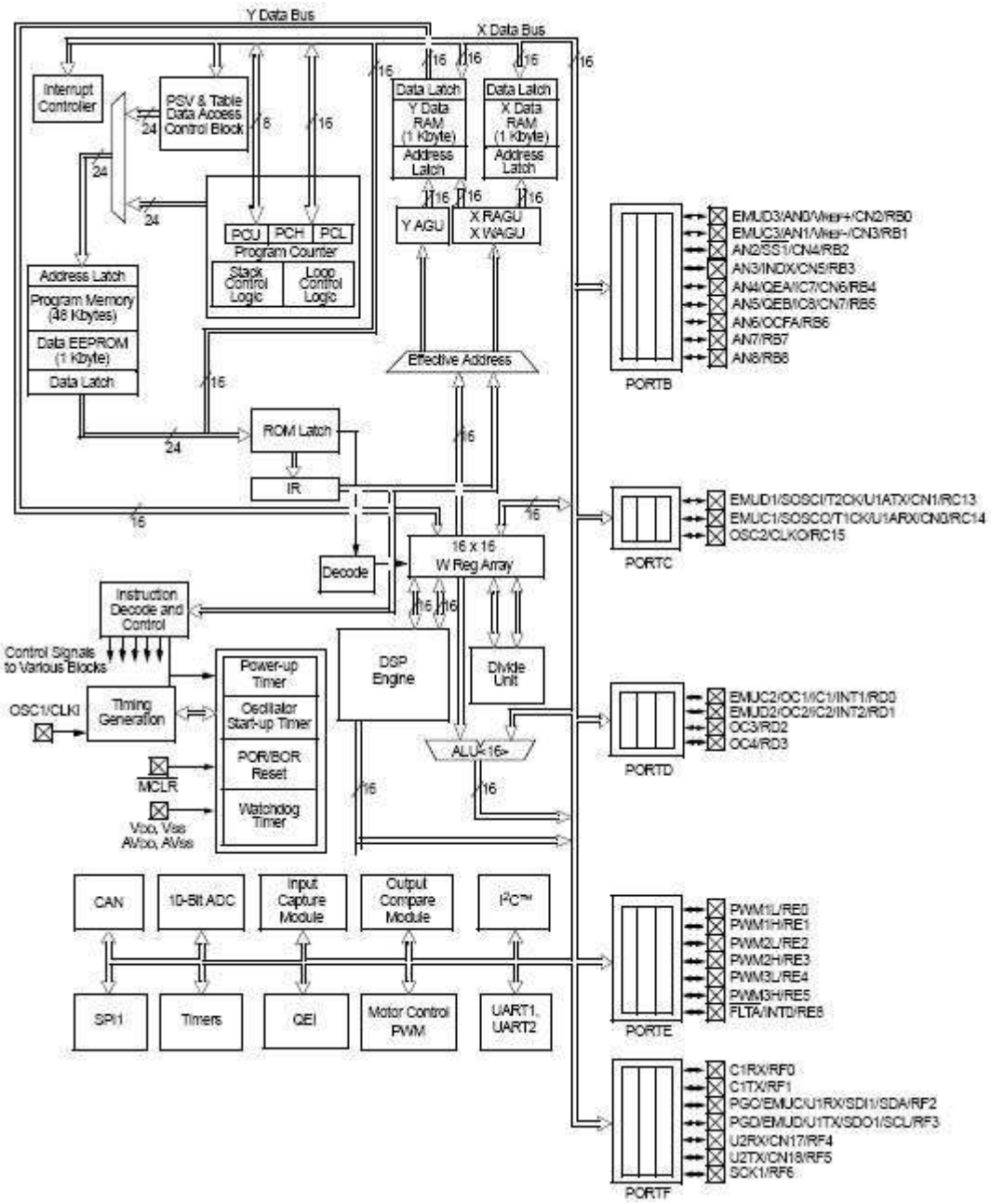

Figura 2.5. Diagrama de bloques del dsPIC30F4011 [3].

Los modos de direccionamiento soportados son los siguientes: inmediato, inherente, directo, relativo, registro directo, registro indirecto y registro de desplazamiento. Además se permiten direccionamientos circulares o de bit invertido pero esto solo para el espacio de memoria X. Los dos últimos direccionamientos son útiles en el Motor DSP.

El camino de los datos se basa en los 16 registros W que posee el dsPIC30F4011. Estos registros son de 16 bits y alimentan la ALU y el Motor DSP. Sus operaciones son de 40 bits, así como también alimenta la unidad de división.

Las 7 puertas de entrada del dsPIC30F4011 permiten la comunicación con el exterior. La mayoría de estas puertas tienen varias funciones multiplexadas. La funcionalidad básica de estas puertas son los pines de los periféricos que al estar multiplexados, no se podrá usar toda la diversidad de periféricos al mismo tiempo.

Para la gestión del sistema, el dsPIC30F4011 usa como generador de la señal de reloj un cristal de cuarzo o un resonador externo. Para generar un reset se usa la activación de la patilla MCLR# (*Master Clear Reset*) o mediante dos eventos que son POR (*Power On Reset*), que consiste en controlar el rizado de la señal, o mediante el evento BOR (*Brown Out Reset*), que consiste en controlar el umbral de VDD y si éste baja de un umbral programado y luego lo supera, se genera el reset. Esto controla si la señal de alimentación no esta estabilizada.

Los modos de bajo consumo son dos: en el primero de ellos (SLEEP) deja de funcionar la CPU y los periféricos puesto que se detiene el reloj del sistema, siendo el modo que consume menos potencia ya que se desactivan todos los osciladores. En el modo IDLE solo se desactiva la CPU, siendo la desactivación de los periféricos optativa. En este modo solo se desactiva el oscilador principal [3][12].

2.4.3.1 El camino de datos.

El camino de datos es el bloque encargado de hacer las operaciones que conlleva la instrucción en curso. Para un DSC el camino de datos es capaz de manejar las instrucciones propias de una MCU de 16 bits, además de las propias de un DSP. Consta de 4 bloques principales: El banco de registros, la ALU de 16 bits, el Motor DSP, y la Unidad de División.

2.4.3.1.1. El banco de registros.

El banco de registros corresponde a 16 registros de 16 bits cada uno, W0-W15, y pueden contener datos, direcciones o desplazamientos. A veces los registros ya tienen una función concreta según la instrucción que se usa, como en la multiplicación fractional,

donde los registros W4 y W5 son los que contienen los operandos en formato fractional a multiplicar [3][12].

2.4.3.1.2. La ALU. Las Instrucciones MCU.

La ALU (*Aritmetic Logic Unit*) o UAL (Unidad Aritmético Lógica), se encarga de las instrucciones MCU que son las instrucciones aritmético lógicas. La ALU controla 5 bits del registro de estado (SR). Son los siguientes:

- C: Bit de acarreo.
- Z: Cero.
- OV: Sobrepasamiento.
- N: Negativo.
- DC: Acarreo en el 4º bit.

Las operaciones de ALU son: suma, resta, desplazamiento de un bit, complemento a dos, OR, AND y EOR, siendo las operaciones de 8 o 16 bits. Las operaciones en 8 bits no modificaran la parte alta de los registros y para pasar de un dato de 8 bits a uno de 16 existe una instrucción para extender el signo, incluso hay otra instrucción que puede poner a cero los 8 bits más altos de un registro [3][12].

2.4.3.1.3. El motor DSP. Las instrucciones MAC.

Las instrucciones MAC de multiplicar y acumular son las instrucciones típicas de los DSP, por ser las instrucciones básicas para el procesamiento digital de señal. Por todo esto el flujo de datos entre los distintos bloques del DSP es muy frecuente. Los datos el motor DSP los recoge desde los registros W, pero a su vez posee registros para contener el resultado.

Para las instrucciones con dos operandos se usan los registros W4-W7 y para las demás instrucciones se usa solo el bus del espacio de memoria X. Toda operación con el DSP se retiene en algún acumulador A o B, o en una posición de memoria según indique la instrucción.

El multiplicador es de 17 x 17 bits y realiza operaciones con o sin signo. Genera los resultados de todo tipo, desde enteros de 32 bits hasta fractional con 31 bits para la parte decimal. Los datos de entrada de 16 bits se convierten en 17 bits para hacer la operación.

Para las operaciones MCU, la propia instrucción ya nos selecciona el tipo de datos, pero en las operaciones MAC el tipo de dato necesita ser programado siendo por defecto o tras un reset las multiplicaciones de formato fractional.

Los acumuladores son dos: el ACCA y ACCB de 40 bits cada uno, pudiendo ser el dato guardado de 16, 32, o 40 bits. Se puede acceder a cualquier formato de datos del acumulador.

El sumador/restador de 40 bits, puede hacer operaciones sobre los acumuladores, como por ejemplo, el valor obtenido de una multiplicación. Actúa sobre 6 bits del registro de estado (SR), que son los siguientes:

- OA y OB sobrepasamientos de los bits de guarda de los acumuladores A y B, respectivamente;
- OAB es el resultado de hacer la OR lógica entre OA y OB;
- SA y SB: saturación de los acumuladores;
- SAP es el resultado de hacer la OR lógica entre SA y SB.

Estos bits pueden generar excepciones por error aritmético, lo cual es programable como si fuese una interrupción.

El registro de desplazamiento realiza desplazamientos de hasta 16 bits a izquierda o derecha. Para el desplazamiento se usa un número con signo, siendo su signo el que determina la dirección del desplazamiento de bits [3][12].

2.4.3.1.4. La unidad de división.

La unidad de división hace divisiones entre datos de 16 o 32 bits, incluso mezclando formatos, por ejemplo, dividir un número de 32 bits entre uno de 16 bits. En toda división el cociente esta en W0 y el resto en W1. Para usar divisores de 32 bits hay que usar registros W correlativos. Una división se ejecuta como un bucle REPEAT de 18 iteraciones, por lo que se recomienda guardar el contenido de los registros W de la rutina de interrupción ISR.

En la figura 2.6 se muestra el esquema de todos los subbloques del motor DSP:

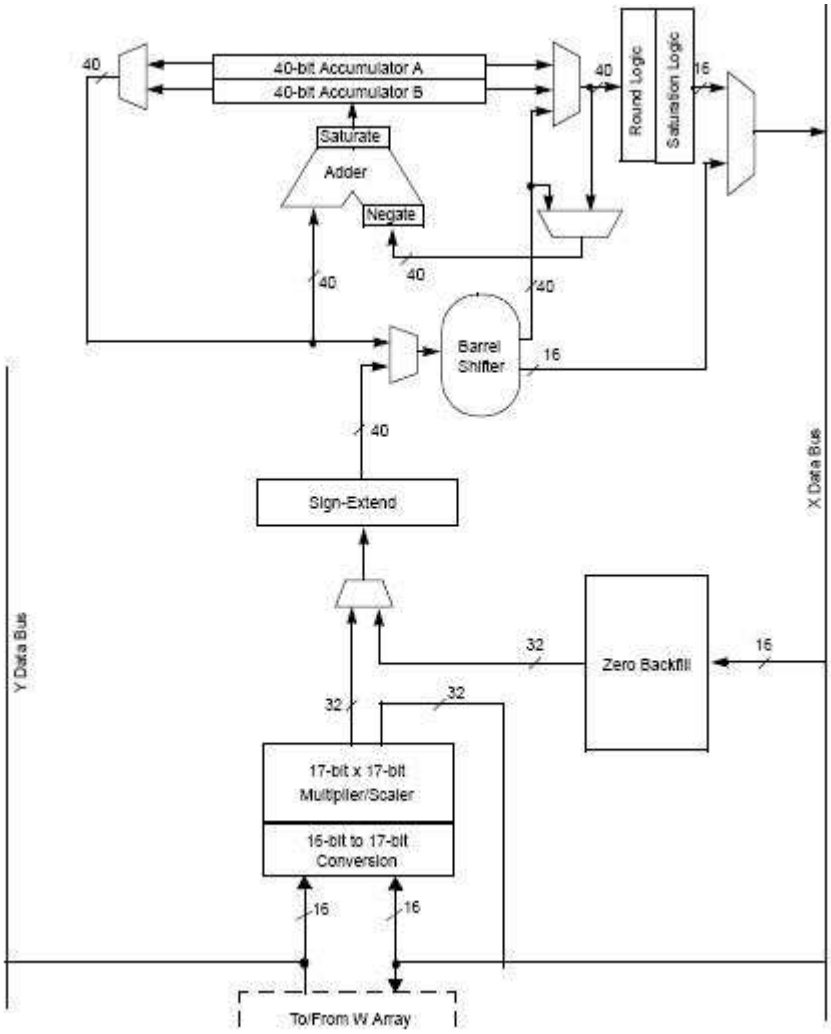

Figura 2.6. Diagrama de bloques del Motor DSP [3][12].

2.4.3.2 El modelo del procesador.

El banco de registros denominados W0-W15 es la base del camino de datos, siendo la utilidad básica contener datos o direcciones de memoria. Algunas instrucciones que hacen uso implícito de WREG se refieren al registro W0. Para una división W0 guarda el cociente y W1 el resto de la una división. W3 y W2 guardan el resultado de la multiplicación. W4-W7 se usan para operandos de instrucciones MAC; W8-W11 contienen direcciones de datos de instrucciones MAC, W12 es el offset de las instrucciones; W13 es el registro de trabajo de operaciones MCU, y registro de post o pre escritura de acumulador; W14 es el puntero de Marco de pila; por último W15 es el puntero de pila.

Se mostrará a continuación en la figura 2.7 con el modelo del procesador:

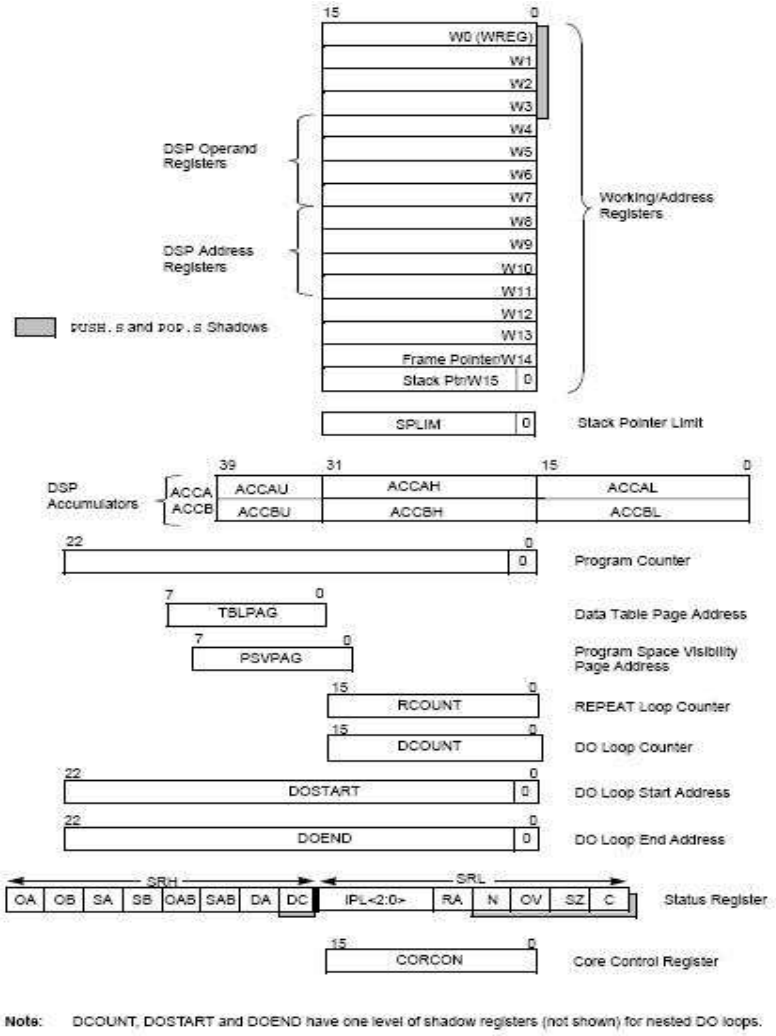

Figura 2.7. Modelo del programador [3][12].

Para las operaciones PUSH y POP se usan los registros W0-W3, usándose para la restauración de la pila tras la llamada a una rutina o el depósito en los registros de los datos de la cima de la pila.

Para la instrucción DO se usan los siguientes registros, DOSTART, DOEND, DCOUNT, para llevar el control de donde empieza el bucle DO, y donde acaba dicho bucle DO.

El registro de estado (SR) informa del estado y condiciones tras la ejecución de operaciones aritméticas.

El registro de control de núcleo (CORCON), establece preferencias de funcionamiento del núcleo como nivel de privilegio de interrupción de la CPU, bits de control de la multiplicación, del bucle DO, de redondeo. Los bits importantes son los siguientes:

- US: Determina si la multiplicación del Motor DSP es con o sin signo.
- SATA, SATB: Activa o desactiva la saturación del acumulador correspondiente.
- ACCSAT: Selecciona el tamaño de la parte entera del acumulador en 1 o 9 bits. Es decir, si se tienen en cuenta o no los bits de guarda del acumulador.
- RND: Determina si el redondeo es convencional o convergente.
- IF: Selecciona si la multiplicación es entera o fractional [4].

Otros registros son:

- TBLPAG: Registro de página para las instrucciones de tabla.
- PSVPAG: Registro de página para visibilidad del espacio de programa.
- MODCON: Registro de control del direccionamiento circular, como por ejemplo, los registros W que llevan a cabo el direccionamiento circular de cada espacio de memoria.
- XMODSRT, XMODEND: Registros de inicio y fin para controlar el direccionamiento circular del espacio X.
- YMODSRT, YMODEND: Igual que los anteriores pero para controlar el direccionamiento circular en el espacio de datos Y.
- XBREV: Registro para el control del direccionamiento por bit invertido.
- DISCNT: Registro Contador para desactivación de Interrupciones [3][12].

2.4.3.3 Memoria de datos.

La memoria RAM de los DSC consta de dos espacios X e Y, los cuales son independientes para algunas instrucciones DSP. Cada espacio va acompañado de sus buses de datos y de direcciones de 16 bits, siendo capaces de direccionar 64KB.

La memoria se divide en 5 bloques, siendo de 2KB para los registros de control, dos espacios de datos X e Y, zona no implementada, y los últimos 32 KB se corresponden a una zona de datos opcional que se puede mapear en la memoria de programa.

En la figura 2.8 se muestra la organización de la memoria de datos de modo gráfico.

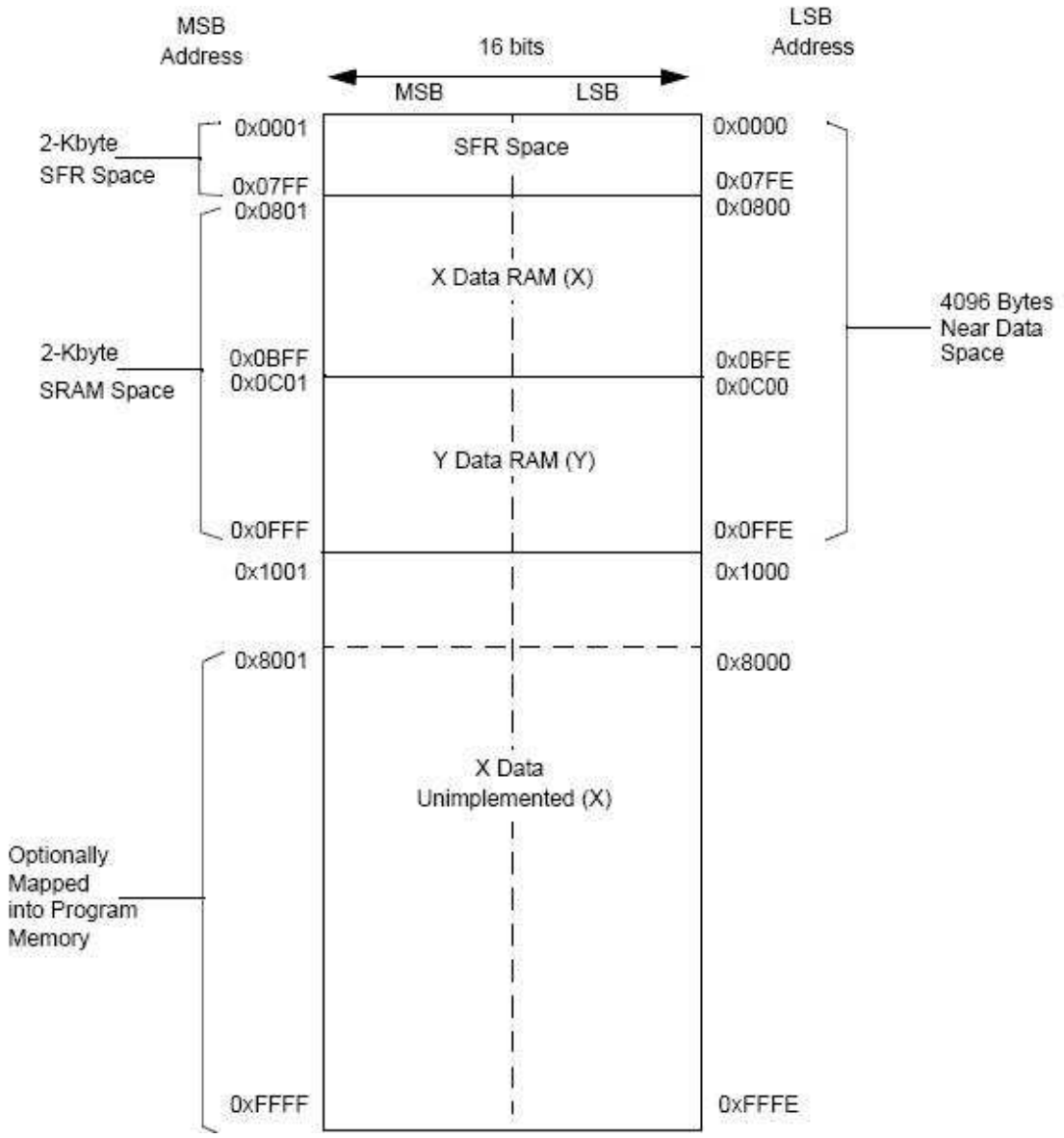

Figura 2.8. Organización de la Memoria de datos.

Los 8KB inferiores son la memoria cercana y son los que puede usar el direccionamiento directo puesto que solo se usan 13 bits para la dirección. En instrucciones DSP se pueden acceder de modo simultáneo a datos de ambos espacios mediante punteros en los registros W8 y W9 para el espacio X y W10, W11 para el espacio Y [3][12].

2.4.3.3.1. Los generadores de direcciones.

El núcleo de los dsPIC DSC posee dos generadores de direcciones independientes: AGUX y AGUY. AGUY solo soporta datos de tamaño word para las instrucciones del MAC del DSP. Soporta 3 modos de direccionamiento de datos: lineal, circular y de bit invertido, siendo los dos primeros para espacio de datos y código, y el último solo para datos.

El mapeado del espacio de datos en la memoria de programa consiste en usar los 32 Kbytes del espacio de datos en una página del espacio de memoria de programa de 16 K palabras de 24 bits. De cada palabra se descartan los 8 bits altos. Para lograr la dirección real se usan 15 bits bajos de un registro W, los 8 siguientes se cogen de la página que hemos seleccionado. Esta información es el registro PSVPAG [3][12].

2.4.3.3.2. Direccionamiento circular.

El módulo de direccionamiento circular, es el mecanismo hardware para ejecutar buffers circulares. Puede ser ejecutado en los espacios de memoria X e Y, pudiendo usar cualquier registro excepto los W14 y W15. Para hacer un módulo de direccionamiento hay que escribir 4 registros, que indican el inicio y fin de la zona de memoria reservada, estos son XMODSRT, XMODEND, YMODDSRT, YMODEND. El tamaño total no tiene valores restringidos, solo tiene un máximo posible de 32Kwords o 64Kbytes.

El módulo de direccionamiento circular hay que habilitarlo con un número distinto de 15 en los siguientes bits: MODCON [3:0] para el espacio X, y MODCON [7:4] para el espacio Y, para determinar qué registro funciona con este direccionamiento [3][12].

2.4.3.3.3. Direccionamiento de bit invertido.

El direccionamiento de bit invertido solo esta disponible en el espacio de direcciones X. Hay que poner en los bits BWM del registro MODCON el número del registro que usará direccionamiento de bit invertido excepto el 15, luego hay que poner el bit BREN, en el registro XBREV a 1. No usa todos los tamaños, y el tamaño será de 2 exp N bytes. [3][12].

2.4.3.4. Memoria de programa.

La memoria de programa puede ser de hasta 4 millones de palabras de 24 bits. Se direcciona con los 23 bits del contador de programa (PC). Las direcciones (0x000000 – 0x7FFFFE) son la zona de memoria del usuario, siendo la restante para la configuración. El bit TBLPAG [7] determina si se accede a la memoria de usuario o a la memoria de configuración.

La figura 2.9 muestra la memoria de programa:

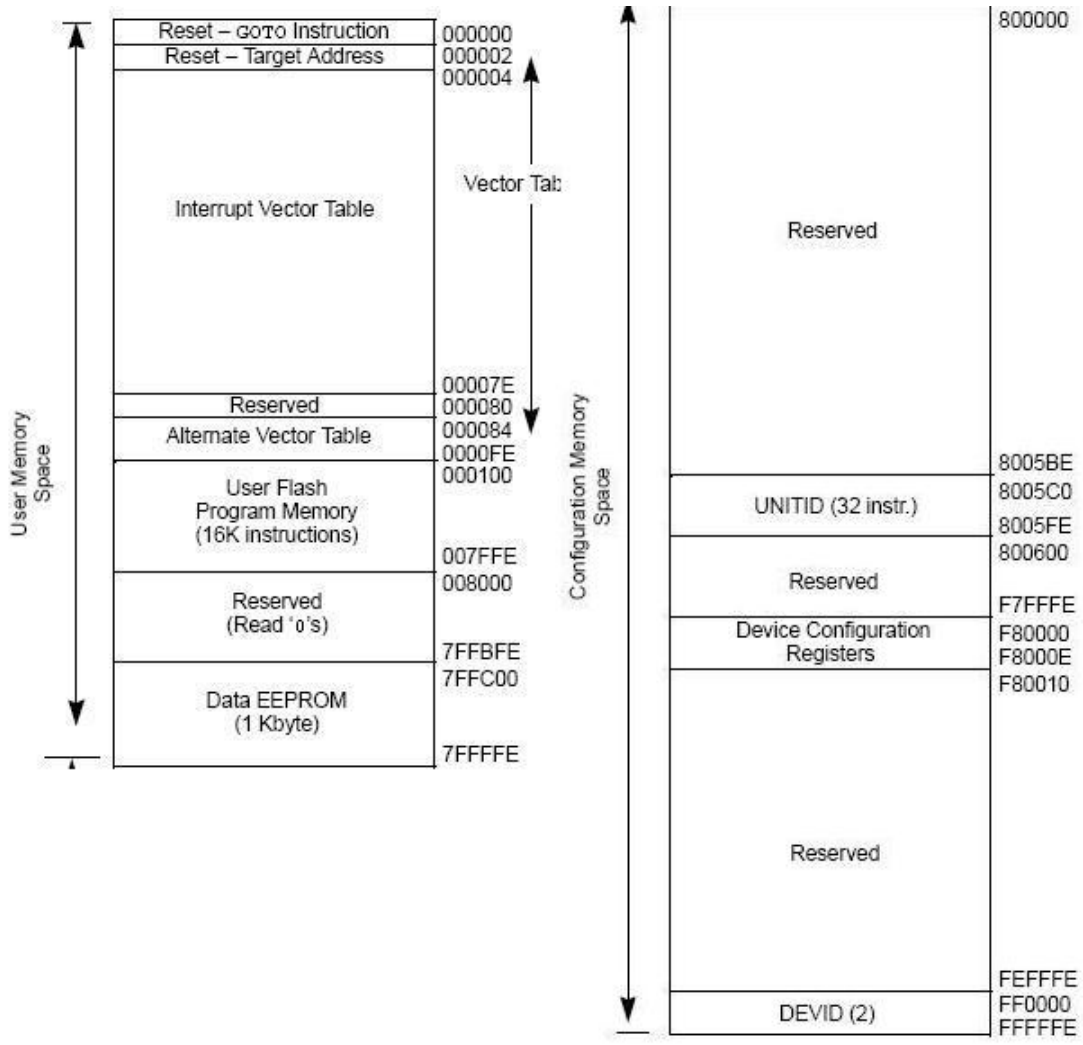

Figura 2.9. Organización de la memoria de programa.

Existen 3 métodos para acceder al espacio de direcciones de memoria de programa: directamente a través del Contador de Programa (PC), mediante instrucciones como lectura de tabla o escritura de tabla, o mapeando 32Kb del espacio de datos en la memoria de programa.

La memoria Flash contiene el código ejecutable para nuestro dsPIC, existen dos métodos para cargar el programa en la memoria Flash: ICSP y RTSP.

ICSP (*In-Circuir Serial Programming*): Son dos líneas simples para reloj y para datos, PGC y PGD y otras tres líneas de Tensión VCC, Tierra GND y Clear MCLR.

RTSP (*Run-Time Self-Programming*): Con este procedimiento se puede escribir o borrar 96 bytes por ciclo de reloj en la memoria de programa. Usa cuatro registros del SFRs para su labor, registros usados en las memorias EEPROM [3][12].

2.4.3.5. Interrupciones y excepciones.

Son las causas que pueden desviar el flujo de control en la ejecución de instrucciones. Las interrupciones son generadas externamente y las excepciones son generadas cuando el procesador detecta errores o anomalías.

El dsPIC30F4011 tiene 30 fuentes de interrupción y 4 traps del procesador, las cuales son arbitradas por un sistema de prioridades. En la figura 2.10 se muestra la organización de memoria de las traps y de las tablas de interrupción.

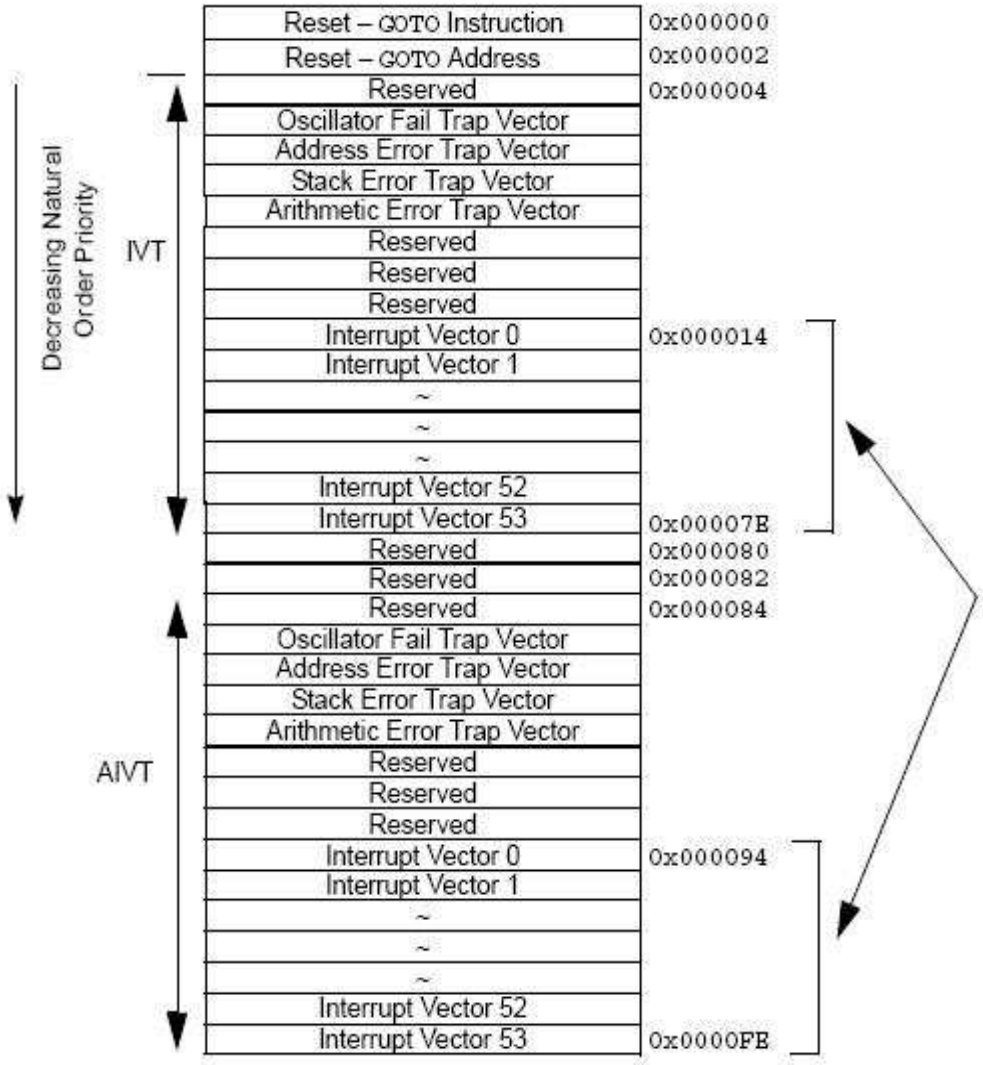

Figura 2.10. Organización de la zona de memoria para las tablas de interrupciones y excepciones [3][12].

La CPU es la responsable de leer el correspondiente vector de interrupciones de la tabla de vectores (IVT) y mandar el vector correcto al contador de programa (PC). Además de la IVT, existe la tabla de vectores alternativa (AIVT), que se encuentra a continuación de la IVT y ocupa 54 posiciones, ocupando la AIVT 62 posiciones. Las primeras 8 posiciones de la IVT guardan la dirección de inicio de las 8 excepciones. La AIVT se usa para procesos de simulación y emulación y posee la misma estructura de la IVT.

Existen una serie de registros que llevan el control dentro del procesador para determinar cómo actuar ante eventos que pueden provocar interrupción, son estos:

- IFS: Son marcas que avisan que se produjo el evento que genera la interrupción.
- IEC: Habilitan o deshabilitan la interrupción.
- IPC: Son tres bits por interrupción, y es la prioridad de cada interrupción.

2.4.3.6. Periféricos.

2.4.3.6.1. Puertos E/S.

El dsPIC30F4011 posee una gran variedad de periféricos y por tanto las patillas de E/S se multiplexan en más funciones, y cuando un periférico la usa, deja de ser una E/S digital que normalmente se usaría para la lectura de datos de sensores.

Los puertos E/S usan registros LATx para evitar el problema de que una puerta pueda ser leída, modificada o escrita. Por tanto las escrituras se hacen en el registro LATx en vez de en el puerto PORTx. Una lectura recoge el dato del PORTx si se lee la patilla o de LATx si se lee el valor guardado en el LATx. La figura 2.11 muestra la arquitectura completa de un puerto E/S.

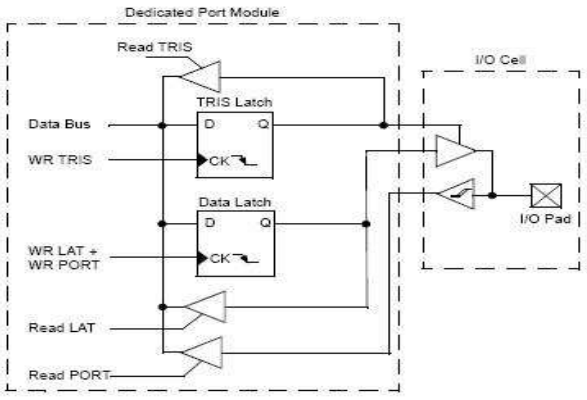

Figura 2.11. Arquitectura de los puertos E/S [3][12].

2.4.3.6.2. Temporizadores.

Existen tres temporizadores distintos, se detalla un poco cada uno de ellos:

- TIMER1: Corresponde a un temporizador de 16 bits, tiene 3 modos de funcionamiento, reloj normal, y contador asíncrono o síncrono.
- TIMER2/3: Pueden funcionar juntos como un temporizador de 32 bits, o como dos relojes de 16 bits independientes. Pueden ser usados por otros periféricos como el módulo de captura de entrada, o el comparador de salida. Con estos temporizadores se genera una señal de activación evento para el ADC. Tanto en modo de 32 como de 16 bits no se acepta modo asíncrono de la cuenta.
- TIMER4/5: Son similares a los TIMER2/3 pero más simples, ya que no posee la característica del ADC y no son usados por los otro periféricos [3][12].

2.4.3.6.3. Módulo de captura de entradas.

Su misión fundamental es medir el tiempo que dura un evento externo producido por la señal de entrada aplicada al módulo. El tiempo se mide con los temporizadores 2 o 3, y es útil para medidas de frecuencia o de periodos de un pulso.

Puede funcionar en varios modos: por flanco descendente, ascendente, cada 4 flancos ascendentes, y cada flanco ya sea ascendente o descendente de una señal.

Se detalla su arquitectura en la figura 2.12.

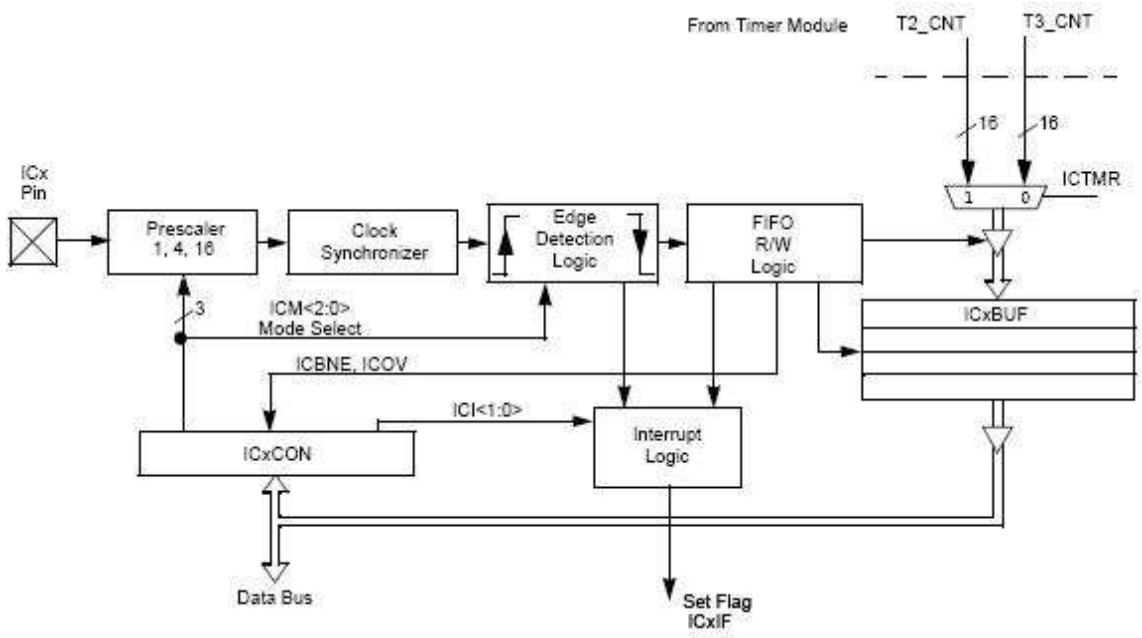

Figura 2.12. Arquitectura del módulo de captura de entradas [3][12].

2.4.3.6.4. Módulo comparador de salida. (*Output Compare Module*).

Se usa para generar impulsos de anchura variable y para aplicaciones que necesitan operaciones simples de PWM, aunque para esto existe otro módulo más específico, pero no todos los modelos de la familia poseen, pero sí en nuestro caso.

Posee modos de trabajo de comparación simple, doble modo de comparación de salida por pulso simple o continuo de salida., y modulación simple de ancho de pulso con/sin protección de errores. Su arquitectura se muestra en la figura 2.13.

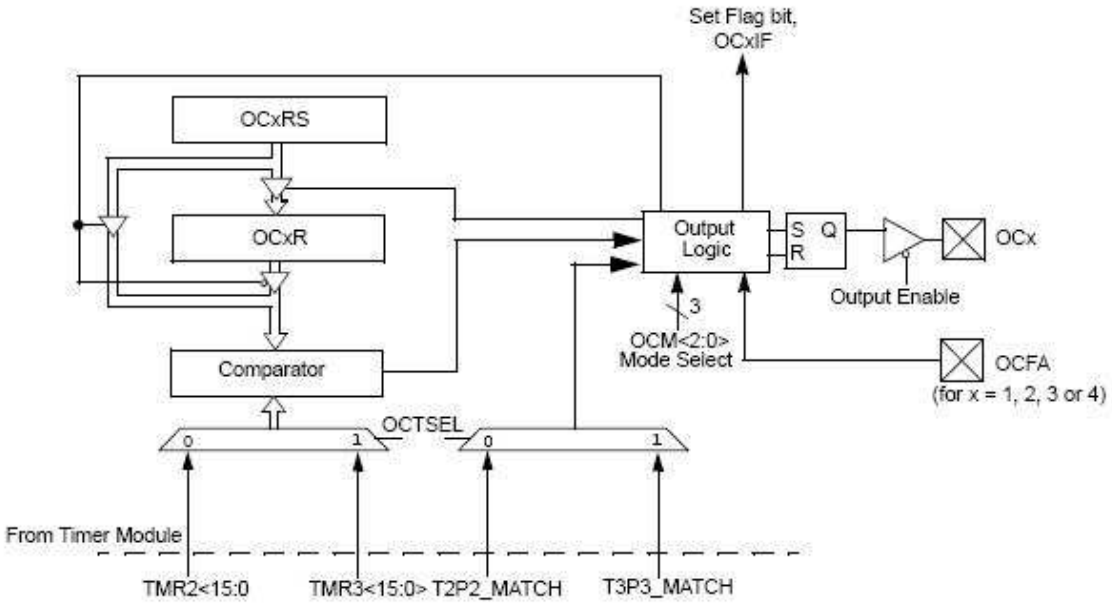

Figura 2.13. Arquitectura del módulo comparador de salida. [3],[12].

2.4.3.6.5. Módulo QEI. (Quadrature Encoder Interface)

La interfaz del codificador de cuadratura nos ofrece un modo cómodo para adaptar los codificadores incrementales usados para detectar la posición y velocidad de los sistemas rotacionales propios de los ejes de los motores, así como conocer la exactitud de la posición del eje del motor. Proporciona tres señales, FASE A, FASE B, e INDEX PULSE.

Sus principales características son:

- Tres patillas: una por cada señal.
- Filtros programables de ruidos a la entrada.
- El decodificador de cuadratura posee un contador de pulsos ascendente y descendente.

- Resolución de 2x y 4x.
- Contador de propósito general de 156 bits ascendente y descendente.
- Interrupciones [3][12].

2.4.3.6.6. Módulo PWM (*Pulse-Width Modulated*) para control de motores.

Este módulo genera trenes de pulsos de anchura variable para controlar la potencia que se entrega a un motor y así regular la velocidad de dicho motor con precisión. Ahora se detallarán las características y modos de trabajo.

Características:

- 6 salidas PWM y 3 generadores de los impulsos con 16 bits de resolución.
- Salidas agrupadas por parejas y así controlar la polaridad.
- Modos de salida por flancos o por el centro del pulso.
- Modo de generación de pulso único.
- Genera interrupciones.

Modos de trabajo:

- Modo normal. Se configura un número para que cuando llegue el contador generar una interrupción.
- Modo de pulso único. Se configura un número y al llegar ahí el contador, se genera un pulso.
- Modo ascendente/descendente. Se configura un número y al llegar ahí, se empieza a descender hasta 0 y luego a subir hasta el número, esto hace que se genere un pulso de ancho controlado.
- Modo de disparo especial. Combinado con el módulo CAD hace operaciones combinadas[3][12].

2.4.3.6.7. Módulo SPI (Serial Peripheral Interface)

Es un interfaz digital de comunicación serie sincronía. Se usa para comunicación con otros periféricos como las EEPROMs, drivers de displays, conversores A/D y D/A u otros microcontroladores. Es compatible con el SPI de Motorola y con los interfaces SIOP.

Consta de un registro de desplazamiento que se usa tanto para el dato de entrada como el de salida, un buffer donde se cargan los datos antes de la transmisión (dato de salida) y donde se reciben al acabar los 16 ciclos de recepción (dato de entrada), además de un registro de control y otro de estado que determinan el modo de funcionamiento del módulo.

Consta de 4 pines, datos de entrada y salida (SDI y SDO), un pin de reloj de bit (SCLK) y una señal para determinar el inicio de la transmisión (FF).

Para comunicarse con el bus interno del dsPIC30F4011 existe el registro SPI1BUF, donde se guarda el dato antes de trasmitirlo a través de SDO, o donde se guarda el dato que llegó a través del pin SDI. También posee un registro de desplazamiento SPI1SR que recibe en cada ciclo un bit y envía un bit, por lo que al inicio de la transmisión contiene el dato a transmitir y al final contiene el dato recibido.

Otros bloques son el preescalado de la señal de reloj, el sistema de selección de flanco activo, y el bloque de generación de la señal FF. En la figura 2.14 se ve la arquitectura con los bloques, registros y señales comentadas.

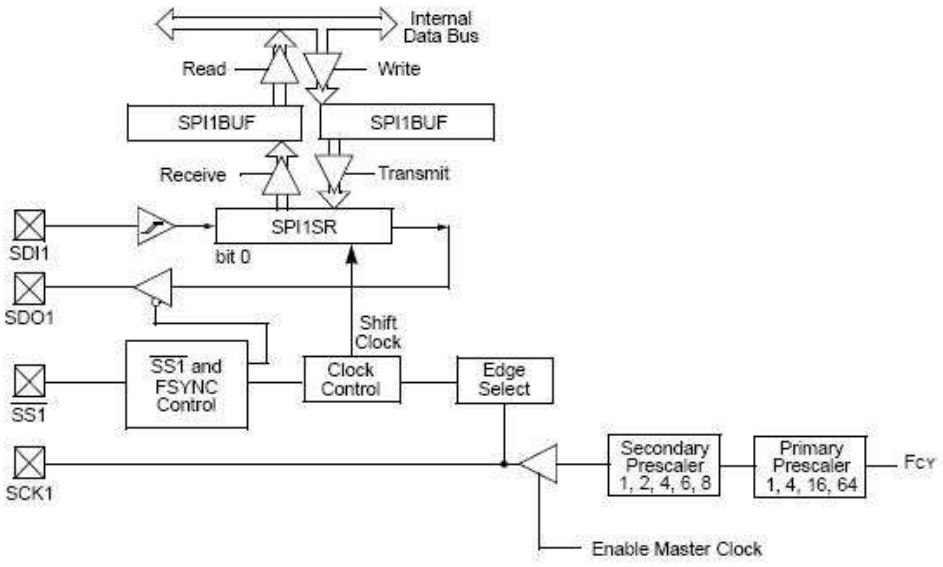

Figura 2.14. Arquitectura del modulo SPI [3],12].

2.4.3.6.8. Módulo I2C (*Inter-Integrated-Circuit*).

Proporciona un completo hardware para modos esclavo y multi-maestro de la interfaz de comunicaciones I2C de 16 bits. Consta de dos hilos, uno para lo datos (SDA) y otro para el reloj que sincroniza toda la transmisión, (SCL).

El módulo I2C tiene las siguientes características:

- Soporta interfaz I2C maestro o esclavo.
- Soporta en ambos modos direccionamiento de 7 o 10 bits.
- Permite transferencias bidireccionales entre esclavos y maestros.
- Posee técnicas de detección de colisiones y arbitraje del bus.

En la figura 2.15 se muestra un esquema con toda la arquitectura del módulo I2C, con sus registros principales como el registro de desplazamiento (I2CRSR), el registro donde se leen los datos que llegan al módulo (I2CRCV), otro registro donde se escriben los datos a mandar por el módulo (I2CTRN). El registro I2CCADD posee la dirección del esclavo donde se manda el dato, así como los registros de control y estado (I2CCON e I2CSTAT).

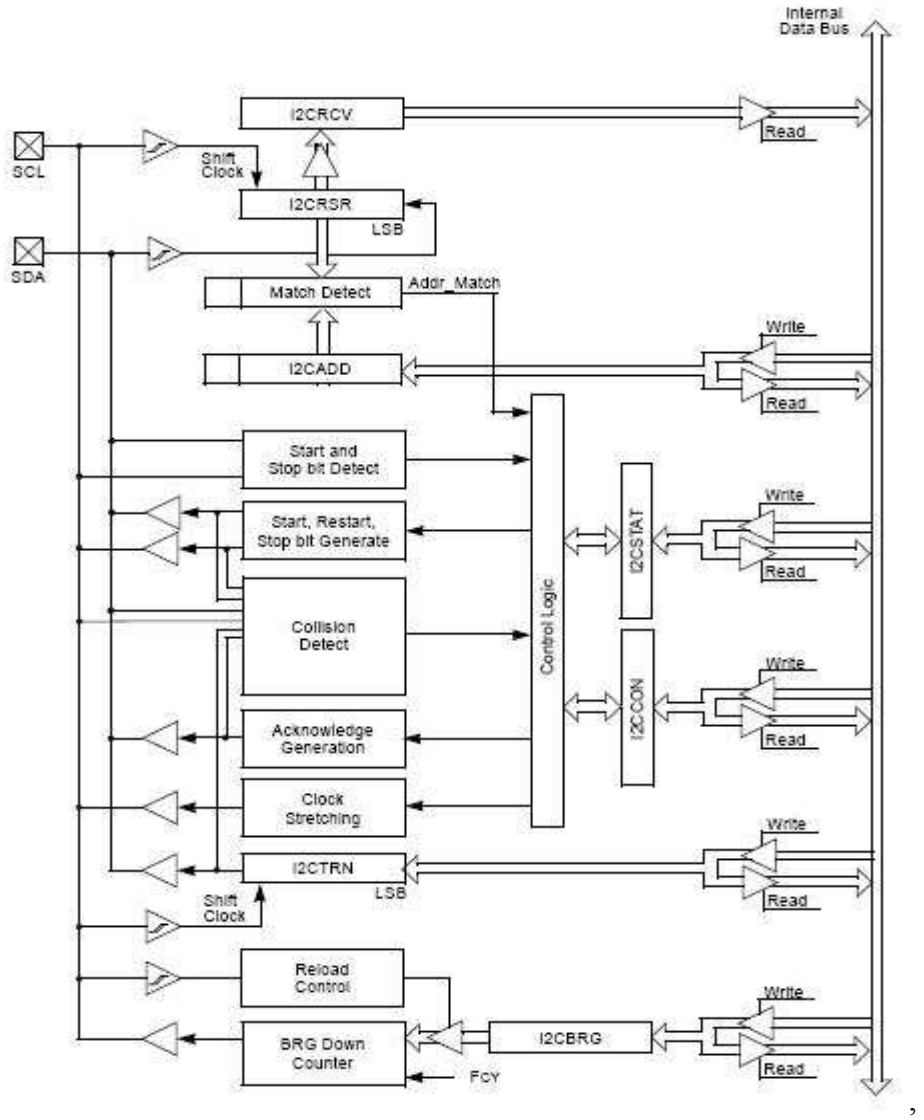

Figura 2.15. Arquitectura del módulo I2C [3][12].

2.4.3.6.9. Módulo UART (Universal Asynchronous Receiver/Transmitter).

Es un módulo de comunicaciones serie asíncrona que tiene dos líneas: UTX de transmisión y URX de recepción de datos. Los bits entran y salen a una frecuencia controlada internamente gracias a un generador de baudios y a unos registros, generándose así la frecuencia deseada de la transmisión. Es un módulo muy versátil gracias a las siguientes características:

- Comunicación *full duplex* de 8 y 9 bits.
- Distintas opciones de paridad.
- Uno o dos bits de parada.
- Generador de baudios de 16 bits.
- Buffer que soportan hasta 4 word.
- Detectan errores de paridad, de tramas, y de buffer lleno.
- Interrupciones separadas para transmisión y recepción.
- Soporta desde 38 Hz hasta 2,5Mbps [3][12].

2.4.3.6.10. Módulo CAN (*Controller Area Network*).

Es un módulo de interfaz serie usado para comunicaciones con otros módulos CAN ya sean de un periférico o de otro microcontrolador. Soporta los protocolos siguientes: CAN 1.2, CAN 2.0A y CAN 2.0B tanto en versión pasiva como versión activa.

El bus CAN tiene la ventaja de que reduce las líneas de trabajo y funciona bien en entornos ruidosos. Los mensajes de este protocolo no tienen direcciones sino identificadores. Todos los nodos analizan el identificador del mensaje y si es el suyo, recogen el mensaje o trama.

Como resumen de sus características:

- Tramas de datos estándar y extendido, y soporte para tramas remotas.
- Tamaño de datos de 0 a 8 bytes.
- Tasa de bits (número de bits por segundo) programable hasta 1Mbit/sec.
- Doble buffer en el receptor.
- Modos de ahorro de energía.
- 6 filtros completos: 2 para el buffer en máxima prioridad, y 4 para la baja.

- 2 máscaras de filtros para prioridad alta y baja.
- Puede generar interrupciones si se generan errores [3][12].

2.4.3.6.11. Módulo Conversor ADC (10-bit, High Speed A/D Converter Module).

Este módulo esta basado en Arquitectura SAR (Registros de aproximaciones sucesivas) con un máximo de frecuencia de muestreo de 1 Msps (millones de símbolos por segundo). El módulo tiene 16 entradas analógicas que son multiplexadas en 4 amplificadores de muestreo y retención (*sampled and hold*) y luego se pasan al conversor A/D. La multiplexación de los canales de entrada determina la frecuencia de muestreo.

Sus características son:

- Velocidad hasta 1Msps.
- Buffer de 16 word para almacenar resultados.
- Generación de interrupción cuando la conversión está lista.
- Hasta 4 entradas analógicas para muestreo simultáneo.

En la figura 2.16 se muestra la arquitectura del módulo conversor A/D.

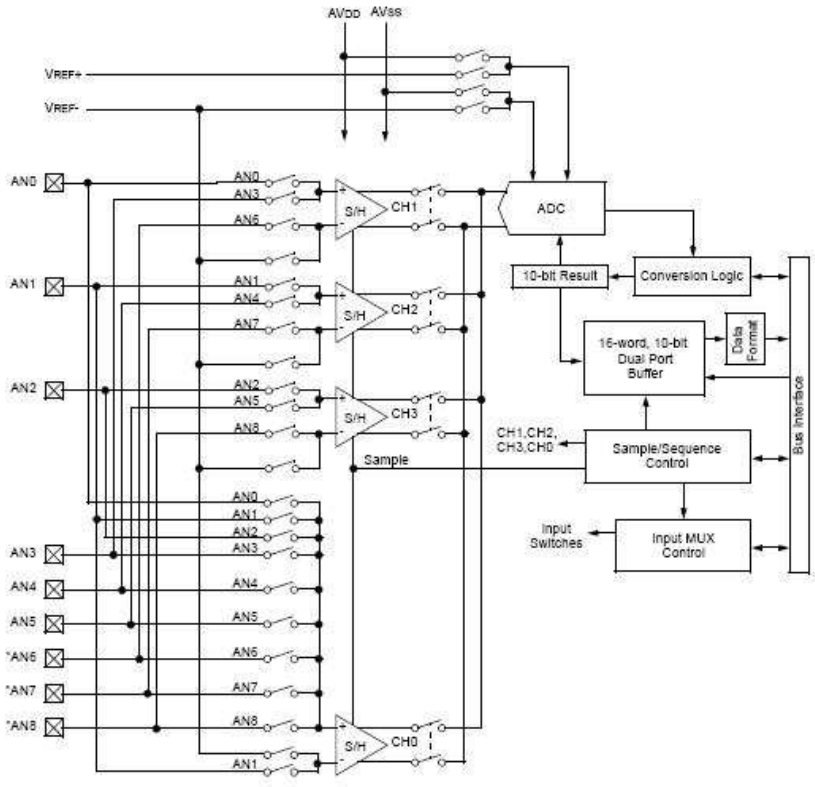

Figura 2.16. Arquitectura del módulo conversor A/D [3][12].

2.4.3.7 El perro guardián.

El perro guardián. (WDT) es un temporizador que vigila el procesamiento del flujo de control y reinicia el procesador ante los fallos. Los eventos básicos donde actúa son: cuando el procesador se queda en un bucle infinito o esta esperando un evento que nunca se produce. El perro guardián se puede deshabilitar mediante el bit FWDTEN del registro FWDT.

Funciona con un registro donde el valor cargado, es el valor que una vez llegado a él, se reinicia el procesador. Pero hay lugares donde se pone a cero el contador y así se evita el reset. Entonces si el procesador se queda demasiado tiempo en un bucle o espera demasiado un evento, no se refresca el contador y se reinicia el procesador.

2.5. LAS PLACAS DE DESARROLLO DE DSPIC.

2.5.1. EL MERCADO DE LAS PLACAS DE DESARROLLO DE DSPIC.

Los PIC de Microchip son unos microcontroladores muy extendidos y muy usados en el mundo de la Ingeniería, por tanto, son muchas las placas de desarrollo que ofrece el mercado y varios los fabricantes que se dedican a su desarrollo.

Estas placas de desarrollo están enfocadas principalmente a facilitar el aprendizaje de los PIC o dsPIC, pero según el modelo, puede que sea más favorable para unos diseños u otros, como placas más específicas para el control de motores.

A continuación se verán algunos ejemplos de estas placas de desarrollo con las características principales de cada una de ellas.

2.5.1.1. dsPICDEM 1.1 Plus.

dsPICDEM 1.1 Plus es una placa de desarrollo de Microchip. Soporta todos los dsPIC de 16 bits y esta ideada para poder desarrollar multitud de aplicaciones distintas, como medidas de temperatura o voltaje, medidas de las características en frecuencia de una onda o para hacer filtros digitales FIR o IIR.

En la figura 2.17 se muestra el aspecto de la placa dsPICDEM 1.1 Plus.

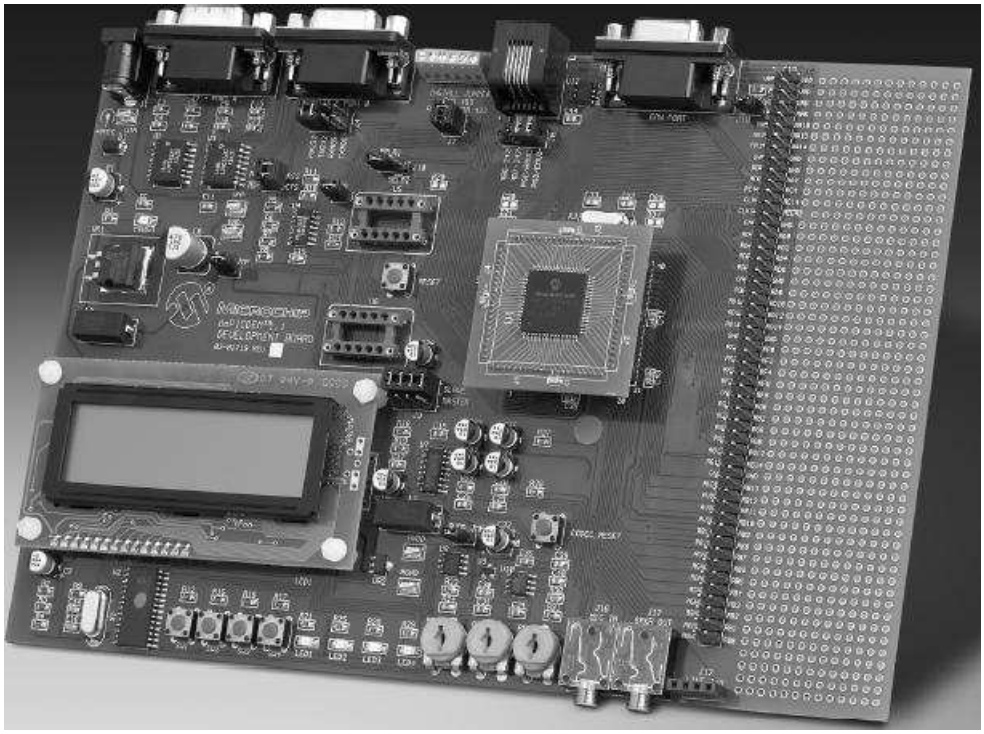

Figura 2.17. La placa de desarrollo dsPICDEM 1.1 Plus.

Consta de varias interfaces de comunicaciones serie como son los módulos SPI, I2C, y dos bloques UART. Consta de un códec de voz Si3000, LCDs, leds, potenciómetros, sensor de temperatura, reguladores de tensión analógica y digital separados, así como un potenciómetro para regular el CAD.

Para nuestra aplicación hubiese sido interesante esta placa puesto que consta de un códec de voz, y se hubiese podido elegir un dsPIC con el bloque de comunicaciones DCI, más apropiado para el tratamiento de muestras de audio [22].

2.5.1.2. dsPICDEM 2.

Desarrollada por Microchip, esta placa de desarrollo usa microcontroladores de la familia dsPIC30F, desde el dsPIC30F2010 hasta el dsPIC30F4013, siendo el dsPIC30F4011 el incluido en su compra.

Incluye interfaces de comunicaciones como RS232, CAN. Incluyes leds, botones y switches, potenciómetro, un sensor de temperatura, y una pantalla LCD de 2x16 caracteres.

No consta de un códec de audio, así que se la puede catalogar como una placa de baja calidad pues los recursos que dispone adicionales al dsPIC son limitados [23].

2.5.1.3. Audio PICtail Plus.- EXPLORER 16.

Junto a la placa de desarrollo Explorer 16 de Microchip, la tarjeta Audio PICtail Plus es una herramienta potente para aplicaciones de audio como las tratadas en gran parte en este PFC, debido a la inclusión del WM8510 Mono CODEC de Wolfson Microelectronic, que es un códec de 24 bits y trabaja a frecuencias de muestreo de hasta 48 KHz, y 4 Mbits de memoria Flash para usar en las aplicaciones. La figura 2.18 muestra la tarjeta Audio PICtail Plus.

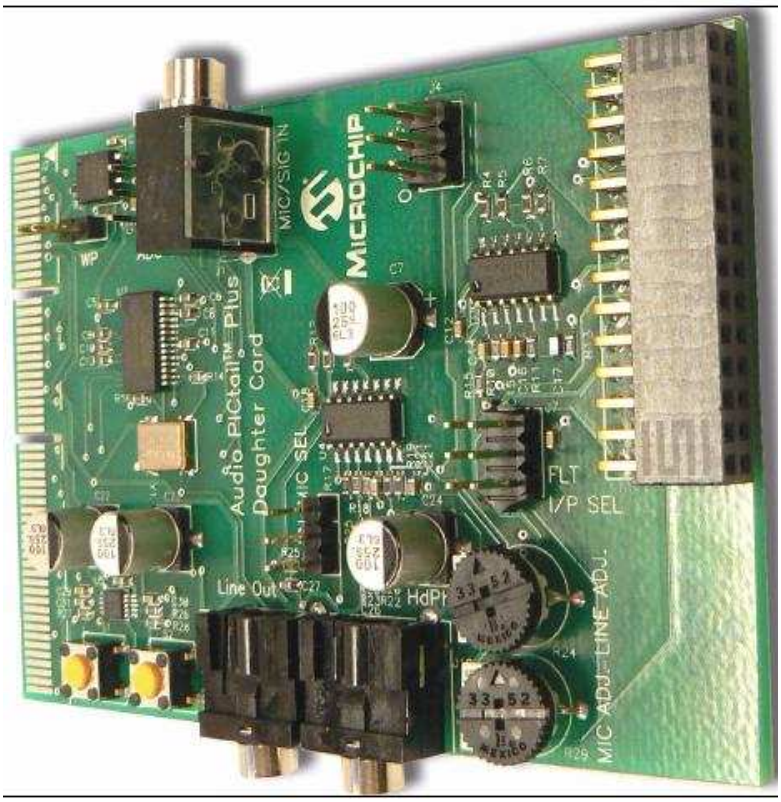

Figura 2.18. La placa de desarrollo Audio PICtail Plus.

Consta de conectores Jack de audio de entrada y salida, ganancias programables, líneas de entrada y salida adicionales, la interfaz elegida para comunicarse es la mencionada anteriormente DCI y el I2C. Para la comunicación con la memoria Flash se usa la interfaz SPI .

Es una placa de desarrollo muy potente para el procesado digital de audio al poseer un códec de prestaciones elevada y una memoria adicional para el almacenamiento de muestras de audio [24].

2.5.1.4. UIB-PC104.

Esta cuarta alternativa evaluada, diseñada por Ingenia S.A. Es una alternativa muy eficaz. Consta de LCDs, leds, pulsadores, interfaces de comunicaciones, en resumen de todo lo que se puede pedir a una placa de desarrollos para dsPIC, excepto el códec de audio.

La ausencia de un códec, como en la anterior dsPICDEM2, hace a UIB-PC104 menos atractiva que las otras alternativas, aunque si bien hay que decir a su favor que es una placa de desarrollo tanto o más completa que las otras tres en cuanto a recursos de todo tipo como LCDs o pulsadores.

Por ser ésta la placa que disponía el DTE, fue la placa elegida para el desarrollo del PFC. A continuación se explicará más a fondo todas sus características, y en el capítulo tercero cómo se subsanó el problema de la falta de un códec de voz.

2.5.2. UIB-PC104. INGENIA.

UIB-PC104 es una placa interfaz de usuario para poder interaccionar con un módulo de control mediante entradas y salidas, en nuestro caso particular, interaccionaremos con un módulo de control basado alrededor del dsPIC30F4011 de Microchip.

Con esta interfaz se necesitará menos tiempo y esfuerzo para proporcionar entradas al sistema e interpretar las salidas, por lo tanto, se convierte el UIB-PC104 en una herramienta de entrenamiento sencilla y eficiente.

UIB-PC104 dispone de distintas interfaces para comunicarse con el exterior y para poder llevar a cabo gran variedad de aplicaciones más allá del simple aprendizaje. Estas interfaces son: Leds, LCD alfanumérico y gráfico, potenciómetros, pulsadores y zumbador, además de conectores para proporcionar la alimentación, un bus de expansión con los pines más importantes y unos switches de configuración para hacer selección del funcionamiento de la placa.

El diseño de UIB-PC104 se basa en una arquitectura apilable y respetando la norma para placas PC/104, hace que nuestra placa interfaz pueda ser ampliada con otras placas, según las necesidades del usuario.

En la figura 2.19, se muestra el aspecto de la placa de desarrollo UIB-PC104, pudiéndose observar en ella tanto las interfaces del módulo, como el zócalo para el módulo de comunicaciones, el bus de expansión, el LCD alfanumérico, o la localización del zumbador, entre otros detalles.

Figura 2.19. UIB-PC104. [1].

2.5.2.1. Zócalo principal para el módulo de control (iCM4011).

Se trata del lugar dedicado para la ubicación del módulo de control, a través de este zócalo, el módulo accede a todos los periféricos de la placa, así como suministra toda la alimentación.

2.5.2.2. Alimentación.

UIB-PC104 posee distintas maneras de alimentarse, una de ellas es el interfaz USB y otra de ellas es una entrada de alimentación no regulada que corresponde al conector P1. Se usará la opción mediante la interfaz USB, ya que esta interfaz es necesaria para cargar los programas en el dsPIC30F4011 a través del *bootloader*.

UIB-PC104 también ofrece pines de alimentación de salida, tanto en el conector adicional J3 como en el bus de expansión, como se puede observar el la tabla 2.1.

Bus de expansión (J1)	**Pin**
VIN	2
V5	1
GND	16
Conector adicional (J3)	**Pin**
V5	2
GND	1

Tabla 2.1. Pines de salida de alimentación de UIB-PC104. [1]

2.5.2.3. Bus de expansión.

Debido a que UIB-PC104 tiene una estructura apilable, posee un conector de 40 pines siguiendo la forma y taladros de la norma PC/104. En la figura 2.20 se ven los pines que están presentes en el bus de expansión (J1):

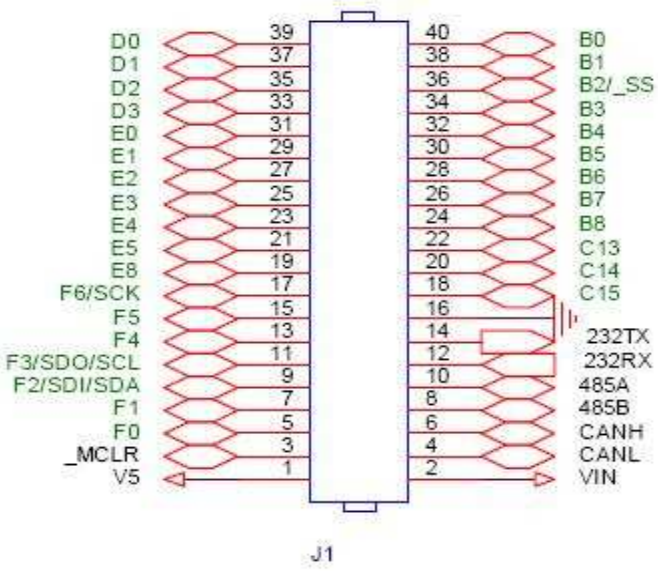

Figura 2.20. Bus de expansión de UIB-PC104 [1].

2.5.2.4. Comunicaciones.

En este punto se describirán brevemente las interfaces de comunicación que posee el UIB-PC104 junto al módulo de comunicaciones iCM4011.

Las comunicaciones disponibles son las siguientes:

- Interfaz USB accesible directamente a través del modulo iCM4011.
- A través del bus de expansión y de un conector DB9 macho UIB-PC104 ofrece las comunicaciones RS232, RS485 y CAN.
- Interfaces de comunicaciones SPI e I2C solo accesibles a través del bus de expansión.

En la tabla 2.2 se muestra el pineado del conector macho DB9 y la localización de los pines de dicho conector en la figura 2.21.

COMUNICACIONES	CONECTOR	PIN
No conectado	P2	1
232RX	P2	2
232TX	P2	3
CANH	P2	4
GND	P2	5
CANL	P2	6
485B	P2	7
485ª	P2	8
No conectado	P2	9

Tabla 2.2. Pineado del conector DB9 (P2).

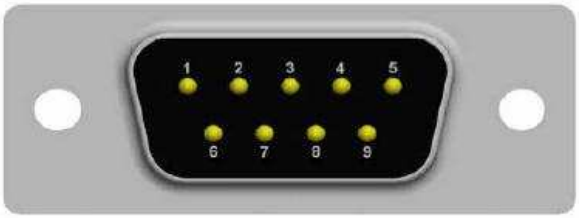

Figura 2.21. Aspecto del conector DB9 (P2).

Además, para los interfaces RS485 y CAN, UIB-PC104 contiene resistencias de terminación de bus de 120 ohmios accesibles mediante los switches del componente J2, como se muestra en la tabla 2.3.

Bus terminación	Componente	Posición
Resistencia terminación RS485	J2	3
Resistencia terminación CAN	J2	4

Tabla 2.3. Posición de la activación de las resistencias de terminación.

En la tabla 2.4 se muestra el pineado de las señales de las interfaces I2C y SPI dentro del bus de expansión.

Comunicaciones I2C	**Conector**	**Pin**
SDA	J1	9
SCL	J1	11
Comunicaciones SPI	**Conector**	**Pin**
SDI	J1	9
SDO	J1	11
SCK	J1	17
/SS	J1	36

Tabla 2.4. Pineado de las interfaces I2C y SPI en el bus de expansión (J1) [1].

2.5.2.5. LCDs.

Se dispone de dos LCDs que solo podrán usarse por separado, aunque cada LCD tiene su propio conector.

El LCD alfanumérico CFAH1602BTMLIP es un LCD *super Twist nematic* de 2 filas de 16 caracteres con retroiluminación blanca, usa un controlador HD44780 con interfaz de 4 o 8 bits.

El LCD gráfico CFAG12232DYYHN es un LCD *super Twist nematic* de 122x32 píxeles con retroiluminación verde, usa un controlador SED1520 que solo permite una interfaz de 8 bits.

En ambos LCD, el potenciómetro R2 se usa para ajustar el contraste de los LCDs. Además, estos LCD pueden desactivarse o activarse mediante el switch correspondiente del conector J2 como se muestra en la tabla 2.5.

LCDs	**Componente**	**Posición**
LCDs y potenciómetro R2 activación/ desactivación	J2	2

Tabla 2.5. Posición de la activación del LCD y del potenciómetro R2 [1].

2.5.2.6. Leds.

Se puede acceder desde la placa a 6 leds, dos por cada color: rojo, naranja, verde. Están conectados en cátodo común. El ánodo de cada led está conectado a un pin del zócalo principal M1 a través de una resistencia de 1120 ohmios.

Los pines del zócalo principal se podrían usar para otras funciones. Por eso habría que desactivar los leds mediante un switch del componente J2 como se muestra en la tabla 2.6.

Leds	Componente	Posición
Leds y zumbador activación/ desactivación	J2	1

Tabla 2.6. Posición de la activación de los leds y del zumbador [1].

2.5.2.7. Potenciómetros.

El potenciómetro R1 esta siempre activo y entrega un valor analógico entre 0 y 5 V al pin B7 del zócalo principal. El potenciómetro R2 como se comentó anteriormente se usa para el contraste de los leds, pero puede ser desactivado mediante el switch de la posición 2 de componente J2 como se ve en la tabla 2.5. Estaría conectado al pin B8 del zócalo principal.

LCDs	Componente	Posición
LCDs y potenciómetro R2 activación/ desactivación	J2	2

Tabla 2.5. Posición de la activación del LCD y del potenciómetro R2 [1].

2.5.2.8. Pulsadores.

En la figura 2.22 se muestra el esquema que usan los pulsadores. Son 12 pulsadores en 4 filas de 3 pulsadores cada una. Solo se usan 7 entradas al zócalo principal puesto que están codificadas por línea y columna.

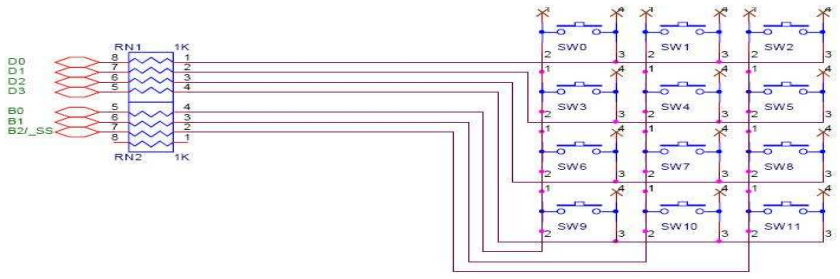

Figura 2.22. Esquema de los pulsadores de UIB-PC104 [1].

2.5.2.9. Zumbador.

El zumbador es capaz de generar tonos de entre 1KHz y 10KHz. El zumbador debe ser excitado a la frecuencia deseada a través del pin B7, que justo es la entrada del potenciómetro R1. Para ello existe un switch de activación, que es el mismo para los leds. Es la posición 1 del componente J2 como se ve en la tabla 2.6.

Leds	Componente	Posición
Leds y zumbador activación/ desactivación	J2	1

Tabla 2.6. Posición de la activación de los leds y del zumbador [1].

2.5.2.10. Switches de configuración

La tabla 2.7 muestra la información completa del componente J2, es decir de los 4 switches de configuración.

Módulo que controla el switch	Componente	Posición
LEDs y zumbador	J2	1
LCDs y Potenciómetro R2	J2	2
Resistencia 120 ohmios RS485	J2	3
Resistencia 120 ohmios CAN	J2	4

Tabla 2.7. Resumen de los switches de configuración [1].

2.5.2.11. iCM4011

El gran objetivo del módulo de comunicaciones es dar una gran variedad de opciones y la mayor capacidad de cálculo posible para hacer prototipos de todo tipo. Para ello consta de un procesador que cubre una necesidad de cálculo media. Este es un procesador digital de señal de última generación, el dsPIC30F4011 de Microchip junto a unos *transceivers* para los estándares de comunicación más usuales como el USB, RS232, CAN…

El procesador viene programado con un *firmware* de Ingenia que permite reprogramar dicho dispositivo, así se podrá cargar cualquier programa a través del *bootloader* que usa la interfaz USB para cargar dicho programa sin necesidad de programador externo. También puede ser programado o depurado mediante herramientas como ICD2, puede ser alimentado a través de la interfaz USB, y ofrece funcionalidad plug & play completa.

Como se muestra en la figura 2.23, iCM4011 dispone de los siguientes bloques funcionales: regulación de alimentación, oscilador, generador de reset, procesador, e interfaces.

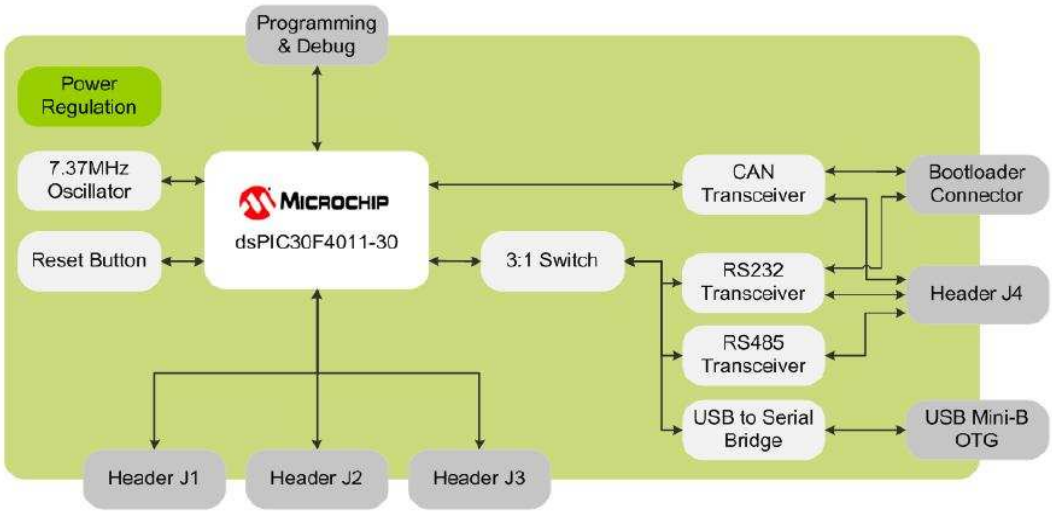

Figura 2.23. Esquema del iCM4011 [2].

2.5.2.11.1. Alimentación.

Se puede alimentar iCM4011 de tres modos distintos: el modo regulado que se usa cuando el usuario disponga de una tensión entre 6 y 12 V; el modo no regulado que se usa cuando en usuario disponga de una tensión entre 3.3 y 5 V y no se use la interfaz USB, y el modo USB que se usa para alimentar a través de la interfaz USB.

Este tercer método de alimentación, es por su sencillez y por el hecho de tener que usarla para programar el dsPIC30F4011, es el método usado, Así no se estará pendiente de unas restricciones que existen en los otros modos, pero si se tendrán que instalar los drivers correspondientes la primera vez que se alimente mediante el interfaz USB [2].

2.5.2.11.2. Unidad de proceso.

La unidad de proceso es el dsPIC30F4011 y el oscilador es de 7,37Mhz, que posee las características que se muestran en la tabla 2.8:

Tipo de procesador	RISC MCU + DSP (Hasta 30MIPS)
Memoria Flash de programa	48 KBytes
Memoria RAM de datos	2 KBytes
Memoria EEPROM	1 KByte
Puertos de E/S	30
Conversores A/D	9 canales, 10 bits por muestra.
Características de control de motor	6 canales PWM, Quadrature Encoder Interface
Otras características	5 temporizadores, 4 entradas para captura o comparación, brown out, reset
Pines disponibles externamente	Todos los del dsPIC , excepto AVDD y OSC1/CLKIN

Tabla 2.8: Características del dsPIC30F4011 de iCM4011 [2].

2.5.2.11.3. Interfaces de comunicación.

→ I2C y SPI.

Estos protocolos de comunicación serie síncrona están presentes en iCM4011 y son accesibles a través del bus de expansión de UIB-PC104 a través de los conectores J1 y J3 de iCM4011.

I2C es un protocolo de 2 hilos, y SPI es un protocolo de 3 o 4 hilos, según las exigencias de la comunicación. Estos protocolos ya se han detallado brevemente en este capítulo. Además, el protocolo SPI se detallará en el capítulo tercero, por ser el usado para conectar el dsPIC30F4011 y el Si3000.

→ CAN.

iCM4011 ofrece un *transceiver* CAN compatible con la norma ISO-11881 que une la unidad de proceso y el bus CAN, siendo la velocidad máxima de 1Mb/s y el máximo número de nodos 112.

→ Comunicaciones serie.

iCM4011 ofrece tres tipos de comunicaciones serie, estos modos son RS485, RS232 y USB, de las cuales solo una podrá usarse en cada momento, ya que están multiplexados a la entrada de la AlternateUART1 del dsPIC30F4011.

Para seleccionar el modo de transmisión elegido existe un jumper en el conector J8 como se muestra en la tabla 2.9. Realmente con el jumper solo se selecciona la recepción, puesto que la transmisión se hace indistintamente de la posición del jumper.

Interfaz	Conector	Pins
RS485	J8	1-2
RS232	J8	3-4
USB	J8	5-6

Tabla 2.9. Funcionalidad del jumper de selección de comunicación serie. [2].

2.6. CONCLUSIONES.

El capitulo 2 da una amplia visión a la familia dsPIC3xF de Microchip y a la placa de desarrollo UIB-PC104 de Ingenia S.A., ya que se ha estudiado con detalle cada bloque de los que consta, sea mas o menos importante para nuestro proyecto, quizás dejando un poco de lado la programación en ensamblador, es decir, modos de direccionamiento y instrucciones en ensamblador, ya que las aplicaciones futuras serán diseñadas en un lenguaje de alto nivel.

Además se repaso la evolución de los microcontroladores de Microchip, así como en las distintas opciones del mercado de las placas de desarrollo, para conocer como esta el mercado y conocer que otras opciones existen. Esto es interesante para saber si hay otras opciones mejores para el objetivo de este PFC.

Quizás la parte más interesante del aprendizaje de los dsPIC3xF es la similitud que poseen todos los microcontroladores de Microchip y el modo de programarlos, así como las herramientas que se usan para hacerlo. Con esto entiende que este aprendizaje se podría extender a que se aprendió a usar de modo básico cualquier microcontrolador de Microchip.

Del análisis de las características del dsPIC30F4011 se extraen carencias de este. Existen otros dsPIC mucho más potentes como los dsPIC33F que tienen más velocidad de cálculo, e incluso algunos ya sean de la familia dsPIC30F o dsPIC33F poseen como se dijo la interfaz DCI que hubieran hecho a las aplicaciones más potentes.

CAPÍTULO 3: Si3000.

3.1. INTRODUCCIÓN.

Debido a las necesidades de las aplicaciones de audio a desarrollar, lo ideal hubiese sido tener una placa de desarrollo con un códec incorporado, pero el hecho en sí de no tener dicho códec, hizo al PFC completo e interesante.

La creación del PCB necesario para incorporar el códec a la placa de desarrollo constó de varias fases: comprender qué es realmente un códec de audio, qué bloques le caracterizaban, qué parámetros tenía y en que rango se movían, como la frecuencia de muestreo.

Una vez comprendido que el códec consta de dos conversores, uno digital-analógico y otro analógico-digital, que la frecuencia máxima es de 48KHz debido a las características de la señal de audio, que podían tener etapas para adaptar la señal de entrada analógica de audio, se pasó a buscar cómo estaba el mercado de dichos componentes.

Una vez elegido el Si3000, se pasó al estudio de éste y de su módulo SPI para incorporarlo a la placa de desarrollos UIB-PC104 mediante su bloque de interfaz digital SPI.

Para acabar, mediante PCB123 (herramienta de diseños de PCB) se hizo el diseño de la PCB y por último en el laboratorio se hizo todo el proceso de creación de la PCB, desde la insolación de las placas positivas hasta el soldado del último componente.

3.2. NECESIDAD DE UN CÓDEC DE AUDIO.

3.2.1. ¿QUÉ ES UN CÓDEC DE AUDIO?

La señal de audio que se genera a través de elementos como un micrófono, tiene naturaleza analógica, es decir, la información está codificada en una señal de tensión eléctrica continua variable, con unos limites de tensión dependientes del componente físico que se use para convertir la voz en señal analógica, como el propio micrófono o una señal directa del canal de audio de un PC.

Debido a la naturaleza analógica de la señal de audio, no puede ser tratada directamente por un procesador, puesto que estos, solo entienden de símbolos digitales, por lo que se necesita transformar dicha señal analógica a otra señal digital equivalente, intentando perder la mínima información posible para el posterior procesado de la señal de audio.

Pero la conversión no solo es en el sentido analógico-digital, porque una vez se procesó la señal de audio digital, se necesita volverla a convertir a analógica, para poder oír o comprobar mediante un osciloscopio el resultado del procesado.

Existen parámetros a tener en cuenta, como son la frecuencia de muestreo de la señal analógica, es decir, la frecuencia con la que se toman las muestras analógicas y se hará la posterior conversión a digital. El segundo parámetro es el número de bits que se usa para codificar la señal analógica. Estos dos parámetros, cuanto más elevados sean, mayor será la calidad conseguida en la conversión.

Por lo tanto, se necesita un circuito integrado que haga esta doble función, ese circuito es un códec. El códec consta básicamente de un par de conversores, uno de ellos hace la conversión de la señal analógica a la señal digital, además posee otro conversor para la conversión opuesta. Si solo hiciera una de las conversiones, se trataría simplemente de un conversor.

Además de los conversores, un códec debe poseer otras características útiles y necesarias para cualquier aplicación, como una interfaz digital de comunicación para la transmisión y recepción de las muestras digitales, necesaria para comunicarse con el procesador, que hará el procesado digital de señal.

Y como la señal digital que se tratará en este PFC es la señal de audio, se necesitará un códec de audio, que hará las dos conversiones de una señal de audio, en los parámetros que la señal de audio utiliza, es decir, nunca tendrá una frecuencia de muestreo muy superior a 48 KHz.

Un códec consta de elementos que ayudan al mejor acople de la señal analógica con las características del audio, porque incorpora etapas de ganancia programable a la entrada de los micrófonos, y etapas de atenuación variable para la salida hacia los altavoces, esto se debe a que no todos los micrófonos y altavoces trabajan en el mismo rango de valores analógicos.

3.2.2. ¿POR QUÉ SE NECESITA AÑADIRLO A UIB–PC104?

El PFC consiste en crear un entorno para la asignatura de Laboratorio de Sistemas Digitales Avanzados, similar al que disponía la asignatura anteriormente, por lo tanto, se necesita una plataforma que proporcione una entrada y una salida analógica de audio, así como un procesador, para hacer el procesado digital de señal de audio.

Para el procesador se dispone de una plataforma de desarrollo UIB-PC104 de Ingenia (Ingeniería e Integración Avanzadas) S.A., que incorpora un dsPIC30F4011 de Microchip.

El dsPIC30F4011 cuenta con un bloque conversor analógico digital de 10 bits con 9 canales distintos de entrada a una velocidad de 1000 ksps (kilo símbolos por segundo). Con estas características se podría haber cubierto las necesidades para la conversión A/D, aunque hubiese sido un muestreo de baja calidad debido a los 10 bits de codificación. Pero esta opción conllevaba la necesidad de usar otro conversor D/A [3].

Por este motivo, se tomo la decisión de buscar un códec de audio, que incorporará la doble funcionalidad con conversores A/D y D/A para llevar a cabo las conversiones necesarias. Basándose en el códec se diseñará toda la adaptación de la señal de audio necesaria para el procesado digital de audio, que se llevará a cabo en el PFC en un mismo PCB.

3.3. VARIEDAD DEL MERCADO DE LOS CÓDEC.

3.3.1. MODELOS EVALUADOS.

Una vez se vio la necesitad de añadir un códec para la aplicación, es necesario hacer una evaluación de las alternativas del mercado, para posteriormente hacer la elección adecuada.

El mercado ofrece muchos códecs de prestaciones muy altas, incluso demasiado altas en la mayoría de los casos, así que lo primero que se debe hacer, es saber qué características posee la placa de desarrollo UIB-PC104 antes de comenzar la búsqueda.

Una primera característica a evaluar, son los interfaces digitales de comunicación que ofrece el dsPIC30F4011, y se ve que existe una amplia variedad de interfaces como SPI, I2C, CAN, UART, etc. Pero por su contra, no posee la interfaz digital más importante para un códec de audio, la interfaz DCI (*Digital Converter Interface*), basado en el protocolo

SPI, pero mejorado con transferencias de paquetes de datos de 16 bits y múltiplos de 16 bits, así como trabajar a velocidades estándar de audio [3].

De las posibilidades que posee el dsPIC30F4011, hay que comprobar cuáles usan normalmente los códecs, y se observa que éstas son básicamente I2C y SPI, así que se centró la búsqueda en estas dos interfaces digitales.

La segunda característica a evaluar es la velocidad de muestreo. Esto es importante puesto que se puede seleccionar un códec demasiado rápido para el dsPIC30F4011, entonces el procesado digital no seria correcto, al llegar las muestras al dsPIC30F4011 antes de que el procesado hubiese terminado. En la evaluación de las placas de desarrollo, se vió como se usaba en muchas el Si3000, que es un códec de voz, con lo que se intuye, que no se pueden usar códec demasiados rápidos para este tipo de dsPIC30F4011.

Por último y no menos importante hay que ver cuántos bits proporciona cada códec, esto podría decantar la elección entre dos códec de características similares, pero con distinto número de bits, puesto que se perdería menos información en el procesado digital.

Se analizaron componentes de 4 fabricantes: *AKM (Asahi Kasei Microsystems), Analog Device, Texas Instruments y Silicon Laboratories*. Se muestran los resultados de la comparativa en las tablas 3.1, 3.2, 3.3, 3.4.

Característica \ Modelo	**AK4564**	**AK4631**	**AK4640**	**AK4642**
Resolución	16	16	16	16
Frecuencia de muestreo	50KHz	48KHz	48KHz	48KHz
Número de pines	48	28	57,52	32
Interfaces disponibles	SPI	SPI	SPI	I2C, SPI
Precio	-	-	-	-

Tabla 3.1. Comparativa códec de AKM [15][16][17][18].

Característica \ Modelo	**AD73311**	**AD74111**
Resolución	16	16,20,24
Frecuencia de muestreo	64KHz	48KHz
Número de pines	20	16
Interfaces disponibles	SPI	SPI
Precio	3,47Euros*	4,6 Euros

Tabla 3.2. Comparativa códec de Analog Devices [25][26].

***Precio del AD73311 para más de 200 unidades.**

Característica \ Modelo	Aic111	Tlv320aic12	Tlv320aic15
Resolución	16	16	16
Frecuencia de muestreo	40KHz	104KHz	104KHZ
Número de pines	32	30,32	30
Interfaces disponibles	SPI	SPI, I2C	SPI, I2C
Precio	8 Euros	2.32Euros *	4.19Euros *

Tabla 3.3. Comparativa códec de Texas Instrument [27][28][29].

***Precio para la compra de 1000 unidades.**

Característica \ Modelo	Si3000
Resolución	16 bits
Frecuencia de muestreo	12,5KHz
Número de pines	16
Interfaces disponibles	SPI
Precio	1.255 Euros *

Tabla 3.4. Comparativa códec de Silicon Laboratorios [5].

*** Precio para 1000 unidades.**

3.3.2 ¿POR QUÉ EL SI3000?

Hay distintos motivos por los que se eligió el Si3000 para ser el códec que se usará en este PFC, por ejemplo, poseer una interfaz de comunicaciones digital como la interfaz SPI, tener bloques para adaptar la señal de entrada y salida de audio, y ser el códec usado en placas de desarrollo parecidas a la usada en este PFC.

Las dos primeras características las cumplían todos los códec evaluados, solo había dudas del protocolo a usar, siendo el SPI más sencillo, con lo que ya se tiene la primera ventaja del Si3000.

Placas de desarrollo como DSPIC DEM 1.1 ya evaluadas, que usan PIC de la gama dsPIC30F, usan como códec de audio el Si3000. Esto hizo prever que un códec de mayor frecuencia de muestreo podría ser demasiado rápido para aplicaciones con procesado digital con mucho cálculo en el Motor DSP, como el filtrado digital mediante filtros FIR e IIR.

Otro motivo para decantarse definitivamente por el Si3000, fue la facilidad para encontrar muestras gratuitas directamente desde el fabricante del Si3000 a través de uno de sus proveedores en España.

3.4. EL SI3000.

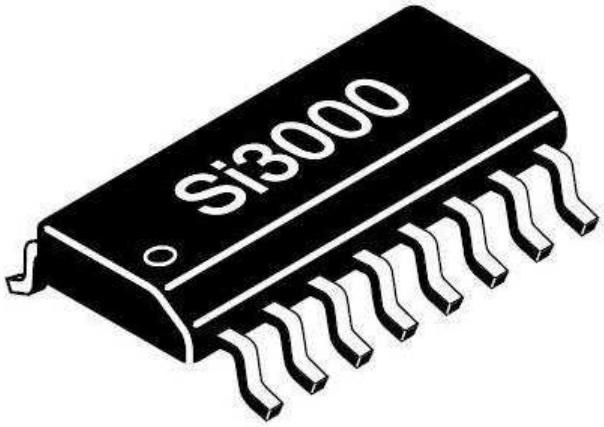

Figura 3.1. El aspecto exterior de un Si3000 [5].

3.4.1. CARACTERÍSTICAS GENERALES.

El Si3000 es un completo códec cuyo ancho de banda es el de voz, (12,5khz). Ofrece una gran integración y posee facilidades como poder programar la ganancia de entrada y la atenuación de salida. Esta adaptado para la entrada del micrófono y para una salida de altavoces, así como determinar la frecuencia de muestreo.

Sus características más importantes son:

- Rango dinámico de 84 dB para el ADC y el DAC.
- Muestreo de 4-12Khz.
- Hasta 30dB de ganancia para el Micrófono.
- Ganancia de entrada y salida programable entre -36dB y 12dB.
- Resistencia de 32 ohmios para los altavoces.
- Interfaz directa con DSPs, se trata de la interfaz SPI.
- Conexión directa con otros chips de Silicon Laboratories como Si3035, Si3034, Si3044.
- Encapsulado con 16 pines [5].

3.4.2. DESCRIPCIÓN FUNCIONAL.

Para comenzar se detalla el significado y uso del pineado del Si3000 en la tabla 3.5.

Nº PIN	Nombre	Descripción.
1	SPKRR	**Speaker Right Output.** Salida analógica para el canal derecho con una resistencia de 50 ohmios.
2	MBIAS	**Microphone bias output.**
3	HDST	**Handset Input/Output.** Entrada y salida para el teléfono.
4	SDI	**Serial Port Data In.** Pin de entrada para las muestras en el protocolo digital SPI.
5	SDO	**Serial Port Data Out.** Pin de salida para las muestras en el protocolo digital SPI.
6	/FSYNC	**Frame Sync Output.** Pin del protocolo SPI que indica el inicio de una trama.
7	MCLK	**Master Clock Input.** Reloj de entrada para generar SCLK, proviene de un micro o reloj externo.
8	SCLK	**Serial Port Bit Clock Input/Output.** Pin del protocolo SPI que indica cuando son validos los datos en SDI o SDO.
9	/RESET	**Reset.** Manda al Si3000 a su estado inicial, sus registros se restauran con su valor por defecto.
10	MIC	**MIC Input.** Entrada adaptada para tensión analógica de un micrófono. Contiene una ganancia programable de 0, 10, 20 o 30 dB.
11	LINEI	**Line Input.** Entrada normal con una ganancia programable de 0, 10 o 20 dB.
12	VC	**Digital Supply Voltaje.** Pin para alimentar el Si3000 con una tensión digital entre 3.3V y 5V.
13	VA	**Analog Supply Voltaje.** Pin para alimentar el Si3000 con una tensión digital entre 3.3V y 5V.
14	GND	**Ground.** Tierra.
15	LINEO	**Line Output.** Entrada normal con una ganancia programable de 0, 10 o 20 dB.
16	SPKRL	**Speaker Left Output.** Salida analógica para el canal derecho con una resistencia de 50 ohmios.

Tabla 3.5. Características del pineado del Si3000 [5].

En la figura 3.2 se observa su diagrama de bloques del Si3000, los bloques que lo componen son: Entradas posibles, salidas posibles, conversores, bloques de ganancia digital, interfaz digital, pines que intervienen en la transmisión.

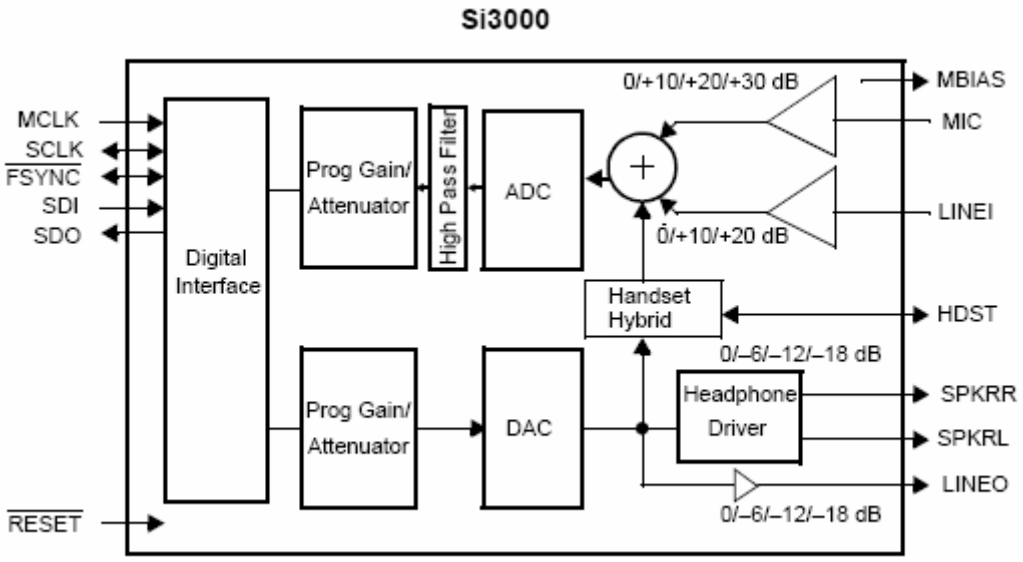

Figura 3.2. Diagrama de bloques del Si3000 [5].

La entrada analógica puede provenir de un micrófono de ganancia configurable mediante un registro del Si3000, por una línea de entrada o por la entrada de un teléfono (handset). Todas las entradas se mezclan antes de entrar en el conversor A/D de 16 bits, pero mediante los registros de configuración, se puede hacer que algunas de estas entradas no se mezclen, haciendo que no se tengan en cuenta en la mezcla.

Después del conversor A/D, el Si3000 posee un bloque para programar una ganancia, para comunicarse con el dsPIC30F4011 al nivel digital adecuado, y así aprovechar todo el rango de valores de los 16 bits disponibles. Una vez procesado el dato en el dsPIC30F4011, éste vuelve al códec, pero antes de pasar al conversor D/A, el Si3000 ofrece otra ganancia programable.

Por último, la señal de salida analógica podrá ir en varias direcciones, los altavoces, la línea de salida, y la salida del teléfono. Todas tienen sus correspondientes ganancias de atenuación de señal.

Los registros del Si3000 sirven para configurar los valores de las ganancias de las distintas entradas, las atenuaciones de las distintas salidas, las entradas y salidas activas, así como qué señales hay que dejar inactivas según las necesidades del proyecto.

Además, se configuran las ganancias programables de transmisión y recepción de la interfaz digital, también configura la frecuencia de muestreo, el divisor del PLL, los filtros disponibles, entre otras características [5].

3.4.3. INTERFAZ DIGITAL DEL SI3000.

El Si3000 puede actuar de dos modos distintos respecto el control de la transmisión SPI: esclavo y maestro. Siendo el trabajo en maestro de dos maneras distintas, dependiendo de la funcionalidad de la señal FF del protocolo SPI.

Para hacer la configuración del modo no se usan registros, se usan unas resistencias de *pull - up* en los pines SDO y SCLK. Estas serán leídas en el primer ciclo de la señal MCLK tras el RESET al Si3000. Las distintas configuraciones se ven en la tabla 3.6.

Modo	SCLK	SDO	Descripción
0	0	0	Modo maestro con FSYNC a nivel bajo durante los 16 bits de cada trama.
1	0	1	Modo maestro con FSYNC a nivel bajo el primer bit de cada trama.
2	1	0	Modo esclavo.
3	1	1	Reservado.

Tabla 3.6. Modos de trabajo del Si3000 [5].

El modo de operación usado para este PFC es el modo 1. Por ser un modo maestro, se necesita un reloj para excitar el Si3000 a través del pin MCLK, siendo el reloj base de la transmisión una salida hacia el dsPIC30F4011 desde el Si3000, quien lleva el control de la transmisión SPI.

La frecuencia de MCLK y los registros 3 y 4 del Si3000 determinan la frecuencia de muestreo FS. SCLK es 256 veces más rápido que FS, como se vera en el protocolo SPI que se detallará a continuación.

Entre el dsPIC30F4011 y el Si3000 la información se envía en dos tramas de 16 bits, siendo estas tramas transmitidas en las dos direcciones al mismo tiempo, a través de los pines SDI y SDO de ambos dispositivos. En la figura 3.3 se aprecia como es el esquema temporal de la transmisión SPI del Si3000, para cualquiera de los modos de configuración. Se ve la alternancia de las tramas de datos y las tramas de configuración. El pulso de FSYNC es genérico pues como se comentó, puede ser de dos modos distintos y generado por el códec (maestro) , o generado por el otro dispositivo que trabaje en la transmisión SPI.

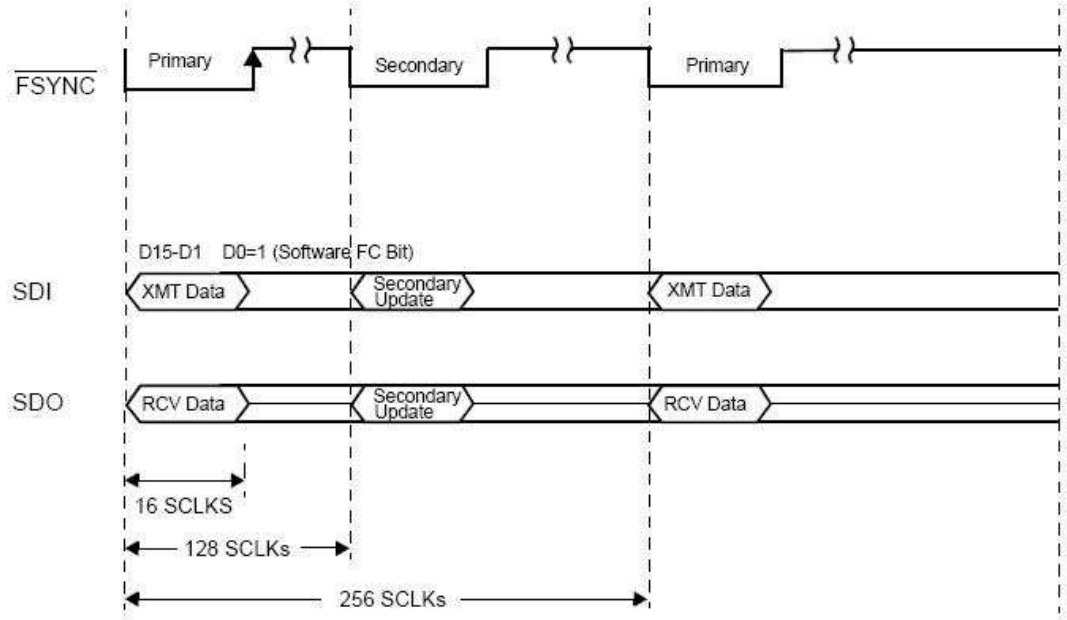

Figura 3.3. Esquema de tiempos en la transmisión SPI del Si3000 [5].

La primera trama es la que hace la transferencia de las muestras digitales de audio y está siempre presente. La segunda trama se usa para acceder a los registros del Si3000 y así poder configurarlo. Esta trama no siempre está presente, y necesita una señal para activarla, siendo el bit 0 de la primera trama de la transferencia dsPIC30F4011 – Si3000, quien activa la segunda trama, por lo que, cuando se pasan muestras de audio analógico hay 16 bits de información, pero al tratarlo por el dsPIC30F4011 solo existen 15 bits de audio, puesto que, la última es simplemente este flag, y habrá que ponerla a cero para no desconfigurar el Si3000.

La estructura de la segunda trama tiene 16 bits, de los cuales, uno de ellos es para indicar lectura o escritura del registro, siendo 1 para lectura y 0 para escritura, 5 bits para indicar la dirección del registro que se desea acceder, y 8 bits que indican el valor para escribir el registro, o el valor de lectura devuelto de dicho registro. Estos 8 últimos bits, se devolverán por SDI en el ciclo de escritura y SDO en el ciclo de lectura.

En las siguientes figuras, se muestra un ejemplo de la transmisión de la trama de configuración. La figura 3.4 muestra una trama de configuración de lectura y la figura 3.5 muestra una trama de configuración de escritura [5].

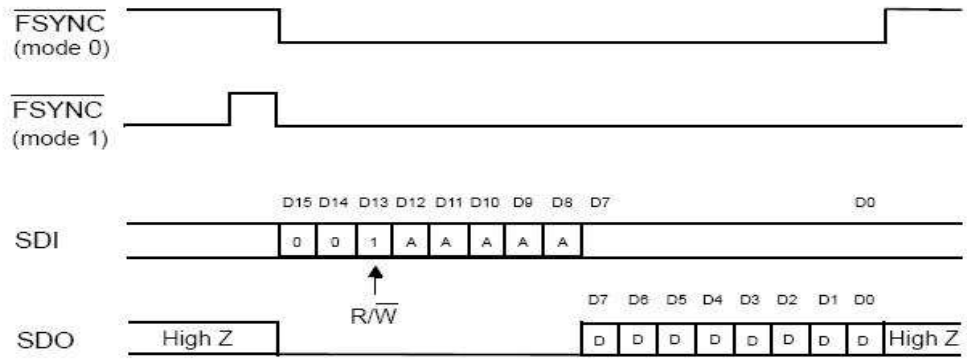

Figura 3.4. Ejemplo de trama de configuración de lectura para el Si3000 [5].

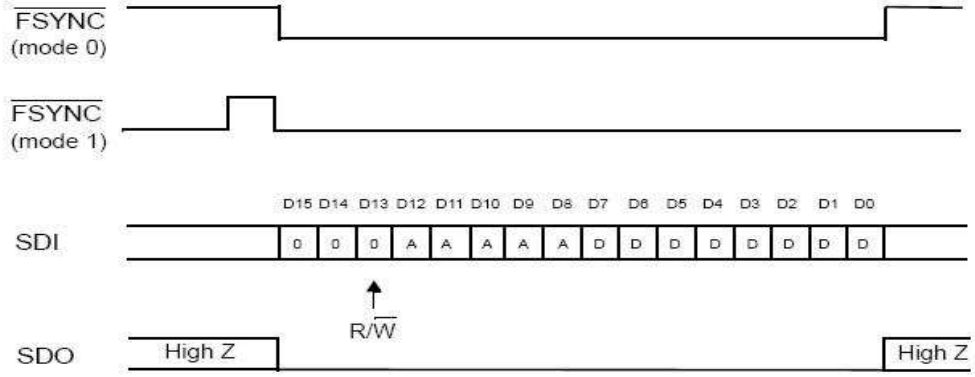

Figura 3.5. Ejemplo de trama de configuración de escritura para el Si3000 [5].

3.5. DISEÑO.

3.5.1. PROTOCOLO SPI.

A continuación, se detalla a nivel general qué es la comunicación SPI, cómo funciona dicho protocolo, las señales que posee, de modo más particular cuáles son los módulos que intervienen en la transferencia de muestras de audio de este PFC, y cómo están configurados dichos módulos para llevar a cabo el intercambio de datos.

3.5.1.1. Teoría del SPI. Señales.

El Bus SPI (*Serial Peripheral Interface*) es un estándar de comunicaciones usado principalmente para la transferencia de información entre circuitos integrados en equipos electrónicos.

El bus SPI es un estándar para controlar casi cualquier electrónica digital, que acepte un flujo de bits serie regulado por un reloj [6].

Básicamente, este estándar funciona con dos líneas de datos, SDI y SDO, que van cruzados entre los distintos dispositivos que efectúan la transmisión de datos. También hay una señal de reloj base de la transmisión SCLK, la cual lleva la velocidad de un bit, y servirá para sincronizar ambos dispositivos. Por último una señal FF, que será la encargada de iniciar la transmisión, poniéndose a un cierto nivel durante un ciclo de SCLK, o una señal que estará a un cierto nivel durante toda la transmisión de datos, permaneciendo en el estado contrario en reposo, es decir, sin transmisión.

La gran ventaja de un bus serie, es minimizar el número de conductores, pines y el tamaño del circuito integrado, reduciendo así el coste de fabricación, montar y probar el componente electrónico.

Casi cualquier dispositivo digital, puede ser controlado con la combinación de las cuatro señales que forman el interfaz SPI.

Los dispositivos que poseen esta interfaz digital, se diferencian en un número amplio de peculiaridades de la transmisión SPI, sobre todo, en cómo se inicia la transmisión mediante la señal FF.

Unos leen el dato cuando el reloj sube, otros cuando el reloj baja, algunos lo leen en el flanco de subida del reloj y otros en el flanco de bajada. Algunos dispositivos tienen dos relojes, uno para capturar o mostrar los datos y el otro para el dispositivo interno.

3.5.1.2. Configuración del módulo SPI del códec Si3000.

Las variedades de interfaz digital en el Si3000 son más limitadas, es por ello que se analizará antes que el módulo SPI del UIB-PC104, que es más flexible.

Tiene dos posibilidades de funcionamiento básicas, maestro o esclavo, incluso el modo maestro, posee dos posibles relojes distintos para la señal FSYNC (señal que controla la sincronización de envió y recepción de datos).

En la figura 3.6 se detallan las distintas opciones mencionadas, donde los modos 0 y 1 son las dos variantes de FSYNC en modo maestro, siendo el modo 2 el modo esclavo [5].

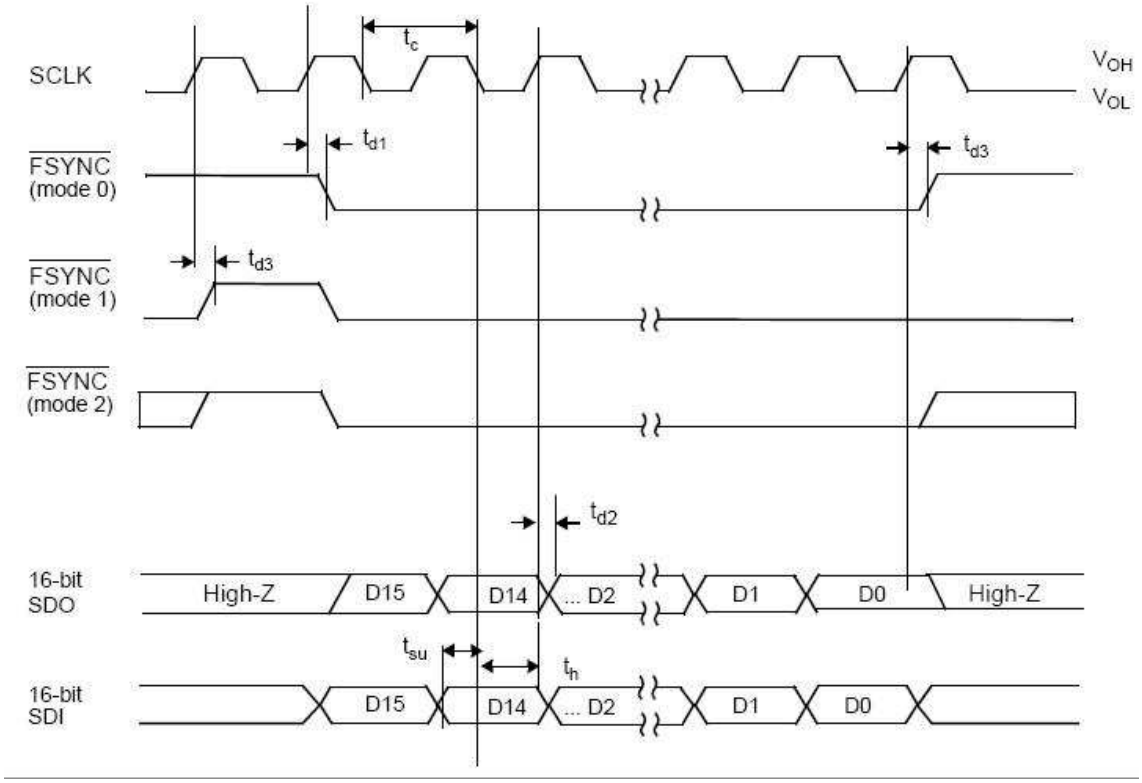

Figura 3.6. Modos del SPI del Si3000 en función de la señal FSYNC [5].

La configuración usada para la transmisión fue el modo 1, actuando el códec en modo maestro. Esta elección se debe, a la mayor flexibilidad del códec al generar el reloj que lleva la transmisión (SCLK), y así ajustar la frecuencia de muestreo a la velocidad máxima del Si3000, 12,5KHz. Con el dsPIC30F4011 de maestro, se lograba más velocidad, pero la escala más cercana a 12,5KHz es alrededor de 8KHz, y así se pierde calidad de audio [5][3].

También anotar que al usar el códec Si3000 como maestro, se necesita excitar el códec con un reloj externo, en este caso se uso un reloj de 10Mhz.

3.5.1.3. Configuración del módulo SPI del dsPIC30F4011 de UIB-PC104.

La interfaz SPI en el dsPIC30F4011 tiene más variantes. Se pueden elegir entre modo maestro o esclavo, entre usar una transmisión con o sin el hilo FF, e incluso ya sea esclavo o maestro, se podría configurar qué dispositivo de los que participan en la transmisión SPI lleva el control y manda la señal FF. A pesar de tener tantas opciones de configuración de la transmisión SPI, no fue la elegida pues las frecuencias que puede generar para controlar la transmisión SPI como maestro son pocas, aunque puede trabajar mucho más rápido que el Si3000.

Para una correcta conexión con el Si3000 y que la transmisión sea correcta, se necesita usar al dsPIC30F4011 en modo esclavo de 4 hilos, donde la señal FF es generada por el Si3000, es decir, la señal FF del dsPIC30F4011 es una entrada enviada por el Si3000.

Ahora queda saber qué forma tiene el reloj base, para hacer una correcta lectura de los datos, es decir, se decidirá el momento justo donde se capturará y se escribirá cada dato por los hilos SDI y SDO para una correcta transmisión.

El dsPIC30F4011 tiene una gran variedad de opciones para seleccionar el momento de captura de los bits en las líneas SDO y el envió en SDI, pero por el contrario el Si3000 no posee información al respecto, con lo que la configuración se hizo cambiando los valores de captura del dsPIC30F4011 hasta una correcta transmisión, comprobada mediante la calidad del audio [3].

3.5.2. ESQUEMÁTICO.

Una vez presentado el protocolo SPI, se llevó a cabo un diseño, donde se conectan físicamente ambos dispositivos, el dsPIC30F4011 a través de su módulo SPI, y el códec Si3000 a través de los pines de la interfaz digital SPI que posee.

Pero no todo es tan sencillo como conectar las cuatro señales del protocolo, hay más detalles a tener en cuenta, como por ejemplo, los modos de operación del Si3000 que se seleccionan con resistencias de *pull – up*.

Como el Si3000 actúa como maestro, se necesita un oscilador que excite el códec, también se necesitan unos circuitos de adaptación para el conector JACK del micrófono y otro para el conector JACK del altavoz, un conector de cuarenta pines para unirse al bus de expansión de la placa UIB-PC104, y unos condensadores para estabilizar las tensiones de alimentación.

Para llevar a cabo el diseño esquemático se usó la aplicación PCB123, y su herramienta de creación de diseños esquemáticos. En la figura 3.7 se aprecia el resultado que posteriormente se detallará.

En dicho esquemático se observan varios bloques: a la izquierda el conector de 40 pines para comunicarse con el dsPIC30F4011, solo tiene conectadas los pines usados. A la derecha se ven los conectores JACK de audio y su circuito de acoplamiento de señales. Abajo se ve el oscilador de 10MHz usado, y en el centro el Si3000.

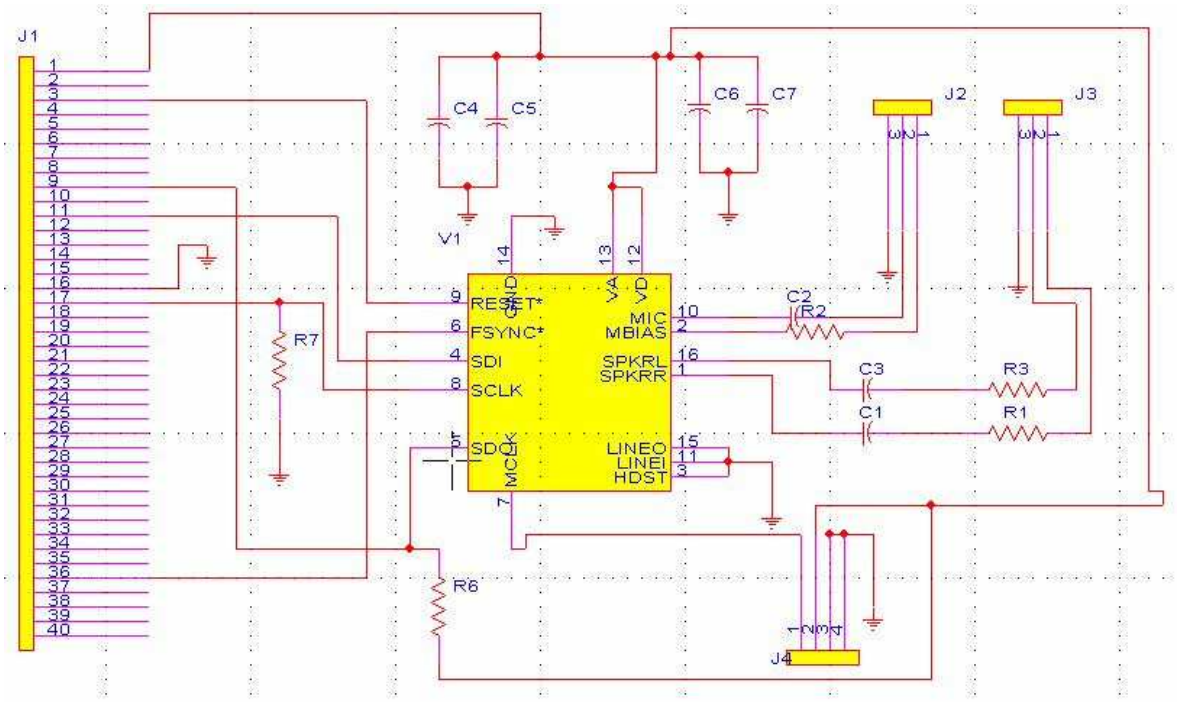

Figura 3.7. Esquemático realizado mediante PCB123.

- Componente J1.

Corresponde con un conector de 40 pines. Físicamente son dos filas de 20 pines, y su utilidad es poder acceder a las señales necesarias del bus de expansión de la placa de desarrollo UIB-PC104 de manera más sencilla.

Las señales que se necesitan del bus de expansión son las del protocolo SPI, (SDO, SDI, SCLK, FF), las señales de alimentación VCC y GND, y por último la señal de RESET que usa el códec Si3000, para iniciarse correctamente.

- Componente J2 y J3.

Corresponden con los conectores JACK de audio, tanto para el canal de entrada, como para el canal de salida.

- Componente J4

Corresponde con un oscilador de 10MHz, necesario para excitar el códec Si3000. Cuenta con 4 patillas: VCC, GND, reloj de salida de 10MHz. y un pin para habilitar la funcionalidad del códec, en nuestro caso, este pin esta puesto a GND para habilitar el oscilador en todo momento.

- Condensadores C4, C5, C6 y C7.

Es un circuito de 4 condensadores en paralelo entre VCC y GND para estabilizar la alimentación que le llega al Si3000. La idea de este circuito se recogió del manual del Si3000 en un ejemplo de uso del Si3000.

- Resistencias R6 y R7.

Corresponden a las resistencias de configuración del Si3000. Al ser la configuración usada del Si3000 el modo 1, las resistencias a usar son una a VCC para el pin SDO, y otra a GND para el pin SCLK.

- Condensadores C1 y C3 y Resistencias R1 y R3.

Son dos pequeños circuitos de adaptación para la salida hacia el conector JACK de salida, es decir, para cada uno de los canales analógicos de salida de audio.

- Condensador R2 y Resistencia C2.

Corresponde a un circuito de adaptación para el conector JACK de audio de entrada, es decir para la señal analógica de entrada.

- Componente V1- Si3000.

Se trata del Si3000, con sus 16 pines, algunos de ellos puestos a GND puesto no son usados. Mediante la figura 3.8, se observará la distribución del pineado del Si3000.

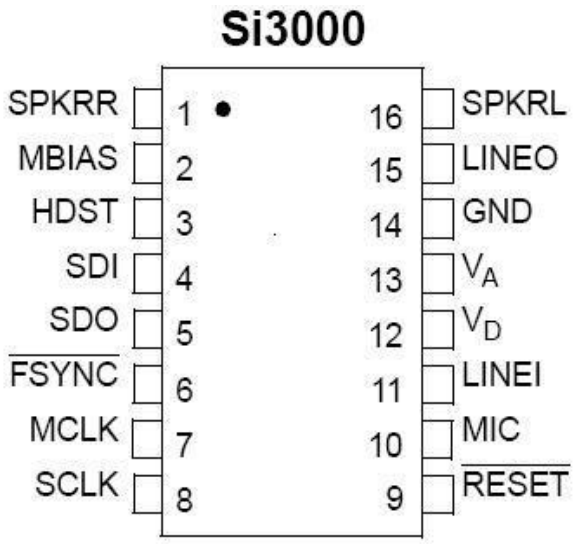

Figura 3.8. Pineado del Si3000 [5].

3.5.3. LAYOUT.

El objetivo del diseño esquemático, es lograr hacer un componente físico para la aplicación y mediante el bus SPI y el códec Si3000 poder pasar muestras de audio entre el dsPIC30F4011 y el códec.

El primer paso fue hacer un esquemático mediante la aplicación PCB123 como se explicó anteriormente. Después mediante la misma herramienta, se genera un layout o como en este caso, generar dos, para la cara superior e inferior de nuestro diseño.

Una vez hecho el diseño del layout, esté es el resultado obtenido para ambas caras del diseño de nuestra futura PCB, por una parte se muestra la cara superior en la figura 3.9 y a continuación, en la figura 3.10, se muestra el resultado de la cara inferior.

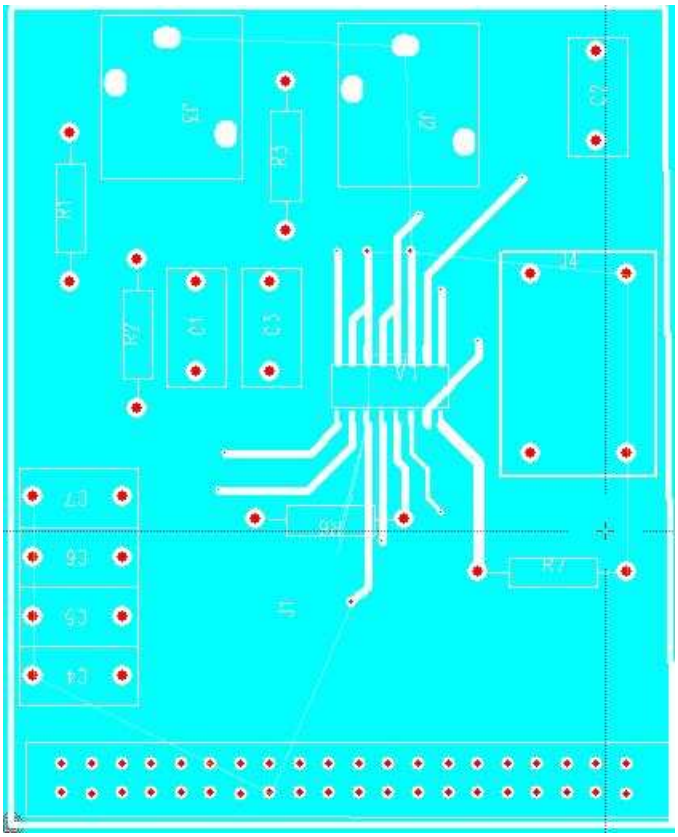

Figura 3.9. Diseño de la cara superior del layout.

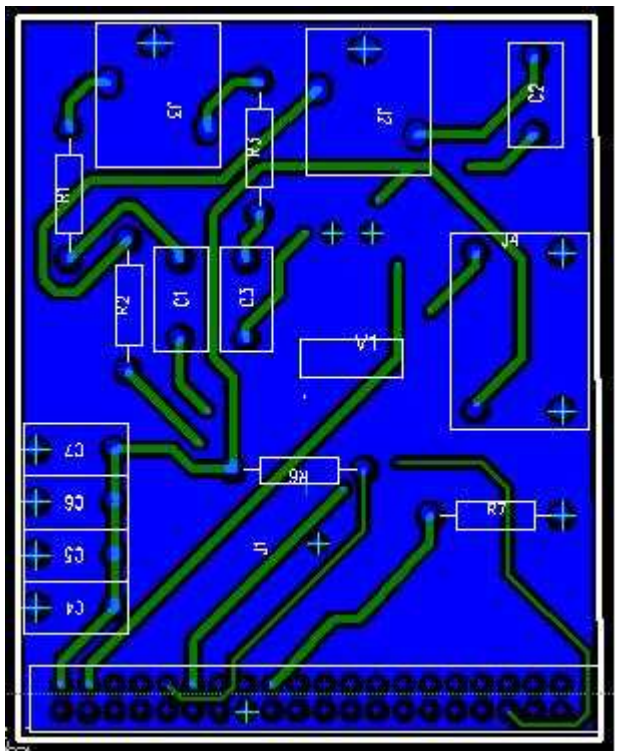

Figura 3.10. Diseño de la cara inferior del layout.

3.6. CONCLUSIONES.

Aunque si se hubiese poseído una placa de desarrollo que incorporara un códec de audio, el trabajo se hubiese reducido, el hecho de añadir esta PCB ha sido muy interesante y gratificante.

De este modo se han cubierto todas las partes del diseño de una aplicación desde la creación de un prototipo, hasta su configuración y posterior programación para la aplicación en las prácticas, como será el hecho de configurar en tiempo real los parámetros configurables del Si3000.

CAPÍTULO 4: PRÁCTICAS.

4.1. INTRODUCCIÓN.

Una vez explicado en el capítulo segundo y tercero la placa de desarrollo, el microcontrolador que posee y cómo se añadió el códec a la placa de desarrollo mediante una PCB, se pasa a detallar la parte software del PFC.

El objetivo de las prácticas propuestas es aprender a usar un dsPIC30F4011 como procesador digital de señal, usando las características DSP que éste posee. El capítulo cuarto consta del desarrollo de cuatro prácticas o aplicaciones, basadas en el sistema creado para conseguir el objetivo del PFC, que es desarrollar un curso de tratamiento de audio mediante dsPIC.

Además del desarrollo de las prácticas, se ha implementado un menú interactivo con los botones de la placa de desarrollo UIB-PC104 para controlar los parámetros del Si3000 en tiempo de ejecución de cualquier aplicación.

4.2. PRELIMINARES.

En este capítulo, se pretende describir una serie de características software, desarrolladas para un mejor manejo de las prácticas. La primera de ellas es la configuración de los parámetros del códec Si3000 en tiempo real mediante los botones que proporciona la placa de desarrollo UIB-PC104, y la segunda característica añadida es la multiplicación del tipo de datos fractional usada en las instrucciones típicas del motor DSP, en este caso la instrucción MPY.

4.2.1. MENÚ PARA EL CONTROL DE PARÁMETROS DEL Si3000.

El objetivo básico de esta función es cambiar la configuración del códec en tiempo de ejecución para cualquiera de las prácticas desarrolladas. Gracias a esta función, se puede comprobar cómo afecta el cambio de cada parámetro a la salida de cualquiera de las prácticas.

Por ejemplo, en un simple delay, el cambio de la frecuencia de muestreo influye en el tiempo de retraso del delay. Aunque no en todas las prácticas, los cambios de parámetros tendrán sentido. Por ejemplo, cambiar la ganancia de entrada en la práctica 2 de generación de ondas, no tiene ningún efecto en la onda generada, al no usarse los datos de entrada del Si3000.

La organización del menú consta de dos partes diferenciadas. La parte superior del LCD o menú principal, donde se puede navegar por los distintos parámetros a configurar. La parte inferior o menú secundario, permite a su vez navegar por las distintas opciones de configuración de cada parámetro. El menú creado consta de una parte superior con ocho parámetros configurables y ocho distintos menús inferiores que serán de tamaño igual a las opciones configurables de cada parámetro.

El uso del menú se hace desde el teclado de UIB-PC104 mediante cuatro de sus botones. Los botones 0 y 1 se usan para el desplazamiento dentro del menú en el cual se navegue en ese instante, ya sea el superior o el secundario. Los botones 3 y 4 se usan para confirmar el cambio y para volver al menú anterior respectivamente.

4.2.1.1. Idea general.

Como se vio en el capítulo tercero, el códec Si3000 es configurado mediante la escritura de datos en el bus SPI, en concreto, en la segunda trama enviada entre el dsPIC30F4011 y el Si3000. Estos datos se escriben en los registros que posee el Si3000 para llevar a cabo dicha configuración.

En el inicio del programa, la rutina de interrupción del SPI es la encargada de escribir esos datos en el Si3000. En este instante se esta llevando a cabo la primera configuración base del Si3000 mediante unos valores iniciales de configuración inicial.

Haciendo uso de los botones y el LCD se crea un método sencillo para cambiar los registros del Si3000. Cada vez que se cambia uno de esos registros, se vuelve a configurar todo el Si3000. Esto se consigue mediante un semáforo software que se enciende al inicio de la ejecución del programa, o cuando un parámetro de configuración del Si3000 es modificado. Una vez acabada la configuración, el semáforo no permite configurar el Si3000 hasta que otro parámetro sea elegido en el menú para modificarse.

Para diseñar toda la interfaz gráfica de configuración de los parámetros se creo una máquina de estados, en la cual, cada estado representaría un punto concreto de la interfaz, representativo con un número.

Para el control de la máquina de estados, se ha de tener en cuenta que cada uno dispone de cuatro eventos de cambio. También hay que controlar las pantallas a mostrar en el LCD y la ejecución que conlleva cada par estado-evento.

Por tanto, se necesitan unas funciones para controlar el texto del LCD, para el cambio de las variables de configuración, y para el control de los cambios de estado.

Además de las funciones se necesitan una serie de constantes para cada posible configuración del Si3000, es decir, por cada registro configurable se crearon constantes con las distintas opciones posibles. Estas constantes serán cargadas en el momento oportuno en unas variables que son las que se escriben en el Si3000 en cada configuración.

4.2.1.2. Parámetros controlables. Posibles configuraciones.

El Si3000 consta de 9 registros de configuración, con los cuales se configuran varias características interesantes para el procesado digital. Estas son las siguientes:

- Activación/Desactivación del filtro paso-alto del Si3000.
- PLL interno del Si3000. Configurable en /5 ó /10.
- Frecuencia de muestreo, esta es en referencia a PLL = /10. Para PLL = /5 esta frecuencia se duplica.
- Ganancia de la interfaz digital tras la recepción de muestras. RXG.
- Ganancia de la interfaz digital para la transmisión de muestras. TXG.
- Ganancia de la señal de entrada de audio. MIC
- Atenuación de la señal de salida de audio. SPEAKER.
- Configuración del filtro IIR o FIR, filtro interno del códec Si3000.

En los siguientes apartados, se muestran los valores posibles para configurar el Si3000 del modo elegido en el menú gráfico, teniendo en cuenta que algunos registros configuran varios parámetros a la vez. El registro 8 no se explica, por ser un registro de solo lectura de algunos eventos, y no ser usado en la configuración.

4.2.1.2.1. Registro 1: Activación del Si3000.

Este registro configura y activa al Si3000 y sus distintos bloques. En la tabla 4.1 se muestran los bits importantes de este registro:

Bit	Nombre	Función
7	**SR**	**Reset software.** 1 = Reset de los registros del Si3000. 0 = Funcionamiento normal del Si3000.
4	**SPD**	**Activación de la línea del altavoz.** 1 = Operación normal. 0 = Apagado de las líneas de altavoz.
3	**LPD**	**Activación de las líneas analógicas.** 1 = Operación normal. 0 = Apagado de las líneas analógicas de E/S.
2	**HPD**	**Activación del telefonillo.** 1 = Operación normal. 0 = Apagado de las líneas del telefonillo
1	**MPD**	**Activación de la línea de micrófono (MIC).** 1 = Apagado. 0 = Operación normal.
0	**CPD**	**Activación del Chip Si3000**. 1 = Apagado. 0 = Encendido.

Tabla 4.1. Bits importantes del registro 1 del Si3000 [5].

El registro 1 mediante el valor SI3000_1_INI (0x0110), activa el Si3000 y los bloques de micrófono y altavoz. Además, desactiva los bloques no usados, como las líneas analógicas de entrada y salida, el telefonillo. No se debe olvidar, que este valor representa por un lado la dirección del registro (primeros 8 bits) y por otro el valor a cargar en el registro seleccionado por la dirección (últimos 8 bits), como se comentó en el capítulo tercero:

4.2.1.2.2. Registro 2: PLL y Filtro paso-alto.

En el registro 2 se configuran dos parámetros al mismo tiempo, el PLL y la activación o desactivación del filtro paso-alto. En la tabla 4.2 se muestran los bits que intervienen de dicho registro.

Bit	Nombre	Función
4	**HPFD**	**Habilitación del filtro paso alto.** 1 = Desactivado ; 0 = Activado.
3	**PLL**	**Etapa final del PLL.** 1 = Frecuencia normal. (PLL dividido por 10). 0 = Frecuencia por 2. (PLL dividido por 5).

Tabla 4.2. Bits importantes del registro 2 del Si3000 [5].

Estas son las 4 opciones para la configuración con estos dos parámetros.

- FPA_ON__PLL_5 0x0200
- FPA_ON__PLL_10 0x0208
- FPA_OFF__PLL_5 0x0210
- FPA_OFF__PLL_10 0x0218

4.2.1.2.3. Registro 3 y 4: Frecuencia de muestreo.

Los registros 3 y 4 determinan la frecuencia de muestreo, aunque también se implica el valor del PLL para determinar la frecuencia de muestreo. Se decidió que uno de estos parámetros se quedase fijo, y solo cambiar el otro para establecer la frecuencia. Éstas son las distintas opciones elegidas en KHz: 6.25, 5.65, 5.3, 4.9, 4.5, 4.25.

El registro 3 corresponde al divisor del bloque PLL, y el registro 4 corresponde al multiplicador del bloque PLL. Estos son los valores de las constantes para conseguir las frecuencias anteriormente mencionadas:

- FREC_A 0x030A
- FREC_B 0x030B
- FREC_C 0x030C
- FREC_D 0x030D
- FREC_E 0x030E
- FREC_F 0x030F
- FREC_BASE 0x048A

Recordar que estas frecuencias son multiplicadas por dos seleccionando la opción de PLL /5 [5].

4.2.1.2.4. Registro 5: MIC y Tipo de filtro paso-alto.

El registro 5 configura dos parámetros a la vez, estos son la ganancia de la entrada de la señal de audio (MIC), y el uso de un filtro FIR o IIR en la transmisión y recepción de las muestras de audio analógicas. En la tabla 4.3 se muestra el significado de los bits más importantes:

Bit	Nombre	Función
4:3	**MCG**	**Ganancia del micrófono (MIC).** 11 = 30dB de ganancia. 10 = 20 dB de ganancia. 01 = 10 db de ganancia. 00 = 0 db de ganancia.
0	**IIR**	**Permiso para IIR o FIR.** 1 = Usa IIR. 0 = Usa FIR.

Tabla 4.3. Bits importantes del registro 5 del Si3000 [5].

Éstas son las distintas constantes para configurar esos dos parámetros mediante el menú de Si3000. Como se puede observar, el parámetro MIC posee cuatro posibilidades por solo dos del parámetro IIR.

- FIR__MIC_0dB 0x0562
- FIR__MIC_10dB 0x056a
- FIR__MIC_20dB 0x0572
- FIR__MIC_30dB 0x057a
- IIR__MIC_0dB 0x0563
- IIR__MIC_10dB 0x056b
- IIR__MIC_20dB 0x0573
- IIR__MIC_30dB 0x057b

4.2.1.2.6. Registro 6 y 7: RXG, TXG.

Los registros 6 y 7 configuran la ganancia anterior a la transmisión digital, a través del bus SPI (RXG) y la ganancia posterior a la recepción digital del bus SPI (TXG). Ambas ganancias estas expresadas en dB.

Dichas ganancias pueden ser configuradas en 32 posibilidades distintas entre -34.5 dB (RXG ó TXG = 00000) y 12 dB (RXG ó TXG = 11111) con paso entre ellas de 1.5 dB.

RXG [6:2] corresponden a los 5 bits del registro 6 para configurar la ganancia RXG. TXG [6:2] corresponden a los 5 bits del registro 7 para configurar la ganancia TXG. Los valores seleccionados para el menú son seis de las treinta y dos posibles. Son las siguientes: 12dB, 6dB, 0dB, -10.5dB, -22.5dB, -34.5dB.

Estos son los valores incluidos como constantes para representar dichas opciones de ganancia, tanto para el parámetro TXG como el parámetro RXG:

- RXG_12dB 0x067c
- RXG_6dB 0x066c
- RXG_0dB 0x065c
- RXG_105dB 0x0640
- RXG_225dB 0x0620
- RXG_345dB 0x0600
- TXG_12dB 0x077f
- TXG_6dB 0x076f
- TXG_0dB 0x075f
- TXG_105dB 0x0743
- TXG_225dB 0x0723
- TXG_345dB 0x0703

4.2.1.2.6. Registro 9: Atenuación de los altavoces.

El registro 9 se usa para configurar la atenuación de la señal de salida de audio. Sus bits 1 y 0 son los responsables de la configuración de este parámetro, siendo la atenuación de 18dB con el valor ‘11’; 12 dB con el ‘10’; 6dB con el ‘01’; y 0 dB con el ’00’.

A continuación, se muestran las constantes para configurar este parámetro:

- SPEAK_18dB 0x0903
- SPEAK_12dB 0x0902
- SPEAK_6dB 0x0901
- SPEAK_0dB 0x0900

4.2.1.2.7. Configuración inicial. Reproducción audio.

La configuración inicial de los parámetros comentados anteriormente es la siguiente:

- SI3000_1_INI → Activación de los módulos usados del Si3000.
- FPA_ON__PLL_5→Filtro paso-alto ON; Se duplica la frecuencia de FREC_A.
- FREC_A → 12.5KHz. Es la frecuencia máxima de trabajo.
- FREC_BASE →No influye. La frecuencia es combinación con FREC_A.
- FIR__MIC_10dB → MIC = 10dB; Filtro FIR para la entrada de señal de audio.
- RXG_105dB → Ganancia RX en -10,5dB.
- TXG_105dB → Ganancia TX en -10,5dB.
- SPEAK_6dB → Atenuación de la señal de salida de audio de -6dB.

4.2.1.3. Funciones y variables.

Las funciones creadas para conseguir el de configuración menú se agrupan en cinco grupos:

- Funciones para el control de interrupciones SPI y Timer1.
- Funciones para moverse dentro de los menús superior e inferior.
- Funciones para mostrar en el LCD el menú correspondiente.
- Funciones para el cambio de parámetros deseado.
- Función para configurar el Si3000.

Además de estas funciones, hay una serie de variables que actúan de control del flujo del programa. Son las variables *control* y *permiso*.

La variable *control* determina si se configura el códec. Cuando *control* es 0 se configura, cuando *control* es 1 no se configura y solo se mandan muestras de audio.

- control = 0 → CONFIG_CODEC_ON.
- control = 1 → CONFIG_CODEC_OFF.

La variable *permiso* esta relacionada con el temporizador T1. Controla que las pulsaciones de los botones sean eficientes. Inicialmente *permiso* es 1, si se pulsa cualquier botón se pone a 0 y se lanza el temporizador, deshabilitando así la posibilidad de pulsar

durante un breve espacio de tiempo, tras el cual la variable *permiso* se coloca en 1 para poder usar los botones de nuevo.

La finalidad de esta variable es evitar que, mediante una sola pulsación, el microcontrolador lea varias pulsaciones debido al bucle infinito del programa principal, eso haría incorrecto el funcionamiento del menú.

- permiso = 0 ---> BOTON_OFF.
- permiso = 1 ---> BOTON_ON.

4.2.1.4. Diagrama de bloques del menú.

Se detallará la parte correspondiente al menú Si3000 mediante un diagrama de bloques, que dará una idea del movimiento de los estados, posteriormente en la figura 4.1 se ve el diagrama de dicha máquina de estados.

Se parte de un estado inicial (estado = 1000), desde él se puede acceder al primer parámetro configurable (estado = 10) mediante el botón 3. En ese momento, a través de los botones 0 y 1 se puede desplazar por los distintos parámetros a configurar. Esto correspondería al menú superior.

En el menú superior, mediante una función, se escribirá en la primera línea del LCD el parámetro seleccionable para configurar.

Éstas son las equivalencias de estados con los parámetros configurables:

- Estado = 10→ Selección de la activación del filtro paso alto digital.
- Estado = 20→ Selección del PLL.
- Estado = 30→ Selección de la frecuencia.
- Estado = 40→ Selección de la ganancia de entrada analógica.
- Estado = 50→ Selección de la atenuación de salida analógica.
- Estado = 60→ Configuración de la ganancia digital RX.
- Estado = 70→ Configuración de la ganancia digital TX.
- Estado = 80→ Selección de filtro FIR o IIR.

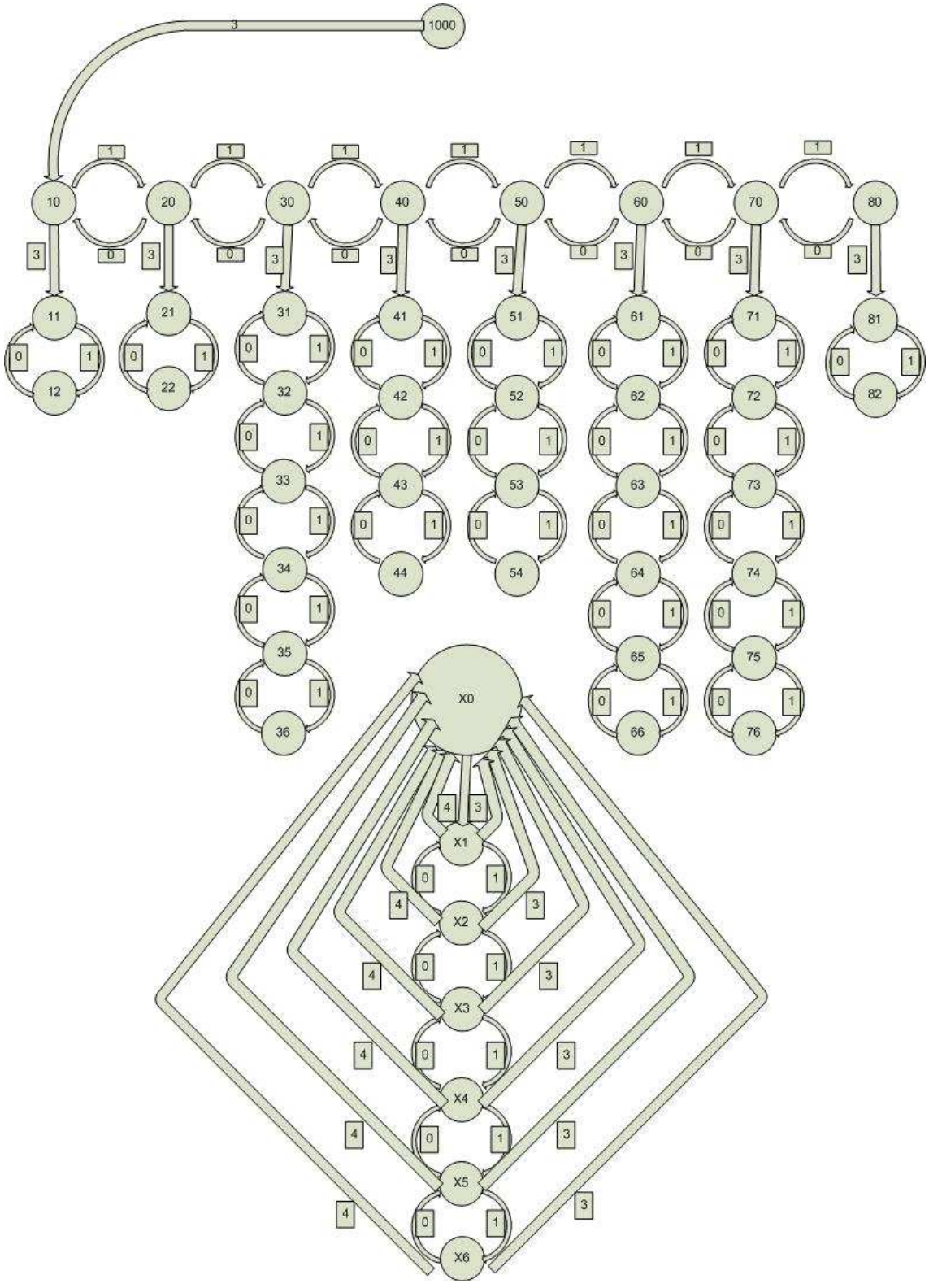

Figura 4.1. Máquina de estados del menú Si3000.

Para acceder al menú inferior, se ha de pulsar el botón 3. A través de los botones 0 y 1 se podrá navegar entre las distintas posibilidades a configurar dentro de cada parámetro.

Las ramas de cada menú inferior son equivalentes, con la salvedad, de que se configura el parámetro elegido en el menú superior, y que cada submenú tiene un tamaño distinto, debido a que cada parámetro posee distinto número de opciones, de dos a seis.

Para volver al menú superior hay dos opciones, la de seleccionar un cambio (botón 3) y la de volver (botón 4). Si se pulsa el botón 4 se regresa al menú superior en el parámetro actual, si se pulsa el botón 3, además, se ejecuta el cambio seleccionado

Cada estado de los menús inferiores tiene un significado de parámetro a cambiar en una opción determinada. Por ejemplo, en el estado número 51, el parámetro a cambiar es la atenuación de la salida analógica al valor -18dB.

En cada submenú inferior se usa una función distinta para controlar el LCD. En caso de seleccionar el cambio, otra función modifica la variable con la constante adecuada en cada estado.

4.2.2. MULTIPLICACIÓN FRACTIONAL.

En los dsPIC se usan distintos tipos de datos; enteros, flotantes, con o sin signo, con más o menos precisión, por tanto, existen una gran variedad de formatos con los que trabajar.

Por la arquitectura de los dsPIC, usar el Motor DSP es el modo más eficiente de hacer algunas operaciones, como las multiplicaciones.

El Motor DSP solo acepta datos enteros o un tipo especial de flotante, el formato fractional. Por lo que multiplicaciones en otro formato usan librerías adicionales, como ocurre en el formato flotante. Por lo que queda descartado el uso de flotantes, porque son necesarias conversiones entre formatos, lo cual consume un tiempo elevado, muy importante para futuras prácticas, como los filtros digitales.

Basándose en los tipos de datos que maneja el códec (fractional) y que los coeficientes de los filtros digitales son en su mayoría del tipo fractional, dicho formato es el usado en para el procesado digital, anteponiéndose al tipo entero.

El tipo de datos fractional se define como un tipo especial de flotante, pero cuya parte entera es siempre 0. Los datos del formato fractional son del tipo “0.abcdef”.

4.2.2.1. El problema de la multiplicación fractional.

El dsPIC30F4011 consta de dos tipos de instrucciones básicas; las instrucciones típicas de la ALU y las instrucciones del motor DSP. Fijándose solo en la multiplicación, el dsPIC30F4011 posee una multiplicación típica de la ALU y otra distinta del Motor DSP. La gran diferencia de estas dos multiplicaciones es el hecho de que solo la del Motor DSP puede hacerlas en tipo fractional. Por este motivo, es necesario usar esta multiplicación en la solución de las aplicaciones.

Al usar el lenguaje C, las multiplicaciones no se consiguen hacer en formato fractional ya que el lenguaje ensamblador generado usa siempre la multiplicación de la ALU, por lo que siempre multiplica enteros y no datos fractional como se requiere.

4.2.2.2. La solución a la multiplicación fractional.

La solución adoptada para solucionar el problema de la multiplicación fractional, es crear una rutina directamente en ensamblador que pueda ser usada cuando sea necesaria. Esto se puede hacer porque el lenguaje ensamblador diferencia entre la multiplicación de la ALU (instrucción MUL), y multiplicaciones del motor DSP (instrucción MPY).

La función MPY toma los dos parámetros de entrada (se cargan automáticamente en los registros W0 y W1) y los carga en los registros W4, W5 propios de la instrucción MPY. El resultado lo pasa del acumulador al registro W0 para devolver el resultado en la variable correspondiente. Esta seria la función implementada:

```
                    .global _multfractional

_multfractional: PUSH CORCON
                    MOV #0x0010,W3
                    MOV W3,CORCON
                    MOV W0,W4
                    MOV W1,W5
                    MPY W4*W5,A
                    SAC A,W0
                    POP CORCON
                    return
                    .end
```

Esto es un ejemplo de una llamada a esta función en el lenguaje C:

```
Salida = multfractional(Multiplicador1, Multiplicador2);
```

4.2.3. LA LIBRERÍA UIBLIB.

La librería UIBLIB es proporcionada por los diseñadores de la placa de desarrollo usada en este proyecto para el mejor manejo de los recursos que dispone. Mediante esta librería, se han controlado tres recursos básicos de la placa de desarrollo. Se trata de las pantallas LCD, los botones, y los leds. Se comentarán las funciones y variables necesarias para el control de esos tres recursos.

El control del LCD se hace mediante las siguientes funciones:

- LCDWriteString((char*)texto);
- LCDGotoFirstLine();
- LCDGotoSecondLine();

Con estas funciones, se puede elegir la línea del LCD donde escribir y el texto a poner.

Para el control de los pulsadores se hace una lectura en unas variables. Existen 12 variables debido a la existencia de 12 pulsadores en la placa de desarrollo. El formato de las variables es "Buttons.SWx", donde x es un valor entre 0 y 11. Estas variables son útiles en el diseño de los distintos menús que dispone el PFC.

Para el control de los leds se hace una escritura en unas variables. Existen por tanto 6 variables para el control de los 6 leds de la placa de desarrollo. El formato de las variables es "LEDx", donde x es un valor entre 0 y 5. Estas variables consiguen encender el led correspondiente si se carga un 1, y apaga el led cargando un 0.

4.2.4. LA LIBRERÍA SPI.

Microchip provee de una librería para el manejo de todos los bloques de sus DSPs, entre ellas, dispone de una librería para el control y uso del interfaz digital de comunicaciones SPI. Mediante esta librería, se usa el bloque SPI, para unir el dsPIC30F4011 y el Si3000. De esta librería, se usan las funciones de recepción y transmisión del interfaz, porque la configuración de la interrupción SPI y activación del bloque, se hace directamente escribiendo los bits correspondientes.

Las funciones son las siguientes: WriteSPI1() y ReadSPI1() y se usan en la rutina de interrupción del interfaz SPI.

4.3. PRÁCTICA 1: EFECTOS DE AUDIO.

La práctica consta de cómo se logra reproducir el audio a través del dsPIC, y tres sencillas aplicaciones de dificultad ascendente.

La organización de cada apartado consta de una breve descripción del efecto a diseñar, una pequeña descripción de cómo se consiguió la solución, y un apartado de restricciones que tiene diseñar estas aplicaciones en el dsPIC30F4011.

4.3.1. MUESTREO Y EFECTOS DE AUDIO.

La idea de incluir este apartado es hacer una introducción de cómo esta organizado el proyecto creado en MPLAB, y dónde hay que colocar la función de procesado de audio para llevar a cabo cualquiera de las aplicaciones.

4.3.1.1. Funcionamiento básico de la reproducción de audio.

El muestreo del audio o la ejecución de cualquier procesado sobre las muestras en este PFC se hace básicamente en la rutina de tratamiento de interrupción del interfaz SPI. Esto se debe a que las muestras llegan a través de este interfaz al dsPIC30F4011 desde el Si3000. Una vez ejecutado el procesado digital son enviadas por este interfaz de nuevo hacia el Si3000.

Cuando se produce una interrupción hay dos fases, una primera para configurar el códec Si3000 y otra donde solo se pasan muestras de audio en ambas direcciones entre el Si3000 y el dsPIC30F4011. En el caso que nos ocupa no hay función de procesado de señal, solo se limita a coger la muestra de entrada, y escribirla en el buffer para una correcta transmisión de datos en ambas direcciones.

Se podría decir que la reproducción de audio consiste en la continua escritura en el buffer del bus SPI del dato recibido anteriormente.

4.3.1.2. El fichero “procesado.c”.

Lo interesante no es reproducir audio, esto solo implica leer un buffer y volver a escribirlo, lo importante radica en hacer algo con esas muestras de entrada, hacer efectos como el delay, eco o reverberación.

Se necesitan unas funciones y variables para llevar a cabo la ejecución de todo procesado, éstas estarán en un fichero de C, el fichero "procesado.c", en el que se guardará todo lo relacionado con el procesado de cada una de las prácticas. Posteriormente se tendrá que hacer una llamada a la función de procesado, que estará colocada después de la lectura del buffer SPI y antes de su escritura.

El tiempo entre la llegada de dos muestras es el tiempo del que se dispondrá para ejecutar todo el procesado. Este tiempo es proporcional a la frecuencia, siendo para una frecuencia más alta, menor el tiempo disponible.

4.3.2. DELAY.

Para empezar las prácticas de procesado de audio se propone hacer un retraso de la señal de entrada. Primero se verá un estudio teórico simple del efecto y posteriormente se describe una posible solución, por último se comenta la práctica y sus posibles restricciones debido al uso del dsPIC30F4011.

4.3.2.1. Definición.

Un retraso es un efecto que consiste en la multiplicación y retraso modulado de una señal sonora, es decir, la señal de salida tendrá las mismas características que la original de entrada pero tendrá un simple retraso, y si la multiplicación es distinta de 1, poseerá una atenuación o ganancia, siendo este el único procesado de las muestras .

Se muestra en la figura 4.2 cómo se lleva a cabo el delay.

Figura 4.2. Esquema del delay.

Un retraso dispone de dos parámetros configurables, el tiempo de retraso y la posible ganancia aplicada, aunque esta suele ser 1, siendo el tiempo de retraso el parámetro característico de un delay.

4.3.2.2. Descripción de la solución.

Para hacer un delay solo se necesita conocer una característica del dsPIC30F4011, el espacio en memoria del que se dispone para guardar muestras y escribirlas en el buffer lo

más tarde posible, puesto que no se requiere ningún procesado. El siguiente código se muestra el procesado realizado:

```
fractional Delay(fractional Entrada)
{
	fractional Salida;
	Salida = XBUFF[i];
	XBUFF[i] = Entrada;
	i++;
	if  (i == 500 ){i = 0;};

	return Salida;
}
```

En nuestro caso se tienen 2KBytes de memoria de datos, lo que supone que se puede almacenar 1000 muestras de audio en memoria de datos, pero no se puede coger toda la memoria, porque la sobrante es la usada para la pila del dsPIC30F4011 y esto haría al sistema inestable.

Para este caso, se decidió usar la mitad de memoria del dsPIC30F4011 para almacenar muestras, aunque se podría haber usado más cantidad de memoria, pero al tener este tamaño se consigue determinar el retraso de la señal más fácilmente.

El parámetro que en este caso influye en el retraso, es la frecuencia de muestreo. Por ejemplo, con un buffer de 500 muestras y una frecuencia de muestreo de 1KHz se consigue medio segundo de retraso, con 2KHz se consiguen 0.25 segundos.

La única restricción de este efecto de audio es la poca cantidad de memoria para almacenar muestras, lo que hace que los retrasos generables con el dsPIC30F4011 sean retrasos muy cortos. Para hacerlos más altos se tendría que disminuir la calidad de la señal, bajando la frecuencia de muestreo.

4.3.3. ECO.

En esta segunda aplicación de audio ya se usa por primera vez la multiplicación fractional, esto conlleva el primer procesado digital.

4.3.3.1. Definición.

El eco es un fenómeno relacionado con la reflexión del sonido. La señal acústica se ha extinguido, pero aún devuelve sonido en forma de onda reflejada. Se produce cuando la onda sonara se refleja perpendicularmente en una pared, y para ello la superficie reflectante

debe estar separada del foco sonoro 17m para sonidos musicales y 11,34 para sonidos secos, debido a la persistencia acústica [9].

El eco es un efecto sonoro sencillo, consiste en reproducir la señal de audio del instante actual, junto a la señal de audio retrasada y atenuada.

La figura 4.3 muestra el diagrama de bloques del eco.

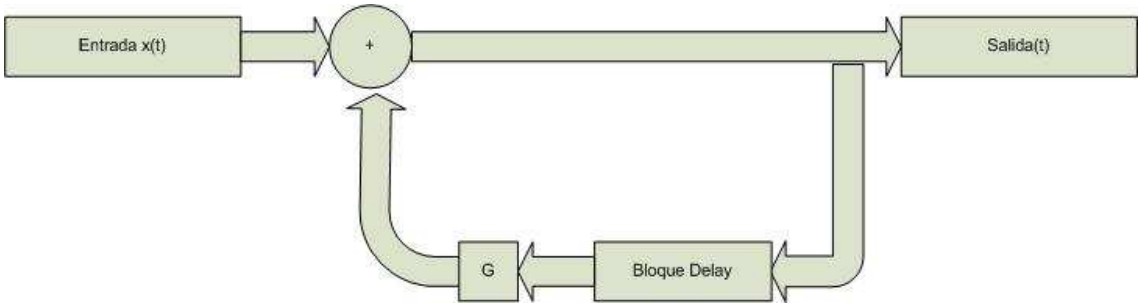

Figura 4.3. Esquema de Eco.

Como se puede apreciar, un eco dispone de dos parámetros configurables, son el tiempo de retraso y la ganancia de eco.

4.3.3.2. Descripción de la solución.

Como sucedió en la aplicación de delay, en el eco se debe tener en cuenta el tamaño de la memoria de datos, del mismo modo se optó por tener 500 muestras en el buffer, con lo que a una frecuencia de 5KHz se consigue un eco cuyo tiempo de retraso es de 0.1s.

Para lograr la solución deseada se deben guardar en memoria las salidas anteriores para ser sumada a la entrada. Dicha salida se obtuvo hace 500 iteraciones y está multiplicada por una ganancia, así se logra que el eco sea acumulativo. A continuación se muestra el diseño en C, de la aplicación:

```
fractional EcoFract(fractional Entrada)
{
	fractional Salida, EcoTotal, Eco;

	Salida = XBUFF[i];
	Eco = multfractional(Salida ,0x4000 );
	EcoTotal = Eco + Entrada;
	XBUFF[i]=EcoTotal;
	i++;
	if(i==500){i=0;}

	return EcoTotal;
}
```

4.3.3.3. Restricciones.

Al igual que en el delay, para obtener un retraso considerable, habría que bajar en exceso la frecuencia de muestreo, por lo que la calidad del audio seria baja. Habría que llegar a un compromiso para apreciar bien el eco, con una calidad de audio decente, siendo en este caso una buena opción es hacer el muestreo a 4KHz, con lo que se logra un tiempo de retraso de 125ms.

La frecuencia de muestreo es un tercio de la máxima para conseguir un eco muy corto, esto hace ver las limitaciones del dsPIC30F4011 para generarlo.

4.3.4. REVERBERACIÓN.

Esta aplicación, es una mezcla de las dos anteriores para unirlas en una más interesante. Además, se añadirá un menú para configurar los parámetros propios de la reverberación en tiempo de ejecución.

El menú para cambiar parámetros no se hace en el delay o eco, porque modificando la frecuencia se modificaba el parámetro fundamental y único, el tiempo de retraso.

4.3.4.1. Reverberación. Teoría.

La reverberación es un fenómeno derivado de la reflexión del sonido consistente en una ligera prolongación del sonido una vez que se ha extinguido el original, debido a las ondas reflejadas. Estas ondas reflejadas sufrirán un retardo no superior a 100 milisegundos, que es el valor de la persistencia acústica, tiempo que corresponde a una distancia de 34 metros recorridos (17 de ida y 17 de vuelta) a la velocidad del sonido [9].

En un recinto pequeño la reverberación puede resultar inapreciable, pero al aumentar las dimensiones del recinto, la percepción del oído de ese retardo es mejor. Se usan para determinar la reverberación de un recinto una serie de parámetros físicos, como el tiempo de reverberación. [9].

Como se aprecia en la definición teórica, la reverberación es una mezcla de eco y delay, esto también se puede apreciar en la figura 4.4, que se usó de base para hacer nuestra aplicación, aunque cabe resaltar que hay multitud de implementaciones de este efecto, e igual de eficientes.

En la implementación de la reverberación, la salida de audio es la suma, de la entrada directa, de una etapa de retraso, y de una etapa de eco acumulativo o etapa de reverberación.

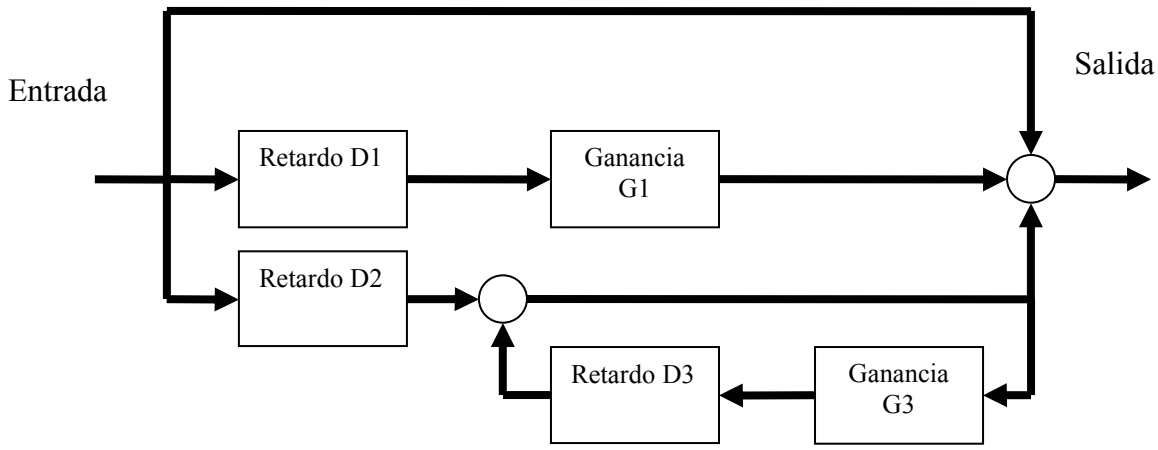

Figura 4.4. Esquema de la reverberación [14].

- D1=ITD, D2=2*D1, D3=50ms, G1=G3, G3= $10^{e}((-3*D3)/tr)$.

Como se aprecia, hay dos parámetros claves, el retardo ITD, y el tiempo de reverberación.

4.3.4.2. Descripción de la solución.

En el diagrama existen tres células de retardo, con lo cual, se intuye que se necesitan tres buffer de memoria donde guardar las muestras de entrada en cada punto donde aparecen células de retraso.

Como se comentó, la salida es una suma de tres señales distintas con lo que hay que tener cuidado con una posible saturación de la señal de salida.

Para lograr el diseño de la reverberación con los dos parámetros que posee, se crean unas constantes para cargarlas en dos variables de configuración, una por cada parámetro.

El primer parámetro es la atenuación configurable dependiente del tiempo de reverberación (tr), el cual influye directamente en las atenuaciones del diagrama de bloques de la solución implementada. El segundo es el tiempo de eco inicial, que controla el tamaño del buffer en tiempo de ejecución, aunque realmente, el buffer es estático y lo que se controla es el tamaño del buffer que se usa.

Este el código de la resolución de la reverberación:

```
//XBUFF,YBUFF,ZBUFF son las distintas celulas de retardo del diagrama de bloques.
//GReverb, GReverb2 , son las dos ganancias de la aplicación.
//RetardoReverb, RetardoReverb2 controla el numero de muestras a usar en el buffer.
fractional Reverb(fractional Entrada){

//TRATAMIENTO XBUF

        Delay = XBUFF[ix];
        DelayG = multfractional(Delay , GReverb);
        XBUFF[ix]=Entrada;
        ix++;
        if (ix>=RetardoReverb){ix=0;LED2=1;}

//TRATAMIENTO YBUFF

        Delay2= YBUFF[iy];
        YBUFF[iy] = Entrada;
        iy++;
        if (iy>=(RetardoReverb2)){iy=0;LED3=1;}

//TRATAMIENTO ZBUFF

        Reverbb = ZBUFF[iz];

        Delay2R= (Delay2 + Reverbb);
        Delay2RG = multfractional(Delay2R,GReverb);

        ZBUFF[iz]= Delay2RG;
        iz++;
        if (iz==200){iz=0;LED4=1;}

        //SALIDA

        Salida= DelayG + Entrada + Delay2R;

        return Salida;
}
```

4.3.4.3. Parámetros variables. Obtención.

Estos son los parámetros del diseño de la reverberación implementada:

- D1=ITD, D2=2*D1, D3=50ms, G1=G3, G3= $10^{e}((-3*D3)/tr)$.
- ITD corresponde con el tiempo de eco inicial.
- Tr corresponde con el tiempo de reverberación [14].

Se añade un menú para modificar los dos parámetros entre sus seis opciones. Este menú irá unido al menú inicial de la configuración del Si3000, pudiéndose cambiar de un menú a otro sin tener que parar la ejecución de la reverberación.

Con la combinación de los siguientes parámetros se puede simular la reverberación de lugares tan usuales como un estudio, un auditorio o una iglesia. La tabla 4.4 muestra ejemplos de estos parámetros en ambientes típicos donde se produce este efecto sonoro como un auditorio o una iglesia.

LUGAR	ITD (ms)	Tiempo de reverberación (s)
Estudio	20	0,4
Auditorio	40	0,75
Sala de concierto	25	1,8
Iglesia	50	3,25

Tabla 4.4. Parámetros para la reverberación de algunos ambientes [14].

Estos son los valores elegidos para las constantes que serán cargadas en las variables de los parámetros de tiempo de eco inicial (ITD) y del tiempo de reverberación. Ambos parámetros controlaran la ejecución de la reverberación.

- GA 0x35fa tr = 0.4 G3 = 0.4217
- GB 0x50c1 tr = 0.75 G3 = 0.6309
- GC 0x5ff9 tr = 1.2 G3 = 0.7498
- GD 0x69a7 tr = 1.8 G3 = 0.8254
- GE 0x6f7a tr = 2.5 G3 = 0.8709
- GF 0x7316 tr = 3.25 G3 = 0.8991

- RETARDOA 60 15ms RETARDOA2 120 30ms
- RETARDOB 80 20ms RETARDOB2 160 40ms
- RETARDOC 100 25ms RETARDOC2 200 50ms
- RETARDOD 120 30ms RETARDOD2 140 60ms
- RETARDOE 160 40ms RETARDOE2 320 80ms
- RETARDOF 200 50ms RETARDOF2 400 100m

Estos parámetros dependen de la frecuencia de muestreo y están elegidos para una frecuencia de muestreo de 4KHz. Por ello, antes de usar esta práctica habría que configurar en el menú del Si3000 el parámetro de la frecuencia de muestreo para el correcto funcionamiento respecto las exigencias temporales teóricas.

4.3.4.4. Interfaz con menú para el control de la reverberación.

El menú de reverberación tiene muchas similitudes con el de configuración del Si3000, y además se ha diseñado para poder pasar de uno a otro. Este segundo menú funciona también como una máquina de estados unida a la anterior. Cada parámetro tiene un menú superior y seis opciones en su parte inferior para poder elegir una de ellas.

La figura 4.5 muestra la máquina de estados correspondiente al menú de reverberación.

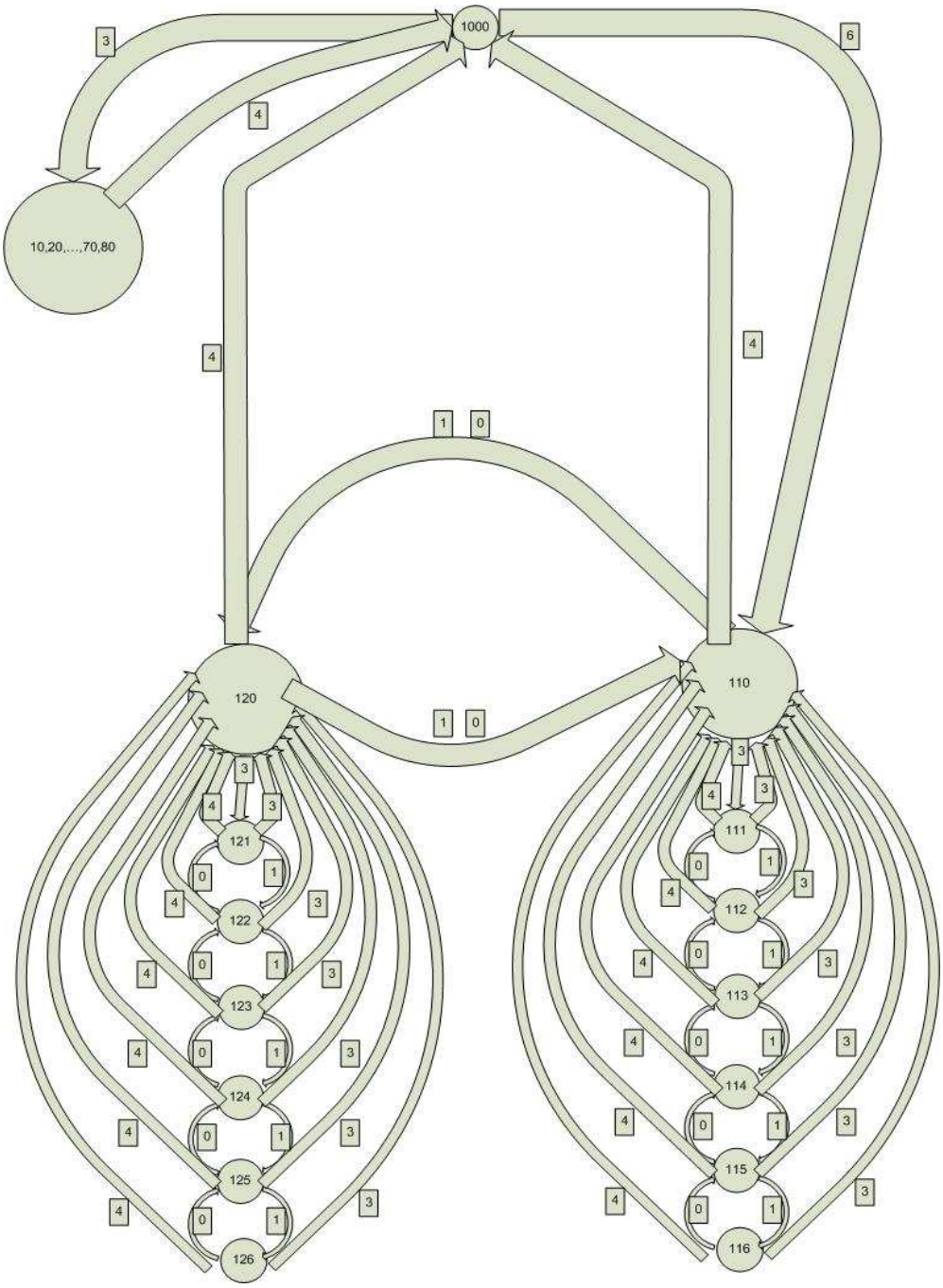

Figura 4.5. Máquina de estados para el Menú de reverberación.

Se añadió la opción de moverse por los dos menús: el de configuración del códec Si3000 y el de los parámetros de reverberación.

Para la implementación de este menú se ha seguido la misma filosofía del anterior, con lo que se crearon funciones para moverse por la máquina de estados, para controlar el LCD en cada estado, y para ejecutar los cambios modificando las variables oportunas.

La parte correspondiente al menú del Si3000 no se ha mostrado en el diagrama, pero se intuye dónde colgaría esa zona de la maquina de estados.

Los estados correspondientes a 120, 130, están relacionados con los dos parámetros ITD y tiempo de reverberación y los estados, 12A y 13B, (A y B entre 1 y 6) corresponden a los estados para modificar cada parámetro en sus valores anteriormente citados.

La funcionalidad de los botones es similar a la usada en el menú Si3000. Los botones 0 y 1 sirven para moverse por los menús tanto superior e interior, y los 3 y 4 para ejecutar un cambio o volver al menú anterior respectivamente.

4.3.4.5. Restricciones.

En las especificaciones se comprueba que el tiempo D3 = 50ms, y en la tabla 4.4, se ve que ITD es como máximo 50ms, pero como el parámetro D2=2*D1 = 2*ITD, esto hace ver que se necesitan dos buffer de un tamaño igual, y otro del doble de tamaño.

Como se comento, para el eco y el retraso, el tamaño de la memoria del dsPIC30F4011 usado en este PFC, solo permite guardar 1000 muestras de 16 bits, y no se puede ocupar toda la memoria porque la pila debe llevar a cabo sus funciones.

El porcentaje de memoria que se eligió para su uso en esta aplicación, es un 80%, por lo que se dispone de espacio para 800 muestras, distribuidas de modo que los buffer son de tamaño 200, 200 y 400.

Para conseguir los tiempos de las especificaciones hay que usar una frecuencia de muestreo de 4KHz, siendo esta frecuencia un tercio de la frecuencia máxima del Si3000, esto es debido a la escasez de memoria del dsPIC30F4011. Con esto se consiguen que los buffer sean de 50ms el buffer fijo y de 100ms y 50ms como máximo los dos buffer de tamaño variable.

4.4. PRÁCTICA 2: GENERACIÓN DE ONDAS.

En esta segunda práctica el objetivo es conseguir generar ondas periódicas típicas, como una onda cuadrada o una onda senoidal, entre otras. En esta práctica la parte de procesado digital se reduce a alguna multiplicación de amplitud.

Ahora no se usa la entrada de datos desde el SPI, puesto que la salida generada no tiene nada que ver con la entrada de datos.

4.4.1. LA ONDA CUADRADA.

4.4.1.1. Definición.

Una onda cuadrada consiste en una señal periódica que alterna entre dos extremos de amplitud A y -A, sin pasar por los valores intermedios.

En la figura 4.6, se aprecia la forma típica de esta onda.

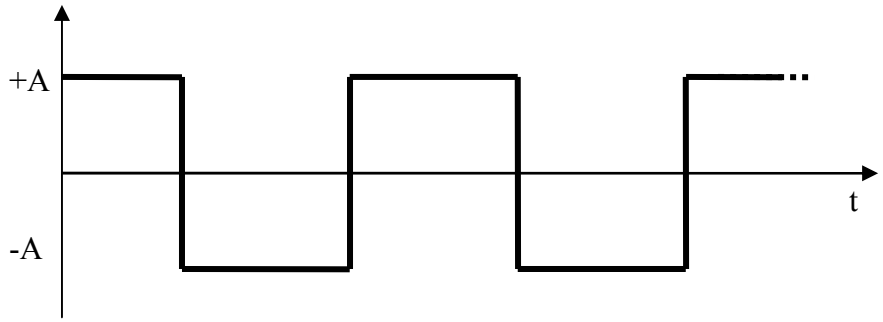

Figura 4.6. La onda cuadrada.

4.4.1.2. Descripción de la solución.

La solución consiste en escribir en el bus de SPI una constante y su opuesta de modo alternado, cuanto mayor número de veces se envié una constante, mejor quedará definida la señal generada.

```
fractional OndaCuadrada()
{
	fractional Salida;
	cont = cont + 1;
	if (cont<10) { Salida = 0x7fff; }
	else { Salida = 0x8000; }
	if (cont == 20) {cont=0; }
	return Salida;
}
```

En este apartado son interesantes las posibilidades de configuración del Si3000 para ver cómo afecta la salida y poder observarla en el osciloscopio. Pero no es interesante modificar parámetros como la ganancia de entrada analógica.

4.4.1.3. Restricciones.

La restricción más importante es la frecuencia máxima a poder generar, debido a que la frecuencia de muestreo máxima del Si3000 es 12,5KHz. Además, como no se puede hacer una onda cuadrada alternando una muestra positiva y otra negativa, las frecuencias posibles bajan considerablemente.

Para que sea visible y no haya problemas de estabilidad, al menos se deberían poner cinco muestras de cada nivel positivo o negativo, con lo que se generarían ondas cuadradas de 1,2 KHz.

4.4.2. LA ONDA TRIANGULAR.

4.4.2.1. Definición.

La onda triangular es un tipo de onda periódica que presenta unas velocidades de subida y bajada constantes. Normalmente dichas velocidades son iguales por lo que la onda es periódica y simétrica. La figura 4.7 muestra un ejemplo de una onda triangular.

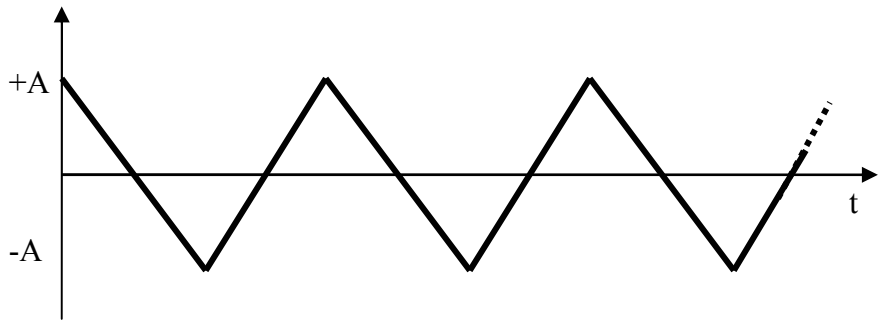

Figura 4.7. La onda triangular.

4.4.2.2. Descripción de la solución.

Para generar esta onda se necesita simular la velocidad de subida y bajada de esta señal. Para ello, se multiplica una constante de amplitud por una variable que ira pasando de ‘1’ a ‘x’ en el primer intervalo y de ‘x’ a ‘1’ en la segunda mitad de la señal, simulando así las

rampas de subida y bajada características de esta señal, tal y como se ve en el diseño en C de la onda triangular.

```
fractional OndaTriangular(){

        fractional SalidaS;
        int Aux;

        cont = cont + 1;
        if(cont<=10)
                SalidaS = cont * 0x0500; //Contador de 1 a 10
        else{
                Aux= 20-cont; //Aux cambia de 9 a 1
                SalidaS = Aux*0x0500;
        }
        if(cont==19){cont=0;}

        return SalidaS;
}
```

4.4.2.3. Restricciones.

Al igual que en la onda cuadrada, la única restricción es la frecuencia, pero en este caso se necesita un cierto número de muestras para que sean apreciables las subidas o bajadas. Por ejemplo 10 muestras para subida y bajada, lo que haría una señal periódica con una frecuencia 20 veces más baja a la frecuencia de muestreo del Si3000, obteniendo así una frecuencia de alrededor 600Hz.

4.4.3. LA ONDA DE SIERRA.

4.4.3.1. Definición.

Una onda de sierra es aquella que solo posee velocidad de subida o velocidad de bajada, y que, al llegar a su límite superior o inferior, vuelve al inicio para comenzar la subida o bajada. Se ven las dos posibilidades en las figuras 4.8, y 4.9:

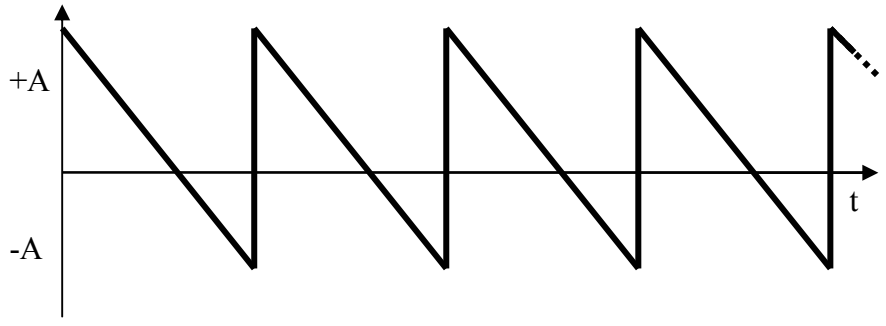

Figura 4.8. La onda de sierra. Velocidad de bajada.

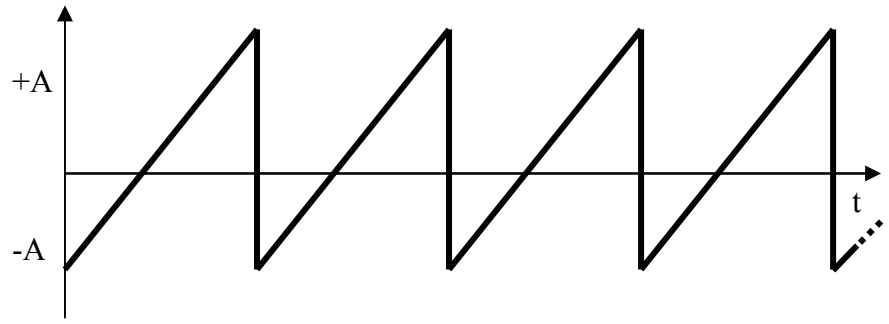

Figura 4.9. La onda de sierra. Velocidad de subida.

4.4.3.2. Descripción de la solución.

Una onda de sierra es una onda muy parecida a la triangular, pero solo posee un tiempo de subida o un tiempo de bajada, por lo que solo se necesitará simular uno de estos tiempos según el tipo de onda de sierra que se quiera conseguir. Se muestra el código resultante de una onda de sierra con velocidad de subida.

```
fractional OndaSierra()
{
	fractional SalidaS;
	int Aux;

	cont = cont + 1;
		SalidaS = cont * 0x0300; //Contador de 1 a 20
	if (cont == 20){cont = 0;}

	return SalidaS;
}
```

4.4.3.3. Restricciones.

Al igual que para la onda cuadrada y triangular, la única restricción, son las frecuencias representables, pero en este caso, al solo necesitar definir en el osciloscopio un tiempo de subida, solo serian necesarios unos 10 puntos por periodo. Con esto se logran fácilmente frecuencias de 1,2Khz.

4.4.4. LA ONDA SENOIDAL.

4.4.4.1. Definición. Teoría.

La onda senoidal es una señal analógica, por el hecho de poseer infinitos valores entres dos puntos cualesquiera del dominio. De hecho esta onda es la gráfica de la función matemática seno como se aprecia en la figura 4.10.

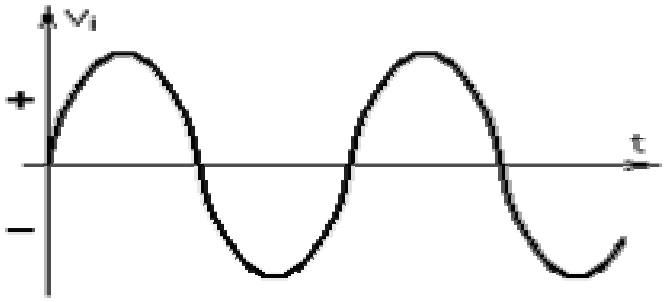

Figura 4.10. La señal senoidal.

Para conseguir representar esta señal se podrían guardar en el espacio de datos X los coeficientes en amplitud y reproducirlos periódicamente mediante el dsPIC30F4011, pero existe un modo matemático para conseguirlo.

Este método consiste en una ecuación diferencial de segundo orden para reproducir la señal senoidal.

$$y[n] = B1 * x[n-1] + A1 * y[n-1] + A0 * y[n-2]$$

con : $B1 = 1$

$A0 = -1$

$A1 = 2\cos(\Theta)$

$x[n]$ = Amp si n = 0; $x[n] = 0$ en otro caso.

La frecuencia de la señal generada es:

$$F = (\Theta / 2\pi)\, Fs = \frac{Fs}{2 * \pi} \arccos(A1/2)$$

Siendo *Fs* la frecuencia de muestreo. *A1* debe estar entre -2 y 2 para cualquier frecuencia. La amplitud de la oscilación es 1/sen(Θ) veces el impulso inicial *x[n]*.

4.4.4.2. Descripción de la solución.

La solución es parecida a las anteriores, solo hay que tener dos variables para guardar las últimas dos muestras de salida, tal y como indica la expresión anterior.

La solución, consiste en implementar la función matemática descriptiva de la señal senoidal, eligiendo un parámetro de A1 entre -1 y 1 al ser otros valores no representables en formato “fractional”.

```
fractional OndaCoseno(){
        fractional Salida;
        XBUFF[2]=XBUFF[1];
        XBUFF[1]=XBUFF[0];
        if(i=0){
                XBUFF[0]=0x7fff;
        }
        else{
                XBUFF[0]=XBUFF[1]-XBUFF[2];
        }
        Salida=XBUFF[0];
        return Salida;
}
```

4.4.4.3. Restricciones.

Como se acaba de comentar, no se puede elegir el parámetro *A1* en todos sus posibles valores, tampoco se podrán generar todas las frecuencias como ocurría en las anteriores generaciones de ondas, por lo demás, el procesado no presenta ninguna restricción importante.

4.5. PRÁCTICA 3. FILTRO DIGITAL.

Se entiende por filtro digital a un sistema que dependiendo de las variaciones de la señales de entrada en el tiempo y amplitud realiza un procesado matemático sobre dicha señal, obteniéndose en la salida el resultado del procesamiento matemático o la señal de salida.

En la figura 4.11 se muestra el esquema básico de un filtrado digital, donde el bloque filtro digital corresponde a cualquier operación matemática aplicada sobre la entrada para lograr la salida.

Figura 4.11. Esquema básico de un filtrado digital.

El filtrado digital consiste en la realización interna de un procesado de datos de entrada. En general, el proceso de filtrado es un muestreo digital de la señal de entrada, el procesamiento posterior considerando las entradas actual y anteriores, y por último la reconstrucción de la señal de salida.

La mecánica es: tomar muestras actuales y algunas anteriores, (previamente almacenadas) para multiplicarlas por unos coeficientes definidos, también se podría usar valores de la salida anteriores, y multiplicarlos por otros coeficientes. Todos los resultados de estas multiplicaciones se suman, obteniéndose la salida para el instante actual [10].

En este apartado, se van a describir los filtros más usados, como son los filtros FIR e IIR, se detalla un método de obtención de los coeficientes para ambos y se verán las restricciones que conlleva usar el dsPIC30F4011 para llevar a cabo estos filtros.

4.5.1. FILTRO FIR.

4.5.1.1. Definición de un filtro FIR.

FIR es un acrónimo en inglés para (*Finite Impulse Response*) o Respuesta Finita al Impulso. Se trata de un tipo de filtros digitales en el que, como su nombre indica, si la entrada es una señal impulso, la salida tendrá un número finito de términos no nulos.

Para obtener la salida actual sólo se basa en entradas actuales y anteriores, como muestra el diagrama básico de un FIR en la figura 4.12. La salida puede expresarse como la convolución de la señal de entrada *x(n)* con la respuesta impulsional *h(n)*.

$$y_n = \sum_{k=0}^{N-1} h_k x_{n-k}$$

Aplicando la transformada Z a la expresión anterior:

$$H(z) = \sum_{k=0}^{N-1} h_k z^{-k} = h_0 + h_1 z^{-1} + \cdots + h_{N-1} z^{-(N-1)}$$

Figura 4.12. Esquema básico de un filtro FIR.

La estructura básica de un FIR es la mostrada en la figura 4.13:

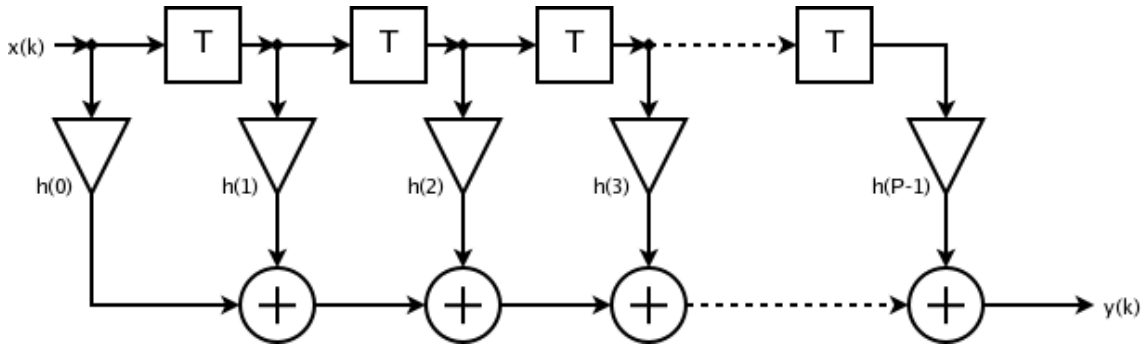

Figura 4.13. Estructura de un filtro FIR.

Pueden hacerse multitud de variaciones de esta estructura. Hacerlo como varios filtros en serie, en cascada, etc. Estos filtros tienen todos los polos en el origen, por lo que son estables. Los ceros se presentan en pares de recíprocos si el filtro se diseña para tener fase lineal.

Los filtros FIR tienen la gran ventaja de que pueden diseñarse para ser de fase lineal, lo cual hace que presenten ciertas propiedades en la simetría de los coeficientes. Este tipo de filtros tiene especial interés en aplicaciones de audio. Además son siempre estables. Por contra también tienen la desventaja de necesitar un orden mayor respecto a los filtros IIR para cumplir las mismas características. Esto se traduce en un mayor gasto computacional [7]

4.5.1.2. Obtención de coeficientes para un FIR. Matlab: la función “fir1”.

En un filtro digital FIR, los coeficientes son lo más importante. Un filtro digital consiste en una serie de multiplicaciones y sumas, pero los coeficientes de esas multiplicaciones y los signos de esas sumas determinaran las características de dicho filtro, como por ejemplo la atenuación o la frecuencia de corte.

Para la obtención de estos coeficientes se ha usado el siguiente método basado en Matlab, y una función de esta aplicación matemática: la función “fir1”.

La función “fir1” de Matlab puede proporcionar los coeficientes en formato flotante para un filtro FIR. Para ello consta de una serie de parámetros de entrada que se describirán a continuación.

```
a = fir1(N,Wn,'tipo_filtro');
```

→ N (número de coeficientes).

Es el número de coeficientes que poseerá el filtro FIR a diseñar.

→ ‘tipo de filtro’.

Se elige el tipo de filtro FIR a diseñar. La tabla 4.5 se muestran las distintas opciones que se puede elegir en este parámetro:

FILTRO A DISEÑAR	‘tipo_filtro’
PASO BAJO	Low
PASO ALTO	High
PASO BANDA	Bandpass
BANDA ELIMINADA	Stop

Tabla 4.5. Opciones para el parámetro ‘tipo de filtro’ de la función ‘fir1’.

→ Wn.

Es un valor o array de valores que determinará las frecuencias donde trabajará el filtro. Es un valor dependiente de la frecuencia de muestreo. Teniendo en cuenta el teorema de Nyquist, solo se puede analizar como máximo las frecuencias por debajo de un medio de la frecuencia de muestreo, entonces *Wn* establece el porcentaje donde se llevará a cabo el filtro deseado.

Por ejemplo, para un FIR paso bajo con frecuencia de muestreo 12KHz la frecuencia máxima muestreable será de 6KHz, entonces una *Wn* de 0.5 establece el corte a 3KHz, y una *Wn* de 0.2 establecerá el corte en 6*0.2 KHz. Para un filtro paso alto, *Wn* funciona igual. Para los filtros de paso-banda y banda-eliminada se necesitan dos valores, es decir, necesita dos frecuencias, de inicio y fin de la banda que se eliminará para un filtro FIR banda eliminada o que se dejarán pasar para un filtro FIR paso banda.

→ a (Array de coeficientes).

Corresponde al array devuelto por la función, contiene los coeficientes característicos del filtro FIR diseñado. Estos coeficientes están representados en formato fractional, con lo cual, solo se necesita pasar a hexadecimal y estarán listos para usar en el correspondiente filtro.

4.5.1.3. Descripción de la solución.

En resumen, matemáticamente un filtro FIR tiene la siguiente estructura:

$$y[n] = h[0]x[n] + h[1]x[n - 1] + \ldots\ldots + h[N - 1]x[n - (N - 1)].$$

Un filtro FIR, es un conjunto de multiplicaciones y sumas, con lo que la solución consiste en un bucle para hacer todas las operaciones, por la forma de la ecuación de *y[n]* es la multiplicación de dos arrays factor a factor. Uno de los arrays son los coeficientes del filtro, que se sitúan en memoria de programa como constantes para así ahorrar memoria de datos. El segundo array son las muestras de entrada almacenadas. Éstas sí se almacenan en memoria de datos, puesto que van modificándose cada vez que llega una muestra. A continuación se muestra el algoritmo de resolución del filtro FIR

```
// XBUFF ES EL BUFFER DE ENTRADAS ANTERIORES.
// FIR ES EL ARRAY DE COEFICIENTES DEL FILTRO.
//N = Número de coeficientes – 1.
fractional FiltroFir(fractional Entrada)
{

      fractional muestra_filtrada = 0;
      fractional Aux = 0;
//ACTUALIZACIÓN DEL BUFFER DE MUESTRAS DE ENTRADA ANTERIORES.
      for (i = N  ;  i > 0; i--)
            XBUFF[i]= XBUFF[i-1];
      XBUFF[0]=Entrada;

// PROCESADO DEL FILTRO FIR.
      for (i=0;i<=N;i++)
      {
            muestra_filtrada = multfractional(XBUFF[i],FIR[i]);
            Aux+=muestra_filtrada;
      }
      return Aux;
}
```

4.5.1.4. Restricciones.

Esta práctica de un filtrado FIR se lleva a cabo sobre una señal en tiempo real, es decir, las muestras van llegando en tiempo real con una cierta frecuencia al dsPIC30F4011 desde el Si3000, esto hace limitar en varias medidas al filtrado FIR.

Para comenzar con las restricciones se comentará que lo ideal de este procesado hubiese sido usar un tipo de datos flotante, con ilimitados bits para la parte decimal y entera, pero la realidad del dsPIC30F4011 obligaba a números flotantes de 64 bits, con los cuales no se perdía mucha información. Estos números tenían la contrapartida de un excesivo tiempo de cálculo. Esto hace que sea totalmente inviable usar números flotantes.

Por esta razón se usaron los números fraccionales o 'fractional'. Estos números solo poseen 16 bits, pero sus operaciones no son tan lentas y se puede llevar a cabo un buen procesado sin olvidar que se pierde información en el filtrado.

Por tanto, se tiene que llegar a un compromiso en estos momentos, puesto que tampoco se puede llevar a cabo infinitas multiplicaciones en formato fractional, mientras llega la siguiente muestra. Los parámetros a tener en cuenta son la frecuencia de muestreo y el número de coeficientes.

Para un mayor número de coeficientes del filtro FIR seria necesaria una menor frecuencia para llevar a cabo todo el procesado, pero también es cierto que la calidad del filtrado depende de los coeficientes, por lo que no se puede sacrificar en exceso el número de ellos. Posteriormente, se verá en el apartado de pruebas, cómo se relacionan el número de coeficientes y la frecuencia de muestreo en la ejecución de distintos filtros FIR.

Otra limitación del dsPIC30F4011 es el uso de un tipo de 16 bits, esto en una multiplicación hace que para números pequeños, el resultado no sea representable en 16 bits, existiendo una pequeña pérdida de información, que puede hacer bajar la calidad del filtro.

4.5.1.5. Coeficientes usados.

Para hacer un amplio número de pruebas, se optó por hacer un filtro de cada tipo con diferente número de coeficientes y así comprobar a qué frecuencia de muestreo dejaba de funcionar el filtro, es decir, a que frecuencia llegaba una muestra antes de procesar la anterior.

Se eligieron estas condiciones para la prueba: 4 filtros de 17 coeficientes, 4 filtros de 27 y 37 coeficientes. Se optó por filtros paso bajo, paso alto, paso banda y banda eliminada. En las tablas 4.3, 4.4 y 4.5 se muestran los coeficientes para los 12 filtros FIR diseñados.

Mediante los siguientes comandos de Matlab, se consiguieron los coeficientes de los 4 filtros FIR de 17 coeficientes que se exponen en la tabla 4.6:

- a = fir1(16, 0.5, 'high')
- a = fir1(16, 0.5, 'low')
- a = fir1(16, [0.4,0.6],'bandpass')
- a = fir1(16, [0.4,0.6],'stop')

	Paso Alto	**Paso Bajo**	**Paso Banda**	**Banda Eliminada**
Coeficiente 1	-0.0000	-0.0000	0.0051	-0.0038
Coeficiente 2	-0.0052	-0.0052	-0.0000	0.0000
Coeficiente 3	-0.0000	0.0000	-0.0294	0.0218
Coeficiente 4	-0.0232	0.0232	0.0000	0.0000
Coeficiente 5	-0.0000	-0.0000	0.1197	-0.0821
Coeficiente 6	-0.0761	-0.0761	-0.0000	0.0000
Coeficiente 7	-0.0000	-0.0000	-0.2193	0.1625
Coeficiente 8	-0.3077	0.3077	-0.0000	0.0000
Coeficiente 9	0.5009	0.5009	0.2710	0.8031
Coeficiente 10	-0.3077	0.3077	-0.0000	0.0000
Coeficiente 11	-0.0000	-0.0000	-0.2193	0.1625
Coeficiente 12	-0.0761	-0.0761	-0.0000	0.0000
Coeficiente 13	-0.0000	-0.0000	0.1197	-0.0821
Coeficiente 14	-0.0232	0.0232	0.0000	0.0000
Coeficiente 15	-0.0000	-0.0000	-0.0294	0.0218
Coeficiente 16	-0.0052	-0.0052	-0.0000	0.0000
Coeficiente 17	-0.0000	-0.0000	0.0051	-0.0038

Tabla 4.6. Coeficientes para los Filtros FIR de 17 coeficientes.

Mediante los siguientes comandos de Matlab, se consiguieron los coeficientes de los 4 filtros FIR de 27 coeficientes que se exponen en la **tabla 4.7.**

- a = fir1(26, 0.5, 'high')
- a = fir1(26, 0.5, 'low')
- a = fir1(26, [0.4,0.6], 'bandpass')
- a = fir1(26, [0.4,0.6], 'stop')

	Paso Alto	Paso Bajo	Paso Banda	Banda Eliminada
Coeficiente 1	-0.0020	0.0020	-0.0000	-0.0000
Coeficiente 2	-0.0000	-0.0000	-0.0031	0.0029
Coeficiente 3	0.0038	-0.0038	-0.0000	0.0000
Coeficiente 4	-0.0000	-0.0000	-0.0000	-0.0000
Coeficiente 5	-0.0098	0.0098	0.0000	0.0000
Coeficiente 6	-0.0000	-0.0000	0.0186	-0.0176
Coeficiente 7	0.0220	-0.0220	-0.0000	0.0000
Coeficiente 8	-0.0000	-0.0000	-0.0635	0.0599
Coeficiente 9	-0.0447	0.0447	0.0000	0.0000
Coeficiente 10	-0.0000	-0.0000	0.1281	-0.1210
Coeficiente 11	0.0937	-0.0937	-0.0000	0.0000
Coeficiente 12	-0.0000	-0.0000	-0.1872	0.1768
Coeficiente 13	-0.3135	0.3135	-0.0000	0.0000
Coeficiente 14	0.4941	0.4941	0.2113	0.7979
Coeficiente 15	-0.3135	0.3135	-0.0000	0.0000
Coeficiente 16	-0.0000	-0.0000	-0.1872	0.1768
Coeficiente 17	0.0937	-0.0937	-0.0000	0.0000
Coeficiente 18	-0.0000	-0.0000	0.1281	-0.1210
Coeficiente 19	-0.0447	0.0447	-0.0000	0.0000
Coeficiente 20	-0.0000	-0.0000	-0.0635	0.0599
Coeficiente 21	0.0220	-0.0220	-0.0000	0.0000
Coeficiente 22	-0.0000	-0.0000	0.0186	-0.0176
Coeficiente 23	-0.0098	0.0098	0.0000	0.0000
Coeficiente 24	-0.0000	-0.0000	-0.0000	-0.0000
Coeficiente 25	0.0038	-0.0038	-0.0000	0.0000
Coeficiente 26	-0.0000	-0.0000	-0.0031	0.0029
Coeficiente 27	-0.0020	0.0020	-0.0000	-0.0000

Tabla 4.7. Coeficientes para los Filtros FIR de 27 coeficientes.

Mediante los siguientes comandos de Matlab, se consiguieron los coeficientes de los 4 filtros FIR de 37 coeficientes que se exponen en la tabla 4.8.

- a= fir1(36,0.5, 'high')
- a= fir1(36,0.5, 'low')
- a= fir1(36,[0.4,0.6], 'bandpass')
- a= fir1(36,[0.4,0.6], 'stop'

	Paso Alto	Paso Bajo	Paso Banda	Banda Eliminada
Coeficiente 1	-0.0000	0.0000	0.0017	-0.0017
Coeficiente 2	-0.0016	0.0016	-0.0000	0
Coeficiente 3	-0.0000	-0.0000	-0.0041	0.0041
Coeficiente 4	0.0030	-.0030	0.0000	0.0000
Coeficiente 5	-0.0000	-0.0000	0.0081	-0.0081
Coeficiente 6	-0.0060	0.0060	-0.0000	-0.0000
Coeficiente 7	-0.0000	-0.0000	-0.0097	0.0097
Coeficiente 8	0.0111	-.0111	-0.0000	0.0000
Coeficiente 9	-0.0000	-0.0000	-0.0000	-0.0000
Coeficiente 10	-0.0191	0.0191	0.0000	0.0000
Coeficiente 11	-0.0000	-0.0000	0.0290	-0.0290
Coeficiente 12	0.0317	-.0317	-0.0000	0.0000
Coeficiente 13	-0.0000	-0.0000	-0.0777	0.0778
Coeficiente 14	-0.0531	0.0531	0.0000	0.0000
Coeficiente 15	-0.0000	-0.0000	0.1350	-0.1352
Coeficiente 16	0.0994	-0.0994	-0.0000	0.0000
Coeficiente 17	-0.0000	-0.0000	-0.1819	0.1821
Coeficiente 18	-0.3157	0.3157	-0.0000	0.0000
Coeficiente 19	0.4994	0.4994	0.1990	0.8008
Coeficiente 20	-0.3157	0.3157	-0.0000	0.0000
Coeficiente 21	-0.0000	-0.0000	-0.1819	0.1821
Coeficiente 22	0.0994	-0.0994	-0.0000	0.0000
Coeficiente 23	-0.0000	-0.0000	0.1350	-0.1352
Coeficiente 24	-0.0531	0.0531	0.0000	0.0000
Coeficiente 25	-0.0000	-0.0000	-0.0777	0.0778
Coeficiente 26	0.0317	-0.0317	-0.0000	0.0000
Coeficiente 27	-0.0000	-0.0000	0.0290	-0.0290
Coeficiente 28	-0.0191	0.0191	0.0000	0.0000
Coeficiente 29	-0.0000	-0.0000	-0.0000	-0.0000
Coeficiente 30	0.0111	-0.0111	-0.0000	0.0000
Coeficiente 31	-0.0000	-0.0000	-0.0097	0.0097
Coeficiente 32	-0.0060	0.0060	-0.0000	-0.0000
Coeficiente 33	-0.0000	-0.0000	0.0081	-0.0081
Coeficiente 34	0.0030	-0.0030	0.0000	0.0000
Coeficiente 35	-0.0000	-0.0000	-0.0041	0.0041
Coeficiente 36	-0.0016	0.0016	-0.0000	0
Coeficiente 37	-0.0000	-0.0000	0.0017	-0.0017

Tabla 4.8. Coeficientes para los Filtros FIR de 37 coeficientes.

4.5.1.6. Resultados.

Para los filtros FIR, tras evaluar las pruebas se ha llegado a varias conclusiones en cuanto a la frecuencia, el número de coeficientes y la calidad del filtro.

Respecto a la relación entre el número de coeficientes y la frecuencia máxima que permite el dsPIC30F4011, se establecen estas características resumidas en la tabla 4.9:

Número de coeficientes	Frecuencia de fallo (Hz)
17	Nunca falla.
27	9000, no falla.
37	6250, no falla.

Tabla 4.9. Relación número coeficientes – Frecuencia máxima filtro FIR.

Como se ve en la tabla 4.9, al aumentar el número de coeficientes, baja la frecuencia máxima a la que funciona correctamente el códec Si3000. Esto es debido a dos causas: la primera es que aumentan las multiplicaciones a realizar y la segunda es que se disminuye el tiempo disponible para hacerlas.

Respecto a la calidad del corte de la señal, no hay grandes diferencias, puesto que el corte se produce a la misma frecuencia, habiendo alguna diferencia en la velocidad de la caída para las frecuencias eliminadas en cada filtro. Esto se podría deber a que las multiplicaciones por coeficientes pequeños tienden a cero debido a la falta de bits para representar dichos valores, y para filtros con más coeficientes abundan más los coeficientes bajos.

4.5.2. FILTRO IIR.

4.5.2.1. Definición de un filtro IIR.

IIR es una sigla en inglés para *Infinite Impulse Response* o Respuesta Infinita al Impulso. Se trata de un tipo de filtros digitales en el que, como su nombre indica, si la entrada es una señal impulso, la salida tendrá un número infinito de términos no nulos, es decir, nunca vuelve al reposo.

La salida de los filtros IIR depende de las entradas actuales y pasadas, como se aprecia en la figura 4.14 y además de las salidas en instantes anteriores. Esto se consigue mediante el uso de realimentación de la salida.

$$y_n = b_0 x_n + b_1 x_{n-1} + \cdots + b_N x_{n-N} - a_1 y_{n-1} - a_2 y_{n-2} - \cdots - a_M y_{n-M}$$

Donde *a* y *b* son los coeficientes del filtro. El orden es el máximo entre los valores de *M* y *N*.

Aplicando la transformada Z a la expresión anterior:

$$H(z) = \frac{\sum_{k=0}^{M} b_k z^{-k}}{1 - \sum_{k=1}^{N} a_k z^{-k}}$$

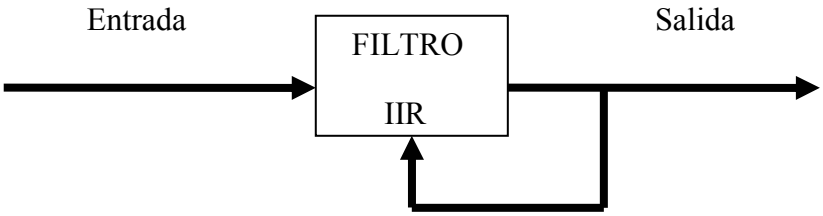

Figura 4.14. Esquema básico de un filtro IIR.

Hay numerosas formas de implementar los filtros IIR. La estructura afecta a las características finales que presentará el filtro como la estabilidad. Otros parámetros a tener en cuenta a la hora de elegir una estructura es el gasto computacional que presenta. La figura 4.15 muestra un ejemplo de estructura de un filtro IIR.

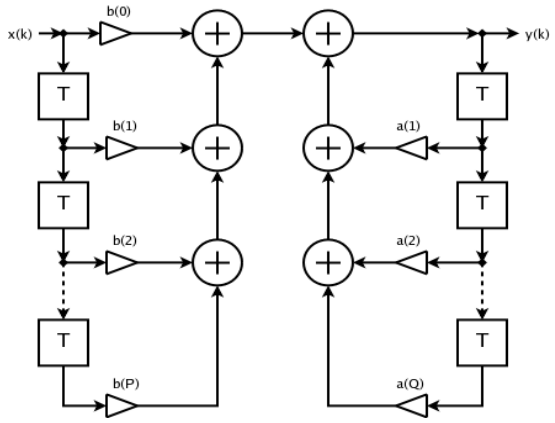

Figura 4.15. Estructura de un filtro IIR.

Este tipo de filtros presenta polos y ceros que determinan la estabilidad y la causalidad del sistema. Cuando todos los ceros están en el interior de la circunferencia unidad se dice que es fase mínima. Si todos están en el exterior es fase máxima. Si algún polo está fuera de la circunferencia unidad el sistema es inestable.

Las principales diferencias respecto a los filtros FIR es que los IIR pueden cumplir las mismas exigencias que los anteriores pero con menor orden de filtro. Esto es importante a la hora de implementar el filtro, pues presenta una menor carga computacional.

Este tipo de filtros pueden ser inestables, aún cuando se diseñen para ser estables. En principio no pueden diseñarse para tener fase lineal pero se pueden aplicar algunas técnicas como el filtrado bidireccional para lograrlo. [8]

4.5.2.2 .Obtención de coeficientes para un filtro IIR. Matlab: la función "butter".

Al igual que en un filtro digital FIR, los coeficientes vuelven a ser lo más importante en un filtro digital IIR, puesto que determinaran las características de dicho filtro.

Para la obtención de estos coeficientes, se usa un método basado en Matlab y la función "butter", como se verá a continuación genera los coeficientes de un filtro IIR de *Butterwoth.*

Esta función, posee muchas similitudes con la función "fir1" usada para la obtención de los parámetros de un filtro FIR. Al igual que "fir1", devuelve los coeficientes en formato flotante y consta de una serie de parámetros de entrada idénticos a la función "fir1", pero posee, al ser un filtro IIR, dos arrays de coeficientes de salida.

Aquí se muestra como se hace una llamada a esta función:

```
[a,b] = butter(N,Wn,'tipo_filtro');
```

Por su similitud con la función "fir1", solo se comentará brevemente los parámetros de salida, ya que los de entrada también aparecen en la llamada a "fir1" ya comentada.

→ a,b (coeficientes).

Son los coeficientes característicos del filtro IIR diseñado. Son dos arrays que contienen dichos coeficientes. al igual que para la función "fir1" que diseñaba filtros FIR, la función "butter" devuelve el resultado en formato fractional y solo es necesario pasar esos valores a formato hexadecimal para usar en la correspondiente función del filtro.

4.5.2.3. Descripción de la solución.

En resumen matemáticamente un filtro IIR tiene la siguiente estructura:

$$y[n] = b0x[n] + b1x[n - 1]+\ldots\ldots+ bNx[n – (N - 1)] + a1y[n - 1] + a2y[n - 2]+\ldots.+aNy[n - N]$$

El filtro IIR es un conjunto de multiplicaciones y sumas, y la solución es muy parecida a la del filtro FIR, pero ahora se tienen dos arrays de coeficientes y dos buffers de muestras pasadas para las entradas y las salidas respectivamente. A continuación se muestra el algoritmo con su solución.

```
// XBUFF, YBUFF buffer de entradas y salidas pasadas respectivamente.
//IIRA ,IIRB coeficientes del filtro IIR.
//N = Número de coeficientes – 1.
fractional FiltroIIR(fractional Entrada)
{
      fractional Muestra_filtrada,Auxfra;
      fractional Salida,Aux1=0x0000,Aux2=0x0000;
//ACTUALIZACIÓN DE LOS BUFFER DE ENTRADA Y SALIDA.
      for (i = N ;  i > 0; i--)
      {
            YBUFF[i]= YBUFF[i-1];
            XBUFF[i]= XBUFF[i-1];
      }
      XBUFF[0]=Entrada; //Actualizo la entrada
//PRIMERA PARTE DEL FILTRADO, Entradas.
      for (i=0;i<=N;i++) .
      {
            Muestra_filtrada = multfractional(XBUFF[i],IIRB[i]);
            Aux1 += Muestra_filtrada;
      }
//SEGUNDA PARTE DEL FILTRADO, Salidas.
      for (i=1;i<=N;i++)
      {
            Muestra_filtrada=multfractional(YBUFF[i],IIRA[i]);
            Aux2+=Muestra_filtrada;
      }
// SALIDA = SUMA DE AMBOS FILTRADOS DE ENTRADA Y SALIDA.
      YBUFF[0]=Aux1+Aux2;

      return YBUFF[0];
}
```

4.5.2.4. Restricciones.

En los filtros IIR, se poseen varias restricciones adicionales a la pérdida de información de la multiplicación de 16 bits y la limitación de número de coeficientes o limitación de frecuencia de muestreo típica de los filtros FIR anteriormente comentado. Para empezar, hay que valorar que no todos los filtros IIR poseen unos coeficientes de formato fractional, sino de tipo flotante, lo cual no era viable por el alto tiempo de computación de sus

multiplicaciones. Eesto hace que no todos los filtros IIR sean realizables con el dsPIC30F4011.

Otra restricción, es que en un filtro IIR, normalmente los coeficientes son casi todos positivos y muchos de ellos cercanos a la unidad, por lo que existe un riesgo real de conseguir que la salida sea superior a uno, lo cual no es representable en formato fractional. Esto significaría un ruido, y guardar en el buffer de salida unas muestras saturadas incorrectas.

Como solución, se optó por dividir los coeficientes por algún valor, aunque esto conlleva unas multiplicaciones con operandos pequeños, y su consecuente perdida de información en las multiplicaciones.

Unido a las limitaciones del FIR, con multiplicaciones de números pequeños se pierde precisión y no poder tener un número de coeficientes infinito. Hay que añadir que no todos los filtros IIR poseen coeficientes representables en el formato de datos fractional, y la necesidad de escalar los coeficientes para no saturar la salida.

4.5.2.5. Coeficientes usados.

Para conseguir hacer unas pruebas de la calidad y limitaciones reales del filtro IIR, como sucedió con los filtros FIR, se eligieron 2 bloques de 2 filtros distintos cada uno, es decir, para cada tipo de filtro.

El hecho de no usar filtros paso banda o banda eliminada se debe a que los coeficientes obtenidos no se conseguían en formato fractional. Por otra parte, una vez se alejaban de 15 coeficientes los filtros IIR era imposible encontrar coeficientes de formato fractional.

Debido al doble de procesado necesario en un filtrado IIR, se optó por usar 8 y 13 como el número de coeficientes a usar en las pruebas. Las frecuencias de corte se elegirán del mismo modo que se eligieron para los filtros FIR para hacer a su vez una cierta relación entre ellos.

Mediante los siguientes comandos de Matlab, se consiguieron los 8 coeficientes de los 2 filtros IIR, que se exponen en la tabla 4.10.

- [b,a]= butter(7, 0.5, 'high')
- [b,a]= butter(7, 0.5, 'low')

	Paso Alto Coeficientes b	Paso Alto Coeficientes a	Paso bajo Coeficientes b	Paso bajo Coeficientes a
Coeficiente 1	0.0166	1.0000	0.0166	1.0000
Coeficiente 2	-0.1160	-0.0000	0.1160	-0.0000
Coeficiente 3	0.3479	0.9200	0.3479	0.9200
Coeficiente 4	-0.5798	-0.0000	0.5798	0.0000
Coeficiente 5	0.5798	0.1927	0.5798	0.1927
Coeficiente 6	-0.3479	-0.0000	0.3479	0.0000
Coeficiente 7	0.1160	0.0077	0.1160	0.0077
Coeficiente 8	-0.0166	-0.0000	0.0166	-0.0000

Tabla 4.10. Coeficientes para los Filtros IIR de 8 coeficientes.

Mediante los siguientes comandos de Matlab, se consiguieron los 13 coeficientes de los 2 filtros IIR, que se exponen en la tabla 4.11.

- [b,a]= butter(12, 0.5, 'high')
- [b,a]= butter(12, 0.5, 'low')

	Paso Alto Coeficientes b	Paso Alto Coeficientes a	Paso bajo Coeficientes b	Paso bajo Coeficientes a
Coeficiente 1	0.0009	1.0000	0.0009	1.0000
Coeficiente 2	-0.0109	-0.0000	0.0109	-0.0000
Coeficiente 3	0.0597	1.6178	0.0597	1.6178
Coeficiente 4	-0.1989	-0.0000	0.1989	-0.0000
Coeficiente 5	0.4476	0.8763	0.4476	0.8763
Coeficiente 6	-0.7162	-0.0000	0.7162	-0.0000
Coeficiente 7	0.8356	0.1928	0.8356	0.1928
Coeficiente 8	-0.7162	-0.0000	0.7162	-0.0000
Coeficiente 9	0.4476	0.0167	0.4476	0.0167
Coeficiente 10	-0.1989	-0.0000	0.1989	-0.0000
Coeficiente 11	0.0597	0.0004	0.0597	0.0004
Coeficiente 12	-0.0109	-0.0000	0.0109	-0.0000
Coeficiente 13	0.0009	0.0000	0.0009	0.0000

Tabla 4.11. Coeficientes para los Filtros IIR de 13 coeficientes.

Para poder usar estos coeficientes en un procesado IIR con el dsPIC30F4011, habrá que pasar los resultados a hexadecimal, además de dividir los coeficientes por algún factor para no tener saturación en la salida.

4.5.2.6. Resultados.

Para poder implementar estos filtros sin saturación, se dividieron los coeficientes del filtro de 8 coeficientes por 2 y el de 13 coeficientes por 3. Así se logró una salida audible y sin saturación del formato flotante.

En cuanto a las frecuencias máximas, se obtuvo que para el filtro IIR de 8 coeficientes, nunca fallaba y para el de 13 coeficientes, lo hace para frecuencias mayores de 9,8 Khz. Estos resultados, concuerdan con los obtenidos para el filtro FIR, puesto que el filtro IIR tiene la particularidad de tener el doble de multiplicaciones a realizar que un filtro FIR, por tanto un filtro de 7 coeficientes IIR es equivalente a un filtro FIR de 14 coeficientes.

Comparando el filtro IIR de 13 coeficientes al filtro FIR de 26, en las pruebas del filtro FIR se obtuvo, que para un filtro FIR de 27 coeficientes, la frecuencia máxima esta en 9000Khz, curiosamente un valor cercano a los 9,8KHz del filtro IIR de 26 coeficientes.

Respecto a la calidad del filtrado, tampoco hay muchas diferencias de usar 8 o 13 coeficientes, ya que en ambos la frecuencia de corte esta al 25 % de la frecuencia de muestreo, tal y como se indicó en la función “butter”. La razón debe estar en que las multiplicaciones de los términos muy bajos siempre darán resultados no representables en el formato fractional del motor DSP.

4.5.3. FILTRO FIR ADAPTATIVO.

4.5.3.1. Teoría FIR adaptativo.

En los filtros FIR e IIR implementados anteriormente, los coeficientes de los filtros estaban diseñados de acuerdo a unas especificaciones. En muchas aplicaciones, los coeficientes de los filtros no pueden especificarse a priori debido a un entorno cambiante. Para aplicaciones como esta, es necesario el uso de filtros adaptativos.

Un filtro adaptativo, es un filtro ajustable en el que los coeficientes son actualizados regularmente. Estas actualizaciones se determinan por un algoritmo que intenta optimizar la respuesta del filtro con respecto a algún criterio de rendimiento. El filtro que realiza las modificaciones de la señal y el algoritmo adaptativo son generalmente dos partes distintas del filtro adaptativo, como se muestra en la figura 4.16.

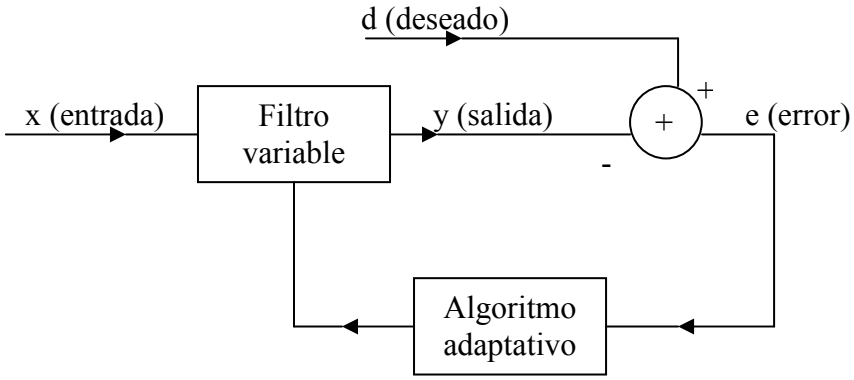

Figura 4.16. Estructura general de un filtro adaptativo

Otra consideración importante en el uso de filtros adaptativos, es la elección del algoritmo de adaptación para ajustar los coeficientes del filtro. Un algoritmo muy usado es el Least Mean Square (LMS).

En el algoritmo LMS se usa una señal de error: $e[n] = d[n] - y[n]$ para ajustar los coeficientes del filtro, donde $d[n]$ es la salida deseada del filtro e $y[n]$ es la salida actual del filtro. Los coeficientes del filtro se actualizan de acuerdo a la ecuación:

$$h[n+1, k] = h[n,k] + 2 * \mu * e[n] * x[n-k] \quad k = 0, 1, 2, \ldots, N-1$$

donde μ es una constante llamada "tamaño de paso", $x[n]$ es la señal de entrada y $h[n, k]$ es el k-ésimo coeficiente del filtro en el tiempo n.

El "tamaño de paso" μ controla la convergencia del algoritmo. Un valor grande de μ hace un gran tamaño de ajuste y, por lo tanto, a una rápida convergencia. Pero, si μ es demasiado grande, el algoritmo no converge. Para asegurar la convergencia, μ se debe elegir en el rango: $0 < \mu < \frac{1}{N * Px}$

donde N es la longitud del filtro FIR adaptativo y P_x es la energía de la señal de entrada. P_x puede aproximarse por: $P_x = \frac{1}{M+1} \sum_{n=0}^{M} x[n]^2$

donde M es una constante [13].

4.5.3.2. El cancelador de eco.

Cancelar el eco de una señal de audio es una de las aplicaciones donde se usa el filtro adaptativo, debido a que el ruido es cambiante.

En la figura 4.17, se muestra el diagrama de un cancelador de ruido. Aquí, *d[n]* es la señal deseada y *x[n]* es la señal contaminada con la señal *s[n]*. La entrada al filtro es en este caso *s'[n],* que es la medida del ruido que será similar a *s[n]* pero no necesariamente exactamente igual. *s'[n]* no debe contener ninguna componente de *x[n]*.

Se usa un filtro adaptativo para estimar el ruido en *d[n]* y el ruido estimado *y[n]* se resta de *d[n]* para producir *e[n]*. Si *x[n]* está incorrelado con *s[n],* la salida del filtro *y[n]* se aproximará al ruido *s[n]*. La diferencia *d[n] – y[n]* aproximará *x[n]* y tendremos un cancelador de ruido.

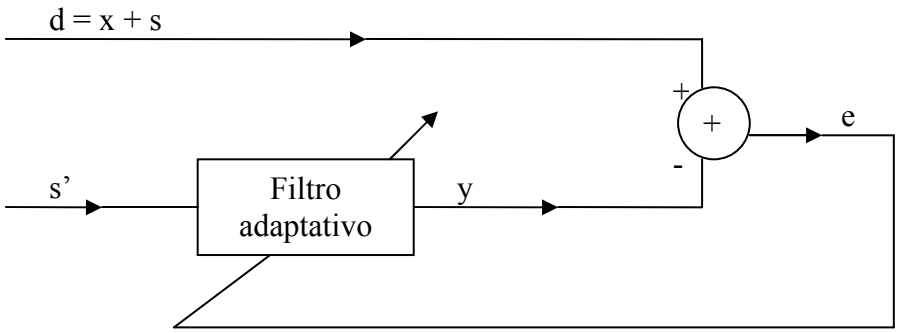

Figura 4.17. Cancelación de ruido [13].

4.5.3.3. Restricciones.

El valor del tamaño de paso (μ), es un valor delicado en este algoritmo, pues establecerá si el sistema converge a una solución o no. El rango de valores de este parámetro es del rango 10e-8, por lo que de entrada es irrepresentable en el dsPIC30F4011 mediante el formato fractional.

Se intento ejecutar el algoritmo adaptativo para el cancelador de ruido, usando valores del orden de 10e-3 o 10e-2, por ser un número representable en fractional y no muy pequeño, evitando así multiplicaciones que tiendan a cero. El resultado fue que el algoritmo no converge como se preveía, por lo no es ejecutable el filtro FIR adaptativo.

4.6. ECUALIZADOR.

Para acabar con las prácticas, se diseñará algo un poco más complejo, un ecualizador, pero como se verá a continuación, las restricciones del dsPIC30F4011 harán que no sea una práctica demasiado compleja.

Se comentará brevemente qué es un ecualizador, se definirá el ecualizador a diseñar en concreto, que parámetros usar y modificar. Por último se añadirá una pequeña interfaz gráfica mediante el LCD para poder cambiar dichos parámetros en tiempo de ejecución.

4.6.1. DEFINICIÓN.

El ecualizador es un dispositivo que procesa señales de audio. Para modificar el contenido de las frecuencias de las señales que procesa Cambia las amplitudes de sus coeficientes de Fourier. Esto hace que se dé diferentes volúmenes o amplitud para distintas frecuencias.

Con un ecualizador se podría seleccionar ciertas bandas de frecuencias o se podría acentuar la audición de unas bandas, según como se configurase el ecualizador. Para ello, se podría definir a un ecualizador como un conjunto de filtros paso banda donde cada banda posee una serie de atenuaciones o ganancias para acentuar o atenuar, según se configure en cada momento, como se puede ver en la figura 4.18.

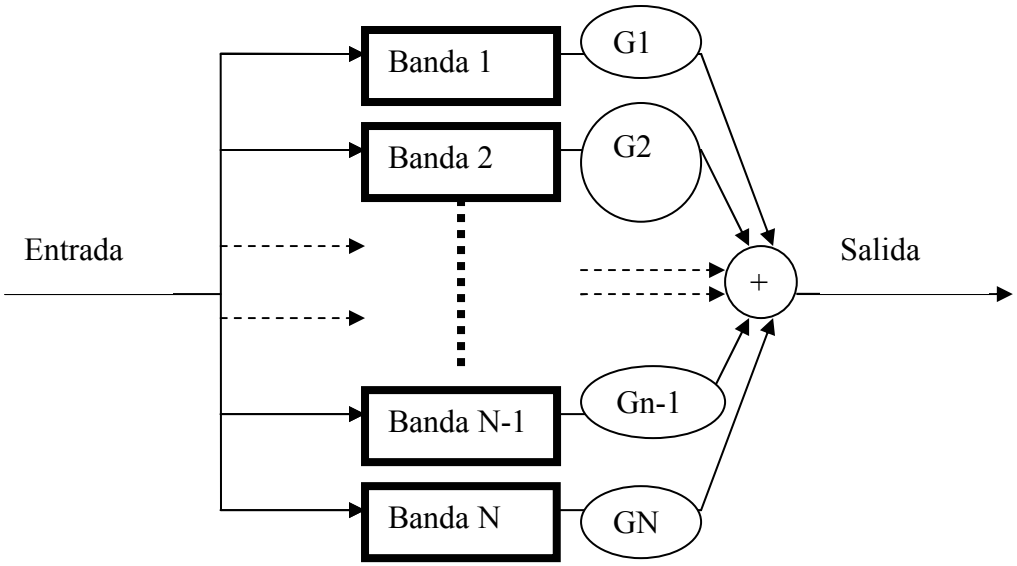

Figura 4.18. Esquema de un ecualizador.

4.6.2. DESCRIPCIÓN DE LA SOLUCIÓN.

Como se ve en el diagrama de bloques de un ecualizador, se trata básicamente de hacer una batería de filtros a una señal de entrada, para luego sumar las señales parciales de cada filtro y sacarlas por la salida. Si los filtros fueran ideales, para los de ganancia 1, la suma de las salidas de cada filtro debería dar la entrada.

La solución, se puede diseñar con filtros IIR o FIR, esto es indiferente para lograr el ecualizador. Se podría decir que lo más importante es lograr filtros muy cercanos al ideal, y que para frecuencias fuera de la banda de paso de cada filtro la atenuación fuera la máxima. Pero ante todo habrá que buscar filtros que se puedan ejecutar en el dsPIC30F4011. Aquí se muestra un ejemplo del algoritmo del ecualizador mediante filtros FIR.

```
// EA,EB,EC,ED array de coeficientes.
// XBUFF array de entradas anteriores.
// GEcu1, GEcu2, GEcu3, GEcu4 son variables donde cargar la atenuación elegida en el menú.
//N = Número de coeficientes – 1.
fractional ECUALIZADOR(fractional Entrada)
{
        int i;
        fractional m_filtrada1,m_filtrada2,m_filtrada3,m_filtrada4;
        fractional Salida1=0x0000,Salida2=0x0000,Salida3=0x0000,Salida4=0x0000;
        fractional Salida1G,Salida2G,Salida3G,Salida4G;
        fractional Salida;
    // ACTUALIZACIÓN DE LAS ENTRADAS ANTERIORES Y CARGA DE LA ACTUAL
        for (i = N  ;  i > 0; i--)
        {
                XBUFF[i]= XBUFF[i-1];
        }
        XBUFF[0]=Entrada;
    // FILTRADO FIR PARA LOS 4 FILTROS.
        for (i=0;i<=N;i++)
        {
                m_filtrada1 = multfractional(XBUFF[i],EA[i]);
                Salida1+=m_filtrada1;
                m_filtrada2 = multfractional(XBUFF[i],EB[i]);
                Salida2+=m_filtrada2;
                m_filtrada3 = multfractional(XBUFF[i],EC[i]);
                Salida3+=m_filtrada3;
                m_filtrada4 = mult fractional(XBUFF[i],ED[i]);
                Salida4+=m_filtrada4;
        }
    // APLICACIÓN DE LAS ATENUACIONES.
        Salida1G=multfractional(Salida1,GEcu1);
        Salida2G=multfractional(Salida2,GEcu2);
        Salida3G=multfractional(Salida3,GEcu3);
        Salida4G=multfractional(Salida4,GEcu4);
    // SALIDA = SUMA DE SALIDAS PARCIALES.
        Salida=Salida1G+Salida2G+Salida3G+Salida4G;
        return Salida;
    }
```

Como se verá en las restricciones, se usaron filtros FIR para conseguir el ecualizador, por lo tanto, solo se necesita un buffer de las entradas pasadas de tamaño igual al número de coeficientes usado por los filtros FIR. Posteriormente se llevará a cabo la batería de filtros FIR para luego sumar las salidas que están multiplicadas por un variable, que determinará la atenuación de cada banda, tal y como se vio en el diagrama de bloques del ecualizador.

4.6.3. RESTRICCIONES.

La primera restricción esta enfocada al número de multiplicaciones que se pueden ejecutar antes de la llegada de otra muestra, tal y como ocurría en los filtros FIR e IIR en función de la frecuencia de muestreo. Es por esto que se eligen filtros FIR para los filtros paso banda, puesto que necesitan menos coeficientes. Además, como se puede recordar del apartado de los filtros IIR, estos filtros poseen muchas restricciones para diseñarlos en un dsPIC, como por ejemplo, que los filtros paso banda IIR no se pueden hacer en un dsPIC. Además, hereda de los filtros FIR sus otras restricciones como el hecho de que una multiplicación de datos en formato fractional hace perder información, puesto que el resultado es un valor no representable en 16 bits.

4.6.4. ECUALIZADOR DISEÑADO.

Para la aplicación del ecualizador, se ha optado por usar 4 bandas de 13 coeficientes, todas las bandas son equidistantes, como se muestra en la figura 4.19.

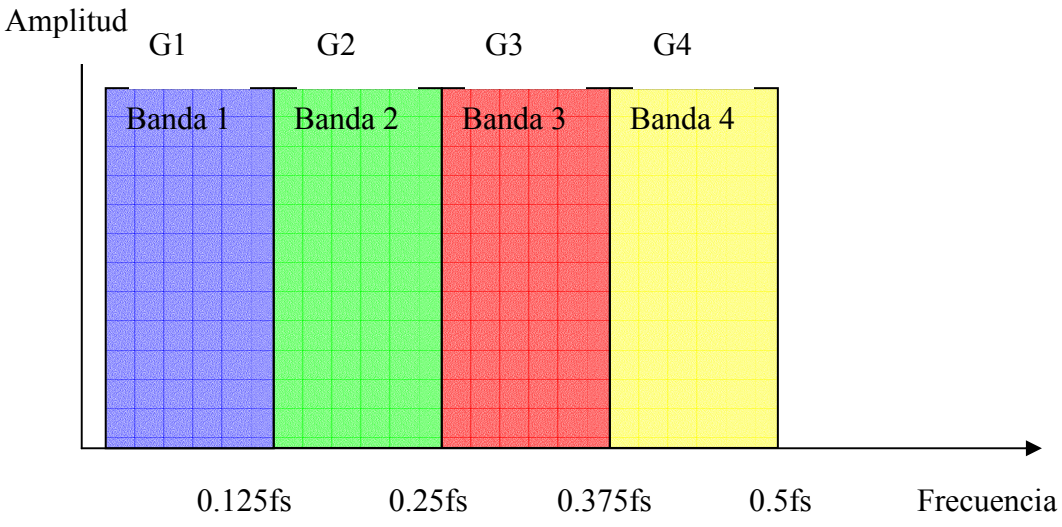

Figura 4.19. Distribución de las bandas respecto la frecuencia de muestreo.

En la tabla 4.12 se muestran ejemplos de las frecuencias de corte de las bandas según ciertas frecuencias de muestreo.

Frecuencia muestreo	Inicio Banda 1	Fin Banda 1	Inicio Banda 2	Fin Banda 2	Inicio Banda 3	Fin Banda 3	Inicio Banda 4	Fin Banda 4
2000	0	250	250	500	500	750	750	1000
3000	0	375	375	750	750	1125	1125	1500
5000	0	625	625	1250	1250	1875	1875	2500
10000	0	1250	1250	2500	2500	3750	3750	5000
12500	0	1562,5	1562,5	3125	3125	4687,5	4687,5	6250

Tabla 4.12. Frecuencias de las bandas respecto la frecuencia de muestreo.

Mediante los siguientes comandos de Matlab, se consiguieron los coeficientes para los filtros deseados:

- a=fir1(12,[0.01,0.25],'bandpass')
- a=fir1(12,[0.25,0.5],'bandpass')
- a=fir1(12,[0.5,0.75],'bandpass')
- a=fir1(12,[0.75,0.99],'bandpass')

En la tabla 4.13, se muestran los coeficientes obtenidos para las 4 bandas del ecualizador mediante filtros FIR de 13 coeficientes.

	Banda 1	Banda 2	Banda 3	Banda 4
Coeficiente 1	-0.0062	0.0060	0.0060	-0.0062
Coeficiente 2	-0.0097	0.0218	-0.0218	0.0097
Coeficiente 3	-0.0038	-0.0000	0.0000	-0.0038
Coeficiente 4	0.0435	-0.1387	0.1387	-0.0435
Coeficiente 5	0.1424	-0.1737	-0.1737	0.1424
Coeficiente 6	0.2502	0.1240	-0.1240	-02502
Coeficiente 7	0.2975	0.3544	0.3544	0.2975
Coeficiente 8	0.2502	0.1240	-0.1240	-0.2502
Coeficiente 9	0.1424	-0.1737	-0.1737	0.1424
Coeficiente 10	0.0435	-0.1387	0.1387	-0.0435
Coeficiente 11	-0.0038	-0.0000	0.0000	-0.0038
Coeficiente 12	-0.0097	0.0218	-0.0218	0.0097
Coeficiente 13	-0.0062	0.0060	0.0060	-0.0062

Tabla 4.13. Coeficientes de los 4 filtros FIR del ecualizador.

Con estos coeficientes, se ha logrado un ecualizador de 4 bandas de filtros FIR paso-banda, donde las frecuencias de paso de cada banda dependen de la frecuencia de muestreo, abarcando todas ellas por debajo de la mitad del muestreo.

Estos valores habrá que colocarlos en el fichero "procesado.c" como constantes, pero en valor hexadecimal, para lo que habrá que hacer la conversión anteriormente a la ejecución del programa. Si se hiciese en tiempo de ejecución, ese tiempo haría ineficiente el sistema, es decir, consumiría todo el periodo disponible para el filtrado simplemente cambiando de formato los coeficientes.

4.6.5. RESULTADO.

Como se vió en el apartado de filtros FIR, un filtro de 13 coeficientes de manera individual no da ningún problema, posee una buena respuesta y no deja de funcionar en el dsPIC30F4011 para ninguna de las frecuencias posibles del Si3000. Pero la combinación de estos 4 filtros en cascada, conlleva cuadruplicar las multiplicaciones, lo que hace que el ecualizador de 4 bandas de filtros FIR de 13 coeficientes, no funcione a frecuencias mas altas de 6.25 KHz.

Analizando esta frecuencia, se supone que un ecualizador de 4 bandas de 13 coeficientes debería de tener el mismo tiempo de cálculo que un filtro FIR de 52 coeficientes, pero se observa que el ecualizador funciona como un filtro FIR de 37 coeficientes, esto es debido, a que el buffer de muestras de entrada es compartido por los 4 filtros FIR del ecualizador, esto lleva a un ahorro del tiempo de cálculo correspondiente a los accesos a memoria tanto de lectura y escritura ahorrados, que permite hacer mas multiplicaciones.

4.6.6. PARAMETROS VARIABLES. OBTENCIÓN.

Los parámetros a configurar en un ecualizador, son las atenuaciones de cada banda, y para ello habrá que definir los valores a usar y establecer en el menú del ecualizador.

Para el menú, se optó por tener 8 opciones: 0dB, -2dB, -4dB, -6dB, -10dB, -15dB, -20dB, -30dB, que se verán traducidas a una atenuación en decimal mediante la siguiente expresión:

Atenuación = 20lg (p1/p0) (dB).

Si tomamos p0 como referencia a ‘1’, se obtiene la atenuación en decimal en p1, para cualquier valor mostrado en atenuación en dB.

En la tabla 4.14 se muestran los valores obtenidos.

Atenuación en dB	Atenuación en decimal
0dB	1.0000
-2dB	0.794328
-4dB	0.6309570
-6dB	0.5011870
-10dB	0.3162277
-15dB	0.778279
-20dB	0.1000
-30dB	0.031622

Tabla 4.14. Relación de la ganancia en dB y en decimal.

4.6.7. INTERFAZ CON MENÚ PARA EL CONTROL DEL ECUALIZADOR.

El menú del ecualizador posee la misma filosofía de todos los menús anteriores. Posee funciones para moverse por la máquina de estados correspondiente al menú, posee funciones para escribir por el LCD y funciones para ejecutar los cambios en las variables oportunas, en este caso son las descritas en el apartado anterior.

El modo de intercambiar entre el menú del Si3000 y este menú es equivalente al usado en la reverberación, sigue la misma estructura, pero con 4 parámetros a cambiar y 8 opciones cada parámetro.

Las funciones cumplen la misma funcionalidad pero adaptadas a los nuevos textos a poner en el LCD y a los nuevos parámetros a modificar. No se pondrá el diagrama por la similitud con el diagrama de la reverberación.

4.7. CONCLUSIONES.

Para acabar el capítulo, se resumirán las conclusiones obtenidas de llevar a cabo las prácticas en el dsPIC30F4011.

La primera práctica de generación de efectos de sonido, al no poseer demasiada complejidad, no posee grandes conclusiones, pero sí es cierto que los efectos se han quedado un poco cortos, es decir, el eco y delay son muy pequeños, debido a la memoria de datos disponible, por lo que habría sido útil disponer de más memoria o ampliar la memoria existente con una externa.

Para la reverberación, ocurre algo muy parecido, pero en este caso las especificaciones temporales eran mas pequeñas, pero a su vez estrictas, al estar definidos algunos de sus tiempos de modo fijo, con lo que hubo que bajar la frecuencia de muestreo hasta los 4000 Hz.

En la generación de ondas, la conclusión básica es que el códec no es demasiado rápido, esto hace que las ondas generadas no posean demasiada frecuencia. Esta lentitud tiene una contrapartida en los filtros digitales, puesto que en ellos la frecuencia se debe bajar para lograr un buen filtrado.

Tanto en los filtros FIR como en el ecualizador, por ser una suma de filtros FIR, se pueden hacer varias puntualizaciones, porque realmente la calidad de este motor DSP no es demasiado elevada en comparación con un DSP de gama más alta. Los números en flotante no están preparados para ser usados con dicho motor, y las librerías para hacer las conversiones de formatos consumen un tiempo demasiado elevado, de tal modo, que varias conversiones de formato, o varias multiplicaciones en formato flotante hacían el sistema tan lento que apenas se podían diseñar filtros de 2 o 3 coeficientes.

Se podría decir, que el sistema se podría haber hecho más eficiente añadiendo varios detalles que se pueden deducir de las conclusiones anteriores. En primer lugar tener más memoria para datos, habría hecho los efectos más eficientes. Por otro lado, tener un códec más rápido habría dado más juego, por ejemplo generando las señales. Para los filtros FIR y ecualizador poco se podría haber mejorado, ya que la capacidad de procesado del dsPIC30F4011 no es modificable, se hubiese necesitado otro dsPIC.

CAPÍTULO 5: CONCLUSIONES Y LINEAS FUTURAS.

5.1. Conclusiones.

El resultado final de este PFC ha sido positivo. Se ha conseguido por una parte completar la placa de desarrollo UIB-PC104 mediante una PCB que añade el Si3000 necesario para el procesado digital de señal. Con esto, ya se posee el sistema completo para llevar a cabo todas las prácticas de audio propuestas para el PFC.

Estas prácticas se consiguieron realizar en lenguaje C, mediante la herramienta MPLAB, y el resultado fue satisfactorio, salvo una serie de restricciones debido al uso de un dsPIC de Microchip. Estas restricciones son debido a que los dsPIC son en realidad microcontroladores que poseen características de DSP añadidas, pero no es realmente un DSP.

Como primera restricción, esta el hecho de que las multiplicaciones en formato flotante son excesivamente lentas, quizás debido a que la arquitectura del motor DSP del dsPIC no esta diseñada para usar este formato, hecho que no ocurre en un DSP puro.

Siguiendo con las restricciones, y como consecuencia de la anterior, el formato que se puede usar para multiplicaciones en un ciclo de reloj, el formato fractional, es un formato que conlleva otras restricciones para algunas prácticas.

Otro dato importante del dsPIC30F4011, es la poca capacidad de memoria para datos no estáticos. Esto para un DSP que trabaja en parte con grandes buffers de muestras digitales es una desventaja o restricción importante, como se ha visto en las prácticas.

Otra restricción es el uso de lenguaje C para el diseño de las prácticas. Con el lenguaje C se dejan de usar los buffer circulares, que es una cualidad muy interesante de los DSP, y sobre todo con el lenguaje C se pierde precisión en las multiplicaciones, ya que en las operaciones tipo MAC, como las usuales de un filtro, el acumulador puede poseer 16 bits más que si se hace en lenguaje C. Esto conlleva perdida de calidad de los filtros.

Los puntos interesantes de este PFC han sido personalmente y en este orden, el aprendizaje de la programación en lenguaje C de un microcontrolador de Microchip, con la consecuente extensión del aprendizaje a todos los dispositivos de este fabricante, debido a su filosofía de diseño. En segundo lugar, la comprensión de un Motor DSP, y en su extensión la compresión de cómo actúa un DSP y por qué se usa, su arquitectura y particularidades respecto a un microcontrolador.

En tercer lugar, pero quizás más divertido y llamativo que los anteriores, el aprendizaje de creación de una PCB para el acople de un códec a la placa de desarrollo que se poseía. Este apartado conllevó toda la fase de diseño, desde la elección hasta la inserción del último componente, pasando por la compresión de un protocolo de comunicación digital como es el SPI.

Como conclusión, al acabar el PFC se podría comentar que el objetivo fue cubierto, pues se logró hacer un estudio del dsPIC30F4011 para poder usarlo en unas prácticas en lenguaje C, tal como se pretendía. Aunque estas prácticas serán siempre conseguidas respecto unas restricciones.

5.2. Líneas futuras.

Realmente en este apartado se deberían de comentar líneas para mejorar este PFC, pero creo que es más sencillo y quizás sensato pensar en qué componentes usar para poder conseguir mejor el objetivo de este PFC.

Como líneas de mejora de este propio PFC se podría añadir una memoria externa de datos. Esta memoria hubiese solucionado los problemas de la práctica 1 que no aprovechaba toda la capacidad de muestreo del Si3000, pero para poder añadir dicho componente, habría sido necesario otro bloque de comunicaciones SPI, y en este caso el dsPIC30F4011 solo posee uno y ya es usado por el Si3000.

Otra línea de mejora de este PFC podría haber sido usar el lenguaje ensamblador para las aplicaciones, o al menos parte de dichas aplicaciones. Esto habría llevado a rutinas más eficientes en tiempo de cálculo y precisión matemática.

Ahora se comentan diversas opciones para tener una capacidad de procesado mayor y unas características globales mejores para el desarrollo de aplicaciones para DSP.

Una opción sería usar otro códec, puesto que el Si3000 es bastante lento, si bien para prácticas más potentes como el filtrado o ecualizador, el Si3000 debía de disminuir su frecuencia para el correcto funcionamiento, pero para otras como la generación de ondas o la reverberación, hubiese sido muy interesante disponer de más frecuencia de muestreo.

Otra posible opción, es usar otro dsPIC. En este caso habría que valorar ciertas características como la inclusión del bloque de comunicación DCI que dejaría libre el bloque SPI para una posible memoria externa de datos, así como el uso de un dsPIC de la familia dsPIC33F que poseen más capacidad de cálculo. Aunque esto conllevaría no usar la placa de desarrollo UIB-PC104.

Y como gran opción y más drástica, está la elección de otra placa de desarrollo con dsPIC, por ejemplo la placa de desarrollo Explorer 16 de Microchip junto a la tarjeta Audio PICtail Plus, que incluye el WM8510 mono códec de Wolfson Microelectronic, que es un códec de 24 bits y trabaja a frecuencias de muestreo de hasta 48 KHz. Además tiene 4 Mbits de memoria Flash para usar en las aplicaciones. Es decir, consta de un códec potente, de memoria externa y de un dsPIC de la familia dsPIC33F.

Referencias:

[1] **UIB-PC104 User Interface Board. Manual del producto.**

[2] **iCM4011. ingenia Communication Module 4011.**

[3] **dsPIC30F4011/4012 Data Sheet.- dsPIC30F Family Referente Manual**

http://ww1.microchip.com/downloads/en/DeviceDoc/70135F.pdf.

http://ww1.microchip.com/downloads/en/DeviceDoc/70046E.pdf.

[4] **dsPIC30F/33F Programmer´s Reference Manual.**

http://ww1.microchip.com/downloads/en/DeviceDoc/70157C.pdf.

[5] **Datasheet Si3000.**

http://www.datasheetcatalog.org/datasheet/SiliconLaboratories/mXvtzqw.pdf

[6] **Información DSP.**

http://es.wikipedia.org/wiki/Procesador_digital_de_se%C3%B1al

[7] **Información FIR.**

http://es.wikipedia.org/wiki/FIR

[8] **Información IIR.**

http://es.wikipedia.org/wiki/IIR

[9] **Información Reverberación.**

http://es.wikipedia.org/wiki/Reverberaci%C3%B3n

[10] **Información Filtro digital.**

http://es.wikipedia.org/wiki/Filtro_digital

[11] **Información Microchip.**

http://www.ibars.com/assets/presentations/16-bit%20(sph).pdf

[12] José María Angulo Usategui et al. dsPIC. Diseño práctico de aplicaciones. Mcgraw-Hill, 2006.

[13] Enrik V. Sorensen y Jianping Chen. A digital signal processing laboratory using the TMS320C30. Prentice Hall, 1997.

[14] Emilio Soria Olivas et al. Tratamiento digital de señales: Problemas y ejercicios resueltos. Pearson Prentice Hall, 2003

[15] **Datasheet AK4564.**

http://www.datasheetcatalog.org/datasheets2/39/391282_1.pdf

[16] **Datasheet AK4631.**

http://www.datasheetcatalog.org/datasheets2/58/582585_1.pdf

[17] **Datasheet AK4640.**

http://www.datasheetcatalog.org/datasheet2/7/0yec1hsse4g3jxeeu6q9pyxjpjpy.pdf

[18] Datasheet AK4642.

http://www.datasheetarchive.com/pdf-datasheets/Datasheets-30/DSA-597447.html

[19] Información familias de 8 bits de Microchip.

http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=74

[20] Información familias de 16 bits de Microchip.

http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=2629&param=en533465

[21] Información de los PIC de 32 bits.

http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=2607#producttable

[22] Datasheet dsPICDEM 1.1 plus.

http://ww1.microchip.com/downloads/en/DeviceDoc/dsPICDEM%201.1%20Plus%20UG%2070099d.pdf

[23] Datasheet dsPICDEM 2.

http://ww1.microchip.com/downloads/en/DeviceDoc/dsPICDEM%202%20Users%20Guide%20DS-51558A.pdf

[24] Datasheet Audio PICTail.

http://ww1.microchip.com/downloads/en/DeviceDoc/70297A.pdf

[25] Datasheet AD73311.

http://www.analog.com/static/imported-files/data_sheets/AD73311.pdf

[26] Datasheet AD74111.

http://www.analog.com/static/imported-files/data_sheets/AD74111.pdf

[27] Datasheet AIC111.

http://www.datasheetcatalog.org/datasheet2/e/0zlgza8i45xz7iyhhxhuyprelfyy.pdf.

[28] Datasheet Tlv320aic12.

http://www.datasheetcatalog.org/datasheet/texasinstruments/tlv320aic12.pdf

[29] Datasheet Tlv320aic15.

http://www.datasheetcatalog.org/datasheet/texasinstruments/tlv320aic15.pdf