

CAPÍTULO 1: Introducción

A continuación se explican los objetivos que debe cumplir el proyecto. También se comentan las herramientas de las que se dispone para su realización, junto a una descripción de algunas de sus características principales. Por último se deducen algunas especificaciones que ha de cumplir el proyecto para el cumplimiento de los objetivos dados y se explica brevemente la estructura de esta memoria, indicando el contenido principal de sus capítulos.

1.1. OBJETIVOS

El objetivo principal es la realización de un sintetizador musical electrónico a través de un sistema digital predeterminado, el DSP (Digital Signal Processor - Procesador Digital de Señal) de Texas Instruments TMS320C30, que tenga un interfaz a través de un PC con el usuario. Para ello, el proyecto debe cumplir los siguientes objetivos:

- El usuario será el encargado de introducir los parámetros los sonidos a generar, que son nota y octava y el tiempo de duración de cada nota, además podrá elegir la intensidad que quiere para cada sonido, es decir, el volumen. Esto será introducido a través del PC con un interfaz gráfico. Además el sintetizador debe estar orientado a cualquier usuario, pudiendo el usuario no ser experto en síntesis musical.
- Debe reproducir las doce notas musicales en que se divide una octava.
- Ofrecer la posibilidad al usuario de generar notas con más calidad de sonido que un tono puro.
- Capacidad de generar sonidos polifónicos, es decir, que suenen varias notas musicales simultáneamente.

Para la realización del proyecto será necesario trabajar con el DSP TMS320C30 y con el PC. En el siguiente apartado se comentan brevemente las herramientas con las que se dispone en el laboratorio para trabajar con ambos sistemas.

1.2. HERRAMIENTAS DE TRABAJO.

Para la realización de este proyecto se dispone de un PC y del entorno de desarrollo para el DSP TMS320C30.

El DSP TMS320C30 de Texas Instruments (TI) es un procesador que trabaja en punto flotante, de 32 bits y con una frecuencia de reloj de 30 MHz. Como todos los DSPs, está especializado en tareas de procesamiento de la señal, siendo en estas tareas más eficiente que los procesadores de propósito general.

Para trabajar con el DSP se usará la herramienta de desarrollo EVM (Evaluation Module – Módulo de Evaluación). Es uno de los entornos de desarrollo más populares en esta gama de DSPs. Es una tarjeta de expansión ISA del PC, por lo que se eliminan los problemas de espacio y precio de otros sistemas de desarrollo.

Cuando al EVM se conecta a una entrada y una salida analógica, se convierte en un sistema de procesamiento de señal. La señal analógica muestreada puede transferirse al ordenador y éste puede pasar muestras al EVM para que las convierta en una señal analógica.

El PC contiene al EVM. La comunicación entre ambos se realizará a través de un protocolo descrito por Texas Instruments. El TMS320C30 realiza transferencias de 16 bits y el bus ISA es de 8 bits, por eso el EVM dispone de un hardware que se encarga de realizar estas transferencias de forma transparente al programador, al escribir o leer dos palabras consecutivas de 8 bits. El usuario es el encargado de introducir los distintos parámetros al sintetizador desde el PC, por lo que el PC será el “maestro” y el DSP el “esclavo”.

El PC con el que se va a realizar el proyecto tiene un microprocesador Pentium III con una frecuencia de reloj de 733 MHz, un disco duro de 12 GB y una memoria RAM de 256 MB. Con estas características hay potencia suficiente para realizar el interfaz y el control del sintetizador sin restricciones a la hora de elegir el software con el que realizar la parte correspondiente al PC.

1.3. ESPECIFICACIONES

Con los objetivos dados es posible tener una primera aproximación de las especificaciones que va a cumplir el sintetizador.

El sintetizador debe interpretar todas las notas y tiempos de las notas a generar. Esto permite predefinir los valores correspondientes a las notas y los tiempos. Esto no impide generar sonidos que no se correspondan con las notas y tiempos exigidos.

No se especifica el número de octavas, por lo que se realizarán todas las octavas posibles.

El volumen debe ser controlado por el usuario, por lo que se controlará de forma similar a lo que se hace en otras aplicaciones musicales.

El interfaz debe ser sencillo en su manejo y evitar que sea necesario tener conocimientos de síntesis musical para trabajar con él. Esto no impide que tenga opciones avanzadas para usuarios que sí tengan conocimiento de síntesis musical.

Para generar sonidos de mayor calidad, el sistema deberá implementar al menos un tipo de síntesis que genere timbres más ricos que un tono puro de la nota deseada.

El sintetizador debe ser polifónico, debe sonar más de una nota a la vez, es decir, debe tener más de un canal. No se especifica el número de canales, por lo que también se implementará la mayor cantidad posible.

Se ha comentado de forma muy general lo que debe hacer y lo que se puede hacer para cumplir los objetivos con el sintetizador. A lo largo de esta memoria se comentarán las opciones posibles y las opciones elegidas para el cumplimiento de todos los objetivos. También se explicarán las mejoras introducidas respecto a los objetivos, así como el motivo de su introducción.

A continuación se explica brevemente la estructura de esta memoria, indicando el contenido de cada capítulo:

- **CAPÍTULO 2: Descripción del entorno de desarrollo TMS320C30 EVM.** Breve explicación de las características principales del entorno de desarrollo EVM, centrándose sobre todo en el DSP TMS320C30.
- **CAPÍTULO 3: Fundamentos de la síntesis musical electrónica.** Se hace una pequeña introducción a la síntesis musical electrónica y se hace una breve explicación de los distintos tipos de síntesis musical.
- **CAPÍTULO 4: Implementación del sintetizador musical con el DSP TMS320C30.** Se explican las elecciones tomadas para la implementación del sintetizador, su funcionamiento y su implementación.
- **CAPÍTULO 5: Implementación del interfaz con el PC.** Se explica el funcionamiento de todo el interfaz para su correcto uso y su implementación.
- **CAPÍTULO 6: Conclusiones y líneas futuras.**
- Bibliografía.

CAPÍTULO 2: Descripción del entorno de desarrollo TMS320C30 EVM

El entorno de desarrollo TMS320C30 EVM presenta una arquitectura ideada para poder trabajar con el DSP TMS320C30 a través de una tarjeta de expansión ISA del PC. En este capítulo se hace una síntesis de los recursos y herramientas hardware y software del EVM y del DSP. Se han destacado los elementos más importantes y de los que ha habido que tener un mayor conocimiento de ellos para la realización del sintetizador. Toda la información necesaria sobre el EVM y el DSP TMS320C30 aparece en los manuales de Texas Instruments disponibles en el laboratorio o en su página web.

2.1. ELEMENTOS HARDWARE DEL TMS320C30 EVM

El EVM contiene los siguientes recursos hardware:

- Un DSP en punto flotante de 32 bits TMS320C30.
- Una SRAM de 16K palabras de 32 bits con accesos sin estados de espera.
- Adquisición de datos analógicos con calidad de voz a través del AIC (Analog Interface Circuit – Circuito de Interfaz Analógico) TLC32044.
- Jacks RCA estándar para la entrada y salida analógica.
- Puerto serie externo, que está libre y dispone de un conector externo para poder ser utilizado.
- Puerto de comunicaciones con el PC a través del bus ISA.
- Soporte de emulación embebida (dentro del propio PC) a través de un controlador de bus 74ACT8990 (TBC).

En la figura 2.1 aparece el diagrama de bloques del TMS320C30 EVM:

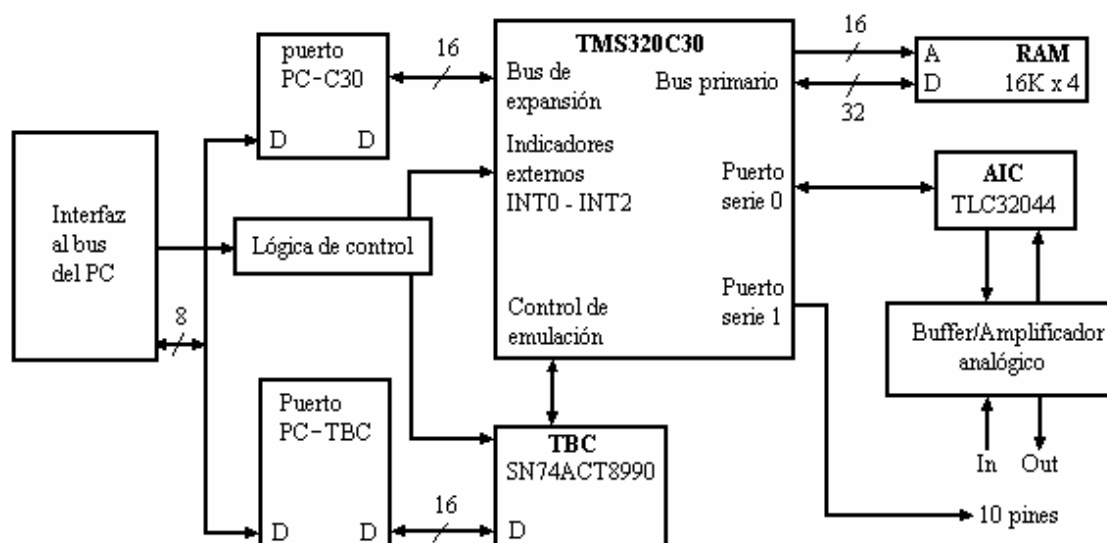


Figura 2.1. Diagrama de bloques del TMS320C30 EVM

A continuación se describen las características más importantes del DSP TMS320C30, que es el elemento que condiciona la arquitectura del EVM, y de los otros dispositivos del EVM.

2.1.1. EL DSP TMS320C30

El DSP TMS320C30 es un procesador de alto rendimiento, especialmente en tareas de procesamiento de la señal. Tiene tecnología CMOS, trabaja en punto flotante, es de 32 bits y tiene un reloj de 30 MHz.

Integra funciones de un sistema de control y de un sistema de cálculo intensivo en un mismo controlador. Esto permite rápidos y sencillos movimientos de datos a la vez que se puede realizar procesamiento numérico de alta velocidad.

La arquitectura usada permite realizar hasta 60 millones de instrucciones en punto flotante por segundo (60 MFLOPS), incorporando un alto grado de paralelismo que permite a los usuarios realizar hasta 11 operaciones en una sola instrucción.

En la figura 2.2 se muestra el diagrama de bloques del TMS320C30:

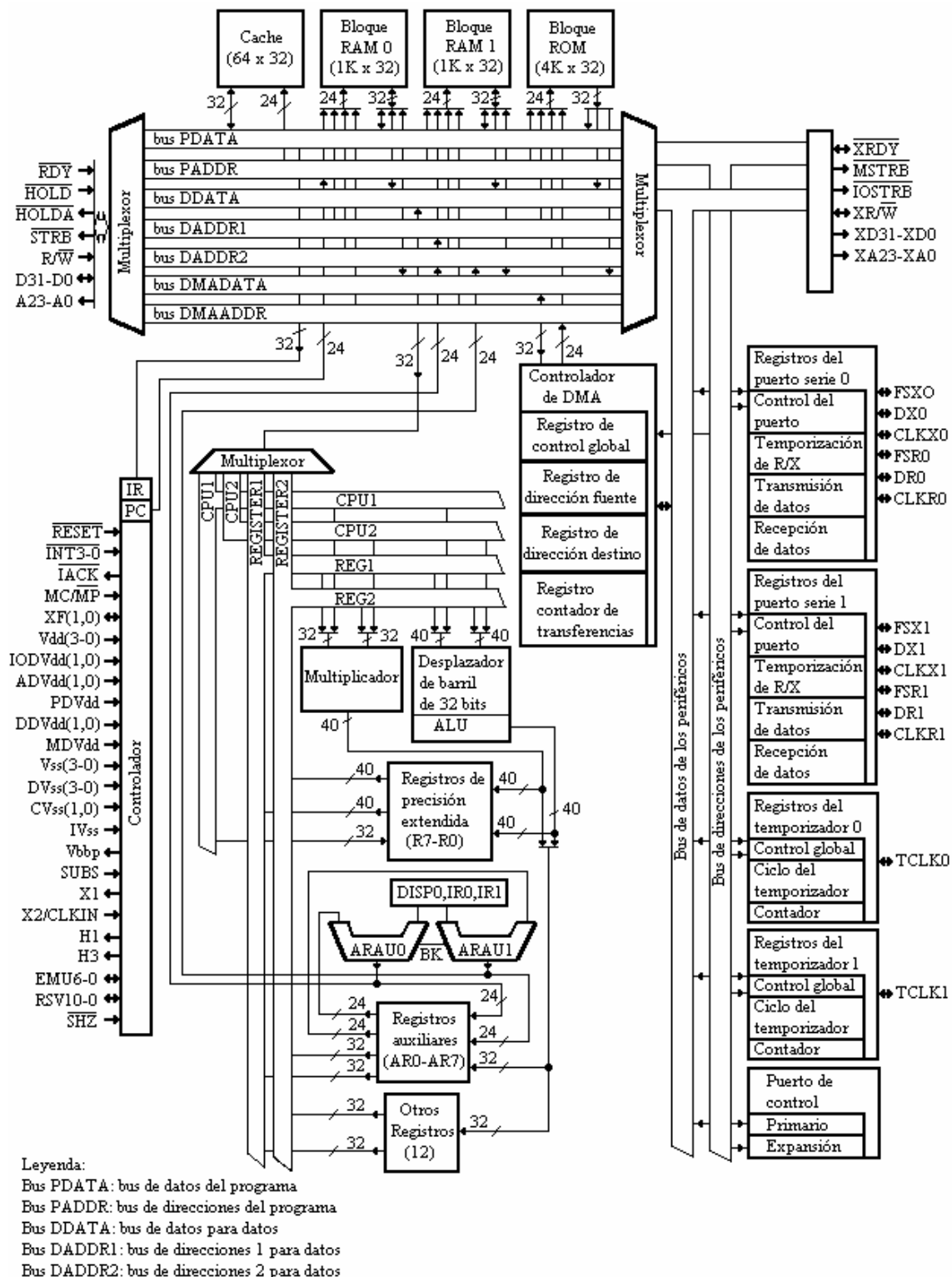


Figura 2.2. Diagrama de bloques del TMS320C30

Las características principales de este DSP, cuya arquitectura aparece en la figura 2.2, son las siguientes:

- Capacidad de trabajo con números en punto flotante, capaz de realizar hasta dos operaciones en punto flotante por ciclo de reloj.
- Conjunto de registros de propósito general.
- 4K palabras de ROM interna.
- Caché de programa de 64 palabras de 32 bits.
- Dos bloques de memoria RAM de 1K palabras cada una de dos accesos por ciclo.
- Dos buses de memoria. Al primario se conecta la memoria SRAM del EVM y el de expansión se conecta al interfaz con el PC.
- Un canal de DMA interno que puede tener acceso a cualquier zona de memoria.
- Dos temporizadores.
- Dos puertos serie. Uno de ellos está conectado con el exterior a través del EVM.
- Modos de bajo consumo.
- Hardware específico para la realización de buffer circular, bit invertido y bucles hardware.
- Arquitectura con pipeline.

En los siguientes apartados se comentan los elementos hardware del TMS320C30.

2.1.1.1. La unidad central de proceso (CPU)

La arquitectura de la CPU está basada en registros. Éstos son sus componentes:

- Multiplicador de números en punto flotante o enteros.
- Unidad Aritmético Lógica (ALU).
- Desplazador de barril de 32 bits.
- Buses internos.
- Dos Unidades Aritméticas de Registros Auxiliares (ARAUSs).
- Conjunto de registros primarios de la CPU.
- PC (contador de programa).
- IR (registro de instrucción).

El multiplicador realiza multiplicaciones con enteros de 24 bits y con números en punto flotante de 32 bits. Para mejorar el rendimiento se pueden usar instrucciones paralelas para realizar una multiplicación y una operación de la ALU en un solo ciclo de reloj.

El multiplicador tiene unas entradas de 32 bits y una salida de 40 bits si los números son en punto flotante y unas entradas de 24 bits y una salida de 32 bits si los números son enteros.

La ALU (Unidad Aritmético Lógica) realiza operaciones con enteros de 32 bits, en punto flotante de 40 bits y operaciones lógicas de 32 bits en un solo ciclo de reloj. Posee un desplazador de barril que se usa para desplazar hasta 32 bits a la izquierda o a la derecha el contenido de un registro en un solo ciclo de reloj.

Existen cuatro buses internos (CPU1, CPU2, REG1, REG2) por los que circulan los dos operandos de memoria y los dos de los registros. De esta forma, se permite la multiplicación en paralelo con la suma o resta sobre cuatro operandos enteros o en punto flotante en un solo ciclo de reloj.

Las dos Unidades Aritméticas de Registros Auxiliares (ARAU0 Y ARAU1) pueden generar dos direcciones de memoria en un solo ciclo de reloj. La ARAU opera en paralelo con el multiplicador y la ALU, soportando direccionamiento con desplazamiento, registro de índice (IR0 e IR1) simple, circular y bit invertido.

Existen dos registros especiales que no se pueden modificar normalmente. Aparecen en la tabla 2.1:

Registro	Nombre función	Descripción
PC	Contador de programa	Contiene la dirección de la siguiente instrucción a buscar. Puede ser modificado por las instrucciones de control de flujo.
IR	Registro de instrucción	Mantiene los códigos de instrucción durante la fase de decodificación de la instrucción. No es accesible por la CPU.

Tabla 2.1. Registros especiales de la CPU

Hay otros 28 registros primarios en la CPU. Éstos se describen en la tabla 2.2, junto con sus funciones especiales:

Registro	Nombre función	Descripción
R0...R7	Registros de precisión Extendida	Soportan operaciones de enteros de 32 bits y números en punto flotante de 40.
AR0...AR7	Registros auxiliares	Generan direcciones de 24 bits. Pueden ser accedidos por la CPU y por las dos ARAUSs, además de por la ALU y el multiplicador como contadores de bucles en direccionamiento indirecto o como registros de propósito general de 32 bits.
DP	Puntero a página de datos	Los 8 bits menos significativos son usados como puntero en modo de direccionamiento directo, apuntando a la página de datos de 64K a usar.
IR0, IR1	Registros índice	Son usados como registros auxiliares para la ARAU en direccionamiento indexado.
BK	Registro de tamaño de bloque	Lo usa la ARAU en direccionamiento circular para especificar el tamaño de un bloque de memoria.
SP	Puntero de pila	Contiene la dirección de la cima de la pila del sistema. Apunta al último elemento incorporado a la pila. Permite la creación de pilas y colas con registros auxiliares.
ST	Registro de estado	Contiene información acerca del estado de la CPU. Habilita las interrupciones de forma general.
IE	Registro de habilitación de interrupciones de la CPU y el DMA	Habilita o deshabilita las interrupciones de la CPU o del DMA.
IF	Registro de flags de interrupción	Indica si la correspondiente interrupción está activa o no.
IOF	Registro de flags de E/S	Controla la función de los pines externos XF0 y XF1.
RS	Dirección comienzo de repetición	Contiene la dirección de comienzo en el modo de repetición.
RE	Dirección fin de repetición	Contiene la dirección de final en el modo de repetición.
RC	Contador de repetición	Contiene el número de veces menos una que se va a repetir el bucle.

Tabla 2.2. Registros de la CPU

2.1.1.2. Formato de los datos

Los datos del TMS320C30 están organizados en tres tipos fundamentales: enteros, enteros sin signo y punto flotante.

Los enteros tienen dos formatos, entero corto (16 bits) y entero de precisión normal (32 bits). El rango es $[-2^a, 2^a - 1]$ donde 'a' es 15 para entero corto y 31 para entero de precisión normal.

Los enteros sin signo también presentan formato corto y de precisión normal. El rango es $[0, 2^a - 1]$ donde 'a' es 16 para el corto y 32 para el de precisión normal.

Existen tres formatos para números en punto flotante. El formato corto para operandos en punto flotante inmediatos (16 bits), precisión normal (32 bits) y precisión extendida (40 bits). Todos los formatos tienen un campo exponente, un campo bit de signo y un campo fracción (los dos últimos forman la mantisa). En la ecuación 2.1 aparece la ecuación general para calcular el valor de un número en punto flotante:

$$x = s\bar{s}.f_2 \cdot 2^e \quad (2.1)$$

donde 's' es el valor del bit de signo, ' \bar{s} ' es el inverso del valor del bit de signo, 'f' es el valor binario del campo fracción y 'e' es el equivalente decimal del campo exponente.

2.1.1.3. Organización de la memoria

El bus de direcciones del TMS320C30 es de 24 bits, por lo que puede direccionar hasta 16M palabras de 32 bits. Programa, datos y espacio de E/S se encuentran en el mismo espacio de direcciones, permitiendo compartir la ROM y la RAM. De esta forma se maximiza el uso de la memoria, asignando el programador cada sección de código donde desee.

El TMS320C30 posee memorias RAM, ROM y caché internas. Los dos bloques de RAM (RAM0 y RAM1 en la figura 2.2) tienen cada uno 1K palabras de 32 bits. La ROM es de 4K palabras de 32 bits. La memoria caché es de instrucciones y contiene 64 palabras de 32 bits.

Los buses de programa, datos y DMA permiten realizar búsquedas en paralelo de instrucciones de programa, leer o escribir datos y realizar transferencias de DMA en el mismo ciclo de reloj. Los buses externos de memoria pueden ser utilizados por el DMA para realizar escrituras o lecturas en memoria.

En el TMS320C30, los registros de los periféricos internos se encuentran mapeados en memoria, comenzando en la dirección 808000h. Los vectores de interrupción se encuentran en el espacio comprendido entre 00h – 3Fh.

El TMS320C30 tiene dos modos de operación, el modo microprocesador y el modo microcomputador. En el primero, los 4K de ROM interna no están disponibles en el mapa de memoria, mientras que en el segundo sí. En el modo microprocesador los vectores de interrupción están mapeados dentro de la memoria externa y en el modo microcomputador estos vectores se mapean en la ROM interna.

El EVM trabaja con el modo microprocesador, donde el espacio de direcciones que correspondería a la ROM se usa como memoria externa. En la figura 2.3 se muestra el mapa de memoria del TMS320C30:

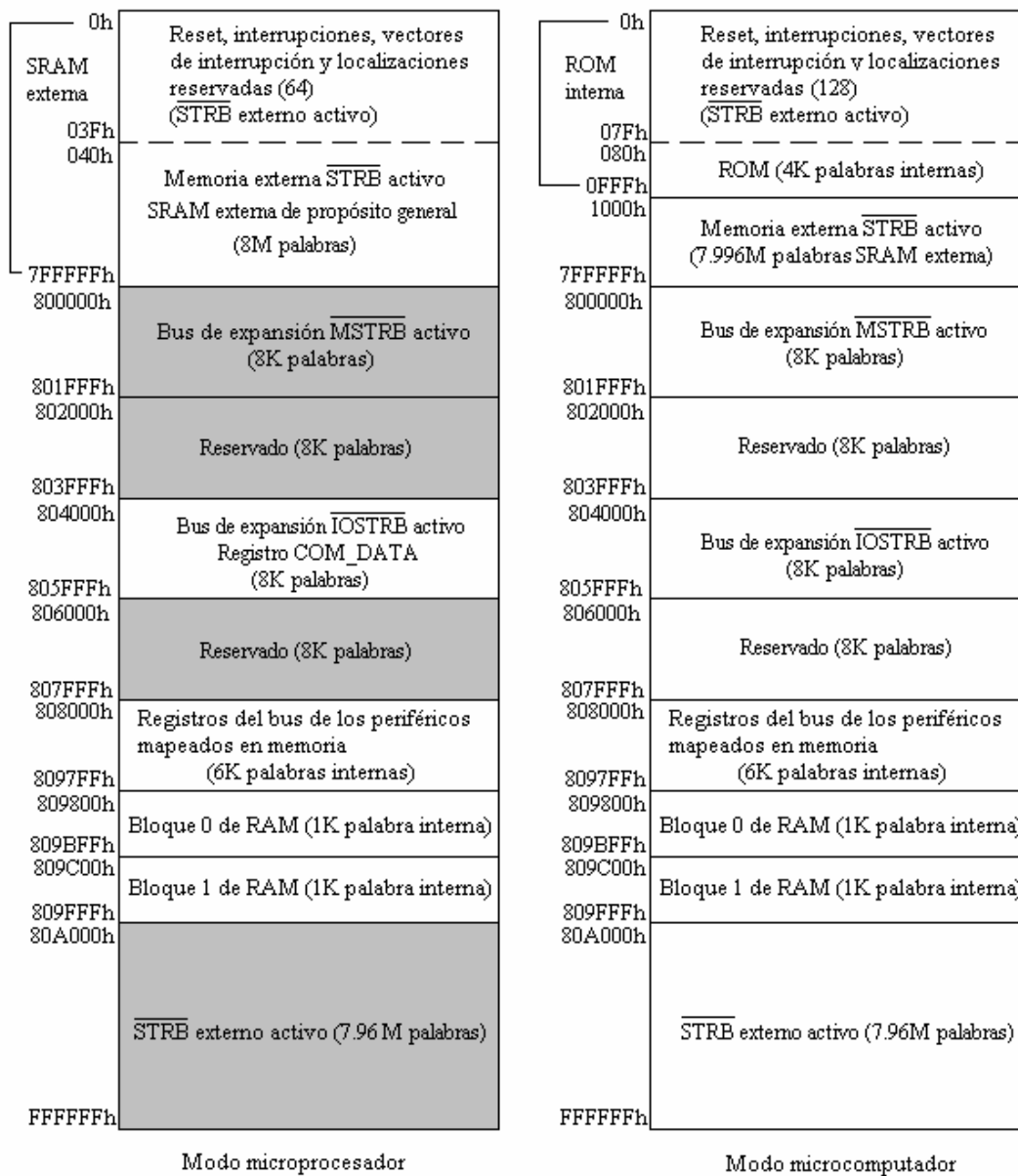


Figura 2.3. Mapa de memoria del TMS320C30

La parte no sombreada de la figura en el modo microprocesador corresponde al mapa útil del EVM. Las señales \overline{STRB} , \overline{IOSTRB} y \overline{MSTRB} se comentan en el apartado 2.1.1.5. y el registro COM_DATA es donde el EVM leerá o escribirá en las comunicaciones con el PC. Esta parte se tratará en el apartado 2.1.2.

2.1.1.4. Modos de direccionamiento

El TMS320C30 soporta 8 tipos de modos de direccionamiento. A continuación se enumeran y explican brevemente:

- Direccionamiento a registro: un registro de la CPU contiene el operando.
- Direccionamiento directo: la dirección del dato está formada por la concatenación de los 8 bits menos significativos del puntero DP (página de datos) con los 16 bits menos significativos de la palabra de instrucción.
- Direccionamiento indirecto: la dirección del operando en memoria se especifica a través del contenido de un registro auxiliar. Opcionalmente se pueden desplazar los registros de índice.
- Direccionamiento inmediato: el operando es un valor de 16 bits (direccionamiento inmediato corto) ó 24 bits (direccionamiento inmediato largo), contenidos en los 16 ó 24 bits menos significativos de la palabra de instrucción.
- Direccionamiento relativo al PC: es usado para saltos en las instrucciones. Suma la dirección de la palabra de instrucción a los 16 ó 24 bits menos significativos del operando fuente.
- Direccionamiento circular: este direccionamiento lo incorporan los DSPs porque hay algoritmos de procesamiento digital de la señal que requieren un buffer circular en memoria. Este buffer actúa como una ventana deslizante que contiene el dato más recientemente procesado. El nuevo dato sobrescribe al más antiguo, e incrementa el puntero de datos. Cuando el puntero accede al final del buffer, apunta otra vez al principio del buffer.
- Direccionamiento a bit invertido: también, como el anterior, es un direccionamiento que traen los DSPs para mejorar la eficiencia en algunos algoritmos de procesamiento digital de la señal, como por ejemplo FFT (Transformada rápida de Fourier). Los datos son accedidos por la CPU en un orden diferente, invirtiendo los bits de la dirección.

- Modos de repetición: son 2 instrucciones que realizan bucles hardware con 0 ciclos de sobrecarga, permitiendo reducir el tiempo de cómputo en muchas ocasiones.

2.1.1.5. Interfaz de memoria externa

El TMS320C30 tiene dos interfaces externos, el bus primario y el bus de expansión. Ambos disponen de un bus de datos de 32 bits y señales de control distintas. El bus primario tiene un bus de direcciones de 24 bits y está conectado a la memoria SRAM del EVM y el de expansión tiene un bus de direcciones de 13 bits, y es usado para el interfaz con el PC.

Cada interfaz posee configuraciones separadas controladas por registros de control de los interfaces mapeados en memoria; petición de señales de espera y asentimiento para poner las señales de memoria externa en estado de alta impedancia, para prevenir los accesos a bus externo; estados de espera programables controlados por software, hardware o por una mezcla de ambos y espacio de memoria unificado para datos, programa y accesos de E/S.

La señal \overline{STRB} que aparece en el mapa de memoria de la figura 2.3 es la señal de validación del acceso al bus primario y las señales \overline{IOSTRB} y \overline{MSTRB} son las señales de validación del acceso a los periféricos en el bus de expansión y validación del acceso al bus de expansión respectivamente.

2.1.1.6. Periféricos

Los periféricos son controlados a través de registros mapeados en memoria sobre un bus dedicado a tal efecto. Este bus de los periféricos tiene 32 bits para datos y 24 bits para direcciones.

El TMS320C30 tiene dos temporizadores y dos puertos series. Los temporizadores son contadores de eventos de propósito general de 32 bits, con dos modos de señalización y posibilidad de que el reloj sea interno o externo. Cada temporizador tiene un pin de E/S que puede ser usado como reloj de entrada al temporizador, como señal de salida controlada por el temporizador o como pin de E/S de propósito general.

Los puertos series son bidireccionales e independientes. Cada uno puede ser configurado para transferir 8, 16, 24 ó 32 bits por palabra. El reloj para cada puerto

puede ser generado internamente (mediante un divisor de frecuencia) o externamente. Los pines pueden ser configurados como pines de E/S de propósito general. Además, los puertos pueden ser configurados como temporizadores. En el EVM, el puerto serie 1 es el puerto con un conector externo que está libre y el puerto serie 0 es el que se usa para la adquisición de datos analógicos.

2.1.1.7. Acceso directo a memoria (DMA)

El controlador de DMA que incorpora el TMS320C30 en el chip puede leer o escribir en cualquier posición de memoria sin interferir con la operación de CPU. Sirve como interfaz para memorias y periféricos lentos, evitando que se reduzca el rendimiento del TMS320C30.

El controlador de DMA tiene su propio generador de direcciones y sus propios registros de direcciones fuente y destino y un registro contador de transferencias.

Una transferencia de DMA consiste en transferir una palabra o un bloque de palabras desde o hacia la memoria.

En el caso de que se pretenda acceder a los recursos simultáneamente por el DMA y la CPU, tiene la CPU mayor prioridad, generando el DMA un estado de espera.

2.1.1.8. Las interrupciones

El TMS320C30 soporta interrupciones hardware y software. Los vectores de interrupción contienen la dirección de la rutina de tratamiento de la interrupción correspondiente y se encuentran mapeados desde la dirección 0h a la 03Fh en el mapa de memoria. Algunas de estas direcciones son reservadas.

Las interrupciones hardware pueden ser internas o externas. Pueden interrumpir a la CPU o al DMA. Cuando la CPU responde a una interrupción hay un pin ($\overline{\text{IACK}}$) que puede ser usado para señalar un reconocimiento de interrupción al exterior.

Las interrupciones internas pueden ser generadas por el controlador de DMA, los dos temporizadores y los dos puertos series.

Las interrupciones externas son cinco. Una no enmascarable señalizada por la señal de $\overline{\text{RESET}}$ y otras cuatro a través de las señales ($\overline{\text{INT0}}$ - $\overline{\text{INT3}}$). Las interrupciones externas son activadas por nivel.

Las interrupciones software son internas y hay 28. Se generan con la ejecución de la instrucción ‘TRAP N’, donde ‘N’ es un número del 0 al 27.

Cuando dos interrupciones ocurren en el mismo ciclo de reloj o están esperando a ser servidas, la CPU atiende en primer lugar a la que tiene a su vector de interrupción en una posición más baja en la tabla de vectores de interrupción.

En la tabla 2.3 aparece la tabla de vectores de interrupción del TMS320C30:

Dirección	Nombre	Función
00h	RESET	Entrada de la señal de $\overline{\text{RESET}}$ externa.
01h	INT0	Interrupción de la señal externa $\overline{\text{INT0}}$.
02h	INT1	Interrupción de la señal externa $\overline{\text{INT1}}$.
03h	INT2	Interrupción de la señal externa $\overline{\text{INT2}}$.
04h	INT3	Interrupción de la señal externa $\overline{\text{INT3}}$.
05h	XINT0	Interrupción interna generada cuando el buffer de transmisión del puerto serie 0 está vacío.
06h	RINT0	Interrupción interna generada cuando el buffer de recepción del puerto serie 0 está lleno.
07h	XINT1	Interrupción interna generada cuando el buffer de transmisión del puerto serie 1 está vacío.
08h	RINT1	Interrupción interna generada cuando el buffer de recepción del puerto serie 1 está lleno.
09h	TINT0	Interrupción interna generada por el temporizador 0.
0Ah	TINT1	Interrupción interna generada por el temporizador 0.
0Bh	DINT	Interrupción interna generada por el controlador de DMA.
0Ch	Reservada	
· · ·	· · ·	
1Fh	Reservada	
20h	TRAP0	Interrupción interna generada por la instrucción TRAP 0.
· · ·	· · ·	
3B.h	TRAP27	Interrupción interna generada por la instrucción TRAP 27.
3Ch	TRAP28 (reservada)	
3Dh	TRAP29 (reservada)	
3Eh	TRAP30 (reservada)	
3Fh	TRAP31 (reservada)	

Tabla 2.3. Vectores de interrupción del DSP TMS320C30

2.1.2. INTERFAZ DEL TMS320C30 EVM CON EL PC

El interfaz entre el PC y el TMS320C30 es simple. Está basado en registros y proporciona un ancho de banda de 200 Kbytes por segundo. El interfaz debe convertir los accesos del PC de 8 bits a los accesos de 16 bits del TMS320C30.

El soporte de emulación embebida se produce gracias al controlador de bus (test bus controller o TBC) SN74ACT8990 y al puerto de emulación (bus de expansión) del TMS320C30. El código del TMS320C30 es cargado a través de su puerto de emulación, con lo que no es necesario tener memoria ROM o EPROM con un programa cargador para iniciar el sistema.

El TMS320C30 EVM es una tarjeta de expansión ISA que reside en el espacio de direcciones del PC, necesitando que éste tenga un total de 96 bytes libres distribuidos en tres páginas de 32 bytes. Cada página se encuentra aparte en 1 Kbyte.

El mapa de memoria del EVM que se puede observar desde el PC puede alojarse en diferentes posiciones de memoria para evitar conflictos, cambiando los dos primeros conmutadores de los cuatro que hay en la placa. De esta forma, según la posición de estos dos conmutadores, la posición de los primeros 32 bytes se muestra en la tabla 2.4:

Espacio de direcciones de E/S	Conmutadores	
	Conmutador 1	Conmutador 2
0x0240 – 0x025F	ON	ON
0x0280 – 0x029F	ON	OFF
0x0320 – 0x033F	OFF	ON
0x0340 – 0x035F	OFF	OFF

Tabla 2.4. Conmutadores de las direcciones de memoria en el PC del EVM

En la tabla 2.5 se muestran los registros del EVM mapeados en la memoria del PC, junto con el desplazamiento respecto a la primera dirección de la primera página:

Registro	Desplazamiento	Tamaño	Acceso
CONTROL0	0x0000	16 bits	R/W
CONTROL1	0x0002	16 bits	R/W
CONTROL2	0x0004	16 bits	R/W
CONTROL3	0x0006	16 bits	R/W
CONTROL4	0x0008	16 bits	R/W
CONTROL5	0x000A	16 bits	R/W
CONTROL6	0x000C	16 bits	R/W
CONTROL7	0x000E	16 bits	R/W
CONTROL8	0x0010	16 bits	R/W
CONTROL9	0x0012	16 bits	R/W
MINOR_CMD	0x0014	16 bits	R/W
RESERVADO0	0x0016 – 0x001E	-	-
STATUS0	0x0400	16 bits	R
RESERVADO	0x0402 – 0x041F	-	-
COM_CMD	0x0800	8 bits	R/W
COM_DATA	0x0808	16 bits	R/W
SOFT_RESET	0x0818	0	W

Tabla 2.5. Registros del EVM mapeados en la memoria del PC

Entre el PC y el EVM la comunicación es del tipo “maestro – esclavo”, siendo el PC el “maestro” y el EVM el “esclavo”. Cuando el PC escribe un comando, escribe 8 bits en el registro COM_CMD y cuando escribe un dato, escribe 16 bits en el registro COM_DATA. El TMS320C30 usa el registro COM_DATA para leer datos y comandos (16 y 8 bits respectivamente) y escribir datos de 16 bits. El TMS320C30 accede a este registro al escribir o leer en cualquier dirección comprendida entre 804000h y 805FFFh, que corresponde a una parte del bus de expansión.

Se pueden producir tres interrupciones en la comunicación entre el PC y el TMS320C30 EVM. Las interrupciones son ‘INT0’ (cuando el PC envía un comando al EVM), ‘INT1’ (cuando el PC envía un dato al EVM) e ‘INT2’ (se produce para indicar que el PC ha leído un dato escrito por el EVM).

El EVM no interrumpe cuando escribe un dato para el PC, por lo que éste debe controlar por ‘polling’ la recepción de datos enviados por el EVM.

Lo primero que hay que hacer para trabajar con el EVM es resetearlo. Esto se puede hacer desde el PC. Tanto el reset del EVM como la comunicación entre el PC y el EVM siguen un protocolo descrito en el manual del TMS320C30 EVM de Texas Instruments.

2.1.3. INTERFAZ DEL TMS320C30 EVM CON EL EXTERIOR

El EVM tiene un conversor analógico – digital (ADC) y otro digital – analógico (DAC) que permiten obtener muestras a partir de una señal de entrada o generar una señal analógica de salida a partir de las muestras generadas por el TMS320C30. Los dos conversores están integrados en un chip, el TLC32044, llamado AIC por TI y conectados al exterior por jacks RCA externos.

El AIC muestrea en teoría hasta 19200 muestras/segundo, aunque en la práctica se consigue muestrear hasta 17361 muestras/segundo. Esto no es suficiente para obtener calidad de audio de CD, donde la frecuencia de muestreo estándar es 44 KHz. La frecuencia de muestreo del AIC puede modificarse mediante unos registros accesibles por el programador.

Aunque los registros de transmisión y recepción del AIC son de 16 bits, sólo los 14 bits más significativos corresponden con la conversión de una muestra. Tras leer del ADC o antes de escribir en el DAC hay que hacer un desplazamiento hacia la izquierda de dos bits de la muestra.

El AIC es complejo de configurar para que funcione como se desee y TI en su manual incorpora indicaciones y ejemplos para su configuración.

2.2. SOFTWARE DEL TMS320C30 EVM

En este apartado sólo se van a nombrar las herramientas de programación del TMS320C30 y los pasos a seguir para la ejecución de programas con el EVM. En los manuales de Texas Instruments puede encontrarse la información necesaria para el uso del software del TMS320C30 EVM.

El TMS320C30 es programable en C cruzado y en ensamblador. Para ello posee un ‘*Debugger*’ que permite desarrollar, testear y refinar programas en ambos lenguajes. Para abrir el ‘*Debugger*’ hay que ejecutar el comando ‘evm30’.

El ‘*Debugger*’ posee varias ventanas de visualización y una serie de comandos para la ejecución y depuración de programas.

El TMS320C30 puede programarse usando un solo lenguaje o usando ambos a la vez, ya que desde C pueden usarse variables y rutinas implementadas en ensamblador y viceversa. Todas las rutinas (sea en C o ensamblador) de una aplicación deben ser compiladas y linkadas, permitiéndose numerosas opciones. El linkador agrupa todos los ficheros ensamblados en un solo fichero ejecutable.

Los ficheros ensamblados poseen formato COFF, que facilitan la programación modular usando las secciones. Las secciones son bloques de código o datos que ocupan un espacio contiguo de memoria. El programador puede crear secciones y decidir en qué secciones ubicar los distintos módulos de su programa.

Otros archivos importantes son los archivos de comandos (tienen extensión ‘.cmd’) que se le pueden pasar al linker. Estos archivos permiten especificar en qué posición del mapa de memoria ubicar las distintas secciones, además de contener el nombre de todos los ficheros ensamblados a linkar.

CAPÍTULO 3: Fundamentos de la síntesis musical electrónica

En este capítulo se explica la síntesis musical electrónica, se enumeran los tipos de síntesis más importantes con una pequeña descripción de cada una, destacando las ventajas e inconvenientes que presenta cada tipo para su implementación.

3.1. INTRODUCCIÓN A LA SÍNTESIS MUSICAL ELECTRÓNICA

Un sintetizador es un instrumento musical cuyo objetivo es la creación de nuevos sonidos, por tanto, no tiene un timbre ni unos sonidos que lo identifiquen como ocurre con el resto de instrumentos musicales, tales como una guitarra o una flauta. Un sintetizador permite crear sonidos propios, es decir, sonidos originales que aún no han sido reproducidos por otro instrumento ya conocido. Esa capacidad de generar sonidos también permite la imitación de sonidos de otros instrumentos musicales.

Un sistema de síntesis musical tiene dos tareas principales, la generación de ondas sonoras y su procesado. Los osciladores, es decir, las señales periódicas, son las fuentes primarias del sonido en un sistema de síntesis, ya sea digital o analógico. Todos los tipos de síntesis se basan en realizar operaciones y procesado a osciladores.

Actualmente los sintetizadores digitales construyen una señal oscilatoria a partir de un conjunto de muestras digitales generadas internamente.

Las muestras digitales pueden ser generadas por un microprocesador y transformarse en señales analógicas (que pueden activar un altavoz) mediante un convertidor digital – analógico (CDA). Para la realización de este sintetizador, el microprocesador utilizado es el DSP TMS320C30 de acuerdo con las especificaciones exigidas.

A la hora de generar señales oscilatorias hay que tener en cuenta las tres principales características en la implementación de osciladores: eficiencia, complejidad de cálculo y precisión, siendo la eficiencia la más importante porque es la que permite tener una buena relación calidad – precio.

La eficiencia y la complejidad de cálculo van relacionadas, ya que un algoritmo más sencillo en su codificación se ejecutará más rápido, permitiendo realizar más tareas en el mismo tiempo que otro más complejo, aunque a veces la sencillez puede conllevar la necesidad de incluir otros algoritmos que mejoren la funcionalidad o el control de flujo. La precisión a la hora de generar una nota no tiene que ser exacta ya que no importa percibir una nota de 220 Hz a 223 ó 218 Hz. Sin embargo, es más importante en el tiempo de duración, ya que los valores menores de la escala temporal de un pentagrama son valores pequeños que pueden confundirse si la precisión no es la adecuada. Los procesadores actuales son suficientemente precisos para definir los valores de tiempo deseados.

3.1.1. SEÑALES DIGITALES

Ya se ha comentado que los osciladores actuales son digitales, sin embargo las señales que oímos son analógicas. Por ello hay que conocer la relación existente entre una señal analógica y una digital.

Una señal analógica viene definida por la variable continua tiempo (t) y una digital por una variable discreta de números enteros no negativos (n) que determina el índice de la muestra digital.

Si tenemos la señal senoidal siguiente, que se muestra en la ecuación 3.1:

$$y(t) = \cos(\omega \cdot t + p) \quad (3.1)$$

donde ' ω ' es la frecuencia angular en radianes por segundo, ' t ' el tiempo en segundos y ' p ' el desfase en segundos y un dispositivo con una frecuencia de muestreo ' f_m ' en muestras por segundo, siendo ' n ' el índice de la muestra (0, 1, 2...), para $n = 0$ tenemos $t = 0$ y para $n = f_m$ tenemos $t = 1$. La señal digital será como aparece en la ecuación 3.2:

$$y[n] = \cos(\omega \cdot (n/f_m) + p) \quad (3.2)$$

Hay que tener en cuenta el teorema de Nyquist, con lo cual no se podrá reproducir una señal cuya frecuencia máxima sea superior a $f_m/2$. Una señal tendrá un timbre más rico, es decir, sonará mejor, mientras más armónicos posea.

3.1.2. CONCEPTOS MUSICALES

Para la realización de un sintetizador musical hay que considerar una serie de conceptos musicales válidos para cualquier instrumento musical.

En un pentagrama convencional aparece la siguiente información: nota, octava y tiempo de duración de cada nota. La nota y la octava van relacionadas y determinan la frecuencia de una onda sonora.

Una octava musical se divide en 12 notas musicales con unos valores de frecuencia determinados. Cuando se sube una octava se dobla la frecuencia y cuando se baja se divide la frecuencia a la mitad. Las notas musicales son: Do, Do#, Re, Re#, Mi, Fa, Fa#, Sol, Sol#, La, La#, Si o Do, Reb, Re, Mi, Fa, Solb, Sol, Lab, La, Sib, Si. La nota de menor frecuencia es 'Do' y 'Si' la de mayor frecuencia.

La 'b' se lee como bemol y el símbolo '#' como sostenido, y lo llevan aquellas notas que son frecuencias intermedias entre Do, Re, Mi, Fa, Sol, La o Si. 'La#' se lee como La sostenido y 'Lab' como La bemol. Los sostenidos se aplican a los tonos de menor frecuencia y los bemoles a los de mayor frecuencia, siendo 'Do#' y 'Reb' la misma nota musical.

Según la ISA (Internacional Standard Association), la frecuencia de una nota 'La' en la cuarta octava de un piano debe ser 440 Hz. Si se enumeran las notas musicales de 0 a 11 (Do = 0, Do# = 1, Re = 2, Re# = 3, Mi = 4, Fa = 5, Fa# = 6, Sol = 7, Sol# = 8, La = 9, La# = 10, Si = 11), la frecuencia de cualquier nota musical puede calcularse mediante la ecuación 3.3:

$$\text{Frecuencia} = \text{FREC_LA_4} * 2^{((\text{nota} - 9) / 12) + (\text{octava} - 4)} \quad (3.3)$$

donde 'nota' es el número correspondiente a cada nota, 'octava' es el número de octava en el que suene la nota y 'FREC_LA_4' es 440 Hz. Para frecuencias más altas los sonidos son más agudos y más graves para frecuencias más bajas.

La nomenclatura que se usa para el tiempo de duración es la siguiente: redonda, blanca, negra, corchea, semicorchea, fusa y semifusa. La unidad es la negra, siendo la blanca 2 negras y la redonda 4 negras (2 blancas). Los tiempos nombrados a continuación de la negra son la mitad del anterior, siendo la corchea $1/2$ de una negra, la semicorchea $1/4$ de una negra (y $1/2$ de una corchea) y así hasta la semifusa (es $1/16$ de una negra).

Otra característica fundamental de un sonido es la intensidad con la que suena. En una señal sonora esa intensidad viene dada por su amplitud. También es importante el número de notas que es posible hacer sonar simultáneamente, lo que se conoce con el número de canales.

Ya se han descrito los parámetros fundamentales de cualquier sonido, aunque no son los únicos. Como ya se comentó en el apartado 3.1, un sintetizador tiene dos tareas principales, la generación de ondas sonoras y su procesamiento, que como mínimo deben crear señales controlables en frecuencia, para generar las notas de las octavas posibles, y en amplitud.

3.2. TIPOS DE SÍNTESIS

A la hora de hablar de síntesis se puede distinguir entre síntesis tradicional y síntesis no tradicional. La síntesis tradicional engloba a las técnicas usadas en los sintetizadores más vendidos y utilizados en el mercado. Dentro de la síntesis tradicional, aparecen los métodos que usan modulaciones, que son métodos que generan timbres más ricos. A continuación se van a explicar brevemente los distintos tipos de síntesis más usados.

3.2.1. MÉTODOS DE SÍNTESIS TRADICIONAL

- Síntesis aditiva: consiste en la suma algebraica de varias ondas senoidales, construyendo una señal a partir de sus armónicos, ya que mediante las series de Fourier, cualquier señal periódica puede ser descrita así. En este método se define la frecuencia fundamental de la señal y el número de cada armónico a sumar, junto con su amplitud y fase. La frecuencia de cada armónico será la frecuencia fundamental multiplicada por el número del armónico. De esta forma quedan definidos los senos

o cosenos. Es un método sencillo, pero muy tedioso a la hora de sintetizar sonidos con timbres más ricos, ya que éstos tienen muchas componentes frecuenciales y hay que realizar numerosas sumas.

- **Síntesis sustractiva:** es el concepto inverso a la síntesis aditiva. Se parte de una señal rica en armónicos y se le aplican filtros hasta obtener el espectro deseado. Los filtros pueden realizarse discretizando un filtro analógico conocido, a partir de sus ecuaciones, o realizando un diseño directo mediante diagramas de polos y ceros sobre el plano z . Con este método se pueden generar sonidos con un espectro rico en componentes frecuenciales y que sólo contenga aquellas componentes deseadas. Sin embargo, su implementación implica crear señales con un espectro rico y realizar filtros precisos.

Con las modulaciones se obtienen sonidos con timbres más ricos sin necesidad de usar muchos osciladores. Los tipos de síntesis que aparecen a continuación usan la modulación de algún parámetro.

- **Síntesis AM:** la amplitud de una onda modula la amplitud de otra, es decir, se trata de la modulación AM entre dos señales. En este tipo de síntesis, la señal portadora tiene la frecuencia de la nota musical y la señal moduladora una frecuencia múltiplo de la portadora. La onda portadora es una señal senoidal y la moduladora puede ser cualquier señal periódica, siendo mejor el sonido para moduladoras con un espectro más rico. Es un método que sólo necesita dos osciladores.
- **Síntesis FM:** es una modulación FM, donde la frecuencia de la señal portadora se modula en función de la amplitud de la moduladora. En esta modulación también la frecuencia de la nota musical es la frecuencia de la portadora y la frecuencia de la moduladora es múltiplo de la frecuencia de la portadora. Se generan infinitos armónicos alrededor de la frecuencia fundamental, aunque a partir de un número, determinado por las funciones de Bessel, la amplitud de los armónicos es despreciable. Este tipo de síntesis, al contrario que la AM, permite que un oscilador se module a sí mismo. Esto permite crear timbres bastantes complejos con varios osciladores. Se pueden realizar modulaciones en cascada combinando modulaciones AM y FM. Esta síntesis es sencilla y permite crear timbres ricos en armónicos.

- Síntesis PM: es una modulación PM, pero no se va a explicar, ya que es muy parecida a la síntesis FM desde el punto de vista matemático y los espectros resultantes son prácticamente iguales.
- Envolventes: una envolvente es una señal que da forma a otra señal a lo largo del tiempo. Las envolventes suelen usarse para modular la amplitud de los sonidos, aunque pueden aplicarse a cualquier parámetro. Su duración suele ser mayor que las señales audibles. Se usa para simular aquellos sonidos que se van dejando de oír pausadamente, por ejemplo, la tecla de un piano.
- LFOs (Low Frequency Oscillators – Osciladores de Baja Frecuencia): son generadores de señal que sólo difieren de los osciladores normales en la posibilidad de generar frecuencias por debajo de 15 Hz (menor que el umbral de audición). También se usan para modular cualquier parámetro de una señal. Se usan más que como base para generar sonidos, para producir distintos efectos, al igual que las envolventes.

3.2.2. MÉTODOS DE SÍNTESIS NO TRADICIONAL

Los métodos de síntesis no tradicional suelen ser variantes de los tradicionales en muchos casos y muchos de ellos están pensados para la emulación de sonidos pertenecientes a grupos de instrumentos ya existentes, como sonidos de instrumentos de cuerda o de percusión. Existen además métodos y sintetizadores cuyo objetivo es la emulación de un instrumento ya existente. A continuación se describen brevemente los más usados:

- Sincronización de onda: consiste en dotar al oscilador de una entrada que al ser activada reinicia la fase de la onda a cero. Esto produce saltos bruscos en la señal del oscilador generándose armónicos a frecuencias altas. Solo hay que implementar un oscilador.
- Síntesis vectorial: es un método versátil y sencillo que mezcla varias formas de onda de forma lineal. Un ejemplo sería sumar cuatro osciladores, cada uno con una

amplitud diferente, donde todas las amplitudes estarían en función de algún parámetro.

- **Secuenciación de onda:** este método se basa en la utilización de muchas formas de onda diferentes, controlando la mezcla final mediante LFOs o envolventes. Su inconveniente es que hay que implementar muchos osciladores distintos y además el LFO o la envolvente.
- **Tablas de ondas:** se basa en la reproducción de sonidos previamente grabados, ya sean de instrumentos reales, ruidos o cualquier tipo de sonido. Consiste en muestrear una misma nota en varias octavas (para ello se multiplica o divide por dos las distintas frecuencias de muestreo) y una vez obtenida esa nota, a partir de ella, se obtienen las demás notas. Se trata de un método complejo.
- **Modelado físico:** la síntesis mediante modelado físico es un método basado en modelos de comportamiento de instrumentos reales, ya sea para emularlos o para crear nuevos timbres. El método directo de describir un instrumento real mediante sus ecuaciones diferenciales es muy complejo y no se usa. Lo que se hace es una buena aproximación al modelado físico directo mediante las guías de ondas. Una guía de ondas es un registro de desplazamiento de muestras que simboliza la trayectoria de una onda. Para emular un instrumento se crean tantas guías de ondas como ondas sonoras produzca el instrumento. Es un método bastante complejo, ya que previamente hay que estudiar el comportamiento del instrumento y los instrumentos tienen muchos parámetros que los caracterizan.

3.3. TIPOS DE OSCILADORES

Como ha podido verse en los distintos tipos de síntesis, la síntesis musical electrónica se basa en el uso de osciladores.

Los osciladores más usados en música electrónica son las señales senoidales, las señales triangulares, las señales diente de sierra y las señales cuadradas o rectangulares.

A la hora de implementar un oscilador con un sistema digital puede ser importante, para una determinada aplicación, elegir un método que reduzca el tiempo de cómputo, es decir, su complejidad.

Una opción para usar osciladores es llamar a una función matemática de una librería ya implementada. Pero esas funciones suelen emplear cálculos muy complejos y consumir mucho tiempo de procesador. Para solucionar ese problema pueden usarse resonadores o tablas de ondas precalculadas.

Un resonador electrónico es un circuito amplificador sintonizado a una determinada frecuencia y que, bajo ciertas circunstancias, puede provocar una autooscilación y convertirse en un generador de señal senoidal cuya frecuencia es la frecuencia de sintonía del amplificador. Un resonador digital puede obtenerse mediante la discretización de las ecuaciones diferenciales de un resonador electrónico o a partir de diagramas de polos y ceros en el plano z . El inconveniente es que sólo puede generar ondas senoidales.

Las tablas de ondas precalculadas permiten implementar osciladores de cualquier señal periódica y requieren poco tiempo de procesador. Para ello se almacenan en una tabla en memoria muestras de la señal a reproducir durante un periodo. En el procedimiento para ir leyendo muestras de forma correcta hay que usar interpolación. La calidad del oscilador dependerá de la resolución de la tabla (mejor mientras más muestras tenga) y del tipo de interpolación que se use. Este método es sencillo pero necesita de gran cantidad de memoria.

Las señales senoidales tienen su función principal como portadoras en técnicas que usen modulaciones y el resto de osciladores son más útiles a la hora de crear señales sonoras que tengan un espectro más rico.

Como se ha mostrado en este capítulo, lo más importante que un diseñador debe tener en cuenta a la hora de implementar un sintetizador musical son el tipo de síntesis y las señales a usar. En un sintetizador es posible que aparezcan más de un tipo de síntesis y de señales e incluso combinaciones tanto de métodos de síntesis como de señales.

CAPÍTULO 4: Implementación del sintetizador musical con el DSP TMS320C30

El sintetizador musical ha sido implementado a través del entorno de desarrollo del DSP TMS320C30 EVM. El EVM es el encargado de crear y procesar ondas sonoras que cumplen los objetivos de este proyecto. El DSP TMS320C30 crea y procesa las muestras digitales que se convierten en ondas analógicas sonoras a través del AIC del EVM.

4.1. DECISIONES DE IMPLEMENTACIÓN Y DE DISEÑO

La primera elección a tomar fue si empezar a trabajar con el EVM o con el PC. Se empezó con el EVM para conocer las características del sintetizador, ya que éste iba a depender de los recursos y posibilidades del hardware del TMS320C30, y así saber lo que ofrecer al usuario en el interfaz con el PC.

Otra elección que se tomó antes de empezar a realizar el sintetizador era si se quería emular a un instrumento real o crear sonidos propios. Se eligió la segunda opción porque la música electrónica comercial tiene esa tendencia, porque para una buena emulación se necesita un procesador más rápido, el TMS320C30 sólo tiene calidad de voz, y porque al crear sonidos nuevos el usuario puede tener mayor sensación de originalidad en las composiciones que realice.

Tras el conocimiento de las distintas posibilidades para la realización de síntesis musical electrónica y de los recursos hardware y software del DSP TMS320C30 se detallan las elecciones adoptadas para la realización del sintetizador musical, acorde con los objetivos y las especificaciones dados, y se describe el funcionamiento general del sintetizador y de cada uno de sus elementos.

4.1.1. LENGUAJE DE PROGRAMACIÓN DEL DSP TMS320C30

La programación del DSP TMS320C30 se realizará a través del entorno de desarrollo EVM que ofrece Texas Instruments. El EVM, como la mayoría de entornos de desarrollo que hay en el mercado, puede programarse en ensamblador o C cruzado. Ambos lenguajes disponen de compilador, linkador y depurador.

El programador tiene tres posibilidades: programar sólo en ensamblador, sólo en C o usando los dos, ya que se posibilita la utilización de ambos lenguajes en una misma aplicación debido a que desde C se pueden usar variables y rutinas definidas en ensamblador y viceversa.

Se ha elegido usar ambos lenguajes, siendo el C cruzado mucho más usado y el que realiza la mayor parte de la implementación del sintetizador.

El lenguaje ensamblador se ha usado para la escritura o lectura de los distintos puertos y registros internos del DSP accesibles para el programador, ya que se realiza de forma más sencilla que en C cruzado. Así se ha usado para la escritura de las muestras que generan las ondas sonoras tras la conversión digital – analógica y para la inicialización de los registros del TMS320C30.

La elección del C cruzado ha sido principalmente por traer ya implementadas funciones matemáticas complejas de programar en su librería <math.h>. Como ya se ha explicado anteriormente, las funciones senoidales son básicas en la síntesis musical y C ya trae esas funciones de forma que el programador solo ha de llamarlas. Además también incorpora otras funciones como división y potencia en punto flotante que no vienen definidas en ensamblador y que son útiles para trabajar con señales y también son complicadas de programar. Todo esto ahorra mucho tiempo de trabajo que puede ser empleado en otros aspectos de la síntesis musical. Además C tiene ventajas de un lenguaje de alto nivel como tener sentencias y estructuras de lenguaje para el control de flujo de un programa o la transparencia para el programador en el uso de ciertos recursos del DSP, a la vez que permite programar a bajo nivel, siendo sencillo el manejo de las interrupciones.

Al elegir las funciones de C para la implementación de ondas senoidales y demás osciladores, se han descartado el resto de alternativas que también implementan estos tipos de señales. Esas alternativas tienen la ventaja de ahorrar tiempo de procesador, sin embargo las funciones seno y coseno y el resto de funciones útiles para la creación de las señales usadas son funciones implementadas en ensamblador por Texas Instruments en librerías específicas para el TMS320C30, por lo que esas funciones son eficientes para trabajar con el TMS320C30.

Al no tener conocimiento de la existencia de otros proyectos realizados en esta escuela con el DSP TMS320C30 en lenguaje C, se detallarán ciertos aspectos sobre la programación del sintetizador, sobre todo aquellos que más fácilmente pueden inducir a cometer errores, aunque se disponga de la suficiente documentación, o que no estén suficientemente claros en la documentación de Texas Instruments; esto se explicará en el apartado 4.3. De esta forma, además de dar las explicaciones oportunas propias del proyecto realizado, se pretende aclarar posibles dudas que puedan surgirles a aquellas personas que se enfrenten en un futuro a la programación de este DSP en el mismo lenguaje.

4.1.2. TIPOS DE SÍNTESIS Y DE SEÑALES

Se ha optado por implementar dos tipos de síntesis y cinco tipos de señales. De esta forma se ha conseguido, como se explicará posteriormente, que el usuario tenga bastantes sonidos distintos para una misma nota de una misma octava. Además se pueden realizar combinaciones entre ellas produciéndose una mayor diversidad de sonidos y mejorando el timbre de cada sonido.

Los tipos de síntesis que se han usado pertenecen a la síntesis tradicional. Son síntesis AM y síntesis FM. La síntesis aditiva, en la que se controla la frecuencia y la amplitud de todos los armónicos de forma sencilla, ha sido descartada porque para la realización de sonidos más ricos, que necesitan mayor número de componentes frecuenciales, tiene que sumar numerosos osciladores. La síntesis sustractiva es más compleja de implementar y no aporta timbres mejores que las modulaciones y los otros tipos de síntesis son para realizar efectos sobre sonidos ya creados o para emular instrumentos reales, y la opción elegida es la de crear sonidos propios. Sin embargo, las

modulaciones sí pueden crear esos sonidos más ricos con sólo dos osciladores, variando la amplitud (AM) o la frecuencia (FM) de una onda en función de la amplitud de la otra.

Los tipos de señales son coseno, triangular, diente de sierra, cuadrada y tren de pulsos cuadrados. Las cinco señales son usadas en los dos tipos de síntesis.

El coseno se usa como portadora y moduladora en las dos modulaciones. Las otras señales son usadas como moduladoras ya que presentan mayor número de armónicos.

Todas las señales son osciladores controlables en frecuencia y amplitud y por tanto capaces de generar un sonido de una intensidad, nota y octava determinadas. Por ello, también se ofrece la posibilidad de usar estas señales como ondas sonoras sin ningún tipo de modulación para sintetizar sonidos. Cuando sólo aparece un sonido, lo normal es elegir la síntesis de mayor calidad, pero cuando se mezclan distintos sonidos puede interesar que sean lo más distintos posibles o que haya una gran diversidad. Aunque evidentemente el sonido es de menor calidad que en una modulación, se ofrece esta posibilidad para aumentar la diversidad de sonidos.

4.1.3. NÚMERO DE CANALES Y OCTAVAS

Ha sido necesario buscar un compromiso entre el número de canales y el número de octavas, ya que para implementar más canales había que disminuir la cantidad de octavas. Esto es debido a que si hay más octavas las frecuencias de las notas va aumentando y por tanto la frecuencia de muestreo del AIC del EVM debe ser mayor para cumplir el teorema de Nyquist y por otro lado, si aumentamos la frecuencia de muestreo disminuimos el tiempo para procesar una muestra hasta que llegue la siguiente, con lo que se disminuye el número de canales, ya que la forma de obtener canales diferentes es procesar de distintas formas una misma muestra.

Se han implementado cuatro octavas de las ocho posibles, ocho son las que tiene un piano. Las octavas van desde la primera hasta la cuarta. Algunas notas podían obtenerse en la sexta octava, pero no con todos los métodos de síntesis realizados, sino con los de menor calidad. Si se hubiesen implementado menos octavas apenas se conseguirían sonidos agudos y así se ha conseguido obtener sonidos graves y agudos. De haber obtenido una octava más, que es posible, sólo tendría un canal y la gama de sonidos iba a variar poco.

Se han implementado dos canales. Para poder implementar dos canales hay que bajar la frecuencia de muestreo desde 17361 Hz hasta 8013 Hz. Si se hubiesen obtenido más canales no se podrían haber implementado las cuatro octavas, al tenerse que disminuir la frecuencia de muestreo aún más, y se ha preferido tener canales de más calidad aunque en menor número. De esta forma se obtiene un sonido polifónico, que cumple los objetivos, con canales que ofrecen un sonido de calidad.

También se podían conseguir más canales disminuyendo los tipos de síntesis o señales que se han usado, así se disminuye el tiempo de proceso de cada muestra, pero también perdiendo calidad en cada canal. Además con esta opción se pierde variedad de sonidos y se reducen las posibilidades de mezcla de sonidos distintos.

4.2. FUNCIONAMIENTO DEL SINTETIZADOR

El hardware necesario para implementar el sintetizador es un sistema digital predeterminado y ya montado, que no podía ser elegido, por lo que para la realización de este instrumento musical únicamente se ha desarrollado software y no ha habido que elegir un procesador ni realizar ningún montaje sobre alguna placa, ya que todo el hardware (el EVM) necesario estaba preparado para trabajar con él en el laboratorio. En la figura 4.1 se muestra el diagrama funcional del sintetizador:

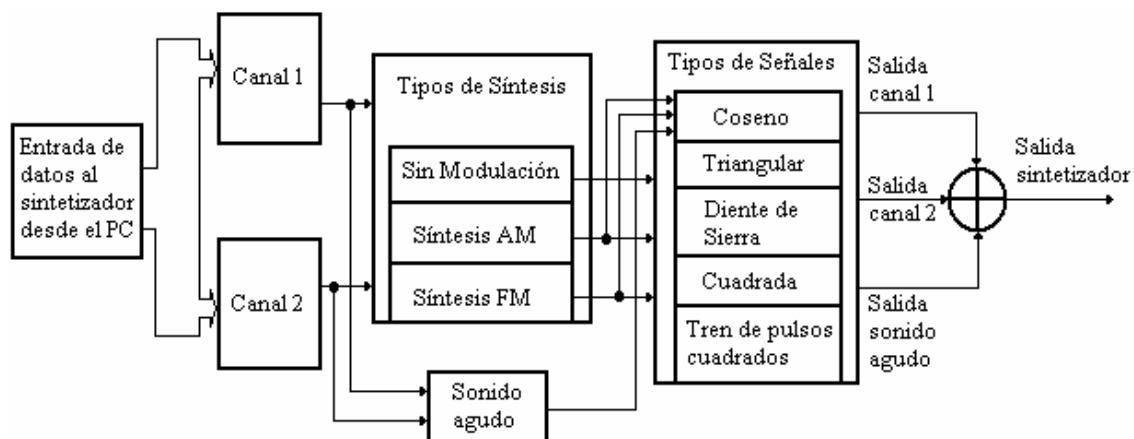


Figura 4.1. Diagrama funcional del sintetizador

El sintetizador tiene dos canales que pueden activarse o desactivarse por separado. Esto permite que emita sonidos polifónicos o no polifónicos, según deseo del usuario. Además, el sintetizador es capaz de realizar síntesis AM, síntesis FM y de generar los

cinco tipos de señales que aparecen en el diagrama funcional de la figura 4.1 (coseno, triangular, diente de sierra, cuadrada, tren de pulsos cuadrados). Los dos canales funcionan de la misma forma y hacen lo mismo. Cada canal puede realizar de manera independiente las siguientes funciones:

- Activar el efecto llamado ‘sonido agudo’, que es un coseno de 900 Hz que cuando está activado suena sólo mientras suene algún canal, ya que es un efecto y no una nota musical. Por tanto, si los canales están desactivados o no hay ninguna nota musical sonando, este efecto no se producirá aunque se encuentre activado. Este sonido no se corresponde con ninguna nota, ya que su intención es producir un sonido agudo, ya que no se pudo obtener más octavas y controlar su nota sería introducir otro canal, que como se ha explicado en el apartado 4.3 no es posible. Cuando este efecto está activado se pueden escuchar tres sonidos distintos.
- Elegir el tipo de síntesis, que también permite la opción de que suene cualquier tipo de oscilador por sí solo, es decir, sin modularse con otra señal, es lo que en el diagrama de bloques aparece como ‘Sin Modulación’.
- Elegir el tipo de señal moduladora. Si se elige alguna modulación siempre se usará el coseno como portadora y cualquier señal podrá elegirse como moduladora, incluso el propio coseno. Si se elige la opción de ‘Sin Modulación’ la señal elegida, sin modularse y sin mezclarse con otra señal, será la encargada de generar sonido.
- Generar cualquier nota musical desde la primera hasta la cuarta octava. Es posible que los dos canales tengan la misma nota en la misma octava. Cuando sucede esto, el sonido no es polifónico, excepto si está activado el efecto sonido agudo, en cuyo caso sí se aprecian sonidos distintos simultáneamente.
- Controlar el volumen de cada nota. Es posible que cada canal tenga un volumen distinto para permitir que una nota musical se aprecie más que otra.
- Elegir el tiempo de cada nota musical. Los valores de tiempo son todos los valores que aparecen en un pentagrama convencional. Por tanto los tiempos van desde la

redonda hasta la semifusa (redonda / 64). Para la redonda se ha elegido un tiempo de 6 segundos, para que la semifusa sea audible.

- Controlar todos los parámetros de las señales que se elijan para realizar síntesis AM o síntesis FM, tanto de la portadora como de la moduladora. Esto permite al usuario producir mucha diversidad de sonidos para la misma nota musical en una misma octava, otorgándole a cada nota muchos matices diferentes.

La salida del sintetizador siempre es la suma de los dos canales. Cuando un canal está desactivado su salida será cero. El efecto de sonido agudo se suma cuando está activado y hay algún canal sonando, si está activado pero ningún canal suena, tampoco emitirá su sonido.

Todos los datos y controles que necesita el sintetizador para poder realizar sus funciones y generar sonidos los envía el usuario a través del interfaz Windows realizado con el PC.

En los siguientes apartados se explican los bloques ‘Tipos de Síntesis’ y ‘Tipos de Señales’.

4.2.1. TIPOS DE SÍNTESIS

Cada canal puede elegir algún tipo de síntesis o que una señal por sí sola sea la encargada de generar sonido, es decir, que esa señal no sea modulada en ninguno de los dos tipos de síntesis. En esta última opción, la señal también genera un sonido de una nota determinada y se pueden controlar las mismas opciones que en las modulaciones, excepto los parámetros propios de una modulación. La opción de elegir una señal sin modulación, aunque el sonido tiene peor calidad, se ha introducido para aumentar la diversidad de sonidos de cada canal, que es una de las principales características de este sintetizador. Al haber cinco señales, sólo por permitir esta opción se crean cinco timbres distintos por nota musical. Cuando no hay modulación, el volumen de la nota es determinado por la amplitud de la señal elegida.

La síntesis FM, además de generar timbres más ricos que la síntesis AM, se oye mejor en tonos más graves y tiene menos problemas de ruido en las octavas más agudas.

4.2.1.1. Síntesis AM

En esta síntesis se han usado dos señales periódicas y se ha hecho una modulación AM. Para ello se usan las cinco señales implementadas, ejerciendo el coseno siempre como portadora. Como señal moduladora puede elegirse entre cualquiera de las cinco, incluyendo el propio coseno.

La ecuación de una señal AM es la que aparece en la ecuación 4.1:

$$Y_{AM}(t) = (C + x(t)) \cdot A \cdot \cos(\omega_c \cdot t) ; \text{ donde } x(t) = M \cdot \cos(\omega_m \cdot t) \quad (4.1)$$

‘ $x(t)$ ’ es la onda moduladora, que tiene una frecuencia angular ‘ ω_m ’ y una amplitud ‘ M ’. ‘ C ’ es la componente continua de la señal y $A \cdot \cos(\omega_c \cdot t)$ es la señal portadora, donde ‘ ω_c ’ es su frecuencia angular y ‘ A ’ su amplitud.

La frecuencia de la señal portadora es la frecuencia fundamental del tono musical que se desea escuchar, mientras que la frecuencia de la moduladora es un producto entre la frecuencia fundamental de ese tono y un factor multiplicador entero positivo y mayor que uno (2,3,...). Se han elegido las frecuencias de esta manera porque es la forma más habitual de trabajar con síntesis AM. La frecuencia angular de la onda moduladora se calcula con la ecuación 4.2:

$$\omega_m = \omega_c \cdot k; \quad k = 2,3,4,\dots \quad (4.2)$$

Esta modulación se ha implementado de forma que es posible elegir el tipo de señal moduladora así como los valores de los siguientes parámetros: componente continua de la señal, factor multiplicador de la frecuencia de la moduladora, amplitud de la señal moduladora y amplitud de la señal portadora. El efecto que produce la variación de los parámetros es el siguiente:

- Factor multiplicador de la frecuencia de la moduladora: es el factor ‘ k ’ de la ecuación 4.2. Su aumento desde 2 hacia números naturales mayores, hace que el sonido sea cada vez más agudo, ya que se aumenta la frecuencia de la señal moduladora, que aporta más armónicos al sonido que la portadora.

- Componente continua de la señal: su variación progresiva desde cero hacia valores positivos produce que haya un sonido de fondo cada vez más audible.
- Amplitud de la señal moduladora: su variación progresiva desde cero hacia valores negativos o positivos produce que el sonido sea más agudo. Al aumentar su amplitud, la señal moduladora va ganando dominio en el sonido respecto a la señal portadora y como la moduladora tiene mayor frecuencia, el sonido se hace más agudo.
- Amplitud de la señal portadora: su aumento desde cero hacia valores positivos aumenta el volumen del sonido. Se usa para controlar el volumen.

4.2.1.2. Síntesis FM

Esta síntesis genera timbres mejores que los de la síntesis AM y se nota menos el ruido que introducen las señales moduladoras en las frecuencias más altas. Surgió cuando aparecieron los osciladores digitales.

En esta síntesis se han usado dos señales periódicas y se ha hecho una modulación FM. También se usan las cinco señales implementadas, ejerciendo el coseno siempre como portadora y como señal moduladora puede elegirse entre cualquiera de las cinco, incluyendo el propio coseno.

La ecuación 4.3 muestra la ecuación de una señal FM:

$$Y_{FM}(t) = A \cdot \cos(\omega_c \cdot t + x(t)) ; \text{ donde } x(t) = I \cdot \cos(\omega_m \cdot t) \quad (4.3)$$

‘x(t)’ es la onda moduladora, que tiene una frecuencia angular ‘ ω_m ’ y una amplitud ‘I’. $A \cdot \cos(\omega_c \cdot t)$ es la señal portadora, donde ‘ ω_c ’ es su frecuencia angular y ‘A’ su amplitud.

‘I’ también es denominado índice de modulación y puede calcularse mediante la ecuación 4.4:

$$I = \frac{f_m}{f_c} \quad (4.4)$$

donde ‘ Δf ’ es el incremento de frecuencia de la portadora, también denominado desviación de frecuencia de la portadora, y ‘ f_m ’ la frecuencia de la moduladora. Por ejemplo, para variar la frecuencia de una señal portadora de 300 Hz entre 290 Hz y 310 Hz, basta con hacer $\Delta f = 10$. En la síntesis FM se generan infinitos armónicos por encima y debajo de la frecuencia de la portadora, teniendo esos armónicos las frecuencias que aparecen en la ecuación 4.5:

$$f_i = f_c \pm (i \cdot f_m); \quad i = 0 \dots (I + 2) \quad (4.5)$$

aunque se generan infinitos armónicos, a partir de ‘ $I + 2$ ’ esos armónicos pueden despreciarse. Su atenuación puede conocerse por las funciones de Bessel.

En esta modulación también la frecuencia de la señal portadora es la frecuencia fundamental del tono musical que se desea escuchar, mientras que la frecuencia de la moduladora es un producto entre la frecuencia fundamental de ese tono y un factor multiplicador entero positivo y mayor que uno (2,3,...). La frecuencia angular de la onda moduladora se calcula con la ecuación 4.6:

$$\omega_m = \omega_c \cdot k; \quad k = 2,3,4,\dots \quad (4.6)$$

Esta modulación se ha implementado de forma que es posible que el usuario pueda elegir el tipo de señal moduladora así como los valores de los siguientes parámetros que controlan la modulación: desviación de frecuencia de la señal portadora, factor multiplicador de la frecuencia de la moduladora (estos dos parámetros influyen en la amplitud de la señal moduladora, es decir, en el índice de modulación) y amplitud de la señal portadora.

Todos estos parámetros, incluidos los parámetros de la síntesis AM, son controlados desde el interfaz implementado con el PC. No se permite al usuario dar cualquier valor a estos parámetros. Los valores permitidos se especificarán y explicarán en el capítulo 5, que está dedicado al interfaz con el PC. El efecto que produce la variación de los parámetros en la síntesis FM es el siguiente:

- Desviación de frecuencia de la señal portadora: su variación progresiva en aumento, siempre con números naturales, hace que el sonido sea cada vez más agudo. Este

parámetro afecta proporcionalmente a la amplitud de la señal moduladora y los efectos que produce son similares a aumentar la amplitud de la señal moduladora en la síntesis AM. En sonidos graves, con la variación de este parámetro se aprecia un efecto de eco.

- Factor multiplicador de la frecuencia de la moduladora: es el factor 'k' de la ecuación 4.6. Su aumento desde 2 hacia números naturales mayores, hace que el sonido sea cada vez más agudo, pero distinto a cuando se varía el parámetro anterior. Al igual que en la síntesis AM, es debido a que se aumenta la frecuencia de la señal moduladora.
- Amplitud de la señal portadora: su aumento desde cero hacia valores positivos aumenta el volumen del sonido. Se usa para controlar el volumen, igual que en la síntesis AM.

4.2.2. TIPOS DE SEÑALES

El objetivo principal de implementar cinco señales distintas es crear timbres distintos, al tener éstas diferentes espectros. En todas las señales se controla la frecuencia y la amplitud por los distintos tipos de síntesis, que le dan los valores adecuados para obtener el volumen, la nota musical y la octava que pida el usuario. Aunque los valores de frecuencia y amplitud sean modificados, la forma de las señales no serán cambiadas. A continuación se explicará la forma matemática de obtener estas señales y se hablará de sus características sonoras, donde quedará claro que la síntesis FM es la mejor. Además se mostrarán gráficas para ver la forma de esas señales.

4.2.2.1. La señal coseno

El coseno se ha implementado, sobre todo, para usarlo como señal portadora en las síntesis AM y FM. Su timbre, cuando suena por sí sola, es decir, sin modular, es muy pobre, no pudiéndose apreciar para las notas más graves, aunque para sonidos agudos sí se oye, aunque con un timbre de poca calidad. Esto mismo ocurre cuando se usa como señal moduladora en la síntesis AM. Sin embargo en la síntesis FM genera timbres de calidad en todas las octavas.

Es la señal usada para realizar el efecto ‘sonido agudo’, ya que al tener un espectro con menor ancho de banda que el resto de señales, es la señal con la que más octavas se pueden conseguir, ya sea como onda sonora o como onda moduladora.

Para generar el coseno se ha recurrido a la función ‘cos’ de la librería ‘math.h’. A esta función se le pasa como parámetro un ángulo en radianes y te devuelve el coseno. La señal coseno, además del argumento, se define por su amplitud. El sintetizador usa dos rutinas (las rutinas que implementan los bloques funcionales del sintetizador y su funcionamiento se explicarán en el apartado 4.3) para implementar cosenos. La primera rutina es la que se usa para generar casi todos los cosenos y la segunda rutina se usa para implementar el coseno como señal portadora en la síntesis FM. En la ecuación 4.7 se muestra la señal que genera la primera rutina y en la ecuación 4.8 la que genera la segunda rutina. Lo único que cambia son los argumentos de los cosenos:

$$Y(t) = A \cdot \cos(\omega_c \cdot t) \quad (4.7)$$

$$Y(t) = A \cdot \cos(\omega_c \cdot t + \text{mod}) \quad (4.8)$$

‘mod’ es la onda moduladora FM. Con la ecuación 4.8 puede obtenerse la 4.7 haciendo ‘mod’ igual a cero, pero la más usada en todo el programa del sintetizador es la ecuación 4.7, por lo que se ha preferido usar la ecuación 4.7, para tener que enviar un parámetro menos en la llamada a la rutina.

4.2.2.2. La señal triangular

La señal triangular tiene un timbre más pobre que el resto de señales que se explicarán en este apartado. Aunque tiene un timbre mejor que el coseno, al igual que éste, no se oye bien cuando suena por sí sola o en la síntesis AM en los tonos más graves y sí tiene un timbre de calidad en la síntesis FM en todas las octavas.

La realización de esta señal y de las tres restantes se ha hecho calculando su valor en el primer periodo respecto al origen de las coordenadas cartesianas, obteniéndose el valor correspondiente en el resto de instantes de tiempo, desplazando hasta el origen un periodo completo.

En la figura 4.2 se muestra la gráfica de esta señal en el primer periodo:

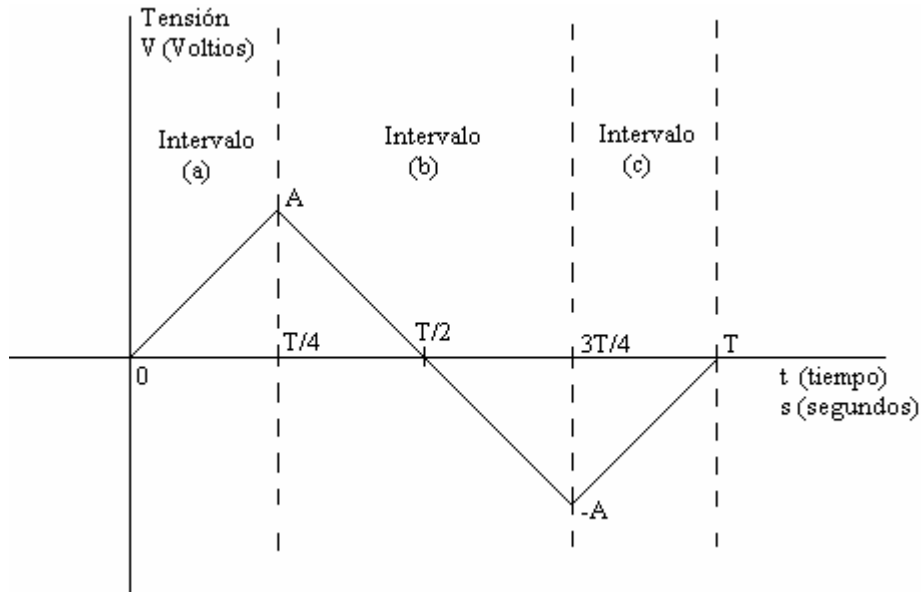


Figura 4.2. Gráfica de la señal triangular en un periodo

Esta señal es completamente simétrica, es decir, los dos triángulos invertidos que forman un periodo, son iguales. Como puede apreciarse en la figura 4.2, para calcular los valores del primer periodo basta con deducir las ecuaciones de las tres rectas que lo forman.

Para generar esta señal se han empleado las ecuaciones 4.9, 4.10 y 4.11, correspondiéndose cada una con un intervalo en que se ha dividido al primer periodo de la gráfica de la figura 4.2:

$$Y(t) = \text{pte} * t ; \quad \text{intervalo (a)} \quad (4.9)$$

$$Y(t) = -\text{pte} * t + b ; \quad \text{intervalo (b)} \quad (4.10)$$

$$Y(t) = \text{pte} * t - b ; \quad \text{intervalo (c)} \quad (4.11)$$

donde 'pte' es la pendiente de la recta, 't' el tiempo y 'b' el punto de corte con el eje de ordenadas. En las tres ecuaciones anteriores la pendiente depende de la amplitud y se calcula con la ecuación 4.12:

$$\text{pte} = (A * 4) / T \quad (4.12)$$

donde 'A' es la amplitud de la señal y 'T' es el periodo de la señal, que pueden cambiarse sin que se afecte a la simetría de la señal. Para obtener 'b' las operaciones son distintas. Así el valor de 'b' se calcula con las ecuaciones 4.13 y 4.14:

$$b = (\text{pte} * T) / 2; \quad \text{intervalo (b)} \quad (4.13)$$

$$b = \text{pte} * T; \quad \text{intervalo (c)} \quad (4.14)$$

4.2.2.3. La señal diente de sierra

Esta señal aporta más armónicos que las dos anteriores. Esta señal tiene buenos timbres en todas las octavas, aunque aparece ruido en la cuarta octava cuando aparece sin modular y en la síntesis AM, siendo el ruido mayor en la síntesis AM. Este ruido no ocurría con las señales anteriores debido a que, al poseer cambios más bruscos, tiene armónicos a frecuencias más altas. Además, en la primera octava con síntesis FM el sonido tiene una intensidad baja. En la figura 4.3 se muestra su gráfica en un periodo:

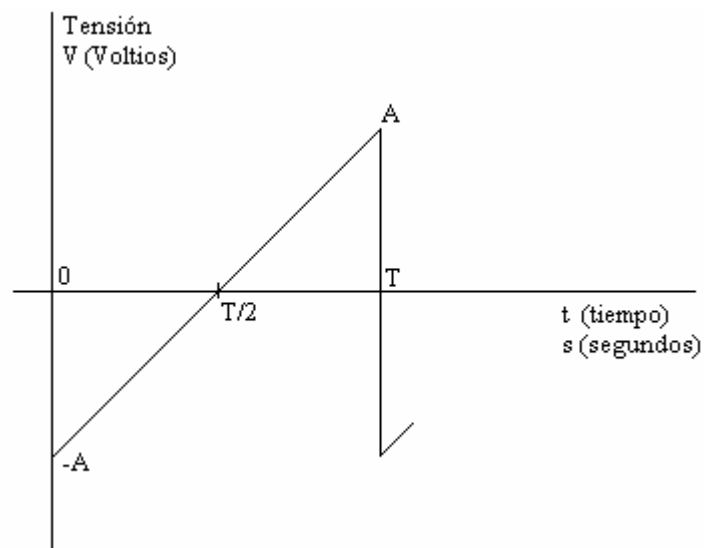


Figura 4.3. Gráfica de la señal diente de sierra

Siempre la señal corta al eje de abscisas en la mitad de su periodo y sus múltiplos. La ecuación de la señal es la ecuación 4.15:

$$Y(t) = \text{pte} * t - b; \quad (4.15)$$

donde 'pte' es la pendiente de la recta, 't' es el tiempo y 'b' el punto de corte con el eje de ordenadas. En la ecuación 4.16 se muestra como se calcula la pendiente:

$$\text{pte} = (A * 2) / T \quad (4.16)$$

donde 'A' es la amplitud que se desea para la señal y 'T' es el periodo de la señal para la frecuencia deseada. En la ecuación 4.17 se observará como calcular 'b':

$$b = (\text{pte} * T) / 2 \quad (4.17)$$

4.2.2.4. La señal cuadrada

La siguiente señal es cuadrada y no rectangular, es decir, la señal se cruza con el eje de abscisas en la mitad de su periodo y sus múltiplos. Esta señal suena bien excepto en la cuarta octava, tanto sin modular como en las dos modulaciones, siendo el ruido menor en la modulación FM. Esto es debido a que también tiene armónicos a frecuencias más altas que las señales triangular y coseno. En la figura 4.4. se muestra su gráfica en el primer periodo:

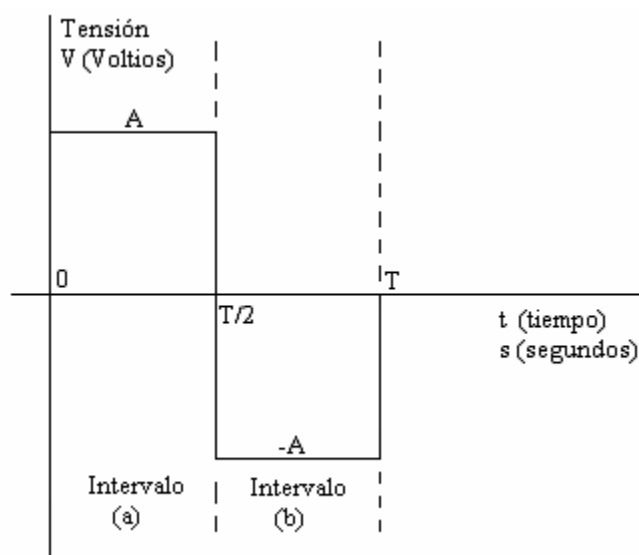


Figura 4.4. Gráfica de la señal cuadrada

La ecuación de la señal es muy simple, teniendo el valor de la amplitud en el semiperiodo positivo y la amplitud negativa en el semiperiodo negativo. Puede verse en las ecuaciones 4.18 y 4.19:

$$Y(t) = A; \quad \text{intervalo (a)} \quad (4.18)$$

$$Y(t) = -A; \quad \text{intervalo (b)} \quad (4.19)$$

donde 'A' es la amplitud.

4.2.2.5. La señal tren de pulsos cuadrados

Con este nombre se denomina a una señal que es como la anterior, excepto que en un semiperiodo sustituye el valor negativo de la amplitud por cero. Tiene un timbre muy parecido a la señal anterior en la mayoría de las notas musicales, a veces la diferencia es casi inapreciable, pero se ha decidido implementarla porque sí hay notas donde se aprecia bien la diferencia, con lo que se aumenta la variedad de sonidos, que es una de las características principales de este sintetizador. Lo comentado sobre el sonido de la señal anterior es válido para ésta, con la diferencia de que esta señal presenta un ruido menor en las modulaciones. En la figura 4.5 se muestra su gráfica en un periodo:

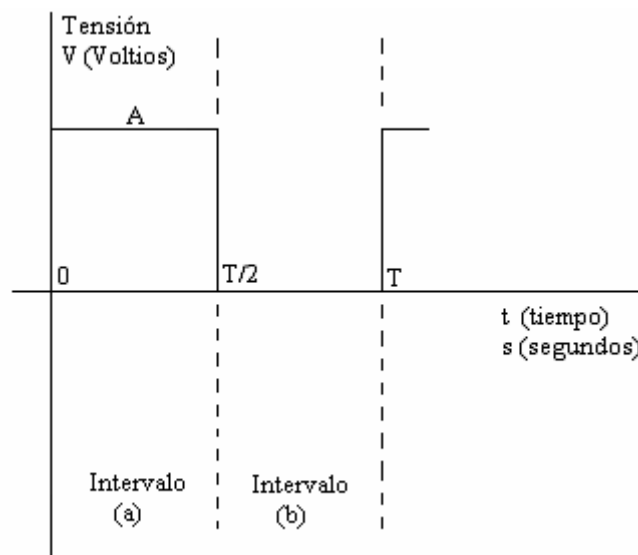


Figura 4.5. Gráfica de la señal tren de pulsos cuadrados

La ecuación de la señal es muy similar a la de la señal del apartado anterior, teniendo el valor de la amplitud en un semiperiodo y cero en el otro. Se muestra en las ecuaciones 4.20 y 4.21:

$$Y(t) = A; \quad \text{intervalo (a)} \quad (4.20)$$

$$Y(t) = 0; \quad \text{intervalo (b)} \quad (4.21)$$

donde 'A' es la amplitud.

En las ecuaciones de las cinco señales que se han explicado ha aparecido la variable continua tiempo. El TMS320C30 trabaja en tiempo discreto, con lo que hay que trabajar con muestras y no con segundos. Para escribir las ecuaciones que den las muestras que tras la conversión digital – analógica generen las señales deseadas, hay que realizar la sustitución en las ecuaciones donde aparecen la variable de tiempo continuo 't' como muestra la ecuación 4.22:

$$t = n / f_m \quad (4.22)$$

donde 'n' es el número de la muestra y 'f_m' es la frecuencia de muestreo del conversor digital – analógico del EVM. Se ha muestreado para la realización del sintetizador a 8013 muestras / segundo. Por tanto, en todas las señales el periodo es 1 / 8013 segundos.

4.3. SOFTWARE DESARROLLADO PARA LA IMPLEMENTACIÓN DEL SINTETIZADOR

La programación desarrollada para implementar el sintetizador con el DSP TMS320C30 usa lenguaje ensamblador y C cruzado.

En ambos lenguajes se debe trabajar siempre en punto flotante a la hora de realizar cualquier tipo de operaciones, aunque los datos puedan realizarse con operaciones de enteros. La lectura y escritura de muestras en los dos conversores, tanto el digital – analógico y el analógico – digital, debe hacerse en formato de enteros. Por ello se ha hecho una conversión de flotante a entero para sacar muestras al exterior. También se ha trabajado con enteros en C con variables que se han usado para las sentencias de control

en estructuras de selección o en bucles. Son los dos únicos casos donde se ha trabajado en formato entero.

En el apartado 4.2 se han expuesto los distintos bloques funcionales, en este apartado se explicarán los códigos que implementan los bloques funcionales del apartado 4.2. Primero se irán mostrando bloques más genéricos para después irlos desglosando hasta explicar el funcionamiento de todas las rutinas del programa realizado. Así el programa principal tiene el diagrama funcional de la figura 4.6:

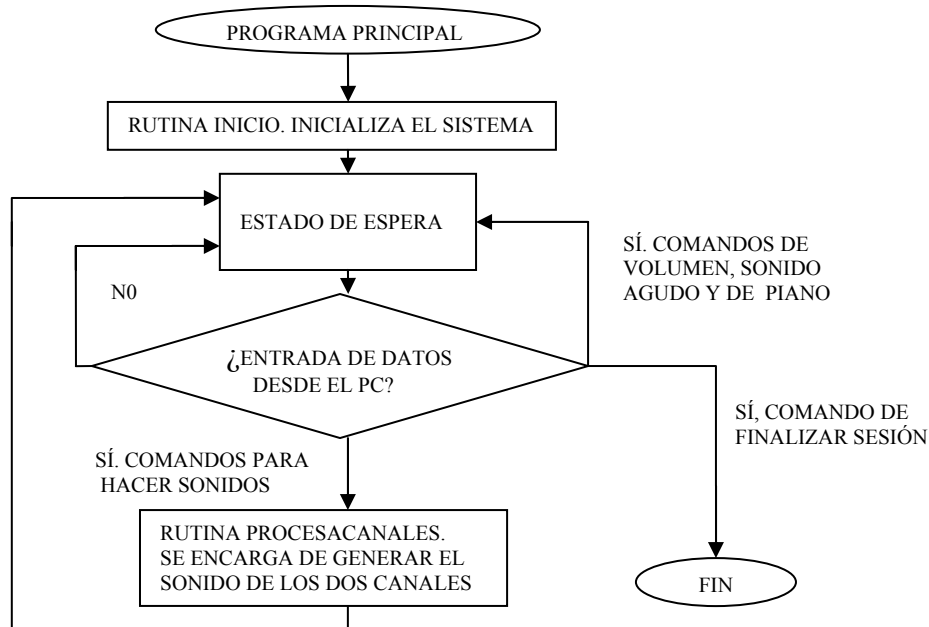


Figura 4.6. Diagrama de flujo del programa principal

El programa principal inicializa el sistema y entra en estado de espera hasta recibir comandos desde el PC. Cuando recibe un comando para generar sonidos, llama a la rutina ‘procesacanales’, que se encarga de generarlos; si el usuario acaba la sesión, recibe un comando para salir del programa, y si el comando es para controlar el volumen de algún canal, para activar o desactivar el efecto ‘sonido agudo’ o para la opción de piano (esta opción se explica en el capítulo 5), actualiza los datos correspondientes y sigue en estado de espera.

El código del programa principal y de todas las rutinas implementadas en C se encuentra en el archivo ‘*sintesis.c*’, que aparece en el CD adjunto a esta memoria.

4.3.1. INTERRUPCIONES EN EL TMS320C30

Las interrupciones del TMS320C30 en lenguaje C deben tener la nomenclatura ‘c_intxx’ para su rutina de tratamiento de interrupción, donde ‘xx’ son dos dígitos, que indican normalmente la dirección del vector correspondiente a esa interrupción, aunque pueden tener cualquier valor. Se puede cambiar el nombre a esa interrupción definiendo una constante cuyo nombre sea ‘c_intxx’.

Antes de definir las interrupciones se ha definido una sección para ubicar los vectores de las interrupciones. Esta sección comienza en la dirección 00h y acaba en la 3fh, que son las direcciones reservadas para los vectores de las interrupciones. En esta sección, en la dirección del vector correspondiente, es donde se da un nombre con la forma ‘c_intxx’ a la rutina de tratamiento de esa interrupción. Esta definición de interrupciones se ha hecho en lenguaje ensamblador en la rutina inicio, que en el CD adjunto a esta memoria se llama ‘inicio.asm’. En lenguaje ensamblador se usa la nomenclatura ‘_c_intxx’, ya que las variables y rutinas definidas en ensamblador y usadas en C han de llevar delante el símbolo ‘_’.

Las rutinas de tratamiento de interrupción en C se usan como cualquier otra rutina, excepto que no se le pueden pasar parámetros, por lo que deben trabajar con variables globales.

En la tabla 4.1 se muestran las interrupciones que se han usado para la realización del sintetizador (con el nombre con que aparecen en la tabla 2.3) y el nombre en C que se le ha asignado:

Nombre de la interrupción	Nombre en C
RESET	c_int00
INT0	c_int01
INT1	c_int02
INT2	c_int03
RINT0	c_int06

Tabla 4.1. Interrupciones usadas y su nombre en C

4.3.2. LA COMUNICACIÓN CON EL PC

La comunicación con el PC se realiza mediante interrupciones. En concreto se usan las interrupciones ‘INT0’ (rutina c_int01), ‘INT1’ (rutina c_int02) e ‘INT2’ (c_int03). En la tabla 4.2 se muestra el nombre de los comandos que recibe el sintetizador desde el PC, su función y un código para identificar a los comandos:

Nombre del comando	Función del comando	Comando
modulacion1	Escribir el tipo de modulación del canal 1.	1
senal1	Escribir el tipo de señal del canal 1.	2
volumen1	Escribir el volumen del canal 1.	3
nota1	Escribir la nota del canal 1.	4
octava1	Escribir la octava del canal 1.	5
tiempo1	Escribir el tiempo de duración de la nota del canal 1.	6
ampmodam1	Escribir la amplitud de la onda moduladora AM del canal 1.	7
contam1	Escribir la componente continua de la síntesis AM del canal 1.	8
desfm1	Escribir la desviación de frecuencia de la onda portadora de la síntesis FM del canal 1.	9
facmodam1	Escribir el factor multiplicador de la frecuencia de la onda moduladora de la síntesis AM del canal 1.	10
facmodfm1	Escribir el factor multiplicador de la frecuencia de la onda moduladora de la síntesis FM del canal 1.	11
modulacion2	Escribir el tipo de modulación del canal 2.	12
senal2	Escribir el tipo de señal del canal 2.	13
volumen2	Escribir el volumen del canal 2.	14
nota2	Escribir la nota del canal 2.	15
octava2	Escribir la octava del canal 2.	16
tiempo2	Escribir el tiempo de duración de la nota del canal 2.	17
ampmodam2	Escribir la amplitud de la onda moduladora AM del canal 2.	18
contam2	Escribir la componente continua de la síntesis AM del canal 2.	19
desfm2	Escribir la desviación de frecuencia de la onda portadora de la síntesis FM del canal 2.	20
facmodam2	Escribir el factor multiplicador de la frecuencia de la onda moduladora de la síntesis AM del canal 2.	21
facmodfm2	Escribir el factor multiplicador de la frecuencia de la onda	22

	moduladora de la síntesis FM del canal 2.	
volumentotal	Escribir el mismo volumen para los dos canales.	23
funciona	Fin de sesión del sintetizador.	24
agudo	Activar o desactivar el efecto 'sonido agudo'.	25
pianoactivo	Se encuentra activo el piano del interfaz.	26
piano1	Activar o desactivar el canal 1 del piano con tiempo libre.	27
piano2	Activar o desactivar el canal 2 del piano con tiempo libre.	28
piano3	Activar o desactivar el canal 1 del piano con tiempo definido.	29
piano4	Activar o desactivar el canal 2 del piano con tiempo definido.	30
ok	Da permiso para generar un sonido en sonido	31

Tabla 4.2. Comandos que recibe el sintetizador desde el PC

En la tabla 4.2 aparecen comandos relativos a un piano del que se hablará en el capítulo 5.

Cuando el PC escribe un comando en el registro 'COM_CMD' se ejecuta la RTI (rutina de tratamiento de interrupción) 'c_int01'. El TMS320C30 lee del registro 'COM_DATA' el comando. A este registro se accede leyendo en cualquier dirección comprendida entre la 804000h y la 805ffffh; el sintetizador accede leyendo en la posición 805000h.

Esta rutina sólo se encarga de guardar el comando y enviar un dato al PC para indicarle que ha recibido el comando, ya que el TMS320C30 no puede interrumpir al PC y éste debe controlar por 'polling' los datos que le envíe el TMS320C30. Todos los comandos de la figura 4.2 relativos al 'canal 1' o 'canal2' indican que el siguiente dato (dato y no comando) a recibir desde el PC debe ser aplicado al parámetro correspondiente a ese comando.

Cuando el PC comprueba que el sintetizador ha recibido el comando, escribe un dato en el registro 'COM_DATA' y se ejecuta la RTI 'c_int02'. El TMS320C30 vuelve a acceder al registro 'COM_DATA', pero esta vez para leer un dato.

'c_int2' se encarga de identificar el comando guardado por 'c_int01' y leer el dato, que será usado para actualizar el parámetro correspondiente. También llama a una subrutina encargada calcular la frecuencia de la señal a partir de una nota y octava

usando la ecuación 3.3 del apartado 3.1.2 y calcular el índice de modulación FM a partir de la desviación de frecuencia de la onda portadora, usando la ecuación 4.4. Por último, esta interrupción envía otro dato al PC para indicarle que ha recibido el dato.

Cada vez que el PC lee un dato enviado por el TMS320C30 se ejecuta la RTI 'c_int03'. Cuando esta rutina se ejecuta tras enviar el sintetizador el dato que indica que ha recibido un comando, no realiza ninguna acción, pero cuando se ejecuta tras enviarse el dato que indica que el sintetizador ha recibido un dato, esta rutina comprueba si se ha recibido algunos de los comandos que hacen que se llame desde el programa principal a la rutina encargada de generar sonido. Esta rutina también comprueba qué parte del interfaz del PC es la que se está ejecutando, ya que de ello dependen los tiempos de las notas musicales. Todo lo referente al interfaz del PC se comenta en el capítulo 5.

Los datos que activan la rutina de generar sonido, según la tabla 4.2, son los siguientes: modulacion1, senial1, nota1, octava1, tiempo1, ampmodam1, contam1, facmodam1, desfm1, facmodfm1, modulacion2, senial2, nota2, octava2, tiempo2, ampmodam2, contam2, facmodam2, desfm2 y facmodfm2.

En la figura 4.7 se representa el diagrama de flujo de la comunicación con el PC:

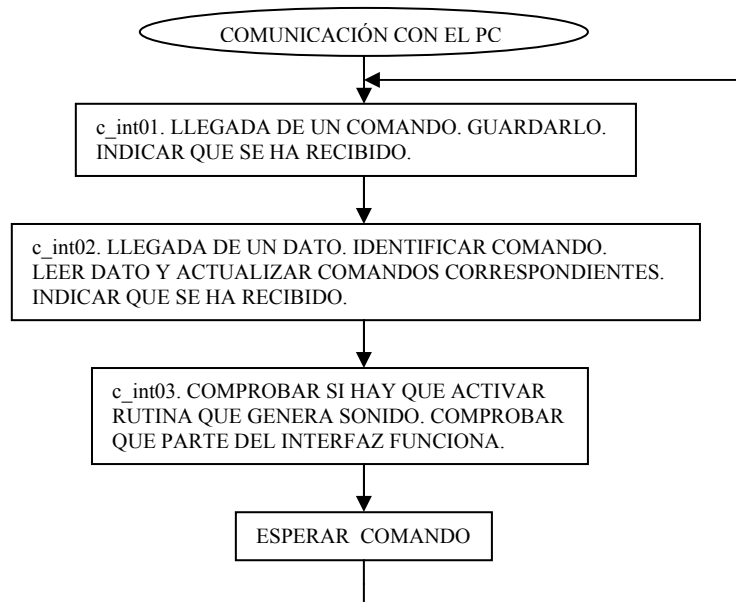


Figura 4.7. Diagrama de flujo de la comunicación con el PC

4.3.3. SINCRONIZACIÓN DEL SINTETIZADOR

Como se vio en el apartado 4.2.2.5 en la ecuación 4.22, para sustituir la variable de tiempo continuo en la ecuación de las ondas sonoras generadas, hay que tener la frecuencia de muestreo y el índice de la muestra. La frecuencia de muestreo es 8013 Hz y el índice de la muestra se obtiene en la RTI 'c_int06'.

Al estar habilitada la interrupción RINT0 (que tiene a 'c_int06' como RTI), se consigue que cada vez que el AIC lea una muestra del exterior (aunque no haya ninguna señal externa, el AIC está configurado para leer una muestra del exterior) se produzca una interrupción. Ésta se produce a una frecuencia de 8013 Hz, es decir, a la frecuencia de muestreo que tiene el conversor analógico – digital del AIC, que es el que permite leer muestras externas. Esta RTI tiene dos contadores, uno por cada canal, que llevan la cuenta del número de interrupciones que se producen, es decir, el número de muestras que se han leído, con lo que cada contador ejerce como índice de las muestras.

Cada vez que se introduce un comando para generar música, no se deja acabar el sonido que se estuviera reproduciendo, en caso de haber alguno, y se genera un sonido nuevo, los parámetros que no se hayan modificado conservan su valor.

El usuario puede exigir que una determinada nota musical tenga una duración de las que aparece en un pentagrama convencional. Si se da este caso, cada vez que se cambie algún parámetro que genera sonido, habrá que reproducir la nota durante el tiempo exigido. Estos tiempos están predefinidos y para fijar su duración es necesario controlar el número de muestras. Como la RTI 'c_int06' está contando indefinidamente, cada vez que se produce esta situación se inicializa a cero el contador del canal que vaya a sonar, para así poder controlar los tiempos de forma independiente, ya que de esta forma se conoce el valor inicial del índice de las muestras y se pueden controlar los tiempos, generando sonido sólo hasta cierto valor del índice de las muestras.

4.3.4. INICIALIZACIÓN DEL SISTEMA

EL bloque para inicializar el sistema está implementado por la rutina ‘inicio’. Esta rutina está implementada en lenguaje ensamblador, ya que escribe en muchos registros del TMS320C30, para configurarlo de la forma deseada. Esta rutina no realiza el reset del sistema (que se explica en el apartado 4.4). En la figura 4.8 se muestra el diagrama de flujo de este bloque:

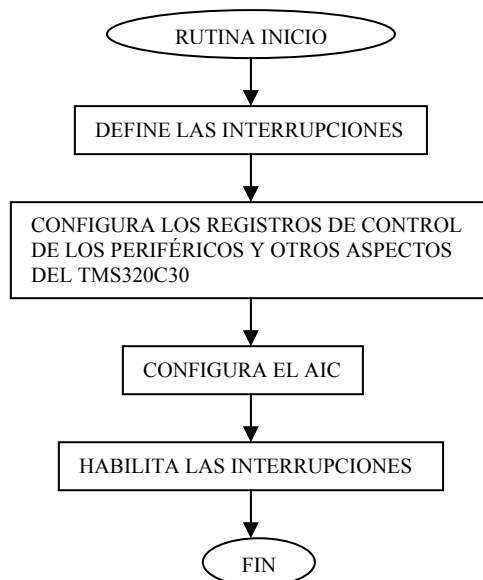


Diagrama 4.8. Diagrama de flujo de la rutina ‘inicio’

Lo primero que se hace es definir las interrupciones que se van a usar. A continuación se dan valores a los registros de control de la memoria caché, el bus de expansión, el bus primario, el temporizador 0 y el puerto serie 0 y a los registros de estado y habilitación de interrupciones.

Los periféricos que se controlan son el temporizador 0, para generar el reloj del AIC, y el puerto serie 0, que es el puerto serie que el EVM conecta al AIC para comunicar al TMS320C30 con el exterior.

El AIC es el encargado de convertir las muestras del TMS320C30 en una señal analógica, que es la onda sonora que genera el sintetizador. Está programado para que tenga una frecuencia de muestreo de 8 KHz (8013 Hz), tanto para escribir muestras como para leerlas. Por último, esta rutina habilita las interrupciones definidas en el apartado 4.3.1.

4.3.5. GENERACIÓN DE SONIDOS

El bloque de generar sonidos es el encargado de generar y procesar las muestras que se convertirán en ondas sonoras, además de enviarlas al AIC, para que las saque al exterior con una frecuencia de muestreo 8013 Hz. La rutina principal de este bloque se ha llamado ‘procesacanales’. Esta rutina es llamada desde el programa principal, aunque si mientras se está reproduciendo un sonido, se produce una interrupción para generar sonidos nuevos, se continuará ejecutando con los parámetros actualizados hasta que se acabe el sonido. Estas interrupciones pueden llegar en cascada de forma ilimitada. En la figura 4.9 se muestra el diagrama de flujo de este bloque:

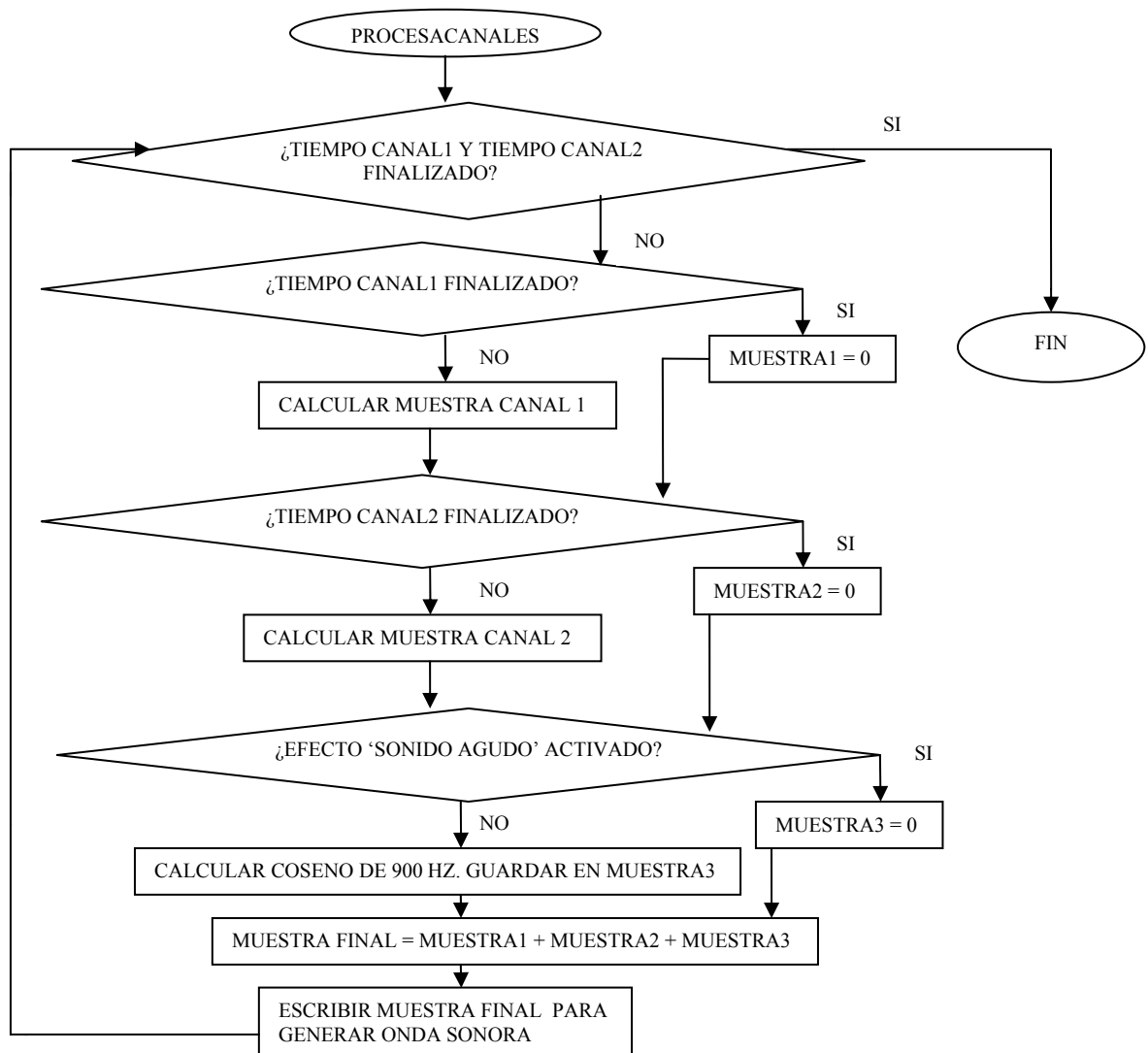


Figura 4. 9. Diagrama de flujo de la rutina ‘procesacanales’

Para calcular las muestras de cada canal, mientras aún no se haya acabado el tiempo de la nota musical del canal correspondiente, se llama a otra rutina que se encarga de calcular el valor de una muestra, que depende del tipo de síntesis, de señal y de los parámetros de la síntesis y la señal elegida. Esa rutina se llama ‘calculamuestra’ y es llamada por ambos canales, pasándole en la llamada los parámetros de cada canal. A su vez esta rutina lo que hace es ver qué tipo de síntesis se pide y llamar a la rutina que implementa ese tipo de síntesis, que le devuelve una muestra que ‘calculamuestra’ devolverá a la rutina ‘procesacanales’. En esa llamada, ‘calculamuestra’ sólo envía los parámetros propios de la síntesis elegida. En la figura 4.10 se muestra el diagrama de flujo de ‘calculamuestra’:

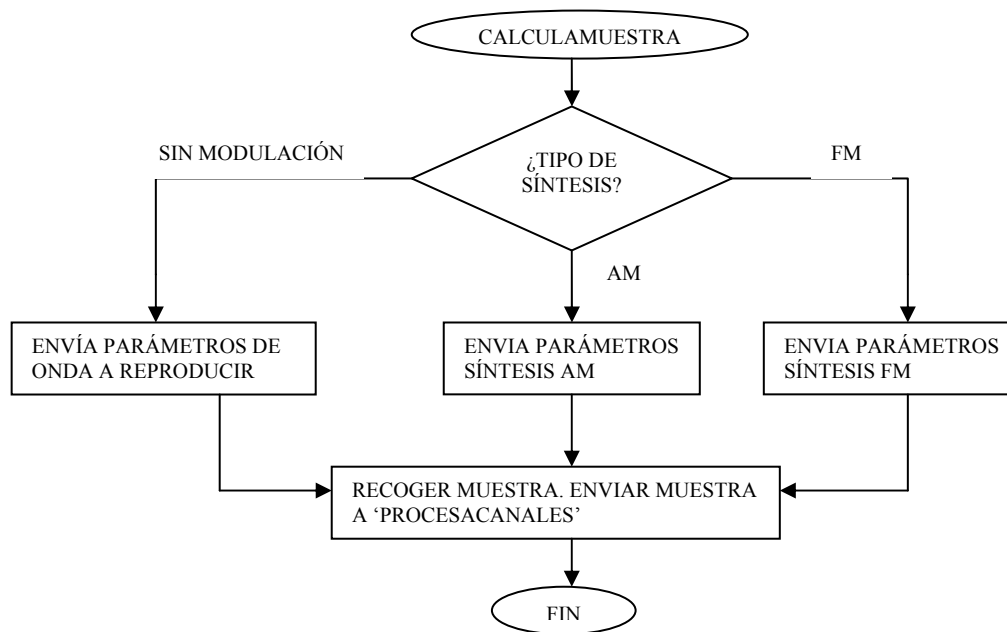


Figura 4.10. Diagrama de flujo de la rutina ‘Calculamuestra’

En cada nota musical, la rutina ‘calculamuestra’ y las rutinas que son llamadas por ‘calculamuestra’, se ejecutan tantas veces como muestras pueda tener el tiempo de duración de esa nota musical, por cada canal.

En la síntesis denominada ‘Sin Modulación’, se llama a la función que genera la señal que se ha especificado en la llamada como moduladora, pasándole a la señal los parámetros amplitud, frecuencia y muestra actual, y se devuelve la muestra calculada. El nombre de esta rutina es ‘sinmod’, se implementa en el archivo ‘sintesis.c’. En la figura 4.11 se muestra su diagrama de flujo:

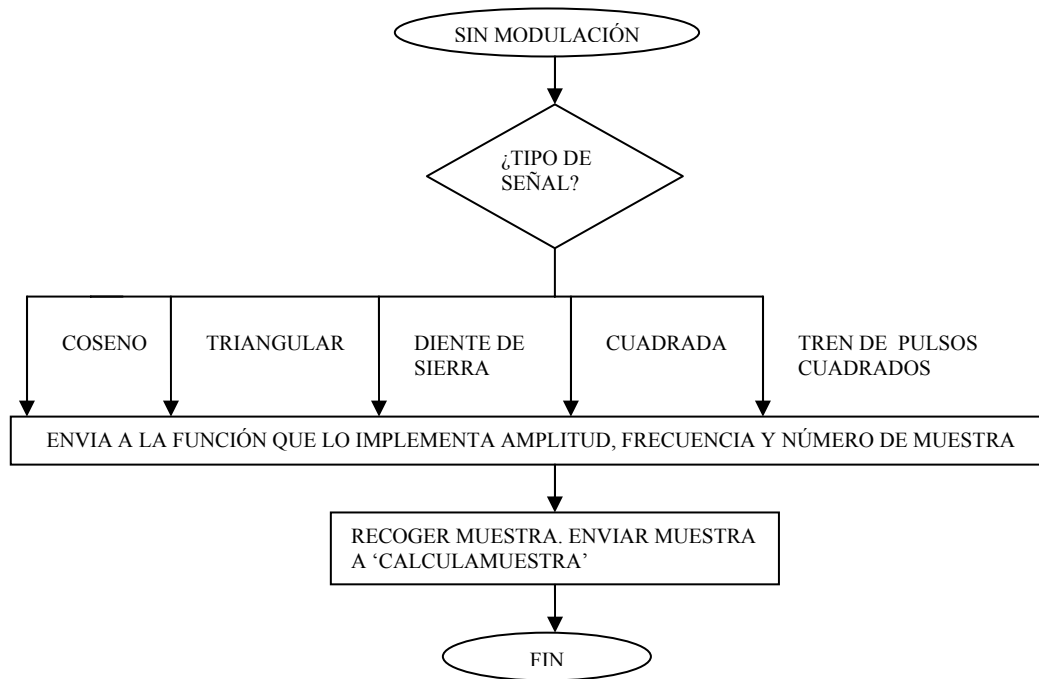


Figura 4.11. Diagrama de flujo de la rutina ‘sinmod’

En la síntesis AM se llama a la función que genera la señal que se ha indicado en la llamada como moduladora, pasándole los parámetros amplitud, frecuencia y número de muestra. Después se llama a la función que implementa el coseno para calcular la onda portadora. Para implementar el coseno como señal portadora, en la síntesis AM se llama a la función ‘coseno’ del archivo ‘sintesis.c’. La amplitud de la portadora se multiplica por la muestra que contiene el valor de la moduladora. A la portadora se le pasa la amplitud, la frecuencia y el número de muestra.

La función ‘coseno’ implementa la ecuación 4.7 del apartado 4.2.2.1, siendo ‘A’ el valor devuelto por la función que implementa a la señal moduladora.

La rutina que implementa la síntesis AM se llama ‘modam’ en el archivo ‘sintesis.c’.

En la figura 4.12 puede verse el diagrama de flujo de la rutina que implementa a la síntesis AM:

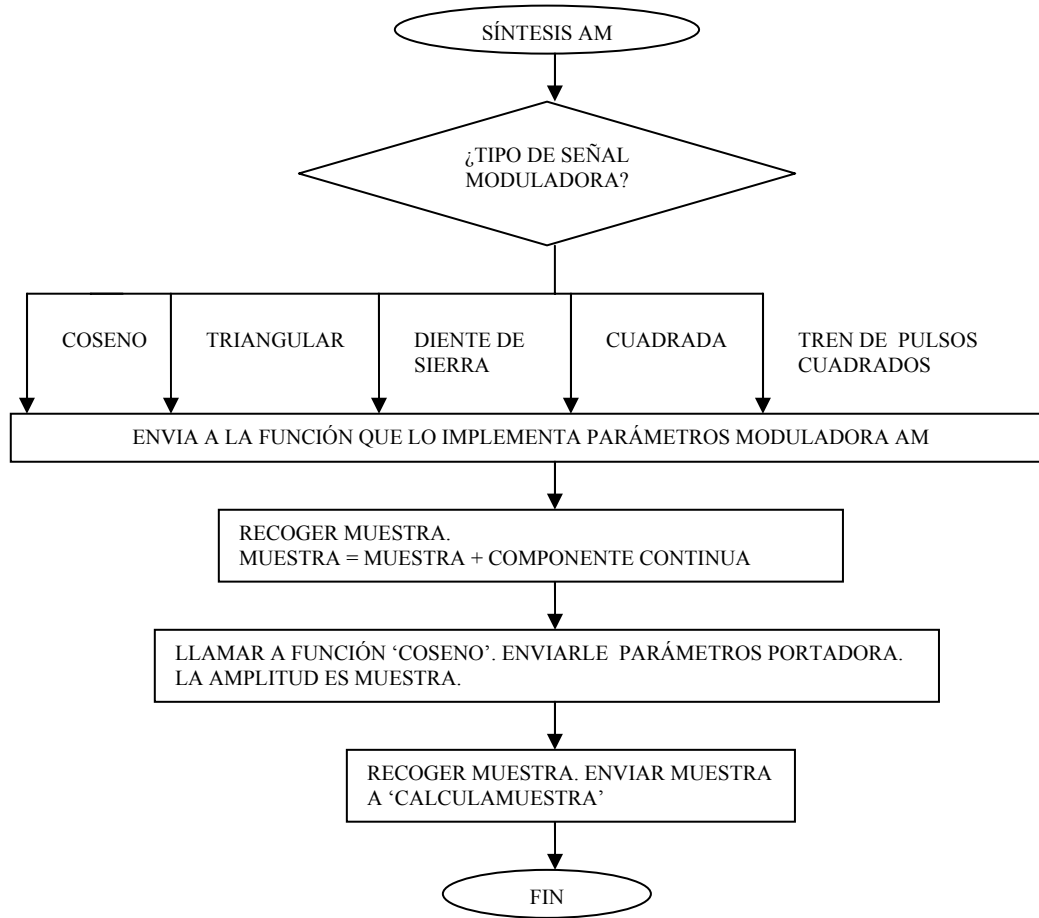


Figura 4.12. Diagrama de flujo de la rutina 'modam'

En la síntesis FM, se llama a la función que implementa la señal que se ha indicado en la llamada como moduladora, pasándole los parámetros amplitud (índice de modulación), frecuencia y número de muestra. Después se llama a la función que implementa el coseno para calcular la onda portadora. Para implementar el coseno como señal portadora, en la síntesis FM se llama a la función denominada 'cosenofm' en el archivo 'síntesis.c'. Como fase de la portadora se pasa la muestra que contiene el valor de la moduladora, también a la portadora se le pasa la amplitud, la frecuencia y el número de muestra.

La función 'cosenofm' implementa la ecuación 4.8 del apartado 4.2.2.1, siendo 'mod' el valor devuelto por la función que implementa a la señal moduladora.

La rutina que implementa la síntesis FM se llama 'modfm' en el archivo 'síntesis.c'.

En la figura 4.13 puede verse el diagrama de flujo de la rutina que implementa a la síntesis FM:

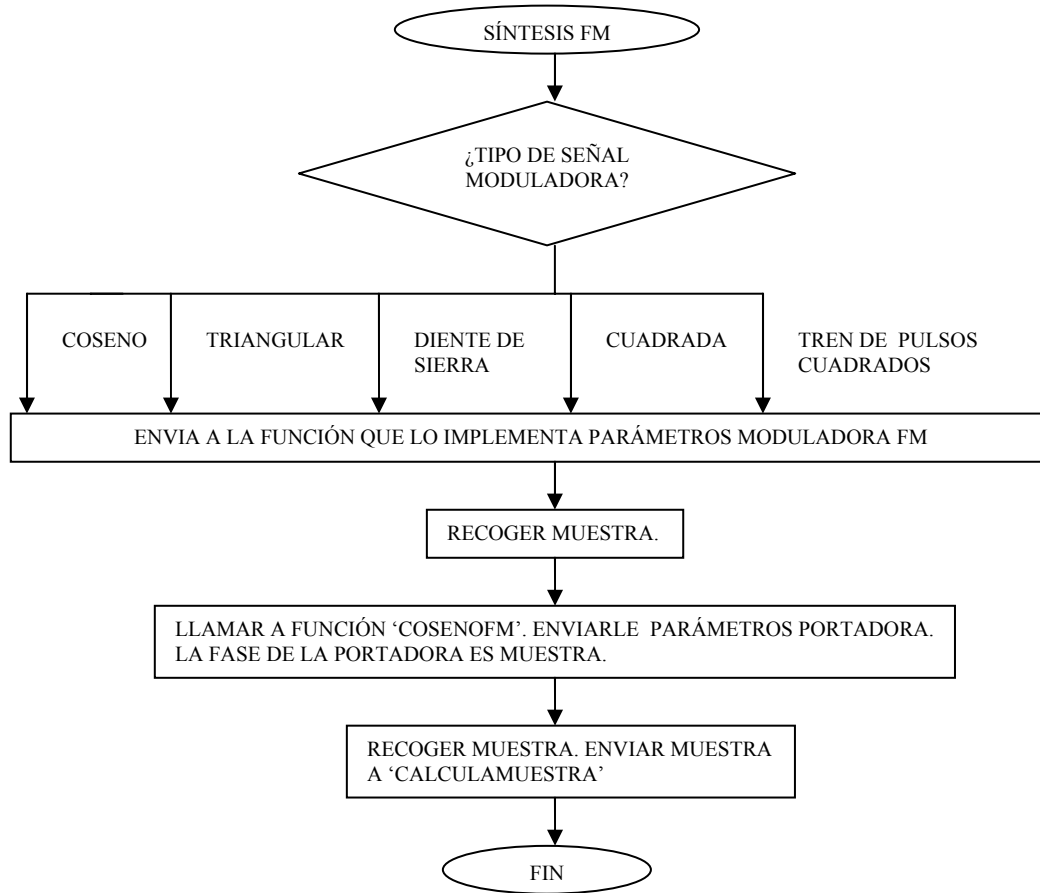


Figura 4.13. Diagrama de flujo de la rutina 'modfm'

Las rutinas que implementan a las señales simplemente ejecutan las fórmulas que aparecen en el apartado 4.2.2, obteniéndose el índice de la muestra gracias a la RTI 'c_int06', que contiene en variables globales el índice de muestra de cada canal. Todas las rutinas que implementan a los distintos tipos de señales devuelven un valor al tipo de síntesis que las hayan llamado. Estas rutinas calculan el valor de las muestras en un periodo, para el resto de periodos, desplazan las muestras el número de periodos necesarios para que la muestra quede dentro del primer periodo y así se determina su valor.

Finalmente, la rutina 'procesacanales', es decir, la rutina principal del bloque para generar sonidos, escribe la muestra en el puerto serie 0 para que el AIC la saque al exterior. La rutina que hace esto está implementada en lenguaje ensamblador y se llama 'wmuestra'. Cuando desde C se llama a una rutina implementada en ensamblador, a la

que se le pasan parámetros, este paso de parámetros puede hacerse a través de registros o de la pila. En este caso, el parámetro que se pasa en la llamada es la muestra a escribir y el paso se hace a través de la pila. La rutina está implementada en el fichero 'wmuestra.asm'.

En la figura 4.14 se muestra el diagrama de flujo de la rutina 'wmuestra':



Figura 4.14. Diagrama de flujo de la rutina 'wmuestra'

4.4. RESET DEL SISTEMA. ARCHIVO DE COMANDOS

Para ejecutar cualquier programa en el EVM hay que resetear todo el sistema, de forma que quede inicializado. Para ello, tras cargar el programa ejecutable, se debe ejecutar en el *'Debugger'* el comando *'reset'*. Este comando es la etiqueta de la dirección 0h de la tabla de vectores, correspondiente a la interrupción *'RESET'*. Esta dirección contiene la dirección de comienzo de la RTI que se ejecutará cada vez que en el *'Debugger'* se ejecute el comando *'reset'*. En ensamblador se le puede poner cualquier nombre a esa RTI y ha de implementarla el programador.

En lenguaje C, la librería *'rts.src'* implementa la RTI para el reset. Esa rutina está implementada en ensamblador, pero se usa en C, por lo que se llama *'_c_int00'*. Se encarga de definir distintas secciones de memoria propias del lenguaje C y también define la pila del sistema, además de llamar al programa principal, que ha de denominarse *'main'*. Por tanto, en C no es necesario implementar la RTI para el reset y hay que tener cuidado de no realizar operaciones de inicialización del sistema contrarias a las que realiza esa RTI. Tampoco debe aparecer *'_c_int00'* en ninguna parte del código C, al contrario que las otras RTI. Lo único que debe hacer el programador es asignar a la dirección cero de la tabla de vectores el nombre *'_c_int00'* para su RTI. El código del sintetizador direcciona a *'_c_int00'* en la rutina *'inicio'*. Si se desea implementar una RTI para el reset distinta, basta con poner un nombre distinto para la RTI o cambiar el nombre de la rutina en la librería *'rts.src'*, recordando que en C todas las interrupciones deben seguir la nomenclatura *'c_intxx'*.

Se ha creado un fichero de comandos llamado *'sintesis.cmd'*. Es el fichero que se usa para linkar. Este fichero define las distintas secciones de memoria y las asigna en memoria física. Además este fichero contiene los ficheros ensamblados a linkar. Para el sintetizador se han creado tres ficheros: *'sintesis.c'*, *'inicio.asm'* y *'wmuestra.asm'*, que en el fichero de comandos deberán tener la extensión *'obj'*. Además, en *'sintesis.cmd'* también se especifica el nombre de las librerías de C propias del TMS320C30 usadas y el nombre que tendrá el fichero a cargar en el *'Debugger'* para su ejecución. A este fichero se le ha denominado *'sintesis.out'*. Cuando el programa principal está implementado en C, hay que indicarlo en el archivo de comandos con la sentencia *'-c'*.

4.5. CONDICIONES PARA LA CREACIÓN DEL EJECUTABLE DEL SINTETIZADOR CON EL EVM

Para la realización del sintetizador se han usado las librerías específicas para el DSP TMS320C30 ‘rts.src’, ‘rts30.lib’ y ‘rts30g.lib’. La librería ‘rts.src’ contiene el código fuente en ensamblador de las funciones de todas las librerías estándares como ‘math.h’ o ‘stdio.h’, siendo ‘math.h’ la única que se ha usado en el sintetizador.

El código C de todas las rutinas y del programa principal se encuentra en el archivo ‘sintesis.c’. También se ha creado el archivo ‘sintesis.h’, que contiene la definición de rutinas, variables y constantes que usa ‘sintesis.c’. En el mismo directorio donde se encuentran estos dos ficheros, se encuentran los ficheros con código ensamblador ‘inicio.asm’ y ‘wmuestra.asm’, ya que todos los ficheros de código deben estar en el mismo directorio. El directorio se llama ‘sintesis’.

También en el directorio ‘sintesis’ se encuentra el fichero ‘sintesis.cmd’. Dentro del directorio ‘sintesis’ se ha creado un subdirectorio llamado ‘libs’. Este subdirectorio contiene las librerías de C que ha usado ‘sintesis.c’. La ruta del directorio donde están las librerías ha de especificarse a la hora de compilar y linkar en C, quedando las sentencias para compilar y linkar en C de la siguiente forma:

Para compilar: `cl30 -ilibs sintesis.c, ‘-i’` es la opción para poner el subdirectorio.

Para linkar: `lnk30 -ilibs sintesis.cmd`, tras haber compilado los archivos en ensamblador; ‘-i’ es la opción para poner el subdirectorio.

CAPÍTULO 5: Implementación del interfaz con el PC

Crear un interfaz gráfico para el usuario es uno de los objetivos que se deben cumplir. El sistema operativo debe ser alguna versión de Windows, ya que el software disponible para compilar y depurar programas para el entorno de desarrollo TMS320C30 EVM con el PC, sólo permite MS-DOS o Windows 3.11, 95, 98 o ME, y como el interfaz debe ser gráfico, no puede ser MS-DOS.

5.1. DECISIONES SOBRE EL SOFTWARE

El PC con el que realizar este proyecto debe disponer de bus ISA, para conectar el EVM.

La versión de Windows debe permitir que, en la comunicación entre el PC y el EVM, el PC acceda a puertos directamente. Esto hace que la versión de Windows sea de 16 bits.

Para la programación del interfaz se buscó un entorno visual, que facilitase la programación visual del interfaz. El problema es que los entornos visuales trabajan con código de 32 bits, al contar con aplicaciones de entorno gráfico.

Entonces se pensó en buscar algún ‘*driver*’, que permitiese comunicarse con el EVM y usar entornos de programación visuales y versiones de Windows de 32 bits, o usar algún lenguaje que no fuese visual y Windows 3.11.

Se encontró el ‘*driver*’ en la librería ‘io.h’ y se eligió el lenguaje Borland C++ Builder 6.0, que es un entorno de programación sencillo, y Windows 98, por ser la versión más estable.

5.2. FUNCIONAMIENTO DEL INTERFAZ

Para que quede más claro el funcionamiento del interfaz de usuario, se hará una explicación basada en sus distintos elementos gráficos. Mediante este interfaz, se eligen los parámetros de las señales y los distintos tipos de síntesis, además de controlar la activación de los dos canales, para la realización de sonido del TMS320C30 EVM. En la figura 5.1 se muestra la imagen del interfaz antes de iniciar una sesión:

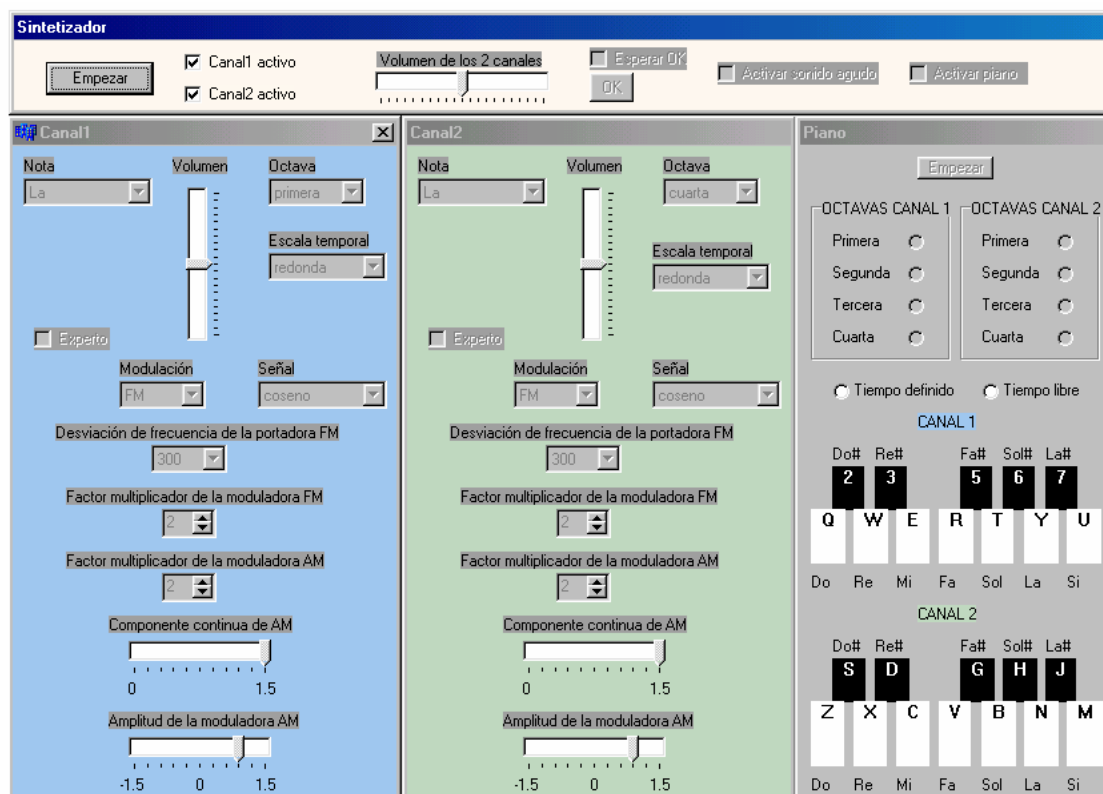


Figura 5.1. Imagen completa del interfaz de usuario del sintetizador

En la figura 5.1 pueden diferenciarse cuatro zonas, si bien dos son iguales. En los siguientes apartados se explicará el funcionamiento de cada ventana.

5.2.1. LAS VENTANAS CANAL1 Y CANAL2

Estas ventanas controlan los parámetros de cada canal. Son las que envían los 22 primeros comandos que aparecen en la tabla 4.2 del apartado 4.3.2 al EVM, los once primeros los envía la ventana ‘Canal1’ y los once siguientes la ventana ‘Canal2’. Las ventanas tienen el mismo funcionamiento y realizan las mismas acciones, pero cada una controla un canal diferente, por lo que sólo se explicará ‘Canal1’. La única diferencia es que la ventana ‘Canal1’ es la encargada de cerrar una sesión del sintetizador. En la figura 5.2 se muestra la imagen de la ventana ‘Canal1’:

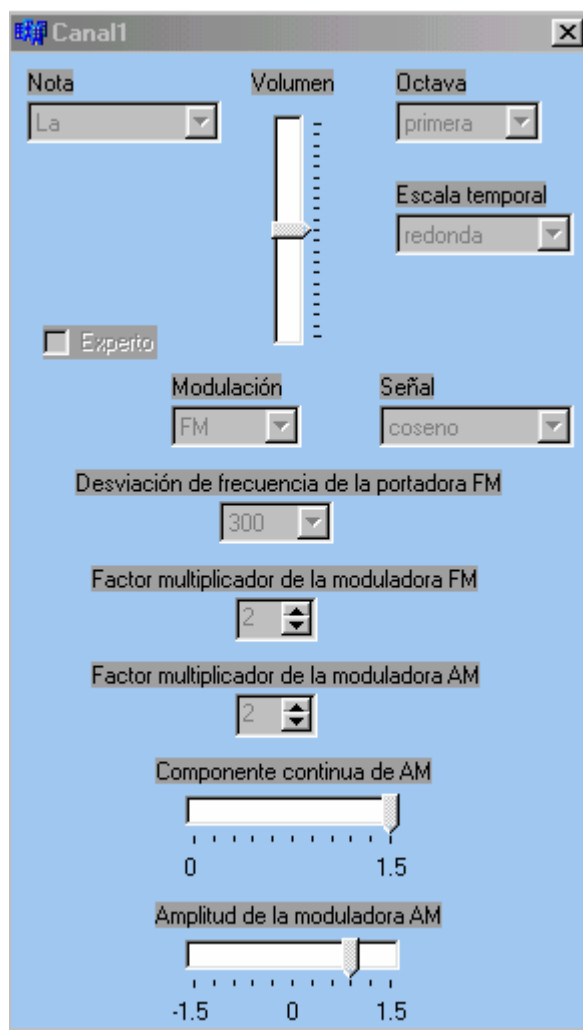


Figura 5.2. Imagen de la ventana Canal1

A continuación se van a explicar los distintos elementos gráficos que aparecen en la figura 5.2, refiriéndose a ellos por la etiqueta que tienen en la figura, y se dirá en los casos oportunos los comandos de la tabla 4.2 que envían al EVM:

- ‘Nota’: contiene las doce notas musicales y permite elegir una nota musical. Al seleccionar ese objeto, se despliega una ventana con las opciones: Do, Do sostenido, Re, Re sostenido, Mi, Fa, Fa sostenido, Sol, Sol sostenido, La, La sostenido, Si. Envía al EVM el comando ‘nota1’. ‘Nota’ se deshabilita cuando se activa la ventana ‘Piano’.
- ‘Volumen’: controla el volumen del canal. El volumen puede variar desde cero, no oyéndose el sonido, hasta la mitad de la amplitud máxima que puede alcanzar una señal. Sólo se alcanza la mitad de la amplitud máxima de una señal, para evitar que se produzca saturación cuando suenan los dos canales a la vez. ‘Volumen’ modifica la amplitud de la señal portadora en las modulaciones y la amplitud de la señal elegida si no hay modulación. El volumen envía el comando ‘volumen1’.
- ‘Octava’: contiene las cuatro octavas que implementa el sintetizador y permite elegir una octava. Al seleccionar ese objeto se despliega una ventana que contiene las opciones: Primera, Segunda, Tercera y Cuarta. Se deshabilita cuando ‘Piano’ está activo. Envía el comando ‘octava1’.
- ‘Escala temporal’: contiene los tiempos de duración de una nota musical que aparecen en un pentagrama, teniendo la redonda una duración de 6 segundos, y permite elegir el tiempo de la nota musical. Seleccionando ese objeto se despliega una ventana que contiene las opciones: desactivado, redonda, blanca, negra, corchea, semicorchea, fusa y semifusa. El valor ‘desactivado’ aparece cuando se desactiva el canal desde la ventana ‘Sintetizador’ y hace que el TMS320C30 le dé un valor de 0 segundos al tiempo del canal 1, ya que el sintetizador siempre suma los dos canales y decide el valor de las muestras en función del tiempo. Si el usuario elige ‘desactivado’, no se oirá ningún sonido en el canal 1. Cuando el canal 1 vuelve a activarse, ‘Escala temporal’ muestra el valor ‘redonda’. ‘Escala temporal’ se

deshabilita cuando ‘Piano’ se activa con la opción ‘Tiempo libre’. Envía el comando ‘tiempo1’ al EVM.

Los cuatro elementos gráficos que acaban de explicarse están deshabilitados hasta que se inicie la sesión desde la ventana ‘Sintetizador’. Una vez iniciada la sesión se habilitan.

- ‘Experto’: permite usar las opciones avanzadas para generar un sonido. Estas opciones son: ‘Modulación’, ‘Señal’, ‘Desviación de la frecuencia de la portadora FM’, ‘Factor multiplicador de la moduladora FM’, ‘Factor multiplicador de la moduladora AM’, ‘Componente continua de AM’ y ‘Amplitud de la moduladora AM’. Este objeto envía los comandos ‘modulacion1’, ‘senal1’, ‘ampmodam1’, ‘contam1’, ‘desfm1’, ‘facmodam1’ y ‘facmodfm1’ cuando se desactiva, para escribir los valores por defecto en las opciones avanzadas. Los valores por defecto generan sonidos de calidad en las cuatro octavas y permiten que el usuario sólo se preocupe de los parámetros musicales que aparecen en un pentagrama y del volumen.
- ‘Modulación’: permite elegir el tipo de síntesis. Cuando se selecciona ese objeto se despliega una ventana que contiene las opciones: ‘ninguna’, indica que no se va a realizar ningún tipo de modulación, se corresponde con la opción ‘Sin Modulación’ de la figura 4.1 del apartado 4.2; ‘AM’, se elige modulación AM y ‘FM’, se elige modulación FM. Su valor por defecto es FM, ya que esta modulación genera sonidos de calidad en las cuatro octavas con las señales coseno y triangular y es la que en general produce sonidos de más calidad. Envía al EVM el comando ‘modulacion1’.
- ‘Señal’: permite elegir la señal moduladora, si se realiza alguna modulación, o la señal que sonará sola, si no hay modulación. Al seleccionar con el ratón ese objeto se despliega una ventana que contiene las opciones: coseno, triangular, diente de sierra, cuadrada y tren de pulsos. La señal por defecto es la señal coseno, ya que genera sonidos de calidad junto a la síntesis FM en todas las octavas. Envía el comando ‘senal1’.

- ‘Desviación de frecuencia de la portadora FM’: permite elegir la desviación de frecuencia de la señal portadora de la síntesis FM, entre 10 valores distintos que aparecerán al seleccionar el objeto. Estos valores son: 100, 200, 300, 400, 500, 600, 700, 800, 900 y 1000. Los saltos de 100 permiten variar el sonido ligeramente, permitiendo que los sonidos sean parecidos pero no idénticos, consiguiéndose matices distintos. El valor máximo es 1000, ya que a partir de este límite se produce mucho ruido en las octavas más altas, y el mínimo es 100, ya que con el valor 0 no se produce modulación FM. Este objeto envía el comando ‘desfm1’.
- ‘Factor multiplicador de la moduladora FM’: determina el número de veces que será mayor la frecuencia de la señal moduladora respecto de la portadora en la síntesis FM. Se incrementa o decrementa en una unidad, siendo 2 el valor mínimo y el valor por defecto y 4 el valor máximo, ya que a partir de 4 el ruido es demasiado grande para la cuarta octava. Envía al EVM el comando ‘facmodfm1’.
- ‘Factor multiplicador de la moduladora AM’: determina el número de veces que será mayor la frecuencia de la señal moduladora respecto de la portadora en la síntesis AM. Se incrementa o decrementa en una unidad, siendo 2 el valor mínimo y el valor por defecto y 4 el valor máximo, ya que a partir de 4 el ruido es demasiado grande para la cuarta octava. Envía al EVM el comando ‘facmodam1’.
- ‘Componente continua de AM’: permite variar la componente continua de la síntesis AM. Los valores posibles están en el rango 0 - 1’5. Su valor por defecto es 1’5. Los saltos entre los valores son de 0’15. Envía el comando ‘contam1’.
- ‘Amplitud de la moduladora AM’: se encarga de controlar el valor de la amplitud de la señal moduladora en la síntesis AM. El rango de valores posibles es -1’5 – 1’5. Cada incremento o decremento es de 0’3. Su valor por defecto es 0’9. Envía el comando ‘ampmodam1’ al EVM.

Los parámetros de la ventana ‘Canal1’ que varían el timbre de un sonido son: ‘Nota’, ‘Octava’, ‘Modulación’, ‘Señal’, ‘Desviación de frecuencia de la portadora FM’, ‘Factor multiplicador de la moduladora FM’, ‘Factor multiplicador de la moduladora AM’, ‘Componente continua de AM’ y ‘Amplitud de la moduladora AM’. Todos estos parámetros pueden modificarse de forma independiente y pueden tener diversos valores, consiguiendo que exista una gran variedad de sonidos.

Esa diversidad de sonidos distintos, permite que una nota musical de una determinada octava pueda generar 5 timbres distintos sin modulación, 1500 timbres con síntesis AM y 150 timbres con síntesis FM, sumando un total de 1655 timbres. Teniendo en cuenta que hay 12 notas musicales y 4 octavas, los timbres distintos se elevan hasta 79440. Además el efecto ‘sonido agudo’ tiene un timbre distinto a los demás, con lo que el número total sonidos con un timbre distinto que genera el sintetizador es 79441. Aunque entre estos sonidos pueda haber sonidos que apenas se oigan, que tengan ruido o que se parezcan mucho a otros (variando sólo un pequeño matiz), la cifra de sonidos de calidad con timbres distintos es muy alta, ofreciendo muchas posibilidades al usuario.

La ventana ‘Canal1’ tiene tres modos de funcionamiento, que se controlan desde la ventana ‘Sintetizador’. Al explicar esta ventana se dirá como se elige un modo u otro.

- ‘Primer modo de funcionamiento’: en este modo, cada vez que se produce un cambio en algún objeto habilitado, excepto en ‘Volumen’ y en ‘Experto’, suena una nota musical, en la que los parámetros para generar ese sonido tienen los valores que se visualizan en ese instante en los distintos objetos. Es el modo que hay por defecto al iniciar la sesión.
- ‘Segundo modo de funcionamiento’: permite que no haya ningún sonido hasta que el usuario lo indique, aunque modifique un número ilimitado de veces los valores de cualquiera de los objetos habilitados. La indicación para generar el sonido se hace desde la ventana ‘Sintetizador’. Así al usuario le da tiempo de cambiar varios parámetros antes de generar un sonido.
- ‘Tercer modo de funcionamiento’: se produce cuando se activa la ventana ‘Piano’. En este modo el cambio en los objetos habilitados no genera ningún sonido, sólo

envía al EVM los valores deseados para cuando se haga sonar una nota desde ‘Piano’.

En los dos primeros modos de funcionamiento, cuando está produciéndose un sonido y el usuario hace que se produzca otro, el primer sonido deja de sonar y se empieza a oír el nuevo sonido que ha decidido el usuario, teniendo la duración determinada en el objeto ‘Escala temporal’ en el momento de generar ese nuevo sonido.

5.2.2. LA VENTANA PIANO

En la figura 5.3 se muestra la imagen de esta ventana:



Figura 5.3. Imagen de la ventana Piano

Esta ventana permite generar sonido enviando las notas musicales con el teclado del PC, emulando a un piano o teclado electrónico. Se activa desde la ventana ‘Sintetizador’. Envía al EVM los comandos ‘piano1’, ‘piano2’, ‘piano3’ y ‘piano4’ que aparecen en la tabla 4.2 del apartado 4.3.2.

A continuación se explican los distintos elementos gráficos que aparecen en la figura 5.3:

- ‘Empezar’: este botón es necesario para que las teclas generen sonido al iniciar la sesión de ‘Piano’ o cuando se selecciona cualquier objeto del interfaz del sintetizador. Al inicio de una sesión contiene el texto ‘Empezar’ y deja de ser visible cuando se pulsa, volviendo a ser visible, con el texto ‘Continuar’, cuando se selecciona algún objeto del interfaz.
- ‘OCTAVAS CANAL 1’: permite elegir la octava en que sonará el canal 1 cuando el piano esté activo. Al habilitarse la ventana ‘Piano’, se activa la octava que el objeto ‘Octava’ de la ventana ‘Canal1’ tenga visible, pudiéndose cambiar de octava en cualquier momento. Al deshabilitarse la ventana ‘Piano’, otra vez aparece como octava activada, la octava que el objeto ‘Octava’ de la ventana ‘Canal1’ tenga visible. Envía el comando ‘octava1’ al EVM.
- ‘OCTAVAS CANAL 2’: permite elegir la octava del canal 2 cuando se habilita la ventana ‘Piano’. Funciona igual que ‘OCTAVAS CANAL 1’, pero respecto a la ventana ‘Canal 2’. Envía el comando ‘octava2’.
- ‘Tiempo definido’: cuando se activa, las notas musicales tendrán la duración indicada en el objeto ‘Escala temporal’ del canal correspondiente. Envía los comandos ‘piano3’ y ‘piano4’.
- ‘Tiempo libre’: cuando se activa, las notas musicales suenan mientras la tecla que las produce esté pulsada. Cuando la tecla deja de pulsarse, el sonido deja de producirse. Envía al EVM los comandos ‘piano1’ y ‘piano2’.

- ‘CANAL 1’ y ‘CANAL 2’: emulan las teclas de un piano, asignando una tecla del PC a cada una de las notas musicales, e indican el canal. Las teclas de color negro, que generan las notas sostenidas, cambian a color azul cuando se pulsan y las teclas blancas cambian a color rojo. En la tabla 5.1 se muestran las teclas del PC, junto a la nota musical que generan y el canal al que pertenecen:

TECLAS CANAL 1	Q	2	W	3	E	R	5	T	6	Y	7	U
NOTAS MUSICALES	Do	Do#	Re	Re#	Mi	Fa	Fa#	Sol	Sol#	La	La#	Si
TECLAS CANAL 2	Z	S	X	D	C	V	G	B	H	N	J	M

Tabla 5.1. Teclas del PC con las notas musicales que generan

Para el correcto funcionamiento de la ventana ‘Piano’ es necesario activar ‘Tiempo definido’ o ‘Tiempo libre’, de lo contrario no se escuchará ningún sonido. Los comandos que envían ‘Tiempo definido’ y ‘Tiempo libre’, permiten al EVM decidir el tiempo de duración de cada nota musical.

Cuando está funcionando el piano, al pulsar una tecla, se envía al EVM los comandos ‘nota1’ o ‘nota2’, dependiendo del canal al que pertenezca la tecla pulsada.

Cuando está activada la opción ‘Tiempo definido’, pueden modificarse desde las ventanas ‘Canal1’ y ‘Canal2’ todos los parámetros que estas ventanas controlan, excepto ‘Nota’ y ‘Octava’. Si la opción elegida es ‘Tiempo libre’, los parámetros que no pueden modificarse desde las ventanas ‘Canal1’ y ‘Canal2’ son ‘Nota’, ‘Octava’ y ‘Escala temporal’, ya que ‘Tiempo libre’ deshabilita el elemento gráfico ‘Escala temporal’ en la ventana ‘Canal1’ y en la ventana ‘Canal2’.

5.2.3. LA VENTANA SINTETIZADOR

Esta ventana es la que se encarga de habilitar o deshabilitar al resto de ventanas. También es la encargada de iniciar la sesión con el sintetizador y de elegir los modos de funcionamiento de las ventanas ‘Canal1’ y ‘Canal2’. En la figura 5.4 se muestra la imagen de la ventana ‘Sintetizador’:

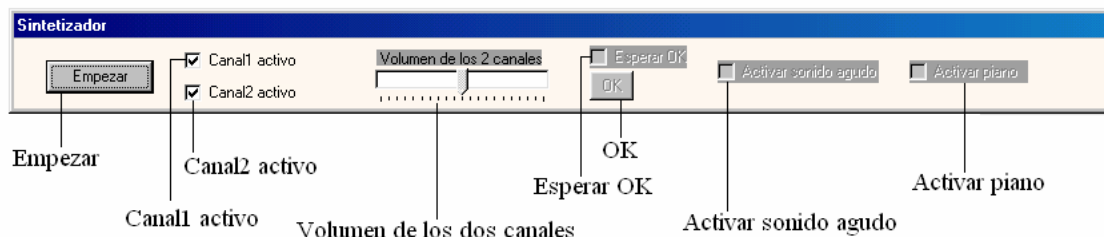


Figura 5.4. Imagen de la ventana Sintetizador

El funcionamiento de los elementos gráficos de la figura 5.4 es el siguiente:

- ‘Empezar’: este botón es el que se encarga de iniciar la sesión. Mientras no se pulse, el resto de ventanas y el resto de elementos gráficos de la ventana ‘Sintetizador’ están deshabilitados. Cuando se pulsa, deja de ser visible y habilita a las ventanas ‘Canal1’ y ‘Canal2’ y a los objetos ‘Nota’, ‘Octava’, ‘Volumen’, ‘Escala temporal’ y ‘Experto’ de las ventanas ‘Canal1’ y ‘Canal2’. También habilita al resto de elementos gráficos de la ventana sintetizador. Al iniciar la sesión, las ventanas ‘Canal1’ y ‘Canal2’ se encuentran en el ‘primer modo de funcionamiento’.
- ‘Canal1 activo’: habilita o deshabilita a la ventana ‘Canal1’. Su estado inicial es activo. Cuando se desactiva, deshabilita a la ventana ‘Canal1’ y en el objeto ‘Escala temporal’ aparece el valor ‘desactivado’ (envía al EVM el comando ‘tiempo1’). Cuando está inactivo, el canal 1 del sintetizador no puede emitir ningún sonido, excepto cuando está activa la ventana ‘Piano’ con la opción de ‘Tiempo libre’. Cuando se activa, habilita a la ventana ‘Canal1’ y en el objeto ‘Escala temporal’ aparece el valor ‘redonda’ (vuelve a enviar al EVM el comando ‘tiempo1’), y el canal 1 puede generar sonido en cualquier modo de funcionamiento.

- ‘Canal2 activo’: habilita o deshabilita a la ventana ‘Canal2’. Funciona igual que ‘Canal1 activo’, pero refiriéndose a la ventana ‘Canal 2’. Envía el comando ‘tiempo2’ cuando se activa o se desactiva.
- ‘Volumen de los 2 canales’: modifica el volumen de los dos canales a la vez. Cuando se pulsa, da a los dos canales el mismo volumen. Envía al EVM el comando ‘volumentotal’.
- ‘Esperar OK’: decide el modo de funcionamiento de las ventanas ‘Canal1’ y ‘Canal2’. Cuando ‘Activar piano’ se activa, ‘Esperar OK’ se deshabilita. Cuando ‘Esperar OK’ se activa, las ventanas ‘Canal1’ y ‘Canal2’ adquieren el ‘segundo modo de funcionamiento’, teniendo que pulsar el botón ‘OK’ para generar un sonido, excepto si el sonido ya se está produciendo. Si el sonido se está produciendo, aunque se esté en el ‘segundo modo de funcionamiento’, se permite variar cualquier parámetro del sonido sin tener que pulsar el botón ‘OK’, para permitir al usuario modificar algún matiz del sonido de forma más rápida. Siempre que se modifique algún parámetro de un sonido, el sonido empieza de nuevo (sólo en el canal modificado) con los parámetros actualizados, excepto si se modifica el volumen. Cuando ‘Esperar OK’ se activa, se habilita el botón ‘OK’. Cuando ‘Esperar OK’ se desactiva, se deshabilita el botón ‘OK’ y las ventanas ‘Canal1’ y ‘Canal2’ vuelven al ‘primer modo de funcionamiento’.
- ‘OK’: este botón envía al EVM el valor de todos los parámetros del sonido de las ventanas ‘Canal1’ y ‘Canal2’, excepto del ‘Volumen’, y el valor de ‘Activar sonido agudo’. Cuando ‘OK’ se pulsa, suenan los dos canales, aunque sólo se hayan modificado parámetros de un canal. Para que sólo suene un canal, habrá que desactivar el otro. Envía los comandos de todos los parámetros de los canales y de ‘Activar sonido agudo’, excepto el comando del ‘Volumen’ de ambos canales, al EVM.
- ‘Activar sonido agudo’: permite que suene el efecto ‘sonido agudo’. Ese efecto tiene un volumen fijo y, si está activado, suena mientras se esté produciendo un sonido, ya que puede activarse en los tres modos de funcionamiento del sintetizador (los tres

modos de funcionamiento del sintetizador son los mismos modos que los de las ventanas ‘Canal1’ y ‘Canal2’, ya que estas ventanas funcionan siempre). Envía al EVM el comando ‘agudo’.

- ‘Activar piano’: cuando se activa, habilita a la ventana ‘Piano’ y deshabilita a ‘Esperar OK’, de la ventana ‘Sintetizador’, ‘Nota’ y ‘Octava’, de las ventanas ‘Canal1’ y ‘Canal2’, y se entra en el ‘tercer modo de funcionamiento’. Envía al EVM el comando ‘pianoactivo’.

Los comandos que los objetos de la ventana ‘Sintetizador’ envían al EVM, son los comandos que aparecen en la tabla 4.2 del apartado 4.3.2.

5.3. SOFTWARE DESARROLLADO PARA LA IMPLEMENTACIÓN DEL INTERFAZ

Todo el interfaz del sintetizador se ha programado con el entorno de desarrollo Borland C++ Builder 6.0. Es un lenguaje visual, que facilita mucho el uso de elementos gráficos, y es un lenguaje orientado a objetos.

Para explicar el software de este proyecto, se explicarán los métodos de cada elemento gráfico. No se explicarán todas las propiedades, ya que sirven para definir características de los elementos gráficos, y los elementos gráficos pueden verse en las figuras 5.1, 5.2, 5.3 y 5.4 que se han mostrado en este capítulo. También se explicarán las relaciones entre las distintas ventanas del interfaz.

5.3.1. COMUNICACIÓN CON EL EVM

Casi todos los objetos del interfaz envían algún comando al TMS320C30 EVM, por lo que se va a explicar en primer lugar la comunicación con el EVM. La clase padre del interfaz del sintetizador es la ventana ‘Canal1’. Esta ventana es la que implementa el método para escribir en el EVM y para leer del EVM. El resto de ventanas heredan este método y lo usan también. El método se llama ‘escribirdsp’ y aparece en el archivo de código ‘Unit1.cpp’, correspondiente a la ventana ‘Canal1’.

En la figura 5.5 se aprecia el diagrama de flujo de ‘escribirdsp’, que siempre realiza dos escrituras y dos lecturas:

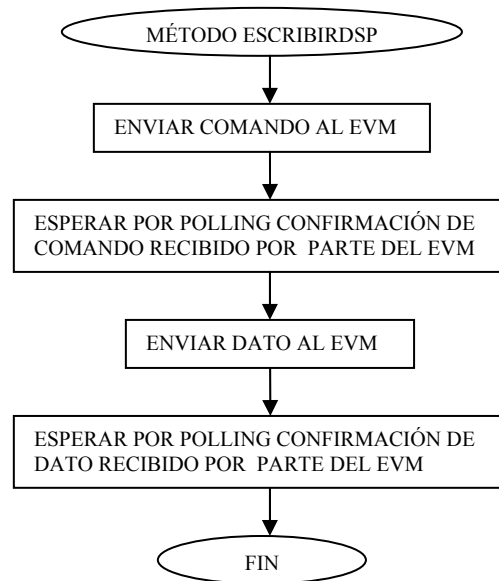


Figura 5. 5. Diagrama de flujo de la rutina ‘escribirdsp’

La comunicación entre el PC y el EVM que se aprecia en la figura 5.5, no sigue el protocolo descrito por ‘Texas Instruments’ en sus manuales, ya que este protocolo (que se implementó en un principio) no funcionaba con las funciones de la librería ‘io.h’ que se han usado para escribir y leer en los puertos del PC con Windows. El método ‘escribirdsp’ es más sencillo que el protocolo de ‘Texas Instruments’, ya que accede a menos registros.

Cuando el PC escribe un comando, escribe en el registro ‘COM_CMD’, provocando la interrupción ‘INT1’ del TMS320C30, y cuando escribe un dato, lo escribe en el registro ‘COM_DATA’, provocando la interrupción ‘INT2’ del TMS320C30. El PC sólo lee datos del EVM, lee los datos del registro ‘COM_DATA’, provocando la interrupción ‘INT2’ en el TMS320C30. El PC espera a leer un dato cuando escribe un comando, cuando el dato llega, comprueba que ese dato es el que espera y repite el proceso, pero esta vez tras escribir un dato. Cuando vuelve a leer un dato correcto, finaliza el método.

Los comandos que envía el interfaz son los que aparecen en la tabla 4.2. Todos los comandos realizan dos escrituras, una con la identificación del comando y otra con un dato.

Todos los datos y comandos que intervienen en la comunicación entre el PC y el EVM son números naturales. Para poder escribir y leer en los registros del PC se han usado dos subprogramas de la librería ‘io.h’.

Para escribir se ha usado el procedimiento ‘PortWordOut (int, int)’, al que se le pasan dos parámetros, el primero es la dirección del puerto en el que se va a escribir, y el segundo es el comando o dato a escribir.

Para leer se ha usado la función ‘PortWordIn (int)’, a la que se le pasa la dirección del puerto a leer y devuelve el dato del puerto.

5.3.2. RELACIÓN ENTRE LAS VENTANAS

Las ventanas del interfaz usan herencia, siendo la ventana ‘Canal1’ la clase ‘padre’. Cada ventana del interfaz del sintetizador se corresponde con una clase distinta. En la figura 5.6 se muestra qué ventanas son herederas de otra:

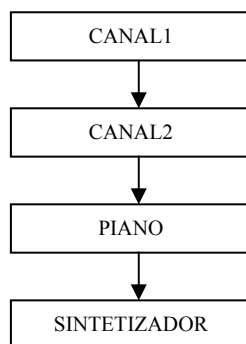


Figura 5.6. Herencia entre las ventanas del interfaz del sintetizador

Se ha decidido usar la herencia de esta manera por distintos motivos. La ventana ‘Canal2’ tiene un funcionamiento y una apariencia casi iguales que la ventana ‘Canal1’. Heredando ‘Canal2’ a partir de ‘Canal1’, en ‘Canal2’ sólo hay que realizar pequeñas modificaciones en algunas de sus propiedades y en la identificación de sus comandos, ahorrando mucho código.

Por otro lado, hay ventanas que han de modificar propiedades y acceder a métodos de los objetos de otras ventanas. La ventana ‘Piano’ modifica propiedades de los objetos

de las ventanas ‘Canal1’ y ‘Canal2’ y la ventana ‘Sintetizador’ modifica propiedades de los objetos de todas las demás ventanas. Estas modificaciones sólo se pueden hacer en ventanas ‘antecesoras’, por eso ‘Piano’ deriva de ‘Canal2’ (si derivara de ‘Canal1’ no podría modificar nada en ‘Canal2’) y ‘Sintetizador’ deriva de ‘Piano’. Las clases herederas pueden acceder a los métodos y propiedades de las clases ‘antecesoras’.

5.3.3. LOS MÉTODOS DE ‘CANAL1’ Y ‘CANAL2’

Los métodos son respuestas de un objeto ante un evento. Todos los objetos de las ventanas ‘Canal1’ y ‘Canal2’ responden ante el evento que se produce cuando se cambia el valor que muestra el objeto, aunque ese valor sea el que el objeto esté mostrando en el momento del cambio.

Los métodos que responden a un evento, de todos los objetos, excepto el objeto ‘Experto’ de la figura 5.2, de ambas ventanas, llaman a otros métodos, que están implementados por ‘Canal1’, y que se encargan de enviar los comandos y los datos al EVM. Por ejemplo, el objeto ‘Octava’, en la ventana ‘Canal1’ ejecuta el método ‘cambiaroctava1’ y en la ventana ‘Canal2’ ejecuta el método ‘cambiaroctava2’, a su vez, estos dos métodos llaman al método ‘cambiaroctava’, enviándole su identificación de comando y su nombre, siendo ‘cambiaroctava’ el método que realmente realiza las operaciones necesarias para enviar el comando al EVM. Si el método ‘cambiaroctava’ es llamado por ‘cambiaroctava1’ enviará al EVM el comando ‘octava1’ de la tabla 4.2, y si es llamado por ‘cambiaroctava2’ enviará el comando ‘octava2’. Al compartir las dos ventanas métodos para los objetos comunes, se ahorra código de programa.

Como los pares de objetos iguales de ‘Canal1’ y ‘Canal2’ llaman al mismo método, con saber lo que hace el método al que se llama, se sabe cómo responde ese par de objetos, perteneciente cada uno a una ventana distinta, cuando se produce el evento que ejecuta los métodos.

La ventana ‘Canal1’ se encarga de llamar a las funciones de la librería ‘io.h’.

En la tabla 5.2 se muestran los métodos asociados a los eventos de los objetos de las ventanas ‘Canal1’ y ‘Canal2’, junto al método que se encarga de realizar las operaciones para los objetos que son iguales en las dos ventanas y la etiqueta del objeto en la figura 5.2:

Etiqueta del objeto al que pertenece el método	Método de la ventana ‘Canal1’	Método de la ventana ‘Canal2’	Método común al que llaman
Nota	cambiarnota1	cambiarnota2	cambiarnota
Volumen	cambiarvolumen1	Cambiarvolumen2	cambiarvolumen
Octava	cambiaroctava1	cambiaroctava2	cambiaroctava
Escala temporal	cambiartiempo1	cambiartiempo2	cambiartiempo
Modulación	cambiarmodulacion1	cambiarmodulacion2	cambiarmodulacion
Señal	cambiarsenial1	cambiarsenial2	cambiarsenial
Desviación de frecuencia de la portadora FM	cambiardesfm1	cambiardesfm2	cambiardesfm
Factor multiplicador de la moduladora FM	cambiarfacmodfm1	Cambiarfacmodfm2	cambiarfacmodfm
Factor multiplicador de la moduladora AM	cambiarmodam1	cambiarmodam2	cambiarmodam
Componente continua de AM	cambiarcontam1	cambiarcontam2	cambiarcontam
Amplitud de la moduladora AM	cambiarmodam1	cambiarmodam2	cambiarmodam

Tabla 5.2. Métodos comunes de las ventanas ‘Canal1’ y ‘Canal2’

Los métodos de la cuarta columna de la tabla 5.2, reciben la identificación del comando y el nombre del objeto, sobre el que se produce el evento, de los métodos de la segunda y la tercera columna de la tabla 5.2.

Los métodos de la cuarta columna de la tabla 5.2, se ejecutan cuando se produce un evento en los objetos de ‘Canal1’ o ‘Canal2’ que aparecen en la tabla 5.2.

En la figura 5.7 aparece el diagrama de flujo general de los métodos que varían los parámetros de un sonido:

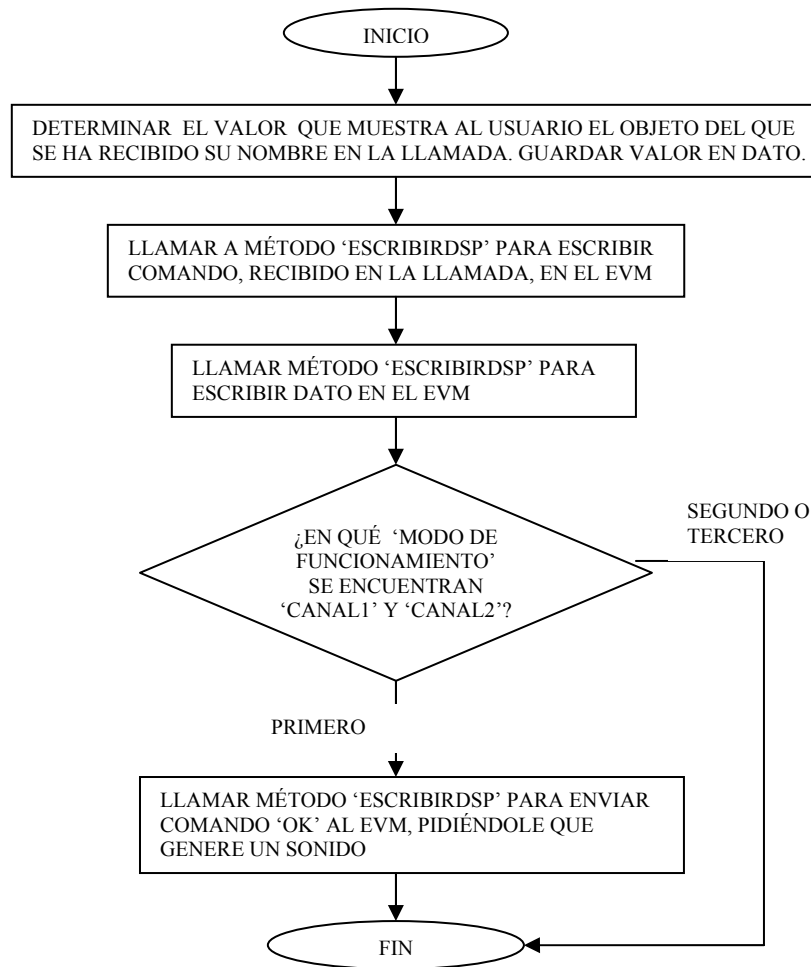


Figura 5.7. Diagrama de flujo general para los métodos de los parámetros que varían un sonido

El objeto 'Volumen' no envía el comando 'ok', de la tabla 4.2, ya que su variación nunca genera un sonido. Es la única variación que tiene con el resto de métodos de los demás parámetros que afectan al sonido.

El objeto 'Experto', que aparece en la figura 5.2, aunque tiene la misma función en 'Canal1' y 'Canal2', tiene un método propio en cada ventana, ya que en la ventana 'Canal1' llama a métodos de esa ventana y en 'Canal2' llama a métodos de la ventana 'Canal2'. En 'Canal1' y 'Canal2', el método de 'Experto' se llama 'cambiarnivel'.

En la figura 5.8 se muestra el diagrama de flujo del método ‘cambiarnivel’, diagrama válido para ‘Canal1’ y ‘Canal2’:

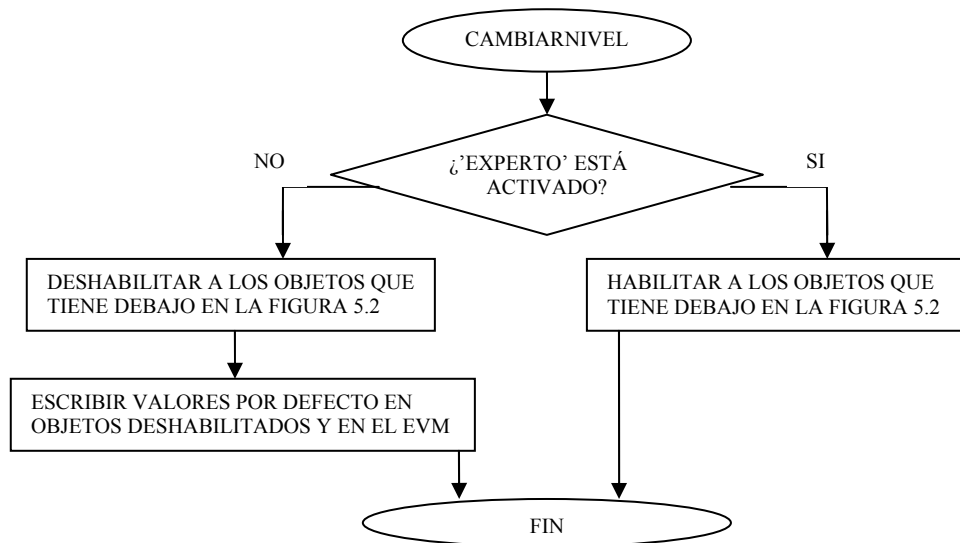


Figura 5.8. Diagrama de flujo del método ‘cambiarnivel’

5.3.4. LOS MÉTODOS DE LA VENTANA PIANO

Los elementos gráficos ‘OCTAVAS CANAL 1’ y ‘OCTAVAS CANAL 2’, que aparecen en la figura 5.3, llaman al método ‘cambiaroctavapiano’ cada vez que se elige una octava en sus respectivos canales. El método ‘cambiaroctavapiano’ escribe en el EVM para modificar la octava del canal elegido. En la figura 5.9 se muestra el diagrama de flujo del método ‘cambiaroctavapiano’:

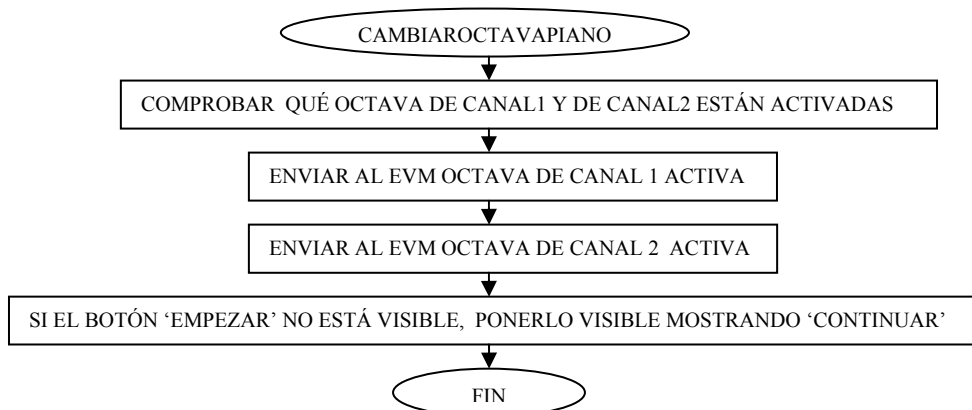


Figura 5.9. Diagrama de flujo del método ‘cambiaroctavapiano’

Los elementos gráficos ‘Tiempo definido’ y ‘Tiempo libre’, que aparecen en la figura 5.3, ejecutan el método ‘cambiatienpopiano’. En la figura 5.10 se muestra el diagrama de flujo del método ‘cambiatienpopiano’:

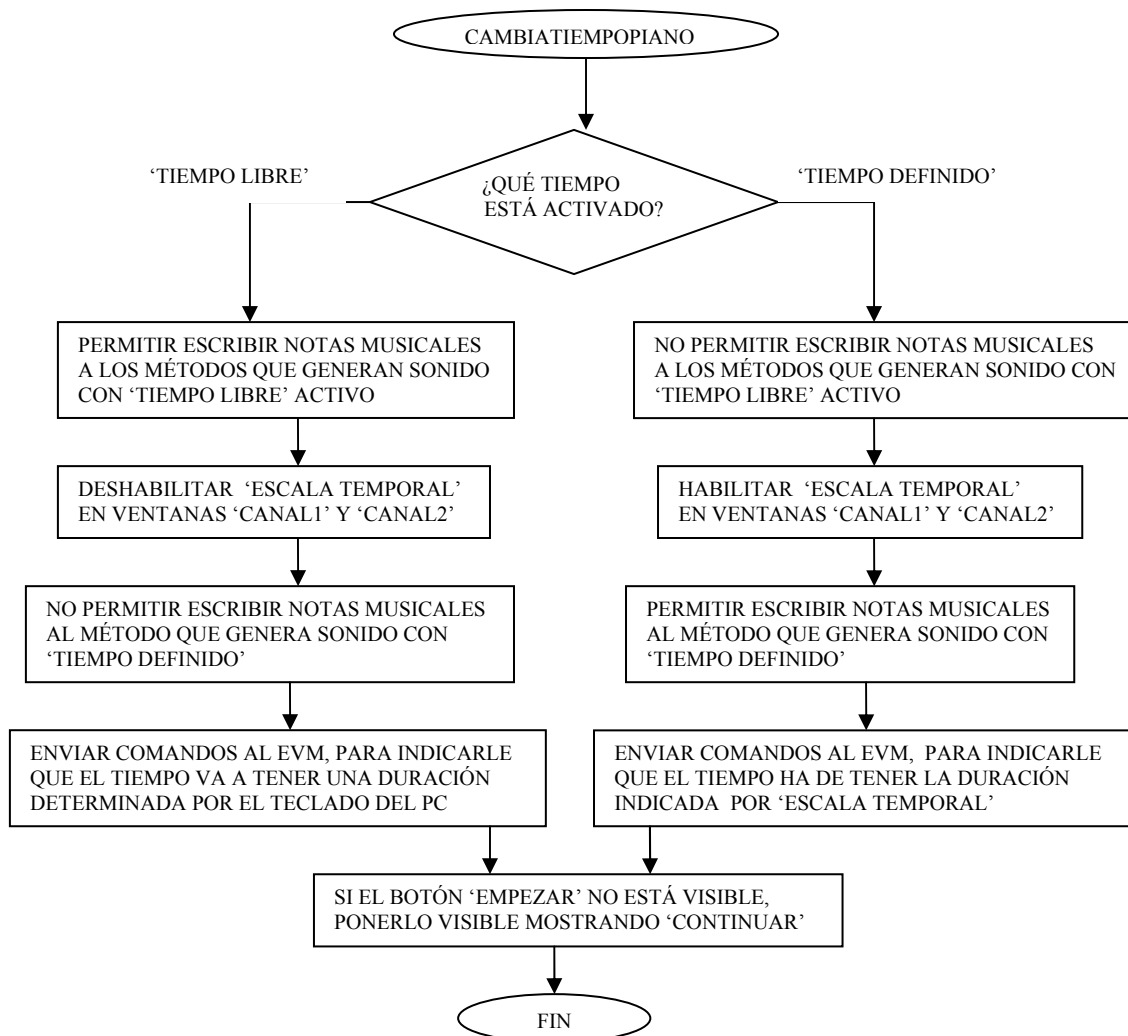


Figura 5.10. Diagrama de flujo del método ‘cambiatienpopiano’

Cuando ‘Piano’ tiene activado ‘Tiempo definido’ y se pulsa una tecla, se ejecuta el método ‘tiempopianodefinido’. Este método se ejecuta siempre, aunque esté activo ‘Tiempo libre’. El método ‘tiempopianodefinido’ recibe el carácter de la tecla pulsada, pudiendo decidir a qué canal pertenece la tecla pulsada, enviar el comando correcto y asignar un color a la tecla pulsada.

En la figura 5.11 se muestra el diagrama de flujo del método ‘tiempopianodefinido’:

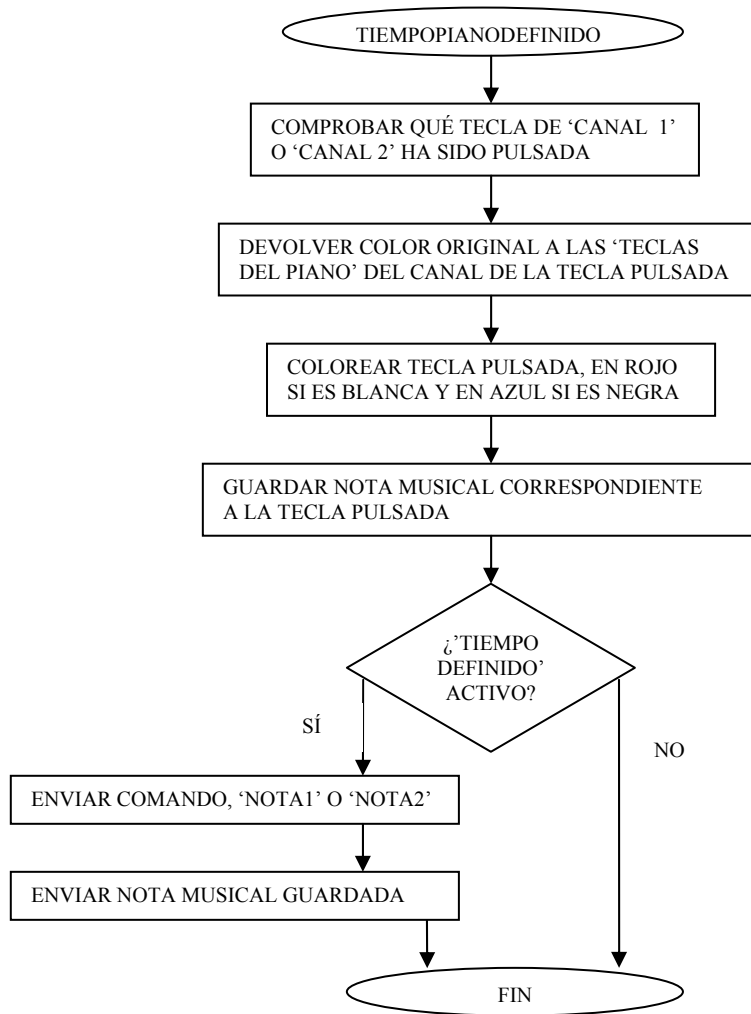


Figura 5.11. Diagrama de flujo del método ‘tiempopianodefinido’

Cuando la ventana ‘Piano’ tiene ‘Tiempo libre’ activo, se ejecutan dos métodos, uno para generar sonido (método ‘pultartecla’), y otro para controlar el tiempo de duración del sonido (método ‘soltartecla’).

Ambos métodos reciben el carácter de la tecla pulsada o que se deja de pulsar, con lo que, dependiendo de la letra, pueden seleccionar el canal, la nota musical y el color de cada tecla.

El método ‘pultartecla’ se ejecuta mientras una tecla está pulsada. Cuando ‘pultartecla’ se ha ejecutado más de una vez desde que se pulsó la tecla, no envía comandos al EVM (para que no esté continuamente enviando comandos al EVM y provocando interrupciones). Este método también se ejecuta cuando ‘Tiempo definido’ está activo.

En la figura 5.12 se muestra el diagrama de flujo de ‘pulsartecla’:

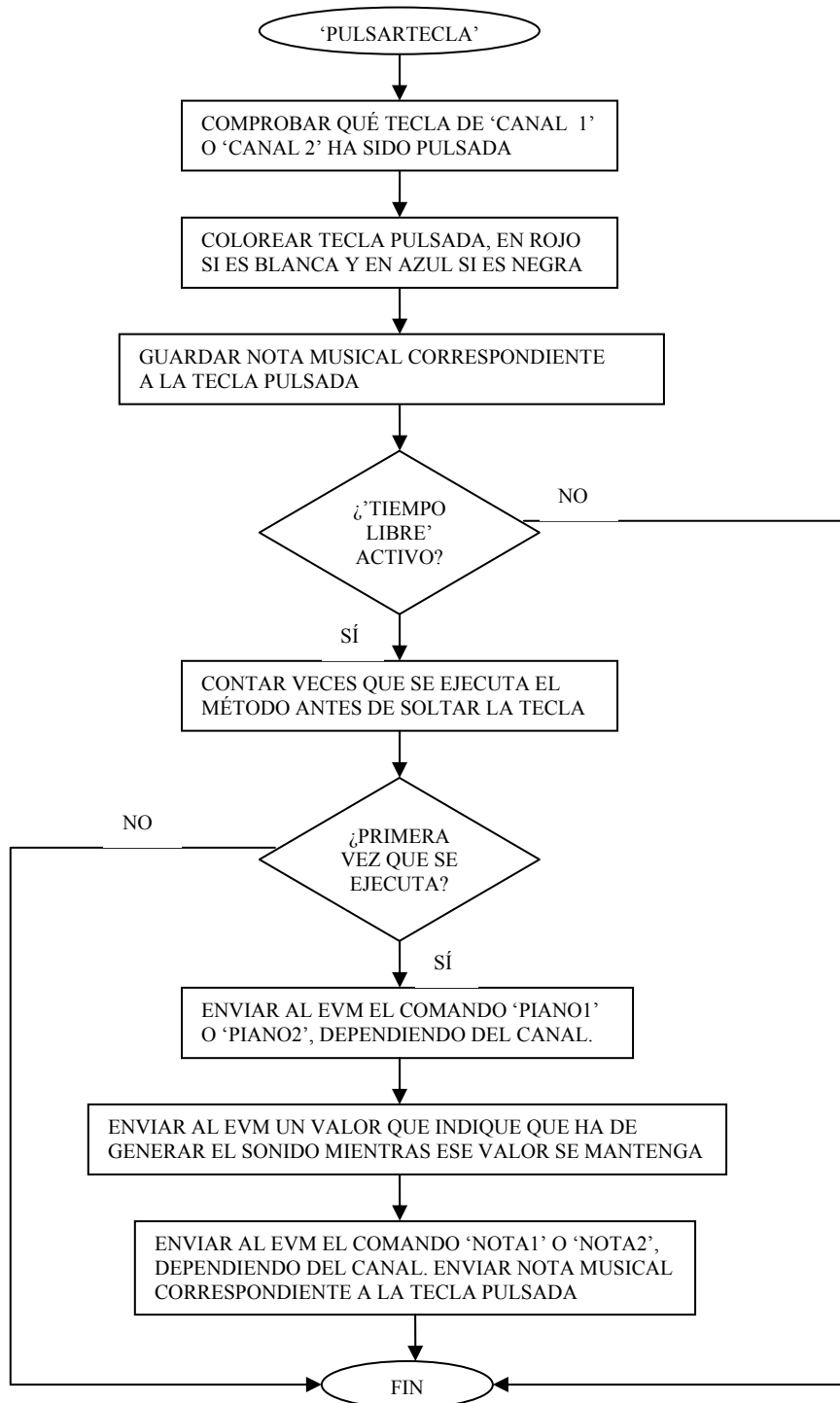


Figura 5.12. Diagrama de flujo del método ‘pulsartecla’

En la figura 5.13 se muestra el diagrama de flujo del método ‘soltartecla’:

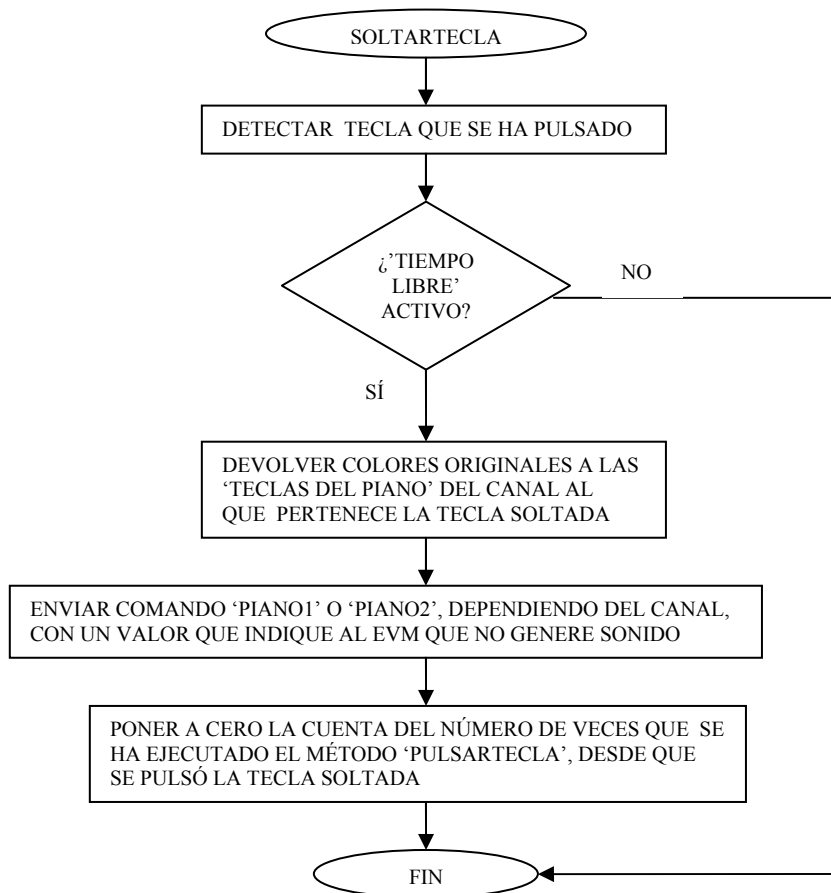


Figura 5.13. Diagrama de flujo del método ‘soltartecla’

5.3.5. LOS MÉTODOS DE LA VENTANA SINTETIZADOR

El botón ‘Empezar’ de la ventana ‘Sintetizador’, que aparece en la figura 5.4, ejecuta el método ‘iniciar’. El diagrama de flujo de ‘iniciar’ se ve en la figura 5.14:

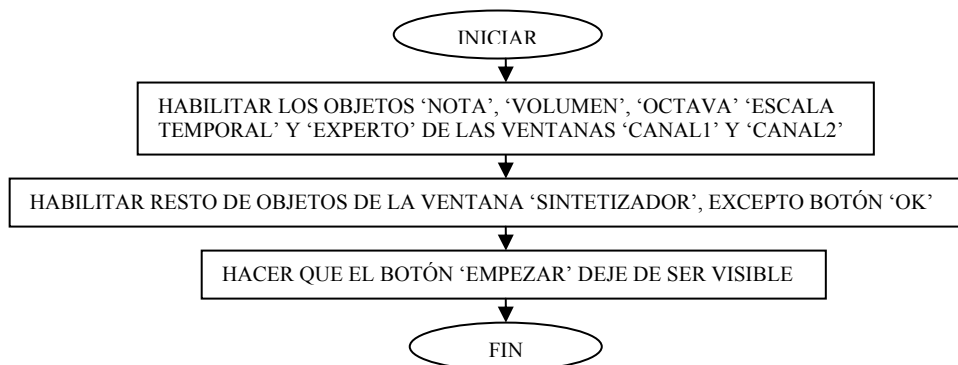


Figura 5.14. Diagrama de flujo del método empezar

Los objetos ‘Canal1 activo’ y ‘Canal2 activo’, de la figura 5.4, ejecutan los métodos ‘activacanal1’ y ‘activacanal2’ respectivamente. Sólo se va a explicar ‘activacanal1’, ya que ‘activacanal2’ realiza las mismas acciones. En la figura 5.15 se muestra el diagrama de flujo del método ‘activacanal1’:

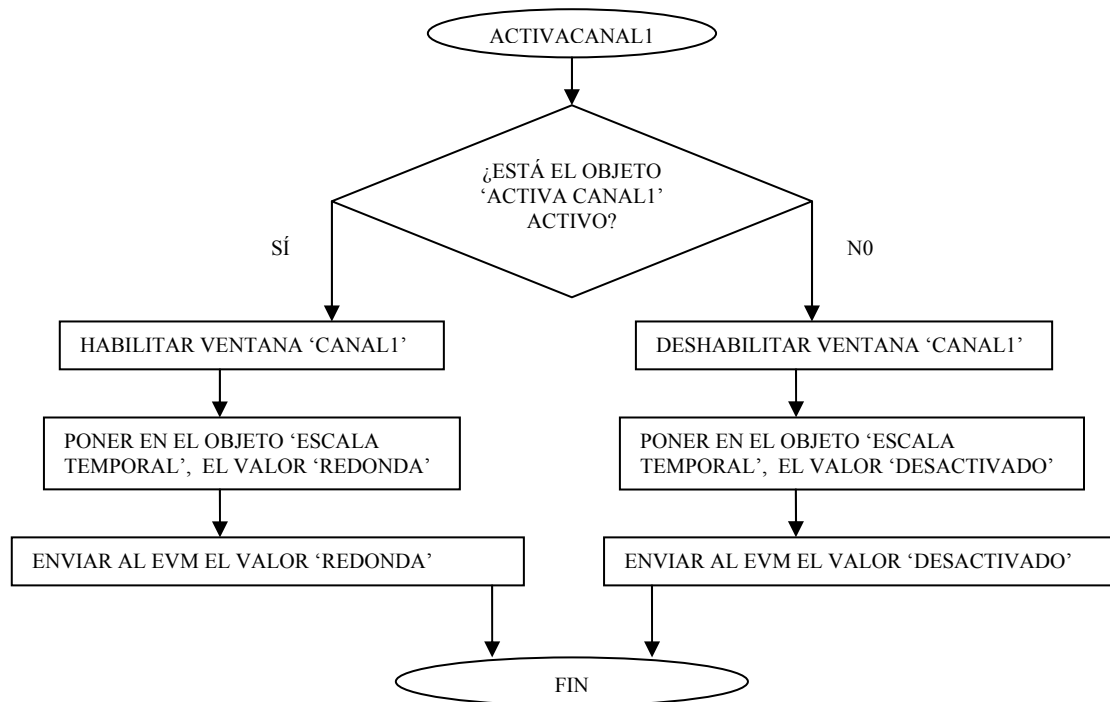


Figura 5.15. Diagrama de flujo del método ‘activacanal1’

El objeto ‘Volumen de los 2 canales’, que aparece en la figura 5.4, ejecuta el método ‘volumentotal’. Este método determina qué valor le ha dado el usuario al volumen, y envía el comando ‘volumentotal’, de la tabla 4.2, con el valor que el usuario le ha dado al volumen.

El objeto ‘Activar sonido agudo’, que aparece en la figura 5.4, ejecuta el método ‘activaragudo’. Este método envía al EVM el comando ‘agudo’, de la tabla 4.2, con un valor que indica si está activo o inactivo.

El objeto ‘Esperar OK’ de la figura 5.4, ejecuta el método ‘esperarkey’. En la figura 5.16 se ve el diagrama de flujo del método ‘esperarkey’:

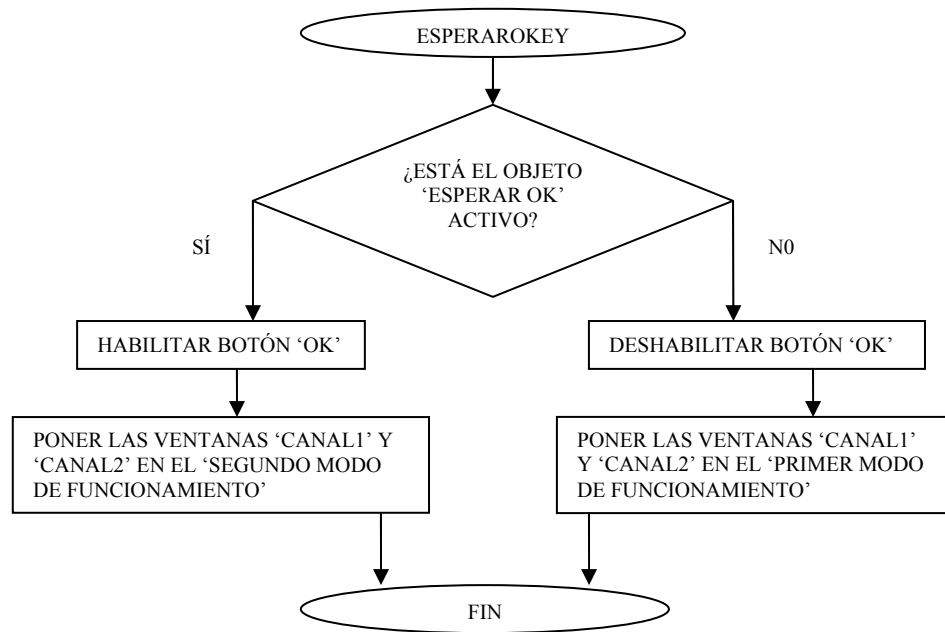


Figura 5.16. Diagrama de flujo del método ‘esperarkey’

El botón ‘OK’, que aparece en la figura 5.4, ejecuta el método ‘enviarkey’. En la figura 5.17 se muestra el diagrama de flujo del método ‘enviarkey’:

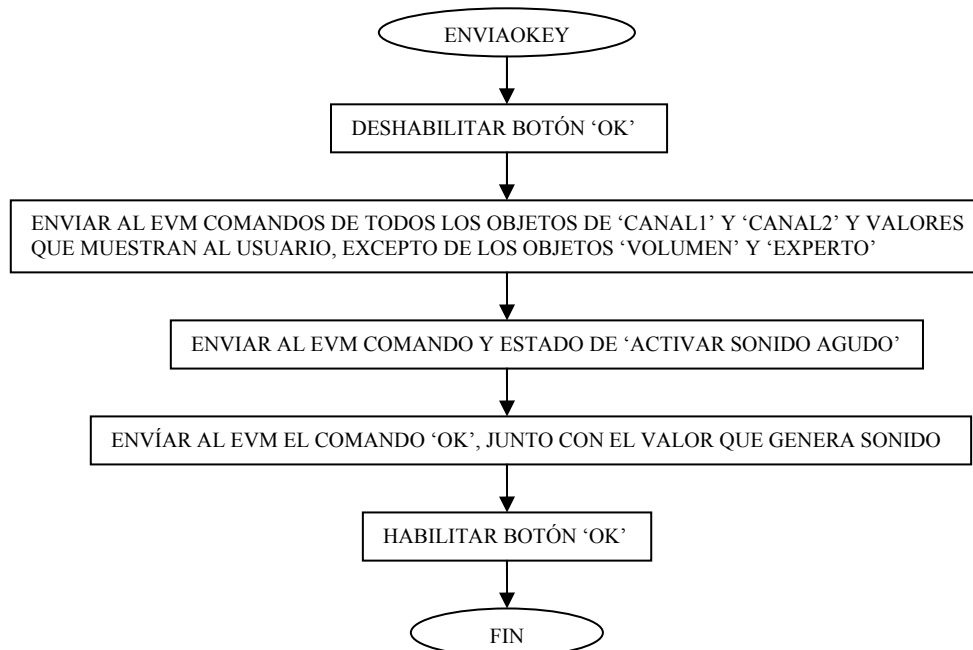


Figura 5.17. Diagrama de flujo del método ‘enviarkey’

El método 'enviaoke' deshabilita, al principio, el botón 'OK', para enviar al EVM todos los comandos antes de que se pueda volver a pulsar, y así evitar problemas de comunicación con el EVM. Cuando ya se han enviado todos los comandos, el botón 'OK' vuelve a ser habilitado.

El objeto 'Activar piano', que aparece en la figura 5.4, ejecuta el método 'activarpiano'. En la figura 5.18 se muestra el diagrama de flujo del método 'activarpiano':

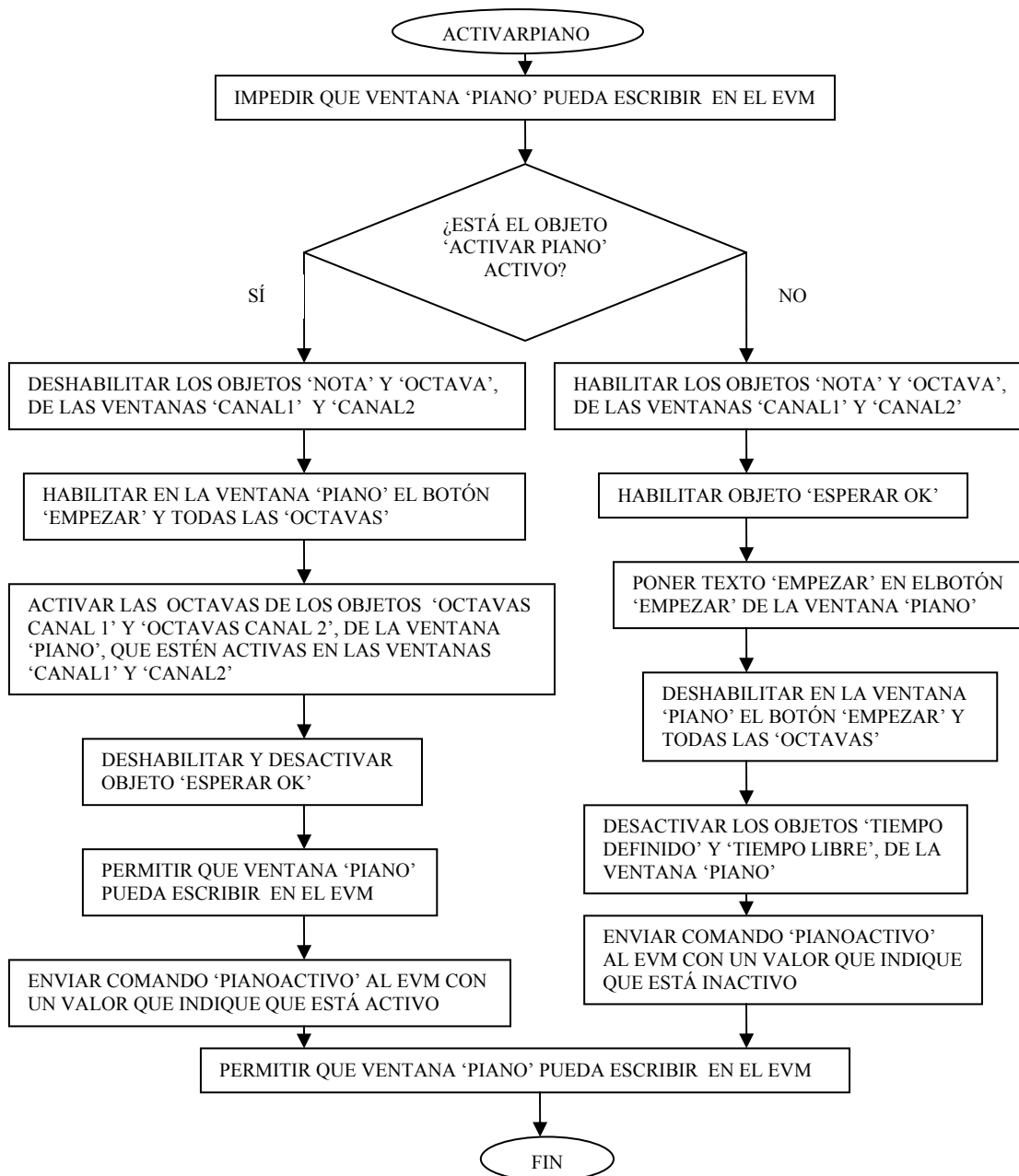


Figura 5.18. Diagrama de flujo del método 'activarpiano'

Al principio del método ‘activarpiano’, se impide que la ventana ‘Piano’ escriba en el EVM, y al final del método se permite que escriba en el EVM. Se hace, para impedir que cuando se desactive el objeto ‘Activar piano’, se genere algún sonido, ya que los objetos que modifica llaman a sus métodos y escriben en el EVM al desactivarse. Al final del método se vuelve a permitir que ‘Piano’ puede generar sonidos, incluso cuando ‘Activar piano’ está desactivado (esto no importa, ya que cuando ‘Activar piano’ se desactiva, deshabilita a ‘Piano’).

5.4. CONDICIONES PARA COMPILAR Y LINKAR EL INTERFAZ DEL SINTETIZADOR. PRUEBAS REALIZADAS

Las ventanas del interfaz del sintetizador y sus archivos asociados los engloba ‘Borland C++ Builder 6.0’ en un proyecto. El proyecto se llama ‘síntesis’ y crea un ejecutable ‘síntesis.exe’, que cuando se abre, muestra la figura 5.1. Para poder compilar y linkar el proyecto ‘síntesis.bpr’, hay que tener los archivos ‘io.h’ e ‘io.cpp’ en el directorio donde se encuentre el proyecto, para poder usar las funciones que permiten escribir y leer en el EVM.

Para comprobar el cumplimiento de los objetivos de este proyecto, desde el interfaz del sintetizador se han enviado todos sus comandos y valores al EVM, realizando todas las combinaciones de valores posibles, y se ha conectado la salida analógica del EVM a un osciloscopio y un altavoz.

Con el osciloscopio se ha comprobado que la señal de salida tenía la amplitud, frecuencia y forma deseadas, y con el altavoz que la señal producía un sonido con el timbre correcto.

CAPÍTULO 6: Conclusiones y líneas futuras

6.1. CONCLUSIONES

Para finalizar, se detallan los resultados obtenidos tras la realización del sintetizador y su interfaz:

- Se ha implementado un sintetizador musical con dos canales musicales que funcionan de forma independiente.
- Cada canal puede conseguir sonidos con las siguientes características:
 1. Generar las doce notas musicales de un pentagrama convencional.
 2. Generar cualquiera de las notas musicales entre la primera y la cuarta octava.
 3. Activar un efecto que produce un sonido agudo.
 4. Controlar el volumen de cada sonido.
 5. Generar una nota musical con un tiempo de duración de los que aparece en un pentagrama convencional.
 6. Generar cinco señales diferentes: coseno, triangular, diente de sierra, cuadrada y tren de pulsos cuadrados.
 7. Capacidad para oír cualquier señal con modulación AM, modulación FM y sin modular.
- El sintetizador consta de un interfaz gráfico para ser controlado por un usuario. El interfaz tiene las siguientes características:
 1. Capacidad para activar o desactivar cada canal.
 2. Control del efecto que produce un sonido agudo.
 3. Dos zonas diferenciadas para controlar cada canal.
 4. Elegir en cada canal una nota musical y su octava, volumen y tiempo de duración.
 5. Una opción avanzada que permite elegir: tipo de señal, tipo de modulación, desviación de la frecuencia de la portadora FM, factor multiplicador de la

frecuencia de la moduladora FM, factor multiplicador de la frecuencia de la moduladora AM, amplitud de la moduladora AM y componente continua AM.

6. Otorgar el mismo volumen a los dos canales simultáneamente.
7. Un emulador de un piano con el teclado del PC, otorgando a las teclas del PC las notas musicales de los dos canales.

El sintetizador cumple todos los objetivos iniciales, ofreciendo al usuario una diversidad de sonidos muy alta y un interfaz sencillo.

6.2. LÍNEAS FUTURAS

El sintetizador se ha implementado de forma modular, posibilitando la introducción de mejoras.

Hay varias posibilidades distintas que permitan mejorar las prestaciones actuales del sintetizador.

Una línea de trabajo es permitir elegir varios modos de trabajo, donde cada modo de trabajo implemente uno o varios tipos de síntesis. Un modo de trabajo serían los tipos de síntesis que usa este sintetizador, otros modos podrían ser:

- Implementar un tipo de síntesis más sencilla y que consuma menos tiempo de procesador, aunque tenga menos calidad de sonido, como la síntesis aditiva, que permitiese elegir la frecuencia de muestreo del TMS320C30 EVM más alta, para implementar más canales y octavas.
- Implementar un tipo de síntesis de modelado físico, como la guía de ondas, que permita emular sonidos de instrumentos reales. Para que el sonido tenga la máxima calidad, podría realizarse de forma que sólo pueda implementarse un canal.

Otra línea de trabajo sería usar un DSP más potente y usar las modulaciones AM y FM en cascada, aumentando el número de osciladores, y mejorando la calidad del sonido y aumentando el número de canales y de octavas.

Respecto al interfaz, una línea de trabajo sería permitir al usuario la introducción de partituras.

Otra línea, sería posibilitar al usuario grabar los sonidos que genere en un archivo y reproducirlos a modo de partitura, con un espacio temporal, entre los sonidos de cada canal, especificado por el usuario.

BIBLIOGRAFÍA

Libros

“Programación con C++ Builder 5”. Francisco Charte, Anaya Multimedia, 2000.

Documentación

TMS320C3x User’s Guide (referencia de TI: SPRU031E)

TMS320C3x/C4x Assembly Language Tools User’s Guide (referencia: SPRU036D)

Setting Up TMS320 DSP Interrupts in C (referencia: SPRA036)

TMS320C3x/C4x Optimizing C Compiler User’s Guide (referencia: SPRU034H)

TMS320C3x C Source Debugger User’s Guide (referencia: SPRU053)

TMS320C3x Peripheral Control Library User’s Guide (SPRU086)

Web

“EE6373 DSP Architectures”. Documento en formato HTML accesible por internet en la dirección:

<http://www.ee.unb.ca/tervo/ee6373/c2asm30.htm>

“IO.DLL”. Documento en formato HTML accesible por internet en la dirección:

<http://www.geekhideout.com/iodll.shtml>

“Math.h”. Documento en formato HTML accesible por internet en la dirección:

<http://www.opengroup.org/onlinepubs/007908799/xsh/math.h.html>

Artículos

“Síntesis musical por ordenador”. Avelino Herrera Morales. Sólo programadores, Revistas Profesionales, 1998